Consultingwerk
software architecture and development

Managing Secrets in OpenEdge Configurations

PUG Challenge 2026 Amsterdam

Peter Judge, Consultingwerk
peter.judge@consultingwerk.com

Peter Judge



Writing 4GL since 1996, working on a variety of frameworks and applications

Focusing on integration with OpenEdge applications, PASOE, software architecture and web technologies

Active member of the OpenEdge community and speaker at international conferences

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Consultingwerk
software architecture and development

Our Focus: Modernizing OpenEdge Applications



Modernization of Legacy OpenEdge Applications



Deep Technical Expertise



Global IT Partner with Local Presence



More than Consulting – We Deliver Tools & Solutions

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

12.8.11: The Update That Broke Monday Morning

12.8.11

OECH1 removed
(and 12.2.19).
Old values now error at runtime

CVSS 9.1

CVE-2025-8095
Critical
CWE-257: recoverable format

Change it

Re-encoding alone
is not enough:
the secret must change

Progress: OECH1-encoded values "must be considered compromised"

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Quiz: Which One Is Safe?

Four ways a secret shows up in OpenEdge config:

```
1   -P oech1::9f3c...          # in a .pf file
2   -P ae2h6::6f24c4...        # in a .pf file
3   password: czNjcjN0LWRlbW8=    # base64 in a config
4   <user password="7055cd1a...$100000$07d5e0..."/>  # tomcat-users.xml
```

Which one is actually safe to store? Have a guess before I click.

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Quiz: Which One Is Safe?

Four ways a secret shows up in OpenEdge config:

```
1   -P oech1::9f3c...          # in a .pf file
2   -P ae2h6::6f24c4...        # in a .pf file
3   password: czNjcjN0LWRlbW8=    # base64 in a config
4   <user password="7055cd1a...$100000$07d5e0..."/>  # tomcat-users.xml
```

1-3 are reversible. 4 is a one-way hash - the only one you can safely store.

Only 4 is safe - a one-way hash used to VERIFY a login, never to supply one
...yet the tomcat-users.xml PASOE 13.0 actually ships says password="tomcat", in clear text

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

What You Will Take Home



Know the words

Encoding vs encryption

vs hashing

The distinction that matters



Find your secrets

Where OpenEdge keeps them

What each place supports

What to grep for first



Move them out

Env vars, mounted files

Vaults and the SCL provider

Rotation, least privilege

No magic tricks, just patterns you can use on Monday

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Encoding Is Not Encryption

and neither is hashing

1. **Encoding is not encryption**
2. Goodbye OECH1
3. Where secrets hide
4. Patterns that hold up
5. Closing

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Three Words, Three Jobs

Encoding

Reversible, no key

Base64, hex

Protects nothing - it is a format

Encryption

Reversible with a key

AES and friends

Only as safe as the key

Hashing

One-way

For verifying a password

You cannot send a hash as -P

Config needs a secret you can USE - so it is encryption, and the question is: who has the key?

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

So What Was OECH1?

What it was for

Progress: "intended to obfuscate values" across the platform

Why it fell

A known cipher, fixed key

Reversible by anyone who has the scheme

The name of the flaw

CWE-257:

storing passwords in a RECOVERABLE format

OECH1 is not a hash. It is reversible - that is the whole problem

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Live Demo 1: Encoding Is Not Protection





Before

- A .pf for a throw-away DB user
- with a base64 "password"
- and an -P oech1:: value

» **Live:**

```
powershell decode of the
base64 value
   -> s3cr3t-demo

copy the .pf to a pre-fix
install (12.2.18):
_progres demo -pf demo.pf
   -> Connected as demo
```



You will see

- Plaintext from base64
- in one command
- The oech1:: value logs in
- on a machine that never
- knew the secret

If OpenEdge can read it, so can whoever copies the file

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Stronger Prefix, Same File?

What a strong prefix buys

It stops:

Shoulder-surfing and casual grep

Password reuse showing up after a leak

Honest people staying honest

What it does not buy

It does not stop:

Anyone with the file AND an OpenEdge install

The encoded value IS the password to -P

A copied config from being usable

Encoding buys time; file permissions and rotation buy safety

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Goodbye OECH1

What changed in 12.2.19, 12.8.11 and 13.0

1. Encoding is not encryption
2. **Goodbye OECH1**
3. Where secrets hide
4. Patterns that hold up
5. Closing

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Which Versions?

Stream	Vulnerable	OECH1 removed	Ticket
12.2	12.2.0 - 12.2.18	12.2.19	OCTA-80321
12.8	12.8.0 - 12.8.9 (CVE)	12.8.11 (prefix now required)	OCTA-80320
13.0	**-**	**removed; aedh0 added**	**Upgrade guide**

Not deprecated. Removed. Rejected at runtime.

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

What Breaks

Config and .pf
Admin passwords in config and .pf files

DB bootstrap
DB admin passwords that start AVM connections

OEE / OEM
Management admin credentials

Domain Access Codes
DAC for ABL clients and utilities

TLS keystores
Keystore alias credentials

Auth Gateway
OEAG keys and passwords

_User accounts do not use OECH1 and are not affected

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

The Prefix Menu

Prefix	What it is	Strength
`aedh0`	**Derived key (PBKDF2), FIPS-ready. New in 13.0**	**Strong**
`oeph{0-9}`	OpenEdge Password Encoding, PBKDF2	Strong
`ae2{b|h}{1-9}`	Advanced Encoding, 64-bit key block	Strong
`ae1{b|h}{1-9}`	Advanced Encoding, 32-bit	Medium
`nopr0`	Pass-through: as strong as whatever made it	Depends

```
genpassword  (no arguments) lists what your install supports
```

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Before and After: genpassword

```
# before  (12.8.10 and earlier)
-db app -H host -S 5566 -U appuser -P oech1::9f3c...

# after  (13.0)
$ genpassword -prefix aedh0 -password <NEW-secret>
aedh0::7b1e4c...
$ genpassword -password <NEW-secret> -verify aedh0::7b1e4c...
The passwords match.

-db app -H host -S 5566 -U appuser -P aedh0::7b1e4c...
```

A new secret, not the old one re-encoded

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

A Migration Plan That Survives Production



Find	Change	Re-encode	Validate	Test
oech1:: everywhere	the secret itself	stronger prefix	configs + .pf	no OECH1 in logs

oech1tlr (development only)

status | grant | deny - temporarily DECODES existing values while you migrate; never creates new ones; logs each one it meets. Remove the executable before production.

Find with findstr /S /I "oech1::" (Windows) or grep -rIl "oech1::" (Linux)

Also search -P and password= : old genpassword without -prefix wrote bare hex, no oech1::

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Live Demo 2: Hunting OECH1

Before

- A demo config tree
- with two oech1:: values
- (a .pf and sts.properties)

» Live:

```
findstr /S /I "oech1::" *.*
   -> 2 hits

start a 13.0 client
with the old .pf
   -> refused (console only)

genpassword -prefix aedh0
fix, reconnect  -> ok
```

You will see

- The exact error:
- Encoding oech1 not
- supported for -P (21040)
- then a clean start

The migration is not optional - old values are rejected now

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Where Secrets Hide

A tour of the files to grep first

1. Encoding is not encryption
2. Goodbye OECH1
3. **Where secrets hide**
4. Patterns that hold up
5. Closing

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Clients and Databases

Where	What	Encoded value?
.pf / command line	`-P`	Yes, prefix::value
DB / server TLS	`-keyaliaspasswd`	Yes (default: encoded "password")
TDE keystore	`passphrase`	Manual (typed) vs autostart (stored)
SQL Explorer	`-password`	Yes; clear text visible to tools



HSM: the hardware end of "who has the key?"

OpenEdge TDE can add a Hardware Security Module as a second layer of keystore authentication, so the stored autostart secret is not enough on its own.

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

PASOE: Five Files to Check

conf/catalina.properties
psc.as.https.keypass - the keystore password, clear text "password" out of the box

conf/openedge.properties
agentStartupParam (SessMgr) with -U/-P - "not recommended"

conf/tomcat-users.xml
Ships clear text (tomcat/tomcat) - store a hash: digest tool + PBKDF2 CredentialHandler

DLC/properties/pasmgr.properties
Encoded tomcatMgrPassword and webPassword (OpenEdge Management)

bin/*_setenv scripts
Whatever you exported - never put it in openedge_setenv (overwritten on upgrade)

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Authentication Gateway and the Unified Broker

sts.properties (OEAG)

`sts.server.keyfilepassword`

via genpassword

`sts.ldap.context.password`

via stspwdutil -> oeph{n}::

ubroker.properties

`keyAliasPasswd`

(+ keyAlias, keyStorePasswd)

13.0 ships it aedh0:: encoded

By hand: use genpassword

Restart, then prove you can still log in

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Default Passwords Are Still Passwords

Tomcat manager
tomcat / tomcat on every new instance, dev AND prod

-keyaliaspasswd
Default is the encoded value of the word "password"

Shipped TLS passwords
ubroker keyAliasPasswd and psc.as.https.keypassship pre-set

If it shipped in the box, everyone has it

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

What Could Possibly Go Wrong?

28.65M

new secrets on public GitHub in 2025 (+34%)

>64%

of 2022 secrets still valid in Jan 2026

6x

more likely in internal repos than public

8.5%

of 337k Docker Hub images leak secrets

Leaked secrets do not expire unless you rotate them

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

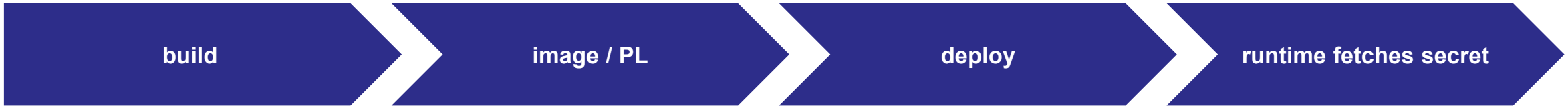
Patterns That Hold Up

Config is for settings; secrets come from outside

1. Encoding is not encryption
2. Goodbye OECH1
3. Where secrets hide
4. **Patterns that hold up**
5. Closing

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Rule Zero: The Secret Is Not Part of the Build



Same artefact everywhere

Test and prod run the identical build

Injected late

Secret arrives at deploy or run time, not before

Different per environment

Each environment gets its own secret

If it is in git, the image or the r-code, it is already shared

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Environment Variables: Handy, Not Hidden

How OpenEdge resolves them

.pf files: no expansion at all

ubroker.properties: ${VAR} and $VAR

PASOE agentStartupParam: ${APP_DB_PW} works

```
ABL: OS-GETENV("APP_DB_PW")
```

Why not to lean on them

Progress (PASOE 13.0): env vars "represent a vulnerability"

Docker: can be "printed in logs"

OWASP: "accessible to all processes"

Better than a password in a config file - but the expanded value lands on the command line, open to any process list

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Secrets as Files

```
/* a protected file, read once at startup */

define variable cPw as longchar no-undo.

copy-lob from file
     "/etc/app/db_password" to cPw.

connect value("-db app -H host -S 5566 "
    + "-U appuser -P "
    + quoter(trim(string(cPw)))) no-error.
```



Why a file beats a -P in config

- Lock it down: owner-read only
- Not inherited by child processes
- Not on any command line
- Not dumped in logs

Read it once at startup, connect, never log the connect string

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

External Secret Stores: It's Just REST

HashiCorp Vault

```
GET /v1/secret/
    data/<path>

X-Vault-Token: ...

-> data.data.<key>
```

Azure Key Vault

```
GET /secrets/<name>
  ?api-version=
   2025-07-01

Bearer <entra>

-> value
```

AWS Secrets Manager

```
GetSecretValue

SecretId: <name>
(SigV4 signed)

-> SecretString
```

The ABL HTTP Client can call all three - the hard part is the first credential

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

A Pluggable Decryption Provider

The shape SCL already has

```
cfg:GetDecryptedValue("db.pw","SFR")
```

Config holds an encrypted value

decryptionProviders.<key> -> an IPasswordEncryptionService class

Built-ins SFR/WIN - or register a VAULT provider that fetches

The lesson it teaches

The built-in SFR key is a constant in the framework

So it hides a value - it does not protect it

"A key that ships with the code is not a key"

That is why the provider is pluggable: give it a key from outside

Good architecture: the config says WHICH provider; the provider owns the key

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Live Demo 3: Fetch, Connect, Forget





Before

- PASOE openedge.properties
- with -P sitting in
- agentStartupParam



» Live:

```
vault -dev (container,
  throw-away token)

curl .../v1/secret/...
  -> the secret

11 lines of ABL:
HTTP Client GET,
then connect
```





You will see

- No password in any
- file under conf/
- findstr over conf/
- comes back empty

Secrets come from outside the build

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Rotation and Least Privilege

Rotation

Automate it; prefer short-lived secrets

Stores keep versions (AWSCURRENT / AWSPREVIOUS, Vault versions) so old and new overlap

12.8.11 lesson: re-encoding without changing the secret is not rotation

Least privilege

A DB user per application, not the DBA account in PASOE

Only the PASOE OS user reads conf/

Engineers do not get every secret

CI: a credentials store, not plain text

Short-lived and scoped beats long-lived and shared

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Keep Secrets Out of Git, Logs, Images and R-code

Git
Pre-commit / IDE scanning
.gitignore + *.example templates

Logs
Never log connect strings
Pipeline output must not leak (OWASP)

Images
No secrets baked into an image
Inject at run time, not build time

R-code
A literal password is a string in the .r
Fetch it, do not compile it in

A secret in an artefact is a secret you have already shared

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Your Monday Morning Checklist

1. Search for oech1:: and un-prefixed values in configs, .pf and CI
2. Change those secrets - do not just re-encode them
3. Pick a prefix (aedh0 on 13.0) and re-encode
4. Change the shipped PASOE defaults (tomcat, restuser, keypass)
5. Get secrets out of git, images and r-code
6. Plan rotation before you need it

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

In Closing

Encoding is a format
not a lock

A stronger prefix
is still a password

Secrets come
from outside the build

Know which is which, and keep the key out of the file

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Get the Code

Samples repo

- Three live demos with one-command scripts (encoding, the OECH1 hunt, fetch-connect-forget with Vault)
- Samples: secret from a file, an env var, Vault via the ABL HTTP Client; an SCL provider that fetches from Vault
- A genpassword / aedh0 note, the demo config tree, and the 13.0 findings
- All placeholders - no real hosts, users or secrets

Managing-Secrets-in-OpenEdge-Samples-

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

My Sessions at PUG Challenge Amsterdam

Session	Day	Time	Room
.NET on Linux. Why bother?	Wed 4	3:00 - 4:00 PM	New York 3
What are user-defined generic types...?	Thu 5	10:15 - 11:15 AM	New York 3
Managing secrets in OE configurations	**Fri 6**	**1:45 - 2:45 PM**	**New York 3**

Highlighted: this session

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Other Sessions Worth Your Time

Missed these? The decks go up afterwards - worth a read

Session	Speaker	Day	Time	Room
ESAM Unlocked: Security Governance	Shelley Chase	Fri 6	10:15 AM	New York 1
Advanced AI Security	Thomas Brom	Thu 5	1:45 PM	Brussels 1
Bringing Zero Trust with OpenEdge	Richard Kelters	Wed 4	11:45 AM	Athens

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.

Questions



Questions?

peter.judge@consultingwerk.com

© 2026 Consultingwerk Application Modernization Solutions Ltd. All rights reserved.


Consultingwerk
software architecture and development