# Analysis of Docker Security

## Thanh Bui et al., Aalto University School of Science, 2015. 01

전상기(sanggi@pel.smuc.ac.kr)

상명대학교 프로토콜공학연구실

# 목 차

# Introduction

- 가상화 기술의 사용이 급격히 증가함에 따른 효율적이고 안전한 가상화 솔루션을 요구함

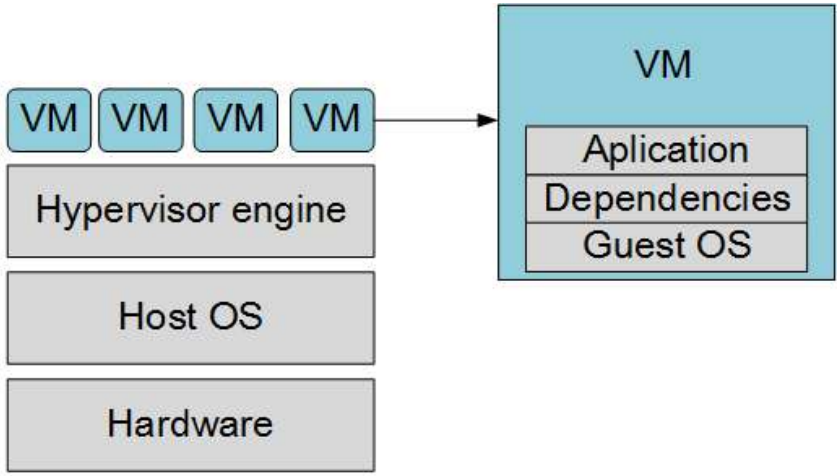
- 가상화 기술의 이점
  - Server virtualization in data centers
    - 관리자는 단일 서버에서 하나 이상의 가상 시스템 인스턴스를 만들 수 있음
  - Desktop virtualization
    - 한대의 컴퓨터가 여러 OS 인스턴스를 실행 할 수 있음

# Introduction

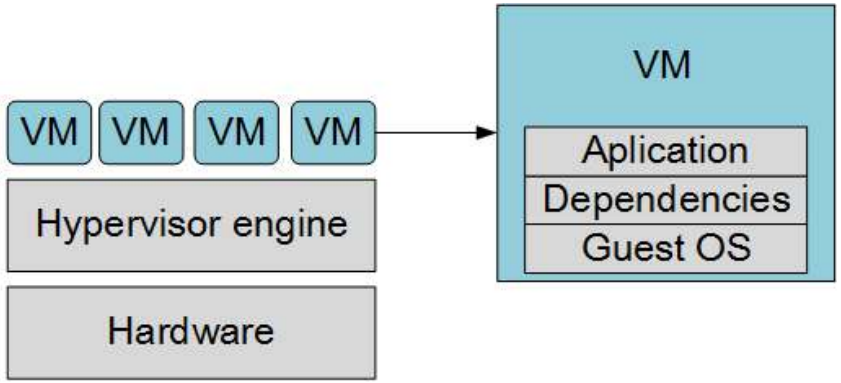
- 가상화 기술 유형
  - Hypervisor-based virtualization
  - Container-based virtualization
- Docker
  - Container-based virtualization
  - Docker Architecture
    - Docker engine
    - Docker container
  - Docker Security Analysis
    - Docker Internal Security
    - Docker and Kernel Security Systems

# Background

- Virtualization 유형
  - Hypervisor-based virtualization
    - 하드웨어 수준에서 가상화를 제공함
    - 호스트 운영 체제 상단에 완벽한 가상 시스템을 구축함
    - 각 가상화 시스템은 응용 프로그램과 그 종속성 뿐만 아니라 별도의 커널과 함께 전체 게스트 OS로 구성됨



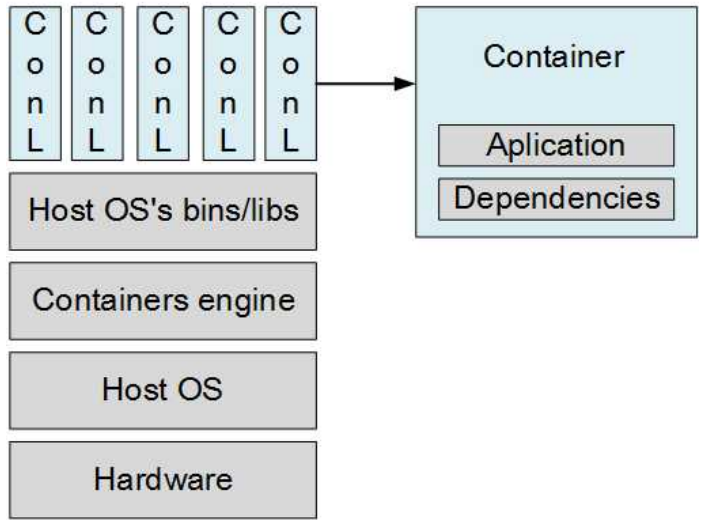

Hosted Hypervisor




Bare-Metal Hypervisor

# Background

- Virtualization 유형
  - Container-based virtualization
    - 운영체제 수준에서 가상화를 제공함
      - 호스트에서 다른 운영 체제 커널을 중복 실행하지 않고 여러 응용 프로그램을 실행 할 수 있음
      - 애플리케이션을 실행하는데 필요한 리소스가 있는 격리된 환경을 제공
    - 호스트 커널을 사용하여 가상 환경을 실행하는 가벼운 가상화 방식
    - 대표적인 세 가지 접근 방식
      - Linux-VServer
      - OpenVZ
      - Linux container(LXC)



# Background

- Differences of architecture
  - Container-based virtualization은 container에 전체 OS가 포함되지 않음
    - 컨테이너에서 응용 프로그램을 실행하는데 필요한 크기와 리소스가 동일한 응용 프로그램을 실행하는 VM의 크기 및 필요한 리소스보다 작음
  - Container-based virtualization은 가상 환경의 높은 밀도를 제공함

# Docker

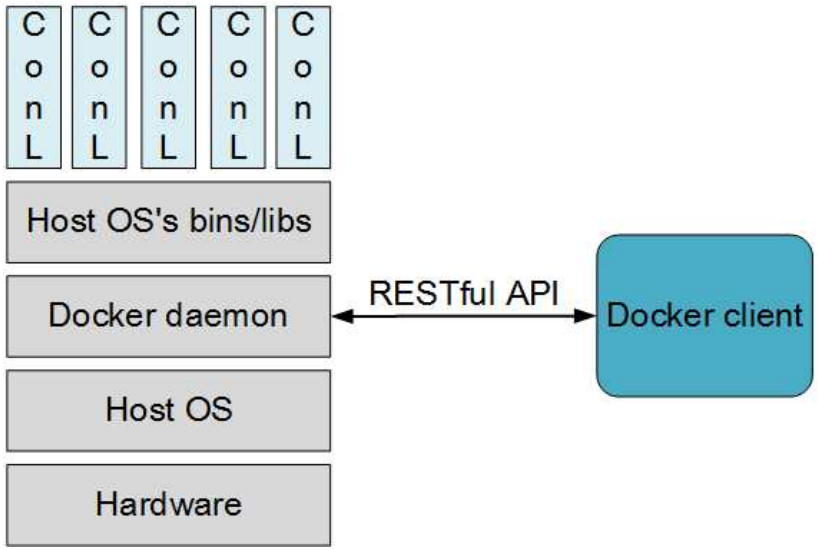
- 개요
  - Container-based virtualiztion
    - Container를 간단하고 안전하게 작성하고 제어하기 위한 인터페이스를 제공
    - 컨테이너 관리 및 배포 프로세스를 단순화하는 Third-party tools과 협력이 가능
    - 오픈 소스 컨테이너 기술
  - 분산 응용 프로그램을 작성, 전달 및 실행하는 기능을 함
  - 동일한 하드웨어에서 다른 기술보다 많은 가상 환경을 배포 할 수 있음

# Docker

- 개요
  - Puppet, Ansible 및 Vargrant와 DevOps 도구를 Docker와 통합하여 Docker container를 클라우드에 쉽게 배포 가능
  - Orchestration tools이 Docker container를 지원
    - Orchestration tools는 Docker에 대한 리소스 관리 및 스케줄링의 추상 계층을 제공함

# Docker

- 구성 요소
  - Docker Engine
    - Container-based virtualization architecture와 유사함
    - Docker daemon
      - Docker container를 실행하고 관리함
    - Docker client
      - 사용자 인터페이스를 제공
      - 사용자로부터 명령을 받아들인 다음 RESTful API를 통해 데몬으로 보냄



# Docker

- 구성 요소
  - Docker Hub
    - 이미지를 공유 할 수 있는 중앙 저장소
    - 사용자는 게시 된 이미지를 검색하여 Docker client로 다운로드 할수 있음
    - Dokcer가 이미지를 서명하고 확인 후 Hub에 제출 했을 때 사용자는 이미지의 진위성과 무결성을 확인할 수 있음

# Docker

- Docker container
  - Docker 0.9 버전 부터 LXC를 libcontainer로 대체함
    - LXC
      - Linux 커널의 cgroup과 namespaces를 이용하여 컨테이너를 관리하는 모듈
    - libcontainer
      - Docker에서 개발한 container driver
      - Docker가 LXC를 거치지 않고 커널의 container API를 직접 호출하여 효율성 향상

# Docker

- Docker container
  - Docker는 Linux 기능, namespaces 및 cgroup을 활용
    - Cgroup
      - 계정에 대한 메커니즘을 제공
      - 각 컨테이너의 프로세스가 액세스 할 수 있는 리소스를 제한
    - Namespaces
      - 운영체제 리소스를 다른 인스턴스로 래핑함
      - 다섯가지 namespaces
        - Mount
        - Hostname
        - IPC(Inter-Process Communcation)
        - PID(Process Identifiers)
        - Network

# Docker

- Docker security analysis
  - 개요
    - 멀티 테넌트 클라우드 시스템의 가상 환경에서 서비스를 실행할 때 보안 문제가 있음
    - Container는 호스트 커널과 직접 통신 할 수 있으므로 호스트 시스템에 침입 했을 때 쉽게 공격 가능

# Docker

- Docker security analysis
  - Docker Internal security
    - 시스템 및 공격자 모델
      - Dos, 권한 상승과 같은 유형의 공격의 위협
      - OS레벨 가상화 솔루션
        - Process isolation
        - Filesystem isolation
        - Device isolation
        - IPC isolation
        - Network isolation
        - Limiting of Resources

# Docker

- Docker security analysis
  - Docker Internal security
    - Process isolation
      - 컨테이너의 프로세스 ID번호 공간을 프로세스 공간과 분리하는 PID namespaces의 지원으로 작동
        - PID namespaces는 계층적임
      - 호스트는 PID namespaces 내부의 프로세스를 관찰하고 영향을 주지만 실행중인 다른 프로세스를 관찰하거나 수행 할 수 없음

# Docker

- Docker security analysis
  - Docker Internal security
    - Filesystem isolation
      - Mount namespaces를 사용하여 다른 컨테이너와 관련된 파일 시스템 계층을 격리함
      - 손상된 container로 인한 발생하는 위협 제한
        - 파일 시스템에 대한 쓰기 권한을 제거
        - 컨테이너의 프로세스가 파일 시스템을 다시 Mount하지 못하게 함
    - Device isolation
      - 컨테이너가 중요 장치 노드에 액세스 할 수 있으면 호스트 시스템에 손상을 줄 수 있음
      - Cgroup의 Device Whitelist Controller 기능을 통해 접근 할 수 있는 장치를 제한

# Docker

- Docker security analysis
  - Docker Internal security
    - IPC isolation
      - Container에서 실행되는 프로세스는 IPC 자원을 통해서만 통신 할 수 있음
      - 다른 IPC namespaces의 IPC 자원을 읽거나 쓸 수 없도록 함
    - Network isolation
      - MitM 및 ARP spoofing 같은 공격을 방지하기 위해 네트워크를 격리
      - 각 컨테이너에는 자체 IP 주소, IP 라우팅 테이블, 네트워크 장치가 있음

# Docker

- Docker security analysis
  - Docker Internal security
    - Limiting of Resources
      - Dos 공격을 막기 위해 각 container에 할당 된 자원을 제한함
      - Docker는 각 container에 할당 된 자원과 관련된 제한 및 제약 조건을 파악함

# Docker

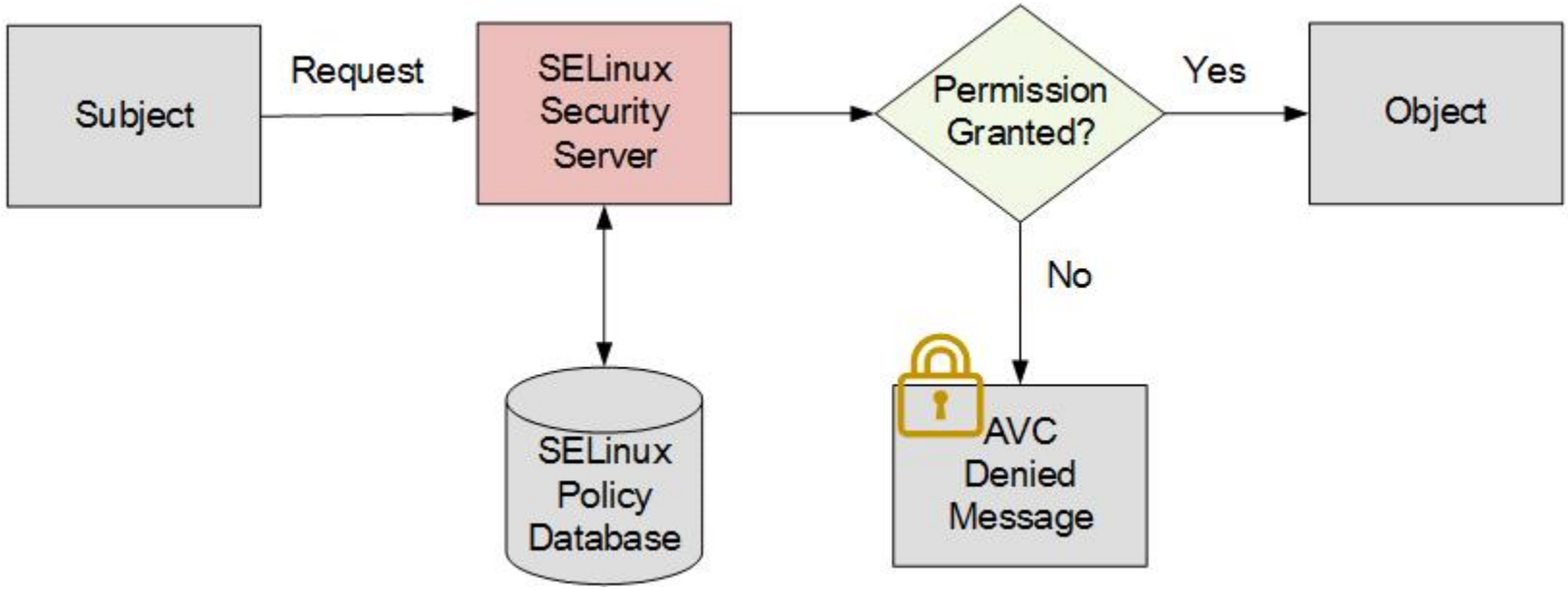
- Docker security analysis
  - Docker and kernal security system
    - 호스트 시스템의 보안을 위한 커널 보안 시스템
      - Linux 기능
      - Docker가 지원하는 LSM(Linux Security Module)
        - Apparmor 및 SELinux
    - Linux 기능
      - 각 프로세스에 할당된 권한을 제한함
      - Container의 루트 기능 중 일부를 제거해도 유용성이나 기능에는 영향을 미치지 않지만 시스템 보안은 효과적으로 향상됨

# Docker

- Docker security analysis
  - Docker and kernal security system
    - SELinux
      - SELinux는 Linux 시스템의 보안을 강화하는 기능임
      - 표준 DAC이 수행 된 후 Mandatory Access Control이라는 권한 검사의 추가 계층을 제공
      - 시스템 관리자는 레이블을 사용하여 프로세스와 시스템 오브젝트 간의 액세스를 제어하는 규칙을 작성
      - Type enforcement 및 MCS 적용
        - Type enforcement는 container의 프로세스에서 호스트를 보호
        - MCS적용은 container를 다른 container로부터 보호함

# Docker

- Docker security analysis
  - Docker and kernal security system
    - SELinux

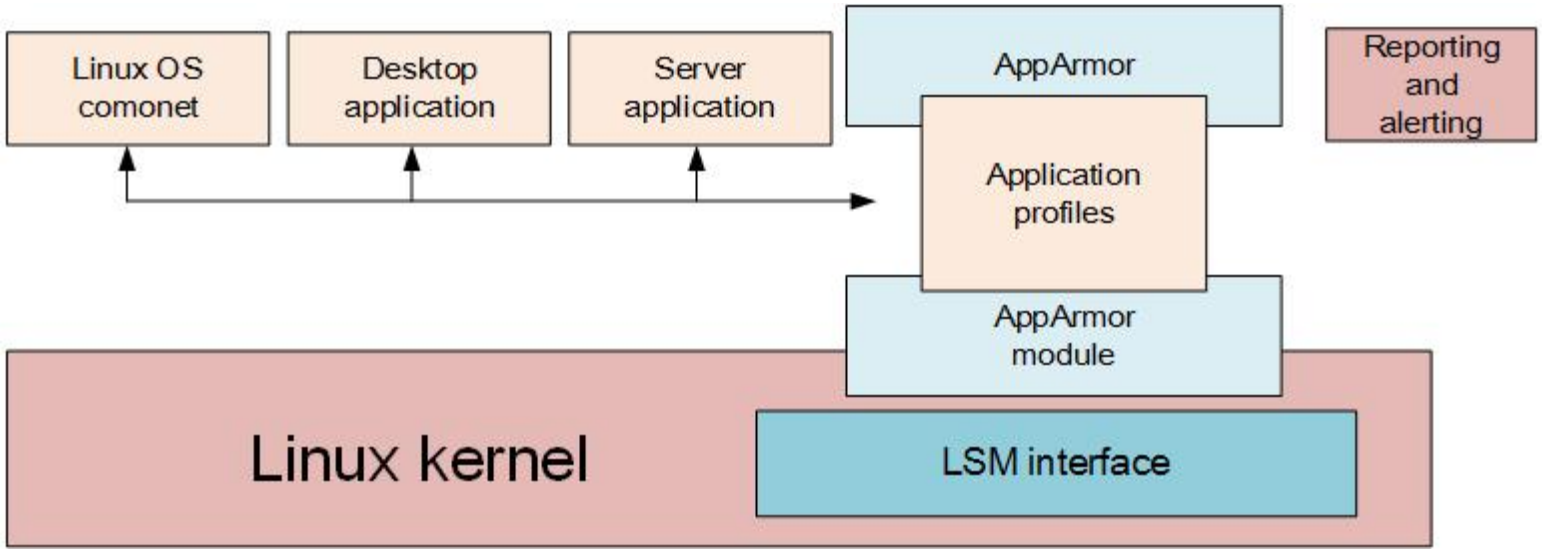


# Docker

- Docker security analysis
  - Docker and kernal security system
    - AppArmor
      - 접근 제어를 기반으로하는 Linux 보안 강화 모델
        - 개별 프로그램에 대한 범위를 제한함
      - Enforcement mode
        - 프로파일에 정의 된 정책을 시행함
      - Complain/learning mode
        - 프로필 정책 위반이 허용되지만 기록이 됨
          - 이 로그는 나중에 프로파일을 개발하는 데 유용 할 수 있음
      - Docker는 새 container를 시작할 때 미리 정의 된 AppArmor 프로파일을 로드하기 위한 인터페이스를 제공
        - 프로세스가 프로파일에 따라 제한되도록 하기 위해 Enfocement mode에서 container로 로드됨

# Docker

- Docker security analysis
  - Docker and kernal security system
    - AppArmor



# Conclusion

- Docker는 namespaces, cgroup 및 filesystem을 사용하여 container에 격리 및 리소스를 제한을 함
- Container-based virtualization은 Hypervisor-based virtualizaiton보다 높은 밀도를 제공
- AppArmor 또는 SELinux와 같은 Linux 커널 보안 솔루션을 통해 보안 수준을 높일 수 있음

감사합니다!