

AWS 활용 IoT

[5] AWS IoT : 보안 터널링



강사 : 고병화

Secure tunneling

Secure tunneling : 보안 터널링 소개

AWS IoT 보안 터널링은 고객이 AWS IoT에서 관리하는 보안 연결을 통해 원격 장치에 대한 양방향 통신을 설정할 수 있도록 지원한다. 보안 터널링은 기존 인바운드 방화벽 규칙을 업데이트할 필요가 없으므로 원격 사이트에서 방화벽 규칙이 제공하는 것과 동일한 보안 수준을 유지할 수 있다.

실습에서는 보안 터널링을 사용하여 Amazon EC2 인스턴스로 ssh 로 접속하여 사용한다

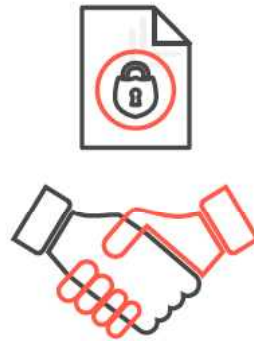


Secure Tunneling AWS IoT Device Management

Provides secure connectivity to individual devices in just a few clicks to diagnose issues and take action to solve them.



Gain remote access to devices on isolated networks or behind firewalls



Establish trusted connections that adhere to customers' corporate security policies



Troubleshoot and solve device issues more quickly and cost-effectively, with no disruption to end user experience



Connectivity
& Control
Services



Secure tunneling : 동작 구조

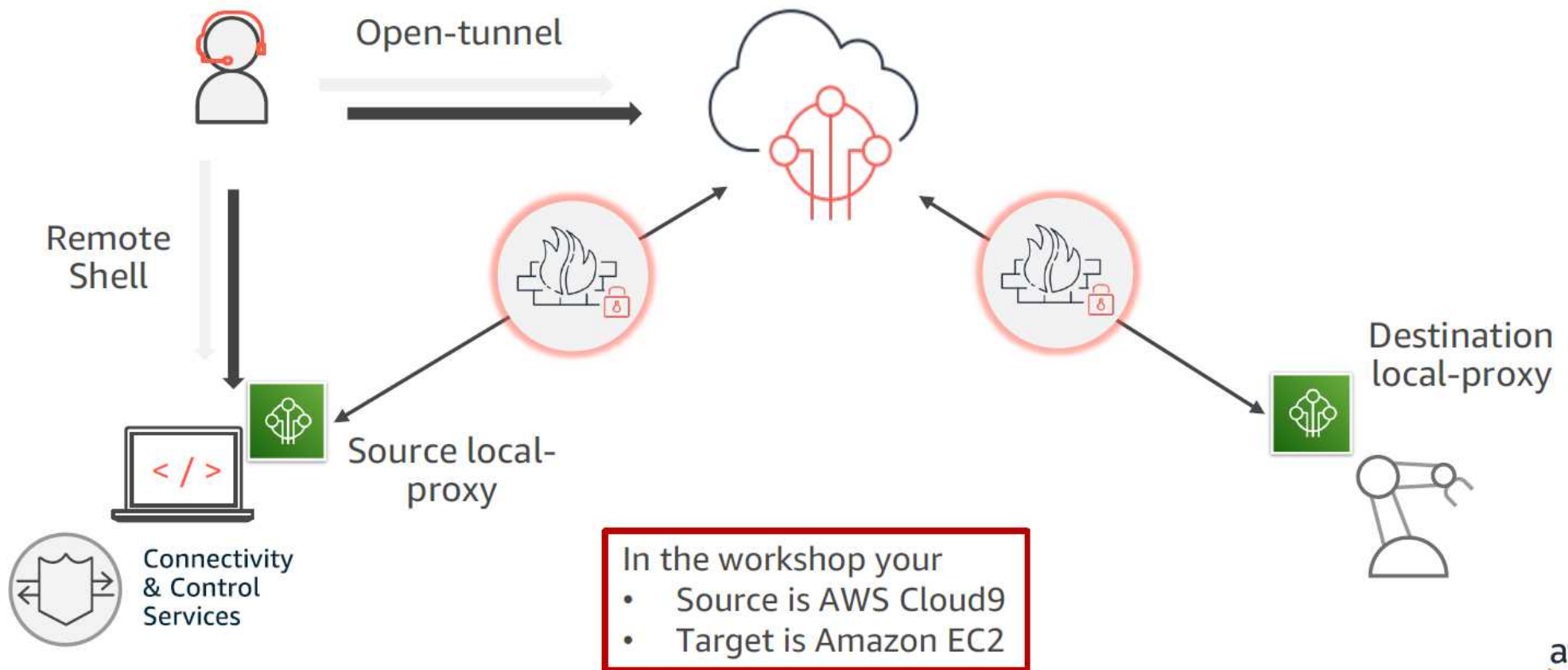
AWS IoT 보안 터널링에는 **소스 디바이스** 와 **대상 디바이스** 가 있다.

두 장치 모두 **로컬 프록시** 가 설치되어 있습니다. 로컬 프록시는 보안 터널링 서비스와 장치 애플리케이션 간에 데이터 스트림을 중계한다. 로컬 프록시는 소스 또는 대상 모드에서 실행할 수 있다.

실습에서 **소스 디바이스**는 **AWS Cloud9 인스턴스**이고 **대상 디바이스**는 **Amazon EC2 인스턴스**이다.



Secure Tunneling



Secure tunneling : 동작 구조

대상 장치(EC2)에서 **리스너 에이전트**가 이미 배포되어 시작되어 있다. 리스너 에이전트는 AWS IoT Core에 대한 **MQTT 연결을 유지하고 해당 새도우 주제를 구독**하는 디바이스이다. 리스너 에이전트의 장치 이름은 **tunneling-listener-agent**

리스너 에이전트는 로컬 프록시 실행에 대한 상태를 아래 주제로 게시한다 **“cmd/sectunnel/tunneling-listener-agent/resp”**

[장치 검색 명령] : aws iot search-index --query-string
"thingName: tunneling-listener-agent "

Secure tunneling : 동작 구조

소스 장치에서 터널 관리자를 사용하여 터널을 연다. 터널 관리자가 시작되면 터널이 생성된다. 터널이 생성되면 한 쌍의 토큰인 CAT(클라이언트 액세스 토큰)이 생성된다. 터널 관리자는 대상 장치에서 리스너 에이전트의 새도우에 연결할 토큰과 원하는 서비스를 게시한다.

리스너 에이전트는 CAT를 수신하면 로컬 프록시를 시작하여 AWS IoT 보안 터널링 연결을 설정한다.

터널 관리자는 소스 장치에서 로컬 프록시도 시작 시킨다.

터널에 연결하는 데 사용되는 로컬 장치에서 포트가 열린다.

Secure tunneling : 준비

Cloud9 에서 사용할 디바이스를 검색하여 본다

```
aicore0427:~ $ aws iot search-index --query-string  
"thingName:tunneling-listener-agent"
```

```
{  
  "things": [  
    {  
      "thingName": "tunneling-listener-agent",  
      "thingId": "4d7de37b-686b-41b0-86ea-0763d1cdf07f",  
      "shadow":  
        "{\\\"reported\\\":{\\\"status\\\":\\\"ready\\\"},\\\"metadata\\\":{\\\"reported\\\":{\\\"status\\\":  
        {\\\"timestamp\\\":1663132210}}},\\\"hasDelta\\\":false,\\\"version\\\":3}  
        {  
          \"connectivity\": {  
            \"connected\": false,  
            \"timestamp\": 1663144173167,  
            \"disconnectReason\": \"CLIENT_INITIATED_DISCONNECT\"  
          }  
        }  
      ]  
    }  
  ]  
}
```

Secure tunneling : 터널 열기

디렉토리를 변경한다

```
cd ~/secure-tunneling
```

터널을 만들고 CAT(client access token)를 리스너 에이전트의
새도우에 게시 한다

```
./tunnel-manager.py -e localhost:22 -p 2333 \  
  --local-proxy ./localproxy_amzn_x86_64 \  
  --region-iot $REGION \  
  --region-tun $REGION \  
  -t '$aws/things/tunneling-listener-agent/shadow/update' -l 60
```

Secure tunneling : 명령 줄 매개변수

다음은 터널 매니저 소스 실행 시 사용되는 매개변수들이다

- e localhost 22** : 대상 장치의 터널이 연결되어야 하는 endpoint
실습에서는 localhost에서 실행되는 ssh 데몬이 된다
- p 2333** : 터널에 액세스하기 위한 소스 장치의 포트
- local-proxy ./localproxy_amzn_x86_64** : 사용할 로컬 프록시.
Python으로 작성된 터널 관리자는 다양한 시스템 아키텍처 또는 운영 체제에서 쉽게 사용할 수 있다.

Secure tunneling : 소스 파라미터

--region-iot \$REGION : 리스너 에이전트가 연결된 AWS 리전

--region-tun \$REGION : 터널을 생성하려는 AWS 리전. 터널 관리자를 사용하면 리스너 에이전트가 연결된 다른 지역의 터널을 사용할 수 있다.

-t '\$aws/things/tunneling-listener-agent/shadow/update' : 터널 관리자가 리스너 에이전트에 대한 CAT 및 엔드포인트를 게시해야 하는 주제(topic)

-l 60 : 터널 지속 시간(분)

Secure tunneling : 생성된 터널 확인

tunnel-manager.py 가 실행되면 아래와 같이 생성된 **터널의 ID**를
로깅 메시지에서 확인할 수 있다

```
2022-09-16 06:50:02,356 INFO: tunnel-manager.py:149 - <module>:  
calling open_tunnel  
response: {'tunnelId': '7c36ca55-9bcb-42ea-bcf4-ddc31e0f525c',  
'tunnelArn': 'arn:aws:iot:us-east-1:844311781633:tunnel/7c36ca55-  
9bcb-42ea-bcf4-ddc31e0f525c', 'sourceAccessToken':  
...
```

Secure tunneling : 생성된 터널 확인

Cloud9에서 새 터미널을 열고 아래 명령어로 새로 생성된 터널을 확인해본다

aws iotsecuretunneling list-tunnels

```
aicore0427:~ $ aws iotsecuretunneling list-tunnels
```

```
{
  "tunnelSummaries": [
    {
      "tunnelId": "7c36ca55-9bcb-42ea-bcf4-ddc31e0f525c",
      "tunnelArn": "arn:aws:iot:us-east-1:844311781633:tunnel/7c36ca55-9bcb-42ea-bcf4-ddc31e0f525c",
      "status": "OPEN",
      "createdAt": "2022-09-16T06:50:02.419000+00:00",
      "lastUpdatedAt": "2022-09-16T06:50:02.419000+00:00"
    }
  ]
}
```

Secure tunneling : shadow 값 확인

장치의 shadow 값 확인

```
aws iot-data get-thing-shadow --thing-name tunneling-listener-agent  
tunneling-listener-agent-shadow.json
```

jq JSON프로세서로 JSON 파일 내용을 출력해서 확인해본다
jq " tunneling-listener-agent-shadow.json

```
"state": {  
  "desired": {  
    "tunnel": "start",  
    "tunnel_lifetime": 60,  
    "endpoint": "localhost:22",  
    "region": "us-east-1",
```

```
"reported": {  
  "status": "ready"  
},  
"delta": {  
  "tunnel": "start",  
  "tunnel_lifetime": 60,  
  "endpoint": "localhost:22",  
  "region": "us-east-1",
```

Secure tunneling : 터널 Open 비용

터널 생성 Open시 과금 비용 : 1회당 \$1.00

IoT Device Management, US East (N. Virginia) ✕

[새 페이지에서 보기](#) 


▼ AWS IoT Device Management USE1-TunnelsOpened 

총 요금 수

1

의 총 세전 요금 USD

USD 5.00

 사용량 설명별 필터링

< 1 >

사용량 설명 ▼

사용량 수량

금액 ▼

\$1.00 per Count for TunnelsOpened in US East (N. Virginia)

5 Count

USD 5.00

Secure tunneling : 터널 사용 하기

```
# Cloud9 터미널에서 s3버킷에서 ssh private key를 다운 받는다  
aws s3 cp s3://$S3_BUCKET/ssh/sec-tunnel.key .  
chmod 400 sec-tunnel.key
```

* AWS IoT 콘솔의 MQTT 테스트 클라이언트에서 아래 주제로
구독을 해놓는다
cmd/sectunnel/tunneling-listener-agent/resp

Secure tunneling : 터널 사용 하기

원격 EC2 인스턴스에 SSH 접속을 한다

```
ssh -i sec-tunnel.key -p 2333 ec2-user@localhost
```

아래와 같은 메시지 나오면 “yes”를 입력한다

```
aicore0427:/tmp $ ssh -i sec-tunnel.key -p 2333 ec2-user@localhost
Warning: Identity file sec-tunnel.key not accessible: No such file or directory.
The authenticity of host '[localhost]:2333 ([127.0.0.1]:2333)' can't be established.
ECDSA key fingerprint is SHA256:SD8rI9ewu010lJtJw5CFyclqo62uqLPiccitnC3TPFY.
ECDSA key fingerprint is MD5:90:60:4a:08:a1:a1:a0:3a:08:ba:5d:76:fb:f6:91:7a.
Are you sure you want to continue connecting (yes/no)? █
```


Secure tunneling : 터널 사용 하기

MQTT 테스트 클라이언트에
우측과 같은 메시지가
나타난다

```
▼ cmd/sectunnel/tunneling-listener-agent/resp
{
  "state": {
    "reported": {
      "tunnels": [
        {
          "pid": 2453,
          "remaining_minutes": 42.233333333333334,
          "status": "running"
        },
        {
          "pid": 2509,
          "remaining_minutes": 54.566666666666667,
          "status": "running"
        }
      ]
    }
  }
}
```

Secure tunneling : 로컬 프록시

로컬 프록시는 WebSocket 보안 연결을 통한 보안 터널링을 사용하여 소스 장치에서 실행 중인 애플리케이션에서 보낸 데이터를 전송한다. GitHub 에서 로컬 프록시 소스를 다운로드할 수 있다

<https://github.com/aws-samples/aws-iot-securetunneling-localproxy>

소스 모드에서 로컬 프록시는 TCP 연결을 시작하는 클라이언트 응용 프로그램과 동일한 장치 또는 네트워크에서 실행된다. 대상 모드에서 로컬 프록시는 대상 응용 프로그램과 함께 원격 장치에서 실행된다. 단일 터널은 터널 다중화를 사용하여 한 번에 최대 3개의 TCP 연결을 지원할 수 있다.

<https://docs.aws.amazon.com/iot/latest/developerguide/local-proxy.html>

Secure tunneling : 로컬 프록시

Raspberry Pi3에서 로컬 프록시 빌드 및 사용법

<https://github.com/aws-samples/aws-iot-securetunneling-localproxy#building-the-local-proxy-via-docker>

각종 라이브러리들 설치 후 아래 빌드 명령을 수행한다

```
git clone https://github.com/aws-samples/aws-iot-securetunneling-localproxy
cd aws-iot-securetunneling-localproxy
mkdir build
cd build
cmake ../
make
```

Secure tunneling : 로컬 프록시

로컬 프록시 시작

랩톱에서 터미널을 열고 소스 클라이언트 액세스 토큰을 복사한 다음 소스 모드에서 로컬 프록시를 시작하는 데 사용합니다. 다음 명령에서 로컬 프록시는 포트 5555에서 새 연결을 수신하도록 구성됩니다.

```
./localproxy -r us-east-1 -s 5555 -t source-client-access-token
```

SSH 세션 시작

다른 터미널을 열고 다음 명령을 사용하여 포트 5555의 로컬 프록시에 연결하여 새 SSH 세션을 시작합니다.

```
ssh username@localhost -p 5555
```

<https://docs.aws.amazon.com/iot/latest/developerguide/secure-tunneling-tutorial-open-tunnel.html>

The End