Уроки финтеха:

основы безопасности мобильных приложений

Авторы

Кирилл Зуев iOS Team Lead

Артем Логутов Information Security Expert

Давайте знакомиться!

ЦФТ входит в

ТОП-5

крупнейших разработчиков ПО
в России

* Рейтинги агентств «Эксперт РА», CNews Analytics

Мобильные приложения ЦФТ

Золотая Корона – **Денежные переводы**

Золотая Корона - **Обмен валюты**

Золотая Корона – **Погашение кредитов**

Мобильные приложения ЦФТ

Онлайн-банк
Faktura.ru

Платёжный кабинет
карты **«Билайн»**

Транспортная
карта

Платёжный кабинет
карты **«Кукуруза»**

Статистика

15 из 24

мобильных банков имеют хотя бы одну критическую уязвимость

https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/FIN-Vulnerabilities-rus.pdf

О чем поговорим

- Хранение данных
- Использование инструментов безопасности ОС
- Передача данных
- Бэкенд

Наши герои

Маша

Петя

Где мы сейчас?

- **Хранение данных**
- Использование инструментов безопасности ОС
- Передача данных
- Бэкенд

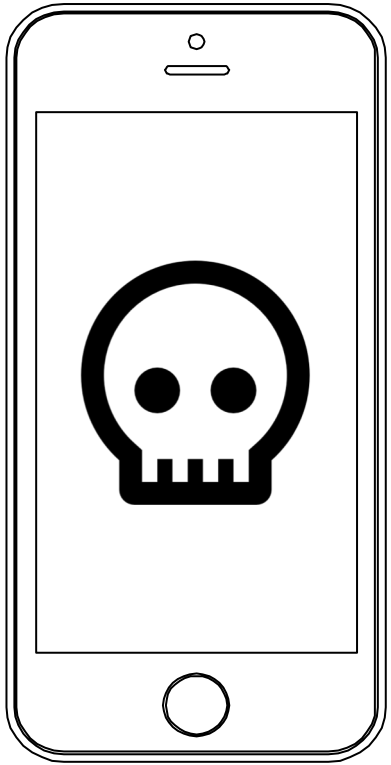
Хранение данных

Потеряли смартфон:

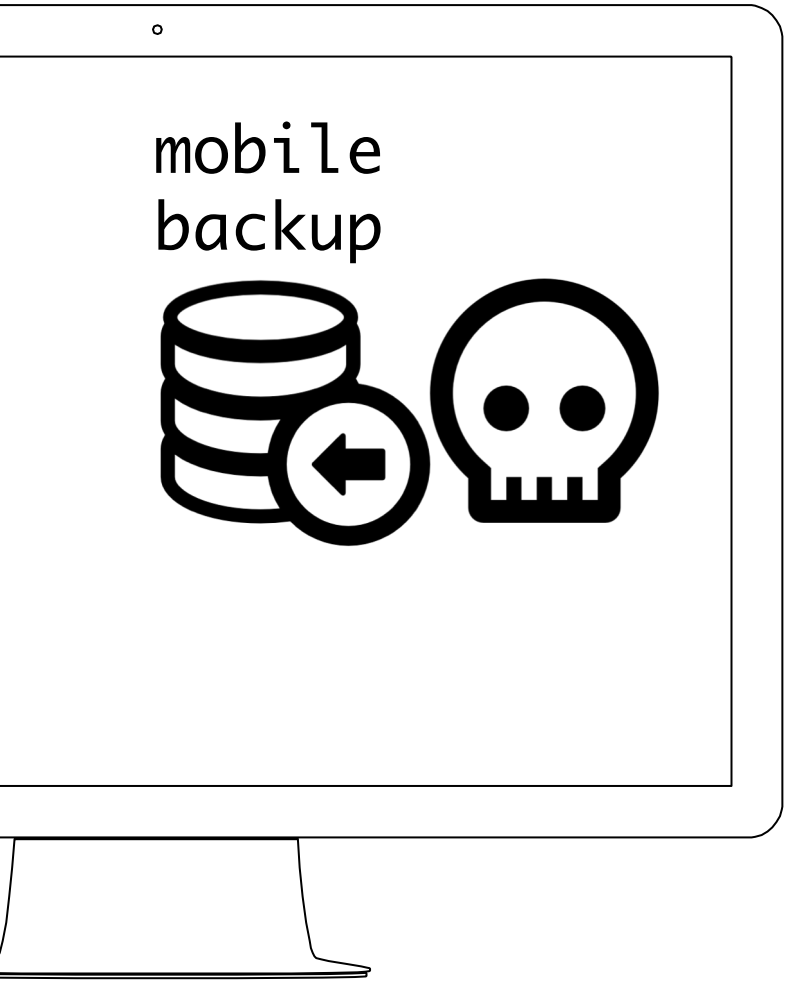
- функция «найти смартфон»
- заблокировать

Защита от вирусов:

- ???



секреты



секреты

```
~$ strings backup | grep МОЙ-ПАРОЛЬ
```

```
~$ strings backup | grep МОЙ-ПАРОЛЬ

<key>DEF_KEY_LOGIN_EMAIL</key>
<string>a.logutov@cft.ru</string>
<key>DEF_KEY_LOGIN_NETWORK</key>
<string>vendors.cft.ru</string>
<key>DEF_KEY_LOGIN_PASSWORD</key>
<string>МОЙ-ПАРОЛЬ</string>
```

Хранение данных

- делаем незашифрованный бэкап
- `strings backup | grep your_password`
- смотрим, каким приложениям доверять нельзя

Хранение данных

не забываем удалить

свой пароль

из `.bash_history`

☺

Как хранить данные в приложении?

- Пароли и прочие секреты
- Файлы
- Конфиги
- Большие объемы данных

Хранение паролей - только Keychain

- Надежно зашифрован
- Блокировать Keychain
  - `kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly`
- Можно шарить между приложениями

Отправляются в бэкап

- /Documents
- /Library/Preferences
  - Тут NSUserDefaults

```
var fileURL = URL(fileURLWithPath: "...")

var resourceValues = URLResourceValues()
resourceValues.isExcludedFromBackup = true

fileURL.setResourceValues(resourceValues)
```

```
var fileURL = URL(fileURLWithPath: "...")

var resourceValues = URLResourceValues()
resourceValues.isExcludedFromBackup = true

fileURL.setResourceValues(resourceValues)
```

```
var fileURL = URL(fileURLWithPath: "...")

var resourceValues = URLResourceValues()
resourceValues.isExcludedFromBackup = true

fileURL.setResourceValues(resourceValues)
```

Большие файлы

- `NSFileProtectionComplete`
  - Только активное приложение

Большие файлы

- `NSFileProtectionComplete`
  - Только активное приложение
- `NSFileProtectionCompleteUnlessOpen`
  - Может работать в фоне

```
fileURL.setResourceValue(
                    URLFileProtection.complete,
                    forKey: .fileProtectionKey)
```

```
fileURL.setResourceValue(
                    URLFileProtection.complete,
                    forKey: .fileProtectionKey)
```

```
fileURL.setResourceValue(
                    URLFileProtection.complete,
                    forKey: .fileProtectionKey)
```

Утечка через логи

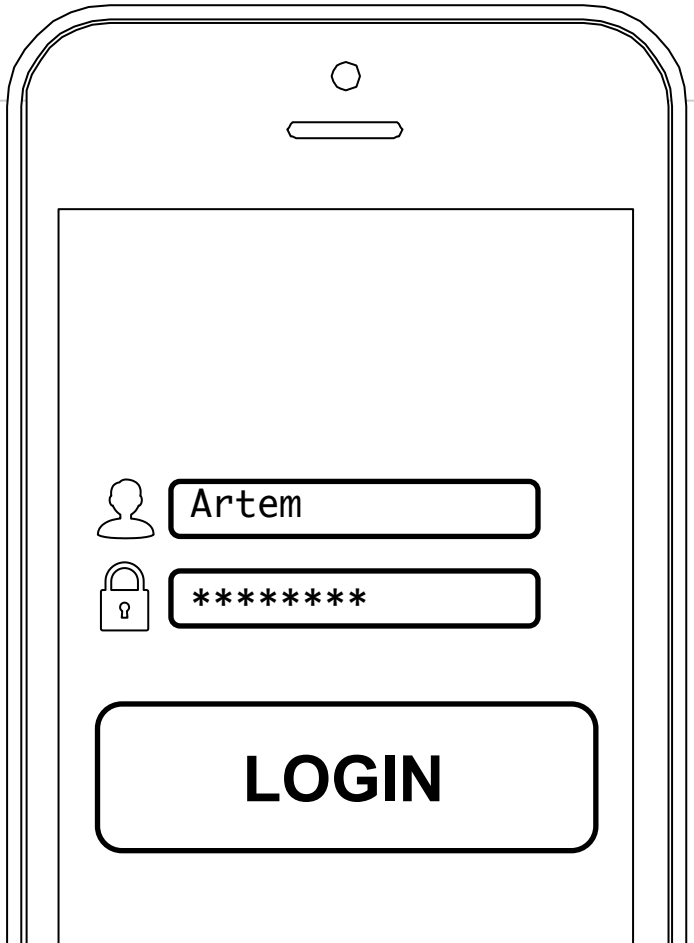
```
2018-04-10 21:16:32.263874+0700 Bank-DEV[63365:6377754]
Log Info:
<-- HTTP Request -->
URL: https://mybank.ru/api/authenticate
Method: POST
Headers:
{
  "Content-Type" = "application/json";
  "instance" = "32D386AB-60FF-4DF4-8077-0149A1C78738";
}
Body:
{"secret":"123456","type":"AUTO",”identity":"Maria"}
```

```
2018-04-10 21:16:32.263874+0700 Bank-DEV[63365:6377754]
Log Info:
<-- HTTP Request -->
URL: https://mybank.ru/api/authenticate
Method: POST
Headers:
{
  "Content-Type" = "application/json";
  "instance" = "32D386AB-60FF-4DF4-8077-0149A1C78738";
}
Body:
{"secret":"123456","type":"AUTO",”identity":"Maria"}
```

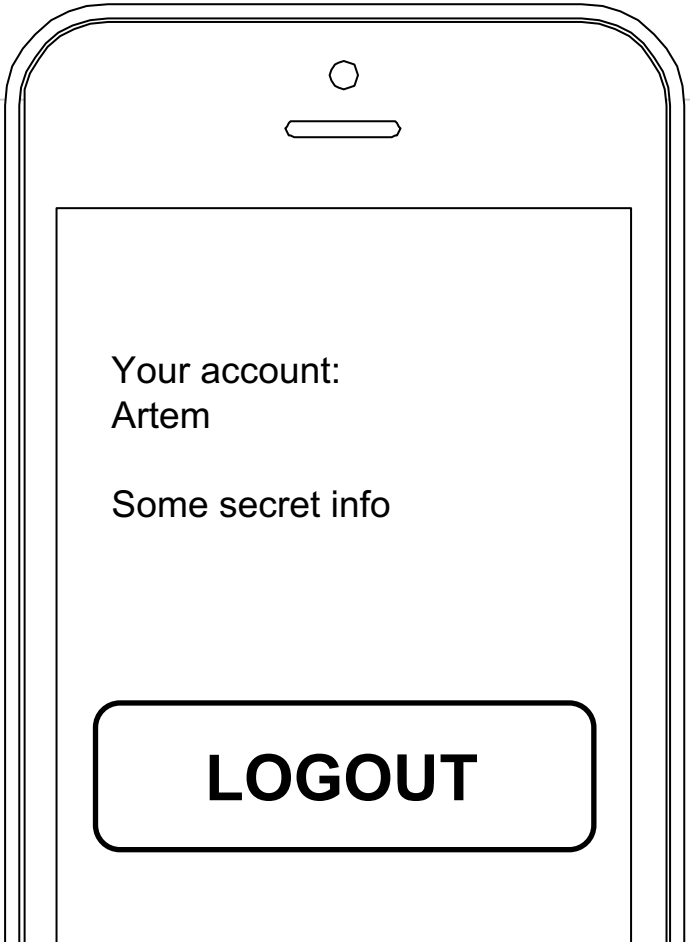
~~секреты~~

Проектируйте приложения так, чтобы
не хранить чувствительные данные

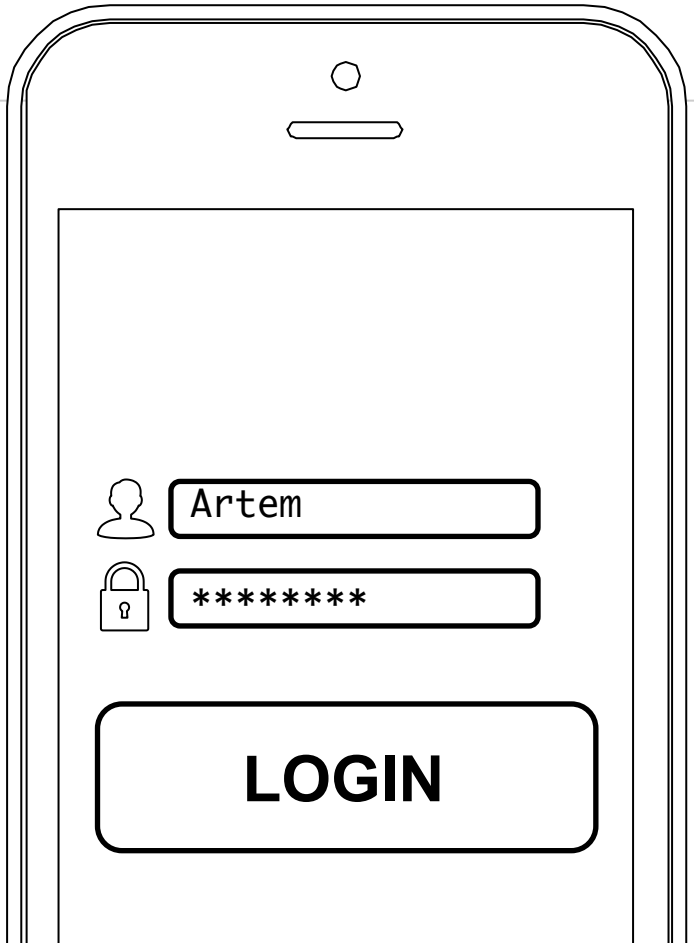
Хранение данных



Хранение данных



Хранение данных



Хранение данных

- Пароли, секретные токены – только в Keychain

Хранение данных

- Пароли, секретные токены – только в Keychain
- Тоже секреты, но большого размера
  - исключаем из бэкапа
  - используем настройку `NSFileProtectionComplete`

Хранение данных

- не записывать чувствительные данные в лог

Хранение данных

- не записывать чувствительные данные в лог
- автозаполнение и автокоррекция

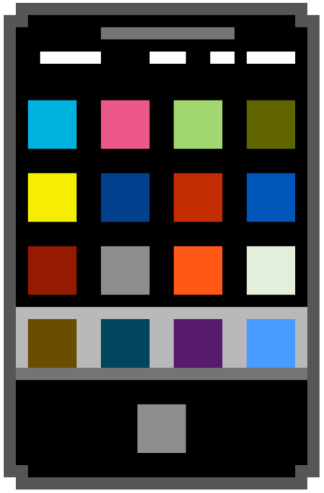
Хранение данных

- не записывать чувствительные данные в лог
- автозаполнение и автокоррекция
- не позволять копировать в буфер обмена секреты

Где мы сейчас?

- Хранение данных
- **Использование инструментов безопасности ОС**
- Передача данных
- Бэкенд

Обход аутентификации



OBJECTION
RUNTIME
MOBILE
EXPLORATION
GIT.IO/OBJECTION

https://github.com/sensepost/objection

Историческая справка

```
func loginWithBiometrics() {
    let context = LAContext()
    ...
    context.evaluatePolicy(..., ..., reply:
    { success, error in
        if success {
            let password = keychain.shortPassword
        }
    })
}
```

```
func loginWithBiometrics() {
    let context = LAContext()
    ...
    context.evaluatePolicy(..., ..., reply:
    { success, error in
        if success {
            let password = keychain.shortPassword
        }
    })
}
```

```
func loginWithBiometrics() {
    let context = LAContext()
    ...
    context.evaluatePolicy(..., ..., reply:
    { success, error in
        if success {
            let password = keychain.shortPassword
        }
    })
}
```

```
func loginWithBiometrics() {
    let context = LAContext()
    ...
    context.evaluatePolicy(..., ..., reply:
    { success, error in
        if success {
            let password = keychain.shortPassword
        }
    })
}
```

```
func loginWithBiometrics() {
    let context = LAContext()
    ...
    context.evaluatePolicy(..., ..., reply:
    { success, error in
        if success {
            let password = keychain.shortPassword
        }
    })
}
```

Можно сделать лучше

```
let accessControl = SecAccessControlCreateWithFlags(...)
let context = LAContext()
context.evaluateAccessControl(accessControl, operation:
.createItem, ...) { (success, error) in
    // Создаем объект с паролем
    // Сохраняем его в Keychain
    SecItemAdd(attributes as CFDictionary, nil)
}
```

```
let accessControl = SecAccessControlCreateWithFlags(...)
let context = LAContext()
context.evaluateAccessControl(accessControl, operation:
.createItem, ...) { (success, error) in
    // Создаем объект с паролем
    // Сохраняем его в Keychain
    SecItemAdd(attributes as CFDictionary, nil)
}
```

```
let accessControl = SecAccessControlCreateWithFlags(...)
let context = LAContext()
context.evaluateAccessControl(accessControl, operation:
.createItem, ...) { (success, error) in
    // Создаем объект с паролем
    // Сохраняем его в Keychain
    SecItemAdd(attributes as CFDictionary, nil)
}
```

```
let accessControl = SecAccessControlCreateWithFlags(...)
let context = LAContext()
context.evaluateAccessControl(accessControl, operation:
.createItem, ...) { (success, error) in
    // Создаем объект с паролем
    // Сохраняем его в Keychain
    SecItemAdd(attributes as CFDictionary, nil)
}
```

Обход аутентификации

Лучший код
это
дата моего
рождения!

Обход аутентификации

1111, 2222,
0000, 1234,
дата
рождения?

Обход аутентификации

К мобильному банку я сделаю код как пин-код к карте. Чтобы не забыть!

Обход аутентификации

Попробуем другой вариант...

```
let context = LAContext()
let domainState = context.evaluatedPolicyDomainState
```

Обход аутентификации

- Отслеживайте изменения в документации

Обход аутентификации

- Отслеживайте изменения в документации
- Поддерживайте механизмы безопасности в актуальном состоянии

Где мы сейчас?

- Хранение данных
- Использование инструментов безопасности ОС
- **Передача данных**
- Бэкенд

Передача данных

- HTTP
- HTTPS self-signed
- HTTPS misconfigured

Передача данных

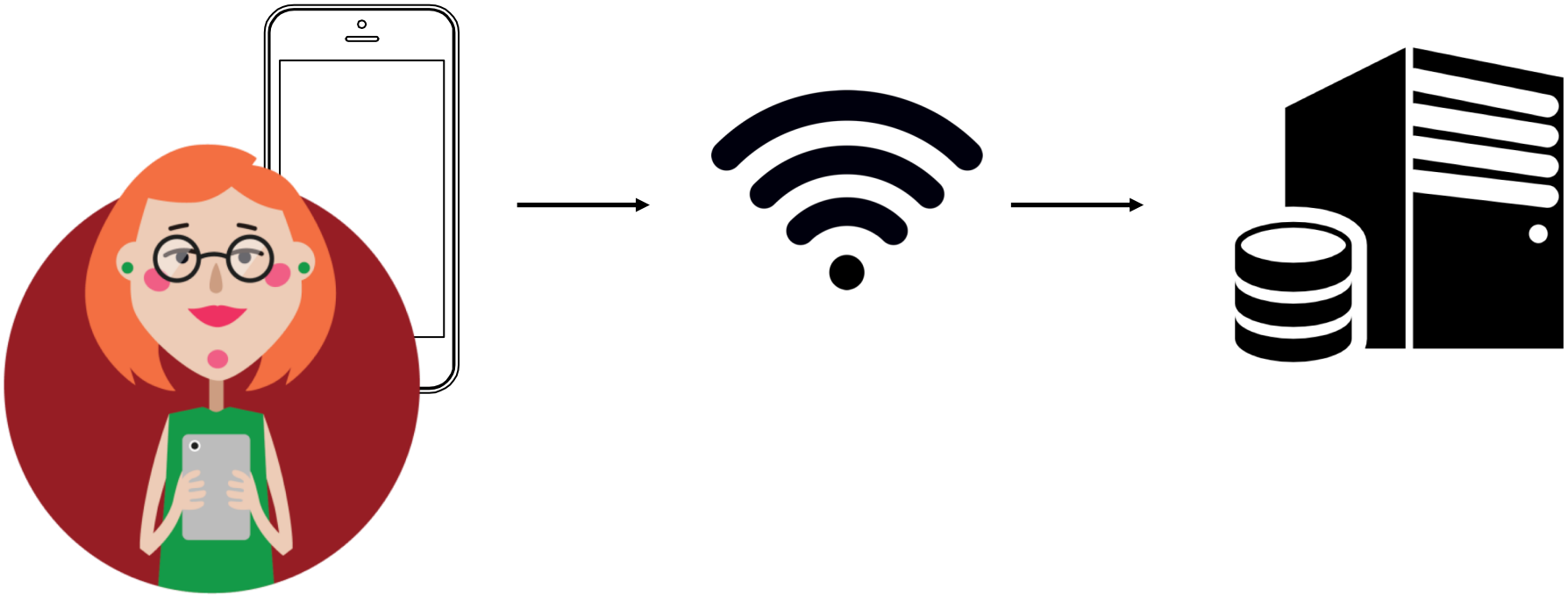
- ~~HTTP~~
- ~~HTTPS self-signed~~
- ~~HTTPS misconfigured~~

Дмитрий Медведев @MedvedevRussia · 4 мин

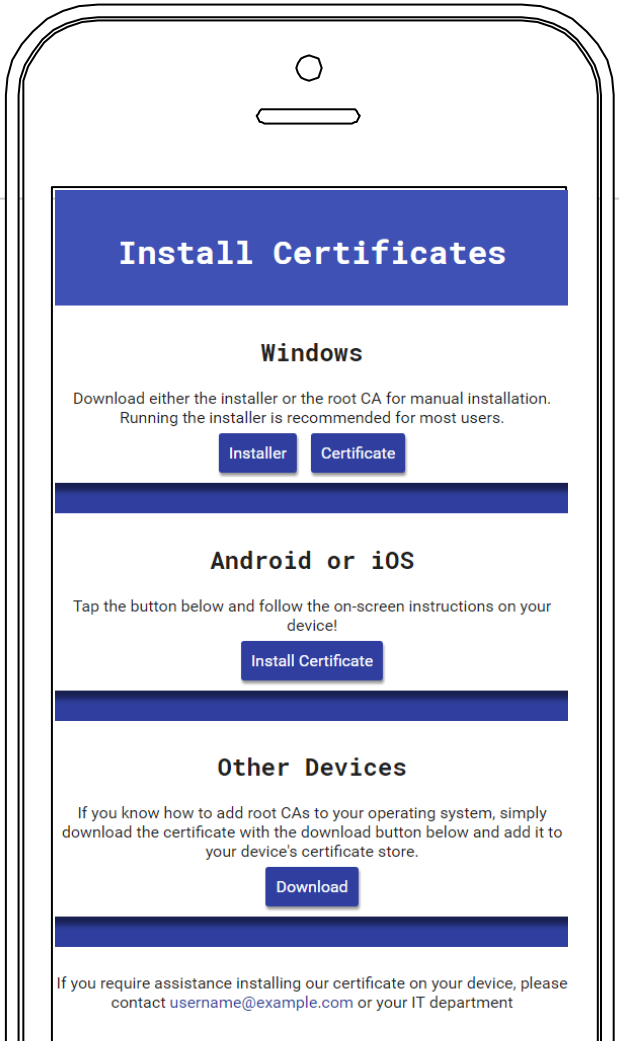
Стану свободным фотографом. Давно мечтал.

467 67

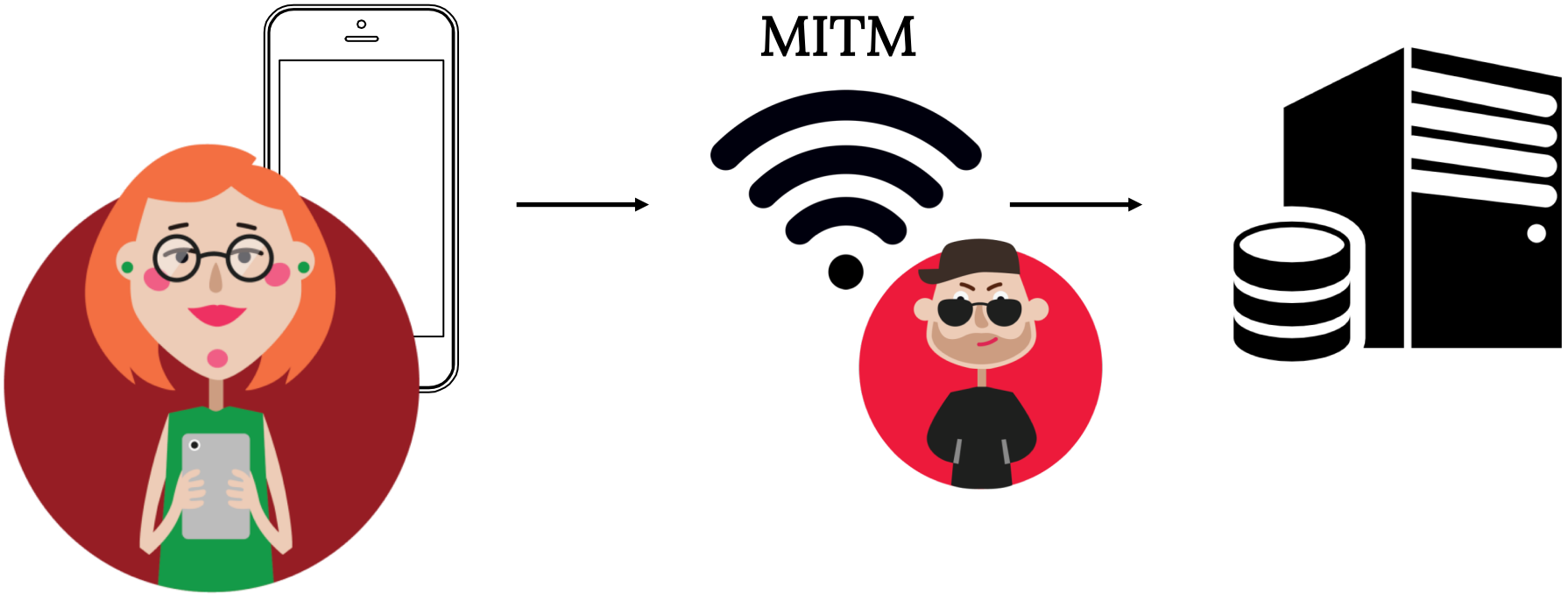
Передача данных



Передача данных

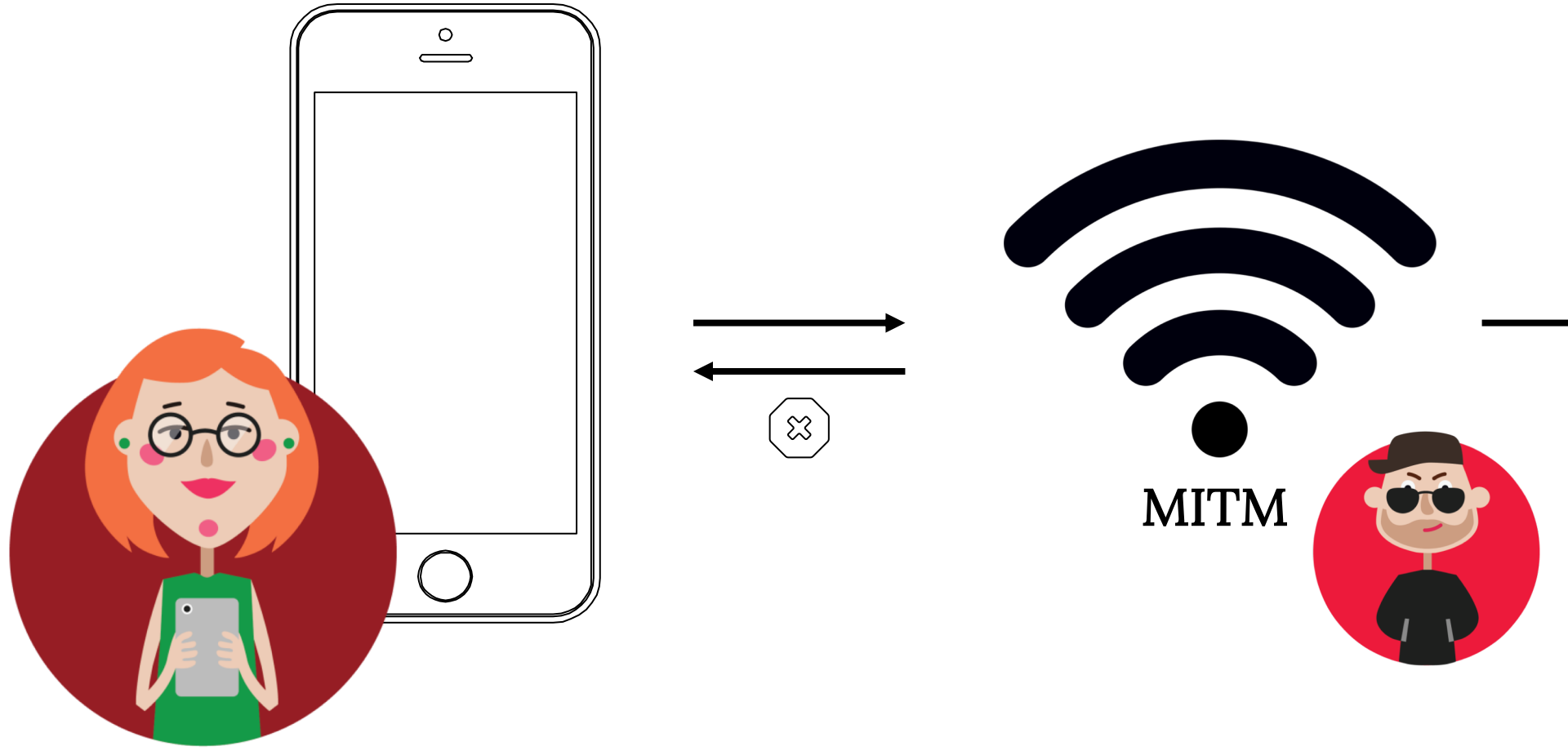


Передача данных



Передача данных

- BetterCap (https://www.bettercap.org)
- MANA Toolkit (https://github.com/sensepost/mana)
- WiFi-Pumpkin (https://github.com/P0cL4bs/WiFi-Pumpkin)


MITM

Сертификат, который
сохранен в приложении

==

Сертификат, который
пришел от сервера

Сертификат сайта

...

Сертификаты промежуточного центра сертификации

...

Сертификат центра сертификации

Сертификат сайта

Сертификаты промежуточного центра сертификации

Сертификат центра сертификации

Пиннинг сертификата сайта

- нарушение доступности сервера, если сертификат просрочен или отозван

Пиннинг сертификата сайта

- нарушение доступности сервера, если сертификат просрочен или отозван
- помогает только обновление всех мобильных приложений

Возможность выбора пользователем

- будет возможность работать приложению дальше, без обновления

Возможность выбора пользователем

- будет возможность работать приложению дальше, без обновления
- пользователь не всегда осознает риски
- при смене сертификата, будем вводить пользователя в заблуждение ошибкой

Проверка даты

- не нарушается доступность

Проверка даты

- не нарушается доступность
- спасает только от истечения сертификата
- обходится изменением даты на смартфоне

Сертификат сайта

Сертификаты промежуточного центра сертификации

Сертификат центра сертификации

Пиннинг корневого сертификата

- очень редко отзывают и большой срок действия
- защищает от перехвата трафика

Пиннинг корневого сертификата

- очень редко отзывают и большой срок действия
- защищает от перехвата трафика
- промежуточные центры сертификации могут выдать сертификат на ваш сайт для злодея

Пиннинг корневого сертификата

- очень редко отзывают и большой срок действия
- защищает от перехвата трафика
- промежуточные центры сертификации могут выдать сертификат на ваш сайт для злодея
- но это маловероятно ☺

Передача данных

- можем хранить отпечаток
- можем хранить сам сертификат
- не является секретом

```
func urlSession(_ session: URLSession,
                task: URLSessionTask,
                didReceive challenge:
                URLAuthenticationChallenge,
                completion: ...)
{
  if let trust = challenge.protectionSpace.serverTrust
  {
    if trustManager.verifyServerTrust(trust) {
        completion(.performDefaultHandling, nil);
        return
    }
  }
  completion(.cancelAuthenticationChallenge, nil)
}
```

```
func urlSession(_ session: URLSession,
                task: URLSessionTask,
                didReceive challenge:
                URLAuthenticationChallenge,
                completion: ...)
{
  if let trust = challenge.protectionSpace.serverTrust
  {
    if trustManager.verifyServerTrust(trust) {
        completion(.performDefaultHandling, nil);
        return
    }
  }
  completion(.cancelAuthenticationChallenge, nil)
}
```

```
func urlSession(_ session: URLSession,
                task: URLSessionTask,
                didReceive challenge:
                URLAuthenticationChallenge,
                completion: ...)
{
  if let trust = challenge.protectionSpace.serverTrust
  {
    if trustManager.verifyServerTrust(trust) {
        completion(.performDefaultHandling, nil);
        return
    }
  }
  completion(.cancelAuthenticationChallenge, nil)
}
```

```
func urlSession(_ session: URLSession,
                task: URLSessionTask,
                didReceive challenge:
                URLAuthenticationChallenge,
                completion: ...)
{
  if let trust = challenge.protectionSpace.serverTrust
  {
    if trustManager.verifyServerTrust(trust) {
        completion(.performDefaultHandling, nil);
        return
    }
  }
  completion(.cancelAuthenticationChallenge, nil)
}
```

```
func urlSession(_ session: URLSession,
                task: URLSessionTask,
                didReceive challenge:
                URLAuthenticationChallenge,
                completion: ...)
{
  if let trust = challenge.protectionSpace.serverTrust
  {
    if trustManager.verifyServerTrust(trust) {
        completion(.performDefaultHandling, nil);
        return
    }
  }
  completion(.cancelAuthenticationChallenge, nil)
}
```

```
func urlSession(_ session: URLSession,
                task: URLSessionTask,
                didReceive challenge:
                URLAuthenticationChallenge,
                completion: ...)
{
  if let trust = challenge.protectionSpace.serverTrust
  {
    if trustManager.verifyServerTrust(trust) {
        completion(.performDefaultHandling, nil);
        return
    }
  }
  completion(.cancelAuthenticationChallenge, nil)
}
```

```
static let rootHash = "fec9b8a5963f898cde203f6159c7a260"
func verifyServerTrust(_ trust: SecTrust) -> Bool
{
    let rootIdx = SecTrustGetCertificateCount(trust) - 1
    let rootCert =
        SecTrustGetCertificateAtIndex(trust, rootIdx)

    let data = SecCertificateCopyData(rootCert)
    let hashed = self.hash(data)

    return hashed == rootHash
}
```

```
static let rootHash = "fec9b8a5963f898cde203f6159c7a260"
func verifyServerTrust(_ trust: SecTrust) -> Bool
{
    let rootIdx = SecTrustGetCertificateCount(trust) - 1
    let rootCert =
        SecTrustGetCertificateAtIndex(trust, rootIdx)

    let data = SecCertificateCopyData(rootCert)
    let hashed = self.hash(data)

    return hashed == rootHash
}
```

```
static let rootHash = "fec9b8a5963f898cde203f6159c7a260"
func verifyServerTrust(_ trust: SecTrust) -> Bool
{
    let rootIdx = SecTrustGetCertificateCount(trust) - 1
    let rootCert =
        SecTrustGetCertificateAtIndex(trust, rootIdx)

    let data = SecCertificateCopyData(rootCert)
    let hashed = self.hash(data)

    return hashed == rootHash
}
```

```
static let rootHash = "fec9b8a5963f898cde203f6159c7a260"
func verifyServerTrust(_ trust: SecTrust) -> Bool
{
    let rootIdx = SecTrustGetCertificateCount(trust) - 1
    let rootCert =
        SecTrustGetCertificateAtIndex(trust, rootIdx)

    let data = SecCertificateCopyData(rootCert)
    let hashed = self.hash(data)

    return hashed == rootHash
}
```

```
static let rootHash = "fec9b8a5963f898cde203f6159c7a260"
func verifyServerTrust(_ trust: SecTrust) -> Bool
{
    let rootIdx = SecTrustGetCertificateCount(trust) - 1
    let rootCert =
        SecTrustGetCertificateAtIndex(trust, rootIdx)

    let data = SecCertificateCopyData(rootCert)
    let hashed = self.hash(data)

    return hashed == rootHash
}
```

```
static let rootHash = "fec9b8a5963f898cde203f6159c7a260"
func verifyServerTrust(_ trust: SecTrust) -> Bool
{
    let rootIdx = SecTrustGetCertificateCount(trust) - 1
    let rootCert =
        SecTrustGetCertificateAtIndex(trust, rootIdx)

    let data = SecCertificateCopyData(rootCert)
    let hashed = self.hash(data)

    return hashed == rootHash
}
```

Alamofire

https://infinum.co/the-capsized-eight/ssl-pinning-revisited

Передача данных

- Пиннинг для web view

```
public func webView(_ webView: WKWebView,
                    didReceive challenge:
URLAuthenticationChallenge,
                    completionHandler: ...)
{
    // То же, что в случае с URLSession
}
```

SFSafariViewController

SFSafariViewController

- Настройки ATS не действуют

SFSafariViewController

- Настройки ATS не действуют
- Куки расшарены с Safari

SFSafariViewController

- Настройки ATS не действуют
- Куки расшарены с Safari
- Пиннинг реализовать нельзя

Передача данных

- Burp Suite
- ZAP Proxy
- mitmproxy

Пиннинг корневого сертификата - наш опыт

- Chrome перестает доверять Symantec
  https://security.googleblog.com/2017/09/chromes-plan-to-distrust-symantec.html
- Для более плавного обновления выпустили версию с проверкой по одному из двух сертификатов

```
static let newHash = "fec9b8a5963f898cde203f6159c7a260"
static let oldHash = "c1a1fd1228da476ab9c5f90ee8cda33d"

func verifyServerTrust(_ trust: SecTrust) -> Bool
{
    let rootIdx = SecTrustGetCertificateCount(trust) - 1
    let rootCert =
        SecTrustGetCertificateAtIndex(trust, rootIdx)

    let data = SecCertificateCopyData(rootCert)
    let hashed = self.hash(data)

    return hashed == oldHash || hashed == newHash
}
```

Передача данных

- только корректно настроенный HTTPS
  - https://github.com/rbsec/sslscan
  - https://www.ssllabs.com/ssltest/

Передача данных

- только корректно настроенный HTTPS
  - https://github.com/rbsec/sslscan
  - https://www.ssllabs.com/ssltest/
- пиннинг CA сертификата
  - для бэкенда
  - и для WebView

Передача данных

- только корректно настроенный HTTPS
  - https://github.com/rbsec/sslscan
  - https://www.ssllabs.com/ssltest/
- пиннинг CA сертификата
  - для бэкенда
  - и для WebView
- отсутствие пиннинга == средняя уязвимость

Где мы сейчас?

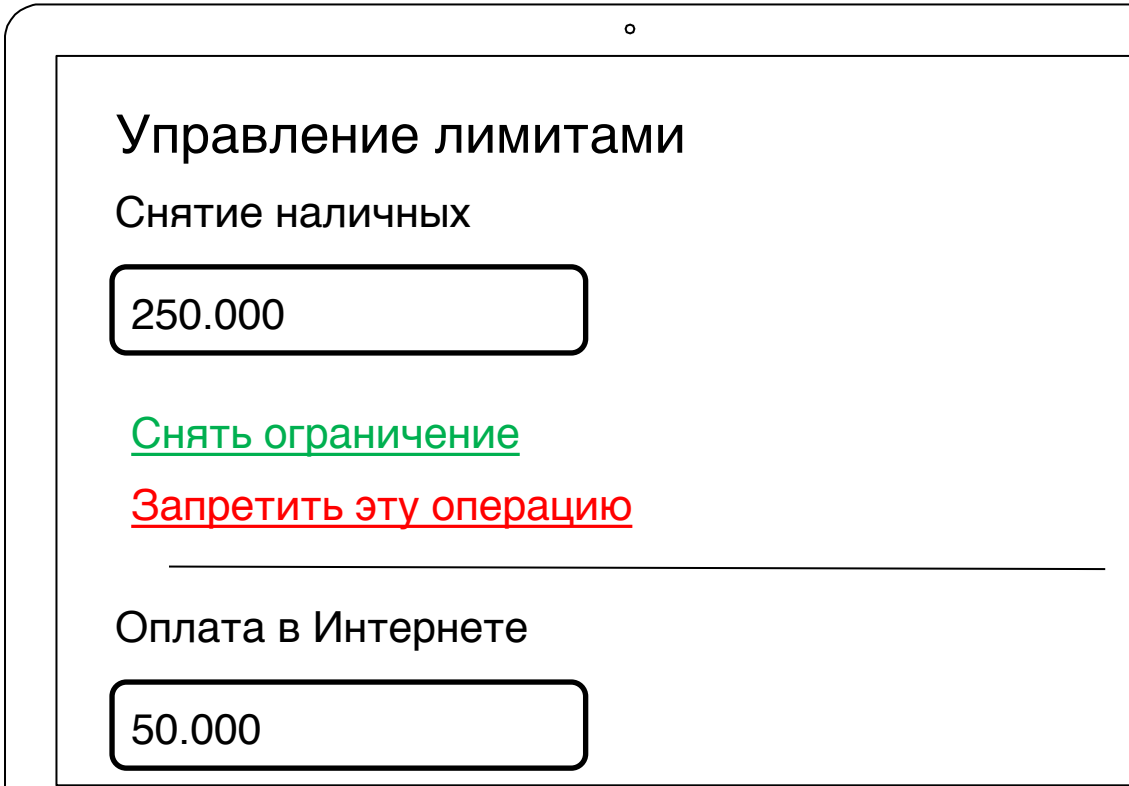
- Хранение данных
- Использование инструментов безопасности ОС
- Передача данных
- **Бэкенд**

Защищенность API

- Анализ защищенности ≈ тестирование, только для поиска уязвимостей
- Анализ защищенности фичи проводим на тестовом комплексе до деплоя на продакшн

Защищенность API

Тестировал функционал с установкой и снятием лимитов на операции



```
POST /api/limits/{id}/update

Cookie: auth=ArtemSession
{limits_at_JSON}
```

Защищенность API

user	cardID
Artem	123
Kirill	815
Maria	26

Request:
POST /api/limits/123/update

Cookie: auth=ArtemSession
{limits_at_JSON}

Response:
OK!

Request:
POST /api/limits/815/update

Cookie: auth=ArtemSession
{limits_at_JSON}

Response:
OK!

Request:
POST /api/limits/26/update

Cookie: auth=ArtemSession
{limits_at_JSON}

Response:
OK!

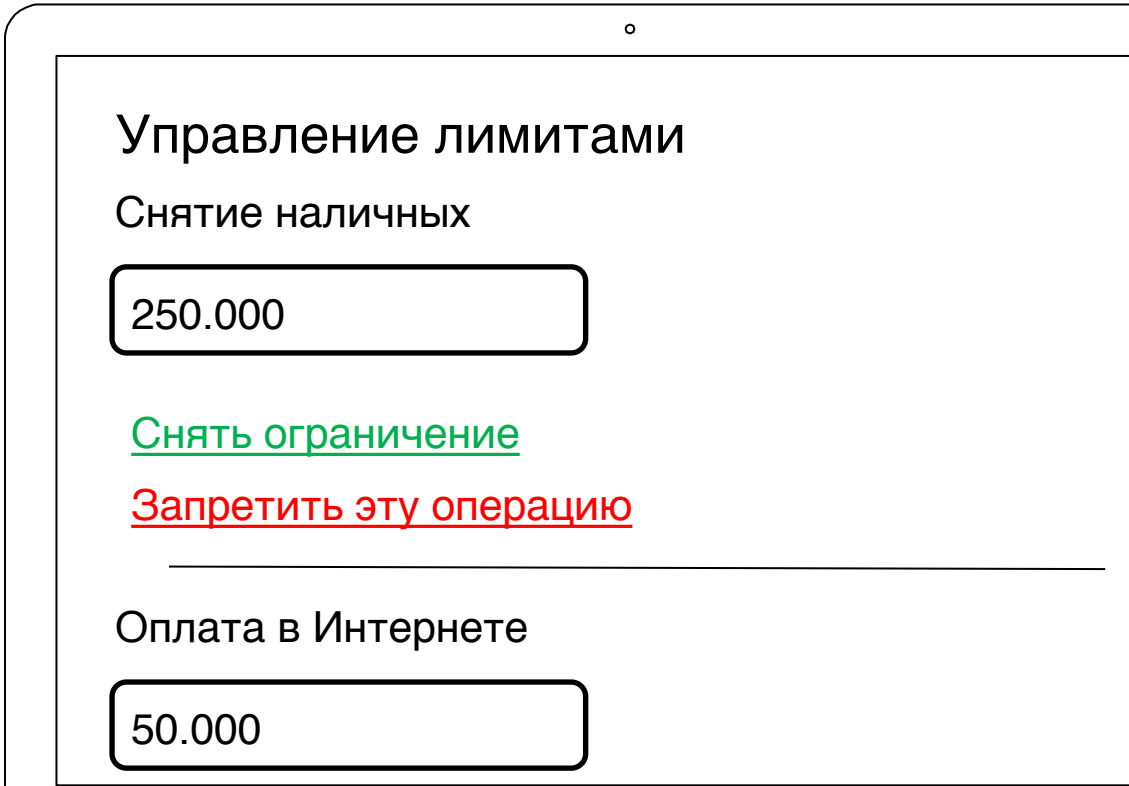
Защищенность API

Из-за некорректной авторизации

имелась возможность заблокировать операции на абсолютно **любой** карте!

Защищенность API

Чтобы **увеличить** лимиты необходимо ввести **код из SMS**



```
POST /api/otp/confirm
Cookie: auth=ArtemSession

{"otp":"{code}"}
```

Request:
POST /api/otp/confirm
Cookie: auth=ArtemSession

{"otp":"0000"}

Response:
FAIL

Request:
POST /api/otp/confirm
Cookie: auth=ArtemSession

{"otp":"0001"}

Response:
FAIL

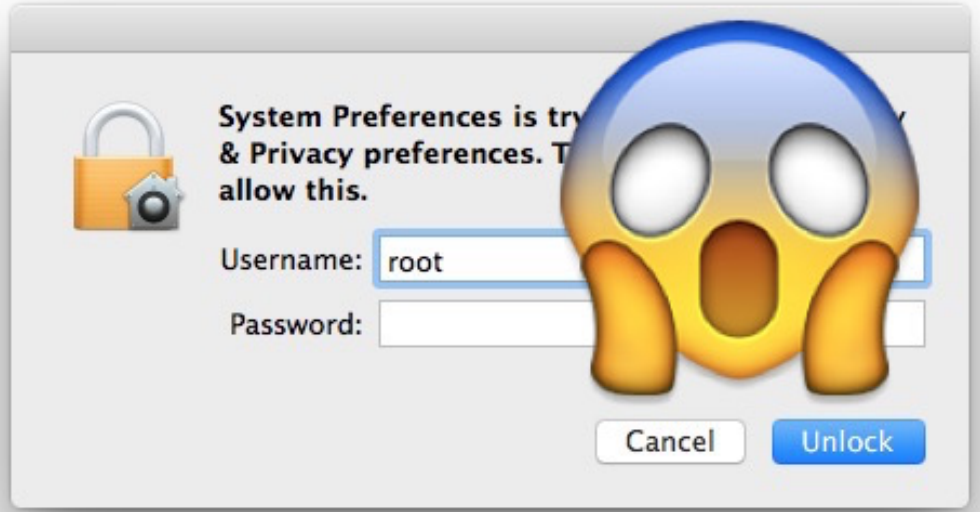
```
Request:
POST /api/otp/confirm
Cookie: auth=ArtemSession

{"otp":"0002"}

Response:
OK!
```

Защищенность API

user	code	response
Artem	0000	FAIL
Artem	0001	FAIL
Artem	0002	OK


System Preferences is try
& Privacy preferences. T
allow this.
Username:
root
Password:
Cancel
Unlock

Защищенность API

Из-за некорректной **авторизации** и **обхода аутентификации**

имелась возможность сбросить **любые лимиты**, на абсолютно **любой** карте!

Инструменты для тестирования

- Burp Suite
  - https://portswigger.net/burp
- wfuzz
  - https://github.com/xmendez/wfuzz

Защищенность API

- помните, что уязвимости сервера очень деструктивные
- внедряйте Secure Development Lifecycle
- составляйте тест-кейсы, ориентированные на проблемы безопасности
- проводите анализы защищенности

Заключение

Заключение

- Пароли – в Keychain

Заключение

- Пароли – в Keychain
- Обновляйте решения по безопасности

Заключение

- Пароли – в Keychain
- Обновляйте решения по безопасности
- Пиннинг сертификата

Заключение

- Пароли – в Keychain
- Обновляйте решения по безопасности
- Пиннинг сертификата
- Разработка и безопасность должны работать вместе

Заключение

OWASP Mobile:

- https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10
- https://github.com/OWASP/owasp-masvs
- https://github.com/OWASP/owasp-mstg

Кирилл Зуев

kirillzuev@me.com

Артем Логутов

Кирилл Зуев

kirillzuev@me.com

Артем Логутов

artemlogutov@gmail.com