Scaling Backend Authentication at Facebook

Kevin Lewi, Callen Rain, Stephen Weis, Yueting Lee, Haozhi Xiong, Benjamin Yang

Facebook
Infrastructure Security

Network Perimeter

Trusted Services
Building from a Root of Trust

- More trust
- Fewer machines

- Less trust
- More machines

"Walled Garden"
How can we scale authentication while minimizing our root of trust?
Trusted Components

- **Key Server** (Holds Master Keys)
- **Login Server** (Signs Sessions)
- **Root CA** (Signs Certificates)
- **Authorization Server** (Signs ACLs)
Authentication and Authorization

### Identities
- User: “Callen Rain”
- Machine: server123.fb.com
- Service: Image Uploading

#### Access Control Lists (ACLs)
- Resource: “Who can access table X in database Y?”
  - Identity1
  - Identity2
  ...
Service Authentication with TLS

Identity Distribution

- Request Cert
- Deploy Cert
- Root CA
- Client

Authorization

- Auth Server
- Server
- ACL
- Check Permission
- TLS

Check Permission

ACL
Service Authentication with TLS

Client → Server

I am “Client” → ALLOW

Check Permission → ACL: “Client is ok”
Service Authentication with TLS

Client 1

Client 2

Client 3

Server

ALLOW

Check Permission

ACL: “Client is ok”
Intermediate Proxies

Client → Proxy: I am "Client"
Proxy → Server: I am "Proxy"
Server: Check Permission
ACL: “Client is ok”
Intermediate Proxies

Client \rightarrow Proxy \rightarrow Server

ACL: "Client is ok" "Proxy is ok"

Check Permission

ACL: "Client is ok" "Proxy is ok"
Intermediate Proxies

Client 1 → Proxy → Server 1
ACL: "Client 1 is ok"

Client 2 → Proxy → Server 2
ACL: "Client 2 is ok"

Client 3 → Proxy → Server 3
ACL: "Client 3 is ok"

Check Permission

ACL: "Client 1 is ok"
ACL: "Client 2 is ok"
ACL: "Client 3 is ok"
Tokens

- Client
- Proxy
- Server

`$` token

TLS

ACL: “Client is ok”

Check Permission
Tokens

1. Certificate-Based Tokens

2. Crypto Auth Tokens (CATs)
Certificate-Based Tokens

Client \( \rightarrow \) Proxy \( \rightarrow \) Server

- Client
  - build()\( ightarrow \)
  - Cert
  - Key

- Proxy
  - $\rightarrow$

- Server
  - verify()\( ightarrow \)
  - CA Cert
Certificate-Based Token Creation

- client certificate
- proxies
- resource
- actions

- metadata
- signature

signature(private key, metadata)

serialize:

1d229271928d3f9e2bb0375bdf572d396f
36fae9206628714fb2ce00f72e94f258b6ce5857596baaa7e917bc7ff34fb8730b48d248969ecc2d86151b63c214b0eba55fe8730b48d248969ecc2d86151b63c214b0eba55fe8730b48d248969ecc2d86151b63c214b0eba55fe8730b48d248969ecc2d86151b63c214b0eba55fe8730b48d248969ecc2d86151b63c214b0eba55fe8730b48d248969ecc2d86151b63c214b0eba55fe8730b48d248969ecc2d86151b63c214b0eba55fe8730b48d248969ecc2d86151b63c214b0eba55fe8730b48d248969ecc2d86151b63c214b0eba55fe8730b48d248969ecc2d86151b63c214b0eba55fe8730b48d248969ecc2d86151b63c214b0eba55fe8730b48d248969ecc2d86151b63c214b0eba55fe8730b48d248969ecc2d86151b63c214b0eba55fe8730b48d248969ecc2d86151b63c214b0eba55fe
Certificate-Based Token Verification

Certificate-Based Token

Token Data

Certificate

Proxy

Resource

Actions

Signature
Caching Certificate-Based Tokens

Client -> Proxy -> Server

hash(metadata) -> LRU Creation Cache

hash($$) -> LRU Validation Cache

metadata
Tradeoffs with Cert-Based Tokens

Pros
- Reliable
- Simple
- Generic

Cons
- Large
- Public-Key
- x509
A Symmetric-Key Variant

(analogous to Kerberos)

Client → Proxy → Server

Key Server

All direct communications are encrypted / authenticated with TLS
"Crypto Auth Tokens" (CATs)

Server

Proxy

Client

Login Server

Key Server

$ = MAC(session key, request) || client + "info"

session key = PRF(service key, "client" + info)

service key = PRF(master key, "service" + info)

All direct communications are encrypted / authenticated with TLS
Summary

1. We build from a small root of trust

2. TLS by itself isn’t enough

3. Tokens
   - Public-Key
   - Symmetric-Key
Acknowledgments