

# Creating a new CA backend for FreeIPA with the help of AI

Thomas Woerner  
Principal Software Engineer / Red Hat



Can we use Claude.AI for development?



# First tries

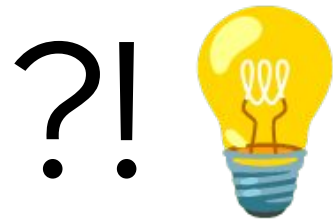
- Remove Python 2 support from the ansible-freeipa client deployment role  
→ Removed a lot of empty lines  
↳ Nice try!  
Retried → Failed again
- Remove support for old FreeIPA version prior to a specific version from ansible-freeipa client deployment role  
→ It simply “fixed” the minimal version check to match the requested version  
↳ Really?  
Retried → Failed again
- Analyze FreeIPA repo and find the optimal topology  
→ You should have 6 replication agreements per server  
That is not correct → Ok

→ That was not not really satisfactory



Why not trying something else  
**and**  
more complicated?





Create a *lightweight* CA  
that is able to act as a  
dogtag replacement for FreeIPA



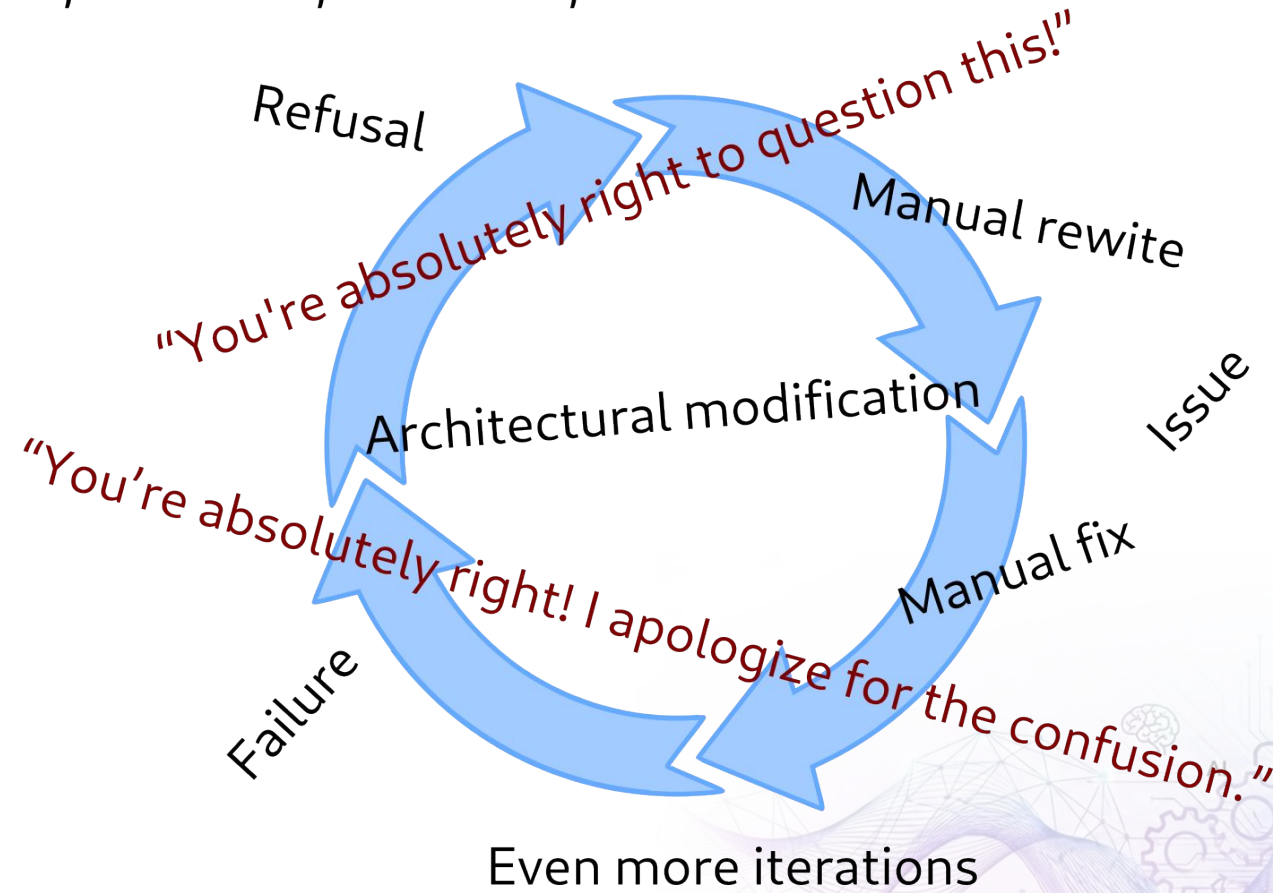
# Wish list

- Use at best only python-cryptography
- Service
- Logs for debugging
- LDAP for storage
- REST API
- Sub-CA
- Profiles
- NSSDB
- ACME - Automated Certificate Management Environment
- OCSP - Online Certificate Status Protocol
- HSM - Hardware Security Module, pkcs11



# Process

Iterations, iterations, iterations, ..



# Process

Simple impression about the iterations, reported by [Claude.AI](#)

- 2nd version with more than 7 iterations -

- Iteration 1-3: AI generates initial code → fails
- Iteration 4-7: Human identifies issues → AI refuses/misunderstands
- Iteration 8-12: Human rewrites approach → AI adapts pattern
- Iteration 13-18: Bug fixes (wrong LDAP DN, timing issues, schema errors)
- Iteration 19-25: Human refactoring and refinement
- Iteration 26++: Integration testing reveals more edge cases

Working code after significant manual rework

Most features required a lot of iterations with substantial manual correction and architectural guidance



# The Challenges

## Architecture Decisions

AI couldn't design things like DN structure or storage backend - manually coded or refined

## Security Implications

AI-generated crypto/auth code regularly wrong - manual rewrites

## Integration Points

AI didn't understand FreeIPA/Dogtag constraints - manual fixes

## Edge Cases

AI missed installation ordering and timing - manually solved

## Domain Knowledge

AI repeatedly forgot PKI/LDAP concepts - constant manual rewrites and re-teaching

## Basic Correctness

AI introduced bugs requiring manual debugging and fixes

## Refusals

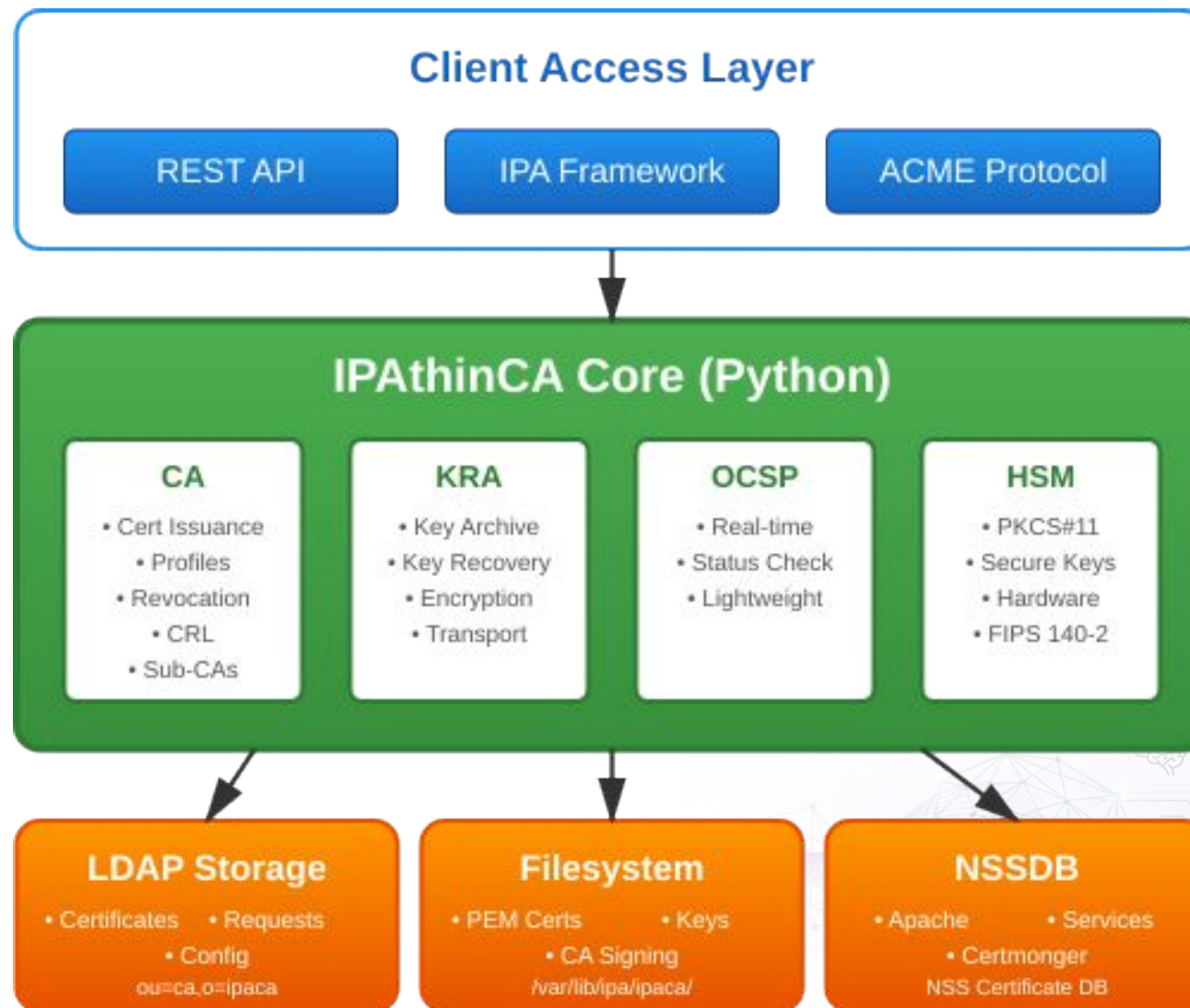
AI sometimes refused correct approaches - manual rewrites



# Result: IPAtinCA



# IPAthInCA Architecture



# Comparison

## IPATHinCA vs Dogtag PKI

Feature	IPATHinCA	Dogtag PKI
Implementation	Pure Python	Java (Tomcat)
Memory Usage	~110 MB	~500-800 MB (JVM)
Startup Time	< 1 second	~20-30 seconds
Storage Backend	LDAP+Files+NSS	LDAP+FileDB+NSS
Certificate Profiles	Python Code	LDAP/File Config
Dependencies	Python + Gunicorn	JRE + Tomcat + JSS
Codebase	~26k LOC	~500k LOC
IPA Compatibility	100%	100%
Cert Issuance	~100 ms	~150 ms

# Benefits

- Deployment much faster
- Lower resource usage
- Minimal dependencies
- Simplified architecture
- 100% IPA compatible
- Usable in limited environments - edge computing
- Usable in debian/Ubuntu



# Next Steps

- Testing, testing, testing, ..
- Replia deployment with CA
- PQC



# Deployment

How to deploy IPA server with IPATHINCA support

```
# ipa-server-install --unattended --ds-password=SomeDMpassword --admin-password=SomeADMINpassword  
--domain=test.local --realm=TEST.LOCAL --setup-dns --auto-forwarders --auto-reverse  
--no-dnssec-validation --no-ntp --setup-kra --random-serial-numbers --use-ipathinca
```

ipactl status output

```
# ipactl status  
Directory Service: RUNNING  
krb5kdc Service: RUNNING  
kadmin Service: RUNNING  
httpd Service: RUNNING  
ipa-custodia Service: RUNNING  
ipathinca Service: RUNNING  
ipa-otpd Service: RUNNING  
ipa: INFO: The ipactl command was successful
```



# Config file /etc/ipa/ipathinca.conf

```
# This file is read by the IPATHINCA service

[global]
realm = TEST.LOCAL
domain = test.local
host = ipaserver.test.local
basedn = dc=test,dc=local

[ca]
ca cert = /etc/ipa/ca.crt
random_serial_numbers = true

[ldap]
pool min connections = 2
pool_max_connections = 10

[server]
bind host = 0.0.0.0
https port = 8443
workers = 2
ssl cert = /var/lib/ipathinca/certs/server.crt
ssl key = /var/lib/ipathinca/private/server.key
pid file = /run/ipathinca/ipathinca.pid
user = ipaca
group = ipaca
```

```
[logging]
level = INFO
log file = /var/log/ipathinca/ipathinca.log
access log = /var/log/ipathinca/access.log
unicorn log = /var/log/ipathinca/unicorn.log
max log size = 10485760
backup_count = 10
```



# /var/lib/ipathinca

```
/var/lib/ipathinca/  
├── audit  
│   └── audit_signing.key  
├── ca  
│   └── ca_signing.key  
├── certs  
│   ├── ca.crt  
│   ├── ca audit.crt  
│   ├── ca crl.der  
│   ├── ca subsystem.crt  
│   ├── ipa ca agent.crt  
│   ├── obsp subsystem.crt  
│   └── server.crt  
├── kra  
│   ├── storage.key  
│   ├── transport.crt  
│   └── transport.key  
└── private  
    ├── ca audit.key  
    ├── ca subsystem.key  
    ├── ipa ca agent.key  
    ├── obsp subsystem.key  
    └── server.key
```

```
└── profiles  
    ├── IECUserRoles.json  
    ├── KDCs PKINIT Certs.json  
    ├── acmeIPAServerCert.json  
    ├── caIPAserviceCert.json  
    ├── caOCSPCert.json  
    ├── caSignedLogCert.json  
    └── caSubsystemCert.json
```



# /var/log/ipathinca

## `/var/log/ipathinca/`

```
|— access.log  
|— audit.log  
|— gunicorn.log  
└— ipathinca.log
```



Upstream repo

<https://github.com/t-woerner/freeipa/tree/IPAthinCA>

Fedora copr repo

<https://copr.fedorainfracloud.org/coprs/twoerner/freeipa/>

gist script using ansible-freeipa infrastructure to start a podman container

<https://gist.github.com/t-woerner/04a009618778aae567641cc5dea146da#file-ipathinca-test-sh>



# Questions?

