Task1 :

Dummy.sh - > hello

Init.rc - > kernel을 초기화하는 함수

X86 - > pc vision-> init.sh로 사용

Boot 과정 1

///Init.sh-> app\_process///->zygote(세포)VM android OS -> task (proess)실행 시 zygote exec해서 사용

Zygote의 목적: cow(copy on write)-> shared data -왜냐면 같은 환경에서는 동일한 파일을복사해야하는데 이것은 시간적으로나 용량적으로나 많은 over head를 발생시킴

리눅스에서 shared data -> fork(일부 데이터를 공유) - > exec(이그젝하면서 추가적인 다른 부분을 로드)

Root 권한 app\_process를 실행하고 나서 zygote를 부여할 때 root권한을 해제함

System/\* root권한이 필요

Task1 : init.rc (커널을 초기화)과정에서 system을 munt해서 집어넣음

Task2: 정상 루트 **Init.sh-> app\_process**

**Init.sh -> my\_app\_process->app\_process**(시나리오)

Init.sh의 성격: app\_process라는 이름을 실행한다.

my\_app\_process->app\_process으로 이름을 변경

my\_app\_process->1. dummy2를 씀 ->2. app\_process를 실행함

**Init.sh -> app\_process(**my\_app\_process **)->original\_app\_process**(app\_process)

Task 3: android boot 과정에서 shell을 사용할 수 없다

Boot할떄 shell을 지원을 안함 - > 우분투를 사용함으로써 shell을 사용할 수 있음

그러므로 root shell을 띄운다고해도 이것이 exit 명령을 실행하지 않으면 다음단계로 진행할 수 없음 daemon을 만들어서 이것을 사용함 -> 언제든 호출에서 shell을 띄워 줌

문제점 : daemon **fd: 0 1 2** 이 있다 child daemon fd를 그대로 물려받음

결국에는 쓸떄는 daemon에서 써야하고 출력도 daemon이 함

Root 권한을 가진 shell을 우회해서 가질 수 있음

Code

fd0=Fd4

업데이트 바이너리: 리커버리 OS의 의해서 실행되어지는 바이너리, 이것은 업데이터 스크립트 를 로드하고 실행한다.

업데이터 스크립트: 설치 스크립트, 업데이트 바이너리에 의해서 인터프레트되어진다. 에디파이라는 스크립트 언어를 사용함

시그니쳐 검증 이후에 파일을 tmp파일로 아래와 같은 형식을 보냄

update-binary 3(버전) stdout(pipe: recorvery와 소통에 사용하기 위한 update-binary) /sdcard/ota.zip