Cisco Email Security용 Azure AD 구성 스크립트

목차

[소개](#)

[사전 요구 사항](#)

- [요구 사항](#)
- [사용되는 구성 요소](#)

[Cisco Email Security용 Azure AD 구성 스크립트](#)

[관련 정보](#)

소개

이 문서에서는 UNIX/Linux 환경에서 실행할 수 있는 스크립트를 제공하여 자체 서명 인증서를 만드는 데 사용되는 프로세스와 Cisco Email Security 구성에 필요한 경우 필요한 Microsoft Azure 단계를 간소화합니다. 이 스크립트는 MAR(Mailbox Auto Remediation), Microsoft Office 365 LDAP 커넥터 또는 Cisco Threat Analyzer for Office 365에 사용할 수 있습니다. 이 스크립트는 독립적이며 모든 버전의 AsyncOS for Email Security Appliance(ESA)에서 사용할 수 있습니다.

참고: 이 문서는 개념 증명이며 예시 자료로 제공됩니다. 이러한 단계가 성공적으로 테스트되었지만 이 문서는 주로 데모 및 설명을 목적으로 합니다. 맞춤형 스크립트는 Cisco의 범위 및 지원 가능성 밖에 있습니다. Cisco TAC(Technical Assistance Center)는 언제든지 외부 스크립트를 작성, 업데이트 또는 트러블슈팅하지 않습니다. 스크립트를 시도하고 구성하기 전에 최종 스크립트를 구성할 때 스크립팅 지식이 있는지 확인하십시오.

참고: Cisco TAC 및 Cisco Support는 Microsoft Exchange, Microsoft Azure AD 또는 Office 365와 관련된 고객 측 문제를 해결할 수 없습니다.

사전 요구 사항

요구 사항

Cisco에서는 Azure AD 및 Office 365 사서함 설정을 읽고 ESA에 대해 구성하는 방법을 이해할 것을 권장합니다.

사용되는 구성 요소

이 문서는 특정 소프트웨어 및 하드웨어 버전으로 한정되지 않습니다.

이 스크립트의 목적과 실행에는 OpenSSL이 설치되어 있다고 가정합니다. 터미널 프롬프트에서 openssl 또는 openssl 버전을 실행하여 설치를 확인합니다.

이 글의 목적을 위해, 스크립트는 my_azure.sh로 호출되어 실행됩니다. 원하는데로 대본의 이름을

지정하세요.

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 미리 숙지하시기 바랍니다.

Cisco Email Security용 Azure AD 구성 스크립트

외부 호스트(UNIX/Linux)에서 스크립트를 만들고 다음 텍스트를 복사하여 붙여 넣습니다.

```
clear
echo "################################################################################
      my_azure.sh by Robert Sherwin (robsherw@cisco.com)  ©2018 Cisco .:|:.:|:.
Using openssl, this script will create a self-signed certificate for you to use in
order to complete the Mailbox Settings configuration for Cisco Email Security.
Please respond to the following prompts:
################################################################################
"
if which openssl >/dev/null; then
    echo "openssl check passed: openssl is installed!" & openssl version
else
    echo "You do not appear to have openssl installed." && exit
fi

echo "
Please enter a name for your cert: "
read my_cert

while [ -f $my_cert.key ];
do
    echo "File exists, please enter a name for your cert: " && read my_cert
done

echo "
Thank you.  The files that will be generated for your cert are: "

crt=$my_cert.crt
key=$my_cert.key
pem=$my_cert.pem

echo $crt
echo $key
echo $pem
echo ""

while true; do
```

```
    read -p "Are you ready to proceed and generate these files for your configuration? $(tput
smso)(y/n)$(tput sgr0) " yn
    case $yn in
        [Yy]* ) openssl req -x509 -sha256 -nodes -days 1825 -newkey rsa:2048 -keyout $key -out
$crt
openssl rsa -in $key -out $key
cat $key $crt > $pem

echo ""
base64Thumbprint=`openssl x509 -outform der -in $crt | openssl dgst -binary -sha1 | openssl
base64`
base64Value=`openssl x509 -outform der -in $crt | openssl base64 -A`
keyid=`python -c "import uuid; print(uuid.uuid4())"`
echo "
################################################################################
Next, $(tput smul)copy$(tput rmul) the following to Azure for your manifest:
################################################################################
"
echo "\"keyCredentials\": [
{
\"customKeyIdentifier\": \"$base64Thumbprint\",
\"keyId\": \"$keyid\",
\"type\": \"AsymmetricX509Cert\",
\"usage\": \"Verify\",
\"value\": \"$base64Value\"
}
],"
echo "
################################################################################
Then $(tput smul)complete$(tput rmul) the Azure configuration to get the $(tput smso)Client
ID$(tput sgr0) and $(tput smso)Tenant ID$(tput sgr0).
################################################################################
"
echo "This is the $(tput smso)Thumbprint$(tput sgr0) for your ESA configuration:
$base64Thumbprint"
echo "This is the $(tput smso)Certificate Private Key$(tput sgr0) for your ESA configuration:
$pem
"; break;;
        [Nn]* ) exit;;
        * ) echo "Please answer yes or no.";;
    esac
done
while true; do
    read -p "Do you wish to review this certificate in detail? $(tput smso)(y/n)$(tput sgr0) "
yn
    case $yn in
```

```
        [Yy]* ) openssl x509 -in $crt -text; echo "
Thank you!" && break;;
        [Nn]* ) echo "Thank you!" && exit;;
        * ) echo "Please answer yes or no.";;
    esac
done
```

팁: 스크립트를 작성했으면 chmod u+x <script_name> 을 입력하여 스크립트를 실행할 수 있도록 합니다.

실행 중인 스크립트의 전체 예는 다음과 같습니다.

```
my_host$ ./my_azure
################################################################################
      my_azure.sh by Robert Sherwin (robsherw@cisco.com)  ©2018 Cisco .:|:.:|:.
Using openssl, this script will create a self-signed certificate for you to use in
order to complete the Mailbox Settings configuration for Cisco Email Security.
Please respond to the following prompts:
################################################################################

openssl check passed: openssl is installed!
LibreSSL 2.2.7

Please enter a name for your cert:
technote_example

Thank you.  The files that will be generated for your cert are:
technote_example.crt
technote_example.key
technote_example.pem

Are you ready to proceed and generate these files for your configuration? (y/n) y
Generating a 2048 bit RSA private key
.....................................................................+++
.........................................+++
writing new private key to 'technote_example.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) []:US
State or Province Name (full name) []:North Carolina
```

```
Locality Name (eg, city) []:RTP
Organization Name (eg, company) []:Cisco
Organizational Unit Name (eg, section) []:Example Dept.
Common Name (eg, fully qualified host name) []:example.local
Email Address []:joe.user@example.local
writing RSA key


################################################################################
Next, copy the following to Azure for your manifest:
################################################################################

"keyCredentials": [
{
"customKeyIdentifier": "wWHhkWEfuhDHTXPzzmHoSEnjbNM=",
"keyId": "338836b8-fc8d-4e1b-9a3f-b252f8368d34",
"type": "AsymmetricX509Cert",
"usage": "Verify",
"value":
"MIIDtDCCApwCCQDV3bbiHmaN2jANBgkqhkiG9w0BAQsFADCBmzELMAkGA1UEBhMCVVMxFzAVBgNVBAgMDk5vcnRoIENhcm9saW5hMQ
}
],

################################################################################
Then complete the Azure configuration to get the Client ID and Tenant ID.
################################################################################

This is the Thumbprint for your ESA configuration: wWHhkWEfuhDHTXPzzmHoSEnjbNM=
This is the Certificate Private Key for your ESA configuration: technote_example.pem
```

스크립트는 인증서를 자세히 검토하라는 메시지를 표시합니다. 스크립트를 완료하려면 y 또는 n을 입력합니다.

```
Do you wish to review this certificate in detail? (y/n) y
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 15410674582220606938 (0xd5ddb6e21e668dda)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=North Carolina, L=RTP, O=Cisco, OU=Example Dept.,
CN=example.local/emailAddress=joe.user@example.local
        Validity
            Not Before: Oct 18 02:00:49 2018 GMT
            Not After : Oct 17 02:00:49 2023 GMT
        Subject: C=US, ST=North Carolina, L=RTP, O=Cisco, OU=Example Dept.,
CN=example.local/emailAddress=joe.user@example.local
        Subject Public Key Info:
```

```
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:a9:58:99:6e:c3:37:e0:31:71:94:1c:a5:cf:21:
                    66:19:af:f7:2a:8c:1e:e9:76:72:35:77:1b:4f:3c:
                    9a:41:ad:45:95:39:29:45:4d:29:96:52:98:c9:67:
                    cb:79:4e:2a:0e:9c:4e:ee:04:cf:85:2e:8a:0c:c2:
                    ff:62:57:11:fd:fe:c0:e8:fd:60:28:4a:f7:66:c4:
                    61:68:d8:b0:a7:99:b5:b2:28:a9:84:5f:1c:4f:92:
                    93:e6:ec:25:be:46:a6:2c:d7:80:f7:18:64:68:de:
                    f3:57:9c:81:a9:a1:0e:b8:3b:35:9a:ed:84:f4:d2:
                    29:ae:19:c6:66:30:a5:09:7a:c4:60:eb:32:2a:68:
                    94:6a:04:35:ff:9e:c8:d0:a8:e5:5c:80:5e:5c:6e:
                    60:7f:26:ea:dd:06:74:fc:3e:54:a1:c9:ee:4f:b8:
                    c0:8f:4a:4d:4c:38:2c:00:68:39:6b:3c:85:49:c3:
                    8b:4c:b3:da:4f:66:a8:db:d3:1b:eb:bb:e4:45:14:
                    32:07:13:59:cf:c8:4a:c5:e3:0b:c9:29:6c:eb:31:
                    b5:e6:48:89:4e:31:52:fa:8d:77:5b:7d:ea:27:1c:
                    8d:a7:75:f6:7e:b5:25:db:30:19:7f:82:0b:53:e5:
                    f9:96:4c:93:cf:c8:40:43:ed:6c:fa:ac:ff:8a:77:
                    72:61
                Exponent: 65537 (0x10001)
    Signature Algorithm: sha256WithRSAEncryption
         42:aa:bb:8b:10:5b:b5:f8:68:ae:b5:a4:ef:7b:82:a1:85:0f:
         46:a5:99:2c:a1:e5:82:cd:54:a4:49:e6:3e:3b:cb:66:22:26:
         63:e3:ba:92:24:7d:89:c0:d5:8c:50:f8:ec:05:be:d2:f6:20:
         de:91:ed:ea:92:96:97:b4:d4:66:98:a5:cf:88:4d:a7:4a:18:
         73:fa:a3:77:a6:82:03:c0:76:28:c9:9b:7e:1d:83:56:19:a9:
         61:65:bc:3f:bc:1b:34:ff:e2:9b:7d:75:e0:5f:f3:26:f0:55:
         9c:78:de:69:8f:4a:b2:e4:d4:53:9e:16:6f:c5:57:d8:51:57:
         e3:4f:d8:16:6f:c7:4c:7a:d7:70:71:f2:5b:2e:57:05:4f:4c:
         15:59:84:bb:e6:2f:e8:92:31:09:a1:20:8f:92:7b:8d:5e:2a:
         19:03:3e:f9:f9:fe:12:94:4f:91:51:e7:f3:8e:07:ce:0c:66:
         e3:46:d1:5b:be:3b:ae:31:ae:c8:ab:2c:f8:4d:ad:8d:62:53:
         e8:e9:83:27:8a:ee:1c:21:5d:be:19:19:be:fc:d5:27:25:67:
         d0:f5:4d:f9:cc:28:27:48:0b:33:ba:76:a1:ae:c9:dc:87:4d:
         67:7a:76:08:c5:ef:15:d6:6c:46:21:45:52:90:48:6c:ad:d5:
         62:51:51:ae
-----BEGIN CERTIFICATE-----
MIIDtDCCApwCCQDV3bbiHmaN2jANBgkqhkiG9w0BAQsFADCBmzELMAkGA1UEBhMC
VVMxFzAVBgNVBAgMDk5vcnRoIENhcm9saW5hMQwwCgYDVQQHDANSVFAxDjAMBgNV
BAoMBUNpc2NvMRYwFAYDVQQLDA1FeGFtcGxlIERlcHQuMRYwFAYDVQQDDA1leGFt
cGxlLmxvY2FsMSUwIwYJKoZIhvcNAQkBFhZqb2UudXNlckBleGFtcGxlLmxvY2Fs
MB4XDTE4MTAxODAyMDA0OVoXDTIzMTAxNzAyMDA0OVowgZsxCzAJBgNVBAYTAlVT
MRcwFQYDVQQIDA5Ob3J0aCBDYXJvbGluYTEMMAoGA1UEBwwDUlRQMQ4wDAYDVQQK
DAVDaXNjbzEWMBQGA1UECwwNRXhhbXBsZSBEZXB0LjEWMBQGA1UEAwwNZXhhbXBs
```

```
ZS5sb2NhbDElMCMGCSqGSIb3DQEJARYWam9lLnVzZXJAZXhhbXBsZS5sb2NhbDCC
ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKlYmW7DN+AxcZQcpc8hZhmv
9yqMHul2cjV3GO88mkGtRZU5KUVNKZZSmMlny3lOKg6cTu4Ez4UuigzC/2JXEf3+
wOj9YChK92bEYWjYsKeZtbIoqYRfHE+Sk+bsJb5GpizXgPcYZGje81ecgamhDrg7
NZrthPTSKa4ZxmYwpQl6xGDrMipolGoENf+eyNCo5VyAXlxuYH8m6tOGdPw+VKHJ
7k+4wI9KTUw4LABoOWs8hUnDiOyz2k9mqNvTG+u75EUUMgcTWc/ISsXjC8kpbOsx
teZIiU4xUvqNd1t96iccjad19n61JdswGX+CC1Pl+ZZMk8/IQEPtbPqs/4p3cmEC
AwEAATANBgkqhkiG9w0BAQsFAAOCAQEAQqq7ixBbtfhorrWk73uCoYUPRqWZLKHl
gs1UpEnmPjvLZiImY+O6kiR9icDVjFD47AW+OvYg3pHt6pKWl7TUZpilz4hNpOoY
c/qjd6aCA8B2KMmbfh2DVhmpYWW8P7wbNP/im3114F/zJvBVnHjeaY9KsuTUU54W
b8VX2FFX4O/YFm/HTHrXcHHyWy5XBU9MFVmEu+Yv6JIxCaEgj5J7jV4qGQM++fn+
EpRPkVHn844Hzgxm4ObRW747rjGuyKss+E2tjWJT6OmDJ4ruHCFdvhkZvvzVJyVn
OPVN+cwoJOgLM7p2oa7J3IdNZ3p2CMXvFdZsRiFFUpBIbK3VYlFRrg==
-----END CERTIFICATE-----

Thank you!
```

현재 .crt, .key 및 .pem의 세 가지 파일이 있습니다.

지침에 따라 keyCredentials 출력을 사용하고 앱 등록을 설정할 때 출력을 Azure에 복사합니다. Cisco Email Security에서 컨피그레이션 단계를 실행할 때 지문 출력 및 인증서 개인 키(.pem)가 필요합니다.

관련 정보

- Cisco Email Security Appliance – 엔드 유저 가이드
- 기술 지원 및 문서 – Cisco Systems

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.