CLI - Why you should not set CNTLM nor Prisma

Documentation about proxies in the Group is abundant.

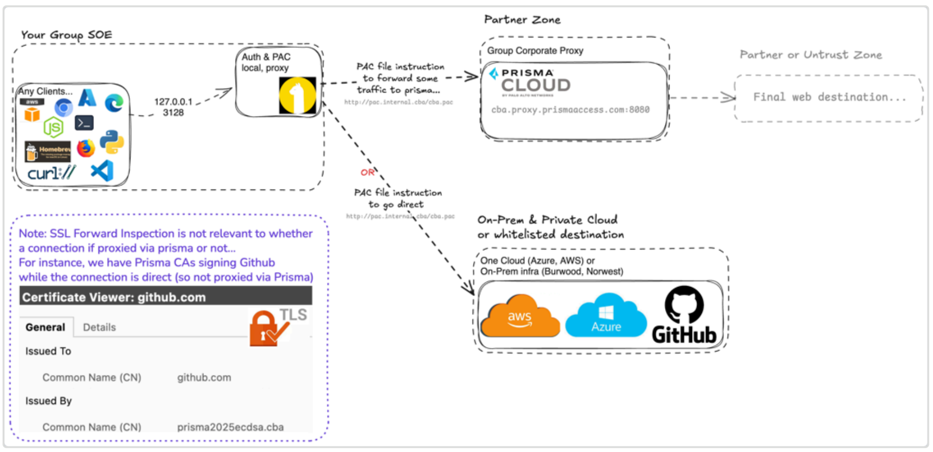
However, it often lacks the detailed information necessary for a clear understanding of setup and functionality. As a result, we have various guides that may have worked for some but may not apply to your situation due to differing user contexts and numerous variables, including:

- Your GUI or CLI client and its specific version
- Your local subnet and upstream proxy
- Your environment variables

This page aims to address your questions about proxies in the user zone (workstations and SOE). While some recommend using CNLTM, Prisma, or proxy.au.eds.com:8080, the following sections will explain why these options may not be suitable.

Note, this is a FAQ.
If you're simply after a quick-fix,
For MacOS, head there → Proxy and SSL Certificate issues | Solution
For Windows, head there → Setting up Alpaca on Windows

How does a good proxy setup look like ?



Note: SSL Forward Inspection is not relevant to whether a connection if proxied via prisma or not...
For instance, we have Prisma CAs signing Github while the connection is direct (so not proxied via Prisma)

Certificate Viewer: github.com

General | Details

Issued To

Common Name (CN) github.com

Issued By

Common Name (CN) prisma2025ecdsa.cba

Like this! but note EPZ has a different PAC URL
and those URLs, IP and FQDN might eventually change...

The problem with Prisma

Why not set the Prisma proxy in your CLI? It doesn't use our PAC file.

Setting `export {http,https,all}_proxy=http://cba.proxy.prismaaccess.com:8080` will forward all traffic to Prisma, so it also requires you to maintain a `no_proxy` variable.

See this → PAC and no_proxy

The problem with CNTLM

Using CNTLM is not advisable because it does not utilise our PAC file.

Previously, it was useful as it allowed our CLI to connect to proxy.au.eds.com:8080 by caching NTLM credentials, but this is no longer necessary.

Why do we need a PAC file ?

A PAC file (or its URL) is essential because it contains the logic that directs a client to connect directly or through specific proxies. Here are some examples of PAC file URLs:

- http://pac.internal.cba/cba.pac
- http://pac.internal.cba/epz.pac
- Another one from eds; I've lost the URL.

The PAC file specifies which proxy to use:

- cba.proxy.prismaaccess.com:8080 (no credentials required)
- proxy.au.eds.com:8080 (credentials required)

What if I don't use a PAC File?

You don't strictly need a PAC file, but you then would have to maintain a `no_proxy` environment variable on your endpoint (Windows, macOS/Unix, or Linux). Why manage that variable when a team already maintains the PAC file containing that logic?

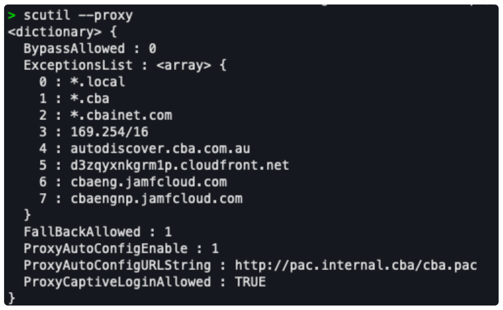
Can I use my own PAC File?

You can use your own PAC file through the CLI (e.g., `alpaca -C "http://localhost/ilikewastingtime.pac"`). However, you cannot set it at the OS level, as JAMF manages that and it takes precedence over network interface or service PAC entries.

~~You can set it at the OS level, but it's not advisable:~~ https://commbank.atlassian.net/wiki/spaces/SBD/pages/1362577886 Can't find link ~~. Doing so would override enterprise settings and could trigger alerts, as it involves modifying the enforced MDM/SOE profile.~~
Edit: Apologies, the blue team was uncomfortable with me sharing the above as it is offensive security which poses a security threat to the Group. Therefore, the article was taken down.

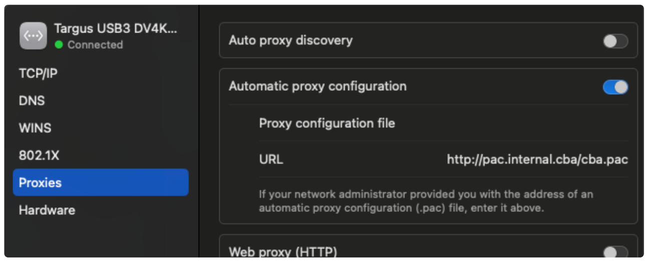
What if I set the PAC URL in the network interface?

You can do that, but it will be ineffective with MDM (JAMF or others). The enterprise profile takes precedence, so it will essentially be ignored.



```
> scutil --proxy
<dictionary> {
  BypassAllowed : 0
  ExceptionsList : <array> {
    0 : *.local
    1 : *.cba
    2 : *.cbainet.com
    3 : 169.254/16
    4 : autodiscover.cba.com.au
    5 : d3zqyxnkgrm1p.cloudfront.net
    6 : cbaeng.jamfcloud.com
    7 : cbaengnp.jamfcloud.com
  }
  FallBackAllowed : 1
  ProxyAutoConfigEnable : 1
  ProxyAutoConfigURLString : http://pac.internal.cba/cba.pac
  ProxyCaptiveLoginAllowed : TRUE
}
```

Global Var set by JAMF



Auto proxy discovery: off

Automatic proxy configuration: on

Proxy configuration file

URL http://pac.internal.cba/cba.pac

If your network administrator provided you with the address of an automatic proxy configuration (.pac) file, enter it above.

Web proxy (HTTP): off

Network service PAC URL

Different proxies...

- ALPACA `{http,https,all}_proxy=http://localhost:3128` with PAC file = **Everything works**
- ALPACA `{http,https,all}_proxy=http://localhost:3128` **without** PAC file = **5% will work**
- UNSET `unset {http,https,all}_proxy` or not defined = **5% will work** (Direct connections only)
- PRISMA `{http,https,all}_proxy=http://cba.proxy.prismaaccess.com:8080` = **95% will work**
- EDS `{http,https,all}_proxy=http://proxy.au.eds.com:8080` = **0% will work** (unless auth'ed)
- CNTLM `{http,https,all}_proxy=http://localhost:3128` fwd to EDS/Prisma =**95,1% will work**
- MITM `{http,https,all}_proxy=http://localhost:8888` = **% will work**

  This is based on the upstream proxies above, but MITM will break HTTPS requests when certificate pinning is in effect unless you exclude those domains from decryption.

  Interestingly, MITM can also resolve a current issue as described here:
  💔SSL Trust - Missing "Authority Key Identifier"

  A MITM proxy can be Fiddler, Charles, ProxyMan, Burp Suite, etc.
  Burp Suite is more focused on security testing than debugging, but you can perform basic hacking with a debugging proxy well using regex to rewrite requests and responses.

Can I run both CNTLM and Alpaca ?

⁉️ Yes, you can run both. However, since both proxies default to TCP port 3128, you must change the port for one to run them simultaneously. It's unclear why you would need CNTLM in this case.

Theoretically, you could route traffic from Burp to Charles, then to CNTLM, followed by Alpaca, and finally to Prisma. However, adding more proxies increases the risk of single points of failure.

Alpaca can also handle authentication, and you can run multiple instances on different TCP ports for testing with various PAC files and upstream proxies.

Do I need Alpaca ?

✓ Yes, you absolutely need Alpaca!
Alpaca detects our PAC file and utilises it effectively. Web requests will follow the PAC file's instructions for direct connections, while requests needing to go through Prisma will be forwarded accordingly.

But I need an authenticated local proxy...

That's fine! Like CNTLM, Alpaca securely hashes and stores your credentials.
To set this up, visit the Keyring section at: GitHub - samuong/alpaca: A local HTTP proxy for command-line tools. Supports PAC scripts and NTLM authentication.

Troubleshooting:

Connection Closed

This issue may occur if the upstream proxy blocks the connection. We blacklist access to certain resources (static and dynamic) based on Palo Alto URL categories, which can change. See Change a Site.

Error 502 - Bad Gateway

This can happen. The best approach is to run Alpaca in interactive mode:

```
$ alpaca
```

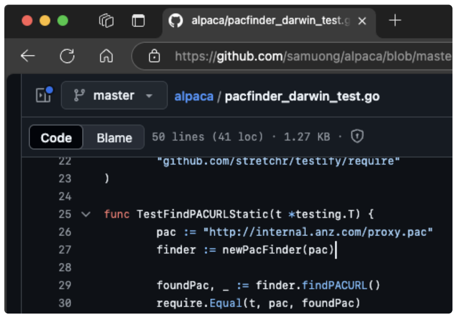
Check the output. **Did it detect the PAC file?**

⚠ If not, that might be the bug on MacOS I've raised with the main dev, and my PR fix is awaiting approval and merge...
GH Issue: MacOS - No PAC URL specified or Detected · Issue #144 · samuong/alpaca
GitHub PR: Fix issue #144: Update pacfinder_darwin.go to check PAC URL in SCDynamicStoreCopyProxies as fallback by Enelass · Pull Request #147 · samuong/alpaca

✓ Update 10 Aug 2025 : Now solved for release 2.0.11+ (Releases · samuong/alpaca)

Other financial institutions face challenges with proxies for CLI.

ANZ faced the same issue and hardcoded their PAC URL in the code.


alpaca/pacfinder_darwin_test.g
https://github.com/samuong/alpaca/blob/maste
master
alpaca / pacfinder_darwin_test.go
Code
Blame
50 lines (41 loc) · 1.27 KB

```
        "github.com/stretchr/testify/require"
)

func TestFindPACURLStatic(t *testing.T) {
        pac := "http://internal.anz.com/proxy.pac"
        finder := newPacFinder(pac)

        foundPac, _ := finder.findPACURL()
        require.Equal(t, pac, foundPac)
```

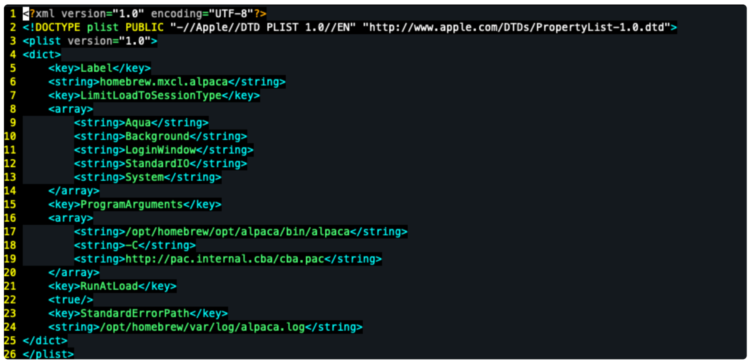
Yes, they did... 🙈

To resolve the issue of Alpaca not locating the PAC URL, provide it as an argument when calling it:

```
alpaca -C "http://pac.internal.cba/cba.pac" -u $(whoami) -d au
```

⁉️ Confirm that this is your correct PAC URL and domain (e.g., EPZ has a different PAC URL).

Alternatively, use the daemon if you installed it via Homebrew.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>homebrew.mxcl.alpaca</string>
    <key>LimitLoadToSessionType</key>
    <array>
        <string>Aqua</string>
        <string>Background</string>
        <string>LoginWindow</string>
        <string>StandardIO</string>
        <string>System</string>
    </array>
    <key>ProgramArguments</key>
    <array>
        <string>/opt/homebrew/opt/alpaca/bin/alpaca</string>
        <string>-C</string>
        <string>http://pac.internal.cba/cba.pac</string>
    </array>
    <key>RunAtLoad</key>
    <true/>
    <key>StandardErrorPath</key>
    <string>/opt/homebrew/var/log/alpaca.log</string>
</dict>
</plist>
```

1. `brew install alpaca`
2. `brew services start alpaca`
3. Now it still fails to detect the PAC file, so stop the daemon
    a. but **not** with ` `brew services stop alpaca` `, as that would delete the plist we need...
    b. disable with `launchctl unload ~/Library/LaunchAgents/homebrew.mxcl.alpaca.plist`
4. then edit the plist: `vim ~/Library/LaunchAgents/homebrew.mxcl.alpaca.plist`
5. and pass the argument in the daemon

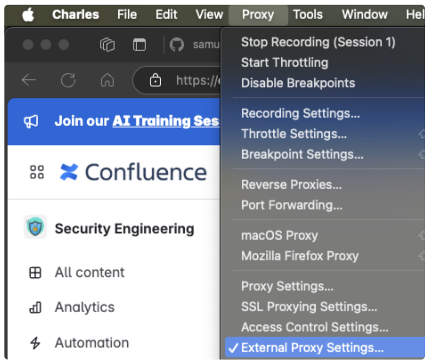
```
        <string>-C</string>`
        <string>http://pac.internal.cba/cba.pac</string>
```

⁉️ Again before you save this, make sure this is your right pac URL

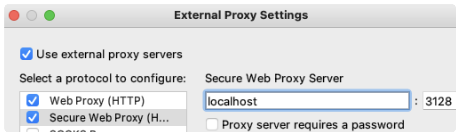
Charles is not working - HTTP 503

Make sure you have set an external proxy in Charles, it needs to know who to pass the traffic to

Pass the traffic to Alpaca or Prisma if I haven't convinced you to use Alpaca



Charles / Proxy/ External Proxy Settings



External Proxy Settings

✓ Use external proxy servers

Select a protocol to configure:
- ✓ Web Proxy (HTTP)
- ✓ Secure Web Proxy (H...

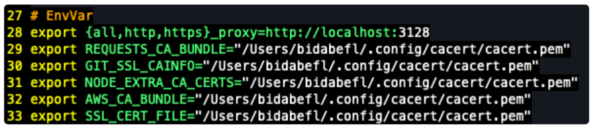
Secure Web Proxy Server: localhost : 3128

☐ Proxy server requires a password

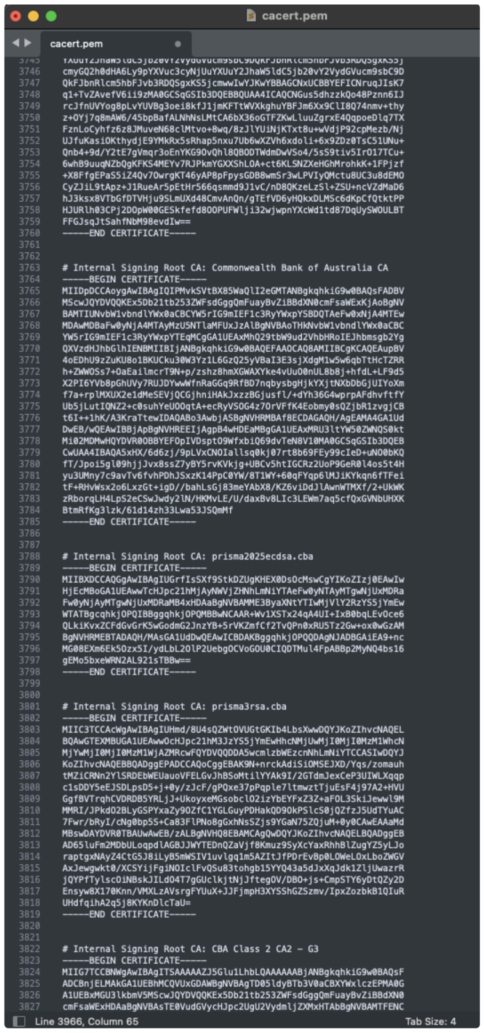
Change **BOTH** Web (http) and Secure Web (https) values

SSL trust issues

Make sure you have the right Environment Variable set and pointing to a Certificate Store which contains all our internal Root CAs
Proxy and SSL Certificate issues



```
# EnvVar
export {all,http,https}_proxy=http://localhost:3128
export REQUESTS_CA_BUNDLE="/Users/bidabefl/.config/cacert/cacert.pem"
export GIT_SSL_CAINFO="/Users/bidabefl/.config/cacert/cacert.pem"
export NODE_EXTRA_CA_CERTS="/Users/bidabefl/.config/cacert/cacert.pem"
export AWS_CA_BUNDLE="/Users/bidabefl/.config/cacert/cacert.pem"
export SSL_CERT_FILE="/Users/bidabefl/.config/cacert/cacert.pem"
```



```
cmyGQ2h0dHA6Ly9pYXVuc3cyNjUuYXUuY2JhaW5ldC5jb20vY2VydGVucm9sbC9D
QkFJbnRlcm5hbFJvb3RDQSgxKS5jcmwwIwYJKwYBBAGCNxUCBBYEFICNruqJIsK7
q1+TvZAvefV6ii9zMA0GCSqGSIb3DQEBBQUAA4ICAQCNGus5dhzzkQo48Pznn6IJ
rcJfnUVYog8pLvYUVBg3oei8kfJ1jmKFTtWVXkghuYBFJm6Xx9ClI8Q74nmv+thy
z+OYj7q8mAW6/45bpBafALNhNsLMtCA6bX36oGTFZKwLluuZgrxE4QqpoeDlq7TX
FznLoCyhfz6z8JMuveN68clMtvo+8wq/8zJlYUiNjKTxt8u+wVdjP92cpMezb/Nj
UJfuKasiOKthydjE9YMkRx5sRhap5nxu7Ub6wXZVh6xdoli+6x9ZDz0TsC51UNu+
Qnb4+9d/Y2tE7gVmqr3oEnYKG9OvQhl8QBODTWdmDwVSo4/5sS9tiv5IrO17TCu+
6whB9uuqNZbQgKFKS4MEYv7RJPkmYGXXShLOA+ct6KLSNZXeHGhMrohkK+1FPjzf
+X8FfgEPaS5iZ4Qv7OwrgKT46yAP8pFpysGDB8wmSr3wLPVIyQMctu8UC3u8dEMO
CyZJiL9tApz+J1RueAr5pEtHr566qsmmd9J1vC/nD8QKzeLzSl+ZSU+ncVZdMaD6
hJ3ksx8VTbGfDTVHju9SLmUXd48CmvAnQn/gTEfVD6yHQkxDLMSc6dKpCfQtktPP
HJURlh03CPj2DOpW00GESkfefd8OOPUFWlji32wjwpnYXcWd1td87DqUySWOULBT
FFGJsqJtSahfNbM98evdIw==
-----END CERTIFICATE-----


# Internal Signing Root CA: Commonwealth Bank of Australia CA
-----BEGIN CERTIFICATE-----
MIIDpDCCAoygAwIBAgIQIPMvkSVtBX85WaQlI2eGMTANBgkqhkiG9w0BAQsFADBV
MScwJQYDVQQKEx5Db21tb253ZWFsdGggQmFuayBvZiBBdXN0cmFsaWExKjAoBgNV
BAMTIUNvbW1vbndlYWx0aCBCYW5rIG9mIEF1c3RyYWxpYSBDQTAeFw0xNjA4MTEw
MDAwMDBaFw0yNjA4MTAyMzU5NTlaMFUxJzAlBgNVBAoTHkNvbW1vbndlYWx0aCBC
YW5rIG9mIEF1c3RyYWxpYTEqMCgGA1UEAxMhQ29tbW9ud2VhbHRoIEJhbmsgb2Yg
QXVzdHJhbGlhIENBMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAupBV
4oEDhU9zZuKU8o1BKUCku30W3Yz1L6GzQ25yVBaI3E3sjXdgM1w5w6qbTtHcTZRR
h+ZWWOSs7+OaEailmcrT9N+p/zshz8hmXGWAXYke4vUuO0nUL8b8j+hfdL+LF9d5
X2PI6YVb8pGhUVy7RUJDYwwWfnRaGGq9RfBD7nqbysbgHjkYXjtNXbDbGjUIYoXm
f7a+rplMXUX2e1dMeSEVjQCGjhniHAkJxzzBGjusfl/+dYh36G4wprpAFdhvftfY
Ub5jLutIQNZ2+c0suhYeUOOqtA+ecRyVSOG4z7OrVFfK4Eobmy0sQZjbR1zvgjCB
t6I++1hK/A3KraTtewIDAQABo3AwbjASBgNVHRMBAf8ECDAGAQH/AgEAMA4GA1Ud
DwEB/wQEAwIBBjApBgNVHREEIjAgpB4wHDEaMBgGA1UEAxMRU3ltYW50ZWNQS0kt
Mi02MDMwHQYDVR0OBBYEFOpIVDsptO9WfxbiQ69dvTeN8V10MA0GCSqGSIb3DQEB
CwUAA4IBAQA5xHX/6d6zj/9pLVxCNOIallsq0kj07rt8b69FEy99cIeD+uNO0bKQ
fT/Jpoi5gl09hjjJvx8ssZ7yBY5rvKVkjg+UBCv5htIGCRz2UoP9GeR0l4os5t4H
yu3UMny7c9avTv6fvhPDhJSxzK14PpC0YW/8T1WY+60qFYqp6lMJiKYkqn6fTFei
tF+RHvWsx2o6LxzGt+igD//bahLsGj83meYAbX8/KZ6viDdJlAwnWTMXf/2+UkWK
zRborqLH4LpS2eCSwJwdy2lN/HKMvLE/U/daxBv8LIc3LEWm7aq5cfQxGVNbUHXK
BtmRfKg3lzk/61d14zh33Lwa53JSQmMf
-----END CERTIFICATE-----


# Internal Signing Root CA: prisma2025ecdsa.cba
-----BEGIN CERTIFICATE-----
MIIBXDCCAQGgAwIBAgIUGrfIsSXf9StkDZUgKHEX0DsOcMswCgYIKoZIzj0EAwIw
HjEcMBoGA1UEAwwTcHJpc21hMjAyNWVjZHNhLmNiYTAeFw0yNTAyMTgwNjUxMDRa
Fw0yNjAyMTgwNjUxMDRaMB4xHDAaBgNVBAMME3ByaXNtYTIwMjVlY2RzYS5jYmEw
WTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAR+Wv1XSTx24qA4UI+IxB0bqLEvOce6
QLkiKvxZCFdGvGrK5wGodmG2JnzYB+5rVKZmfCf2TvQPn0xRU5Tz2Gw+ox0wGzAM
BgNVHRMEBTADAQH/MAsGA1UdDwQEAwICBDAKBggqhkjOPQQDAgNJADBGAiEA9+nc
MG08EXm6Ek5Ozx5I/ydLbL2OlP2UebgOCVoGOU0CIQDTMul4FpABBp2MyNQ4bs16
gEMo5bxeWRN2AL921sTBBw==
-----END CERTIFICATE-----


# Internal Signing Root CA: prisma3rsa.cba
-----BEGIN CERTIFICATE-----
MIIC3TCCAcWgAwIBAgIUHmd/8U4sQZWtOVUGtGKIb4LbsXwwDQYJKoZIhvcNAQEL
BQAwGTEXMBUGA1UEAwwOcHJpc21hM3JzYS5jYmEwHhcNMjUwMjI0MjI0MzM1WhcN
MjYwMjI0MjI0MzM1WjAZMRcwFQYDVQQDDA5wcmlzbWEzcnNhLmNiYTCCASIwDQYJ
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAK9N+nrckAdiSiOMSEJXD/Yqs/zomauh
tMZiCRNn2YlSRDEbWEUauoVFELGvJhBSoMtilYYAk9I/2GTdmJexCeP3UIWLXqqp
c1sDDY5eEJSDLpsD5+j+0y/zJcF/gPQxe37pPqple7ltmwztTjuEsF4j97A2+HVU
GgfBVTrqhCVDRDB5YRLjJ+UkoyxeMGsobclO2izYbEYFxZ3Z+aFOL3SkiJewwl9M
MMRI/JPkdO2BLyGSPYxaZy9OZfC1YGLGuyPDHakQD9OkPSlcS0jQZfzJ5UdTYuAC
7Fwr/bRyI/cNg0bp5S+Ca83FlPNo8gGxhNsSZjs9YGaN75ZQjuM+0y0CAwEAAaMd
MBswDAYDVR0TBAUwAwEB/zALBgNVHQ8EBAMCAgQwDQYJKoZIhvcNAQELBQADggEB
AD65luFm2MDbULoqpdlAGBJJWYTEDnQZaVjf8Kmuz9SyXcYaxRhhBlZugYZ5yLJo
raptgxNAyZ4CtG5J8iLyB5mWSIV1uvlgq1m5AZItJfPDrEvBp0LOWeLOxLboZWGV
AxJewgwkt0/XCSYijFgiNOIclFvQSu83tohgb15YYQ43a5dJxXqJdk1ZljUwazrR
jQYPfTylscOiNBskJILdO4T7gGUclkjtNjJftegOV/DBO+js+CmpSTY6yDtQZy2D
Ensyw8X170Knn/VMXLzAVsrgFYUuX+JJFjmpH3XYSShGZSzmv/IpxZozbkB1QIuR
UHdfqihA2q5j8KYKnDlcTaU=
-----END CERTIFICATE-----


# Internal Signing Root CA: CBA Class 2 CA2 - G3
-----BEGIN CERTIFICATE-----
MIIG7TCCBNWgAwIBAgITSAAAAAZJ5Glu1LhbLQAAAAAABjANBgkqhkiG9w0BAQsF
ADCBnjELMAkGA1UEBhMCQVUxGDAWBgNVBAgTD05ldyBTb3V0aCBXYWxlczEPMA0G
A1UEBxMGU3lkbmV5MScwJQYDVQQKEx5Db21tb253ZWFsdGggQmFuayBvZiBBdXN0
cmFsaWExHDAaBgNVBAsTE0VudGVycHJpc2UgU2VydmljZXMxHTAbBgNVBAMTFENC
```

Helpful? Drop me a thanks on Achievers ! And if you've got knowledge to share, don't hold back - we all grow when we learn from each other 💡