LDAPCON 2026 · ENTERPRISE CASE STUDY

Direct Integration of 389 Directory Server with PrivacyIDEA

one bind · two factors

S3cret 123456

password → 389-ds
OTP → PrivacyIDEA

Sangjun Song
Seojindsa Co., Ltd. / Gachon University

Thu 8 Oct 2026 · 11:45–12:30 · Museum, Uhlandsaal

PART 0 · OPENING

Context & agenda

Speaker context

- Directory & IAM engineering
- Production 389-ds at multinational scale

1. Why MFA at the LDAP bind is hard
2. The architecture: two-factor split at pre-bind
3. Production: TLS, schema, replication
4. Three engineering problems — and how we solved them
5. Live demo
6. Where engineers and developers go from here

PART 1 · THE PROBLEM

The ask

> *“Add MFA to our LDAP authentication.”*

LDAP simple bind

Single factor by design — a DN and a password is the entire protocol.

Source of truth

The directory backs thousands of applications.

No room to experiment

Change the bind and you change every login.

⚠ **CONFIRM** number of integrated applications

PART 1 · THE PROBLEM

The two usual answers

A Front proxy

- One more hop, one more identity translation, one more thing to run
- LDAP-native apps often cannot speak the proxy's protocol

B Sidecar / PAM chain

- Couples every application to an MFA integration
- N integrations to maintain — inconsistent enforcement

PART 1 · THE PROBLEM

The translation-layer tax

Cost	Why it hurts
Extra network hop	Latency on every authentication, forever
Identity translation	uid ↔ proxy-internal ID; breaks on OU moves
Second user store	Two sources of truth to keep consistent
Sync lag	Directory change → refresh window → inconsistency
New HA surface	The proxy becomes a hard dependency for all logins
Enforcement outside the directory	Different apps, different guarantees

forever

Every row is a permanent operational cost — paid for the life of the system.

PART 1 · THE PROBLEM

The idea, and the tools

Make the directory itself the MFA enforcement point.

Split the credential at the bind. Validate the OTP inline.

389-ds

SLAPI plugin architecture — direct access to the bind path

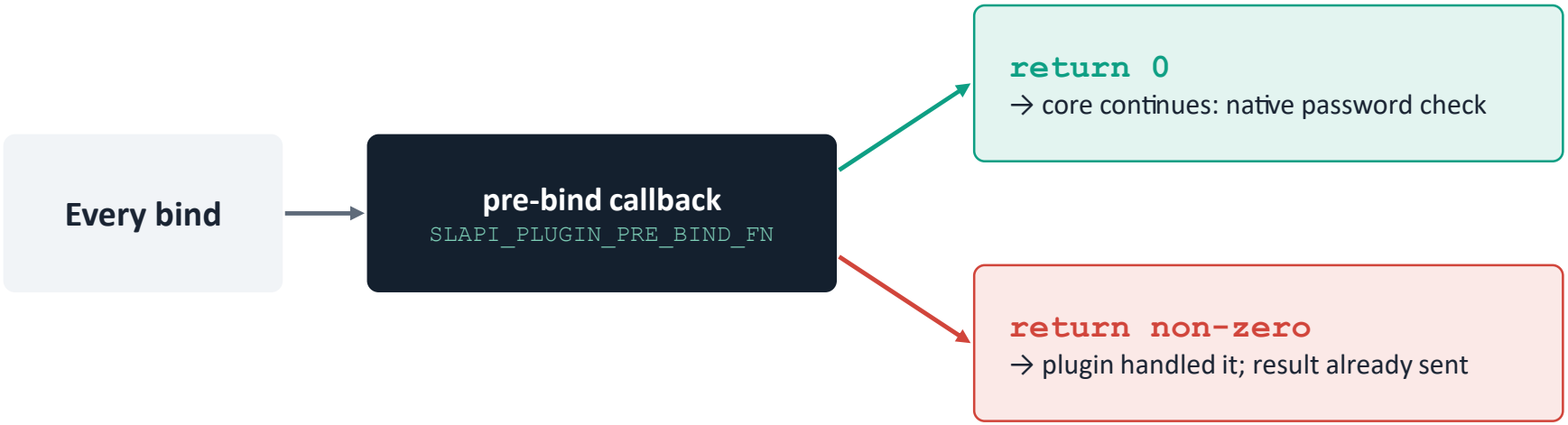
PrivacyIDEA

Open source · REST /validate/check · TOTP / HOTP / push

Both open, both self-hostable → full control, no vendor lock-in

PART 2 · ARCHITECTURE

The SLAPI pre-bind hook



Verified, not guessed: `ldap/servers/slapd/bind.c` — called on every bind; these two branches are the switch the whole design turns on.

PART 2 · ARCHITECTURE

Design: the two-factor split

`pass =` **`<static-password>`** **`<OTP>`**

client appends the OTP — PrivacyIDEA's own convention → no client changes

389-ds · locally
Static password validated exactly as it always was

PrivacyIDEA · one REST call
Only the OTP leaves the node

"Direct" = no identity translation, and **the password never leaves the directory.**

PART 2 · ARCHITECTURE

Flow

OTP ok → cred truncated → native password check

OTP bad → invalidCredentials, bind ends here

PART 2 · ARCHITECTURE

The elegant part: in-place truncation

- `SLAPI_BIND_CREDENTIALS` points at the core's own berval
- **No malloc. No copy.** The core frees the original buffer — we only shortened it
- Every bind goes through this path

0

allocations on the hot path

PART 2 · ARCHITECTURE

Reusing what's already in the tree

Pre-bind interception	→	`plugins/pam_passthru/`
HTTPS client	→	`slapd/hibp_client.c` libcurl · PR_CallOnce · TLS verify
JSON	→	`json-c`

0

new dependencies

libcurl and json-c are already 389-ds build dependencies

Nothing invented: every pattern has in-tree precedent.

PART 2 · ARCHITECTURE

Configuration

Lives in the plugin's own entry: `cn=PrivacyIDEA PassThru,cn=plugins,cn=config`

Endpoint	`piURL`	`piRealm`	`piOTPLength`	`piTimeout`
TLS & policy	`piVerifyPeer`	`piCACert`	`piSecure`	`piFallback`
Mapping & scope	`piMapMethod`	`piMapAttr`	`piIncludeSuffix`	`piExcludeSuffix`

Live reload via DSE callback

Change a value → it takes effect. No directory restart during an MFA rollout.

PART 2 · ARCHITECTURE

Fail-closed by default

Case	Result
OTP ok + password ok	**bind success**
OTP ok + password bad	**invalidCredentials (native)**
OTP bad	**invalidCredentials (plugin)**
PI unreachable · piFallback: off	**unavailable — FAIL-CLOSED**
PI unreachable · piFallback: on	**password-only (availability)**
non-TLS · piSecure: on	**confidentialityRequired**

⚖ *piFallback: on is not a convenience switch — it is availability chosen over security, written where an auditor can find it.*

PART 2 · ARCHITECTURE

Thread-safety, performance, security posture

Concurrency

every bind runs on a worker thread

- Config: rwlock + deep-copy-at-call (the memberof pattern)
- One curl handle per request · PR_CallOnce global init
- No new locks, allocations, or internal searches on the per-op path

Security

what the plugin will never do

- OTP buffer wiped after use
- Never intercept root DN / anonymous / SASL binds
- Scope-limited enforcement via pwIncludeSuffix ← remember this

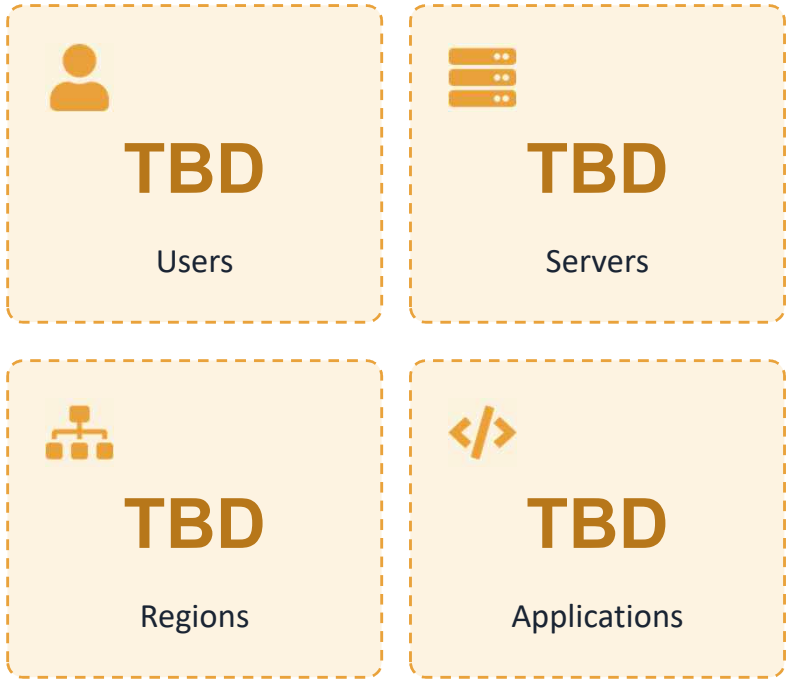
PART 3 · PRODUCTION

Deployment context

Samsung Electronics R&D Institute, India

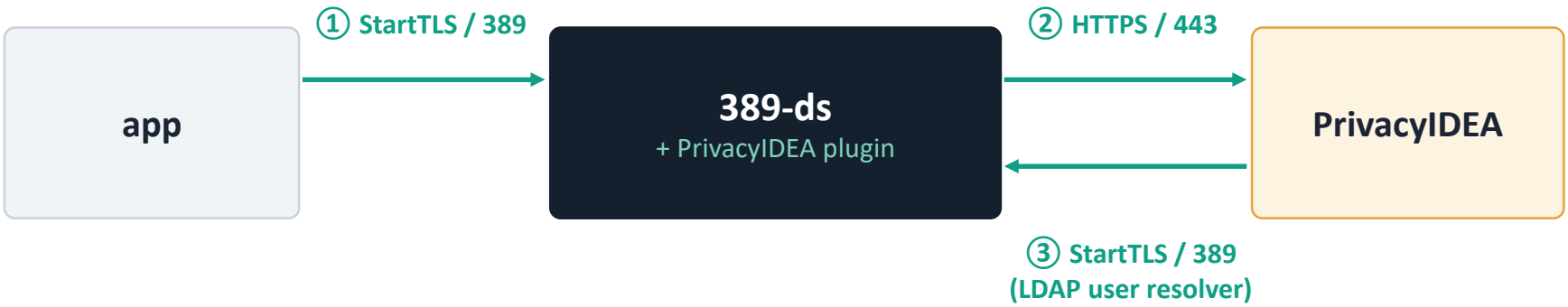
- 389-ds as the enterprise identity backbone
- Requirement: enforce MFA at the directory — low latency, high throughput

The password check never leaves the node → the only remote call is the OTP.



PART 3 · PRODUCTION

All-TLS topology



One internal CA signs the server certificate shared by LDAP and HTTPS → **one trust chain to manage, not three.**

PART 3 · PRODUCTION

Why StartTLS on 389, not LDAPS on 636

One standard port
389 + StartTLS negotiation, no separate LDAPS port → simpler firewalls and client config

`nsslapd-require-secure-binds: on`
The server rejects plaintext authentication

`piSecure: on`
The plugin processes an OTP only over a secured connection

StartTLS counts as secure
Recognised by the plugin via SLAPI_CONN_IS_SSL_SESSION

Same guarantee as a dedicated TLS port — one fewer thing to configure everywhere.

PART 3 · PRODUCTION

Schema customization

Our user entries are not inetOrgPerson

```
objectClass: nsPerson       # structural
objectClass: nsAccount      # auxiliary
objectClass: nsOrgPerson    # auxiliary
objectClass: posixAccount   # auxiliary
```

nsPerson has no sn, no givenName
it uses cn, displayName, legalName — stock presets silently leave fields empty

PrivacyIDEA LDAP resolver mapping

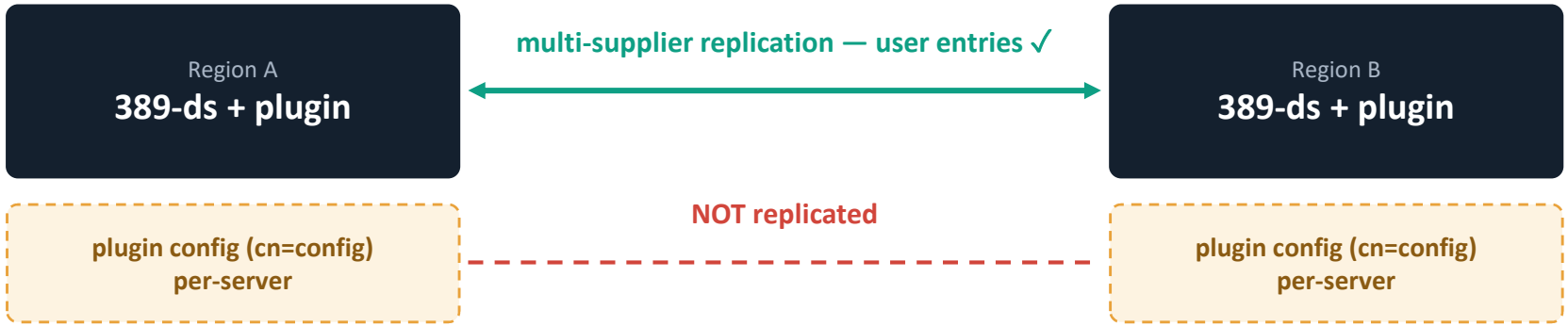
```
{
  "username":   "uid",
  "givenname":  "displayName",
  "surname":    "legalName",
  "email":      "mail"
}
```

Pick the user ↔ token unique ID before issuing any token — changing it later means re-enrolling everyone.

⚠ **CONFIRM** UID Type (nsUniqueId vs entryUUID) · any custom attributes defined

PART 3 · PRODUCTION

Replication & configuration drift



- User entries replicate normally — MFA changes nothing here
- **cn=config does not replicate** → plugin config is per-server
- **Drift raises no error** — a drifted node just authenticates differently
- Managed by config automation + an explicit verification check
- Password check stays node-local → the OTP is the only round-trip

⚠ **CONFIRM** actual topology (suppliers / regions) · tuning items (changelog trimming, nsslapd-idletimeout…)

PART 3 · PRODUCTION

Operational lessons

NTP sync is non-negotiable
TOTP is time-based — clock drift looks exactly like a wrong code

Back up the PrivacyIDEA enckey
Lose it and you lose every enrolled token

Decide piFallback deliberately
Security vs availability — in writing

Exclude service accounts
via piExcludeSuffix

Certificate renewal touches more than TLS
Attribute-encryption keys are wrapped with the server cert's key — put renewal in the runbook

PART 4 · THREE ENGINEERING PROBLEMS

Three failures, three layers

Writing the code was not the interesting part. **Debugging was.**

1

A C language rule

compiles fine, won't load

2

A vendor API divergence

exactly one function differed

3

A configuration value

it “works” — and that is the bug

Each one diagnosed from a single log line.

PART 4 · THREE ENGINEERING PROBLEMS

Bug 1: compiles fine, won't load

```
- ERR - symload_report_error - Netscape Portable Runtime error -5975:
  .../libpi-passthru-plugin.so: undefined symbol: pi_passthruauth_init
- ERR - load_plugin_entry - Unable to load plugin "cn=PrivacyIDEA PassThru,cn=plugins,cn=config"
Error: Failed to start DS
```

Cause

- A **`static`** forward declaration → the whole function got internal linkage
- The init symbol is resolved by name with **`dlsym()`** → it must be external
- **The directory refused to start.** A cn=config plugin entry is a startup dependency

Diagnose

```
nm -D --defined-only $SO | grep passthru # T = exported
nm $SO | grep passthru                   # t = static
nm -D $SO | c++filt | grep passthru      # mangled = C++
```

Recover — server is down, dse.ldif is the only way back

```
cp -a $DSE $DSE.bak          # always, first
# nsslapd-pluginEnabled: on → off  (stopped)
dsctl $INST start
```

Rule adopted: run nm -D before every install — no startup failures since.

PART 4 · THREE ENGINEERING PROBLEMS

Bug 2: the vendor fork's different API

```
warning: implicit declaration of function 'slapi_log_err'
         did you mean 'slapi_log_error'?
```

Target

A vendor 389-ds fork carrying the legacy logging API

Scope

Every other SLAPI call identical — exactly one function differed

Fix

Compatibility macro behind a build flag
`-DPI_LEGACY_LOG`

One compiler warning said: you are not on the platform you think. Writing for a fleet = writing for more than one 389-ds.

PART 4 · THREE ENGINEERING PROBLEMS

Bug 3: it “works” — and that is the bug

- **A password-only bind succeeded.** MFA was silently bypassed.
- Not a code bug: `piIncludeSuffix` pointed at the wrong base
- The scope logic was correct. The value was wrong → out of scope → pass through
- Nothing failed. Nothing logged an error. It just quietly did not enforce.

configured piIncludeSuffix

```
dc=seojindsa,dc=kr
```

real suffix

```
dc=cae,dc=com
```

An MFA control that silently passes is more dangerous than one that errors.

PART 4 · THREE ENGINEERING PROBLEMS

How this was built

Written with an **AI pair-programmer, in one sitting** — not boilerplate: navigating a 2,300-file C server.

Found the references
pam_passthru for the pattern · hibp_client.c for HTTPS · existing build deps

Verified contracts from source
pre-bind return convention · berval ownership · result idempotency

Wrote thread-safe C
matching in-tree idioms; every SLAPI symbol checked against the header

PART 4 · THREE ENGINEERING PROBLEMS

What the machine did, and what it didn't

AI contributed	The human still owned
Code archaeology across the tree	Security posture and threat model
Pattern selection from in-tree precedent	“Is fail-open acceptable here?”
Thread-safe C matching house idioms	End-to-end verification on a real server
Diagnosing bugs 1 and 2 from a log line	**Recognizing that bug 3 was a bug at all**

Bug 3 passed every test that asked “does it work?”
It failed the only question that mattered: **“should this have passed?”**

PART 5 · LIVE DEMO

Demo setup

2021dsldap01
389-ds + PrivacyIDEA on one RHEL 9.6 box — all TLS

demo_user
under ou=people,dc=cae,dc=com

TOTP token enrolled
otppin=none · static password known

PART 5 · LIVE DEMO

Demo

```
# 1) password only → REJECTED  (MFA is actually enforced)
ldapwhoami -x -ZZ -H ldap://2021dsldap01.seojindsa.kr \
  -D "uid=demo_user,ou=people,dc=cae,dc=com" -w "test123"
#   → Invalid credentials (49)

# 2) password + current OTP → SUCCESS
ldapwhoami -x -ZZ -H ldap://2021dsldap01.seojindsa.kr \
  -D "uid=demo_user,ou=people,dc=cae,dc=com" -w "test123<OTP>"
#   → dn: uid=demo_user,ou=people,dc=cae,dc=com

# 3) the log: OTP validated → password delegated to the core
tail -n5 /YullinDS/var/log/dirsrv/slapd-2021dsldap01/errors | grep pi_passthru
```

PART 5 · LIVE DEMO

What just happened

1	2	0
bind	factors	extra hops for the password

OTP validated inline, over TLS
one REST call from inside the bind

Enforcement lives inside the directory
consistent for every LDAP application

0 lines
of application code changed

⚠ **CONFIRM** added bind latency p50 / p99 (ms)

PART 6 · CLOSING

Takeaways

1

Put authentication where authentication already happens.

Every layer you insert is a cost you pay forever.

2

Reuse the platform's own patterns.

pam_passthru, hibp_client.c, existing deps — zero new dependencies.

3

Fail-closed by default.

Make the availability trade-off explicit and written down.

4

A control that silently passes is worse than one that errors.

Test the negative case.

5

A plugin failure is a directory failure.

Know your dse.ldif recovery path. Run nm -D before every install.

6

AI is a force-multiplier for systems C — especially debugging.

Security judgement is still yours.

PART 6 · CLOSING

Where we go from here

Everything in this talk — the plugin, the docs, these slides, the lab videos — **was built with Claude.**

Directory engineers

from operating to owning what must be true

- Own the invariants — what must never succeed
- Turn runbooks into executable checks: negative binds, drift
- Go deeper into internals — it is how you review the machine
- Widen scope: directory → DID / VC identity

Developers

from writing code to specifying and proving it

- Write the spec first: constraints and failure modes
- Verify contracts in the source, not in the answer
- Gate on evidence: tests, nm -D, a real bind
- Build agent auth: delegation, not impersonation

The next principal may not be a person `#DPR` `#DID` `#VC` `#AI_agent_auth`

DPR — Digital Product Rights: `DRM (content)` → `DPP (manufacturing)` → `DPR (products)` · self-sovereign proof of rights to any product

The machine writes faster. ***Deciding what must be true is still our job.***

Thank you

Questions?

- Enterprise MFA at the LDAP bind — direct, TLS end-to-end, low latency
- Reproducible: plugin, configuration, and deployment documented

⚠ **CONFIRM** repository URL (large, if published) · contact

Sangjun Song
Seojindsa Co., Ltd. / Gachon University