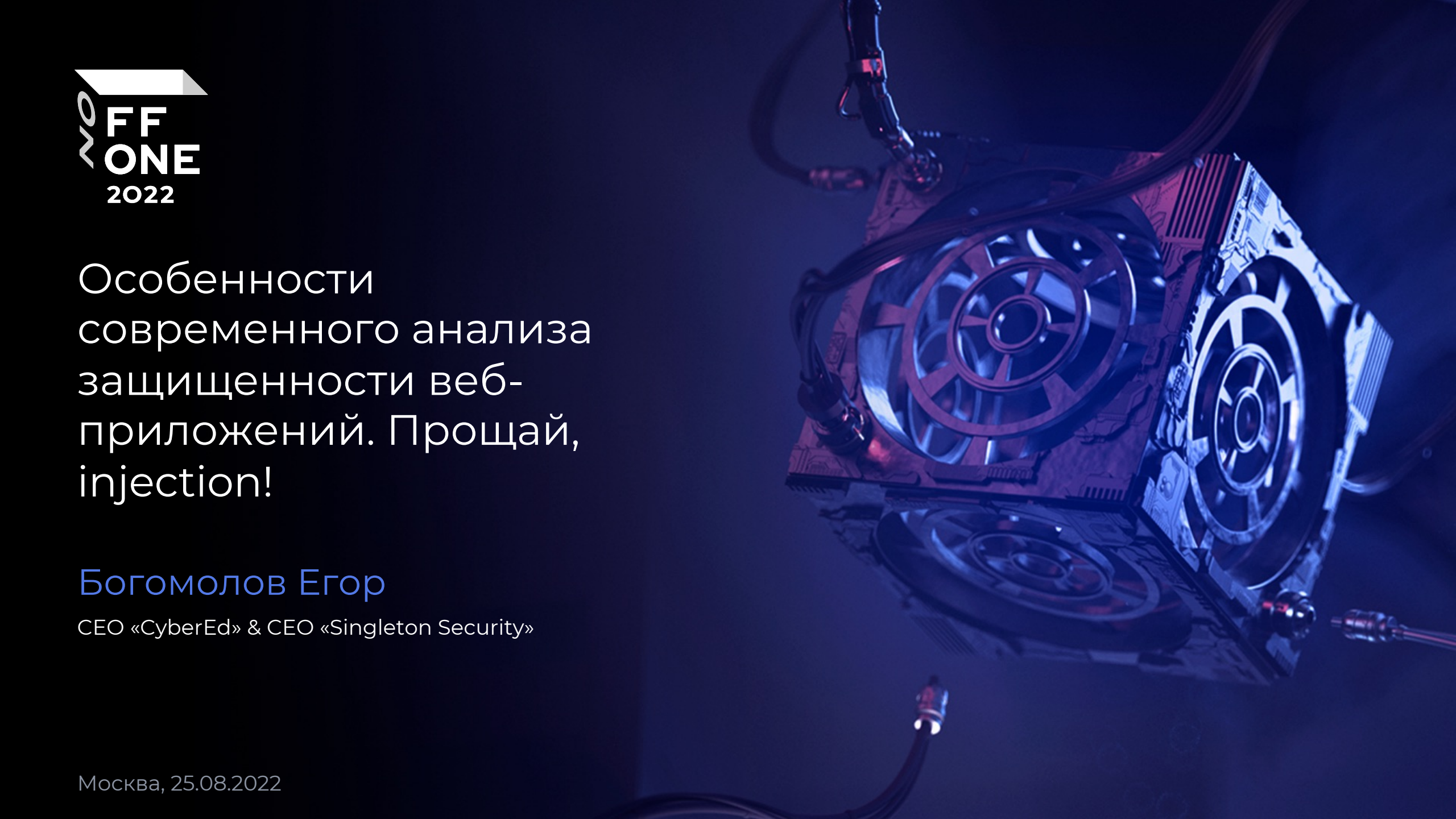


ON FF ONE 2022

Особенности современного анализа защищенности веб-приложений. Прощай, injection!

Богомолов Егор

CEO «CyberEd» & CEO «Singleton Security»

Москва, 25.08.2022

Обо мне

Богомолов Егор

- Управляю и развиваю в **CyberEd**
- Ищу баги вместе с **Singleton Security**
- С детства за **True0xA3**!

О докладе

Есть проблема и есть решение

Начнем издалека...

Почему возникают уязвимости? → Потому что человек ошибается.

Почему человек ошибается? → У него есть возможность допустить ошибку!

Как сделать так, чтобы уязвимости не возникали? → Не давать человеку возможности ошибиться!

О докладе

Есть проблема и есть решение

Современные фреймворки создают условия для разработчиков, в которых человек не имеет возможности сделать ошибку.

Плюсы:

- Пропадают классы уязвимостей, так как больше нельзя допустить приводящие к ним ошибки, если не захотеть это сделать специально: CSRF, XSS, SQLi, SSTI.
- Создано множество сложных механизмов: аутентификация, авторизация, управление сессией, хранение секретов, хранение паролей и прочие, где можно было бы наделать еще больше ошибок.

О докладе

Есть проблема и есть решение

Современные фреймворки создают условия для разработчиков, в которых человек не имеет возможности ошибиться.

Минусы:

- Такой подход нельзя масштабировать;
- Нельзя реализовать заранее все механизмы;
- Нельзя исключить все уязвимости;
- Использование готовых простых решений не приводит к развитию компетенций разработчика.

О докладе

Есть проблема и есть решение

Проблемы разработчиков решены!

Проблемы пентестеров - только начинаются! ;D

Где теперь нам искать баги?!

Содержание

Мы рассмотрим

- Какие проблемы решают современные фреймворки?
- Какие механизмы заранее реализованы в фреймворках?
- Какие уязвимости **не искать** в современных веб-приложениях?
- Какие уязвимости следует **искать** в современных веб-приложениях?
- Примеры уязвимостей, которые встречаются нам в современных веб-приложениях?
- Где проще всего найти такие проблемы?
- Как быть успешнее в поисках подобных проблем и, соответственно, в их исключении?

Предупреждение

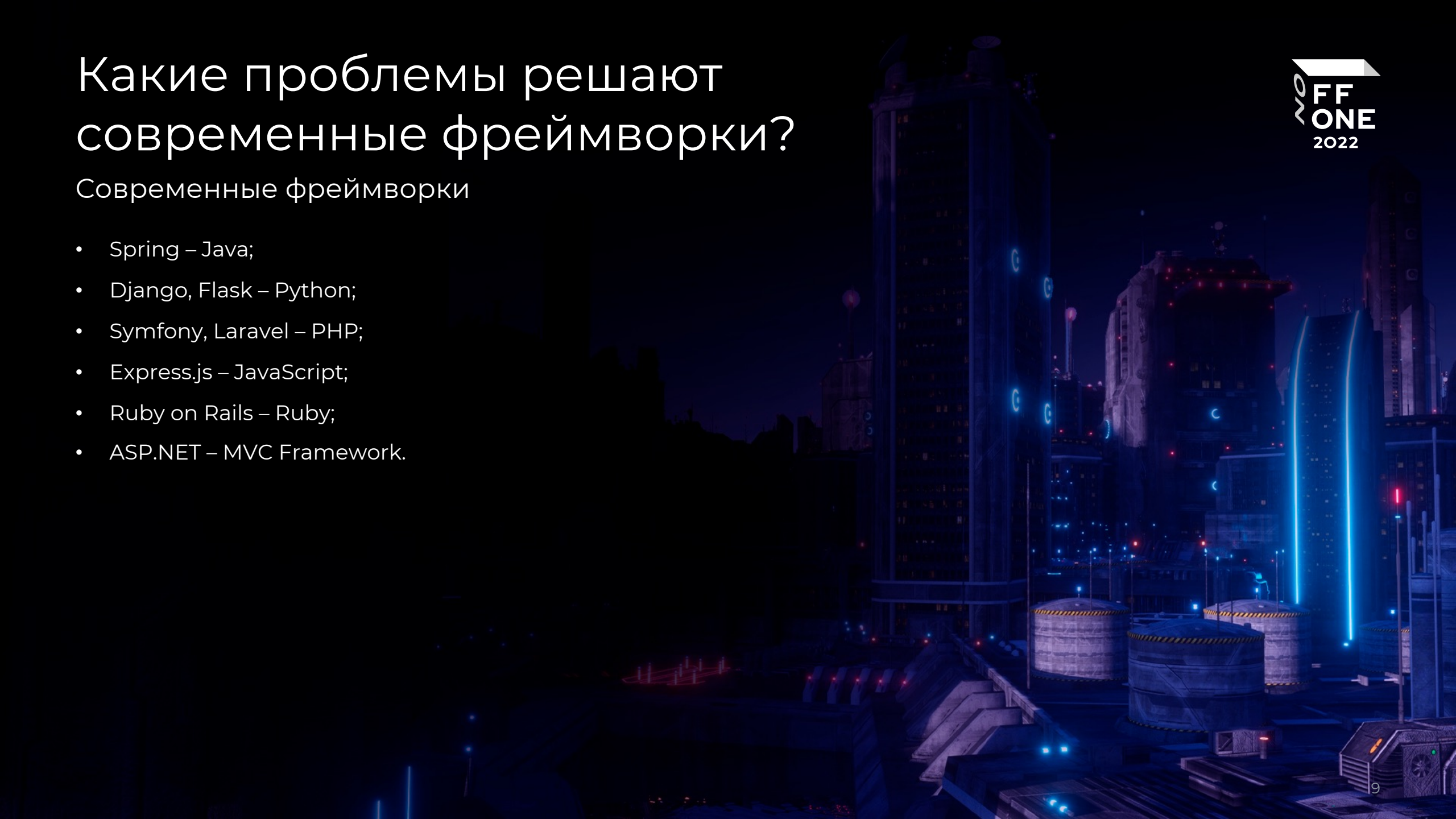
Чтобы понять меня правильно

- Не стоит переоценивать пентестеров, мы часто можем найти только те баги, которые успеваем в рамках проверки. Пентестеры не ищут 0–day в приложениях. Привет @paul_axe...
- Даже если фреймворк устраняет пласт уязвимостей, это не значит, что их не надо искать – это значит, что можно ожидать от фреймворка других уязвимостей.

Какие проблемы решают современные фреймворки?

Современные фреймворки

- Spring – Java;
- Django, Flask – Python;
- Symfony, Laravel – PHP;
- Express.js – JavaScript;
- Ruby on Rails – Ruby;
- ASP.NET – MVC Framework.



Какие проблемы решают современные фреймворки?

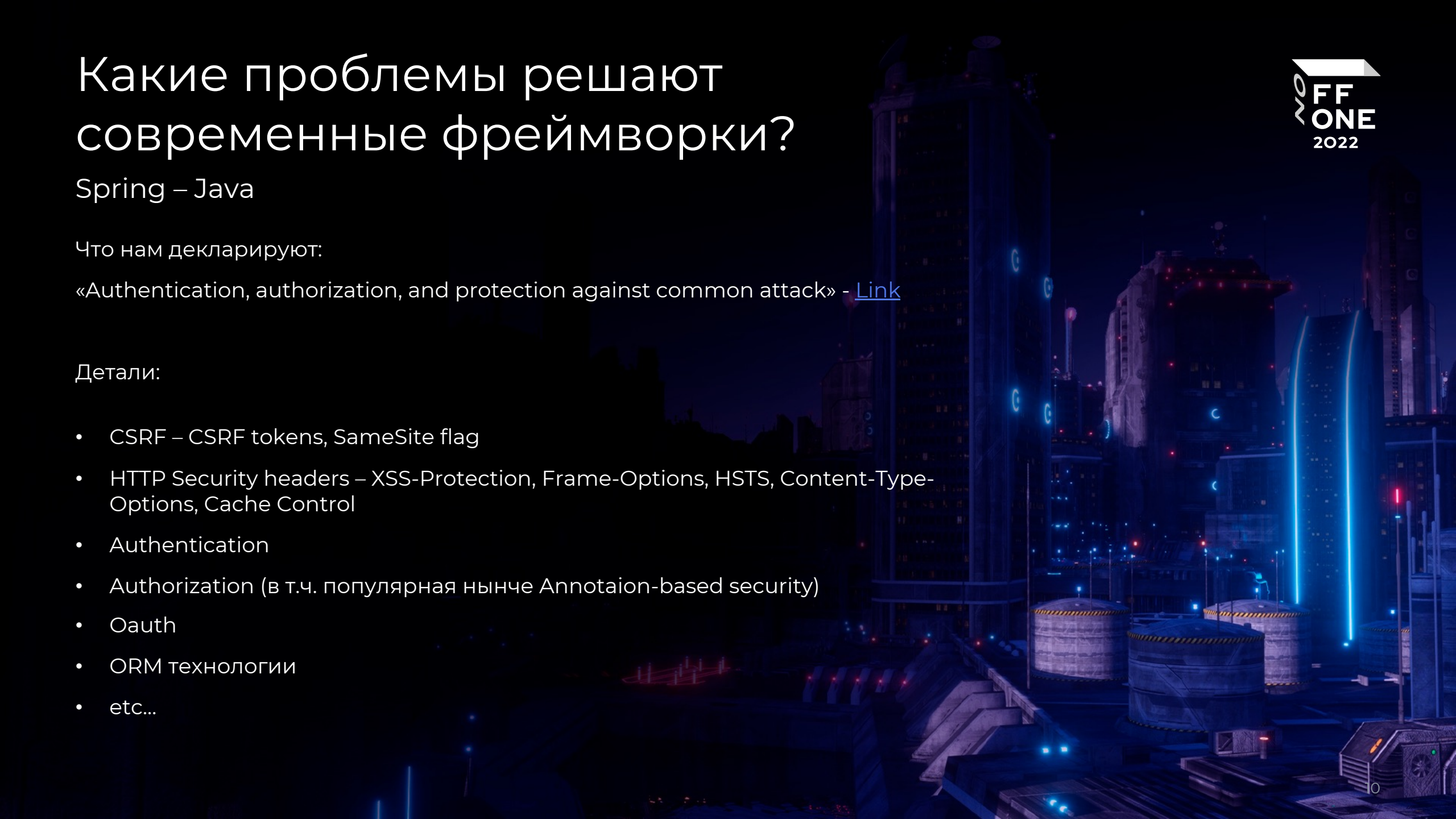
Spring – Java

Что нам декларируют:

«Authentication, authorization, and protection against common attack» - Link

Детали:

- CSRF – CSRF tokens, SameSite flag
- HTTP Security headers – XSS-Protection, Frame-Options, HSTS, Content-Type-Options, Cache Control
- Authentication
- Authorization (в т.ч. популярная нынче Annotaion-based security)
- Oauth
- ORM технологии
- etc...

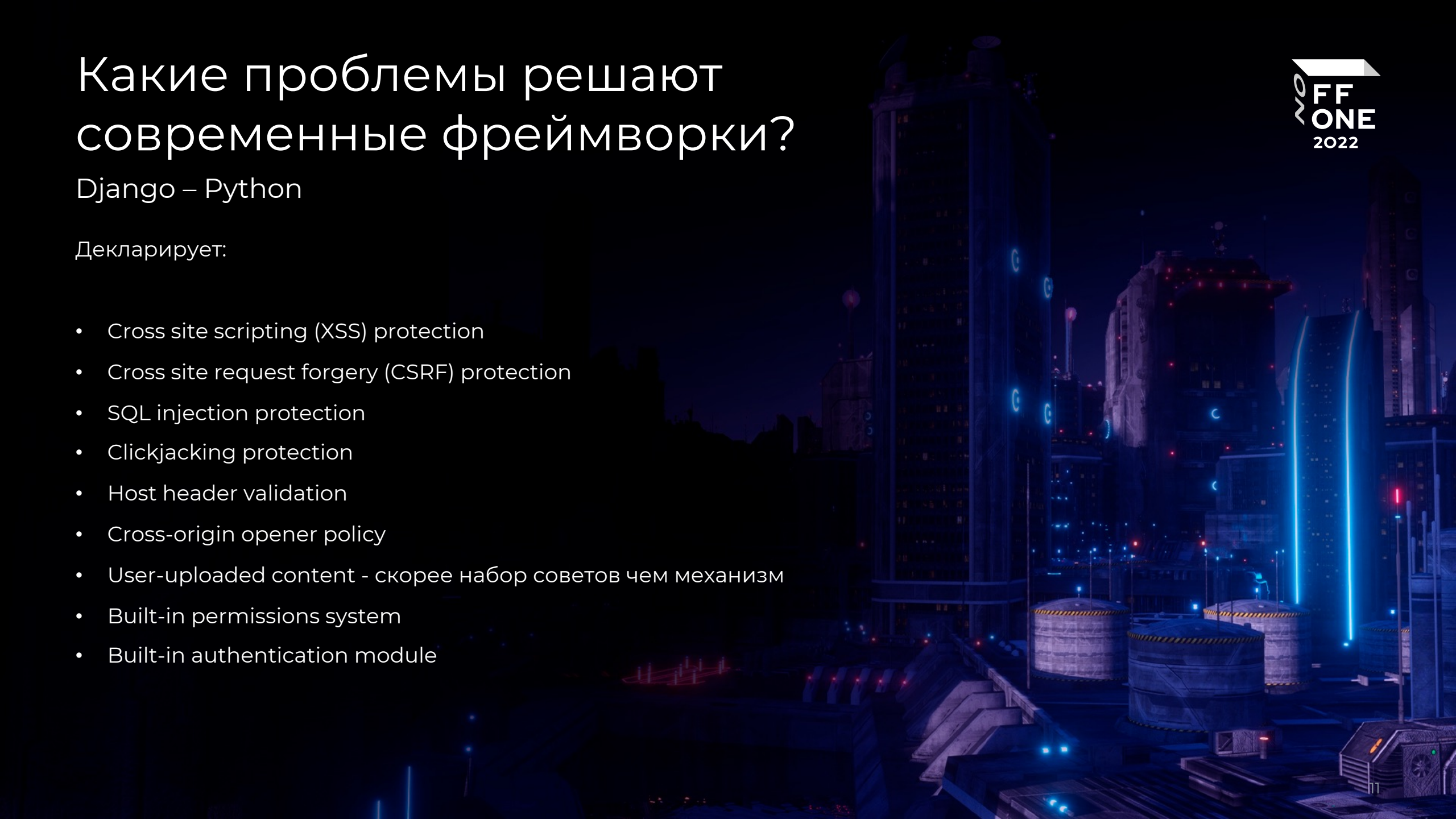


Какие проблемы решают современные фреймворки?

Django – Python

Декларирует:

- Cross site scripting (XSS) protection
- Cross site request forgery (CSRF) protection
- SQL injection protection
- Clickjacking protection
- Host header validation
- Cross-origin opener policy
- User-uploaded content - скорее набор советов чем механизм
- Built-in permissions system
- Built-in authentication module

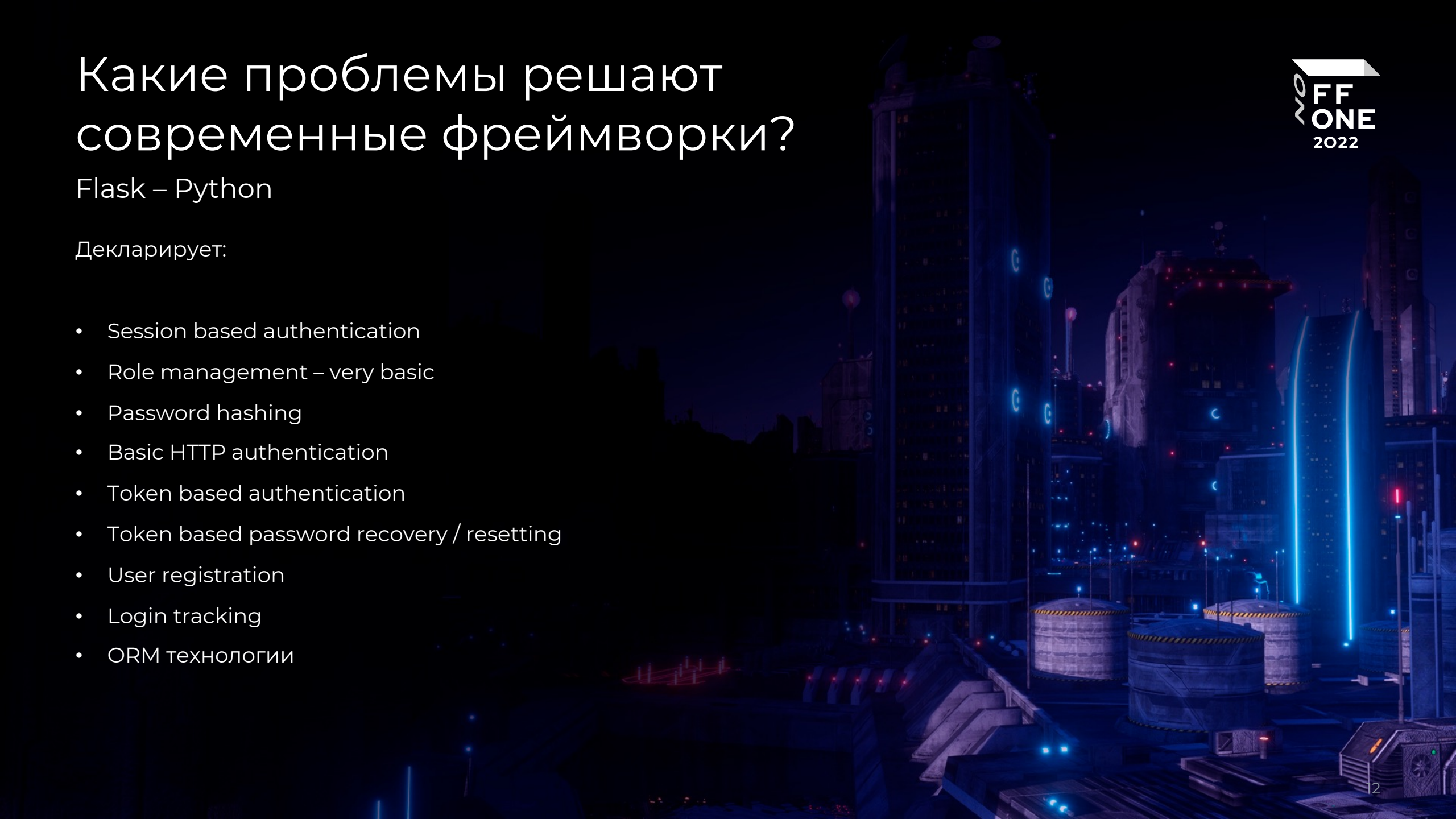


Какие проблемы решают современные фреймворки?

Flask – Python

Декларирует:

- Session based authentication
- Role management – very basic
- Password hashing
- Basic HTTP authentication
- Token based authentication
- Token based password recovery / resetting
- User registration
- Login tracking
- ORM технологии

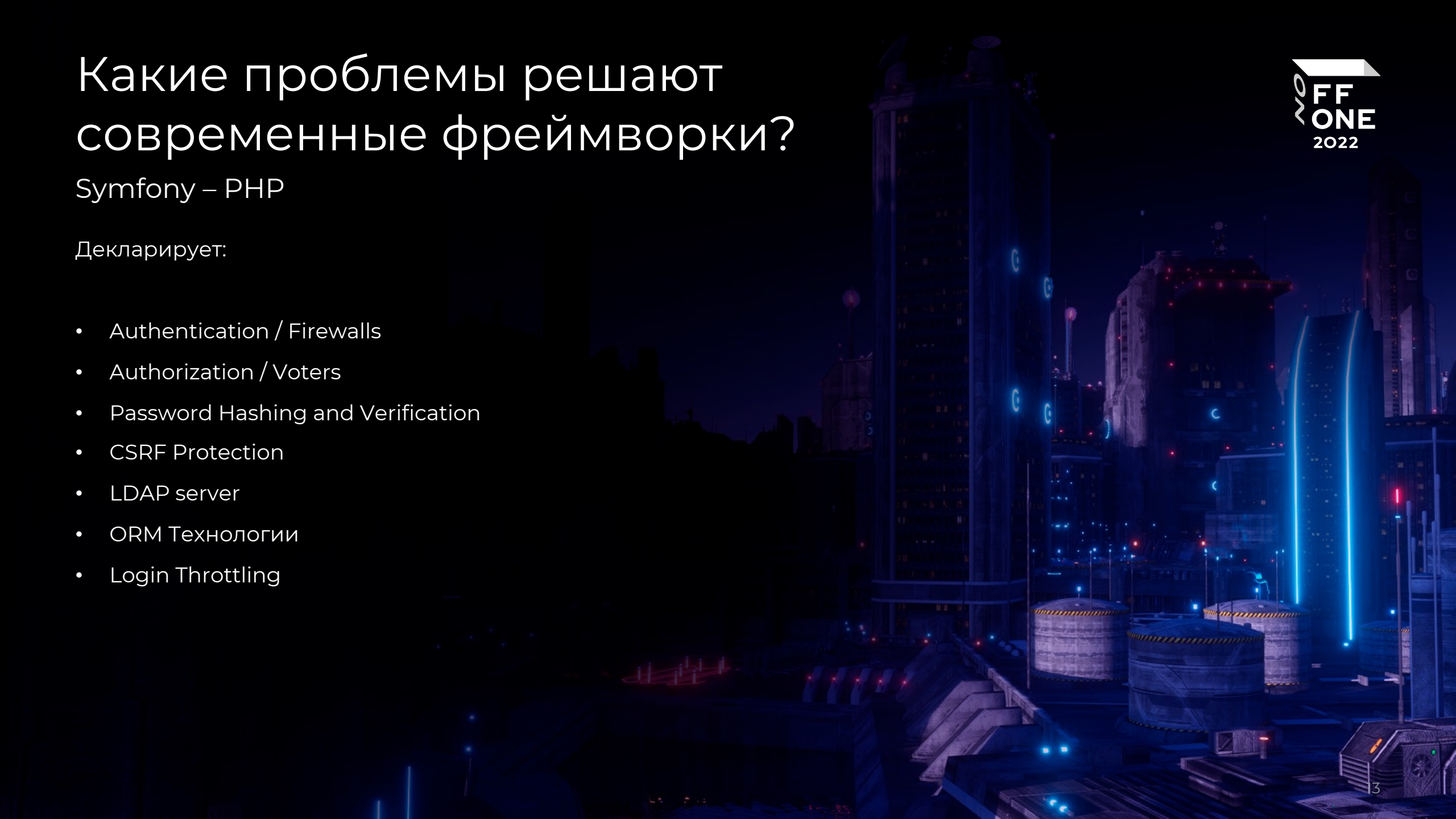


Какие проблемы решают современные фреймворки?

Symfony – PHP

Декларирует:

- Authentication / Firewalls
- Authorization / Voters
- Password Hashing and Verification
- CSRF Protection
- LDAP server
- ORM Технологии
- Login Throttling

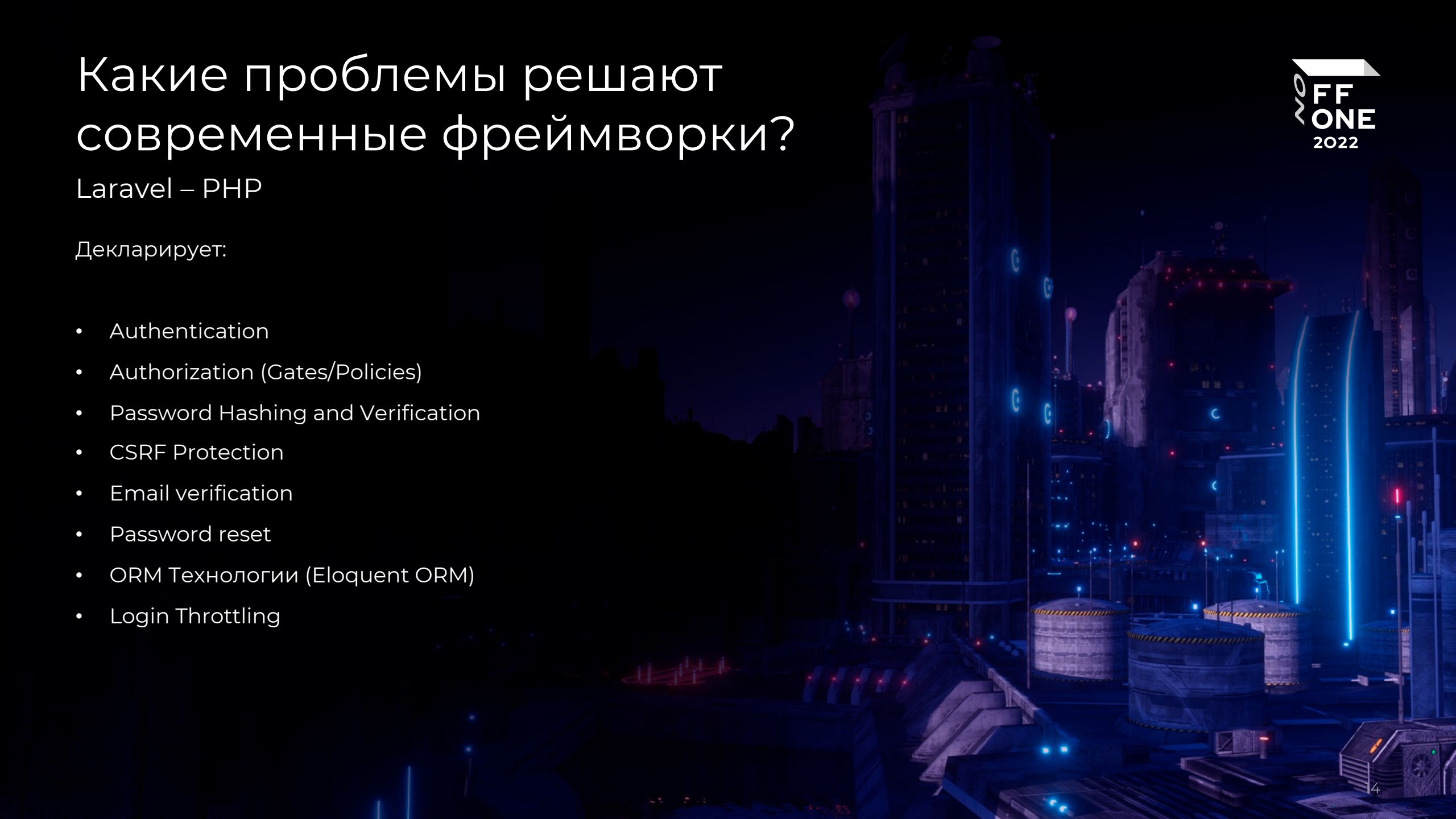


Какие проблемы решают современные фреймворки?

Laravel – PHP

Декларирует:

- Authentication
- Authorization (Gates/Policies)
- Password Hashing and Verification
- CSRF Protection
- Email verification
- Password reset
- ORM Технологии (Eloquent ORM)
- Login Throttling



Какие проблемы решают современные фреймворки?

Express.js – JavaScript

Декларирует:

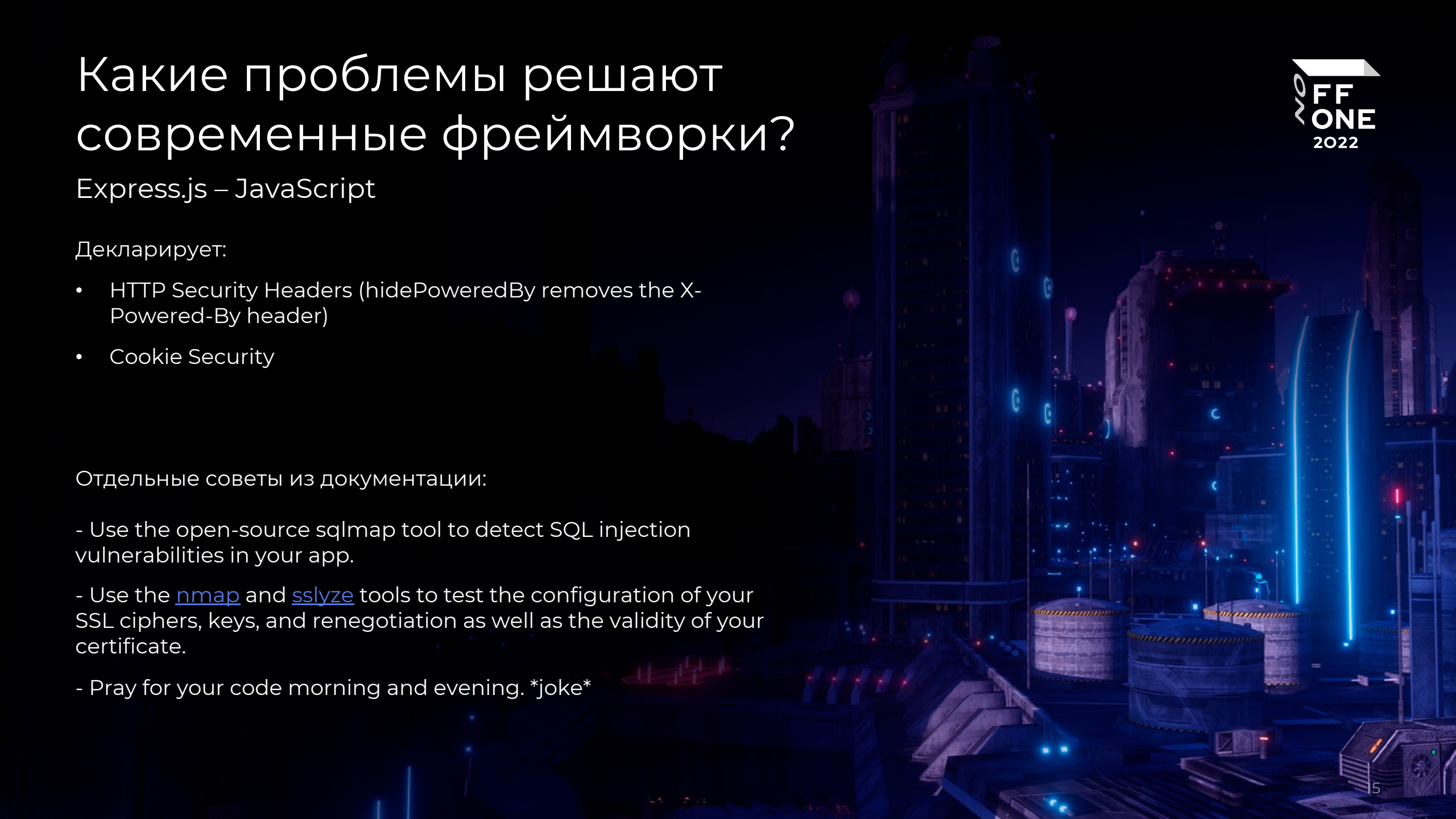
- HTTP Security Headers (hidePoweredBy removes the X-Powered-By header)
- Cookie Security

Отдельные советы из документации:

- Use the open-source sqlmap tool to detect SQL injection vulnerabilities in your app.

- Use the nmap and sslyze tools to test the configuration of your SSL ciphers, keys, and renegotiation as well as the validity of your certificate.

- Pray for your code morning and evening. *joke*



Какие проблемы решают современные фреймворки?

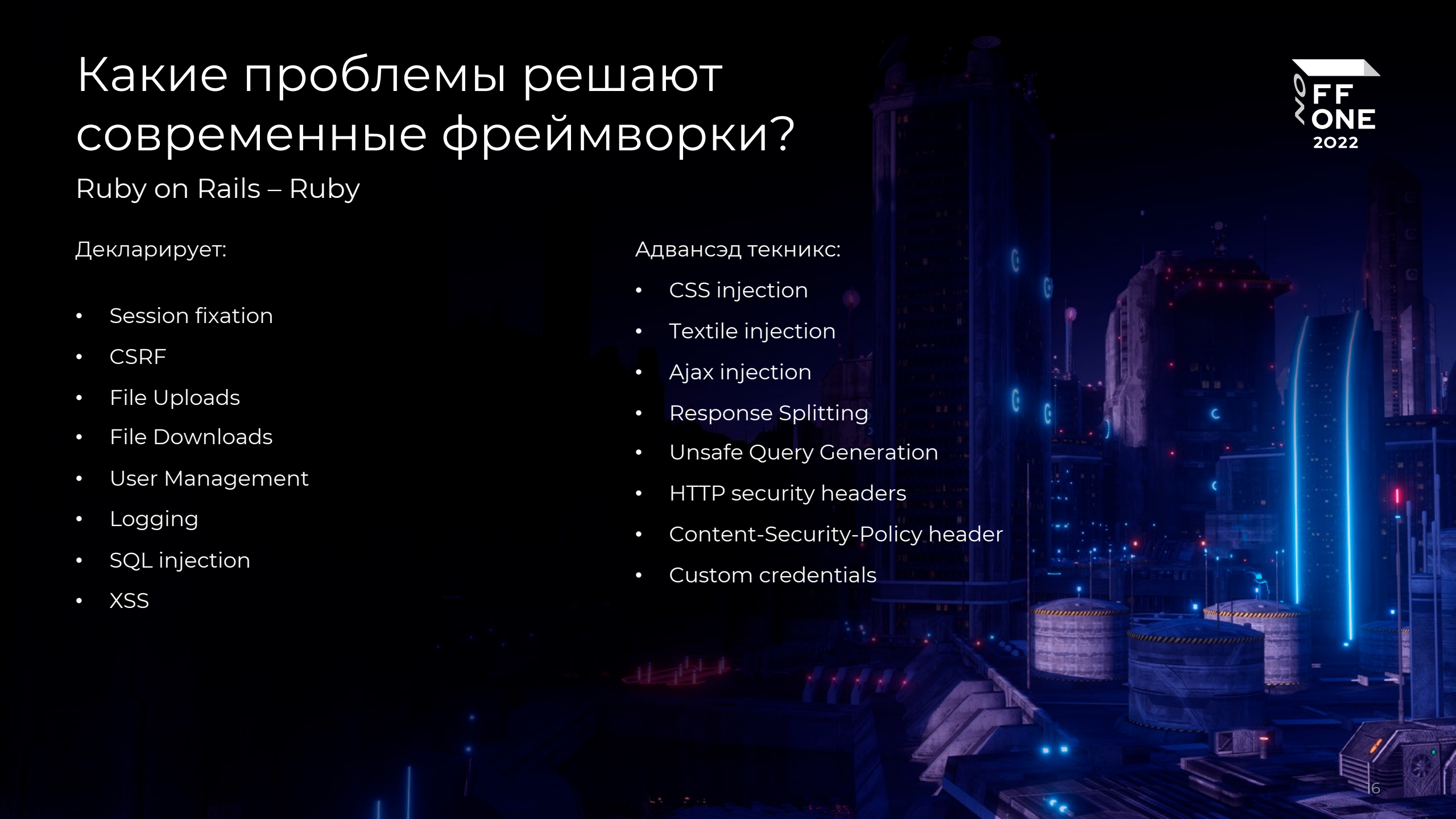
Ruby on Rails – Ruby

Декларирует:

- Session fixation
- CSRF
- File Uploads
- File Downloads
- User Management
- Logging
- SQL injection
- XSS

Адвансэд текникс:

- CSS injection
- Textile injection
- Ajax injection
- Response Splitting
- Unsafe Query Generation
- HTTP security headers
- Content-Security-Policy header
- Custom credentials

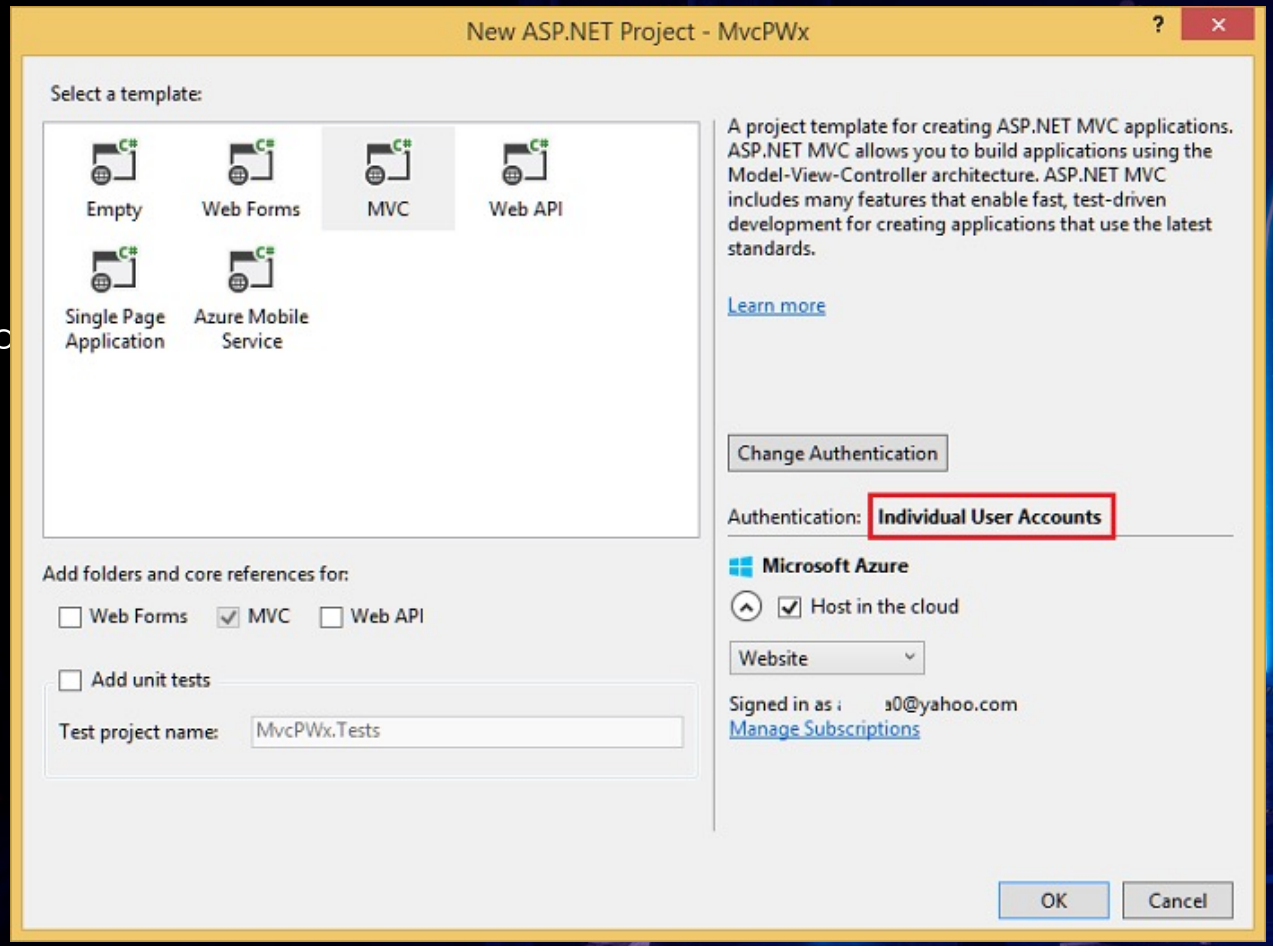


Какие проблемы решают современные фреймворки?

ASP.NET – MVC Framework

Декларирует:

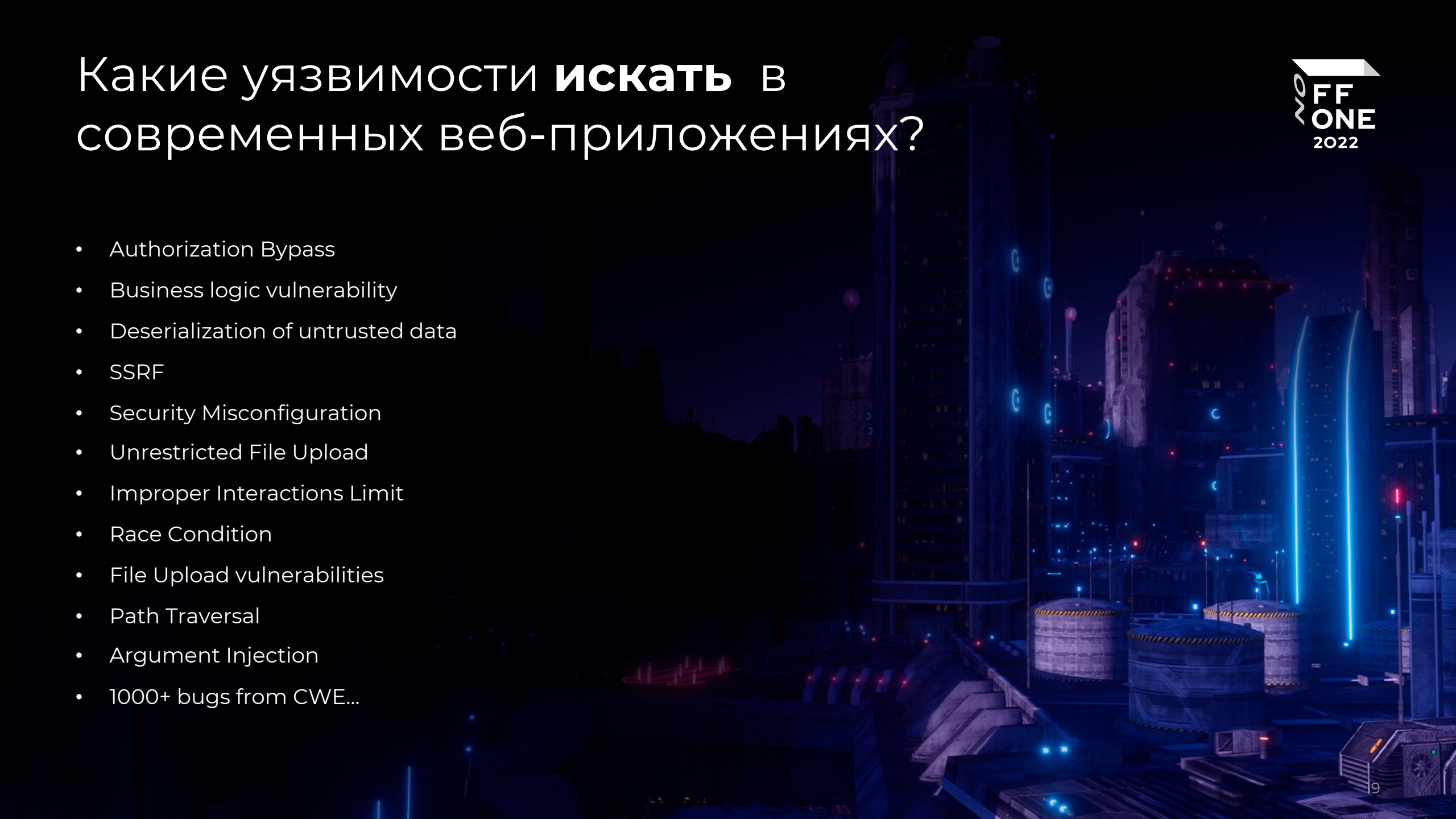
- CSRF Protection
- HTTP Security Headers
- Authorization mechanism right out of the bo
- Open Redirect protection
- Two-factor mechanism right out of the box



Какие уязвимости **не найти** в современных веб-приложениях?

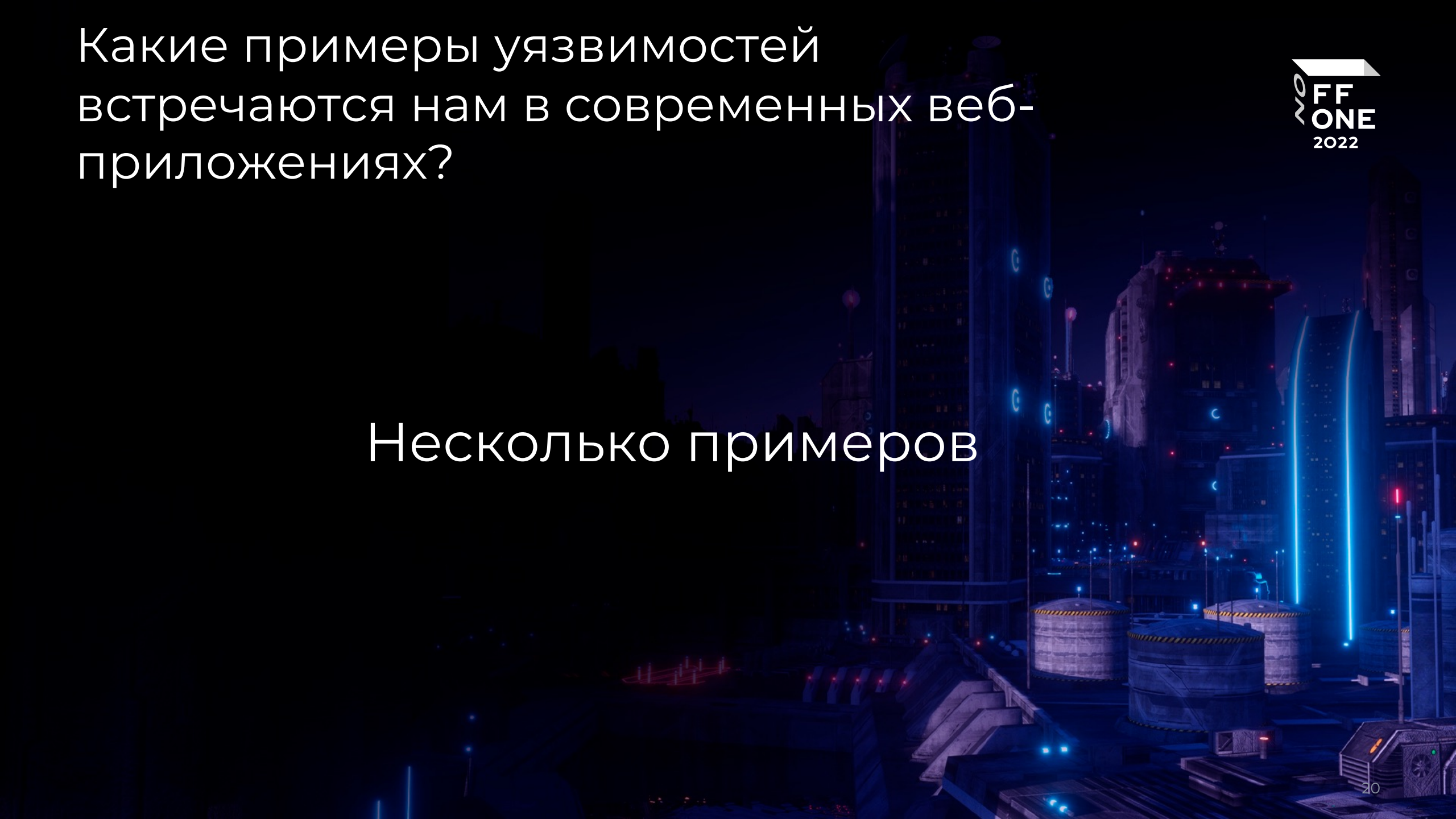
- CSRF
- Reflected / Stored XSS
- SSTI
- SQLi
- Plaintext Storage of a Password
- Session fixation
- HTTP Host Header attack

Какие уязвимости **искать** в современных веб-приложениях?



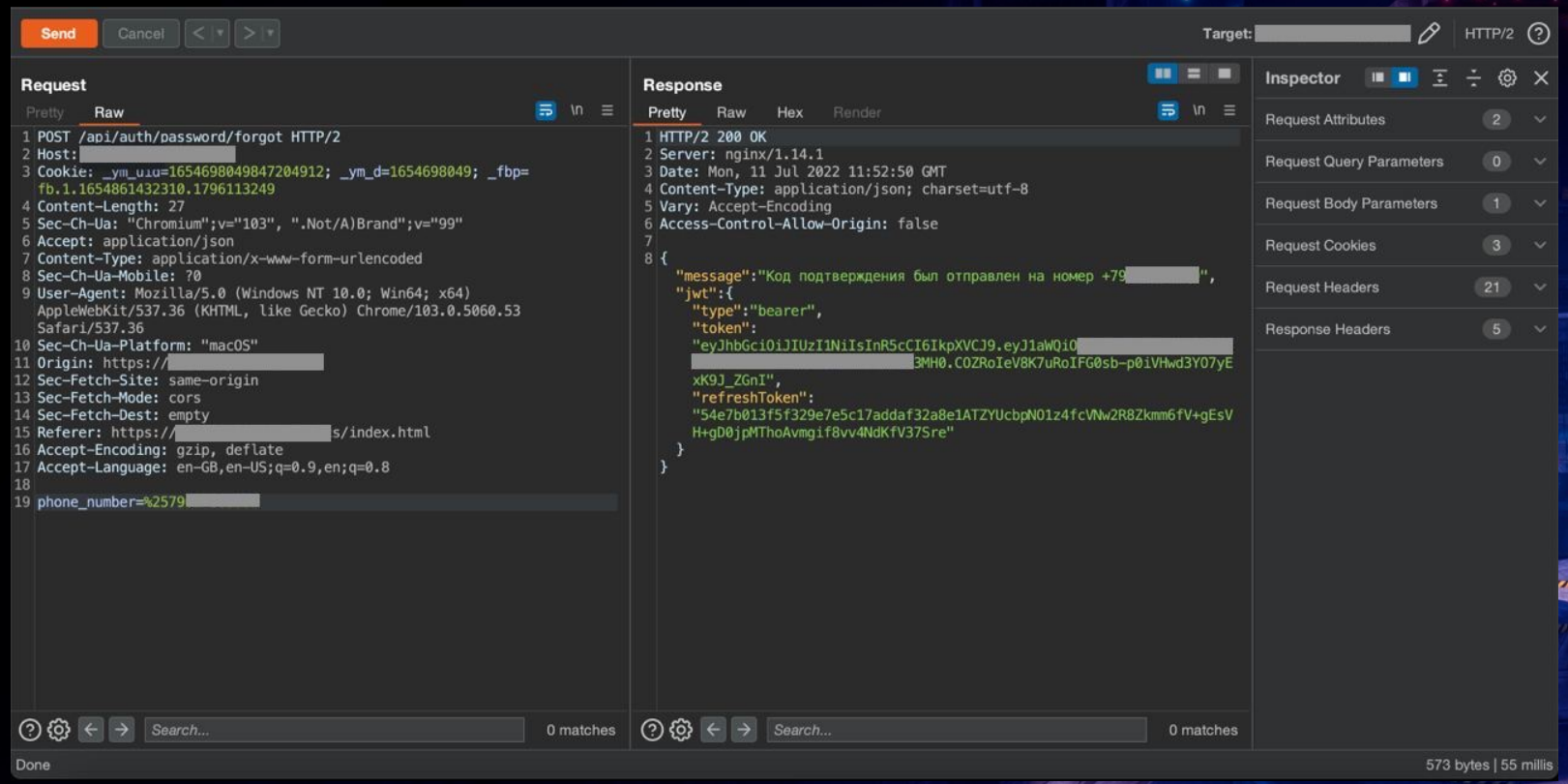
- Authorization Bypass
- Business logic vulnerability
- Deserialization of untrusted data
- SSRF
- Security Misconfiguration
- Unrestricted File Upload
- Improper Interactions Limit
- Race Condition
- File Upload vulnerabilities
- Path Traversal
- Argument Injection
- 1000+ bugs from CWE...

Какие примеры уязвимостей встречаются нам в современных веб-приложениях?



Какие примеры уязвимостей встречаются нам в современных веб-приложениях?

Password forgot from express.js



Какие примеры уязвимостей встречаются нам в современных веб-приложениях?

Spring MVC annotation-based security

Уязвимости

Ticket	Type
#466	Using Components with Known Vulnerabilities
#467	Improper Access Control
#468	Weak Password Requirements
#469	Using Components with Known Vulnerabilities
#470	Sensitive Data in GET Query String
#471	Unrestricted File Upload
#472	Improper Access Control
#473	Improper Access Control
#474	Insecure Direct Object References (IDOR)
#475	Improper Access Control
#476	Improper Access Control
#477	Improper Access Control
#478	Improper Access Control
#479	Improper Access Control
#481	Improper Access Control
#482	Improper Interaction Limits
#486	Improper Access Control
#487	Improper Access Control
#488	Path Traversal
#489	Insecure Direct Object References (IDOR)
#490	Improper Access Control
#491	Improper Access Control
#493	Use of Hard-coded Cryptographic Key
#494	Insecure Direct Object References (IDOR)
#496	Using Components with Known Vulnerabilities
#497	Using Components with Known Vulnerabilities
#498	Using Components with Known Vulnerabilities
#499	Using Components with Known Vulnerabilities
#500	Sensitive Data in GET Query String

Какие примеры уязвимостей встречаются нам в современных веб-приложениях?

Spring MVC annotation-based security

Исследуемое приложение позволяет любому аутентифицированному пользователю удалять любые файлы в файловом хранилище приложения.

Через API метод `DELETE /api/attachments/{{ID}}` можно удалить произвольный файл в файловом хранилище указав идентификатор сообщения, где приложен файл. Идентификаторы сообщений являются целыми числами изменяющимися посредством инкрементирования, что позволяет предсказать идентификатор сообщения.

Потенциальный злоумышленник может перебрать все идентификаторы файловых вложений от 1 до n, где n - номер последнего сообщения с файловым вложением в системе, и удалить все файлы, хранящиеся в файловом хранилище.

Какие примеры уязвимостей встречаются нам в современных веб-приложениях?

Диверсия

```php
    public function authFromToken(ProviderRequest $request): JsonResponse
    {
        $provider = $request->provider;
        $providerToken = $request->token;
        $email = $request->email ?? null;

        if ($provider == 'apple') {
            try {

                $token = $this->appleAuth($providerToken);
                return response()->json(['token' => $token], 200);

            } catch (\Exception $exception) {
                return response()->json(['error' => $exception->getMessage()]
                    , 422);
            }

        }
```

Какие примеры уязвимостей встречаются нам в современных веб-приложениях?

Диверсия

```php
    protected function appleAuth($token)
    {
        // parse JWT
        $key = 'your-256-bit-secret';
        $jwtToken = $token;
        $jwtArr = @array_combine(['header', 'payload', 'signature'], explode(
            '.', $jwtToken));
        if ($jwtArr === false) {
            throw new \Exception('Apple auth not valid token');
        }
        $payload = base64_decode($jwtArr['payload']);
        $payload = \json_decode($payload);

        if (!isset($payload->email) || empty($payload->email)) {
            throw new \Exception('Apple failed token validation');
        }

        $user = User::firstOrCreate(
            [
                'email' => $payload->email,
            ],
            [
                'email_verified_at' => now(),
            ]
        );
```

Где проще всего найти такие проблемы?

OFF ONE 2022

В механизмах авторизации

В механизмах восстановления пароля

В механизмах загрузки / скачивания файлов

В основных бизнес-процессах приложения

В сторонних каналах аутентификации

В механизмах выполнения запросов к сторонним ресурсам

В механизмах денежных переводов и конвертации

В конфигурации и реализации фреймворков и библиотек

Как быть успешнее в поисках таких проблем и соответственно в их исключении?

Изучать особенности конфигурации фреймоворка

Изучать бизнес-логику приложения и сбора «тонких мест» в приложении

Собирать все возможные методы API приложений и работать над покрытием проверками.

Больше прогать и читать код: прокачивать насмотренность в тонких местах.

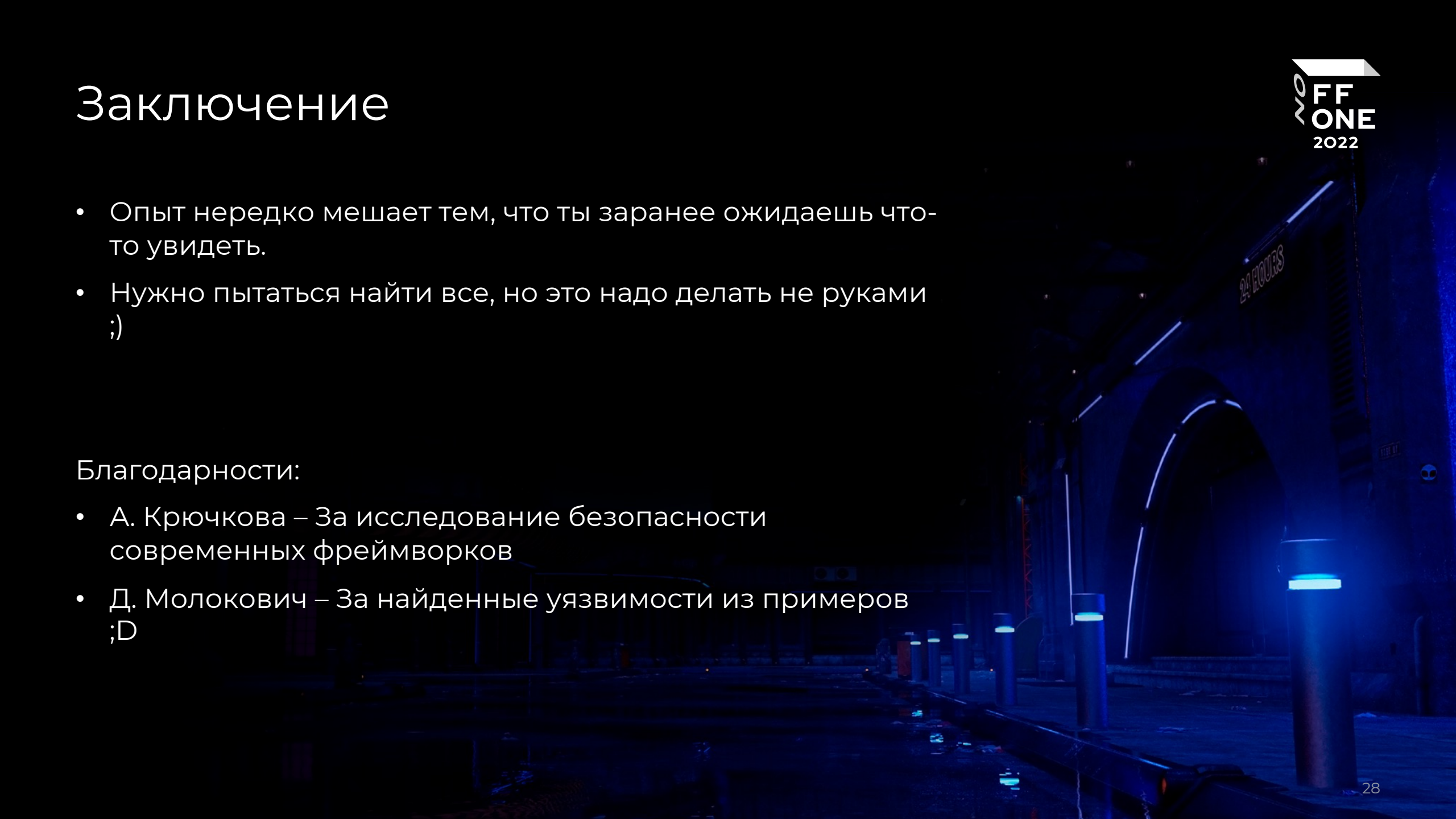
Использовать автоматизацию для проверки Авторизации в API методах

Составлять карту покрытия и модель угроз

Думать головой ☺

Заключение

- Опыт нередко мешает тем, что ты заранее ожидаешь что-то увидеть.
- Нужно пытаться найти все, но это надо делать не руками ;)



Благодарности:

- А. Крючкова – За исследование безопасности современных фреймворков
- Д. Молокович – За найденные уязвимости из примеров ;D

ON
FF
ONE
2022

Спасибо за ваше внимание!


OFF
ONE
2022