

Trust No Token: Hunting Identity-Based Attacks Beyond the IdP

Dr Nestori Syynimaa (@DrAzureAD)



Contents

- How tokens are issued, used and abused
- Stealing the token
- Beyond the IdP: detecting replay attacks



Who am I?

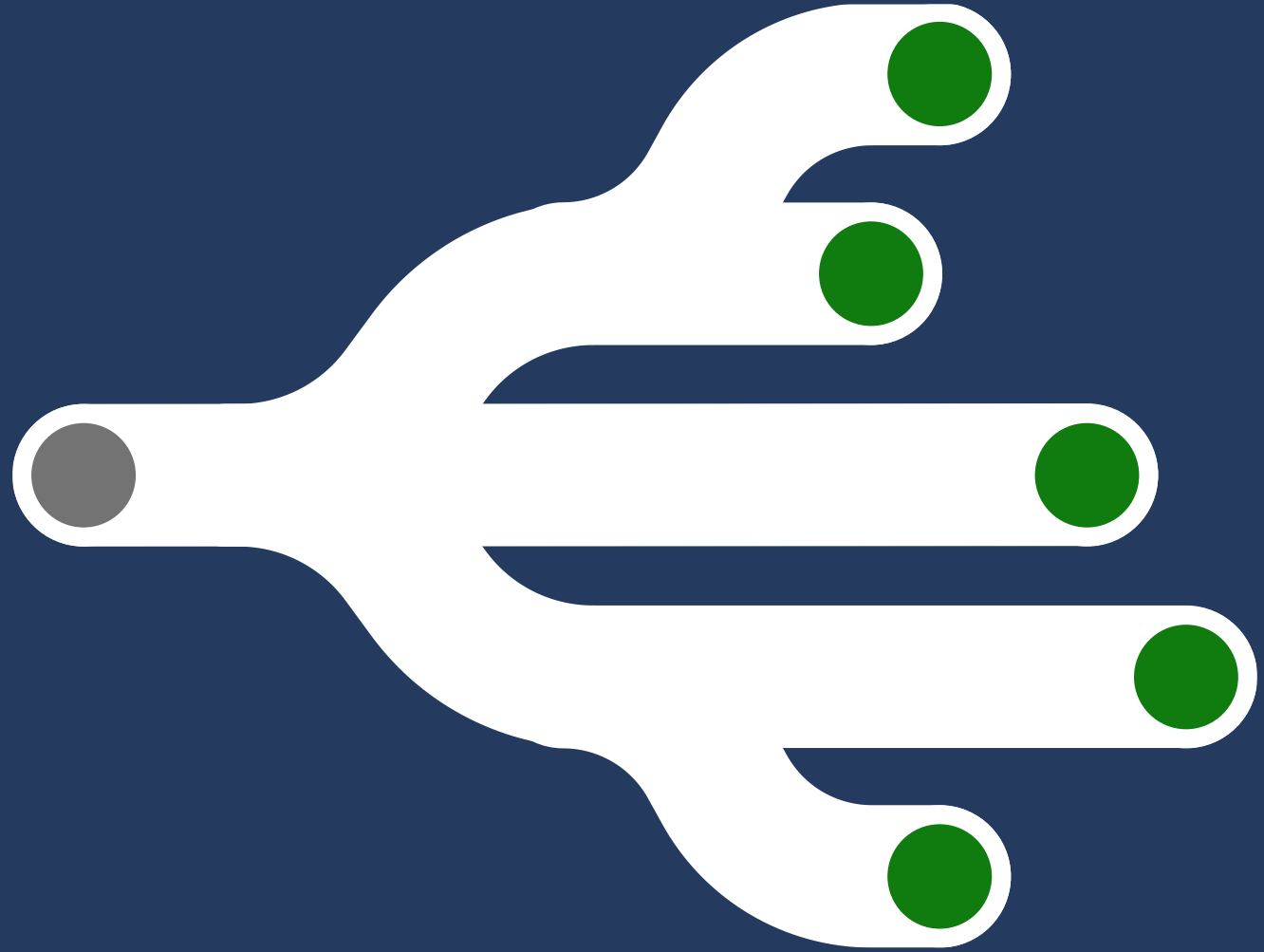
- Dr Nestori Syynimaa (DrAzureAD)
- Principal Identity Security Researcher
- Microsoft Security Research (MSecR)

nsyynimaa@microsoft.com

@DrAzureAD



Tokens and trust



Who plays which role



User

- Consumes services



Service Provider
(SP)

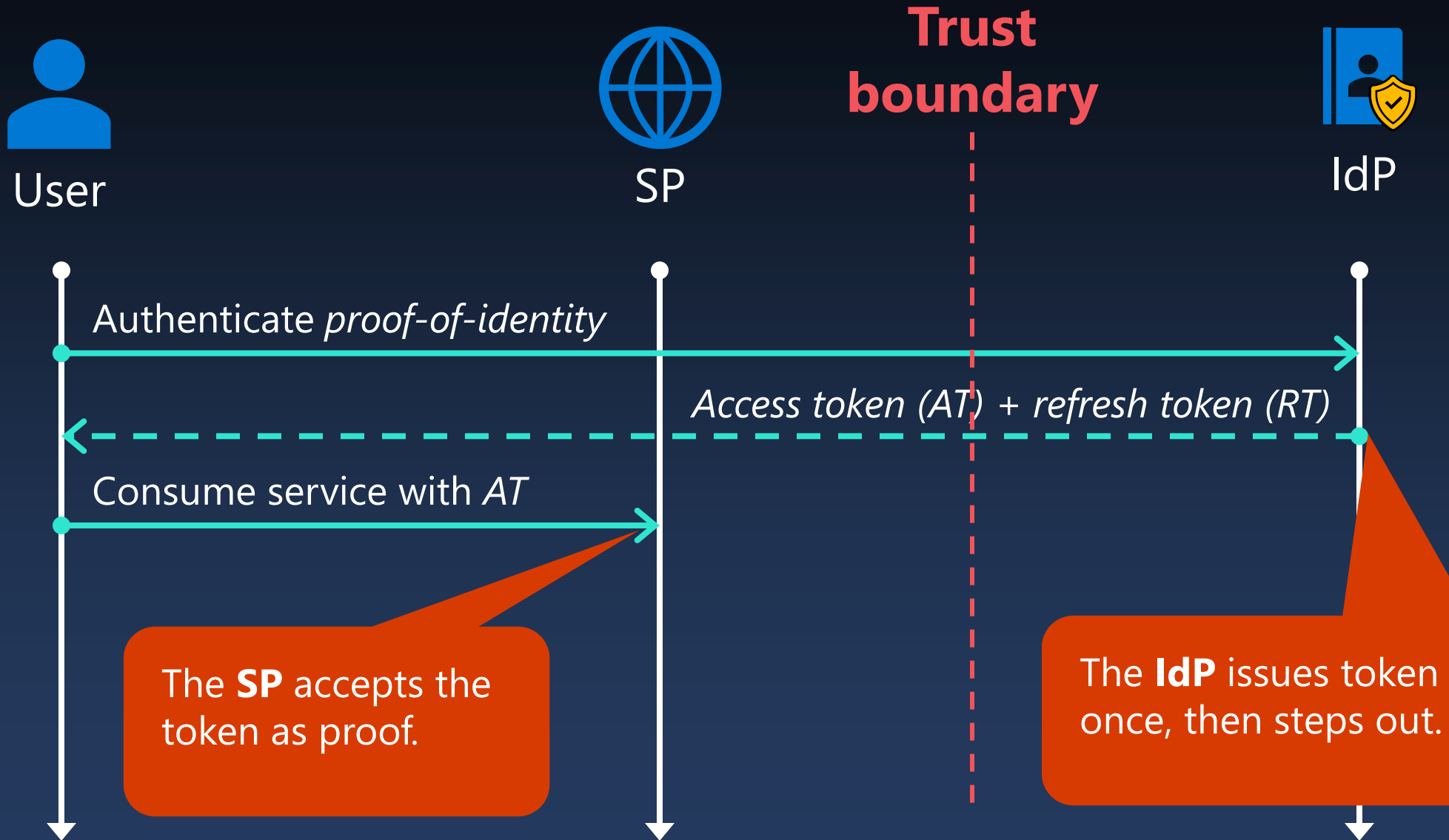
- Provides services



Identity Provider
(IdP)

- Provides identity and access management

How a token is issued and used



What each side can actually see



User



SP



IdP

Authenticate *proof-of-identity*

Access token (AT) + refresh token (RT)

Consume service with AT

Blind spot: everything between the two

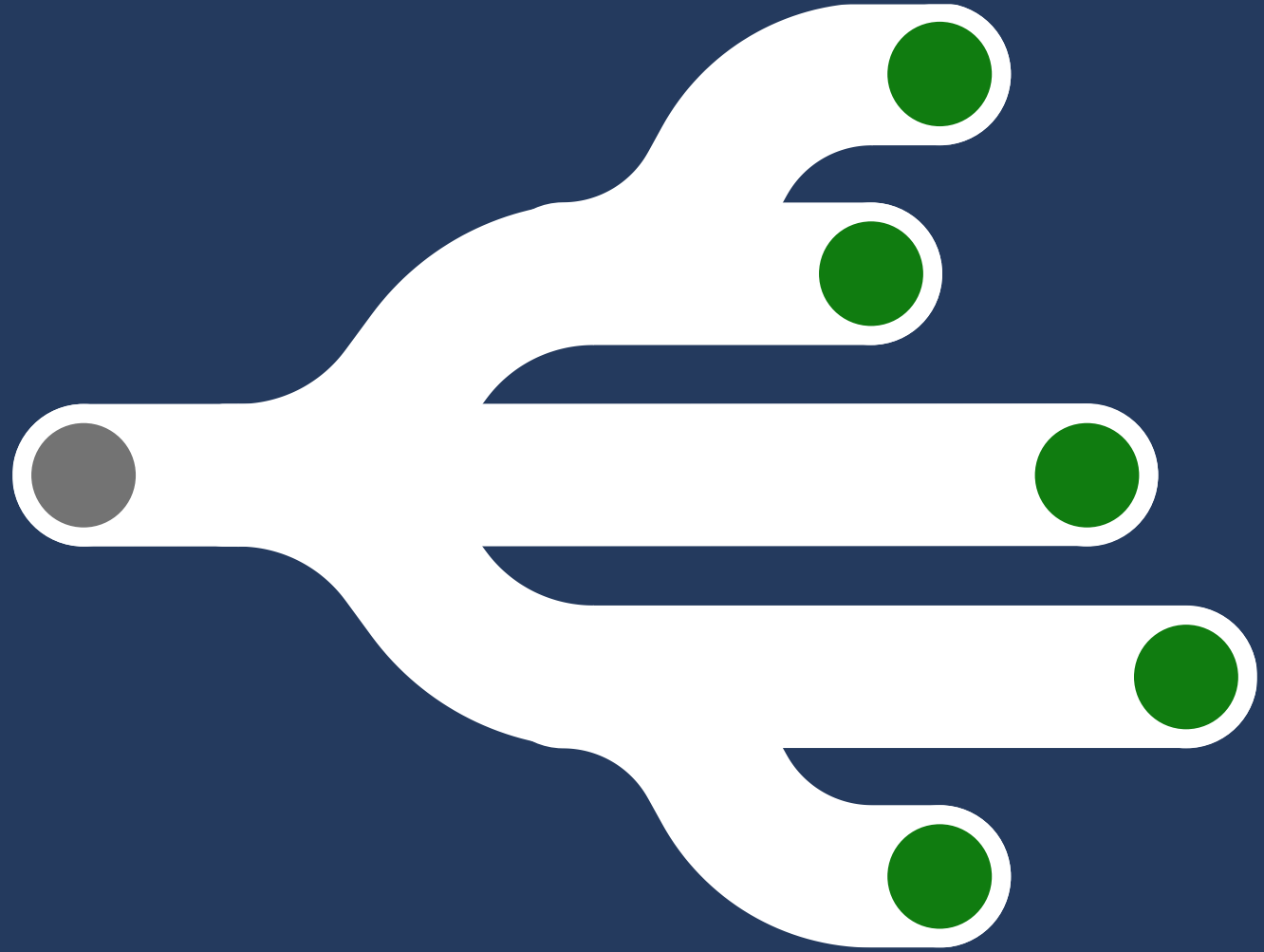
SP view: what the user did - data read, actions taken, client used, etc.

IdP view: who signed in, for which app, how and from where

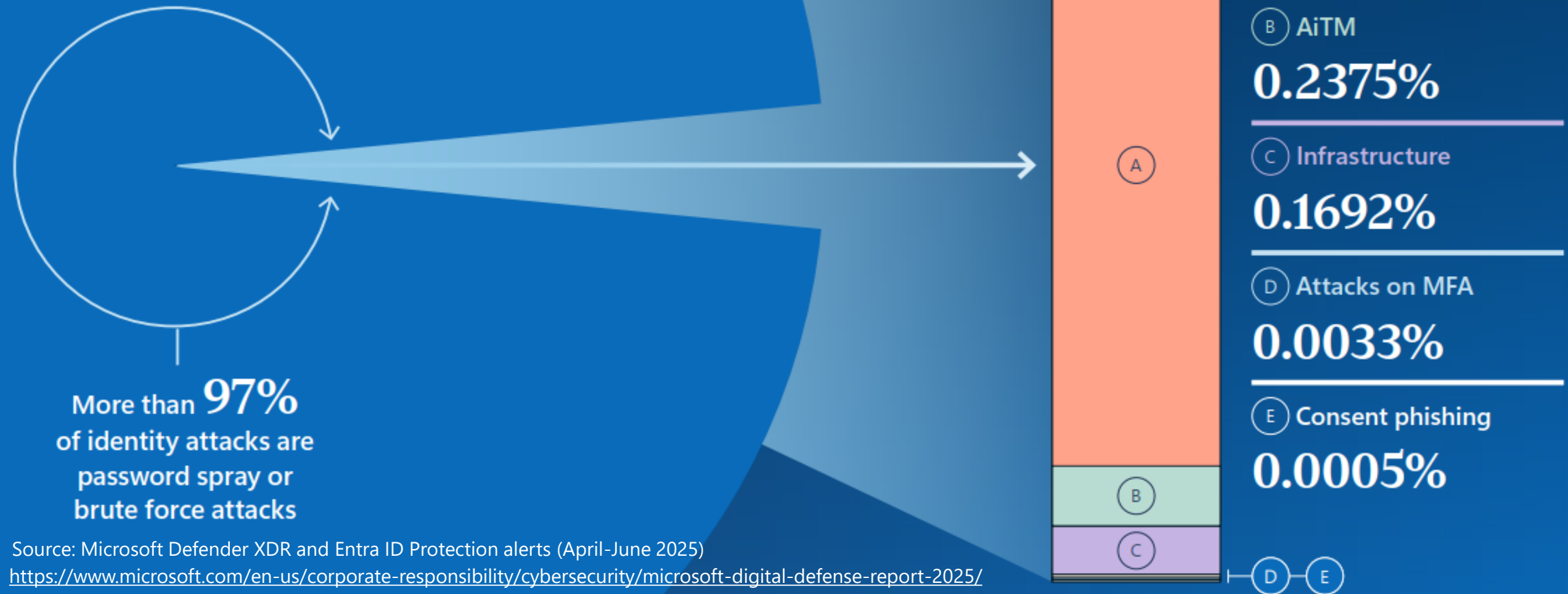
Demo

Once a token is issued, the attacker needs neither the password nor the second factor

Stealing the token



Where token theft sits in the attack mix



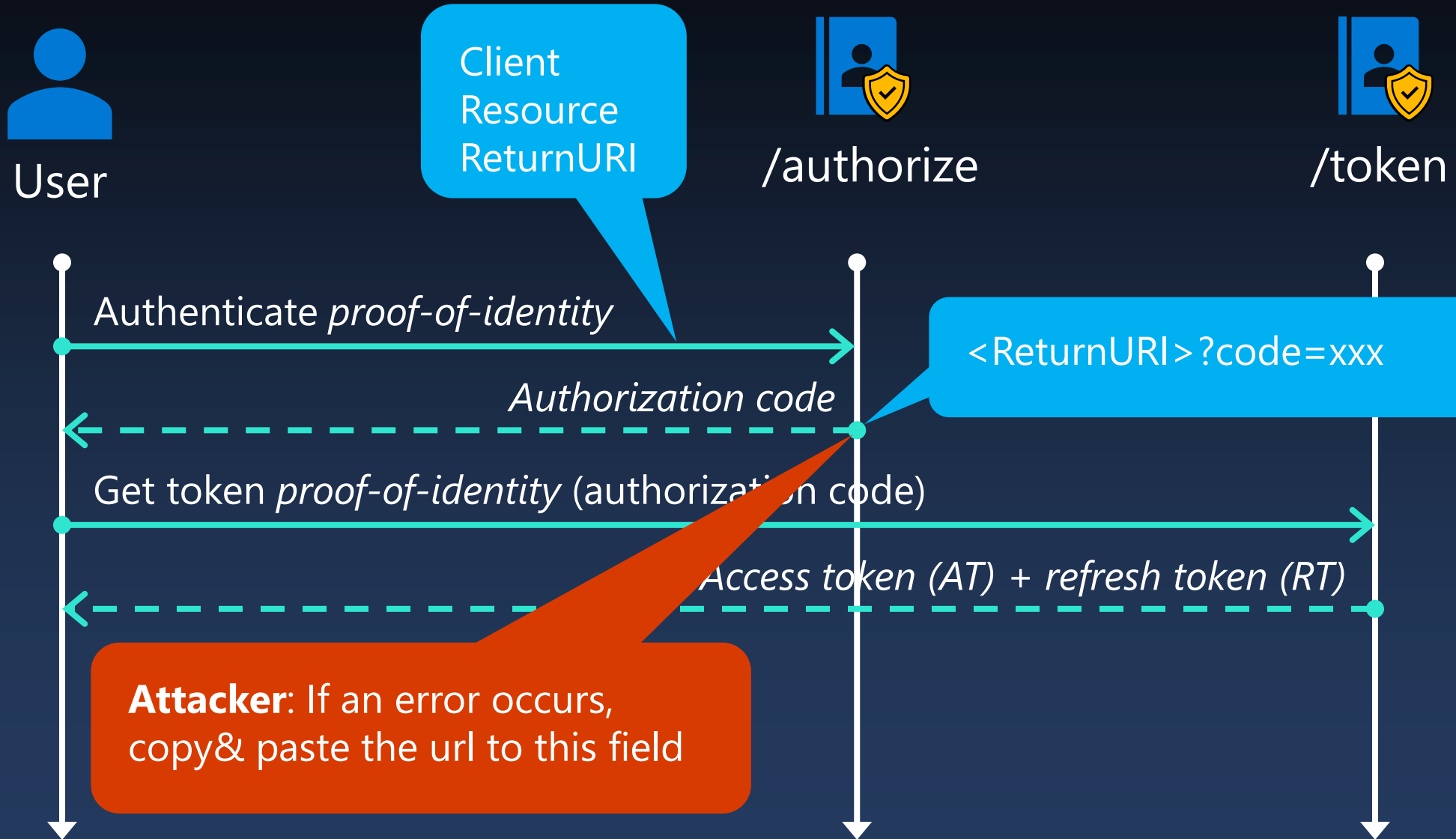
~~Man-in-the-Middle (MitM)~~

Adversary-in-the-Middle (AitM)

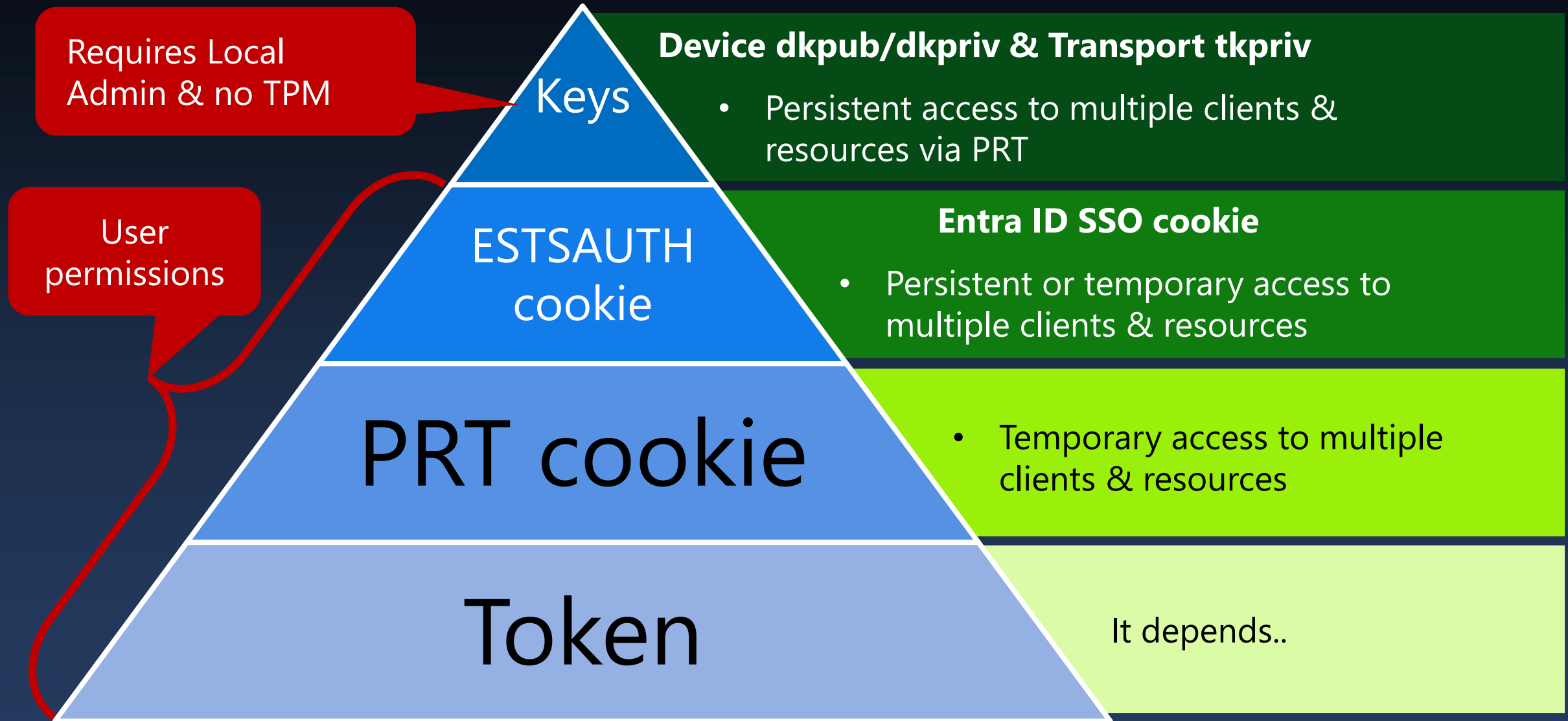
- An attack where the **adversary positions** himself **in between** the **user** and the **system** so that he can intercept and alter data traveling between them.¹



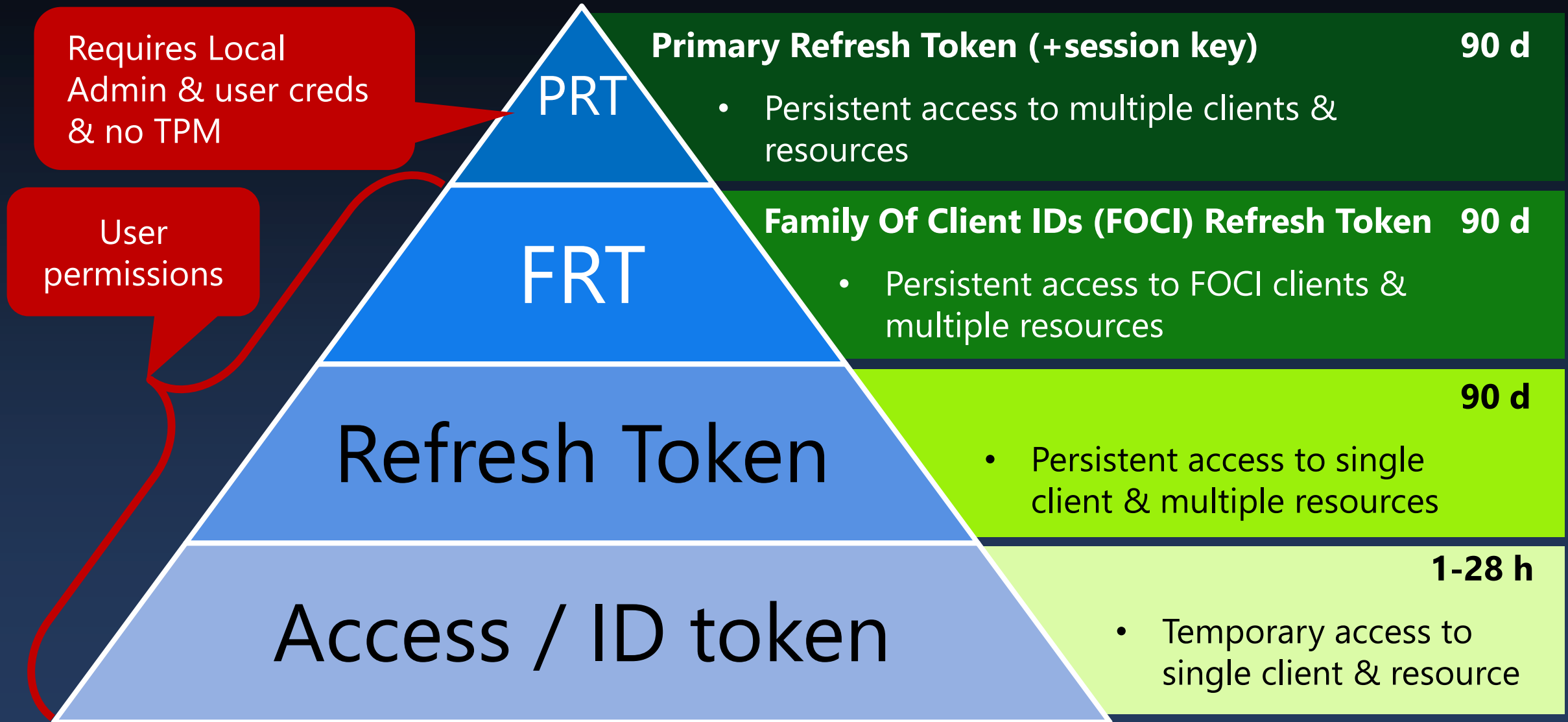
AuthCodeFix (ConsentFix) attack



What sits on a compromised endpoint



Which tokens are worth stealing

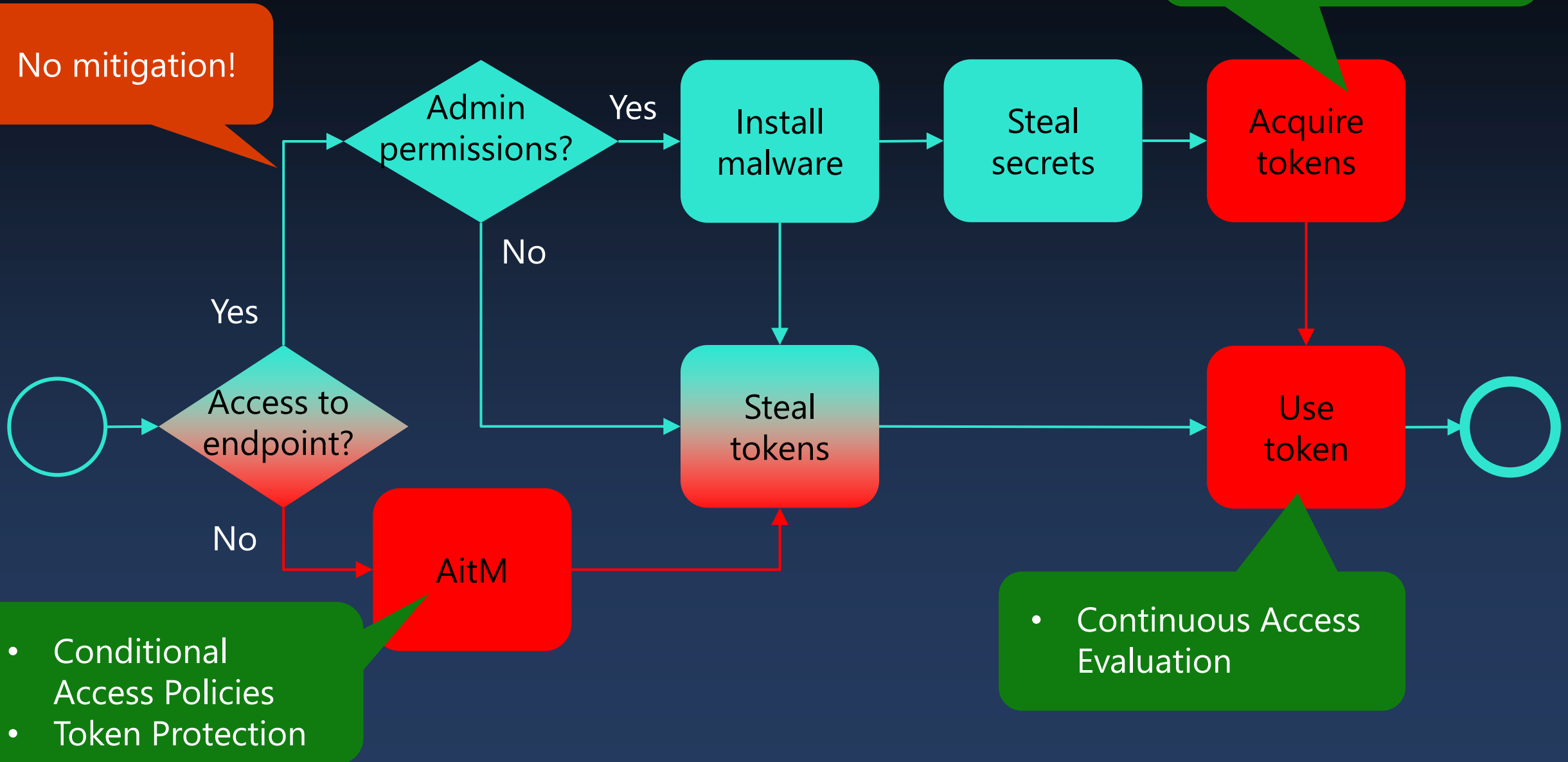


Token security best practices

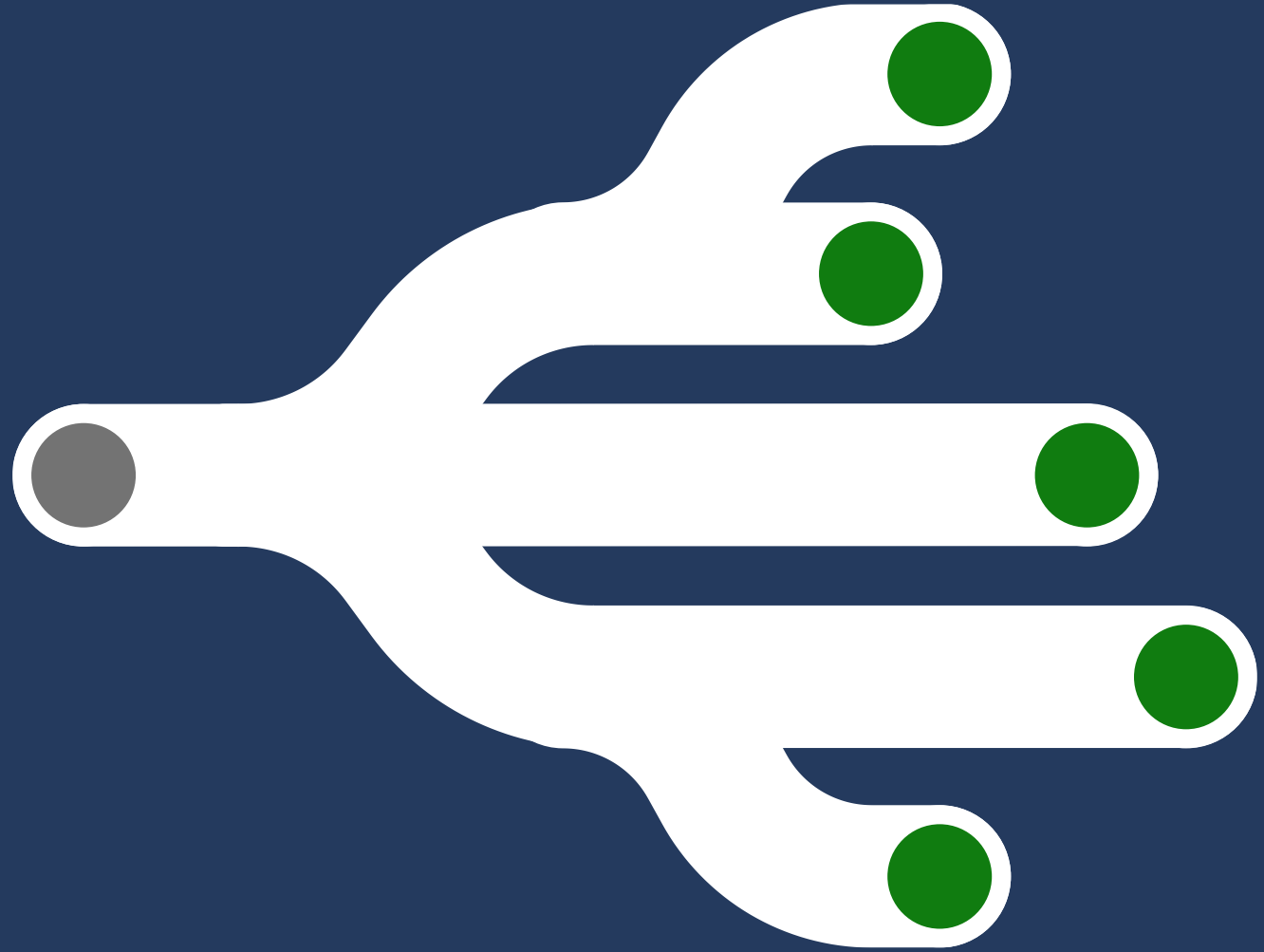
- **Conditional Access Policies:** Use Conditional Access policies to enforce compliant network checks. **This ensures that tokens are only used from trusted networks and devices.**
- **Token Binding:** Implement **Token Protection** (formerly known as token binding) to cryptographically tie tokens to client secrets. **This prevents token replay attacks from different devices.**
- **Continuous Access Evaluation (CAE):** Implement CAE to **continuously evaluate the security state of the session**. This helps in detecting and revoking tokens if there are changes in the user's security posture, such as network location changes.

<https://techcommunity.microsoft.com/blog/coreinfrastructureandsecurityblog/token-protection-by-using-microsoft-entra-id-/4302207>

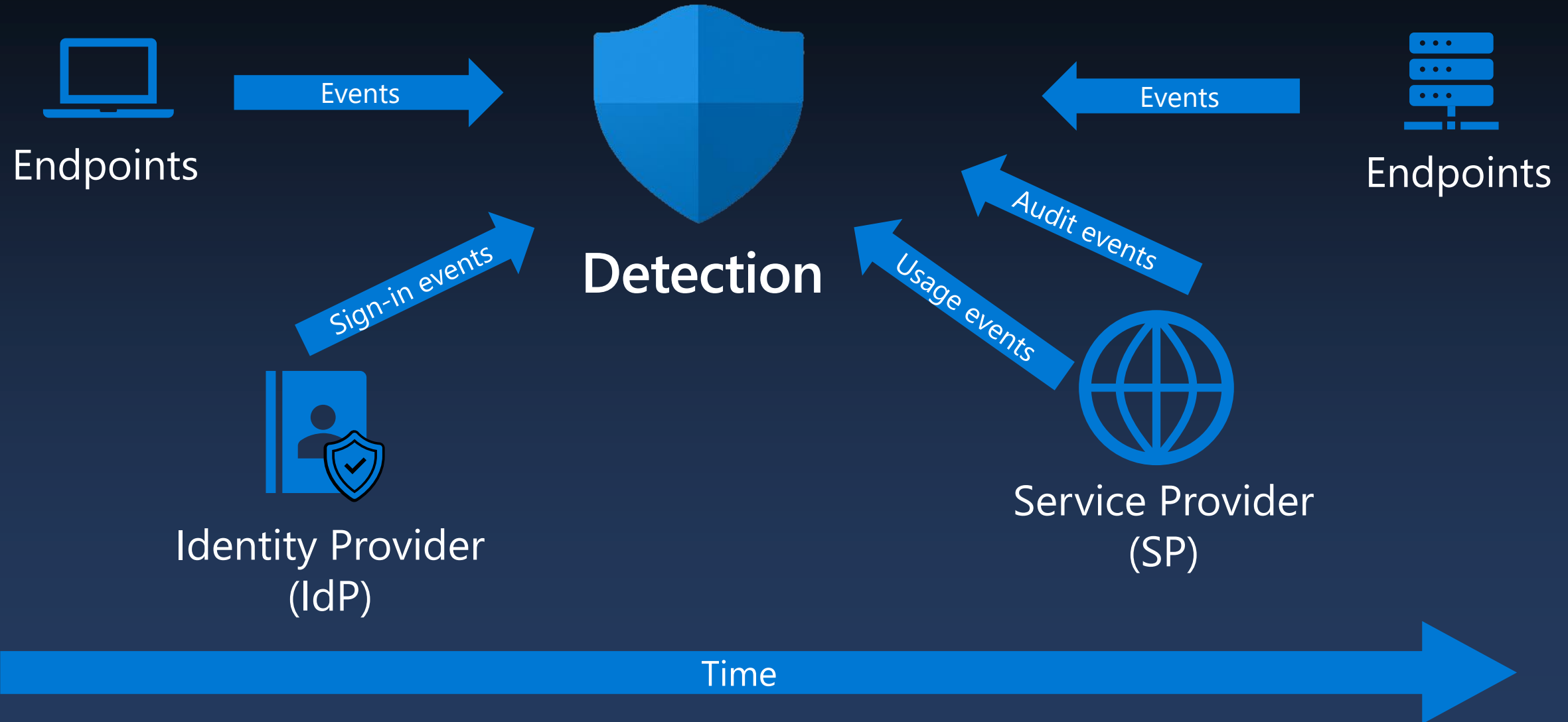
Token security best practices coverage



Beyond the IdP



Where the telemetry comes from

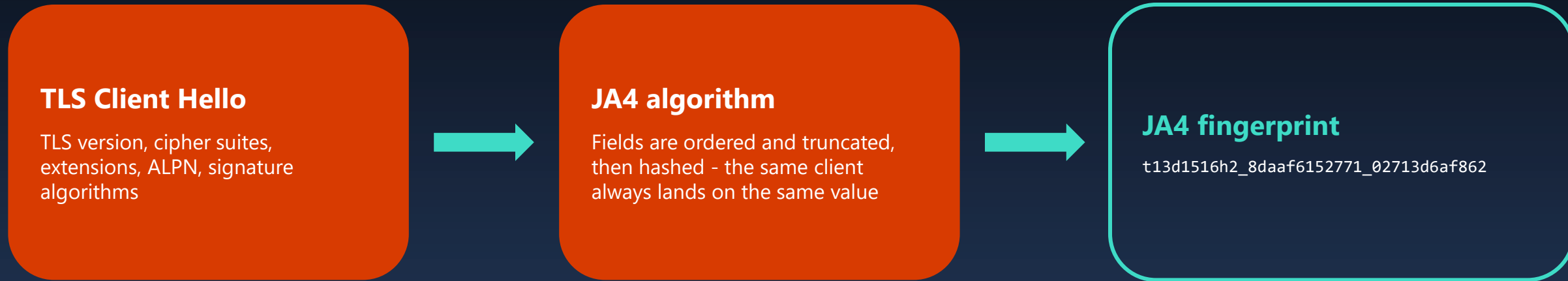


Why IdP telemetry alone fails

- The **IdP logs issuance, not use** - it never sees the token again
- A replayed token produces no failed sign-in and no MFA prompt
- Location and user-agent are attacker-influenced hints, not facts
- **Note:** Replayed refresh tokens and session cookies generate (non)interactive sign-in events 😊

A clean sign-in log is not evidence that the session is clean

JA4: a fingerprint of the client, not the user



Same client software, same fingerprint. Different tooling, different fingerprint - even when the token, the user and the IP are identical.

Current JA4 methods

Full Name	Short Name	Description
JA4	JA4	TLS Client Fingerprinting
JA4Server	JA4S	TLS Server Response / Session Fingerprinting
JA4HTTP	JA4H	HTTP Client Fingerprinting
JA4Latency	JA4L	Client to Server Latency Measurement / Light Distance
JA4LatencyServer	JA4LS	Server to Client Latency Measurement / Light Distance
JA4X509	JA4X	X509 TLS Certificate Fingerprinting
JA4SSH	JA4SSH	SSH Traffic Fingerprinting
JA4TCP	JA4T	TCP Client Fingerprinting
JA4TCPServer	JA4TS	TCP Server Response Fingerprinting
JA4TCPScan	JA4TScan	Active TCP Fingerprint Scanner
JA4DHCP	JA4D	DHCP Fingerprinting
JA4DHCPv6	JA4D6	DHCPv6 Fingerprinting
JA4NTP	JA4N	NTP Fingerprinting
JA4Scan-TLS	JA4Scan-TLS	Active TLS Server Fingerprint Scanner
JA4Scan-QUIC	JA4Scan-QUIC	Active QUIC Server Fingerprint Scanner

Reading a fingerprint in detection

- Strength: it **describes the software making the request**, not a string the client chooses to report.
- Replay signal: one token, two materially different TLS stacks, inside one session window.
- Limit: it **identifies a client family**, not a person - corroboration, never a verdict.

Attacker tooling often has a fingerprint no legitimate corporate client has

Correlation: anchor on the token

- Join sign-in and usage events by session and/or token identifier
- **Contradiction:** issuance and usage contexts disagree - network, ASN, fingerprint, timing
- **Rank by impact:** mail rules, mass download, app consent and role changes first.

One rule to start with: same token, two fingerprints, inside one hour

What to take home

- **Assume tokens leak** - build detection for the period after issuance
- **Collect both sides:** sign-in logs alone cannot answer whether a token was replayed
- Bind/block what you can, correlate the rest - controls shrink the window, correlation closes it
- **Trust no token:** treat every bearer as a claim to be re-checked against behaviour/facts
- Collect at least:
 - Client IP, user-agent, unique token identifier (uti), JA4 (the more the better)

Kiitos!

Questions?

