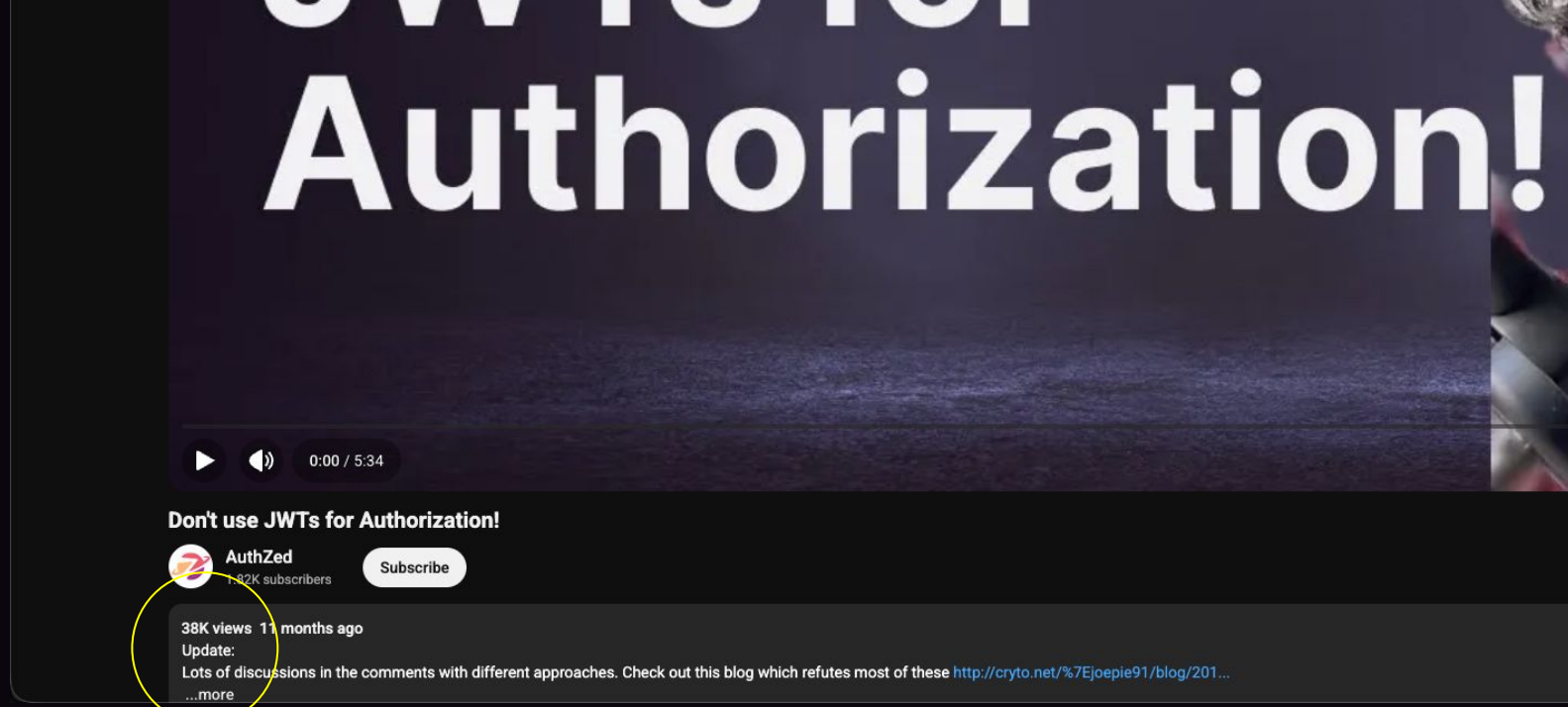
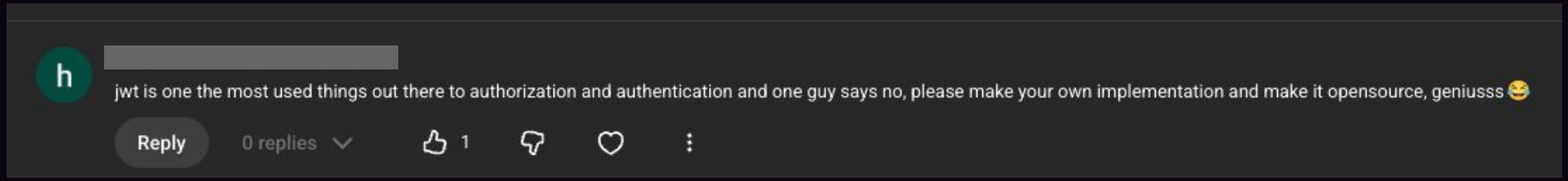
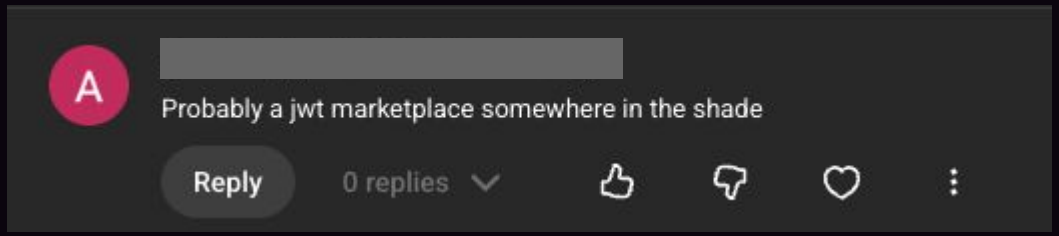
# Don’t Use JWTs for Authorization ⚠

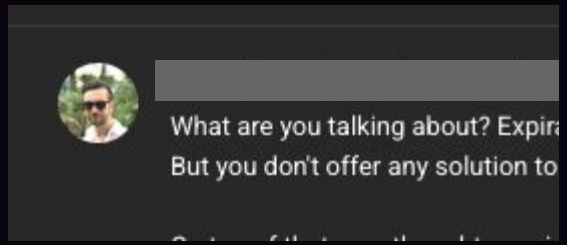
Sohan Maheshwar
Developer Relations @ AuthZed

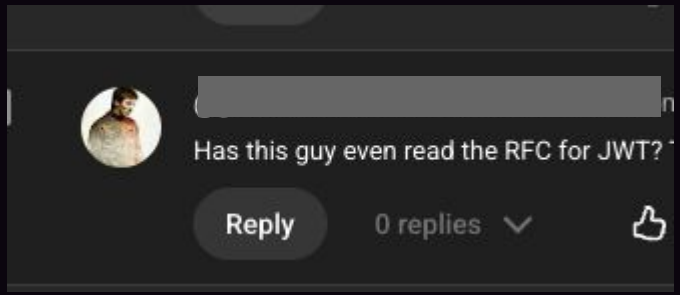

Brave File Edit View History Bookmarks Profiles Tab Window Help
youtube.com/watch?v=_cgcg-h4vJs
YouTube
Search
AUTHZ BASICS
Don't use JWTs for Authorization!
0:00 / 5:34
Don't use JWTs for Authorization!
AuthZed
1.82K subscribers
Subscribe
891
Share
Save
38K views 11 months ago
Update:
Lots of discussions in the comments with different approaches. Check out this blog which refutes most of these http://cryto.net/%7Ejoepie91/blog/201...
...more

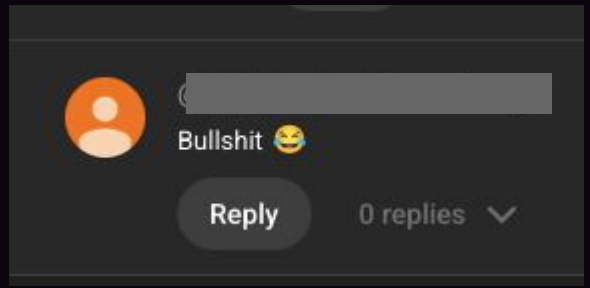

Authorization!
0:00 / 5:34
Don't use JWTs for Authorization!
AuthZed
1.82K subscribers
Subscribe
38K views 11 months ago
Update:
Lots of discussions in the comments with different approaches. Check out this blog which refutes most of these http://cryto.net/%7Ejoepie91/blog/201...
...more


h
jwt is one the most used things out there to authorization and authentication and one guy says no, please make your own implementation and make it opensource, geniusss
Reply
0 replies
1


A
Probably a jwt marketplace somewhere in the shade
Reply
0 replies


What are you talking about? Expira
But you don't offer any solution to


Has this guy even read the RFC for JWT?
Reply
0 replies

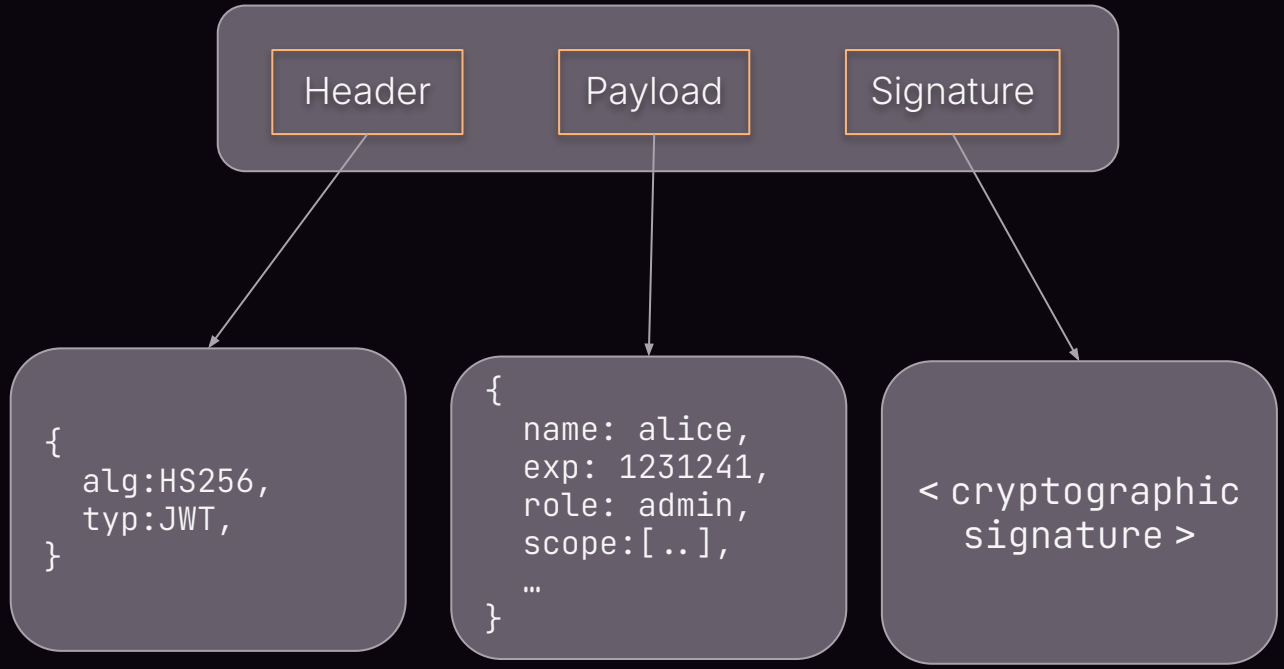

Bullshit
Reply
0 replies

# JSON Web Token aka `jot`

- Header
- Payload (claims)
- Signature

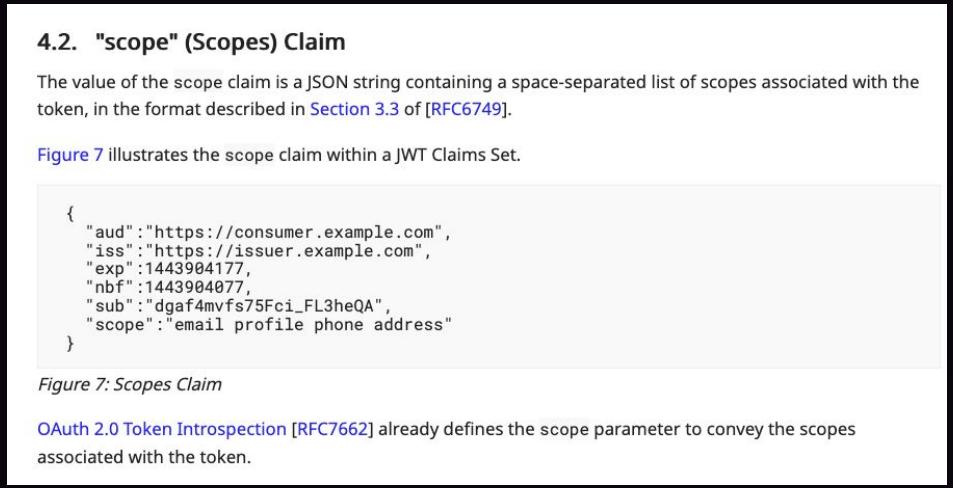
Common claims:

- `exp`
- `iss`
- `aud`
- `scope`



# JSON Web Token aka `jot`

- `scope` - not even from JWT spec!
- OAuth2 Token Exchange spec



### 4.2. "scope" (Scopes) Claim

The value of the scope claim is a JSON string containing a space-separated list of scopes associated with the token, in the format described in Section 3.3 of [RFC6749].

Figure 7 illustrates the scope claim within a JWT Claims Set.

```
{
  "aud":"https://consumer.example.com",
  "iss":"https://issuer.example.com",
  "exp":1443904177,
  "nbf":1443904077,
  "sub":"dgaf4mvfs75Fci_FL3heQA",
  "scope":"email profile phone address"
}
```

*Figure 7: Scopes Claim*

OAuth 2.0 Token Introspection [RFC7662] already defines the scope parameter to convey the scopes associated with the token.

https://www.rfc-editor.org/rfc/rfc8693#name-scope-scopes-claim

## Assumption

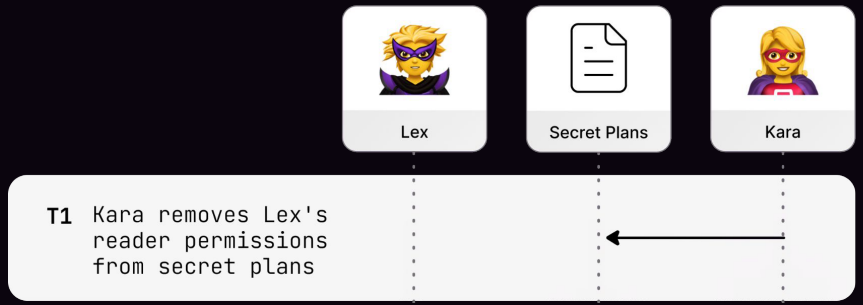
“Permissions don’t change during token lifetime”

# Problem #1 - No Meaningful Revocation

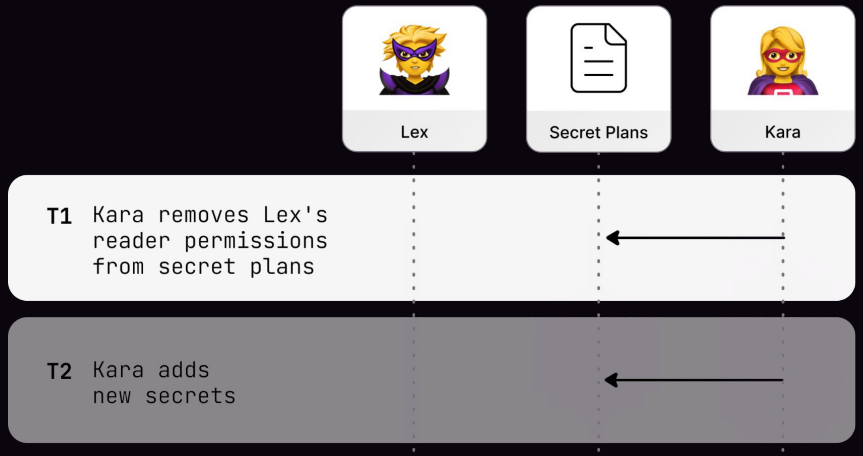
- You can’t invalidate a JWT early
  - User removed from org?
  - Role downgraded?
  - Access revoked?

- JWTs are
  - Self-contained
  - Stateless
- No central authority to check reality
- Token still works until `exp`

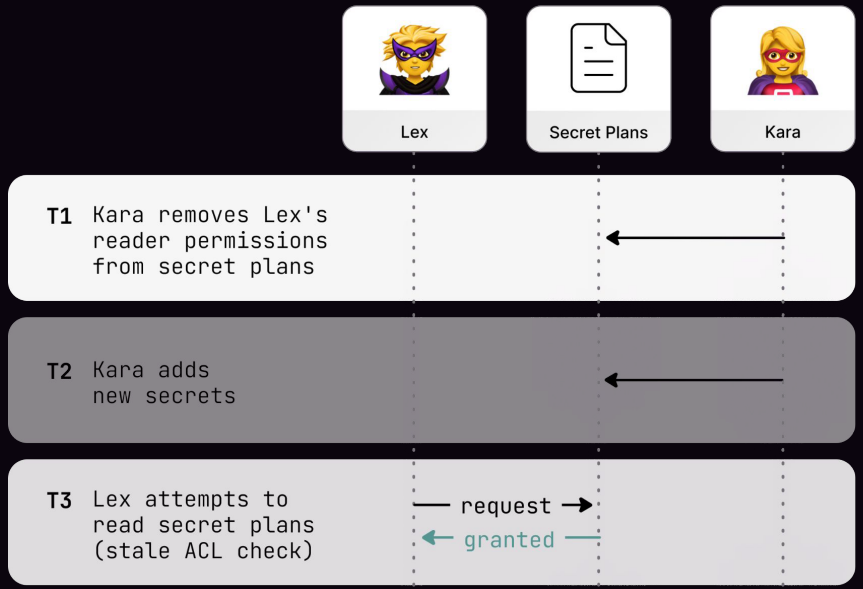
# New Enemy Problem



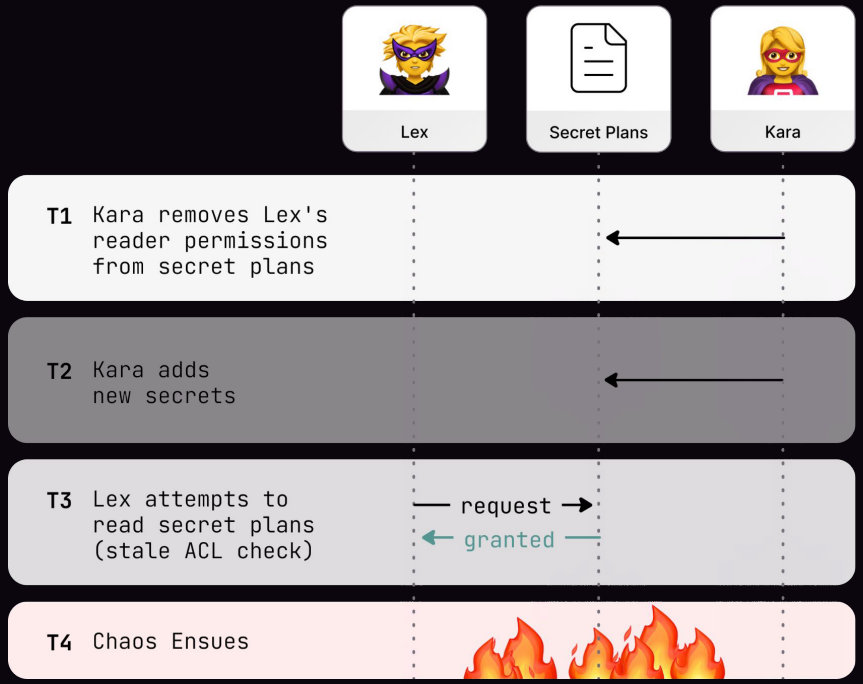
# New Enemy Problem



# New Enemy Problem

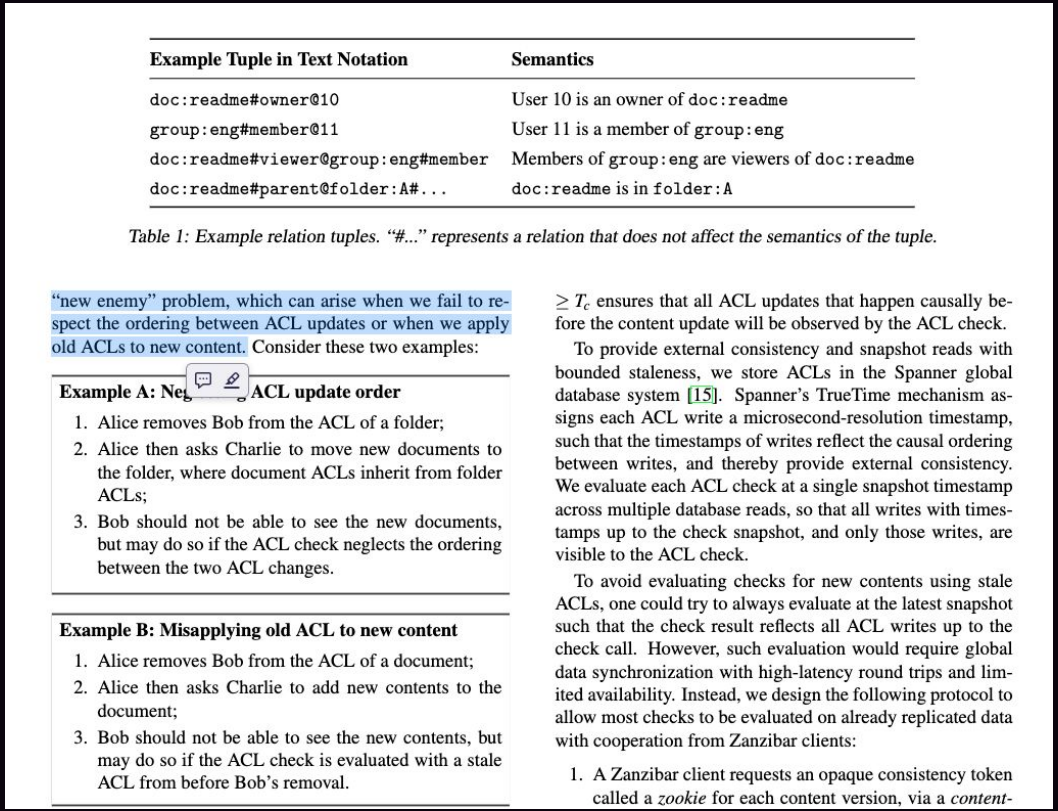


# New Enemy Problem



# Google Zanzibar

- Globally distributed authorization system
- Used in Docs, Sheets, Maps, YouTube, Gmail & more



| Example Tuple in Text Notation | Semantics |
|---|---|
| `doc:readme#owner@10` | User 10 is an owner of `doc:readme` |
| `group:eng#member@11` | User 11 is a member of `group:eng` |
| `doc:readme#viewer@group:eng#member` | Members of `group:eng` are viewers of `doc:readme` |
| `doc:readme#parent@folder:A#...` | `doc:readme` is in `folder:A` |

*Table 1: Example relation tuples. "#..." represents a relation that does not affect the semantics of the tuple.*

"new enemy" problem, which can arise when we fail to respect the ordering between ACL updates or when we apply old ACLs to new content. Consider these two examples:

**Example A: Ne… ACL update order**

1. Alice removes Bob from the ACL of a folder;
2. Alice then asks Charlie to move new documents to the folder, where document ACLs inherit from folder ACLs;
3. Bob should not be able to see the new documents, but may do so if the ACL check neglects the ordering between the two ACL changes.

**Example B: Misapplying old ACL to new content**

1. Alice removes Bob from the ACL of a document;
2. Alice then asks Charlie to add new contents to the document;
3. Bob should not be able to see the new contents, but may do so if the ACL check is evaluated with a stale ACL from before Bob's removal.

$\geq T_c$ ensures that all ACL updates that happen causally before the content update will be observed by the ACL check.

To provide external consistency and snapshot reads with bounded staleness, we store ACLs in the Spanner global database system [15]. Spanner's TrueTime mechanism assigns each ACL write a microsecond-resolution timestamp, such that the timestamps of writes reflect the causal ordering between writes, and thereby provide external consistency. We evaluate each ACL check at a single snapshot timestamp across multiple database reads, so that all writes with timestamps up to the check snapshot, and only those writes, are visible to the ACL check.

To avoid evaluating checks for new contents using stale ACLs, one could try to always evaluate at the latest snapshot such that the check result reflects all ACL writes up to the check call. However, such evaluation would require global data synchronization with high-latency round trips and limited availability. Instead, we design the following protocol to allow most checks to be evaluated on already replicated data with cooperation from Zanzibar clients:

1. A Zanzibar client requests an opaque consistency token called a *zookie* for each content version, via a *content-*

www.zanzibar.tech

# Problem #2 - Scopes don’t scale

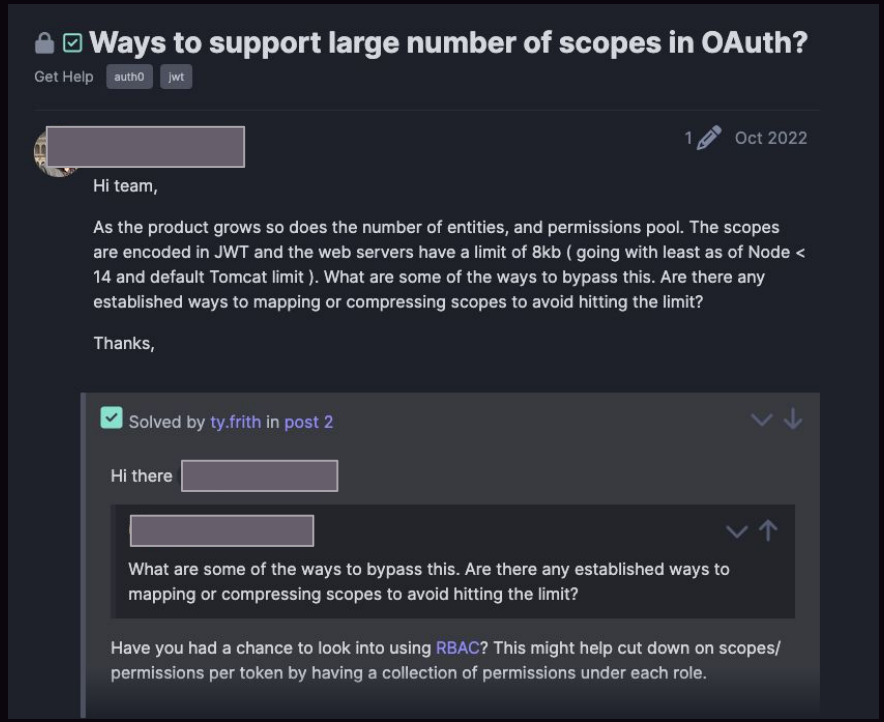
- `scope: "profile:admin"`
- Questions:
  - What data?
  - Whole website?
  - Current user’s profile?

# Fine-grained authorization

- Example:
  - `issue/authzed/spicedb/52:author` instead of `issue:author`
- Users can have access to millions of software objects
- JWTs can’t carry this at scale

# Moar Scopes!

- Tokens get huge
- Network overhead explodes
- Tokens can still be stale and/or ambiguous



# Problem #3 - Predicting Permissions

- JWT is created at request entry point
- It must know every:
  - Downstream service
  - Permission needed
- Impossible in microservices

- Over-scoped tokens
  - Extra permissions leak downstream
  - Attack surface increases
  - Privilege escalation risk

# Modern Systems

- Modern systems are:
- 
- Distributed
- Dynamic
- Composable
- 
- JWT assumes:
- 
- Static permission model
- 
- ➡️ Mismatch

# Fixes

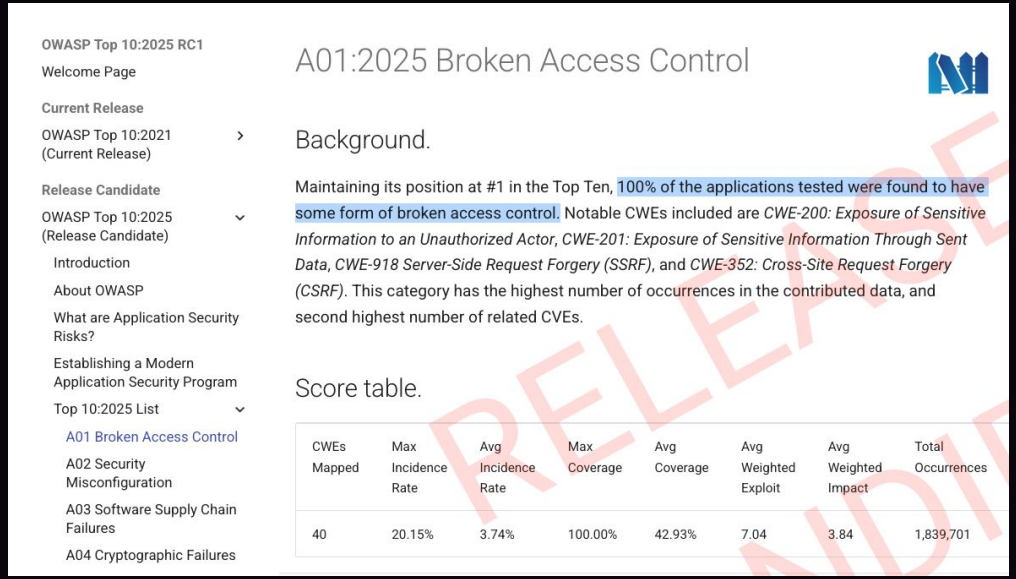
- Macaroons
  - Attenuation (reduce scope)
  - More flexible
- Reality:
  - Complex
  - Hard to adopt
  - Rarely used correctly


https://research.google/pubs/macaroons-cookies-with-contextual-caveats-for-decentralized-authorization-in-the-cloud/

# Access Control Is Complex!

# OWASP Top 10: 2025

- Broken Access Control at #1
- Was #1 in 2021
- 100% of apps tested had broken access control




owasp.org/Top10/2025/0x00_2025-Introduction/

# Recommendation: Use Centralized AuthZ

Mental model:

“Token tells you access” → “Ask a system if access is allowed”

Query on demand: `can(user, action, resource)` → `true/false`

- Real-time decisions
- Revocation works instantly
- Fine-grained access
- No over-scoped tokens

# Modern Authorization Methods

- Google Zanzibar Systems
  - AuthZ as a graph of relationships (ReBAC)
  - Eg: SpiceDB, OpenFGA, Ory Keto

- Policy Decision Points (PDPs)
  - AuthZ as a logic problem solved by evaluating policies.
  - Eg: OPA, Cedar, Cerbos




PERMIFY

# Tradeoffs

You trade:

- Stateless simplicity

For:

- Correctness
- Security
- Flexibility

# When to use JWTs

Valid use case:

- One-time, non-revocable grants
- Examples:
  - Email verification links
  - Short-lived download tokens

AVOID for:

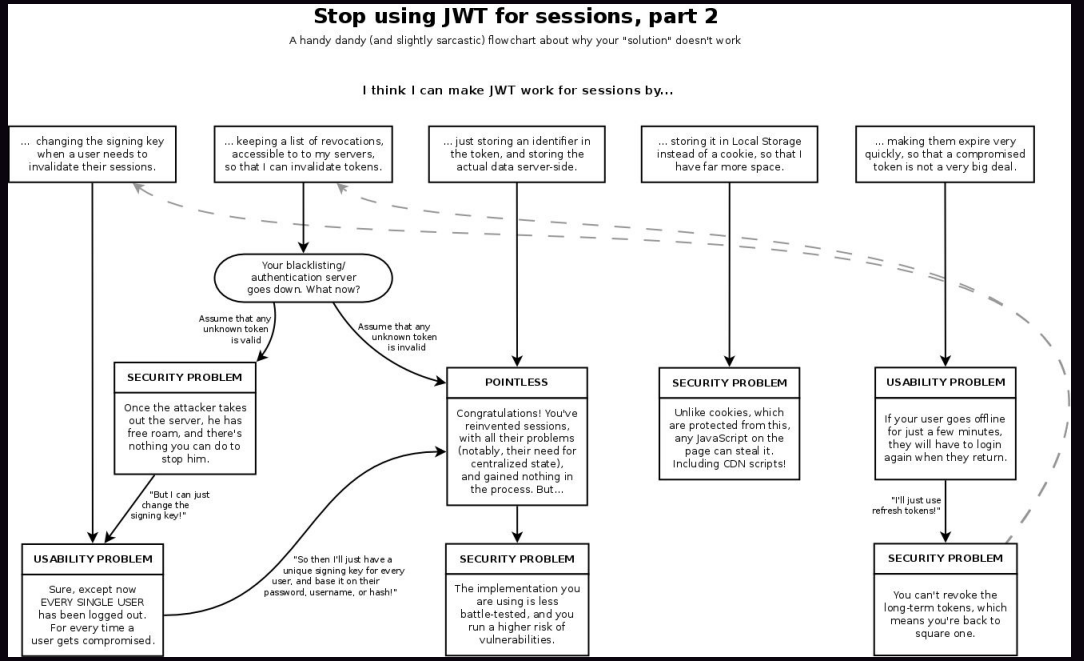
- Long-lived sessions
- App Authorization decisions
- Fine-grained permissions

# Key Takeaways

- JWTs are not for app authorization
- Revocation matters (a lot)
- Scopes don’t scale
- Prediction is impossible
- Centralized authZ fixes these

# Next Steps

- Stop using JWT for sessions:
  - cryto.net/~joepie91/blog/2016/06/19/stop-using-jwt-for-sessions-part-2-why-your-solution-doesnt-work
- API Tokens: A Tedious Survey
  - fly.io/blog/api-tokens-a-tedious-survey/#jwt
- Read the Zanzibar paper
  - zanzibar.tech



# Thank You!

Sohan Maheshwar
linkedin.com/in/sohanmaheshwar/

**Sohan Maheshwar**
Lead Developer Advocate @ AuthZed | Developer Relations