JumpCloud RADIUS Certificate based Auth 3rd Party Tools for BYO Certificates

Background

JumpCloud has integrated the use of digital X.509 certificates for authentication, known as the most secure and most frictionless to the user. The JumpCloud RADIUS service is now capable of certificate based authentication to control access network resources (RADIUS-CBA).
The first release of the RADIUS-CBA allows customers to import certificates (bring your own certificate - BYOC) from any source compliant with X.509 standard. This capability allows customers to keep the existing certificate infrastructure in place for RADIUS network access purposes.
In order to authenticate users with certificates in the JumpCloud RADIUS service 3 fundamental functions are needed:

1. Certificate Management
   Certificate management involves provisioning (minting) certificates and managing their lifecycle. The provisioning function involves creating (minting) a certificate but also defining the chain of trust (origin) of the certificate. RADIUS-CBA supports certificates from a Certificate Authority (CA) or a self signed certificate. The life cycle management allows an IT administrator to “revoke” and “reissue” certificates as required.
2. Certificate Delivery
   A certificate must be physically deployed into a secure area of every target device to be used for authentication. The methodology and process to deploy a certificate varies depending on the Operating System. Since RADIUS-CBA supports Windows, MacOS, iOS, Linux, the delivery tools must be able to support at least those device types or at least those needed by the customer.
3. RADIUS Certificate based Authentication
   This fundamental function is provided by the new JumpCloud RADIUS-CBA

This capability also means IT administrators must manage the following 3 basic certificate lifecycle functions outside of JumpCloud:

- Certificate Provisioning
  The provisioning function involves creating (minting) a certificate but also defining the chain of trust (origin) of the certificate. RADIUS-CBA supports certificates from a Certificate Authority (CA) or a self signed certificate.
- Certificate change of status
  A certificate can have a valid, expired or revoked status. In order to enforce the mentioned statutes, an IT administrator must be able to “revoke” and “reissue” every managed certificate.
- Certificate Delivery to devices
  A certificate must be physically deployed into a secure area of every target device to be used for authentication. The methodology and process to deploy a certificate varies

depending on the Operating System. Since RADIUS-CBA supports Windows, MacOS, iOS, Linux, the delivery tools must be able to support at least those device types or at least those needed by the customer.

Business Case

The objective of this document is to facilitate customers new to certificate authentication to quickly and effectively find all the tools necessary to leverage certificate based authentication.

Problem Statement

The RADIUS-CBA service requires the first two fundamental functions (Certificate Management and Certificate Delivery) to be done apriori and outside of JumpCloud. Therefore, the IT administrator must use one of many tools in the marketplace to perform those actions.
The problem is further compounded for customers new to certificate authentication who must research and test the required solutions.

Proposed Solution

The proposed solution provides a list of some 3rd party tools and providers (commercial and open source) that can perform the required functions. JumpCloud has researched and identified some of the best known tools and vendors available in the industry capable of performing the required functions listed below in no particular order:

- Certificate provisioning tools
  Certificates need to have an origin or root certificate where the trust begins (Certificate Trust Chain). A certificate and corresponding Certificate Trust Chain can be generated by either a Certificate Authority (CA) or Self signed.
    - CA based certificates
        - Google Certificate Authority Services (https://cloud.google.com/certificate-authority-service)
        - Amazon Certificate Manager (https://aws.amazon.com/private-ca/)
        - Digicert (https://www.digicert.com/)
        - Globalsign (https://www.globalsign.com/en)
        - AppviewX (https://www.appviewx.com/products/cert/)
        - Keyfactor (https://www.keyfactor.com/)
        - Venafi (https://www.venafi.com/platform/zero-touch-pki)
    - Self signed based certificates
        - OpenSSL (www.openssl.org/)
        - GnuTLS (www.gnutls.org/manual/html_node/certtool-Invocation.html)
- Certificate lifecycle management
    - AppviewX
    - Keyfactor
    - Venafi
    - Globalsign
- Certificate delivery to devices
    - Jamf (https://www.jamf.com/)

        - Support MacOS, iOS
        - Integration with AppviewX, Venafi, and others.
        - Enables the use of variables to embed and customize certificate fields (See https://jamf.it/macprofile and https://jamf.it/iosprofile for detailed information)
        - Third party certificate delivery is supported by the Jamf SCEP Proxy (Jamf 3rd Party Certificate delivery via SCEP Proxy)
    - Microsoft Intune (https://www.microsoft.com/en-us/security/business/endpoint-management/microsoft-intune?rtc=1)
        - Supports Windows, Linux, MacOS, iOS, Android
        - Integration with Globalsign, AppviewX, Keyfactor, and others.
    - AirWatch (http://www.air-watch.com/)
        - Supports iOS and Android
        - Integration with Globalsign, AppviewX,Venafi, Keyfactor and others
    - MobileIron (https://www.ivanti.com/company/history/mobileiron)
        - Supports iOS, Android, macOS and Windows
        - Integration with Globalsign, AppviewX,Venafi, Keyfactor and others
    - Soti (https://www.soti.net/)
        - Supports iOS, Android, macOS and Windows
        - Integration with AppviewX
- Tools that perform all functions under a single platform
  Unfortunately, there is no provider that can perform all functions under the same platform; however, organizations like Globalsign, AppviewX and Keyfactor have integrated certificate delivery as part of their certificate management systems.

Application of Solution

With the availability of the above tools and the RADIUS-CBA interface, enable IT Administrators to quickly enable certificate based authentication for RADIUS. The sequence of steps to configure RADIUS-CBA is as follows:

1. Create a Certificate Trust Chain
2. Create certificates compatible with RADIUS-CBA (See Appendix - Certificates compatible with RADIUS-CBA)
3. Deploy certificates to target devices
   The delivery of certificates varies depending on the target device (OS platform, OS Version, etc). However, the same certificate applies to all target devices. (See Appendix - Manual Certificate deployment)
4. Import the Certificate Trust Chain into RADIUS-CBA
5. Configure RADIUS access point in RADIUS-CBA
6. Configure authentication factors (MFA) including Certificate Authentication (See Appendix - Admin User Interface)
7. Assign users to the RADIUS access point via User Groups

After the above configuration is completed, end users can now authenticate to RADIUS-CBA by using a device with provisioned certificates.

Finally, the RADIUS-CBA service will obtain the certificate content from the device, verify the standing of the user against the JC directory and perform access control to the desired RADIUS network resource.

Future Direction
Although JumpCloud plans to provide each of the above tools as part of the JumpCloud platform in future releases, many existing JumpCloud customers will continue using their existing certificate lifecycle management tools.

Conclusion
The combination of any of the above tools and the RADIUS-CBA capability enables customers new to certificate based authentication to effectively leverage the RADIUS-CBA service quickly.

IT Administrators to quickly enable certificate based authentication for RADIUS.

- Enabled customer new to certificate based auth to effectively leverage the RADIUS-CBA service
- Provide options based on experience, cost and missing functions

Appendices

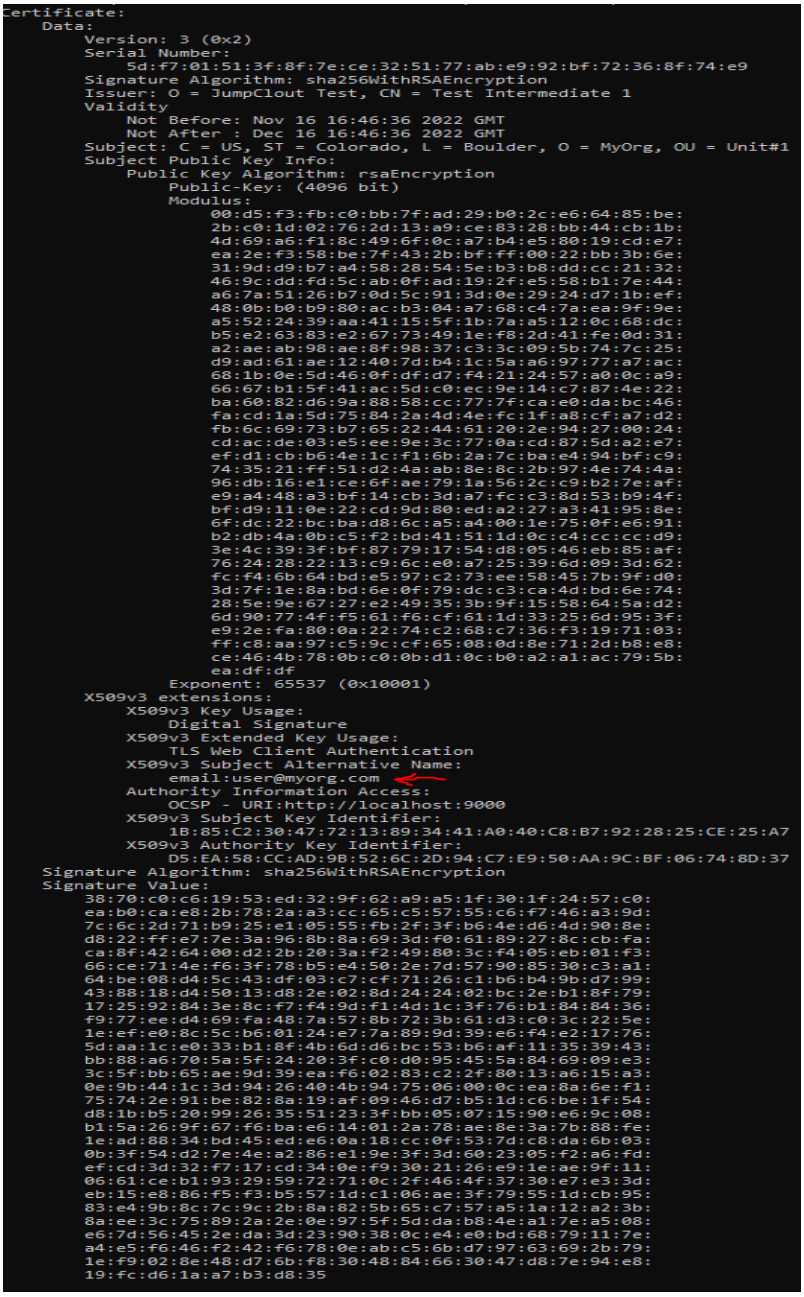
Appendix – Certificates compatible with RADIUS-CBA

RADIUS-CBA supports only certificates that contain a user identifier within the certificate (User Certificates) located on the target client device. The User Certificates can be of various types.

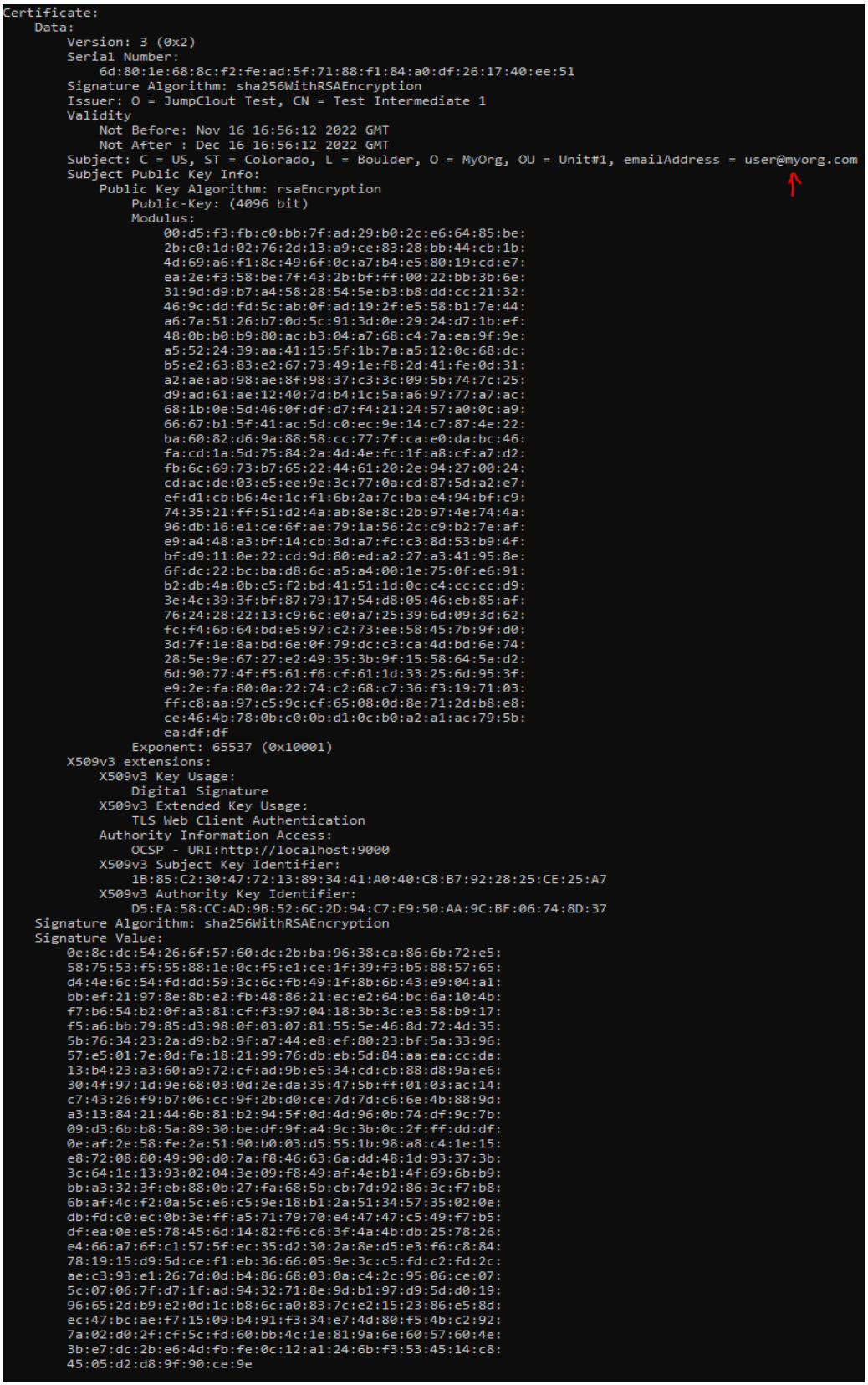
User Certificate types supported by RADIUS-CBA

There are 3 types of certificates RADIUS-CBA supports based on the location (certificate field) where the user identifier resides within the certificate as illustrated below.

a. Client cert with JumpCloud user email in the subject alternative name

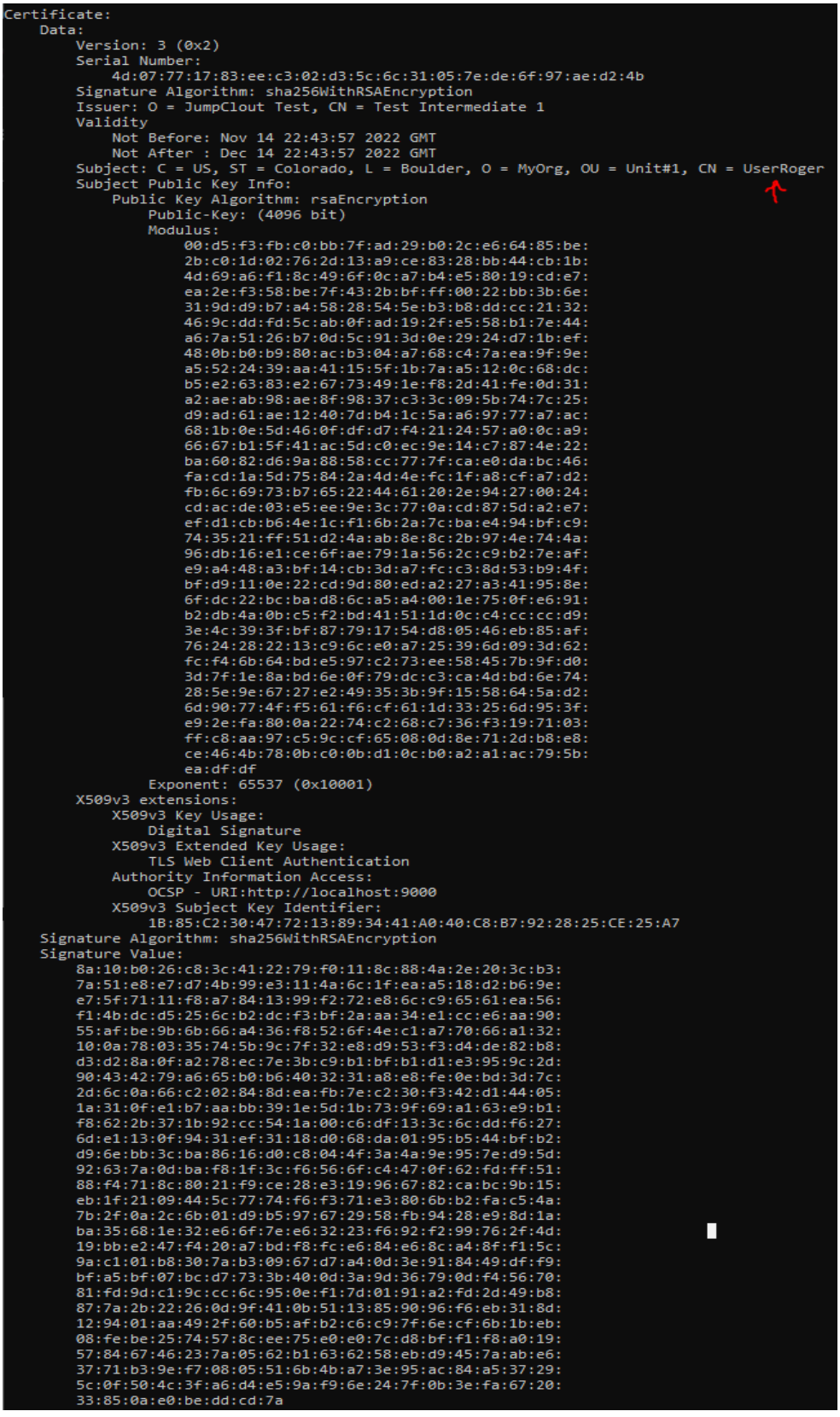


b. Client cert with JumpCloud user email in the subject distinguished name



```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            6d:80:1e:68:8c:f2:fe:ad:5f:71:88:f1:84:a0:df:26:17:40:ee:51
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: O = JumpClout Test, CN = Test Intermediate 1
        Validity
            Not Before: Nov 16 16:56:12 2022 GMT
            Not After : Dec 16 16:56:12 2022 GMT
        Subject: C = US, ST = Colorado, L = Boulder, O = MyOrg, OU = Unit#1, emailAddress = user@myorg.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
                    00:d5:f3:fb:c0:bb:7f:ad:29:b0:2c:e6:64:85:be:
                    2b:c0:1d:02:76:2d:13:a9:ce:83:28:bb:44:cb:1b:
                    4d:69:a6:f1:8c:49:6f:0c:a7:b4:e5:80:19:cd:e7:
                    ea:2e:f3:58:be:7f:43:2b:bf:ff:00:22:bb:3b:6e:
                    31:9d:d9:b7:a4:58:28:54:5e:b3:b8:dd:cc:21:32:
                    46:9c:dd:fd:5c:ab:0f:ad:19:2f:e5:58:b1:7e:44:
                    a6:7a:51:26:b7:0d:5c:91:3d:0e:29:24:d7:1b:ef:
                    48:0b:b0:b9:80:ac:b3:04:a7:68:c4:7a:ea:9f:9e:
                    a5:52:24:39:aa:41:15:5f:1b:7a:a5:12:0c:68:dc:
                    b5:e2:63:83:e2:67:73:49:1e:f8:2d:41:fe:0d:31:
                    a2:ae:ab:98:ae:8f:98:37:c3:3c:09:5b:74:7c:25:
                    d9:ad:61:ae:12:40:7d:b4:1c:5a:a6:97:77:a7:ac:
                    68:1b:0e:5d:46:0f:df:d7:f4:21:24:57:a0:0c:a9:
                    66:67:b1:5f:41:ac:5d:c0:ec:9e:14:c7:87:4e:22:
                    ba:60:82:d6:9a:88:58:cc:77:7f:ca:e0:da:bc:46:
                    fa:cd:1a:5d:75:84:2a:4d:4e:fc:1f:a8:cf:a7:d2:
                    fb:6c:69:73:b7:65:22:44:61:20:2e:94:27:00:24:
                    cd:ac:de:03:e5:ee:9e:3c:77:0a:cd:87:5d:a2:e7:
                    ef:d1:cb:b6:4e:1c:f1:6b:2a:7c:ba:e4:94:bf:c9:
                    74:35:21:ff:51:d2:4a:ab:8e:8c:2b:97:4e:74:4a:
                    96:db:16:e1:ce:6f:ae:79:1a:56:2c:c9:b2:7e:af:
                    e9:a4:48:a3:bf:14:cb:3d:a7:fc:c3:8d:53:b9:4f:
                    bf:d9:11:0e:22:cd:9d:80:ed:a2:27:a3:41:95:8e:
                    6f:dc:22:bc:ba:d8:6c:a5:a4:00:1e:75:0f:e6:91:
                    b2:db:4a:0b:c5:f2:bd:41:51:1d:0c:c4:cc:cc:d9:
                    3e:4c:39:3f:bf:87:79:17:54:d8:05:46:eb:85:af:
                    76:24:28:22:13:c9:6c:e0:a7:25:39:6d:09:3d:62:
                    fc:f4:6b:64:bd:e5:97:c2:73:ee:58:45:7b:9f:d0:
                    3d:7f:1e:8a:bd:6e:0f:79:dc:c3:ca:4d:bd:6e:74:
                    28:5e:9e:67:27:e2:49:35:3b:9f:15:58:64:5a:d2:
                    6d:90:77:4f:f5:61:f6:cf:61:1d:33:25:6d:95:3f:
                    e9:2e:fa:80:0a:22:74:c2:68:c7:36:f3:19:71:03:
                    ff:c8:aa:97:c5:9c:cf:65:08:0d:8e:71:2d:b8:e8:
                    ce:46:4b:78:0b:c0:0b:d1:0c:b0:a2:a1:ac:79:5b:
                    ea:df:df
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage:
                Digital Signature
            X509v3 Extended Key Usage:
                TLS Web Client Authentication
            Authority Information Access:
                OCSP - URI:http://localhost:9000
            X509v3 Subject Key Identifier:
                1B:85:C2:30:47:72:13:89:34:41:A0:40:C8:B7:92:28:25:CE:25:A7
            X509v3 Authority Key Identifier:
                D5:EA:58:CC:AD:9B:52:6C:2D:94:C7:E9:50:AA:9C:BF:06:74:8D:37
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        0e:8c:dc:54:26:6f:57:60:dc:2b:ba:96:38:ca:86:6b:72:e5:
        58:75:53:f5:55:88:1e:0c:f5:e1:ce:1f:39:f3:b5:88:57:65:
        d4:4e:6c:54:fd:dd:59:3c:6c:fb:49:1f:8b:6b:43:e9:04:a1:
        bb:ef:21:97:8e:8b:e2:fb:48:86:21:ec:e2:64:bc:6a:10:4b:
        f7:b6:54:b2:0f:a3:81:cf:f3:97:04:18:3b:3c:e3:58:b9:17:
        f5:a6:bb:79:85:d3:98:0f:03:07:81:55:5e:46:8d:72:4d:35:
        5b:76:34:23:2a:d9:b2:9f:a7:44:e8:ef:80:23:bf:5a:33:96:
        57:e5:01:7e:0d:fa:18:21:99:76:db:eb:5d:84:aa:ea:cc:da:
        13:b4:23:a3:60:a9:72:cf:ad:9b:e5:34:cd:cb:88:d8:9a:e6:
        30:4f:97:1d:9e:68:03:0d:2e:da:35:47:5b:ff:01:03:ac:14:
        c7:43:26:f9:b7:06:cc:9f:2b:d0:ce:7d:7d:c6:6e:4b:88:9d:
        a3:13:84:21:44:6b:81:b2:94:5f:0d:4d:96:0b:74:df:9c:7b:
        09:d3:6b:b8:5a:89:30:be:df:9f:a4:9c:3b:0c:2f:ff:dd:df:
        0e:af:2e:58:fe:2a:51:90:b0:03:d5:55:1b:98:a8:c4:1e:15:
        e8:72:08:80:49:90:d0:7a:f8:46:63:6a:dd:48:1d:93:37:3b:
        3c:64:1c:13:93:02:04:3e:09:f8:49:af:4e:b1:4f:69:6b:b9:
        bb:a3:32:3f:eb:88:0b:27:fa:68:5b:cb:7d:92:86:3c:f7:b8:
        6b:af:4c:f2:0a:5c:e6:c5:9e:18:b1:2a:51:34:57:35:02:0e:
        db:fd:c0:ec:0b:3e:ff:a5:71:79:70:e4:47:47:c5:49:f7:b5:
        df:ea:0e:e5:78:45:6d:14:82:f6:c6:3f:4a:4b:db:25:78:26:
        e4:66:a7:6f:c1:57:5f:ec:35:d2:30:2a:8e:d5:e3:f6:c8:84:
        78:19:15:d9:5d:ce:f1:eb:36:66:05:9e:3c:c5:fd:c2:fd:2c:
        ae:c3:93:e1:26:7d:0d:b4:86:68:03:0a:c4:2c:95:06:ce:07:
        5c:07:06:7f:d7:1f:ad:94:32:71:8e:9d:b1:97:d9:5d:d0:19:
        96:65:2d:b9:e2:0d:1c:b8:6c:a0:83:7c:e2:15:23:86:e5:8d:
        ec:47:bc:ae:f7:15:09:b4:91:f3:34:e7:4d:80:f5:4b:c2:92:
        7a:02:d0:2f:cf:5c:fd:60:bb:4c:1e:81:9a:6e:60:57:60:4e:
        3b:e7:dc:2b:e6:4d:fb:fe:0c:12:a1:24:6b:f3:53:45:14:c8:
        45:05:d2:d8:9f:90:ce:9e
```

- c. Client cert with JumpCloud username in the common name



```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            4d:07:77:17:83:ee:c3:02:d3:5c:6c:31:05:7e:de:6f:97:ae:d2:4b
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: O = JumpClout Test, CN = Test Intermediate 1
        Validity
            Not Before: Nov 14 22:43:57 2022 GMT
            Not After : Dec 14 22:43:57 2022 GMT
        Subject: C = US, ST = Colorado, L = Boulder, O = MyOrg, OU = Unit#1, CN = UserRoger
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
                    00:d5:f3:fb:c0:bb:7f:ad:29:b0:2c:e6:64:85:be:
                    2b:c0:1d:02:76:2d:13:a9:ce:83:28:bb:44:cb:1b:
                    4d:69:a6:f1:8c:49:6f:0c:a7:b4:e5:80:19:cd:e7:
                    ea:2e:f3:58:be:7f:43:2b:bf:ff:00:22:bb:3b:6e:
                    31:9d:d9:b7:a4:58:28:54:5e:b3:b8:dd:cc:21:32:
                    46:9c:dd:fd:5c:ab:0f:ad:19:2f:e5:58:b1:7e:44:
                    a6:7a:51:26:b7:0d:5c:91:3d:0e:29:24:d7:1b:ef:
                    48:0b:b0:b9:80:ac:b3:04:a7:68:c4:7a:ea:9f:9e:
                    a5:52:24:39:aa:41:15:5f:1b:7a:a5:12:0c:68:dc:
                    b5:e2:63:83:e2:67:73:49:1e:f8:2d:41:fe:0d:31:
                    a2:ae:ab:98:ae:8f:98:37:c3:3c:09:5b:74:7c:25:
                    d9:ad:61:ae:12:40:7d:b4:1c:5a:a6:97:77:a7:ac:
                    68:1b:0e:5d:46:0f:df:d7:f4:21:24:57:a0:0c:a9:
                    66:67:b1:5f:41:ac:5d:c0:ec:9e:14:c7:87:4e:22:
                    ba:60:82:d6:9a:88:58:cc:77:7f:ca:e0:da:bc:46:
                    fa:cd:1a:5d:75:84:2a:4d:4e:fc:1f:a8:cf:a7:d2:
                    fb:6c:69:73:b7:65:22:44:61:20:2e:94:27:00:24:
                    cd:ac:de:03:e5:ee:9e:3c:77:0a:cd:87:5d:a2:e7:
                    ef:d1:cb:b6:4e:1c:f1:6b:2a:7c:ba:e4:94:bf:c9:
                    74:35:21:ff:51:d2:4a:ab:8e:8c:2b:97:4e:74:4a:
                    96:db:16:e1:ce:6f:ae:79:1a:56:2c:c9:b2:7e:af:
                    e9:a4:48:a3:bf:14:cb:3d:a7:fc:c3:8d:53:b9:4f:
                    bf:d9:11:0e:22:cd:9d:80:ed:a2:27:a3:41:95:8e:
                    6f:dc:22:bc:ba:d8:6c:a5:a4:00:1e:75:0f:e6:91:
                    b2:db:4a:0b:c5:f2:bd:41:51:1d:0c:c4:cc:cc:d9:
                    3e:4c:39:3f:bf:87:79:17:54:d8:05:46:eb:85:af:
                    76:24:28:22:13:c9:6c:e0:a7:25:39:6d:09:3d:62:
                    fc:f4:6b:64:bd:e5:97:c2:73:ee:58:45:7b:9f:d0:
                    3d:7f:1e:8a:bd:6e:0f:79:dc:c3:ca:4d:bd:6e:74:
                    28:5e:9e:67:27:e2:49:35:3b:9f:15:58:64:5a:d2:
                    6d:90:77:4f:f5:61:f6:cf:61:1d:33:25:6d:95:3f:
                    e9:2e:fa:80:0a:22:74:c2:68:c7:36:f3:19:71:03:
                    ff:c8:aa:97:c5:9c:cf:65:08:0d:8e:71:2d:b8:e8:
                    ce:46:4b:78:0b:c0:0b:d1:0c:b0:a2:a1:ac:79:5b:
                    ea:df:df
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage:
                Digital Signature
            X509v3 Extended Key Usage:
                TLS Web Client Authentication
            Authority Information Access:
                OCSP - URI:http://localhost:9000
            X509v3 Subject Key Identifier:
                1B:85:C2:30:47:72:13:89:34:41:A0:40:C8:B7:92:28:25:CE:25:A7
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        8a:10:b0:26:c8:3c:41:22:79:f0:11:8c:88:4a:2e:20:3c:b3:
        7a:51:e8:e7:d7:4b:99:e3:11:4a:6c:1f:ea:a5:18:d2:b6:9e:
        e7:5f:71:11:f8:a7:84:13:99:f2:72:e8:6c:c9:65:61:ea:56:
        f1:4b:dc:d5:25:6c:b2:dc:f3:bf:2a:aa:34:e1:cc:e6:aa:90:
        55:af:be:9b:6b:66:a4:36:f8:52:6f:4e:c1:a7:70:66:a1:32:
        10:0a:78:03:35:74:5b:9c:7f:32:e8:d9:53:f3:d4:de:82:b8:
        d3:d2:8a:0f:a2:78:ec:7e:3b:c9:b1:bf:b1:d1:e3:95:9c:2d:
        90:43:42:79:a6:65:b0:b6:40:32:31:a8:e8:fe:0e:bd:3d:7c:
        2d:6c:0a:66:c2:02:84:8d:ea:fb:7e:c2:30:f3:42:d1:44:05:
        1a:31:0f:e1:b7:aa:bb:39:1e:5d:1b:73:9f:69:a1:63:e9:b1:
        f8:62:2b:37:1b:92:cc:54:1a:00:c6:df:13:3c:6c:dd:f6:27:
        6d:e1:13:0f:94:31:ef:31:18:d0:68:da:01:95:b5:44:bf:b2:
        d9:6e:bb:3c:ba:86:16:d0:c8:04:4f:3a:4a:9e:95:7e:d9:5d:
        92:63:7a:0d:ba:f8:1f:3c:f6:56:6f:c4:47:0f:62:fd:ff:51:
        88:f4:71:8c:80:21:f9:ce:28:e3:19:96:67:82:ca:bc:9b:15:
        eb:1f:21:09:44:5c:77:74:f6:f3:71:e3:80:6b:b2:fa:c5:4a:
        7b:2f:0a:2c:6b:01:d9:b5:97:67:29:58:fb:94:28:e9:8d:1a:
        ba:35:68:1e:32:e6:6f:7e:e6:32:23:f6:92:f2:99:76:2f:4d:
        19:bb:e2:47:f4:20:a7:bd:f8:fc:e6:84:e6:8c:a4:8f:f1:5c:
        9a:c1:01:b8:30:7a:b3:09:67:d7:a4:0d:3e:91:84:49:df:f9:
        bf:a5:bf:07:bc:d7:73:3b:40:0d:3a:9d:36:79:0d:f4:56:70:
        81:fd:9d:c1:9c:cc:6c:95:0e:f1:7d:01:91:a2:fd:2d:49:b8:
        87:7a:2b:22:26:0d:9f:41:0b:51:13:85:90:96:f6:eb:31:8d:
        12:94:01:aa:49:2f:60:b5:af:b2:c6:c9:7f:6e:cf:6b:1b:eb:
        08:fe:be:25:74:57:8c:ee:75:e0:e0:7c:d8:bf:f1:f8:a0:19:
        57:84:67:46:23:7a:05:62:b1:63:62:58:eb:d9:45:7a:ab:e6:
        37:71:b3:9e:f7:08:05:51:6b:4b:a7:3e:95:ac:84:a5:37:29:
        5c:0f:50:4c:3f:a6:d4:e5:9a:f9:6e:24:7f:0b:3e:fa:67:20:
        33:85:0a:e0:be:dd:cd:7a
```

Self Signed Certificate Manual Creation process using OpenSSL

The above sample certificates were created using the OpenSSL commands located on Sample SSLcommands or listed below:

a. Create self signed root certificate
   - openssl req -x509 -newkey rsa:2048 -days 365 -keyout selfsigned-ca-key.pem -out selfsigned-ca-cert.pem -subj "/C=US/ST=Colorado/L=Boulder/O=JumpCloud CA/OU=ITDept/CN=Test Intermediate 1"

b. Create certificate request
   - openssl req -nodes -new -key selfsigned-ca-key.pem -out cert-req.csr -subj "/C=US/ST=Virginia/L=Leesburg/O=MyOrg/OU=IT Department/CN=MyOrg.com"

c. Create certificate for specific user email in the subject distinguished name
   - openssl genrsa -out basicDemo-subject-email-client-signed.key 2048
   - openssl req -new -key basicDemo-subject-email-client-signed.key -out emailDN.csr -config extensions-emailDN.cnf -subj "/C=US/ST=Virginia/L=Leesburg/O=MyOrg/OU=Sales-Radius-Access/CN=/emailAddress=roger_q@yahoo.com"
   - openssl x509 -req -in emailDN.csr -CA selfsigned-ca-cert.pem -CAkey selfsigned-ca-key.pem -days 30 -CAcreateserial -out basicDemo-subject-email-client-signed-cert.crt -extfile extensions-emailDN.cnf
   - openssl pkcs12 -export -out basicDemo-subject-email-client-signed.pfx -inkey basicDemo-subject-email-client-signed.key -in basicDemo-subject-email-client-signed-cert.crt -passout "pass:password"
     - Contents of extensions-emailDN.cnf configuration file:

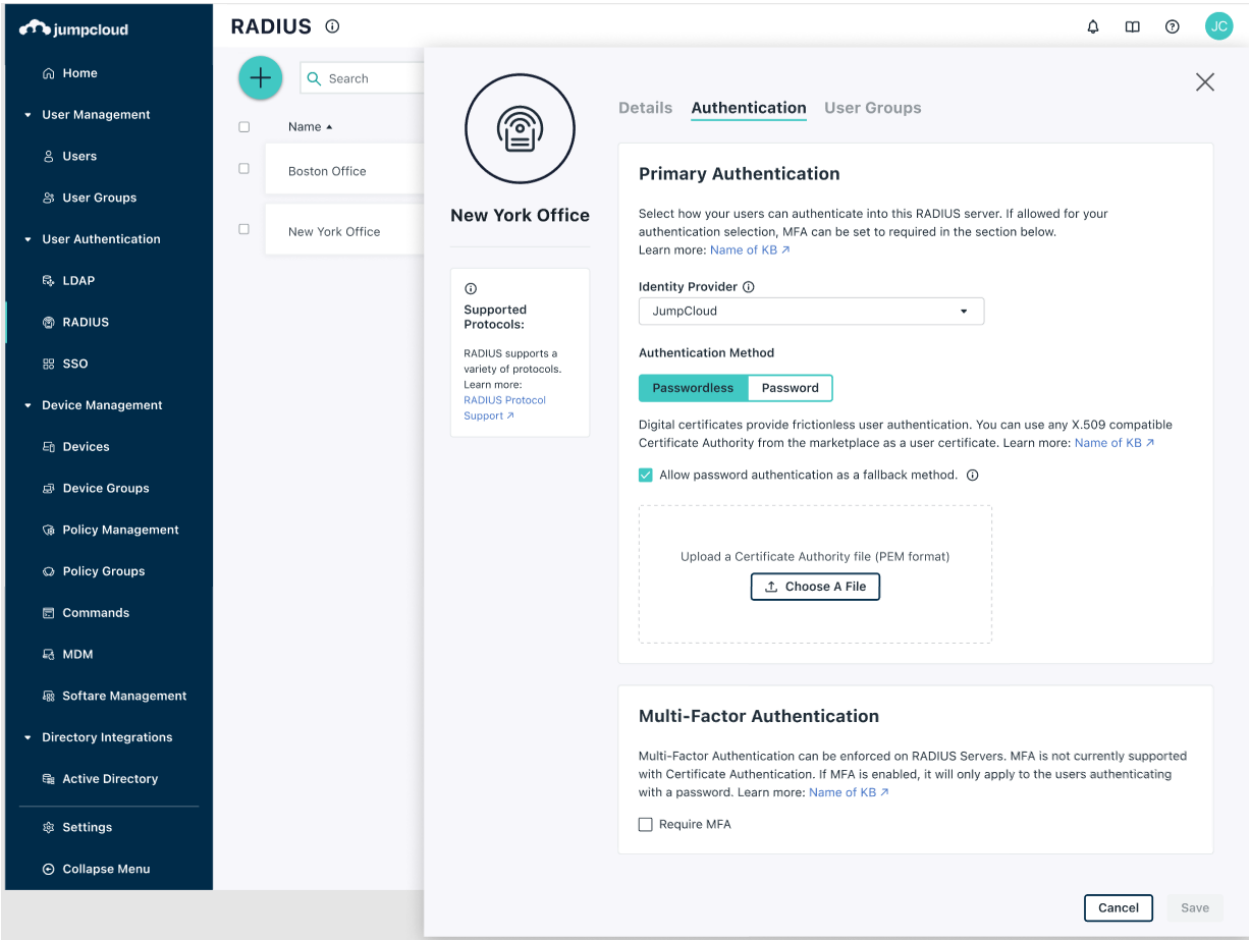
       extensions = v3_req
       [req]
       distinguished_name = req_distinguished_name
       [req_distinguished_name]
       [v3_req]
       # Useful for debugging with DI events
       authorityKeyIdentifier = keyid:issuer
       keyUsage = digitalSignature
       extendedKeyUsage = clientAuth

d. Create certificate for specific username in the Common Name
   - openssl genrsa -out basicDemo-username-client-signed.key 2048
   - openssl req -new -key basicDemo-username-client-signed.key -out usernameCN.csr -config extensions-usernameCN.cnf -subj "/C=US/ST=Virginia/L=Leesburg/O=MyOrg/OU=Sales-Radius-Access/CN=UserRoger"
   - openssl x509 -req -in usernameCN.csr -CA selfsigned-ca-cert.pem -CAkey selfsigned-ca-key.pem -days 30 -CAcreateserial -out basicDemo-username-client-signed-cert.crt -extfile extensions-usernameCN.cnf

- openssl pkcs12 -export -out basicDemo-username-client-signed.pfx -inkey basicDemo-username-client-signed.key -in basicDemo-username-client-signed-cert.crt -passout "pass:password"
  - Contents of extensions-usernameCN.cnf configuration file:
    extensions = v3_req
    [req]
    distinguished_name = req_distinguished_name
    [req_distinguished_name]
    [v3_req]
    # Useful for debugging with DI events
    authorityKeyIdentifier = keyid:issuer
    keyUsage = digitalSignature
    extendedKeyUsage = clientAuth

e. Create certificate for specific user email in the Subject Alternative Name
   - openssl x509 -req -in emailSAN.csr -CA selfsigned-ca-cert.pem -CAkey selfsigned-ca-key.pem -days 30 -CAcreateserial -out basicDemo-SAN-client-signed-cert.crt -extfile extensions-emailSAN.cnf
   - openssl pkcs12 -export -out basicDemo-SAN-client-signed.pfx -inkey basicDemo-SAN-client-signed.key -in basicDemo-SAN-client-signed-cert.crt -passout "pass:password"
     - Contents of extensions-emailSAN.cnf configuration file:
       extensions = v3_req
       [req]
       distinguished_name = req_distinguished_name
       [req_distinguished_name]
       [v3_req]
       # Useful for debugging with DI events
       authorityKeyIdentifier = keyid:issuer
       keyUsage = digitalSignature
       extendedKeyUsage = clientAuth
       #For Client cert with email in the subject alternative name
       subjectAltName = email:roger_q@yahoo.com

Appendix - Admin User Interface



Appendix – Manual Certificate deployment

The RADIUS-CBA User Certificates must be installed on the user or local store of the target devices.
An example of a manual deployment of RADIUS CBA certificates in Windows 10 is accomplished by the indicated manual steps and scripts below.

- **Microsoft Windows**
    a. To install double click on PFX file (ex. basicDemo-SAN-client-signed.pfx) and import on “Current User | Personal Store”
    b. The RADIUS CBA certificate is assumed to have either the Username or email embedded in the certificate as indicated on the Appendix Certificate compatible with RADIUS-CBA
    c. Script to manually installs the RADIUS CBA certificate in the CurrentUser store
        - certutil -user -p pass: -importPFX basicDemo-SAN-client-signed.pfx

Appendix – Authors

- Author: Roger Quint

Appendix – References

- RADIUS CBA - Feature Bulletin Blog

Appendix – Disclaimer

- Disclaimer - The technologies and vendors provided in this document are only a portion of the providers available in the marketplace