원격검증시스템 최종 보고서

문서이력		
2016-11-30	초기작성	강호관

1. 개요

본 문서는 ThingPlug 기반 원격검증시스템 개발 관련 사항들을 정리하며, 주요 내용은 다음과 같다.

- 원격검증 개론
- 원격검증시스템 PoC 설계
- 원격검증시스템 적용
- 원격검증시스템 개발 설명

본 문서에서는 PoC 시스템 이용과 관리에 대한 내용은 언급하지 않는다. 해당 내용들은 별도의 문서[1]에서 다룬다.

2. 원격검증 개론

원격검증(Remote Attestation, 이하 RA)은 점검 대상 장치(이하 단말)의 보안성을 별도의 주체가 원격에서 점검하고 판단하는 체계를 의미한다. 즉, 단말의 상태를 단말 자신이 판단하지 않기 때문에, 단말에 대한 조작만으로는 해당 단말에 대한 점검 주체를 완전히

[1] PoC 시스템_사용자매뉴얼.pdf, PoC 시스템_운영자매뉴얼.pdf

속일 수 없다. 따라서, 원격검증은 특히 다양한 환경에 속한 단말로 이루어진 네트워크에 대한 보안 설계에 유용하다.

원격검증이 장상적으로 이루어지기 위해서는 다음과 같은 환경이 갖춰져야 한다.

- 단말에서 점검 요청을 안전하게 처리할 수 있는 환경
- 단말의 신원 검증을 위한 환경
- 점검 관련 정보의 안전한 전달을 위한 환경

단말에서 점검 요청을 안전하게 처리할 수 있는 환경

단말에 점검 요청이 전달되면, 이에 대한 응답 정보가 작성된다. 이 과정에서 이용하는 정보들의 생성 절차가 조작 가능하지 않아야 결과로서 작성된 응답 정보를 신뢰할 수 있다. 따라서, 응답 정보 작성을 위해 신뢰할 수 있는 환경이 필요하다.

이번 과제에서는 해당 환경으로 하드웨어 보안칩인 TPM(Trusted Platform Module)을 이용한다. TPM 의 특징들 중 원격검증과 관련된 것은 다음과 같다.

- ✓ 보안 저장소 – 외부의 공격으로부터 데이터, 키, 인증서 등을 보호하며 저장
- ✓ 암호화 자체 처리 기능 – 보안 정보 노출 없이 암호화 기능 수행
- ✓ PCR(Platform Configuration Register)[2] 제공 – 단말 상태 정보 저장 공간

이러한 특징들을 기반으로, 응답 정보는 TPM 내에 보관된 각종 정보와 TPM 내에서 생성된 암호화 관련 정보들을 이용하여 안전하게 작성된다.

---

[2] PCR 은 단말의 상태 정보를 저장하기 위한 160 비트(20 바이트) 레지스터로서, TPM 버전에 따라 8 ~ 24 개 존재한다. 개별 PCR 에 저장된 정보는 주로 단말 부팅 과정에 변경되며, 특정 PCR 은 수시로 변경 가능하다.

단말의 신원 검증을 위한 환경

단말에서 응답 정보를 점검 주체에게 전달하면, 점검 주체는 해당 정보가 유효한가를 확인하는 과정에서 단말의 신원(identity)에 대한 검증도 수행한다. 즉, 정상적인 단말에서 보낸 것인지 확인하는 과정을 수행하는 것이다.

TPM 을 이용하는 원격검증의 경우, 응답 정보에는 TPM 에서 생성한 AIK(Attestation Identity Key) 정보가 함께 전달된다. 점검 주체는 이 정보를 이용하여 제 3 신뢰기관(Trusted Third Party, 이하 TTP)인 Privacy CA(이하 PCA)를 통해 유효성을 검증한다. 해당 AIK 정보가 PCA 에 저장되는 절차는 단말에서 시작하여 진행하는데, 이 과정을 AIK 서명(signing)이라 한다. 자세한 과정은 원격검증 절차에 대한 설명을 참고한다.

점검 관련 정보의 안전한 전달을 위한 환경

단말과 점검 주체 사이에 주고받는 정보는 단말의 상태를 판단하는 근거가 되므로 안전하게 전달되어야 한다. 해당 정보가 전달되는 경로는 개방되어 있는 경우가 대부분이므로, 정보를 대상으로 한 암호화와 함께 추가 데이터를 이용한 검증 작업이 필수적이다.

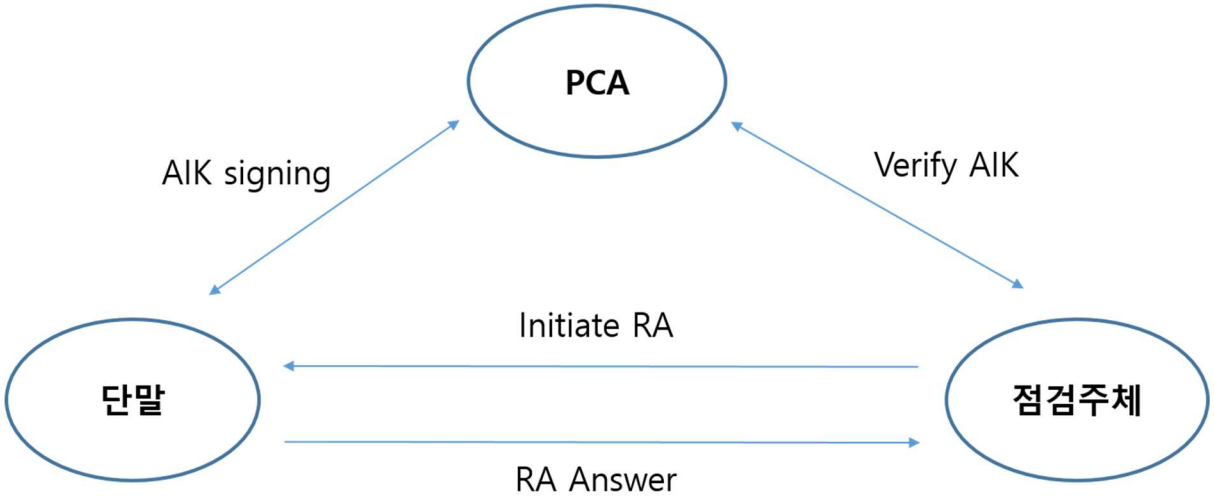
이 과정에서 단말에 대한 신원 검증은 필수적이다. 환경에 따라서는 단말과 점검 주체 사이의 완벽한 종단 암호화가 불가능할 수도 있는데, 이 경우에는 정보 전달 경로의 보안성과 함께 해당 정보에 대한 확인(verification) 기능을 이용한다.

가) 원격검증 절차

우선 앞서 설명에서 언급했던 구성 요소들을 TPM 이용 원격검증 기준으로 정리하면 다음과 같다.

- 단말 – TPM 장착한 사용자 장치
- 점검 주체 – 판단 주체
- 제 3 신뢰기관 – PCA

이들이 원격검증 과정에서 어떤 정보를 어떻게 주고 받는지 정리하면 다음과 같다. 여기서는, 점검 주체가 단말에게 검증 요청을 전달하는 방식에 대해 설명한다.



① RA 시작

- 점검주체가 특정 단말에게 RA 를 시작하도록 요청한다.
- 이때, 응답 정보 구성에 대한 요구사항이 점검주체로부터 단말로 전달된다. 전달 방법은 점검주체가 일방적으로 전달하거나, 단말의 요청에 대한 응답으로 전달한다.
- 해당 요구사항에는 세션을 식별하기 위한 nonce 와 선택된 measurement 대상 목록을 기본으로 포함한다.

② 단말은 점검주체로 보낼 RA 응답 정보에 사용할 signed AIK 를 확보한다.

- AIK 는 기본적으로 OTP(One Time Password)로 이용되지만, 상황에 따라서는 일정 유효 기간 동안 중복 사용할 수도 있다.
- 유효한 signed AIK 가 없는 경우, TPM 을 이용하여 AIK 를 생성한 후, 이를 PCA 에게 전달하여 유효한 signed AIK 를 확보한다. 이 과정에서 PCA 는

단말로부터 TPM 의 고유 정보(EKpub[3])를 함께 전달 받는다. 해당 정보는 비공개가 원칙이지만, AIK 의 유효성을 보증하기 위해 PCA 에 전달된다.[4]

- PCA 는 signed AIK 와 EKpub 의 연관 정보와 함께, 현재 시점에서 유효한 signed AIK 목록을 유지하고 관리한다.

③ 점검주체는 단말이 보낸 응답 정보에 대한 검증 작업을 진행한다.

- 응답 정보에는 단말의 signed AIK 정보가 포함되어 있으며, 이에 대한 유효성은 PCA 에 문의한다.
- 검증된 단말에서 보낸 응답 정보는 앞서 점검주체가 전달한 nonce 와 선택된 measurement 대상 목록의 정보로 구성되어 있어야 한다.
- 검증 결과는 단말에게 전달되지 않는다. 때문에 변조된 단말에서는 점검주체가 자신을 어떻게 판단하고 있는지 알 수 없다.
- 검증 결과는 다른 시스템 구성요소(들)에게 전달하여 (상황에 따른) 추가 작업이 진행될 수 있도록 할 수 있다.

3. 원격검증시스템 PoC 설계

앞서 설명한 원격검증 절차를 ThingPlug 연동 기반으로 진행할 수 있는 환경을 구현하는 것이 원격검증시스템 PoC 의 목표다. 즉, 단말은 GMMP 기반으로 OMP(Open M2M Platform)에 연동하며, 서버는 Open API 를 통해 OMP 와 연동하는 상황에서 원격검증이 진행될 수 있도록 하는 것이 목표다.

---

[3] EK(Endorsement Key)는 TPM 마다 생산 시점에 생성되어 고정되는 2048-bit RSA key 쌍이다. 비밀키(EKpr)는 절대 공개되지 않으며, 공개키(EKpub)는 PCA 에게만 공개된다.

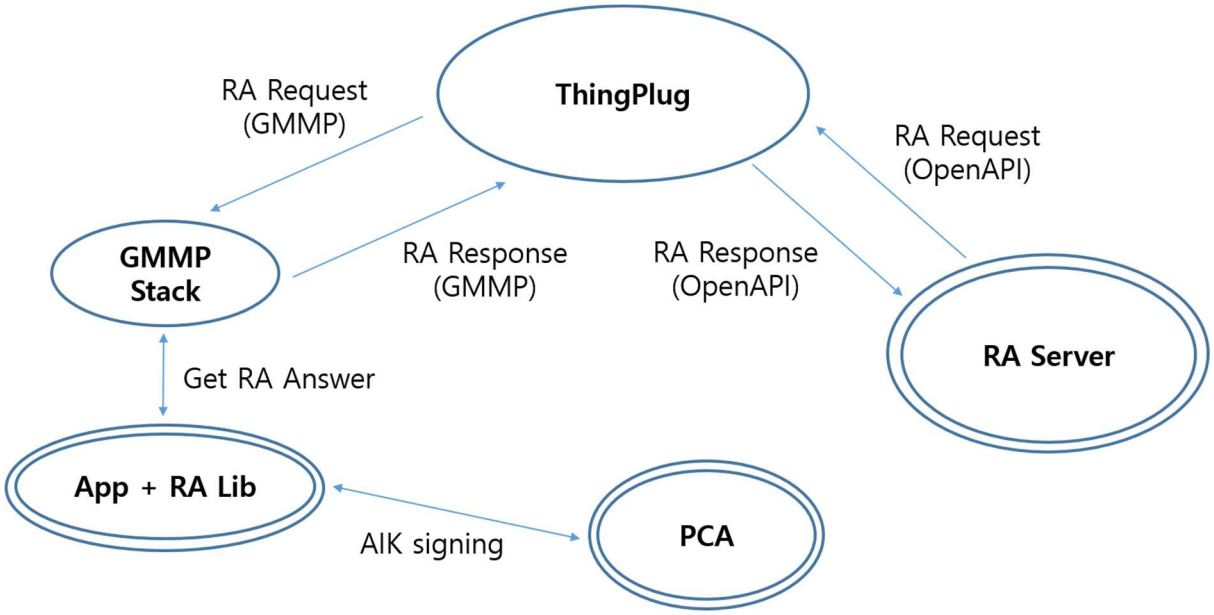
[4] PCA 와 TPM 기반 App 의 통신과 관련한 자세한 스펙은 TCPA Main Specification 1.1b 의 챕터 9.3 과 9.4 를 참조한다.

단말의 연동기반인 GMMP 는 그 특성상, 기존 목적들을 위해 정의된 메시지들을 제외한 범용 메시지의 이용은 매우 한정적이다. 즉, 제어 메시지의 일부인 user-defined control 타입 메시지 중 하나를 지정하여 새로운 용도를 부여할 수 있을 뿐이다. 때문에, 원격검증을 위한 메시지도 user-defined control 타입 메시지 중 하나를 이용한다.

그런데, GMMP 제어 메시지는 OMP 에서 단말로 전달되는 것으로, 단말이 먼저 OMP 에게 전달하거나 OMP 에게 전달을 요청할 수는 없다. 따라서, 추후에 별도의 GMMP 메시지를 정의하지 않는 한, 단말에서 다른 구성요소로 접속을 시작해야 하는 경우는 GMMP 기반 메시지를 이용할 수 없다. 때문에, 현재로서는 단말과 PCA 사이의 통신은 GMMP 를 기반으로 하지 않으며, 당연히 PCA 는 ThingPlug 와 연동하지 않는다. 이는 PCA 와 단말 사이의 AIK signing 과정이 단말에서의 요청으로 시작하기 때문이다.

가) 원격검증시스템 PoC 구조

원격검증시스템 PoC 는 다음과 같은 구조를 가진다.



구성요소별로 보면, RA Server 는 검증주체를 구현한 것이고, 단말에는 GMMP Stack 과 RA Lib 을 이용한 App 이 탑재된다. 단말과 검증주체는 ThingPlug 연동을 통해 정보를 주고 받으며, PCA 는 단말과 (TCP/IP 기반으로) 직접 정보를 주고 받는다. 검증주체와 PCA 사이의 정보 교환은 임의의 단말을 대상으로 해야 하는 PoC 특성상 생략되었다.

PoC 에서의 원격검증 진행 과정은 다음과 같다.

① RA 시작

- 점검주체인 RA Server 는 다음의 과정을 주기적으로 진행한다.
- RA Server 는 RA 대상 단말들에 대한 정보를 조회한다. 해당 정보에는 GMMP 상의 고유 식별자와 measurement 대상 목록이 포함된다.
- RA 대상 단말에 대한 RA 시작 요청을 Open API 를 이용하여 OMP 에 전달한다. 이때 사용되는 Open API 는 "GW User Defined Control"이며, 함께 전달하는 정보는 measurement 대상 목록과 세션 식별을 위한 nonce 가 기본으로 포함된다.
- OMP 에 전달된 RA 시작 요청은 RA Server 가 (URL 로 지정한 user-defined control 타입의) GMMP 패킷으로 변환되어 단말에 전달된다.

② 단말은 RA 응답 정보를 작성하여 OMP 로 전달

- 단말에서 동작 중인 GMMP stack 에 의해 RA 시작 요청 패킷이 App 으로 전달되면, 해당 패킷은 RA Lib 에서 제공하는 API 를 이용하여 처리된다.
- 유효한 signed AIK 가 존재하지 않으면 PCA 와 AIK signing 절차를 진행한다.
- RA 응답 정보는 RA 시작 요청에 포함된 nonce 정보를 포함하고 있어야 하며, RA 시작 요청에 포함된 measurement 목록에서 지정한 정보들로 작성되어야 한다.
- 작성된 RA 응답 정보는 GMMP 패킷으로 OMP 에 전달된다.

③ RA 응답 정보 처리

- RA 응답 정보는 OMP 내부의 단말 관련 정보 저장소에 수집된다. 이를 RA Server 가 주기적으로 조회하여 수집한다.
  - ✓ RA Server 가 고정된 공인 IP 주소(혹은 OMP 에서 접속 가능한 고정 IP 주소)를 가지는 경우에는 ThingPlug 에서 RA 응답 정보를 PUSH 하도록

설정할 수도 있다. 하지만, 이번 PoC 의 RA Server 에서는 지원하지 않는다.

- RA 응답 정보에는 단말의 signed AIK 정보가 포함되어 있으며, 이를 이용하여 RA 응답 정보를 검증(verification)함으로써 부인방지 기능을 제공한다.
- RA 응답 정보에 포함된 nonce 와 최근 RA 시작 요청에 포함된 nonce 의 일치를 확인하여 최신 결과만을 처리한다.
- 검증 결과는 현재 상태를 관리하는 DB 와 검증 이력을 관리하는 DB 에 저장한다.

나) ThingPlug 연동 이슈

ThingPlug 연동을 위한 원격검증 구성요소별 고려 사항들은 다음과 같다.

- 단말 App(+ RA Lib)
  - ➢ ThingPlug 연동 단말은 반드시 단 하나의 서비스에 등록되어야 한다.
    - ✓ 서비스 변경을 위해서는 기존 서비스에서 단말 정보를 삭제한 후, 신규 서비스에서 단말 등록을 해야 한다.
    - ✓ GMMP 패킷 수신을 위한 네트워크 접속 정보는 서비스별 고유 정보.
    - ✓ GMMP 상의 등록(registration) 절차 진행 이전에 해당 단말의 정보는 ThingPlug 에 등록되어 있어야 한다.
    - ✓ 해당 제약 사항으로 인하여 ThingPlug 연동 단말에 대한 원격검증은 서비스별로 적용되는 것이 효과적이다.
  - ➢ GMMP stack 에는 RA 시작 요청을 위한 제어 메시지를 단말 App 에게 전달하는 부분이 추가되어야 한다.
- RA Server

- 단말에 대한 RA 시작 요청은 user-defined 제어 메시지로 OMP 를 경유하여 단말에 전달된다.
  - ✓ 요청과 응답은 비동기적으로 진행된다.
- 제어 메시지의 특성상 1 번의 요청과 응답으로 원격검증 과정이 완결된다.
  - ✓ 복수의 제어 메시지를 이용하여 단계별로 진행할 수도 있겠지만, 비동기적으로 진행되는 요청과 응답으로 인한 컨텍스트(context) 유지 부하는 PoC 범위를 벗어난다. (고도화 작업에서는 보안성 향상과 함께 고려할 수 있다.)
- RA Server 는 OMP 에서 직접 접속할 수 있는 고정 IP 주소를 가지지 않는 경우가 더 많다.
  - ✓ OMP 에서 RA Server 로 직접 접속할 수 있는 상황이라면, OMP 가 PUSH 방식으로 응답 정보를 직접 RA Server 에게 전달할 수도 있다.
  - ✓ RA Server 는 서비스별로 운영되고, 위치도 해당 서비스 운영자의 보안 영역일 수 있기 때문에, 다른 구성요소에서 바로 접속할 수 없는 경우가 더 많을 것이다.
  - ✓ 따라서, RA Server 가 OMP 에 저장된 원격검증 응답 정보를 조회하여 처리하는 방식을 채택한다.

- PCA
  - AIK signing 은 단말이 요청 정보를 전달함으로써 시작되고, 단말이 signing 결과 정보를 수신함으로써 종료된다.
    - ✓ GMMP 에서 단말이 OMP 를 경유하여 자신의 정보를 전달하는 것은 각종 보고 메시지로 한정된다.
    - ✓ 단말이 요청 정보를 전송하고, 해당 요청에 대한 응답 정보를 수신하는데 이용할 수 있는 메시지는 없다.
    - ✓ 따라서, PCA 는 ThingPlug 연동 대상에서 제외한다.

- PCA 의 ThingPlug 연동을 위해서는, GMMP 스펙에 "단말이 요청 정보를 전송하고 그에 대한 응답 정보를 수신할 수 있는" 방법이 제시되어야 한다.

4. 원격검증시스템 적용

지금까지 설명한 원격검증시스템을 실제 운용할 때, 시스템 구현을 변경하지 않고 세부적인 동작 방식을 조정하기 위하여 다음과 같이 다양한 설정 정보가 제공된다.

가) AIK 인증 주기

단말에서 전달한 AIK 에 대해 PCA 에서 작성하는 인증서의 유효 기간을 의미한다. AIK 가 검증(Attestation) 과정에서 단말의 identity 를 나타내고 TPM 에 의해 계속해서 생성될 수 있는 정보이기 때문에 해당 key 의 유효 기간은 짧을수록 바람직하다. 하지만, 인증주기가 단말과 PCA 의 성능에 비해 지나치게 짧으면 정상적인 검증 작업을 방해할 수 있다. 따라서, 단말 개수와 PCA 의 성능 등을 고려하여 인증 주기를 조정한다.

PoC 에서는 PCA 가 Apache 에 탑재되어 동작하는 CGI module 이며, 이에 대한 virtual host 설정에서 (**VALID_DAYS**) 환경 변수로 인증 주기를 조정할 수 있다. 해당 환경 변수가 없는 경우, 기본값은 365 일로 설정된다. 이는 AIK 가 2048-bit RSA 키로서 가지는 안전성을 고려한 설정이다.

나) PCA 서버 인증서 유효 기간

PoC 에서는 PCA 서버 인증서로 사설 인증서를 만들어 이용한다. 사설 인증서 생성을 위한 절차는 일반적인 사설 인증서 생성 과정을 따르는데, 이때 해당 인증서의 유효 기간은 AIK 인증 주기를 고려하여 설정해야 한다.

다) 원격검증 수행 주기

PoC 는 검증주체인 RA Server 가 검증 절차를 주도하는 방식으로 설계되었기 때문에, RA Server 가 RA 시작 요청을 OMP 에 전달하는 시점을 결정한다. 이때 검증 대상 단말 정보는 일정한 주기마다 DB 에서 조회하는데, 해당 주기를 조절함으로써 원격검증의 성격을 조정할 수 있다. (tp_server.conf 파일의 timer_interval 항목. 초단위로 설정)

단말이 여러 작업을 수행하기에 충분한 성능과 역할을 가지고 있다면, 검증 주기는 짧아져야 한다. 또한, 단말이 작업을 위해 이용하는 프로토콜에 취약점이 많고 이에 대한 빠른 조치가 불가능한 환경에 있다면, 또한 검증 주기는 짧아져야 한다. 하지만, 단말 개수가 아주 많고, 성능과 수행하는 작업이 단순하다면, 짧은 검증 주기는 문제가 될 수 있다.

라) User-defined control 메시지 선택

GMMP 의 user-defined control 메시지는 32 가지 정의할 수 있는데, 현재 PoC 에서는 5 번째에 해당하는 값을 RA 시작 요청 메시지로 이용하고 있다. 만약, 원격검증을 적용할 서비스에서 해당 user-defined control 메시지를 이용하고 있는 상황이라면, 다른 값으로 변경하면 된다. (tp_server.conf 파일의 cfg_uds 항목)

그런데, 단말의 경우는 GMMP stack 과 RA Lib 관련 API 의 매개변수를 변경해야 한다. 하지만, 기존 구현 중 일부만 간단히 수정하면 된다.

마) 로그 관리

RA Server 동작 과정의 로그는 syslog 로 기록되며, 운영 체제의 syslog 관련 설정에 따라 기존 legacy 시스템과 연동도 가능하다. PoC 에서는 특정 파일에 해당 로그를 저장하도록 설정되어 있다.

PCA 는 Apache 에 탑재된 CGI module 이므로 stderr 로 출력된 내용은 Apache 의 error 로그에 저장되고, 동작 과정의 주요 로그는 syslog 로 기록된다. RA Server 와 마찬가지로 PoC 에서는 특정 파일에 해당 로그를 저장하도록 설정되어 있다.

단말에서 이용하는 RA Lib 에서 발생하는 로그는 기본적으로 파일에 저장되지만, 로그 초기화 관련 API(init_log)에게 전달되는 매개변수에 따라서는 syslog 로 로그를 저장할 수 있다. PoC 에서는 단말에서의 syslog 관련 설정을 배포하지는 않는다. 필요한 경우 Critical 레벨 로그에 대한 설정으로 RA Lib 관련 로그 저장을 조정할 수 있다.

바) Measurement 항목 설정

점검 대상 단말에게 어떤 정보를 요구할 것인가를 결정하는 것으로, PoC 에서는 PCR 4 번, 5 번, 6 번, 23 번 정보에 대한 조합 방식을 지정하는 것으로 구현되어 있다. 이는

PoC 시스템 이용을 위해 배포된 Raspberry Pi 용 boot loader 와 kernel 등에 대해 미리 수집 가능한 정보들이기 때문이다.

만약, 원격검증 적용 대상 단말이 Raspberry Pi 가 아닌 다른 하드웨어를 기반으로 하거나, boot loader 와 kernel 등이 변경된다면 measurement 항목은 그에 맞춰 변경되어야 한다.

PoC 형상에서 원격검증 대상으로 특정 데이터를 지정하고자 하는 변경의 경우에는 다음의 작업을 통해 measurement 관련 정보를 변경할 수 있다.

(ㄱ) PoC 시스템 이용을 위해 배포된 이미지를 설치한 단말(Raspberry Pi)를 준비

(ㄴ) 원격검증 대상 데이터를 단말에 정상적으로 설치

(ㄷ) RA Lib 관련 소스 코드인 ralib/ralib.c 에 포함된 init_RA 구현 수정

- pcr_reset_extend 함수의 2 번째 매개변수를 원격검증 대상 데이터의 경로로 수정 (주의: 첫 매개변수인 23 은 수정하지 않는다)

(ㄹ) 수정된 소스 코드를 이용하여 RA Lib 과 관련 App 을 다시 빌드한 후, 단말의 전원을 제거했다가 다시 연결한다.

(ㅁ) 단말에서 관련 App 이 실행되어 23 번 PCR 정보가 설정된 것을 확인한다.

- cat /sys/class/tpm/tpm0/device/pcrs

(ㅂ) 배포된 소스 코드의 OMP_RA_server/tools/gen_pcr_answer 디렉터리 내용을 단말에서 빌드하여 GetAnswerbyTPMQuote2 프로그램을 생성한 후 실행한다.

- 출력된 정보를 RA Server 가 참조하는 Answer 정보에 신규 firmware 관련 정보로 등록한다.

위의 과정을 진행할 때 주의할 사항은, 원격검증 대상 데이터는 복사를 통해 배포되어야 한다는 것이다. 동일 소스 코드를 이용한다 하더라도 빌드 환경이 변경되면 생성된 데이터는 서로 다르기 때문이다. 원격검증은 데이터의 변경 여부를 점검할 뿐, 해당 데이터의 실행 양태의 변경 여부를 점검하는 것이 아니기 때문이다.

사) TPM 변경

PoC 환경 구성에 사용된 TPM 은 인피니언(www.infineon.com) 제품이다. 만약, 다른 TPM 제품을 이용하고자 하는 경우에는, 그를 위한 kernel 드라이버를 작성하고 이를 kernel 형상에 반영한 후, 앞서 설명한 measurement 항목 설정과 유사한 절차를 진행하면 된다. 이번에는 RA Lib 이나 관련 App 의 소스 코드 변경은 필요 없으며, 변경된 커널을 설치한 후 전원 재연결을 통해 단말을 재시작한 상태에서 GetAnswerbyTPMQuote2 프로그램을 실행한 결과를 이용하면 된다.

5. 원격검증시스템 개발 설명

가) 소스 코드 설명

원격검증시스템 PoC 를 위한 소스 코드는 다음과 같은 2 개의 주요 디렉터리를 기준으로 구성된다.

- GMMP_RA_client – 단말 환경을 위한 소스 코드
  - GMMP_lib
    - ✓ GMMP 이용을 위한 편이 함수 제공
    - ✓ thread 를 시작하는 함수가 없다. 즉, GMMP stack 의 동작 방식은 해당 라이브러리를 이용하는 App 의 구현에 따라 결정된다.
  - ralib
    - ✓ RA Lib 함수들의 구현. GMMP 기반으로 작성.
    - ✓ App 에서는 init_RA 함수와 GMMP_Do_RA 함수만 호출
  - src
    - ✓ ThingPlug 사이트에세 제공되는 GMMP sample 구현을 기반으로 원격검증 관련 sample 구현을 추가

- ✓ Sample_All.c 는 원격검증 관련 sample 구현. dummy_gw.c 는 설정 파일을 이용할 수 있도록 수정된 GMMP sample 구현. show_gw.c 는 단말 등록 후에야 파악 가능한 gateway ID (in ThingPlug) 정보를 알아내기 위해 간략하게 정리된 GMMP sample 구현.

- OMP_RA_server
  - ➢ privacy_ca
    - ✓ PCA 관련 소스 코드와 (인증서와 key 파일을 포함한) 설정 파일들
    - ✓ Apache 에 탑재되는 CGI module 로서 독립적으로 구현되어 다른 디렉터리의 소스 코드들과는 의존 관계가 없음
  - ➢ attester_verifier
    - ✓ RA Server 구현 중, ThingPlug 관련 부분을 제외한 부분의 소스 코드
    - ✓ ThingPlug 관련 부분은 tp_ralib 라이브러리(libtp_ra.a)를 이용
  - ➢ tp_ralib
    - ✓ ThingPlug 연동을 위한 구현. 요청과 응답이 비동기적으로 처리되기 때문에 내부에 개별 요청을 위한 데이터를 관리.
    - ✓ thread 를 시작하는 함수가 없다. 해당 라이브러리를 이용하는 RA Server 구현에서 thread 를 운용하면서 필요한 API 를 호출
    - ✓ RA Server 에서는 초기화 API 인 init_ralib 과 요청 전달 API 인 issue_ra, 그리고 응답 수집 API 인 get_response 를 호출한다.
  - ➢ tools
    - ✓ gen_pcr_answer - measurement 결과 정보 생성 도구
    - ✓ ra_tool - PoC 용 관리 UI 에서 ThingPlug 와 연동하여 장치 정보를 관리하는 도구들
  - ➢ web_ui

- ✓ PoC 용 관리 UI 구현
- ✓ Python script 로 주요 부분 구현

나) 빌드 방법

빌드 방법은 2 개의 주요 디렉터리 모두 다음과 같다.

```
$ cd GMMP_RA_client (혹은 OMP_RA_server)

$ ./bootstrap

$ ./configure

$ make
```

configure 실행 과정에서 패키지가 없어서 실패하면, 해당 패키지를 추가 설치한 후 다시 진행한다.

위의 과정은 단말 환경과 서버 환경에서 모두 가능하다. 특히, GMMP_RA_client/src 디렉터리의 dummy_client 와 show_gw 는 서버 환경에서 빌드한 경우, 디버깅과 단말 관리를 위한 정보 수집 도구로 유용하다.

한편, OMP_RA_server/tools/gen_pcr_answer 디렉터리의 소스 코드들은 단말 환경에서 빌드해야 measurement 정보 관리를 위한 도구인 GetAnswerbyTPMQuote2 프로그램을 만들 수 있다. 해당 디렉터리의 위치를 GMMP_RA_client 하위 디렉터리로 하지 않은 것은, 프로그램이 생성하는 데이터를 RA Server 에서 이용해야 하기 때문이다. 즉, 이미지 배포를 위한 사전 작업에 필요한 프로그램의 소스 코드라서 OMP_RA_server 하위 디렉터리에 위치를 정했다.

빌드 결과로 생성되는 실행 파일들에 대한 추가 설명은 별도의 문서[5]를 참조한다.

---

[5] 단말은 PoC_시스템_사용자매뉴얼.pdf, 검증주체는 PoC_시스템_운영자매뉴얼.pdf

다) API 설명

원격검증시스템 PoC 와의 연동을 위한 API 는 단말 환경의 RA Lib 과 관련된 API 들과 RA Server 구현에 이용된 API 들로 구성된다.

단말 환경의 RA Lib API

단말 환경에서 동작하는 App 은 원격검증 관련 작업에 영향을 주는 작업을 직접 하지는 않으며, 단지 GMMP 스택을 초기화여 기동하는 과정에서 원격검증 관련 초기화 작업을 함께 진행하면 된다. 이때 다음의 API 를 호출한다.

int init_RA (char *pca_addr, char *x509_path, int udc_no)	
원격검증 관련 필수 정보를 설정	
매개변수	
pca_addr	PCA IP 주소 (port 포함)
x509_path	signed AIK 인증서를 저장할 위치
udc_no	user-defined control 메시지 번호 (1~32, 0 은 all)
결과값	
성공하면 0, 실패하면 -1	

App 이 기동한 GMMP stack 은 RA 시작 요청 메시지 처리를 위해 아래의 API 를 호출하면 된다. 정확한 사용 예는 Sample_All.c 파일을 참고한다.

int GMMP_Do_RA (GMMPHeader *pstGMMPHeader, stControlReqHdr *pstReqHdr)	
RA 시작 요청 GMMP 메시지 처리	
매개변수	
pstGMMPHeader	PCA IP 주소 (port 포함)
pstReqHdr	signed AIK 인증서를 저장할 위치
결과값	
RA_OK	전달된 RA 요청 메시지 처리 (응답 정보 생성 실패도 포함)
RA_BYPASS	설정된 user-defined control 메시지가 아님
RA_BAD_CONFIG	RA Lib 이 초기화되지 않았음
RA_BAD_PACKET	전달된 메시지의 형식에 문제가 있음

결과값이 RA_OK 일 때만 GMMP notify 메시지가 전송되기 때문에, 다른 값일 때는 GMMP stack 에서 GMMP notify 메시지를 전송해줘야 한다.

RA Server 구현에 이용된 API

RA Server 는 원격검증시스템 PoC 의 검증 주체로 구현되었으므로, 원격검증을 적용할 서비스의 운용환경과 검증 시나리오의 변경에 따라 수정될 수 있다. 여기서 설명된 API 들은 RA Server 의 동작을 위해 구현되었다.

int verify_response (MYSQL *mysql, SVR_CONF *s_conf, BYTE *resp, char *gw_id, time_t mtime)	
단말에서 보낸 응답 정보에 대한 유효성 검사	
매개변수	
mysql	DB 접속 객체
s_conf	RA Server 관련 설정 (DB 설정 참고용)
resp	단말의 응답 정보
gw_id	단말의 gateway ID (ThingPlug 내에서의 고유 정보)
mtime	단말의 응답 시간
결과값	
성공하면 0, 실패하면 -1	

int verify_pcr_composite (unsigned char *pcrComposite, int pcrCompositeSize, unsigned char *keyPub, int keyPubSize, unsigned char *signedPCR, int signedPCRSize)	
단말의 응답 정보를 AIK 공개키(AIKpub)로 검증	
매개변수	
pcrComposite	단말의 응답 정보
pcrCompositeSize	응답 정보의 길이
keyPub	단말의 AIKpub
keyPubSize	AIKpub 의 크기
signedPCR	단말의 signed PCR 정보
signedPCRSize	signed PCR 정보의 길이
결과값	
성공하면 SUCCESS, 실패하면 FAILED	

int verify_pcr_digest (unsigned char *pcrComposite, unsigned char *pcrAnswer)	
단말의 응답 정보에 대한 digest 검증	
매개변수	
pcrComposite	단말의 응답 정보
pcrAnswer	예상 응답 정보 (정답 정보)
결과값	
성공하면 SUCCESS, 실패하면 FAILED	

int verify_pcr_nonce (unsigned char *pcrComposite, BYTE *nonce)	
단말의 응답 정보에 포함된 nonce 정보 검증	
매개변수	
pcrComposite	단말의 응답 정보
nonce	예상 nonce (정답 정보)
결과값	
성공하면 SUCCESS, 실패하면 FAILED	

int init_log (uint32_t log_size, const char *log_path, const char *log_name, uint8_t tofile, uint8_t tosys)	
로그 관련 정보 초기화	
매개변수	
log_size	로그 파일 크기 제한
log_path	로그 파일 디렉터리 경로
log_name	로그 파일 이름
tofile	로그 메시지를 로그 파일에 저장할 것인가 여부
tosys	로그 메시지를 syslog 로 저장할 것인가 여부
결과값	
항상 1	

void DebugPrintFunc (int log_lv, const char *file, int line, const char *errmsg, ...)	
주어진 로그 메시지를 설정에 따라 파일과 syslog 로 저장	
매개변수	
log_lv	로그 메시지를 어디에 저장할 것인가
file	해당 API 를 호출한 파일의 이름
line	해당 API 를 호출한 파일 내의 위치
errmsg	로그 메시지

int parse_server_conf (const char *conffile, SVR_CONF *conf_data, const char *separator)	
RA Server 설정 파일 내용을 로딩	
매개변수	
conffile	설정 파일 경로 (파일 이름 포함)
conf_data	설정 정보 저장 자료 구조체
separator	설정에서 key 와 value 쌍을 분리하기 위한 문자
결과값	
성공 1, 실패 0	

int DB_tp_get_nonce(MYSQL *mysql, SVR_CONF *s_conf, char *gw_id, unsigned char *nonce)	
지정한 단말의 nonce 정보를 조회	
매개변수	
mysql	DB 접속 객체
s_conf	RA Server 관련 설정 (DB 설정 참고용)
gw_id	단말의 gateway ID (ThingPlug 내에서의 고유 정보)
nonce	nonce 정보 저장 위치
결과값	
성공 EXIT_SUCCESS, 실패 EXIT_FAILURE	

int DB_tp_get_ra_list(MYSQL *mysql, SVR_CONF *s_conf, ra_request **dev_list, int *dev_num)	
원격검증 대상 단말 정보 조회	
매개변수	
mysql	DB 접속 객체
s_conf	RA Server 관련 설정 (DB 설정 참고용)
dev_list	단말 정보 저장 위치
dev_num	조회된 단말 개수
결과값	
성공 EXIT_SUCCESS, 실패 EXIT_FAILURE	

int DB_tp_update_device_info(MYSQL *mysql, SVR_CONF *s_conf, char *gw_id, int reason, uint8_t result, unsigned long tf)	
원격검증 결과 저장	
매개변수	
mysql	DB 접속 객체
s_conf	RA Server 관련 설정 (DB 설정 참고용)
gw_id	단말의 gateway ID (ThingPlug 내에서의 고유 정보)
reason	원격검증 실패 이유
result	원격검증 결과
tf	원격검증 완료 시각
결과값	
성공 EXIT_SUCCESS, 실패 EXIT_FAILURE	

int DB_tp_update_nonce(MYSQL *mysql, SVR_CONF *s_conf, char *gw_id, unsigned char *nonce)	
지정한 단말의 nonce 값을 변경(설정)	
매개변수	
mysql	DB 접속 객체
s_conf	RA Server 관련 설정 (DB 설정 참고용)
gw_id	단말의 gateway ID (ThingPlug 내에서의 고유 정보)
nonce	nonce 정보 저장 위치
결과값	

성공 EXIT_SUCCESS, 실패 EXIT_FAILURE

int DB_tp_get_pcr_answer(MYSQL *mysql, SVR_CONF *s_conf, unsigned char *answer, char *gw_id)	
지정한 단말의 예상 응답 정보(정답 정보) 조회	
매개변수	
mysql	DB 접속 객체
s_conf	RA Server 관련 설정 (DB 설정 참고용)
answer	정답 정보 저장 위치
gw_id	단말의 gateway ID (ThingPlug 내에서의 고유 정보)
결과값	
성공 EXIT_SUCCESS, 실패 EXIT_FAILURE	

원격검증시스템 PoC 는 ThingPlug 연동 기반이므로, RA Server 는 직접 단말과 데이터를 주고 받지 않고 OMP 를 경유한다. 때문에, RA Server 구현에서는 단말과 데이터를 주고 받는 부분을 분리하여 별도의 API 로 정의했다.

int issue_ra (int cnt, ra_request *req)	
단말들에 대한 RA 시작 요청을 등록	
매개변수	
cnt	RA 시작 요청 개수
req	RA 시작 요청 정보 리스트
결과값	
(최초 오류 발생 직전까지) 등록에 성공한 요청 개수, 실패하면 -1	

int get_response (int budget, ra_response **resp)	
RA 시작 요청에 대한 결과 정보 (단말의 응답 정보, OMP 의 오류 정보) 조회	
매개변수	
budget	(이번 요청에 수신할) 결과 정보의 최대 개수
resp	결과 정보 리스트에 대한 포인터를 저장할 위치
결과값	
결과 정보 개수, 실패하면 -1	

int init_ralib (char *mIa_addr, char *svc_id, int udc_no, char *auth_id, char *auth_key)	
ThingPlug 연동을 위한 초기 정보 설정	
매개변수	
mIa_addr	App 과의 연동을 위해 OMP 에서 제공하는 접속 정보
svc_id	App 의 ServiceID (in ThingPlug)
udc_no	User-defined control 메시지 번호 (mgmtX in ThingPlug)
auth_id	App 의 AuthID (in ThingPlug)
auth_key	App 의 AuthKey (in ThingPlug)
결과값	
성공하면 0, 실패하면 -1	

int init_ralib2 (char *cfname)	
ThingPlug 연동을 위한 초기 정보 설정	
매개변수	
cfname	설정 파일의 경로 (파일 이름 포함)
결과값	
성공하면 0, 실패하면 -1	

6. 부록. 용어정리

- 단말
  - 원격검증 구성 요소들 중 점검 대상을 의미. 점검 주체 보다는 상대적으로 낮은 성능을 가진 특화된 기능을 수행하는 장치. 원격검증시스템 PoC 에서는 TPM 칩과 함께 원격검증 App 을 탑재한 Raspberry Pi 를 의미.
- 점검 주체
  - 원격검증 구성 요소에서 단말에서 보낸 정보로 단말의 상태를 검증하는 주체. 다수의 단말을 대상으로 검증 작업을 수행해야 하므로 그에 맞는 성능을 보유한 서버에서 동작하는 App.
- TPM (Trusted Platform Module)
  - 안전한 암호화 프로세서를 위한 국제 표준. Trusted Computing Group (TCG) 컨소시엄에서 기술 명세를 작성해오고 있다. 해당 표준을 구현한 칩을 TPM 칩이라고 한다.
- PCR (Platform Configuration Register)
  - 단말의 상태 정보를 저장하기 위한 160 비트(20 바이트)의 TPM 칩 내부 레지스터로서, TPM 버전에 따라 8 ~ 24 개 존재한다. 개별 PCR 에 저장된 정보는 주로 단말 부팅 과정에 변경되며, 특정 PCR 은 수시로 변경 가능하다.
- AIK (Attestation Identity Key)
  - 검증(attestation) 작업을 위해 단말이 TPM 칩에서 생성한 RSA 키 쌍. 검증주체에게 보낼 응답 정보에 대해 sign 작업을 수행하여 해당 정보의 안전한 전달을 확인할 수 있게 한다.
- 제 3 신뢰기관 (Trusted Third Party, TTP)

- 사용자 인증(user authentication), 부인 방지(non-repudiation), 키 관리(key management) 등에서 당사자들로부터 신뢰를 얻고 중재, 인증, 증명, 관리 등을 하는 기관.

- measurement 대상 목록
  - 검증주체가 단말에게 어떤 정보로 응답 정보를 생성할 것인가 알려주는 정보. 주로 어떤 PCR 의 정보를 이용할 것인가에 대한 정보.

- EK (Endorsement Key)
  - TPM 칩 마다 생산 시점에 고정되어 저장된 암호화 키 정보(2048-bit RSA key 쌍). 비밀키(EKpr) 부분은 절대 공개되지 않으며, 공개키(EKpub) 부분은 TPM 에 대한 신원(identity) 검증을 위해 이용된다. PCA 에 AIK signing 작업을 요청할 때 공개키 부분을 전송한다.

- User-defined control 타입 메시지
  - GMMP 제어 메시지 중, control 타입 코드가 0x80 ~ 0x9f 영역에 해당하는 메시지. GMMP 제어 메시지 대부분이 특정 목적을 위한 것으로 정의되어 있기 때문에, 신규 목적을 위해서는 해당 메시지들을 이용해야 한다.

- App
  - 본 문서 내의 App 은 단말 App 과 검증주체 App 을 의미한다.
  - 단말 App 은 단말에서 GMMP 기반으로 동작하는 응용 프로그램으로, 원격검증과 상관없이 원래의 서비스를 위해 구현된 형상을 기본으로 한다. 단지, 원격검증을 위한 라이브러리의 정상적인 동작을 위해, 해당 라이브러리의 초기화 함수와 RA 요청 메시지 처리 함수를 호출하는 수정 작업만 필요하다.
  - 검증주체 App 은 검증주체 역할을 수행하는 응용 프로그램으로 충분한 성능을 가진 서버 환경에서 운용된다.

목차