PoC 시스템 운영자매뉴얼

문서이력		
2016-11-17	초기작성	강호관

1. 개요

본 문서는 PoC 시스템 (특히 headend) 운영자를 위한, 시스템 전반의 설치와 관리에 대한 설명을 제공하는 것을 목적으로 한다.

본 문서에서는 PoC 시스템에서 일반 사용자를 위해 제공하는 기능에 대한 내용은 언급하지 않는다. 해당 내용들은 별도의 문서[1]에서 다룬다.

2. PoC 시스템 설치

PoC 시스템은 Remote Attestation (이하 RA) 서버와 Privacy CA (이하 PCA), 그리고 RA 서버 관리 UI 로 구성된다. 이들 각각의 설치 과정은 다음과 같다.

가) Remote Attestation Server 설치

Remote Attestation (이하 RA) 서버의 설치 과정은 다음과 같다.

1. RA 서버를 빌드하기 위한 환경을 구성한다. 시스템 상황에 따라 일부 패키지를 추가 설치해야 할 수도 있다.

   # yum groupinstall “Development Tools”

   # yum install libcurl-devel openssl-devel trousers-devel mysql-devel

[1] PoC 시스템_사용자매뉴얼.pdf

2. RA 원시 코드를 git 로 다운로드한다.

   $ git clone https://
   tde.sktelecom.com/stash/scm/racrypto/remote_attestation_crypto_lib.git

3. RA 서버 관련 원시 코드가 있는 디렉터리로 이동한다.

   $ cd remote_attestation_crypto_lib/_src/remote_attestation/OMP_RA_server

4. 설정 점검 후 빌드를 진행한다.

   $ ./bootstrap && ./configure && make

5. 빌드한 결과물을 설치한다.

   # make install

설치가 제대로 되어 있는지 확인하기 위해 다음의 항목들을 점검한다.

- 실행 파일
  - ✓ /usr/local/bin/TPServer
    - ➢ RA main 서버 프로그램
  - ✓ /usr/local/bin/ra_tool_sp
    - ➢ ThingPlug 에 장치 등록 요청을 전송하는 프로그램
  - ✓ /usr/local/bin/del_gw_sp
    - ➢ ThingPlug 에 장치 삭제 요청을 전송하는 프로그램
- 설정 파일
  - ✓ /etc/conf/tp_server.conf
  - ✓ /etc/conf/tpconf.conf

이들 이외에 다음과 같이 syslog 관련 설정 파일을 복사해야 한다.

cp attester_verifier/conf/60-attester.conf /etc/rsyslog.d/60-attester.conf

이상의 과정으로 RA 서버 관련 기본 설치가 마무리되었다.

나) Privacy CA 설치

Privacy CA (이하 PCA) 설치 과정은 다음과 같다.

1. RA 서버 설치 과정의 "설정 점검 후 빌드" 단계까지 진행
   A. privacy_ca 디렉터리에 dosign 실행 파일 생성
      ✓ CGI 방식으로 동작하는 프로그램
2. Apache 설치 확인 후 설정 변경
   A. APACHE_DIR/conf/httpd.conf 수정
      ✓ Listen Port 추가 (default: 8080)
      ✓ CGI 관련 module 활성화 (cgid module)
      ✓ PCA 관련 virtualhost 설정 include (conf/pca/pca.conf)
   B. PCA 관련 파일들 설치
      ✓ PCA 관련 설정 파일에서 지정한 위치에 다음의 파일이 존재해야 함
      ✓ dosign.cgi – dosign 실행 파일을 이름만 변경하여 복사한 것
      ✓ cert/pca.crt – PCA 서버 인증서
         ➢ 다음과 같이 인증서 내용을 출력하여 유효 기간을 확인한다.
         ➢ $ openssl x509 -noout -text -in ./cert/pca.crt

- ➢ 유효 기간이 만료되었거나 얼마 남지 않은 경우에는 인증서를 신규로 생성하여 이용한다. (Appendix 참고)

✓ key/pca.key – PCA private key

- ➢ 다음과 같이 key 내용을 출력하여 key 길이를 확인한다.
- ➢ $ openssl rsa -noout -text -in ./key/pca.key
- ➢ Key 길이가 적당하지 않은 경우 신규로 생성하여 이용한다. (Appendix 참고)

C. Syslog 관련 설정 추가

✓ PCA 관련 프로세스가 발생하는 로그를 저장하기 위해 /etc/rsyslog.d 디렉터리에 관련 설정 파일을 추가한다.

- ➢ conf/61-pca.conf

✓ rsyslogd 를 재시작한다.

D. Apache 를 재시작한다.

3. PCA 인증서 download 시험

A. $ lynx http://223.39.121.20:8080/pca/cert/pca.crt

✓ 인증서의 내용을 확인한다.

4. Sign 수행 프로그램 기본 동작 확인

A. $ lynx http://223.39.121.20:8080/pca/dosign.cgi

✓ 500 Internal Server Error 발생 확인

✓ /etc/rsyslog.d/conf/61-pca.conf 파일에서 지정한 파일의 내용을 확인

이상의 과정으로 PCA 관련 설치 작업은 마무리된다.

다) RA 서버 관리 UI 설치

RA 서버는 RA 대상 장치 목록 정보를 기반으로 동작한다. 해당 목록은 (PoC 라는 특성상) 일반 사용자에 의해서도 등록될 수 있어야 하기 때문에, 이를 위한 관리 UI 를 설치한다. 해당 UI 는 Django + Python + NGINX 기반으로 제작되었으며, DB 는 MySQL 을 이용한다.

1. 웹서버인 NGINX 를 설치한다.
   A. # yum install epel-release
   B. # yum install nginx
2. Python 2.7 을 설치한다.
   A. Yum 으로 설치되는 Python 은 예전 버전인 2.6 이므로, 직접 설치한다.
   B. # wget –no-check-certificate https://www.python.org/ftp/python/2.7.6/Python-2.7.6.tar.xz
   C. # tar xf Python-2.7.6.tar.xz
   D. # cd Python-2.7.6
   E. # ./configure
   F. # make && make altinstall
   G. # ln -sf /usr/local/bin/python2.7 /usr/local/bin/python
   H. # hash -r
3. Python 라이브러리들을 설치하기 위해 Python2.7 에 맞는 PIP 를 설치한다.
   A. # wget –no-check-certificate https://bootstrap.pypa.io/ez_setup.py
   B. # python ez_setup.py

C. # easy_install-2.7 pip

4. 웹서버와 관련된 초기화 작업을 진행한다. Install 스크립트를 실행하면 된다.

   A. # cd remote_attestation_crypto_lib/_src/remote_attestation/OMP_RA_server/web_ui/dbmanager/install

   B. # install.sh

5. MySQL 관련 설정 작업을 한다.

   A. $ mysql -u root -p

   B. mysql> CREATE DATABASE sp_db;

6. Django 관련 초기화 작업을 한다.

   A. $ cd remote_attestation_crypto_lib/_src/remote_attestation/OMP_RA_server/web_ui/dbmanager/raserver

   B. $ python manage.py createsuperuser

      Username (leave blank to use 'suser'):

      Email address:

      Password:

      Password (again):

      Superuser created successfully.

   C. $ python manage.py migrate

7. 웹 서버 설정 (서버 IP address 와 port) 확인. 필요시 수정.

   A. # vi /etc/nginx/conf.d/sp_django.conf

8. 웹 서비스를 시작한다.

A. # service nginx restart

B. # initctl restart uwsgi

9. 브라우저 (Chrome) 이용하여 해당 주소로 접속, 서비스가 동작함을 확인한다.

A. http://223.39.121.20:8000/

3. PoC 시스템 관리 작업

PoC 시스템은 앞서 설치 과정을 설명한 3 개의 구성요소들(RA 서버, PCA, 관리 UI)이 정상 동작을 함으로써 원하는 서비스를 제공할 수 있다. 따라서, 관리 작업도 이들 개별 구성요소 기준으로 설명한다.

가) RA 서버 관리 작업

RA 서버 관리 작업은 RA 서버 동작 자체에 대한 관리와 RA 서버가 이용하는 데이터에 대한 관리로 나눌 수 있다.

RA 서버 동작 자체에 대한 관리를 위해서는 RA 서버 동작 중에 발생하는 로그들을 저장하는 파일(관련 rsyslogd 설정 참조. 현재는 /var/log/tp_server.log)의 내용을 지속적으로 분석한다.

RA 서버 프로그램 자체에 대한 (재)시작/정지/상태 조회 등의 작업은 initctl 명령을 이용한다.

- 재시작(restart), 시작(start)
  - # initctl restart tpserver
- 정지
  - # initctl stop tpserver
- 상태 조회

- # initctl status tpserver

RA 서버는 동작 과정에서 이용하는 데이터는 DB 를 이용하여 저장/조회/삭제 등의 작업을 수행하며, 동작 과정에서 발생하는 로그 정보는 syslog API 를 이용하여 저장한다. 로그 정보는 생성/저장 기능만 수행하고 있는데, 이는 이들을 기존 레거시 시스템에서 관리하는 것으로 설계했기 때문이다.

RA 서버가 DB 를 이용하여 관리하는 데이터는 관리 UI 를 이용한다(DB 관련 내용에 대한 설명은 Appendix 부분 참고). 일반 사용자에게 제공되는 기능만을 이용하면 RA 대상 장치 등록 작업과 RA 진행 상태 조회 작업만 수행할 수 있으며, 관리자 mode 를 이용하면 추가적으로 RA 대상 장치 삭제 작업과 RA 과정에서 이용할 정답 정보 설정 작업을 수행할 수 있다. 일반 사용자를 위한 UI 이용 방법은 사용자매뉴얼을 참고하고, 여기서는 관리자 mode UI 에 대해서 설명한다.

RA 서버 설정 (관리자)

RA 서버 UI 는 관리자가 수행할 작업을 지원하기 위한 관리자 mode 를 제공한다.

주의: 해당 mode 에서 수행하는 작업은 RA 진행 전반에 영향을 미치는 것이므로 관련 정보의 배포와 공유에 주의를 기울여야 한다.

- 관리자 mode 로 전환
  - http://223.39.121.20:8000/admin/
  - 접속 정보: sktadmin / SKTadmin1!
    - 접속 정보 관련 설정은 Django 관련 초기화 작업에서 다음과 같이 수행한다. (RA 서버 UI 설치 단락 참고)

      $ python manage.py createsuperuser
- 정답지 등록과 제거
  - "PCR 정답지 등록" 메뉴에서 등록

- "펌웨어 별 PCR 정답 값 등록" 항목을 선택하여 진행
  - 항목별 정보는 GetAnswerbyTPMQuote2 프로그램 실행하여 수집
  - 해당 프로그램은 TPM 이 장착된 Raspberry Pi 에서 실행해야 함
  - Source code 의 위치는 OMP_RA_server/tools/gen_pcr_answer/
- 등록된 값들은 목록 형태로 표시되며, 함께 표시된 "삭제" 버튼으로 제거

- RA 대상에서 장치 제거
  - 관리자 mode 상태에서 "RA 장치 목록" 메뉴 진입하면 장치 마다 "삭제" 버튼이 표시되며, 이를 이용하여 RA 대상에서 제거 가능
  - RA 대상에서 제거해도 ThingPlug 에서 해당 장치가 RA 서비스와 완전히 분리되는 것은 아니며, 이를 위해서는 "ThingPlug 에 등록" 메뉴의 "Delete device from ThingPlug" 항목을 이용해야 함
    - 해당 항목에서 요구하는 정보 중, "Gateway ID"와 "Auth Key" 정보는 해당 장치가 ThingPlug 에 register 과정을 거쳐야만 존재하는 정보임.

나) PCA 관리 작업

PoC 시스템에 포함된 PCA 는 Apache 기반으로 동작하는 CGI 프로그램이다. 때문에, 기본적인 관리 작업은 Apache 웹서버에 대한 관리 작업과 동일하다. 즉, access 로그 파일 내용 분석을 통해 성공적인 서비스 제공 상황을 모니터링하며, error 로그 파일 내용 분석을 통해 오류 상황이나 PCA 프로그램에서 출력한 메시지를 확인한다. 이와 함께, 설치 과정에서 설정한 syslog 파일의 내용에서 파악한 특정 세션 관련 PID 정보를 이용하면 상황 파악에 도움이 된다.

PoC 시스템에서는 PCA 용 Apache 웹서버는 다음과 같이 (재)시작/중지한다.

```
# /skt/web/httpd-2.4.18/bin/httpd -k restart|start|stop
```

다) 관리 UI 관리 작업

관리 UI 는 NGINX 와 Django 조합을 기반으로 동작하며, 증상에 따른 대응 작업은 다음과 같다.

- 관리 UI port 로 접속에 문제
  - ✓ 네트워크 설정에 문제가 있는 경우
    - ➢ 방화벽 설정, port 관련 시스템 설정 확인
  - ✓ NGINX 서버 동작에 문제가 있는 경우
    - ➢ NGINX 재시작

      # service nginx restart
- 브라우저 화면 표시 과정에 문제
  - ✓ Django 재시작
    - ➢ 다음과 같이 uwsgi 재시작

      # initctl restart uwsgi

4. Appendix – DB 관련 설정

RA 서버가 동작 과정에서 이용하는 DB 관련 정보는 다음과 같다.

가) 테이블 목록

RA 서버는 작업 대상 장치들의 정보와 작업 진행 이력 정보, 그리고 검증을 위해 필요한 정답지 정보를 DB 에 저장한다. 이들 각각의 테이블 이름은 다음과 같다.

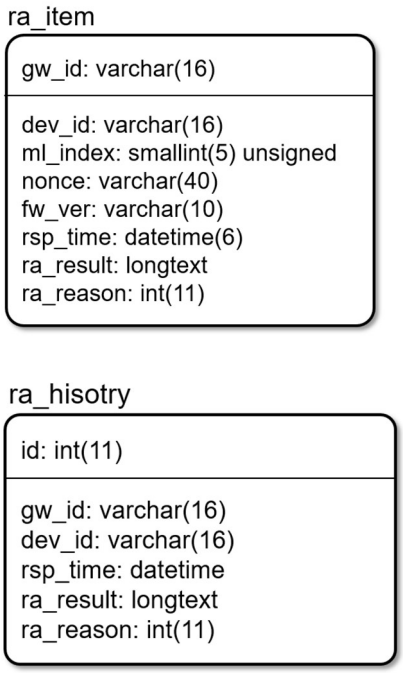
- ra_item

- ✓ 개별 장치 관련 정보들을 포함. 질의를 위한 measurement list 와 nonce 등의 정보도 함께 저장.

- ra_history
  - ✓ RA 결과를 장치 ID 정보와 함께 저장.
- ra_answer
  - ✓ RA 검증용 정보 저장. 펌웨어 버전을 index 로 하여 저장/관리.

나) ERD

PoC 시스템에서는 앞서 설명한 테이블들 사이에 특별한 관계는 없으며, 따라서 다음과 같이 독립적으로 정의되어 관리한다.



ra_answer

fw_ver :varchar(10)
pcr0: varchar(40) pcr1: varchar(40) pcr2: varchar(40) pcr3: varchar(40) pcr4: varchar(40) pcr5: varchar(40) pcr6: varchar(40) pcr7: varchar(40) pcr8: varchar(40) pcr9: varchar(40) pcr10: varchar(40) pcr11: varchar(40) pcr23: varchar(40)

다) 테이블 설계서

ra_item

Field	Type	Null	Key	Default	Extra
gw_id	varchar(16)	NO	PRI	NULL	
dev_id	varchar(16)	NO		NULL	
ml_index	smallint(5) unsigned	YES		NULL	
nonce	varchar(40)	NO		NULL	
fw_ver	varchar(10)	NO		NULL	
rsp_time	datetime(6)	YES		NULL	
ra_result	longtext	NO		NULL	
ra_reason	int(11)	YES		NULL	

ra_history

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
gw_id	varchar(16)	NO		NULL	
dev_id	varchar(16)	NO		NULL	
rsp_time	datetime(6)	YES		NULL	
ra_result	longtext	NO		NULL	

ra_reason	int(11)	YES		NULL	

ra_answer

Field	Type	Null	Key	Default	Extra
fw_ver	varchar(10)	NO	PRI	NULL	
pcr0	varchar(40)	YES		NULL	
pcr1	varchar(40)	YES		NULL	
pcr2	varchar(40)	YES		NULL	
pcr3	varchar(40)	YES		NULL	
pcr4	varchar(40)	YES		NULL	
pcr5	varchar(40)	YES		NULL	
pcr6	varchar(40)	YES		NULL	
pcr7	varchar(40)	YES		NULL	
pcr8	varchar(40)	YES		NULL	
pcr9	varchar(40)	YES		NULL	
pcr10	varchar(40)	YES		NULL	
pcr11	varchar(40)	YES		NULL	
pcr23	varchar(40)	YES		NULL	

라) 테이블 생성 명령어

ra_item

```
CREATE TABLE `ra_item` (`gw_id` varchar(16) NOT NULL PRIMARY KEY, `dev_id` varchar(16) NULL, `ml_index` smallint UNSIGNED NULL, `nonce` varchar(40) NOT NULL, `fw_ver` varchar(10) NOT NULL, `rsp_time` datetime NULL, `ra_result` longtext NOT NULL, `ra_reason` integer NULL);
```

ra_history

```
CREATE TABLE `ra_history` (`id` integer AUTO_INCREMENT NOT NULL PRIMARY KEY, `gw_id` varchar(16) NOT NULL, `dev_id` varchar(16) NULL, `rsp_time` datetime NULL, `ra_result` longtext NOT NULL, `ra_reason` integer NULL);
```

ra_answer

```
CREATE TABLE `ra_answer` (`fw_ver` varchar(10) NOT NULL PRIMARY KEY, `pcr0` varchar(40) NOT NULL, `pcr1` varchar(40) NOT NULL, `pcr2` varchar(40) NOT NULL, `pcr3` varchar(40) NOT NULL, `pcr4` varchar(40) NOT NULL, `pcr5` varchar(40) NOT NULL, `pcr6` varchar(40) NOT NULL, `pcr7` varchar(40) NOT NULL, `pcr8` varchar(40) NOT NULL, `pcr9` varchar(40) NOT NULL, `pcr10` varchar(40) NOT NULL, `pcr23` varchar(40) NOT NULL, `composite` varchar(40) NULL);
```