Certificate Match Test Utility

Background

Troubleshooting two factor authentication for Cohesity systems can be challenging. Cohesity uses both the user certificate from the web browser and the user certificate from Active Directory as the matching mechanism for browser user identity with known Active Directory user identities. More specifically, Cohesity matches these certificates using the Serial Number fields from both certificates.

We have found that many factors can interfere with the matching process. So, we built this utility to help provide some insight into the process.

What the utility does

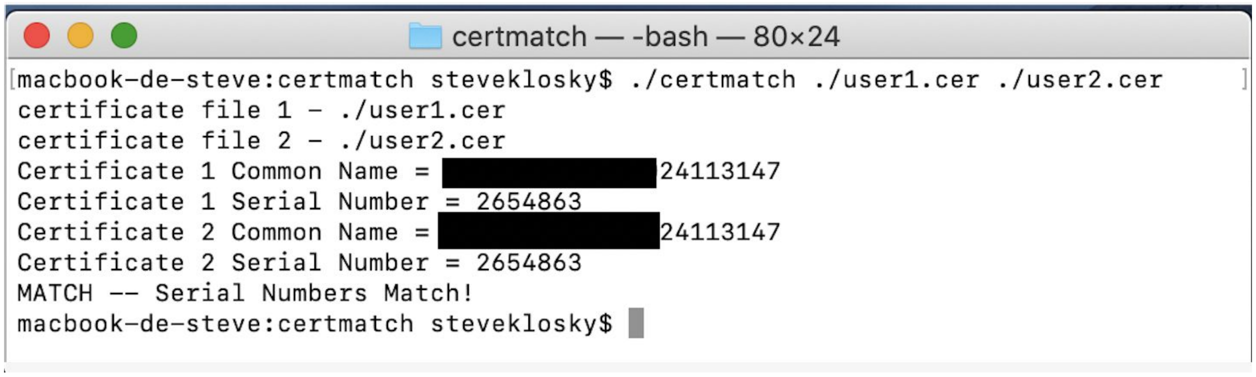
This is a command line utility. Load it onto the appropriate system (currently there is a Mac version and a Linux version). Invoke the utility and give it two x509 certificate files to compare. The utility will use the Cohesity matching logic to determine if the two certificates match. The Cohesity matching logic does a binary comparison of the certificates' serial numbers.

How to install and run the utility

Get the executable file from Cohesity. Currently Steve Klosky builds the executables for the target platforms and delivers using appropriate file sharing systems. Contact Steve at steve.klosky@cohesity.com to make arrangements.

Download the executable and move it to a working directory. That's the installation. It's a very basic golang program.

To run the program, open a command line tool and use appropriate os commands to run "certmatch" (or certmatch.exe on windows). On the same command line, provide the first certificate file as parameter 1 and the second certificate file as parameter 2. (for this example, both the executable and the two certificates are in the same working directory. This isn't a requirement. You just have to give the program two valid file location specs). Then hit enter. The program will attempt to load, parse and compare the two certificate files using the golang x509 utilities. (just like the Cohesity server does). The utility will provide an output indicating whether or not the two certificates match. See the screenshot below for an example.



```
macbook-de-steve:certmatch steveklosky$ ./certmatch ./user1.cer ./user2.cer
certificate file 1 - ./user1.cer
certificate file 2 - ./user2.cer
Certificate 1 Common Name = [redacted]24113147
Certificate 1 Serial Number = 2654863
Certificate 2 Common Name = [redacted]24113147
Certificate 2 Serial Number = 2654863
MATCH -- Serial Numbers Match!
macbook-de-steve:certmatch steveklosky$
```

Troubleshooting Workflow

The intent here is to narrow down possible complications leading to Browser / AD certificate mismatches. Here are some common complications / scenarios which this utility will help to troubleshoot.

Scenarios

1. Missing or Incorrect browser user certificate
   a. Many organizations have complicated certificate schemes which present multiple certificate options from the browser. This utility will help to isolate and test browser certificates one at a time
   b. Sometimes the browser is not presenting the user certificate to the web server. In this case, these troubleshooting steps will help to identify missing user certificates.
2. Missing or Incorrect AD user certificate
   a. Many organizations have complicated certificate schemes which have multiple certificates published for each user in AD. This utility will help to isolate and test AD user certificates one at a time
   b. Sometimes user certificates are not properly published in AD. In this case, these troubleshooting steps will help to identify missing user certificates.
3. Certificate mismatches which are undetectable by visual inspection
   a. When the server does it’s comparison, it uses the binary versions of the certificates. While the visual versions may match, there may be a binary level mismatch. This utility helps identify this mismatch mode.

Troubleshooting Guidelines

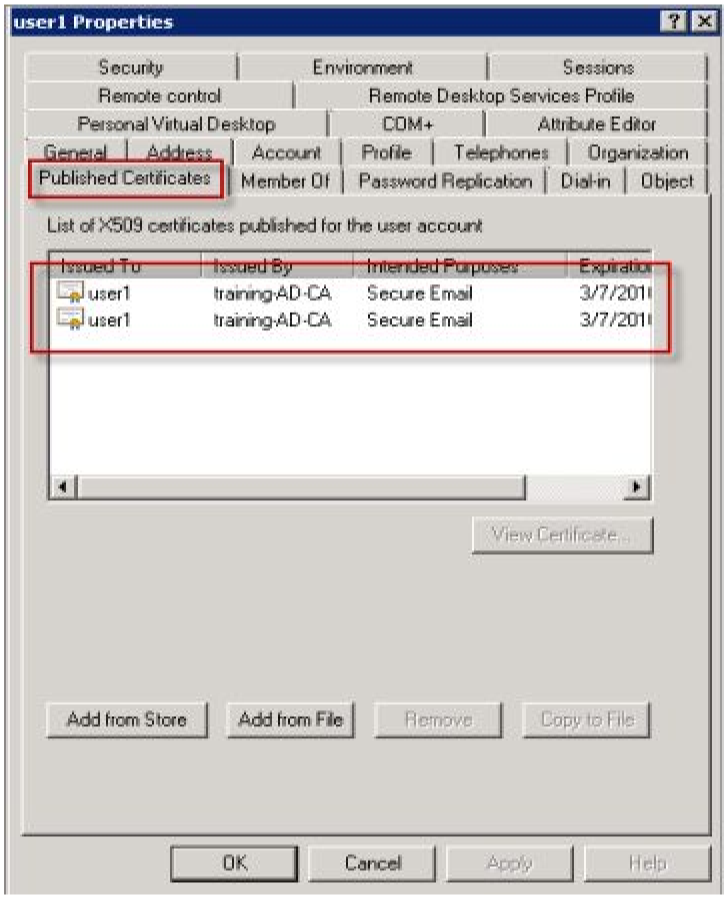
The basic troubleshooting approach for this utility is to manually export the browser based user certificate and the AD based user certificate to a local working directory. Once in the working directory, invoke the certificate matching test utility to determine if the two certificates match.

Here's a procedure to export the user certificate from Chrome

https://www.wikihow.com/Export-Certificate-Public-Key-from-Chrome

Here's a procedure to export the user certificate from AD

- Open Active Directory for Users and Computer as a Domain Admin
- Navigate to the test user
- Inspect the test user AD properties
- Select the "Published Certificates" tab

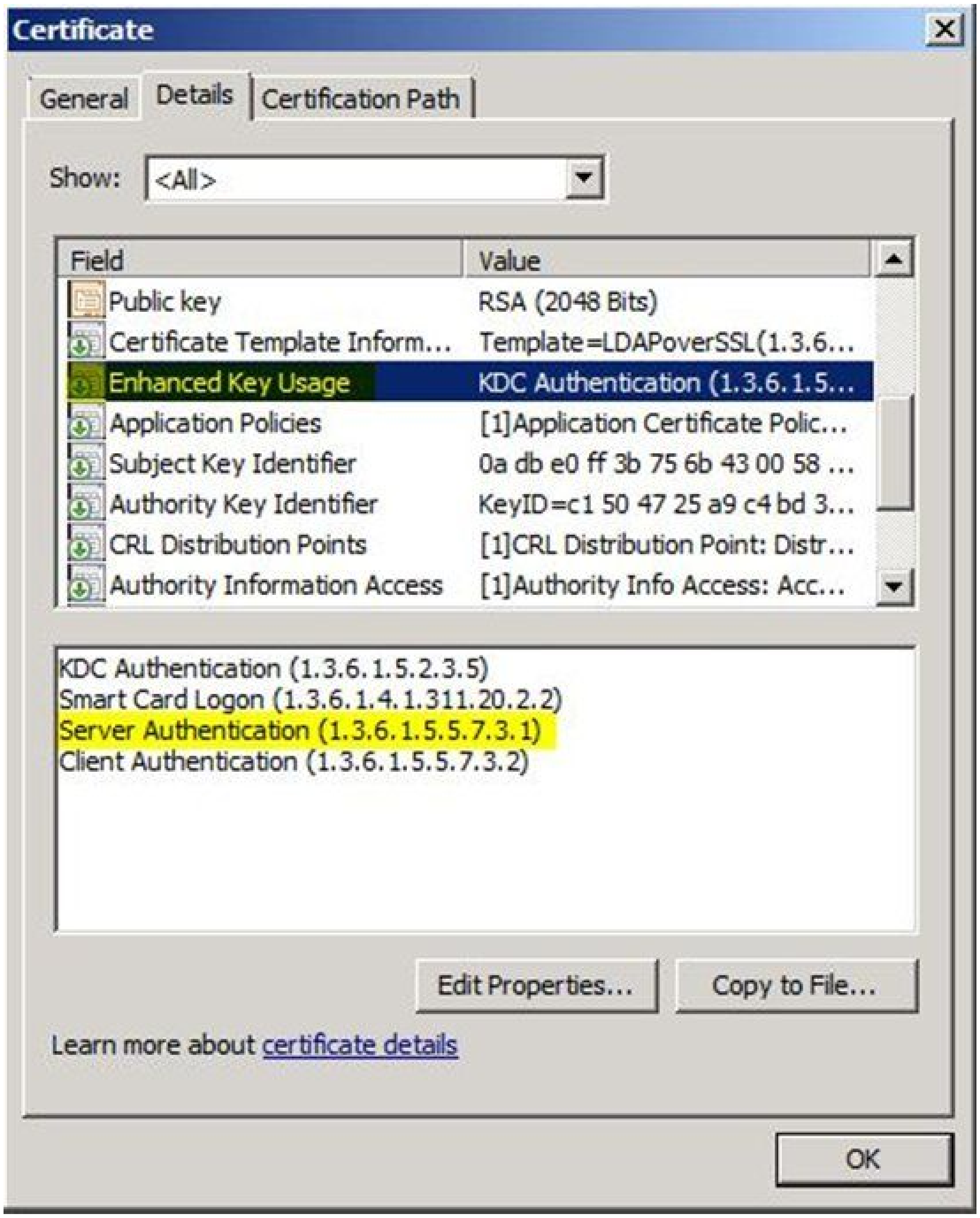


List of X509 certificates published for the user account

Issued To	Issued By	Intended Purposes	Expiratio
user1	training-AD-CA	Secure Email	3/7/201
user1	training-AD-CA	Secure Email	3/7/201

- Select the user certificate to be tested
- Use the "Copy to File" button to export the certificate to a file

- Make sure to copy the Public Key only and use the .DER or .cer file format



Field	Value
Public key	RSA (2048 Bits)
Certificate Template Inform...	Template=LDAPoverSSL(1.3.6...
Enhanced Key Usage	KDC Authentication (1.3.6.1.5...
Application Policies	[1]Application Certificate Polic...
Subject Key Identifier	0a db e0 ff 3b 75 6b 43 00 58 ...
Authority Key Identifier	KeyID=c1 50 47 25 a9 c4 bd 3...
CRL Distribution Points	[1]CRL Distribution Point: Distr...
Authority Information Access	[1]Authority Info Access: Acc...

```
KDC Authentication (1.3.6.1.5.2.3.5)
Smart Card Logon (1.3.6.1.4.1.311.20.2.2)
Server Authentication (1.3.6.1.5.5.7.3.1)
Client Authentication (1.3.6.1.5.5.7.3.2)
```

Notes

If you try one of the above procedures and the certificate is not available for export, that may be the cause of the mismatch.

How to build the utility from source code

In some cases, organizations cannot download “utility” code from Cohesity. In that case, we provide the option to get the source code and compile your own version.

Build Steps

- Setup a golang development environment
  - Here’s the golang setup info -- https://golang.org/doc/install
- Go to github and download the source code
  - The github project is here -- https://github.com/sklosky/certmatch
  - The code is in the file named “certmatch.go”
  - A text version of the code is below in this document for reference purposes
    - Note the github version is the most up to date version
- Put the source code in the appropriate golang directory
- Use standard go build commands to build the executable

Here’s the utility code

Please feel free to copy and modify this code as needed to help with certificate troubleshooting. The code is provided “as is” with no guarantee, warranty, etc.

```
// A utility program to do a binary comparison of two certificate files
// using golang and the standard golang x509 parsing algorithm
//
// Version 1.0
//
// January 2020
//
// Steve Klosky
// steve.klosky@cohesity.com
//

package main

import (
       "fmt"
       "os"
       "crypto/x509"
       )

func main() {
        inargs := os.Args[1:]

       // add input args checking here
```

```
fmt.Printf("certificate file 1 - ")
 fmt.Printf(inargs[0])
 fmt.Printf("\n")
fmt.Printf("certificate file 2 - ")
 fmt.Printf(inargs[1])
 fmt.Printf("\n")

// read certificate file 1 and parse for key fields
// common name and serial number

 certCerFile, err := os.Open(inargs[0])

// add file open error handling here

 derBytes := make([]byte,3000)

 count,err:=certCerFile.Read(derBytes)

 certCerFile.Close()

 // trim the bytes to actual length in call
 cert1,err := x509.ParseCertificate(derBytes[0:count])

 if err != nil {
    fmt.Printf("certificate 1 parse error")
 }

 fmt.Printf("Certificate 1 Common Name = %s\n", cert1.Subject.CommonName)
 fmt.Printf("Certificate 1 Serial Number = %s\n", cert1.SerialNumber.String())


// read certificate file 2 and parse for key fields
// common name and serial number

 certCerFile2, err := os.Open(inargs[1])

// add file open error handling here

 derBytes2 := make([]byte,3000)

 count2,err:=certCerFile2.Read(derBytes2)

 certCerFile2.Close()

 // trim the bytes to actual length in call
 cert2,err := x509.ParseCertificate(derBytes2[0:count2])

 if err != nil {
    fmt.Printf("certificate 2 parse error")
 }

 fmt.Printf("Certificate 2 Common Name = %s\n", cert2.Subject.CommonName)
 fmt.Printf("Certificate 2 Serial Number = %s\n", cert2.SerialNumber.String())

// Binary compare the two serial numbers

if cert1.SerialNumber.Cmp(cert2.SerialNumber) != 0 {
```

```
                fmt.Printf("MISMATCH -- Serial Numbers DO NOT Match!\n")
        } else {
                fmt.Printf("MATCH -- Serial Numbers Match!\n")
        }

}
```