

통합인증 시스템 보안

팀명 : S-Guard
지도 교수 : 양환석 교수님
팀장 : 이유근
팀원 : 이종화, 안혜민

2016. 05

중부대학교 정보보호학과

목 차

1. 서론	1p
2. 관련연구	
2.1 방화벽	4
2.2 SSO	7
2.3 IPS/IDS	11
2.4 vsFTP	13
2.5 백업서버	14
3. 본론	
3.1 방화벽 구축	16
3.2 SSO 구축	20
3.3 IPS/IDS 구축	35
3.4 vsFTP 운영	43
3.5 백업서버 구축	49
4. 결론	54
5. 참고 자료	54
6. 발표 자료	55

1. 서론

여러 개의 사이트를 운영하는 대기업이나 인터넷 관련 기업이 각각의 회원을 통합 관리할 필요성이 생김에 따라 개발된 방식으로, 1997년 IBM이 개발하였으며 우리나라에는 2000년 코리아닷컴이 처음 도입하였고 이후 삼성전자 및 SK 등이 도입하며 활성화되어, OK캐쉬백, SK플래닛, 올레닷컴 등 다양한 사이트에 적용되고 있다.



<SK플래닛 통합ID (SSO의 예)>

최근 회사들이 그룹화되거나 대형화가 되어 여러 사이트들을 통합 관리하는 경우 SSO를 사용하게 된다. SSO 사용 시 관리자는 하나의 아이디로 모든 고객을 통합 관리할 수 있게 되고, 기존 사용자는 정보 변경 없이 하나의 아이디로 여러 사이트들을 로그인하여 쉽게 이용할 수 있다.

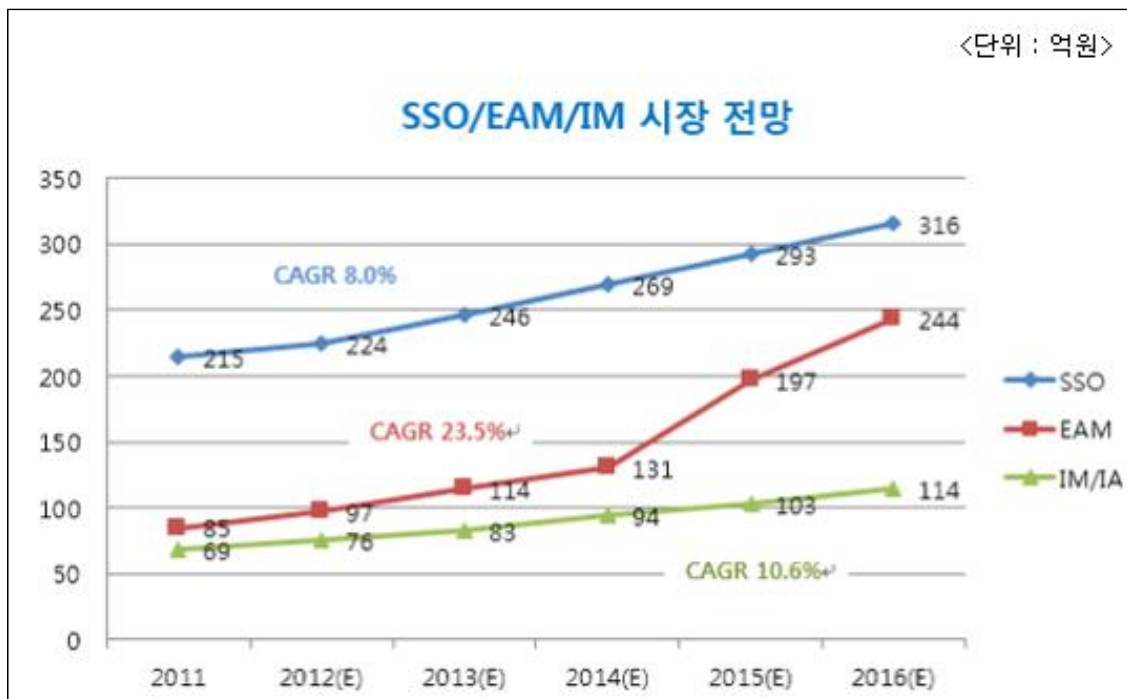
또한 SSO를 도입하게 되면 각각의 시스템마다 인증 절차를 밟지 않고도 1개의 계정만으로 다양한 시스템에 접근할 수 있어 ID, 패스워드에 대한 보안 위험 예방과 사용자 편의 증진, 인증 관리 비용의 절감 효과가 있어 향후 더욱 시장이 커질 것으로 전망된다.

특히 권한관리시스템(EAM)과 함께 사용할 경우 보안성과 효율성을 함께 갖춘 통합인증시스템으로 활용할 수 있어 향후 더욱 인기를 끌 것으로 전망된다.

암호인증(SSO제품) 시장 규모 및 전망

- 정보 보안산업은 제품과 서비스로 나뉘며, SSO는 정보 보안 제품 중 암호인증 제품에 속함
- 정보 보안 제품(대분류) > 암호/인증(중분류) > 싱글사인 온(SSO)
- 암호인증 시장은 2012년 1,250억의 규모를 형성하며, 지속적으로 성장세를 타며 2016년 1,885억 원의 규모로 성장할 것으로 추정 (연평균 성장률(CAGR 10.9%)
- 암호인증은 아래의 8개의 제품군으로 구분되며, EAM(23.5%), 보안 스마트카드(13.5%)의 성장률이 높은 것으로 조사 됨

<암호인증 제품>			
보안 스마트카드 13.5%	H/W 토큰(HSM) 8.4	일회용 비밀번호(OTP) 11.8%	공개키 기반구조(PKI) 8.1%
싱글 사인온(SSO) 8.0%	통합 접근관리(EAM) 23.5%	통합계정 관리(IM/IAM) 10.6%	공인/사설 인증 톨 12.8%



SSO(싱글사인온)는 양면의 동전, 한번에 통하지만 한번에 당할 수도 있어

헤럴드POP

또 인증 정보를 재사용하거나 보안 검증을 받지 않은 암호 모듈을 사용하는 경우, 액티브X 컨트롤 관련해서도 SSO의 취약점이 노출될 수 있다. 이에 전문가들은 SSO일수록 보안 기능을 더욱 촘촘히 갖춰야 한다고...

↳ 통합암호 'SSO'...단 한번의 편리함... 헤럴드경제

하지만 이러한 SSO의 통합 암호가 노출된다면 주요 정보가 한꺼번에 털릴 수 있어 각별한 주의가 요구된다. 해킹을 당하면 파급효과가 일파만파 커질 수 있다는 취약점을 안고 있다. 예를 들어 공격자가 사용자로 가장해 서비스를 제공받거나 정당한 서버로 가장해 사용자 정보를 수집하는 경우다. 이런 경우에는 한 번의 로그인으로 수많은 사이트들의 정보가 노출될 것이다.

이에 대한 취약점을 보완하기 위해 방화벽 구축에서 허용되지 않는 IP는 SSO 서버에 접속하지 못하도록 하고 만약 설정하지 못한 IP가 들어온다면 침입탐지 시스템을 구축하여 예상하지 못한 서비스를 탐지하도록 구현하게 되었다.

2. 관련연구

2.1 방화벽

브릿지란

네트워크 브리지(network bridge, 문화어: 망다리)는 OSI 모델의 데이터 링크 계층에 있는 여러 개의 네트워크 세그먼트를 연결해 준다. 레이어 2 스위치는 브리지라는 용어와 같은 뜻으로 간헐적으로 사용된다. 브리지는 물리 계층에 있는 네트워크 세그먼트를 연결해 주는 장치인 리피터, 네트워크 허브와 비슷하지만 브리지를 사용하면 단순히 주변 네트워크 세그먼트로 다시 전송한다기보다는 특정 네트워크로부터 오는 트래픽을 관리하게 된다. 이더넷 망에서 "브리지"라는 용어는 공식적으로 IEEE 802.1D 표준을 따르는 장치를 뜻한다.

브리지는 허브나 리피터보다 더 복잡한 경향을 가진다. 브리지는 들어오는 데이터 패킷을 분석하여 브리지가 주어진 패킷을 다른 세그먼트의 네트워크로 전송할 수 있는지를 결정할 수 있다.

브리지 작업이 OSI 모델의 데이터 링크 계층에서 이루어지므로 브리지는 브리지가 수신하는 각 프레임의 데이터로부터 정보를 처리한다. 이더넷 틀에서 이것은 프레임의 출발지와 도착지의 MAC 주소를 제공한다. 브리지는 두 가지 방식을 사용하여 MAC 주소가 가지는 네트워크 세그먼트를 결정한다.

투명 브리징(Transparent bridging) - 이 방식은 전송되는 새로운 노드 주소에 따라 이전에 겪은 대로 해당 노드의 경로를 찾아가는 방식이다. 보내는 쪽이 네트워크 브리지가 있는지를 알고 있지 않아도 되는 방식이다.

소스 루트 브리징(Source route bridging) - 이 방식에서는 도착 네트워크 세그먼트로 향하는 루트를 찾기 위해 두 개의 프레임 형태가 쓰인다. 개별 노선(Single-Route; SR) 프레임은 대부분의 네트워크 트래픽을 구성하고 도착지를 설정하는 반면 전체 노선(All-Route; AR) 프레임은 루트를 찾는 데 사용된다.

iptables란

iptables는 리눅스상에서 방화벽을 설정하는 도구로서 커널 2.4 이전 버전에서 사용되던 ipchains를 대신하는 방화벽 도구이다.

iptables는 커널상에서의 netfilter 패킷 필터링 기능을 사용자 공간에서 제어하는 수준으로 사용할 수 있다.

패킷 필터링이란 지나가는 패킷의 헤더를 보고 그 전체 패킷의 운명을 결정하는 것을 말한다. 일반적으로 패킷은 헤더와 데이터를 가진다.

헤더에 필터링할 정보인 출발지 IP:PORT, 도착지 IP:PORT, checksum, 프로토콜 옵션 등을 가지며 데이터는 각각의 전송 데이터가 들어간다.

특정 조건을 가지고 있는 패킷에 대해 허용(ACCEPT)과 차단(DROP) 등을 지정할

수 있으며, 특정 조건 등을 통해 다양한 방식의 패킷 필터링과 처리 방식을 지원한다.

iptables 정책은 여러 구분으로 나뉘지며 중요한 부분은 Chain이다. Chain은 패킷이 조작될 상태를 지정하며 iptables에 내장된 기본 Chain은 다음과 같다.

(기본 Chain은 영구적이며 삭제가 불가능하다. 이외에 -N 옵션으로 지정하는 사용자 정의 Chain이 있다.)

Chain INPUT	서버로 들어오는 기본 정책
Chain FORWARD	서버에서 forwarding 기본 정책
Chain OUTPUT	서버에서 나가는 기본 정책

Linux Server를 목적지로 삼는 모든 패킷은 INPUT Chain을 통과하고

Linux Server에서 생성되어 외부로 보내지는 모든 패킷은 OUTPUT Chain을 통과하게 된다.

FORWARD Chain의 경우 현재의 Linux Server가 목적지가 아닌 패킷이 통과하는 Chain이다.

(FORWARD Chain은 NAT(네트워크 공유) 기능 사용을 위해 사용된다.)

iptables의 구조

먼저 iptables에 대해 살펴보도록 하자.

iptables는 다음의 구조로 구성된다.

iptables -A INPUT -s [발신지] --sport [발신지 포트] -d [목적지] --dport [목적지 포트] -j [정책]

iptables 명령	
-A	새로운 규칙을 추가한다.
-D	규칙을 삭제한다.
-C	패킷을 테스트한다.
-I	새로운 규칙을 삽입한다.
-R	새로운 규칙으로 교체한다.
-L	새로운 규칙을 출력한다.
-F	체인의 모든 규칙을 삭제한다.
-Z	모든 체인의 패킷과 바이트 카운터 값을 0으로 만든다.
-N	새로운 체인을 만든다.
-X	체인을 삭제한다.
-P	기본 정책을 변경한다.

iptables 옵션	
-p	패킷의 프로토콜의 포트번호 또는 이름을 명시한다. (ex : tcp, udp, 21, 22)
-s	패킷의 발신지를 명시한다. (ex : address[/mask])
-d	패킷의 도착지를 명시한다.
-i	규칙을 적용할 인터페이스 이름을 명시한다. (ex : eth0, eth1)
-j	규칙에 맞는 패킷을 어떻게 처리할 것인가를 명시한다.
-y	접속 요청 패킷인 SYN 패킷을 허용하지 않는다.
-f	두 번째 이후의 조각에 대해 규칙을 명시한다.

iptables 정책 순서

모든 방화벽은 순차적 실행이다.

즉 등록 순서에 있어서 먼저 등록한 대해서 효력이 유효하기 때문에 등록 시에는 순서가 매우 중요하다.

모든 입출력 패킷에 대해 거부하는 설정이 먼저 등록되면 그 이후에 포트를 열어주는 설정을 하여도 효과가 없다.

그러므로 허용하는 정책을 먼저 정의한 다음 거부하는 정책을 설정해야 한다.

// 아래와 같이 설정하면 우선적으로 22번 포트가 열린 후 나중에 22번~30번 포트가 막히기 때문에 SSH 접속이 가능하다.

```
# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

```
# iptables -A INPUT -p tcp --dport 22:30 -j DROP
```

```
# iptables -L
```

```
Chain INPUT (policy ACCEPT)
```

```
target      prot opt      source      destination
```

```
ACCEPT      tcp  --  anywhere   anywhere    tcp dpt:ssh
```

```
DROP        tcp  --  anywhere   anywhere    tcp dpts:ssh:30
```

// 아래와 같이 설정하면 우선적으로 22번~30번 포트가 막히기 때문에 뒤에서 아무리 22번 포트를 열어도 외부에서 SSH로 접속할 수 없게 된다.

// iptables로 입력할 경우 바로 적용이 되기 때문에 원격에서 작업할 경우엔 주의하자.

```
# iptables -A INPUT -p tcp --dport 22:30 -j DROP
```

```
# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```


2.2 SSO

SSO란

Single Sign On이란?

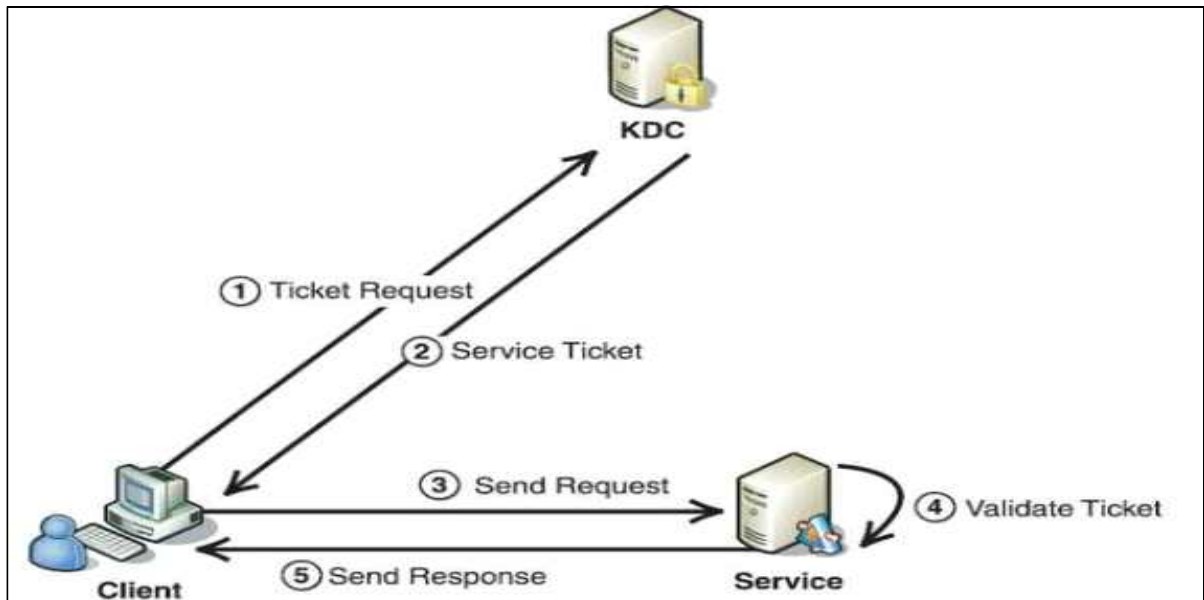
보통 사용자들은 여러 서비스 사이트를 이용함에 있어 여러 개의 아이디와 패스워드를 기억하여 사용한다. 이러한 불편함을 해결하고 관리 측면에서 효과적인 방법으로 제안된 인증 시스템이 SSO이다. SSO는 단 한 번의 로그인만으로 기업의 각종 시스템이나 인터넷 서비스에 접속하게 해주는 보안 응용 솔루션이다. 각각의 시스템마다 인증 절차를 밟지 않고도 1개의 계정만으로 다양한 시스템에 접근할 수 있어 ID, 패스워드에 대한 보안 위험 예방과 사용자 편의 증진, 인증관리비용의 절감 효과가 있다. 이러한 Single Sign On 환경은 메인 서버와 client 서버 간 티켓을 사용하여 인증하는 Kerberos와 네트워크상에서 개인 정보 혹은 파일이나 디바이스 정보들을 찾아보는 것을 가능하게 만들어주는 Ldap을 이용하여 구축할 수 있다.

Kerberos란

Kerberos는 승인된 컴퓨터 네트워크 내에서 서비스 요구를 인증하기 위한 방법으로 미국의 MIT Athena 프로젝트에서 개발되었다. Kerberos는 사용자가 인증 과정으로부터 암호화된 "티켓"을 요청할 수 있게 해주는데 티켓은 서버에 특정 서비스를 요구하는데 사용될 수 있다.

Kerberos 시스템은 이 티켓이라는 개념에 중점을 둡니다. 티켓은 사용자나 서비스를 식별하는 전자 정보 집합이다. 운전면허증이 사용자의 신원을 확인해 주고 소지하고 있는 운전면허를 나타내듯이, 티켓은 사용자와 사용자의 네트워크 액세스 권한을 식별한다. Kerberos 기반 트랜잭션을 수행하는 경우(예: 다른 시스템에 원격 로그인하는 경우) 티켓에 대한 요청이 투명하게 KDC(키 배포 센터)로 전송된다. KDC는 데이터베이스에 액세스하여 사용자의 신원을 인증하고 다른 시스템에 액세스할 수 있는 권한을 부여하는 티켓을 반환한다. "투명하게"란 명시적으로 티켓을 요청할 필요가 없음을 의미한다. 요청은 rlogin 명령의 일부로 수행되며, 인증된 클라이언트만 특정 서비스에 대한 티켓을 가져올 수 있으므로 사용 중인 ID로는 다른 클라이언트가 rlogin을 사용할 수 없다.

티켓은 특정 속성과 연관된다. 예를 들어 티켓이 전달 가능 티켓일 수 있습니다. 이는 새 인증 프로세스 없이도 다른 시스템에서 티켓을 사용할 수 있음을 의미한다. 또한 후일 자 티켓일 수도 있습니다. 이는 지정된 시간까지 티켓이 유효하지 않음을 의미한다. 예를 들어 티켓을 어떻게 사용하여 어떤 사용자가 어떤 유형의 티켓을 얻을 수 있는지는 정책에 의해 설정된다. 정책은 Kerberos 서비스가 설치되거나 관리될 때 결정된다.



1. 텔넷/ssh와 같은 원격접속 또는 기타 이와 비슷한 로그인 요청을 통해, 다른 컴퓨터에서 서버에 액세스하기 원한다고 가정할 경우 client의 요청을 받아들이기 전에, 커버로스 "티켓"을 요구한다.

2. 티켓을 받기 위해, 당신은 먼저 인증 서버에 인증을 요구한다. 인증 서버는 당신이 입력한 패스워드에 기반하여 "세션 키"와, 서비스 요구를 나타내는 임의의 값을 만든다. 세션 키는 사실상 "티켓을 부여하는 티켓"이다.

3. 그 다음에 세션 키를, 티켓 부여 서버, 즉 TGS (ticket-granting server)에 보낸다. TGS는 인증 서버와 물리적으로는 동일한 서버에 있을 수 있다. 그러나 지금은 다른 서비스를 수행한다. TGS는 서비스를 요청할 때 서버에 보낼 수 있는 티켓을 돌려준다.

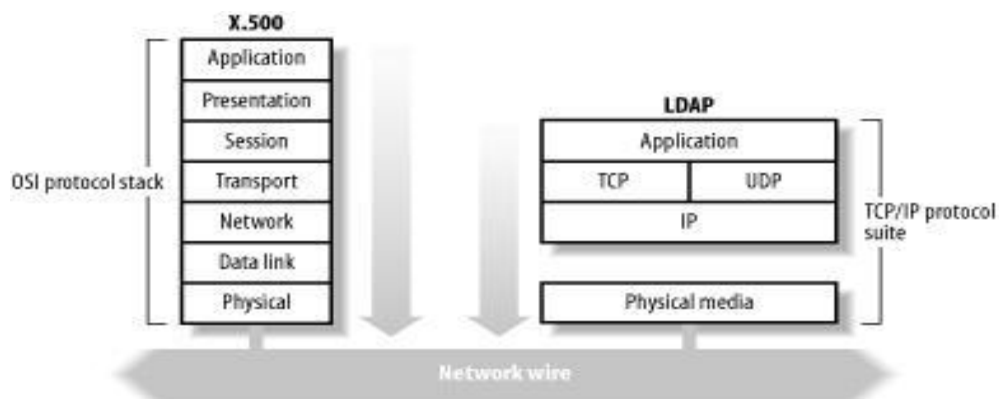
4. 세션키를 넘겨받은 해당 TGS는 티켓을 거절하거나, 또는 받아들여서 해당 서비스를 수행한다.

5. TGS로부터 받은 티켓은 발송일자와 시간이 적혀있기 때문에, 일정 시간 동안 (대체로 8시간 동안) 내에는 재인증 없이도 동일한 티켓으로 다른 추가 서비스를 요청할 수 있다. 티켓을 제한된 시간 동안에만 유효하게 만듦으로써, 후에 어떤 사람이 그것을 사용할 수 없도록 만든다.

LDAP

LDAP(Lightweight Directory Access Protocol)이란 네트워크 상에서 조직이나 개인 정보 혹은 파일이나 디바이스 정보 등을 찾아보는 것을 가능하게 만든 소프트웨어 프로토콜이다.

LDAP은 네트워크 상의 디렉토리 서비스 표준인 X.500의 DAP(Directory Access Protocol)를 기반으로 한 경량화(Lightweight)된 DAP 버전이다.



X.500의 DAP는 OSI 전체 프로토콜 스택을 지원하며 운영에 매우 많은 컴퓨팅 자원을 필요로 하는 아주 무거운 프로토콜이다. 이런 DAP의 복잡성을 줄이기 위해 만들어진 LDAP은 TCP/IP 레벨에서 더 적은 비용으로 DAP의 많은 기능적인 부분을 조작할 수 있도록 설계되었다고 한다. LDAP은 사용자 로그인 처리(SSO)나 전화번호 조회, 회사 내에서의 부서와 같은 항목에서 유용하다. 또한, LDAP은 LDAP 서버 간의 데이터 복제가 실시간으로 잘 되기 때문에 원격지에서 관리되고 있는 정보들을 근거리의 LDAP 서버로 복제해서 사용하는 경우도 많이 있다.

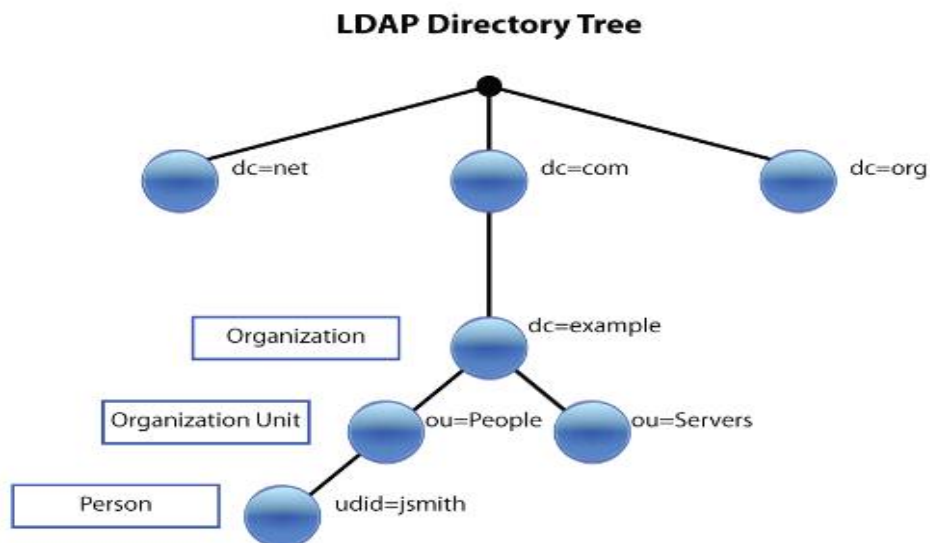
Entry
LDAP 트리(DIT)에서 각 노드를 의미 하나의 데이터를 나타냄 RDBMS의 레코드와 동일한 의미

DN (Distinguished Name)
각 엔트리의 위치와 고유성을 나타냄 파일 시스템처럼 해당 엔트리의 이름에 상위 엔트리의 엔트리 이름을 붙여서 식별 LDAP 트리 경로 전체를 의미

RootDN
말그대로 root(관리자)의 DN

LDIF (LDAP Data Interchange Format)
LDAP 엔트리의 데이터를 읽고 쓸 수 있는 텍스트 형식으로 보여주는 것

objectclass
속성(attribute)를 갖으며, 자바의 클래스처럼 '틀'이 됨 하나의 속성 키에 매핑되는 다수의 값을 가질 수 있다.
ou (organization unit)
cn (common name)
Suffix : DIT의 최상위 Entry



LDAP 디렉토리 서비스 모델은 항목(오브젝트라고도 함)을 기준으로 한다. 각 항목은 이름, 주소 및 유형 같은 하나 이상의 속성으로 구성된다. 유형은 일반적으로 공통 명의 경우 cn 또는 전자 우편 주소의 경우 mail 같은 스트링으로 구성한다.

이러한 LDAP의 정보들은 여러 대의 LDAP 서버들이 LDAP 디렉토리 트리를 구성하는 자료를 갖고 있다. client는 최초 LDAP 서버에 연결을 위해 요청을 보내면 server는 요청에 대해 응답하며 client는 그에 대해 엔트리를 참조하여 동일한 디렉토리를 확인할 수 있다.

2.3 IDS

서론

방화벽(firewall)과 함께 또 하나의 주목받는 보안 솔루션으로는 침입탐지시스템이라 불리는 IDS (Intrusion Detection System)가 있다. 앞에서 살펴보았던 방화벽이 IP나 포트를 기준으로 비정상 트래픽을 차단하는 것이라면 IDS는 포트에 대한 정보뿐만 아니라 패킷의 데이터까지 분석하여 정상적인 트래픽 여부를 결정하는 것으로 이를 통해 실제로 인터넷을 통해 어떠한 위협이 발생하고 있는지를 분석할 수 있게 된다. 이를테면 사내 게이트웨이에 IDS를 설치해 둔다면 사내를 오가는 트래픽을 분석하여 비정상적인 트래픽이 보인다면 바로 로그에 남기도록 할 수 있다. 그리고, 방화벽의 경우 열려있는 포트인 80번을 통해 worm 등과 같은 비정상적인 공격이 있을 경우에는 차단할 수 있는 방법이 없으나, IDS에서는 이러한 공격을 인식할 수 있는 기능이 있다. 따라서 IDS와 방화벽을 연동하여 IDS에 탐지한 비정상 트래픽을 차단하는 기능을 갖춘 솔루션도 있다. 그러나 그렇다고 해서 IDS 가 모든 공격을 100% 정확하게 인식하여 탐지하는 것은 아니며, 몇 가지 근본적인 한계를 가지고 있다. 일단 가장 큰 문제는 “오탐율”이다. IDS에서 오탐은 크게 두 부류로 나눌 수 있는데, 실제로 비정상 트래픽임에도 불구하고 이를 탐지하지 못하는 경우를 “False Negative”라 하며, 비정상 트래픽이 아닌 정상적인 트래픽을 탐지하는 경우를 “False Positive”라 한다. 실제 IDS를 운영하다 보면 위와 같은 오탐이 매우 높다는 것을 알 수 있으며 이 오탐율을 줄이는 것이 IDS의 관건이라 할 수 있겠다. 또한 앞에서 언급한 바와 같이 침입에 대한 탐지는 할 수 있어도 이 공격에 대한 대응은 부족하다는 한계가 있다.

Snort를 이용한 IDS

IDS가 침입을 탐지하는 방식은 몇 가지가 있지만 가장 대표적인 방식은 룰 기반의 패턴 매칭 방식으로 이는 사전에 정의된 패턴 또는 룰에 따라 트래픽이 매칭 되었을 경우 공격 또는 비정상 트래픽으로 판단하는 것이다. 여기에서는 이러한 룰 기반의 공개 IDS 프로그램 중 가장 대중적으로 사용되고 있는 snort라는 프로그램을 이용하여 비정상 트래픽을 탐지할 수 있다.

Snort 란 무엇인가?

Snort는 “sniffer and more”라는 말에서 유래되었는데, 처음 공개되었을 때는 코드도 얼마 되지 않는 단순한 패킷 스니퍼 프로그램이었다. 그러나 이후 현재의 IDS와 패킷 스니퍼(sniffer) 네트워크의 패킷을 읽어 보여주는 기능 패킷 로거(logger) 모니

터링 한 패킷을 저장하고 로그에 남기는 기능 network IDS 네트워크 트래픽을 분석하여 공격을 탐지하는 기능으로 bufferoverflow나 port scan, IP scan 등 대부분의 공격을 탐지할 수 있다. 같이 rule을 이용한 분석 기능이 추가되고, 커뮤니티를 통하여 계속적인 기능 보완과 향상을 통해 지금과 같이 다양한 기능과 탁월한 성능을 갖춘 프로그램이 되었다. snort는 공식 홈페이지인 <http://www.snort.org>를 통해 지속적인 업그레이드를 제공하고 있다.

snort는 오픈 소스로 개발 중인 패킷 캡처 라이브러리인 libpcap을 사용하여 패킷을 캡처하고, 수집된 패킷이 사전에 정의된 snort 공격 룰과 비교하여 만약 매칭 되었을 경우 syslog를 통해 로그를 남기거나 특정 디렉토리의 특정 파일 또는 database에 남기도록 할 수 있다.

snort 의 구조

snort 프로그램은 몇 가지 구성 요소들이 플러그인 형태로 이루어져 있어 쉽게 각자의 환경에 따라 변경하고 수정할 수 있도록 되어 있는데, 기본적으로 다음과 같은 4가지 구성요소로 이루어져 있다.

- ① 스니퍼(sniffer)
- ② preprocessor
- ③ 탐지엔진
- ④ 로깅(출력)

snort는 먼저 스니퍼를 통해 snort IDS를 통과하는 모든 패킷을 수집하게 된다. 여기에서 수집된 데이터는 바로 룰 기반의 탐지 엔진을 거치지 않고 그 전에 preprocessor를 통해 보다 효율적인 공격 탐지를 위해 HTTP 인코딩 플러그인이나 포트스캔 등 몇 가지 플러그인을 먼저 거치면서 매칭이 되는지 확인하게 된다. 물론 preprocessor 역시 모듈화 되어 있어 각자의 환경에 불필요하다면 disable 할 수 있다. 이를테면 RPC 트래픽에 대해 탐지할 필요가 없다면 RPC 관련 preprocessor를 주석처리하면 된다. 그리고 preprocessor를 통과한 패킷은 snort IDS의 핵심이라 할 수 있는 룰 기반의 탐지엔진을 거치면서 사전에 정의된 탐지룰과 매칭 되는지 확인하게 된다. 만약 룰에 매칭 되었을 경우에는 사전에 정의된 정책에 따라 로그에 남게 되고, 그렇지 않은 경우 통과를 하게 된다.

2.4 vsFTP

FTP 란

FTP란 File Transfer Protocol 의 약자로, 인터넷상의 컴퓨터들 간에 파일을 교환하기 위한 표준 프로토콜로서 가장 간단한 방법이다.

화면에 표시할 수 있는 웹 페이지와 관련 파일들을 전송하는 HTTP(Hypertext Transfer Protocol), 전자우편을 전송하는 SMTP(Simple Mail Transfer Protocol) 등과 같이, FTP도 역시 인터넷의 TCP/IP 응용 프로토콜 중의 하나이다.

FTP는 웹 페이지들을 인터넷상에서 모든 사람이 볼 수 있도록 하기 위해 저작자의 컴퓨터로부터 서버로 옮기는 과정(upload)에서 사용된다. 또한, 다른 서버들로부터 자신의 컴퓨터로 프로그램이나 파일들을 옮기는 과정(download) 하는 데에도 사용된다.

FTP의 가장 큰 장점은 월드와이드웹보다 빠른 속도로 전송받을 수 있다는 것이다. 월드와이드웹이 사용하기는 편리하지만 상대적으로 속도도 느리고, 안정적이지 못하다는 단점이 있다.

이러한 단점을 보완한 vsFTP 가 있다.

vsFTP 란

서버 운영에 있어서 가장 중요한 요소는 보안 문제이다.

기존의 proftp와 wu-ftp는 보안 홀에 대한 보고가 빈번히 일어나 서버의 보안이 흔들린 경우가 많다. vsftp는 보안 부분을 특히 강조한 서버 데몬으로서 REDHAT, SUSE, OPEN-BSD에서 기본 FTP 데몬으로 채택하고 있으며 vsftp를 매우 신뢰하고 있다. Very Secure FTP의 약자로서 일반적인 FTP는 보안상 취약하여, 패킷을 스니피하면 로그인 계정, 암호 및 각종 명령어들을 쉽게 확인할 수 있다. 그리고 FTP보다 빠른 데이터전송, 안정성, 보안 요소를 가지고 있는 것이 특징이다.

지원하는 대표적인 기능으로는 가상 IP 지원, 가상 유저 지원, Standalon 과 inetd 지원, 강력한 사용자 설정, 전송 대역폭 조절 기능, 환경설정 파일을 IP 별로 독립적 운영 지원, IP 별 제한 기능 등이 있다. 또한 config 파일의 설정 문법도 아주 간단해서 FTP 서버 관리를 쉽게 할 수 있다.

2.5 백업서버

Rsync 란

Unix 시스템에서의 한 소프트웨어로써, 파일이나 디렉토리 들을 하나의 저장소에 서 다수의 다른 저장소로 동기화 할 수 있는 소프트웨어이다. rsync의 큰 특징중 하나는 유사한 소프트웨어나 프로토콜에 찾아 볼 수 없는 데이터의 전송이 한 방향으로만 전송이 이루어 지는 것이다. rsync는 파일을 복사하거나 혹은 디렉토리내의 내용들을 확인해볼 수도 있으며, 부가적으로 압축을 하거나, 재귀적으로 사용할 수도 있다.

rsync가 데몬모드로 실행이 될때에는, TCP 포트 873을 기본으로 listen 하며, 기본적으로 동일한 rsync 프로토콜을 통해 파일을 공유하거나, RSH, SSH와 같은 프로토콜을 이용해서 파일을 제공하기도 한다. rsync 이외의 다른 프로토콜을 사용할 때에는 local이나, remote host 모두 rsync 클라이언트가 설치되어 있어야 한다.

rsync는 GNU GPL 라이선스를 따르고 있으며, 무료 소프트웨어이다. 대중적으로 많이 사용되고 있는 소프트웨어이기도 한다. 특히 리눅스의 archive와 같은 mirror 사이트의 경우 rsync를 통해 원본 저장소로 부터 동기화를 하는데에 대표적으로 사용되고 있는 소프트웨어이다.



SSH Key란?

-서버에 접속 할 때 비밀번호 대신 key를 제출하는 방식이다.

SSH Key는 언제 사용하는가?

- 비밀번호 보다 높은 수준의 보안을 필요로 할 때
- 로그인 없이 자동으로 서버에 접속 할 때

SSH Key가 동작하는 방식

-SSH Key는 공개키(public key)와 비공개 키(private key)로 이루어지는데 이 두개의 관계를 이해하는 것이 SSH Key를 이해하는데 핵심이다. 키를 생성하면 공개키와 비

공개키가 만들어진다. 이 중에 비공개키는 로컬 머신에 위치해야 하고, 공개키는 리모트 머신에 위치해야 한다. (로컬 머신은 SSH Client, 원격 머신은 SSH Server가 설치된 컴퓨터를 의미한다.)

SSH 접속을 시도하면 SSH Client가 로컬 머신의 비공개키와 원격 머신의 비공개키를 비교해서 둘이 일치하는지를 확인한다.



SSH Client

Private Key



SSH Server

Public Key

crontab 이란

crontab 은 리눅스서버에서 예약된 작업을 실행시키는 스케줄러의 역할을 한다.

요일부터 월, 일, 시, 분 단위로 시간을 설정할 수 있어 서버 관리에 유용하게 사용되는 명령어이다.

crontab 명령어 옵션	
crontab -e	Linux crontab 예약작업 추가 및 수정
crontab -l	Linux crontab 예약작업 리스트 확인
service crond start	Linux crond 예약작업 시작
service crond stop	Linux crond 예약작업 종료
service crond restart	Linux crond 예약작업 재시작

작업형식

[분] [시] [일] [월] [요일] [실행하고자 하는 명령어]	
분	0부터 59까지의 범위안에 명령어가 실행되기를 원하는 분 설정
시	0부터 23까지의 범위안에 명령어가 실행되기를 원하는 시간 설정
일	1부터 31까지의 범위안에 명령어가 실행되기를 원하는 일 설정
월	1부터 12까지의 범위안에 명령어가 실행되기를 원하는 월 설정
요일	1부터 7까지의 범위안에 명령어가 실행되기를 원하는 요일 설정
명령어	실행되기를 원하는 명령어를 입력

3. 본문

3.1 방화벽 구축

브릿지 방화벽 구축

한가지 예를 든다면 우리가 네트워크를 사용하는데 있어 허브는 필수라고 할 수 있다. 하지만 허브는 IP를 가지지 않으면서 컴퓨터들의 인터넷을 연결해 주는 역할을 한다. 어떻게 보면 Hub(허브) 역시 Bridge라고 말할 수 있을 것이다.

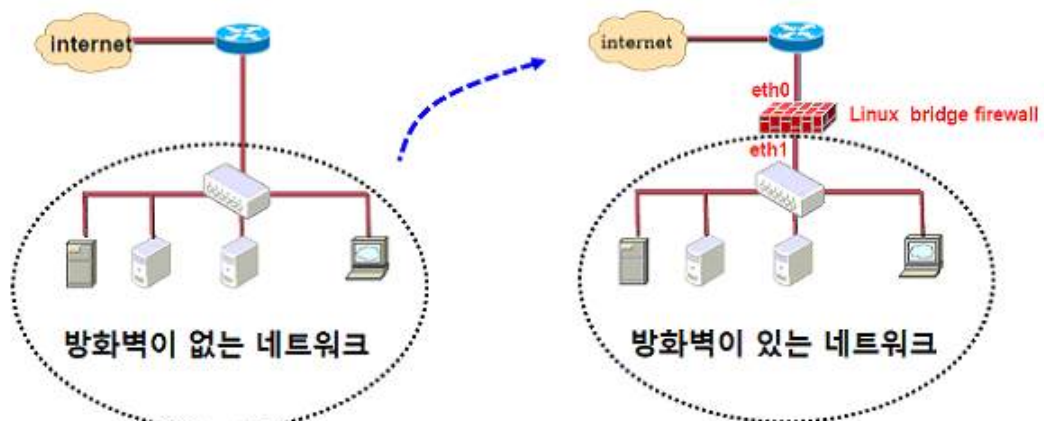
Bridge Firewall 은 이런 개념으로 컴퓨터에 랜카드를 2개를 꼽아서 랜카드 1로 들어와서 랜카드 2로 나가는 방식을 지원한다. 리눅스에서는 커널 레벨에서 Bridge를 지원한다.

허브가 ip 가 없듯이(스위치 허브 중에 특정 허브는 ip를 가지는 경우도 있음) Bridge Firewall 역시 ip가 없다. (단, 관리 목적으로 IP를 가지기도 한다.)

Bridge는 투명하다. traceroute로 해도 나타나지 않는다.

마치 허브처럼 연결을 해주며 이러한 중간에서 패킷을 제어할 수 있는 것이 Bridge Firewall인 것이다.

• 리눅스를 이용한 브릿지 방화벽



• bridge firewall 구성

인터넷 - (eth0) 리눅스방화벽 (eth1) - 스위치 -서버들

준비물

랜카드 2장

허브

크로스케이블

bridge 커널 패치

bridge utils

iptables 커널 패치

iptables 소스

iptables 커널 패치 및 iptables

netfilter/iptables <http://www.netfilter.org>

patch-o-matic-20030107.tar.bz2

iptables-1.2.8.tar.bz2

서비스를 위한 기본 설정

처음 설치시 방화벽을 설정하면 /etc/sysconfig/iptables 파일이 생성된다.

디폴트 iptables 파일을 삭제한 후 아래의 포트를 추가하도록 하자

// 기존 iptables 파일 제거

```
# rm -rf /etc/sysconfig/iptables
```

rm: remove 일반 파일 `/etc/sysconfig/iptables'? y

// iptables 정책 추가

```
# iptables -A INPUT -p tcp --dport 20 -j ACCEPT // ftp-data
```

```
# iptables -A INPUT -p tcp --dport 21 -j ACCEPT // ftp
```

```
# iptables -A INPUT -p tcp --dport 22 -j ACCEPT // ssh
```

```
# iptables -A INPUT -p udp --dport 53 -j ACCEPT // named
```

```
# iptables -A INPUT -p tcp --dport 80 -j ACCEPT // http
```

```
# iptables -A INPUT -p tcp --dport 110 -j ACCEPT // pop3
```

```
# iptables -A INPUT -p tcp --dport 143 -j ACCEPT //imap
```

```
# iptables -A INPUT -p tcp --dport 3306 -j ACCEPT // mysql
```

```
# iptables -A INPUT -p icmp --icmp-type echo-request -j DROP // ping에 대한  
응답 거부
```

```
# iptables -A INPUT -p tcp --dport 1:65335 -j DROP // 서비스포트 모두 거부
```

구축 과정

```
root@localhost:/etc/yum.repos.d
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
[root@localhost yum.repos.d]# yum -y install bridge-utils
Loaded plugins: presto, refresh-packagekit
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package bridge-utils.x86_64 0:1.2-8.fc12 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                Arch          Version           Repository        Size
=====
Installing:
bridge-utils           x86_64        1.2-8.fc12        fedora            27 k
=====

Transaction Summary
=====
Install      1 Package(s)
Upgrade     0 Package(s)
=====
```

브릿지 설정 유틸리티를 설치 해준다.

```
root@localhost:/etc/yum.repos.d
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
[root@localhost yum.repos.d]# service NetworkManager stop
네트워크 관리자 데몬 종료 중: [ OK ]
[root@localhost yum.repos.d]# chkconfig NetworkManager off
[root@localhost yum.repos.d]#
```

네트워크 관리자를 stop off해준다.

```
vi /etc/sysconfig/network-script/ifcfg-br0
```

```
DEVICE=br0
TYPE=Bridge
ONBOOT=yes
IPADDR=192.168.1.44(임의 값)
GATEWAY=192.168.0.254
NETMASK=255.255.255.0
DNS1=dns값
DNS2=dns값
```

vi /etc/sysconfig/network-script/ifcfg-eth0
<pre> DEVICE=eth0 BOOTPROTO=static ONBOOT=yes BRIDGE=br0 TYPE=Ethernet </pre>
vi /etc/sysconfig/network-script/ifcfg-eth1
<pre> DEVICE=eth1 BOOTPROTO=static ONBOOT=yes BRIDGE=br0 TYPE=Ethernet </pre>

각각 /etc/sysconfig/network-script/의 eth0, eth1 br0을 설정해준다.

```

root@localhost:/etc/sysconfig/network-scripts
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
[root@localhost network-scripts]# vi /etc/sysconfig/network-scripts/ifcfg-eth1
[root@localhost network-scripts]# service network restart
인터페이스 eth0 (을)를 종료 중: bridge br0 does not exist!
loopback 인터페이스 종료 중:
loopback 인터페이스 활성화중 입니다:
eth0 인터페이스 활성화중 입니다:
eth1 인터페이스 활성화중 입니다: 장치 eth1가 없는것 같습니다. 초기화를 지연합니다.
br0 인터페이스 활성화중 입니다:
[root@localhost network-scripts]#

```

설정을 완료해 준 뒤에 네트워크를 재시작 해준다.

그림에서는 가상화로 구현을 해서 eth1번의 장치가 없다.

가상화가 아닌 실제 랜카드를 2개를 장착한 뒤 설정하면 장치가 모두 가동한다.

```

root@localhost:/etc/sysconfig
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
# Generated by iptables-save v1.4.5 on Mon Jun 30 09:45:35 2008
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
#특정 아이피 허용 및 차단
-A FORWARD -s 10.100.114.93 -m state --state NEW -j ACCEPT
-A FORWARD -s 10.100.114.129 -m state --state NEW -j ACCEPT
-A FORWARD -s 10.100.114.211 -m state --state NEW -j DROP
#-A FORWARD -s 10.200.100.249 -m state --state NEW -j DROP
-A FORWARD -s 10.200.100.0/24 -m state --state NEW -j DROP
#방화벽 내부로 향하는 라우팅 될 수 없는 사실 ip를 차단
-A FORWARD -s 0.0.0.0/8 -j DROP
-A FORWARD -s 168.254.0.0/16 -j DROP
-A FORWARD -s 172.16.0.0/12 -j DROP
-A FORWARD -s 192.0.2.0/24 -j DROP
-A FORWARD -s 192.168.0.0/16 -j DROP
-A FORWARD -s 224.0.0.0/4 -j DROP
-A FORWARD -s 240.0.0.0/5 -j DROP
-A FORWARD -s 248.0.0.0/5 -j DROP
#이미 세션을 맺어 상태주적 테이블을 목록에 있는 ESTABLISHED RELATED 패킷을 허용
방화벽 성능을 높임
-A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

```

방화벽 규칙을 /etc/sysconfig/iptables에 설정을 해준다.

3.2 SSO 구축

Kerberos 환경구축

-SSO 서버

```
yum install krb5-server
```

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
Transaction test succeeded  
Running transaction  
  Updating      : krb5-libs-1.13.2-12.el7_2.x86_64                1/5  
  Updating      : krb5-workstation-1.13.2-12.el7_2.x86_64        2/5  
  Installing    : krb5-server-1.13.2-12.el7_2.x86_64            3/5  
  Cleanup       : krb5-workstation-1.13.2-10.el7.x86_64          4/5  
  Cleanup       : krb5-libs-1.13.2-10.el7.x86_64                5/5  
  Verifying     : krb5-libs-1.13.2-12.el7_2.x86_64              1/5  
  Verifying     : krb5-workstation-1.13.2-12.el7_2.x86_64        2/5  
  Verifying     : krb5-server-1.13.2-12.el7_2.x86_64            3/5  
  Verifying     : krb5-workstation-1.13.2-10.el7.x86_64          4/5  
  Verifying     : krb5-libs-1.13.2-10.el7.x86_64                5/5  
  
Installed:  
  krb5-server.x86_64 0:1.13.2-12.el7_2  
  
Dependency Updated:  
  krb5-libs.x86_64 0:1.13.2-12.el7_2  krb5-workstation.x86_64 0:1.13.2-12.el7_2  
  
Complete!
```

```
vi /etc/krb5.conf
```

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[logging]  
default = FILE:/var/log/krb5libs.log  
kdc = FILE:/var/log/krb5kdc.log  
admin_server = FILE:/var/log/kadmind.log  
  
[libdefaults]  
dns_lookup_realm = false  
ticket_lifetime = 24h  
renew_lifetime = 7d  
forwardable = true  
rdns = false  
default_realm = JOONGBU.KVM  
default_ccache_name = KEYRING:persistent:%uid  
  
[realms]  
JOONGBU.KVM = {  
  kdc = sso.joongbu.kvm  
  admin_server = sso.joongbu.kvm  
}  
  
[domain_realm]  
.joongbu.kvm = JOONGBU.KVM  
joongbu.kvm = JOONGBU.KVM
```

주석을 모두 제거한 후 example.com을 자신이 사용하고자 하는 도메인으로 수정한다.

```
vi /etc/var/kerberos/krb5kdc/kdc.conf 에서 example.com 변경
```

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[kdcdefaults]  
kdc_ports = 88  
kdc_tcp_ports = 88  
  
[realms]  
JOONGBU.KVM = {  
  #master_key_type = aes256-cts  
  acl_file = /var/kerberos/krb5kdc/kadm5.acl  
  dict_file = /usr/share/dict/words  
  admin_keytab = /var/kerberos/krb5kdc/kadm5.keytab  
  supported_encypes = aes256-cts:normal aes128-cts:normal des3-hmac-sha1:normal arcfour-hmac:normal camellia256-cts:normal camellia128-cts:normal des-hmac-sha1:normal des-cbc-md5:normal des-cbc-crc:normal
```


vi /var/kerberos/krb5kdc/kadm5.acl 에서 example.com을 변경

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
*/admin@JOONGBU.KVM *
```

kdb5_util create -s -r JOONGBU.KVM

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@sso ~]# kdb5_util create -s -r JOONGBU.KVM  
Loading random data  
Initializing database '/var/kerberos/krb5kdc/principal' for realm 'JOONGBU.KVM',  
master key name 'K/M@JOONGBU.KVM'  
You will be prompted for the database Master Password.  
It is important that you NOT FORGET this password.  
Enter KDC database master key:  
Re-enter KDC database master key to verify:
```

사용하려는 JOONGBU.KVM 을 커버로스에 등록

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@sso ~]# systemctl enable krb5kdc  
Created symlink from /etc/systemd/system/multi-user.target.wants/krb5kdc.service to /usr/lib/systemd/system/krb5kdc.service.  
[root@sso ~]# systemctl enable kadmind  
Created symlink from /etc/systemd/system/multi-user.target.wants/kadmind.service to /usr/lib/systemd/system/kadmind.service.  
[root@sso ~]# systemctl start kadmind  
[root@sso ~]# systemctl start krb5kdc
```

부팅 후 자동으로 실행할 수 있도록 등록해준 뒤 데몬실행

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@sso ~]# systemctl status kadmind  
● kadmind.service - Kerberos 5 Password-changing and Administration  
   Loaded: loaded (/usr/lib/systemd/system/kadmind.service; enabled; vendor preset: disabled)  
   Active: active (running) since 금 2016-05-13 17:43:36 KST; 43s ago  
     Process: 41716 ExecStart=/usr/sbin/_kadmind -P /var/run/kadmind.pid $KADMIND_ARGS (code=exited, status=0/SUCCESS)  
    Main PID: 41727 (kadmind)  
      CGroup: /system.slice/kadmind.service  
              └─41727 /usr/sbin/kadmind -P /var/run/kadmind.pid  
  
5월 13 17:43:36 sso systemd[1]: Starting Kerberos 5 Password-changing and Administration..  
5월 13 17:43:36 sso systemd[1]: PID file /var/run/kadmind.pid not readable (yet?) after start.  
5월 13 17:43:36 sso systemd[1]: Started Kerberos 5 Password-changing and Administration.  
[root@sso ~]# systemctl status krb5kdc  
● krb5kdc.service - Kerberos 5 KDC  
   Loaded: loaded (/usr/lib/systemd/system/krb5kdc.service; enabled; vendor preset: disabled)  
   Active: active (running) since 금 2016-05-13 17:43:45 KST; 1min 12s ago  
     Process: 41858 ExecStart=/usr/sbin/krb5kdc -P /var/run/krb5kdc.pid $KRB5KDC_ARGS (code=exited, status=0/SUCCESS)  
    Main PID: 41862 (krb5kdc)  
      CGroup: /system.slice/krb5kdc.service  
              └─41862 /usr/sbin/krb5kdc -P /var/run/krb5kdc.pid  
  
5월 13 17:43:45 sso systemd[1]: Starting Kerberos 5 KDC..  
5월 13 17:43:45 sso systemd[1]: PID file /var/run/krb5kdc.pid not readable (yet?) after start.  
5월 13 17:43:45 sso systemd[1]: Started Kerberos 5 KDC.
```

systemctl status kadmind/krb5kdc 로 서비스가 실행되는지 확인

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@sso ~]# firewall-cmd --permanent --add-service=kerberos  
success  
[root@sso ~]# firewall-cmd --reload  
success
```

방화벽에서 커버로스를 허용하도록 등록 후 재실행

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@sso ~]# kadmin.local  
Authenticating as principal root/admin@OONGBU.KVM with password.  
kadmin.local: addprinc -randkey host/client1.joongbu.kvm  
WARNING: no policy specified for host/client1.joongbu.kvm@OONGBU.KVM; defaulting to no policy  
Principal "host/client1.joongbu.kvm@OONGBU.KVM" created.  
kadmin.local: addprinc -randkey host/client2.joongbu.kvm  
WARNING: no policy specified for host/client2.joongbu.kvm@OONGBU.KVM; defaulting to no policy  
Principal "host/client2.joongbu.kvm@OONGBU.KVM" created.  
kadmin.local: █
```

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@sso ~]# kadmin.local  
Authenticating as principal root/admin@OONGBU.KVM with password.  
kadmin.local: ktadd -k /tmp/client1.keytab host/client1.joongbu.kvm  
Entry for principal host/client1.joongbu.kvm with kvno 2, encryption type aes256-cts-hmac-sha1-96 added to keytab WRFILE: /tmp/client1.keytab.  
Entry for principal host/client1.joongbu.kvm with kvno 2, encryption type aes128-cts-hmac-sha1-96 added to keytab WRFILE: /tmp/client1.keytab.  
Entry for principal host/client1.joongbu.kvm with kvno 2, encryption type des3-cbc-sha1 added to keytab WRFILE: /tmp/client1.keytab.  
Entry for principal host/client1.joongbu.kvm with kvno 2, encryption type arcfour-hmac added to keytab WRFILE: /tmp/client1.keytab.  
Entry for principal host/client1.joongbu.kvm with kvno 2, encryption type camellia256-cts-cmac added to keytab WRFILE: /tmp/client1.keytab.  
Entry for principal host/client1.joongbu.kvm with kvno 2, encryption type camellia128-cts-cmac added to keytab WRFILE: /tmp/client1.keytab.  
Entry for principal host/client1.joongbu.kvm with kvno 2, encryption type des-hmac-sha1 added to keytab WRFILE: /tmp/client1.keytab.  
Entry for principal host/client1.joongbu.kvm with kvno 2, encryption type des-cbc-md5 added to keytab WRFILE: /tmp/client1.keytab.  
kadmin.local: ktadd -k /tmp/client2.keytab host/client2.joongbu.kvm  
Entry for principal host/client2.joongbu.kvm with kvno 2, encryption type aes256-cts-hmac-sha1-96 added to keytab WRFILE: /tmp/client2.keytab.  
Entry for principal host/client2.joongbu.kvm with kvno 2, encryption type aes128-cts-hmac-sha1-96 added to keytab WRFILE: /tmp/client2.keytab.  
Entry for principal host/client2.joongbu.kvm with kvno 2, encryption type des3-cbc-sha1 added to keytab WRFILE: /tmp/client2.keytab.  
Entry for principal host/client2.joongbu.kvm with kvno 2, encryption type arcfour-hmac added to keytab WRFILE: /tmp/client2.keytab.  
Entry for principal host/client2.joongbu.kvm with kvno 2, encryption type camellia256-cts-cmac added to keytab WRFILE: /tmp/client2.keytab.  
Entry for principal host/client2.joongbu.kvm with kvno 2, encryption type camellia128-cts-cmac added to keytab WRFILE: /tmp/client2.keytab.  
Entry for principal host/client2.joongbu.kvm with kvno 2, encryption type des-hmac-sha1 added to keytab WRFILE: /tmp/client2.keytab.  
Entry for principal host/client2.joongbu.kvm with kvno 2, encryption type des-cbc-md5 added to keytab WRFILE: /tmp/client2.keytab.  
kadmin.local: █
```

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@sso ~]# kadmin.local  
Authenticating as principal root/admin@OONGBU.KVM with password.  
kadmin.local: listprincs  
K/M@OONGBU.KVM  
host/client1.joongbu.kvm@OONGBU.KVM  
host/client2.joongbu.kvm@OONGBU.KVM  
kadmin/admin@OONGBU.KVM  
kadmin/changepw@OONGBU.KVM  
kadmin/sso@OONGBU.KVM  
kiprop/sso@OONGBU.KVM  
krbtgt/OONGBU.KVM@OONGBU.KVM  
kadmin.local: █
```

SSO 서버에 SSO 접속을 허용하려는 사용자 (client1,client2) 등록 후 추가목록을 확인


```

root@sso:tmp
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
[root@sso tmp]# ls
anaconda.log          systemd-private-9c52310d5da340ee8e9bf8e7c6dc4893-colord.service-DvKBK5
client1.keytab         systemd-private-9c52310d5da340ee8e9bf8e7c6dc4893-cups.service-mHiyN4
client2.keytab         systemd-private-9c52310d5da340ee8e9bf8e7c6dc4893-rtkit-daemon.service-413H3n
hogsuspend            systemd-private-9c52310d5da340ee8e9bf8e7c6dc4893-systemd-hostnamed.service-Kj5AD4
hsperfdata_root       systemd-private-9c52310d5da340ee8e9bf8e7c6dc4893-systemd-located.service-1l7MBc
hsperfdata_sso        systemd-private-9c52310d5da340ee8e9bf8e7c6dc4893-vmtoolsd.service-wRrASJ
ifcfg.log             systemd-private-b425d4deee8d49a0b5c62d4c82a79e62-colord.service-sdZgoE
ks-script.vqnWYn     systemd-private-b425d4deee8d49a0b5c62d4c82a79e62-cups.service-gVLYmD
packaging.log         systemd-private-b425d4deee8d49a0b5c62d4c82a79e62-rtkit-daemon.service-APYS1x
program.log          systemd-private-b425d4deee8d49a0b5c62d4c82a79e62-vmtoolsd.service-cikW7c
sensitive-info.log   tracker-extract-files.1000
ssh-ZK6bIdCaAIdL     yum.log
ssh-wcxLvAa6jdRY     yum_save_tx.2016-05-13.16-22.U7ugeY.yumtx
storage.log          yum_save_tx.2016-05-13.16-58.6Gl6R8.yumtx

root@sso:/usr/bin
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
[root@sso bin]# scp /etc/krb5.conf /tmp/client2.keytab client2:/tmp/
The authenticity of host 'client2 (192.168.25.156)' can't be established.
ECDSA key fingerprint is 38:60:18:02:09:4f:17:9d:bf:a1:86:35:30:42:6d:a4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'client2,192.168.25.156' (ECDSA) to the list of known hosts.
root@client2's password:
krb5.conf                                100% 479      0.5KB/s   00:00
client2.keytab                          100% 626      0.6KB/s   00:00

```

SSO 서버에 client1.keytab , client2.keytab 파일이 생성된 것을 확인할 수 있다.
 scp /etc/krb5.conf /tmp/client2.keytab client2:/tmp/ 명령으로 client2로 전송

-client2

```

root@client2:/tmp
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
[root@client2 tmp]# ls
anaconda.log
client2.keytab
hogsuspend
hsperfdata_client2
hsperfdata_root
ifcfg.log
krb5.conf

```

krb5.conf 파일과 keytab 파일이 client2로 잘 전송된 것을 확인할 수 있다.

yum -y install pam_krb5 krb5-workstation을 설치

```

root@client2:~
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
Running transaction
  Updating      : krb5-libs-1.13.2-12.el7_2.x86_64                      1/4
  Updating      : krb5-workstation-1.13.2-12.el7_2.x86_64              2/4
  Cleanup       : krb5-workstation-1.13.2-10.el7.x86_64                3/4
  Cleanup       : krb5-libs-1.13.2-10.el7.x86_64                       4/4
  Verifying     : krb5-libs-1.13.2-12.el7_2.x86_64                    1/4
  Verifying     : krb5-workstation-1.13.2-12.el7_2.x86_64              2/4
  Verifying     : krb5-workstation-1.13.2-10.el7.x86_64                3/4
  Verifying     : krb5-libs-1.13.2-10.el7.x86_64                       4/4

Updated:
  krb5-workstation.x86_64 0:1.13.2-12.el7_2

Dependency Updated:
  krb5-libs.x86_64 0:1.13.2-12.el7_2

Complete!
[root@client2 ~]#

```

SSO 서버에서 client2로 전송한 krb5.conf 파일을 /etc/ 경로로 전송한다.

```
Wcp /tmp/krb5.conf /etc/
```

```
root@client2:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@client2 ~]# ktutil  
ktutil: rkt /tmp/client2.keytab  
ktutil: wkt /etc/krb5.keytab  
ktutil: list  
slot KVN0 Principal  
-----  
1 2 host/client2.joongbu.kvm@000NGBU.KVM  
2 2 host/client2.joongbu.kvm@000NGBU.KVM  
3 2 host/client2.joongbu.kvm@000NGBU.KVM  
4 2 host/client2.joongbu.kvm@000NGBU.KVM  
5 2 host/client2.joongbu.kvm@000NGBU.KVM  
6 2 host/client2.joongbu.kvm@000NGBU.KVM  
7 2 host/client2.joongbu.kvm@000NGBU.KVM  
8 2 host/client2.joongbu.kvm@000NGBU.KVM  
ktutil: █
```

client2에서 ktutil 명령으로 keytab파일을 이용해서 사용자를 확인할 수 있다.

-LDAP 환경구축

SSO 서버

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
Verifying : openldap-clients-2.4.40-9.el7_2.x86_64 1/5  
Verifying : openldap-2.4.40-9.el7_2.x86_64 2/5  
Verifying : openldap-servers-2.4.40-9.el7_2.x86_64 3/5  
Verifying : migrationtools-47-15.el7.noarch 4/5  
Verifying : openldap-2.4.40-8.el7.x86_64 5/5  
  
Installed:  
migrationtools.noarch 0:47-15.el7 openldap-clients.x86_64 0:2.4.40-9.el7_2  
openldap-servers.x86_64 0:2.4.40-9.el7_2  
  
Dependency Updated:  
openldap.x86_64 0:2.4.40-9.el7_2  
  
Complete!  
[root@sso ~]# █
```

yum -y install openldap-servers openldap openldap-clients migrationtools 설치

```

root@sso:/var/lib/ldap
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
[root@sso ldap]# pwd
/var/lib/ldap
[root@sso ldap]# ls
DB_CONFIG
[root@sso ldap]# chown -R ldap. /var/lib/ldap/
[root@sso ldap]# ls -l
합계 4
-rw-r--r--. 1 ldap ldap 845 5월 13 22:15 DB_CONFIG
[root@sso ldap]#

```

cp /usr/share/openldap-servers/DB_CONFIG.example /var/lib/ldap/DB_CONFIG
명령으로 파일경로를 이동시키고 디렉토리 권한을 변경

```

root@sso:/etc/openldap/slapd.d/cn=config
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
[root@sso cn=config]# pwd
/etc/openldap/slapd.d/cn=config
[root@sso cn=config]# slappasswd
New password:
Re-enter new password:
{SSHA}HLBi02gNActybggZ827MqTMKPDX6qx55
[root@sso cn=config]#

```

slappasswd 명령으로 생성된 암호를 복사

```

root@sso:/etc/openldap/slapd.d/cn=config
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
[root@sso cn=config]# ls
cn=schema          olcDatabase={1} frontend.ldif  olcDatabase={1} monitor.ldif
cn=schema.ldif     olcDatabase={0} config.ldif    olcDatabase={2} hdb.ldif
[root@sso cn=config]# vi olcDatabase={0} config.ldif

```

```

root@sso:/etc/openldap/slapd.d/cn=config
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
# AUTO-GENERATED FILE - DO NOT EDIT!! Use ldapmodify.
# CRC32 f1984fd3
dn: olcDatabase={0} config
objectClass: olcDatabaseConfig
olcDatabase: {0} config
olcAccess: {0} to * by dn,base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth" manage by * none
structuralObjectClass: olcDatabaseConfig
entryUUID: bea2681a-ad57-1035-8e30-8d4412dd00ae
creatorsName: cn=config
createTimestamp: 20160513131001Z
entryCSN: 20160513131001.273925Z#000000#000#000000
modifiersName: cn=config
modifyTimestamp: 20160513131001Z
olcRootPW: {SSHA}HLBi02gNActybggZ827MqTMKPDX6qx55
~

```

vi /etc/openldap/slapd.d/cn=config/olcDatabase={0}config.ldif 경로에서
복사 해놓은 암호를 저장

```
vi /etc/openldap/slapd.d/cnW=config/olcDatabaseW=W{2W}hdb.ldif
```

```
root@sso:/etc/openldap/slapd.d/cn=config

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
# AUTO-GENERATED FILE - DO NOT EDIT!! Use ldapmodify.
# CRC32 961da5b4
dn: olcDatabase={2} hdb
objectClass: olcDatabaseConfig
objectClass: olcHdbConfig
olcDatabase: {2} hdb
olcDbDirectory: /var/lib/ldap
olcSuffix: dc=joongbu,dc=kvm
olcRootDN: cn=Manager,dc=joongbu,dc=kvm
olcRootPW: {SSHA}HLBi02gNActybggZ827MqTMKPDx6qx55
olcDbIndex: objectClass eq,pres
olcDbIndex: ou,cn,mail,surname,givenname eq,pres,sub
structuralObjectClass: olcHdbConfig
entryUUID: bea284d0-ad57-1035-8e32-8d4412dd00ae
creatorsName: cn=config
createTimestamp: 20160513131001Z
entryCSN: 20160513131001.274660Z#000000#000#000000
modifiersName: cn=config
modifyTimestamp: 20160513131001Z
olcAccess: {0}to attrs=userPassword by self write by dn.base="cn=Manager,dc=joongbu,dc=kvm" write by anonymous auth by * none
olcAccess: {1}to * by dn.base="cn=Manager,dc=joongbu,dc=kvm" write by self write by * read
```

olcAccess: {0}to attrs=userPassword by self write by dn.base="cn=Manager,dc=joongbu,dc=kvm" write by anonymous auth by * none
olcAccess: {1}to * by dn.base="cn=Manager,dc=joongbu,dc=kvm" write by self write by * read
위와 같이 olcAccess를 추가하여 접근이 가능하도록 설정한 후,
dc값을 변경해주고 암호를 추가하고 저장

```
vi /etc/openldap/slapd.d/cnW=config/olcDatabaseW=W{1W}monitor.ldif
```

```
root@sso:/etc/openldap/slapd.d/cn=config

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
# AUTO-GENERATED FILE - DO NOT EDIT!! Use ldapmodify.
# CRC32 945f12e9
dn: olcDatabase={1} monitor
objectClass: olcDatabaseConfig
olcDatabase: {1} monitor
olcAccess: {0}to * by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth" read by dn.base="cn=Manager,dc=joongbu,dc=kvm" read by * none
structuralObjectClass: olcDatabaseConfig
entryUUID: bea27dfa-ad57-1035-8e31-8d4412dd00ae
creatorsName: cn=config
createTimestamp: 20160513131001Z
entryCSN: 20160513131001.274484Z#000000#000#000000
modifiersName: cn=config
modifyTimestamp: 20160513131001Z
```

dc값 변경 후 저장


```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@sso ~]# systemctl enable slapd  
Created symlink from /etc/systemd/system/multi-user.target.wants/slapd.service to /usr/lib/systemd/system/slapd.service.  
[root@sso ~]# systemctl start slapd  
[root@sso ~]#
```

systemctl enable slapd 으로 데몬을 부팅 후 자동으로 시작하도록 설정 후 데몬 실행

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@sso ~]# netstat -nltp  
Active Internet connections (only servers)  
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name  
tcp        0      0 0.0.0.0:749          0.0.0.0:*                 LISTEN      1785/kadmind  
tcp        0      0 0.0.0.0:464          0.0.0.0:*                 LISTEN      1785/kadmind  
tcp        0      0 192.168.122.1:53     0.0.0.0:*                 LISTEN      2065/dnsmasq  
tcp        0      0 0.0.0.0:22           0.0.0.0:*                 LISTEN      4409/sshd  
tcp        0      0 127.0.0.1:631        0.0.0.0:*                 LISTEN      1567/cupsd  
tcp        0      0 0.0.0.0:88           0.0.0.0:*                 LISTEN      1580/krb5kdc  
tcp        0      0 127.0.0.1:25         0.0.0.0:*                 LISTEN      2024/master  
tcp        0      0 0.0.0.0:389          0.0.0.0:*                 LISTEN      33080/slapd  
tcp6       0      0 :::749               :::*                   LISTEN      1785/kadmind  
tcp6       0      0 :::464               :::*                   LISTEN      1785/kadmind  
tcp6       0      0 :::22                :::*                   LISTEN      4409/sshd  
tcp6       0      0 :::1:631             :::*                   LISTEN      1567/cupsd  
tcp6       0      0 :::88                :::*                   LISTEN      1580/krb5kdc  
tcp6       0      0 :::1:25              :::*                   LISTEN      2024/master  
tcp6       0      0 :::389               :::*                   LISTEN      33080/slapd
```

netstat -nltp 명령으로 데몬의 현재 상태를 확인

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@sso ~]# firewall-cmd --permanent --add-service=ldap  
success  
[root@sso ~]# firewall-cmd --reload  
success  
[root@sso ~]#
```

방화벽 허용을 설정해주고 재시작

```
Idapadd 명령을 실행하여 다음 세 개의 설정파일을 사용할 수 있도록 등록  
Idapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/cosine.ldif  
Idapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/nis.ldif  
Idapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/inetorgperson.ldif
```

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
dn: dc=joongbu,dc=kvm  
objectClass: dcObject  
objectClass: organization  
dc: joongbu  
o: joongbu  
  
dn: ou=People,dc=joongbu,dc=kvm  
objectClass: organizationalUnit  
ou: People  
  
dn: ou=Group,dc=joongbu,dc=kvm  
objectClass: organizationalUnit
```

root의 홈 디렉토리에서 base.ldif 파일생성

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@sso ~]# ldapadd -x -D cn=Manager,dc=joongbu,dc=kvm -W -f base.ldif  
Enter LDAP Password:  
adding new entry "dc=joongbu,dc=kvm"  
  
adding new entry "ou=People,dc=joongbu,dc=kvm"  
  
adding new entry "ou=Group,dc=joongbu,dc=kvm"  
  
[root@sso ~]# ldapsearch -x -D cn=Manager,dc=joongbu,dc=kvm -W -b dc=joongbu,dc=kvm  
Enter LDAP Password:  
# extended LDIF  
#  
# LDAPv3  
# base <dc=joongbu,dc=kvm> with scope subtree  
# filter: (objectclass=*)  
# requesting: ALL  
#  
# joongbu, kvm  
dn: dc=joongbu,dc=kvm  
objectClass: dcObject  
objectClass: organization  
dc: joongbu  
o: joongbu  
  
# People, joongbu, kvm  
dn: ou=People,dc=joongbu,dc=kvm  
objectClass: organizationalUnit  
ou: People  
  
# Group, joongbu, kvm  
dn: ou=Group,dc=joongbu,dc=kvm  
objectClass: organizationalUnit  
ou: Group  
  
# search result  
search: 2  
result: 0 Success  
  
# numResponses: 4  
# numEntries: 3  
[root@sso ~]#
```

ldapadd 명령을 이용해서 SSO 서버의 base.ldif 파일을 등록하고
ldapsearch 명령을 통해서 위와 같은 등록정보를 확인할 수 있다.

useradd 명령으로 joongbu1~joongbu5 사용자 추가

/usr/share/migrationtools 경로에서 migrate_common.ph 파일실행 후

```
root@sso:/usr/share/migrationtools  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@sso migrationtools]# pwd  
/usr/share/migrationtools  
[root@sso migrationtools]# vi migrate_common.ph
```

```
root@sso:/usr/share/migrationtools

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

# Default DNS domain
$DEFAULT_MAIL_DOMAIN = "joongbu.kvm";

# Default base
$DEFAULT_BASE = "dc=joongbu,dc=kvm";

# Turn this on for inetLocalMailRecipient
# sendmail support; add the following to
# sendmail.mc (thanks to Petr@Kristof.CZ):
##### CUT HERE #####
#define('confLDAP_DEFAULT_SPEC', '-h "ldap.padl.com"') dn1
#LDAPROUTE_DOMAIN_FILE(' /etc/mail/ldapdomains' ) dn1
#FEATURE(ldap_routing) dn1
##### CUT HERE #####
# where /etc/mail/ldapdomains contains names of ldap_routed
# domains (similar to MASQUERADE_DOMAIN_FILE).
# $DEFAULT_MAIL_HOST = "mail.padl.com";

# turn this on to support more general object classes
# such as person.
$EXTENDED_SCHEMA = 1
```

DEFAULT 값 변경하고 dc값 변경 후 SCHEMA 값 1로 변경

```
grep joongbu /etc/passwd > tmp/users
```

```
grep joongbu /etc/group > /tmp/groups
```

위 명령으로 joongbu1~5 사용자 정보를 /tmp/groups , /tmp/users에 추가

```
root@sso:/usr/share/migrationtools

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

[root@sso migrationtools]# ls
migrate_aliases.pl      migrate_all_offline.sh  migrate_hosts.pl       migrate_protocols.pl
migrate_all_netinfo_offline.sh migrate_all_online.sh  migrate_netgroup.pl    migrate_rpc.pl
migrate_all_netinfo_online.sh migrate_automount.pl   migrate_netgroup_byhost.pl migrate_services.pl
migrate_all_nis_offline.sh migrate_base.pl        migrate_netgroup_byuser.pl migrate_slapd_conf.pl
migrate_all_nis_online.sh migrate_common.ph      migrate_networks.pl
migrate_all_nisplus_offline.sh migrate_fstab.pl       migrate_passwd.pl
migrate_all_nisplus_online.sh migrate_group.pl       migrate_profile.pl
[root@sso migrationtools]# ./migrate_passwd.pl /tmp/users /tmp/users.ldif
[root@sso migrationtools]# ./migrate_group.pl /tmp/groups /tmp/groups.ldif
```

```
./migrate_passwd.pl /tmp/users /tmp/users.ldif
```

```
./migrate_group.pl /tmp/groups /tmp/groups.ldif
```

위 명령으로 users.ldif, groups.ldif 파일을 생성

```
root@sso:/usr/share/migrationtools

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

[root@sso migrationtools]# ldapadd -x -D cn=Manager,dc=joongbu,dc=kvm -W -f /tmp/groups.ldif
Enter LDAP Password:
adding new entry "cn=joongbu1,ou=Group,dc=joongbu,dc=kvm"

adding new entry "cn=joongbu2,ou=Group,dc=joongbu,dc=kvm"

adding new entry "cn=joongbu3,ou=Group,dc=joongbu,dc=kvm"

adding new entry "cn=joongbu4,ou=Group,dc=joongbu,dc=kvm"

adding new entry "cn=joongbu5,ou=Group,dc=joongbu,dc=kvm"

[root@sso migrationtools]# ldapadd -x -D cn=Manager,dc=joongbu,dc=kvm -W -f /tmp/users.ldif
Enter LDAP Password:
adding new entry "uid=joongbu1,ou=People,dc=joongbu,dc=kvm"

adding new entry "uid=joongbu2,ou=People,dc=joongbu,dc=kvm"

adding new entry "uid=joongbu3,ou=People,dc=joongbu,dc=kvm"

adding new entry "uid=joongbu4,ou=People,dc=joongbu,dc=kvm"

adding new entry "uid=joongbu5,ou=People,dc=joongbu,dc=kvm"
```

ldapadd 명령으로 생성된 users.ldif , groups.ldif 파일들을 등록

```

root@sso:/usr/share/migrationtools

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
[root@sso migrationtools]# ldapsearch -x -D cn=Manager,dc=joongbu,dc=kvm -W -b dc=joongbu,dc=kvm
Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <dc=joongbu,dc=kvm> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# joongbu, kvm
dn: dc=joongbu,dc=kvm
objectClass: dcObject
objectClass: organization
dc: joongbu
o: joongbu
# People, joongbu, kvm
dn: ou=People,dc=joongbu,dc=kvm
objectClass: organizationalUnit
ou: People
# Group, joongbu, kvm
dn: ou=Group,dc=joongbu,dc=kvm
objectClass: organizationalUnit
ou: Group
# joongbu1, Group, joongbu, kvm
dn: cn=joongbu1,ou=Group,dc=joongbu,dc=kvm
objectClass: posixGroup
objectClass: top
cn: joongbu1
userPassword: : e2NyeX90fXg=
gidNumber: 1003

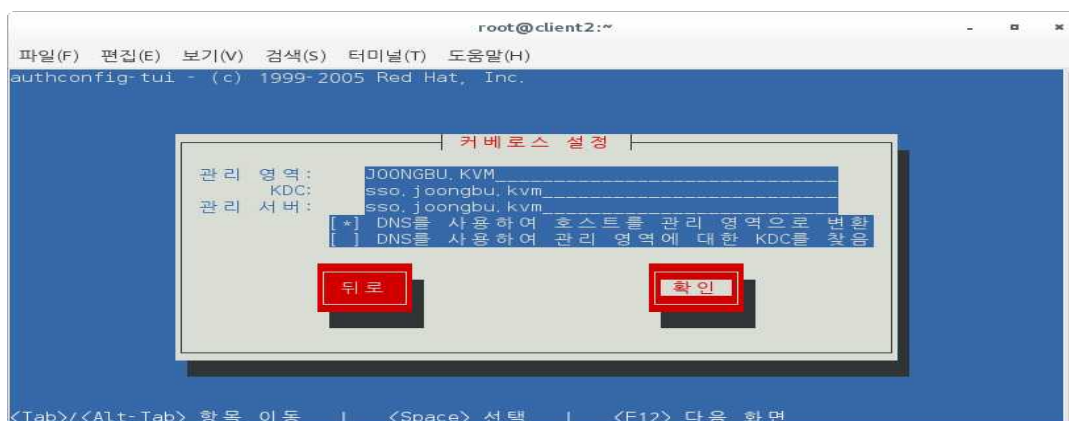
```

users.ldif, groups.ldif 정보가 등록된 것을 확인할 수 있다.

client2



yum -y install nss-pam-ldapd 설치 후 authconfig-tui 명령 실행



ldap, 커버로스 사용 체크 후 SSO서버 주소를 입력


```
grep /joongbu /etc/passwd
getent passwd joongbu1
명령으로 SSO서버의 joongbu1 사용자 엔트리확인
id joongbu1~5 (joongbu1~5 사용자 확인)
```

SSO 서버

```
yum -y install nfs-utils autofs
```

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@sso ~]# yum -y install nfs-utils autofs  
Loaded plugins: fastestmirror, langpacks  
Loading mirror speeds from cached hostfile  
* base: ftp.kaist.ac.kr  
* extras: ftp.kaist.ac.kr  
* updates: ftp.kaist.ac.kr  
Package 1:nfs-utils-1.3.0-0.21.el7_2.x86_64 already installed and latest version  
Package 1:autofs-5.0.7-54.el7.x86_64 already installed and latest version  
Nothing to do  
[root@sso ~]#
```

nfs-utils를 설치함으로써 다른 호스트에 있는 파일시스템의 일부를 자신의 디렉토리처럼 사용할 수 있도록 함. 즉, SSO서버를 여러 client가 사용할 수 있도록 한다. autofs는 자동마운트 데몬의 작동을 제어하는 프로그램이다. 자동마운트 데몬은 자동으로 파일 시스템을 마운트하는데 사용된다.

```
vi /etc/exports
```

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
/home *(rw, sync)  
|
```

/etc/exports는 SSO 서버의 공유목록을 관리하는 파일이며 편집해서 공유목록을 관리.

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@sso ~]# systemctl enable rpcbind  
[root@sso ~]# systemctl start rpcbind  
[root@sso ~]# systemctl enable nfs-server  
Created symlink from /etc/systemd/system/multi-user.target.wants/nfs-server.service to /usr/lib/systemd/system/nfs-server.service.  
[root@sso ~]# systemctl start nfs-server  
[root@sso ~]# firewall-cmd --permanent --add-service nfs  
success  
[root@sso ~]# firewall-cmd --reload  
success  
[root@sso ~]#
```

systemctl enable / start 명령으로 rpcbind, nfs-server를 등록해주고, 방화벽 허용 목록에 추가한 후 방화벽 재가동

```
vi /etc/auto.master
```

```
root@client2:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
+auto.master  
/home /etc/auto.autofs --timeout=600  
□
```

자동으로 마운트 될 디렉토리와 대상 설정파일을 설정하는 파일이다.

```
vi /etc/auto.autofs
```

```
root@client2:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
* sso: /home/ &  
□
```

SSO 서버의 디렉토리를 설정하는 설정파일

```
root@client2:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@client2 ~]# systemctl enable autofs  
Created symlink from /etc/systemd/system/multi-user.target.wants/autofs.service  
to /usr/lib/systemd/system/autofs.service.  
sy[root@client2 ~]# systemctl start autofs
```

systemctl enable / start 명령으로 autofs 를 실행

```
vi /etc/ssh/ssh_config
```

```
root@client2:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
# Host *  
# ForwardAgent no  
# ForwardX11 no  
# RhostsRSAAuthentication no  
# RSAAuthentication yes  
# PasswordAuthentication yes  
# HostbasedAuthentication no  
# GSSAPIAuthentication yes  
# GSSAPIDelegateCredentials yes  
# GSSAPIKeyExchange no  
# GSSAPITrustDNS no  
# BatchMode no  
# CheckHostIP yes  
# AddressFamily any  
# ConnectTimeout 0
```

GSSAPIAuthentication yes GSSAPIDelegateCredentials yes 로 수정

```
vi /etc/ssh/sshd_config
```

```
root@client2:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
# GSSAPI options  
GSSAPIAuthentication yes  
GSSAPICleanupCredentials yes  
#GSSAPIStrictAcceptorCheck yes  
#GSSAPIKeyExchange no
```

GSSAPIAuthentication yes GSSAPICleanupCredentials yes 로 수정

SSO 서버

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@sso ~]# kadmin.local  
Authenticating as principal root/admin@00ONGBU.KVM with password.  
kadmin.local: addprinc joongbu1  
WARNING: no policy specified for joongbu1@00ONGBU.KVM; defaulting to no policy  
Enter password for principal "joongbu1@00ONGBU.KVM":  
Re-enter password for principal "joongbu1@00ONGBU.KVM":  
Principal "joongbu1@00ONGBU.KVM" created.  
kadmin.local: addprinc joongbu2  
WARNING: no policy specified for joongbu2@00ONGBU.KVM; defaulting to no policy  
Enter password for principal "joongbu2@00ONGBU.KVM":  
Re-enter password for principal "joongbu2@00ONGBU.KVM":  
Principal "joongbu2@00ONGBU.KVM" created.  
kadmin.local: addprinc joongbu3  
WARNING: no policy specified for joongbu3@00ONGBU.KVM; defaulting to no policy  
Enter password for principal "joongbu3@00ONGBU.KVM":  
Re-enter password for principal "joongbu3@00ONGBU.KVM":  
Principal "joongbu3@00ONGBU.KVM" created.  
kadmin.local: addprinc joongbu4  
WARNING: no policy specified for joongbu4@00ONGBU.KVM; defaulting to no policy  
Enter password for principal "joongbu4@00ONGBU.KVM":  
Re-enter password for principal "joongbu4@00ONGBU.KVM":  
Principal "joongbu4@00ONGBU.KVM" created.  
kadmin.local: addprinc joongbu5  
WARNING: no policy specified for joongbu5@00ONGBU.KVM; defaulting to no policy  
Enter password for principal "joongbu5@00ONGBU.KVM":  
Re-enter password for principal "joongbu5@00ONGBU.KVM":  
Principal "joongbu5@00ONGBU.KVM" created.  
kadmin.local: 
```

kadmin.local 명령을 실행하고 addprinc 명령으로 joongbu1 ~ joongbu5 사용자 등록

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
kadmin.local: listprincs  
K/M@00ONGBU.KVM  
host/client1.joongbu.kvm@00ONGBU.KVM  
host/client2.joongbu.kvm@00ONGBU.KVM  
joongbu1@00ONGBU.KVM  
joongbu2@00ONGBU.KVM  
joongbu3@00ONGBU.KVM  
joongbu4@00ONGBU.KVM  
joongbu5@00ONGBU.KVM  
kadmin/admin@00ONGBU.KVM  
kadmin/changepw@00ONGBU.KVM  
kadmin/sso@00ONGBU.KVM  
kiprop/sso@00ONGBU.KVM  
krbtgt/JOONGBU.KVM@00ONGBU.KVM  
kadmin.local: 
```

listprincs 명령으로 사용자가 등록되었는지 확인할 수 있다.

```
root@client2:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@sso ~]# ssh client2  
root@client2's password:  
Last login: Sat May 14 20:37:35 2016  
[root@client2 ~]#
```

sso – client2가 정상적으로 접속한 것을 확인할 수 있다.

```
joongbu1@client2:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[client2@client2 ~]$ ssh joongbu1@client2  
Could not create directory '/home/client2/.ssh'.  
The authenticity of host 'client2 (192.168.25.156)' can't be established.  
ECDSA key fingerprint is 38:60:18:02:09:4f:17:9d:bf:a1:86:35:30:42:6d:a4.  
Are you sure you want to continue connecting (yes/no)? yes  
Failed to add the host to the list of known hosts (/home/client2/.ssh/known_hosts).  
joongbu1@client2's password:  
Last login: Sat May 14 20:25:18 2016 from client2  
[joongbu1@client2 ~]$ klist  
Ticket cache: KEYRING:persistent:1003:krb_ccache_tE1YmOI  
Default principal: joongbu1@00NGBU.KVM  
  
Valid starting Expires Service principal  
2016-05-14T20:50:37 2016-05-15T20:50:37 krbtgt/J00NGBU.KVM@00NGBU.KVM  
[joongbu1@client2 ~]$
```

client2에서 sso서버의 joongbu1 사용자로 접속한 것을 확인할 수 있다.

그리고 klist 명령으로 정상적인 티켓팅이 이루어졌는지도 확인할 수 있다.

/var/log/krb5kdc.log

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
5월 15 21:29:12 sso krb5kdc[5763](info): TGS_REQ (4 etypes {18 17 16 23}) 10.100.114.129: LOOKING_UP_SERVER: authtime 0, host/client1.joongbu.kvm@00NGBU.KVM for nfs/sso@00NGBU.KVM, Server not found in Kerberos database  
5월 15 21:29:12 sso krb5kdc[5763](info): TGS_REQ (6 etypes {18 17 16 23 25 26}) 10.100.114.129: LOOKING_UP_SERVER: authtime 0, host/client1.joongbu.kvm@00NGBU.KVM for nfs/sso@00NGBU.KVM, Server not found in Kerberos database  
5월 15 21:29:12 sso krb5kdc[5763](info): TGS_REQ (4 etypes {18 17 16 23}) 10.100.114.129: LOOKING_UP_SERVER: authtime 0, host/client1.joongbu.kvm@00NGBU.KVM for nfs/sso@00NGBU.KVM, Server not found in Kerberos database
```

/var/log/secure

```
root@sso:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
May 15 21:44:59 sso sshd[26985]: Connection closed by 10.100.114.129 [preauth]  
May 15 21:45:12 sso sshd[27108]: Accepted password for root from 10.100.114.129 port 59828 ssh2  
May 15 21:46:07 sso sshd[27108]: pam_unix(sshd:session): session opened for user root by (uid=0)  
May 15 21:46:07 sso sshd[27108]: Received disconnect from 10.100.114.129: 11: disconnected by user  
May 15 21:46:07 sso sshd[27108]: pam_unix(sshd:session): session closed for user root  
May 15 21:46:35 sso sshd[27503]: Accepted password for root from 10.100.114.129 port 59829 ssh2  
May 15 21:46:35 sso sshd[27503]: pam_unix(sshd:session): session opened for user root by (uid=0)
```

추가적으로 위의 두 경로에서 로그파일을 통해서 정상적인 티켓팅이 성립되었는지를 확인할 수 있다.

3.3 IDS 구축

snort 설치 및 설정

snort 설치하려면 다음의 패키지와 URL이 필요하다. 깔려있지 않는 것은 yum 으로 설치하면 된다.

- gcc
- flex
- bison
- zlib
- libdnet(libdnet-devel 포함)
- pcre
- tcpdump

snort를 설치 하기위해 <http://www.snort.org> 홈페이지에 들어가면 snort 파일과 daq 파일이 있다.

Downloads

daq-2.0.6-1.centos7.x86_64.rpm
snort-2.9.8.2-1.centos7.x86_64.rpm

여기서 다운받아 쓰면 된다.

```
[root@backup ~]# yum install https://snort.org/downloads/snort/snort-2.9.8.2-1.centos7.x86_64.rpm
Loaded plugins: fastestmirror, langpacks
snort-2.9.8.2-1.centos7.x86_64.rpm | 3.0 MB 00:00:06
Examining /var/tmp/yum-root-wAyBlG/snort-2.9.8.2-1.centos7.x86_64.rpm: 1: snort-2.9.8.2-1.x86_64
Marking /var/tmp/yum-root-wAyBlG/snort-2.9.8.2-1.centos7.x86_64.rpm to be installed
Resolving Dependencies
--> Running transaction check
--> Package snort.x86_64 1:2.9.8.2-1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package Arch Version Repository Size
=====
Installing:
snort x86_64 1:2.9.8.2-1 /snort-2.9.8.2-1.centos7.x86_64 6.8 M
Transaction Summary
=====
Install 1 Package
Total size: 6.8 M
Installed size: 6.8 M
Is this ok [y/d/N]: y
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : 1:snort-2.9.8.2-1.x86_64 1/1
Verifying : 1:snort-2.9.8.2-1.x86_64 1/1
Installed:
snort.x86_64 1:2.9.8.2-1
Complete!
```

snort 파일을 yum 으로 받아와서 설치를 했다.


```
[root@backup ~]# yum install https://snort.org/downloads/snort/daq-2.0.6-1.centos7.x86_64.rpm
Loaded plugins: fastestmirror, langpacks
daq-2.0.6-1.centos7.x86_64.rpm                               | 147 kB      00: 02
Examining /var/tmp/yum-root-wAyBtG/daq-2.0.6-1.centos7.x86_64.rpm: daq-2.0.6-1.x86_64
Marking /var/tmp/yum-root-wAyBtG/daq-2.0.6-1.centos7.x86_64.rpm to be installed
Resolving Dependencies
--> Running transaction check
--> Package daq.x86_64 0:2.0.6-1 will be installed
--> Finished Dependency Resolution

base/7/x86_64                                               | 3.6 kB      00: 00
extras/7/x86_64                                             | 3.4 kB      00: 00
updates/7/x86_64                                            | 3.4 kB      00: 00

Dependencies Resolved

=====
Package Arch Version Repository Size
=====
Installing:
daq x86_64 2.0.6-1 /daq-2.0.6-1.centos7.x86_64 649 k
Transaction Summary
=====
Install 1 Package

Total size: 649 k
Installed size: 649 k
Is this ok [y/d/N]: y
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : daq-2.0.6-1.x86_64 1/1
  Verifying : daq-2.0.6-1.x86_64 1/1

Installed:
daq.x86_64 0:2.0.6-1

Complete!
```

daq 파일도 yum으로 설치 하였다.

```
[root@backup ~]# snort -v
Running in packet dump mode

==== Initializing Snort ====
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "virbr0".
Decoding Ethernet

==== Initialization Complete ====

--> Snort! <*-
o"~ Version 2.9.8.2 GRE (Build 335)
"~ By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
"~ Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.
"~ Copyright (C) 1998-2013 Sourcefire, Inc., et al.
"~ Using libpcap version 1.5.3
"~ Using PCRE version: 8.32 2012-11-30
"~ Using ZLIB version: 1.2.7

Commencing packet processing (pid=11468)
```

snort -v 로 snort 버전을 확인할 수 있다.

```
[root@backup ~]# snort --daq-list
Available DAQ modules:
pcap(v3): readback live multi unpriv
ipfw(v3): live inline multi unpriv
dump(v2): readback live inline multi unpriv
afpacket(v5): live inline multi unpriv
```

snort --daq-list 로 daq 모듈을 확인해 보았다.

snort 패키지에는 룰이 포함되어 있지 않다. 따라서 www.snort.org 에서 무료 이용 가능한 기본적인 snort rule을 받아와 수정해서 사용한다.



[snortrules-snapshot-2980.tar.gz](http://www.snort.org/rules/snortrules-snapshot-2980.tar.gz)

31-Mar-2016 11:30 37M

```
root@backup:/home/backup/다운로드
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
[root@backup 다운로드]# tar zxvf snortrules-snapshot-2980.tar.gz
rules/
rules/VRT-License.txt
rules/app-detect.rules
rules/attack-responses.rules
rules/backdoor.rules
rules/bad-traffic.rules
rules/blacklist.rules
rules/botnet-cnc.rules
rules/browser-chrome.rules
rules/browser-firefox.rules
rules/browser-ie.rules
rules/browser-other.rules
rules/browser-plugins.rules
rules/browser-webkit.rules
rules/chat.rules
rules/content-replace.rules
rules/ddos.rules
rules/deleted.rules
rules/dns.rules
rules/dos.rules
rules/experimental.rules
rules/exploit-kit.rules
rules/exploit.rules
rules/file-executable.rules
rules/file-flash.rules
rules/file-identify.rules
rules/file-image.rules
rules/file-java.rules
rules/file-multimedia.rules
rules/file-office.rules
rules/file-other.rules
rules/file-pdf.rules
rules/finger.rules
rules/ftp.rules
rules/icmp-info.rules
rules/icmp.rules
rules/imap.rules
rules/indicator-compromise.rules
rules/indicator-obfuscation.rules
rules/indicator-scan.rules
rules/indicator-shellcode.rules
rules/info.rules
rules/local.rules
rules/malware-backdoor.rules
rules/malware-cnc.rules
rules/malware-other.rules
rules/malware-tools.rules
rules/misc.rules
rules/multimedia.rules
rules/mysql.rules
```

파일을 받아와서 압축을 풀어준다.

```
#####
# Step #1: Set the network variables. For more information, see README.variables
#####

# Setup the network addresses you are protecting
ipvar HOME_NET 10.100.114.0/24

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET !$HOME_NET

# Path to your rules files (this can be a relative path)
# Note for Windows users: You are advised to make this an absolute path,
# such as: c:\snort\rules
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules
```

vi /etc/snort/snort.conf 들어가서 로컬네트워크 주소 설정과 외부 네트워크 주소 설정을 해주고 rules, so_rules, preproc_rules를 절대경로로 지정해준다.
권한 설정을 아래와 같이 바꿔준다.

```
cd /usr/local/src
```

```
chown -R snort:snort daq-2.0.6-1
chmod -R 700 daq-2.0.6-1
chown -R snort:snort snort-2.9.8.2-1
chmod -R 700 snort-2.9.8.2-1
chown -R snort:snort snort_dynamicsrc
chmod -R 700 snort_dynamicsrc
```

```
cd /var/log
chmod 700 snort
chown -R snort:snort snort
```

```
cd /usr/local/lib
mkdir -p snort_dynamicrules
chown -R snort:snort snort*
chown -R snort:snort pkgconfig
chmod -R 700 snort*
chmod -R 700 pkgconfig
```

```
cd /usr/local/bin
chown -R snort:snort daq-modules-config
chown -R snort:snort u2*
chmod -R 700 daq-modules-config
chmod 700 u2*
```

```
cd /etc
chown -R snort:snort snort
chmod -R 700 snort
```

이제 snort를 사용하기 위한 기본적인 설정을 다 되었다. 이제 사용하고자 하는 룰만 추가해 주면 된다.

우리가 추가해 준 룰들은 다음과 같다. (룰은 /etc/snort/rules 들어가서 파일로 만들어 추가해준다.)

• telnet.rules

vi /etc/snort/rules/telnet.rules
alert tcp 10.100.114.93 23 -> any any (msg:"Telnet login; content:"login failed"; nocase; sid:1000005;)
alert tcp 10.100.114.93 23 -> any any (msg:"Telnet login; content:" "; nocase; sid:1000006;)

sid가 1000005로 구성된 룰로 23포트로 전송하는 패킷중에서 어느 IP든지 자기 자신에 들어오는 것을 검출하라는 것이다. 로그인 실패한 경우 Wireshark에서 "login failed" 문구가 뜨고 로그인 성공한 경우에는 공백 칸이 나타나 두 경우에 모두 "Telnet login" 라는 메시지가 뜨게 추가 하였다.

• ssh.rules

vi /etc/snort/rules/ssh.rules
alert tcp 10.100.114.93 22 -> any any (msg:" SSH alert "; content: "SSH-2.0"; nocase; sid:1000010;)

이 룰은 응답으로 SSH서버가 패킷을 전달할 때 관리자 IP주소가 아닌 주소들을 탐지하라는 것이다. SSH는 모든 패킷이 암호화가 되는 것이 아니라 이전에 버전을 확인하는 평문 구간이 있기 때문에 이 구간에서 content로 사용할 문자열을 찾는다.

• brutessh.rules

vi /etc/snort/rules/brutessh.rules
alert tcp 10.100.114.93 22 -> !10.100.114.211 any (msg:"Brute SSH"; content:"SSH-2.0"; sid:1000011;)

Hydra 도구를 사용해 공격을 받은 10.100.114.93 SSH서버는 응답 패킷으로 자신을 제외한 모든 IP에 content SSH-2.0이라는 문자열을 전송했을 때 검출하라는 룰이다.

• brutetelnet.rules

vi /etc/snort/rules/brutetelnet.rules
alert tcp 10.100.114.93 23 -> any any (msg:"Brute telnet"; content:"Login Failed"; sid:1000012;)

Brute Force Attack을 당하면 탐지한다.

• ftpbrute.rules

vi /etc/snort/rules/ftpbrute.rules
alert tcp 10.100.114.93 21 -> !10.100.114.93 any (msg:"Brute ftp"; content:"Login incorrect"; nocase; sid:1000013;)

Hydra 도구를 이용한 Brute Force Attack을 탐지한다.

• nmap.rules

```
vi /etc/snort/rules/nmap.rules
```

```
alert tcp !10.100.114.0/24 any -> any any (msg:"SYN Check"; flags:S;
sid:1000051;)
alert tcp 10.100.114.0/24 any -> 10.100.114.0 any (msg:"ACK Check"; flags:A;
sid:1000052;)
alert tcp !10.100.114.0/24 any -> any any (msg:"PSH Check"; flags:0;
sid:1000053;)
```

이 룰은 옵션으로 flags를 설정한 이유는 여러번 핑이 발생해야 스캔으로 감지하기 때문이다.

• hping.rules

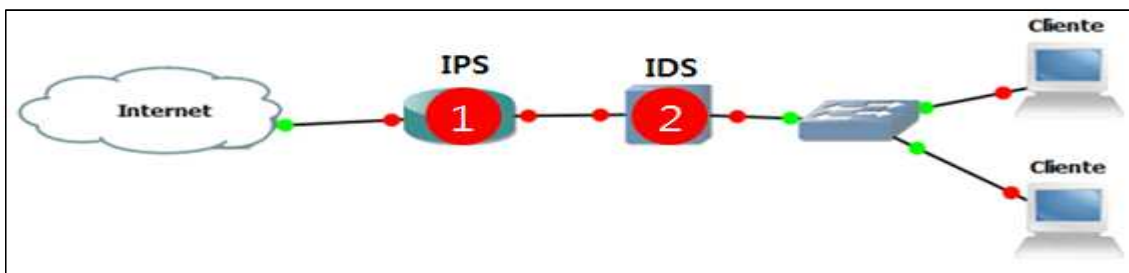
```
vi /etc/snort/rules/hping.rules
```

```
alert ip any any -> 10.100.114.93 any (msg:"Pingof Death";
content:"[5858585858585858]"; sid:1000055;)
alert ip any any -> any any (msg:"Land Attack"; sameip; sid:1000056;)
```

어느 ip에서 오든지 타겟 IP에 5858585858585858 문자열을 가진 패킷을 탐지 한다.

동작 과정

IDS IPS 보안 구조



IDS, IPS를 사용해 침입을 탐지, 방어를 하기위해 위 그림처럼 구현을 하였다.

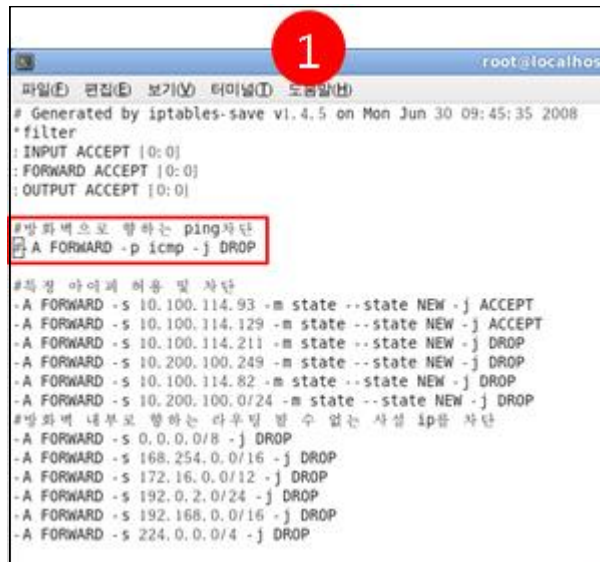
```
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
[ root@so ~ ]#
[ root@so ~ ]# snort -A fast -b -i enp3s0 -u snort -g snort -c /etc/snort/snort.conf
Running in IDS mode

--== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined : [ 80: 81 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 3037 3128
3702 4343 4848 5250 6988 7000: 7001 7144: 7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118
8123 8180: 8181 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090: 9091 9443 9999 11371 34443: 34444 41080 5
0002 55555 ]
```

IDS(snort)를 가동 시켜 침입탐지를 구동 시킨다.

```
vi /etc/snort/rules/icmp.rules
alert icmp any any -> any any (msg : "ICMP test"; sid:500001;)
```

이때 침입탐지 규칙에서는 icmp패킷을 침입이라 판단하도록 설정 하였다.



```
root@localhost
# Generated by iptables-save v1.4.5 on Mon Jun 30 09:45:35 2008
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
#방화벽으로 할하는 ping차단
-A FORWARD -p icmp -j DROP

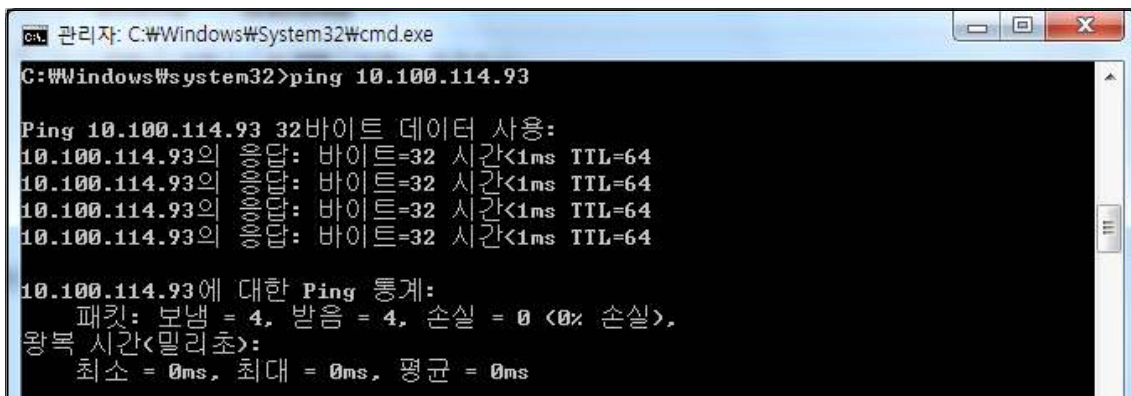
#특정 영역의 허용 및 차단
-A FORWARD -s 10.100.114.93 -m state --state NEW -j ACCEPT
-A FORWARD -s 10.100.114.129 -m state --state NEW -j ACCEPT
-A FORWARD -s 10.100.114.211 -m state --state NEW -j DROP
-A FORWARD -s 10.200.100.249 -m state --state NEW -j DROP
-A FORWARD -s 10.100.114.82 -m state --state NEW -j DROP
-A FORWARD -s 10.200.100.0/24 -m state --state NEW -j DROP
#방화벽 내부로 할하는 라우팅 할 수 없는 사설 ip를 차단
-A FORWARD -s 0.0.0.0/8 -j DROP
-A FORWARD -s 168.254.0.0/16 -j DROP
-A FORWARD -s 172.16.0.0/12 -j DROP
-A FORWARD -s 192.0.2.0/24 -j DROP
-A FORWARD -s 192.168.0.0/16 -j DROP
-A FORWARD -s 224.0.0.0/4 -j DROP
```

IPS에서는 모의침입이 보여지기 쉽게 icmp패킷을 허용을 하였다.



```
root@localhost /etc
[ root@localhost etc ]# service iptables restart
iptables: 방화벽 규칙을 지웁니다: [ OK ]
iptables: chain을 ACCEPT 규칙으로 설정 중: filter [ OK ]
iptables: 모듈을 언로드하는 중: [ OK ]
iptables: 방화벽 규칙 적용 중: [ OK ]
```

방화벽(iptables)을 수정한 뒤에 방화벽을 재실행 해준다.

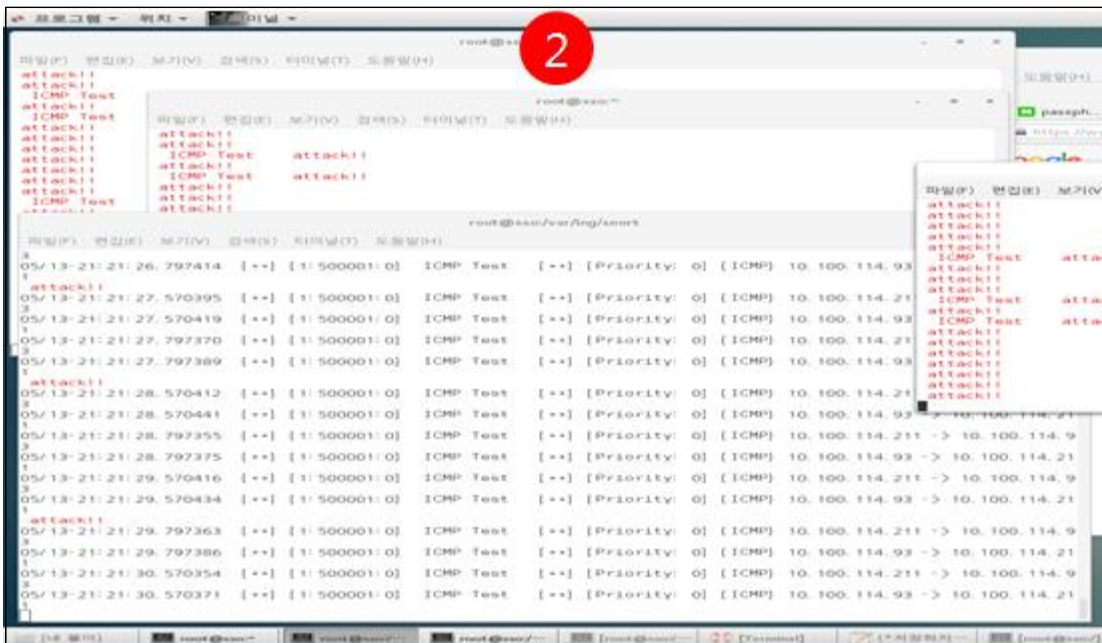


```
C:\Windows\system32>ping 10.100.114.93

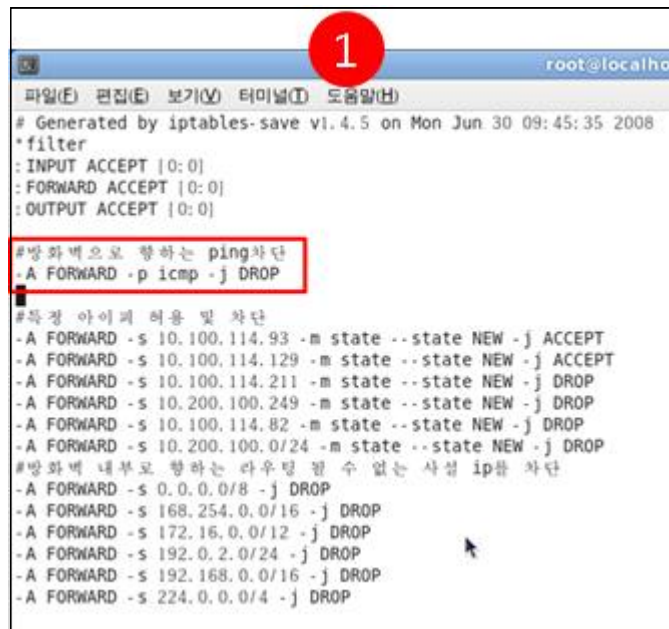
Ping 10.100.114.93 32바이트 데이터 사용:
10.100.114.93의 응답: 바이트=32 시간<1ms TTL=64
10.100.114.93의 응답: 바이트=32 시간<1ms TTL=64
10.100.114.93의 응답: 바이트=32 시간<1ms TTL=64
10.100.114.93의 응답: 바이트=32 시간<1ms TTL=64

10.100.114.93에 대한 Ping 통계:
    패킷: 보낸 = 4, 받음 = 4, 손실 = 0 (0% 손실),
    왕복 시간<밀리초>:
        최소 = 0ms, 최대 = 0ms, 평균 = 0ms
```

외부에서 icmp패킷을 전송한다.



icmp패킷을 전송하면 IDS(snort)로 패킷을 확인, 경고 문구를 확인 할 수 있다.



침입을 확인을 한다면 IPS에서 특정 패킷을 차단 또는 특정 아이피를 차단 할 수 있다. 그림에서는 특정 패킷을 차단하였다.

이렇게 경고문구(alert)와 로그를 확인을 하게 되면 방화벽에서는 해당 패킷 또는 ip를 차단하여 침입에 보다 자세하게 보안을 할 수 있다.

3.4 vsFTP 운영

vsftp를 이용한 로그파일 백업

```
root@backup:~  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@backup ~]# yum -y install ftp vsftpd  
Loaded plugins: fastestmirror, langpacks  
Loading mirror speeds from cached hostfile  
* base: ftp.daumkakao.com  
* extras: ftp.daumkakao.com  
* updates: ftp.daumkakao.com  
Package ftp-0.17-66.el7.x86_64 already installed and latest version  
Package vsftpd-3.0.2-11.el7_2.x86_64 already installed and latest version  
Nothing to do  
[root@backup ~]#
```

yum -y install ftp vsftpd로 설치

```
root@backup:/etc/vsftpd  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[root@backup ~]# cd /etc/vsftpd  
You have new mail in /var/spool/mail/root  
[root@backup vsftpd]# ls  
chroot_list  ftpusers  user_list  vsftpd.conf  vsftpd_conf_migrate.sh  
[root@backup vsftpd]#
```

설치 후 /etc/vsftpd에서 ftpusers , user_list , vsftpd와 같은 초기설정파일들을 확인

```
vi /etc/vsftpd/user_list
```

```
root@backup:/etc/vsftpd  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
## vsftpd userlist  
# If userlist_deny=NO, only allow users in this file  
# If userlist_deny=YES (default), never allow users in this file, and  
# do not even prompt for a password.  
# Note that the default vsftpd pam config also checks /etc/vsftpd/ftpusers  
# for users that are denied.  
root  
bin  
daemon  
adm  
lp  
sync  
shutdown  
halt  
mail  
news  
uucp  
operator  
games  
nobody  
backup  
admin
```

먼저 user_list 파일은 ftp접속을 할 수 있도록 사용자들을 등록하는 설정파일이다. 그러므로 ftp서버에 접속을 가능하게 하려면 user_list 파일에 사용자를 명시하여야 한다.

```
vi /etc/vsftpd/ftpusers
```

```
root@backup:/etc/vsftpd
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
# Users that are not allowed to login via ftp
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
news
uucp
operator
games
nobody
```

ftpusers 파일은 ftp접속에 대하여 허용을 하지 않는 사용자를 등록하는 파일이며 해당 파일에 특정 사용자가 명시되었을 경우 ftp 접속이 불가능하다.

```
vi /etc/vsftpd/vsftpd.conf
```

```
root@backup:/etc/vsftpd
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
## Example config file /etc/vsftpd/vsftpd.conf
#
# The default compiled in settings are fairly paranoid. This sample file
# loosens things up a bit, to make the ftp daemon more usable.
# Please see vsftpd.conf.5 for all compiled in defaults.
#
# READ THIS: This example file is NOT an exhaustive list of vsftpd options.
# Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's
# capabilities.
#
# Allow anonymous FTP? (Beware - allowed by default if you comment this out).
anonymous_enable=NO
#
# Uncomment this to allow local users to log in.
# When SELinux is enforcing check for SE bool ftp_home_dir
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
write_enable=YES
#
# Default umask for local users is 077. You may wish to change this to 022,
# if your users expect that (022 is used by most other ftpd's)
local_umask=022
```

```
anonymous_enable=NO (익명접속 차단)
anon_upload_enable=NO (익명 업로드 차단)
anon_mkdir_write_enable=YES (익명 디렉토리 생성제한)
chroot_local_user=YES
chroot_list_enable=YES
chroot_list_file=/etc/vsftpd/chroot_list
```

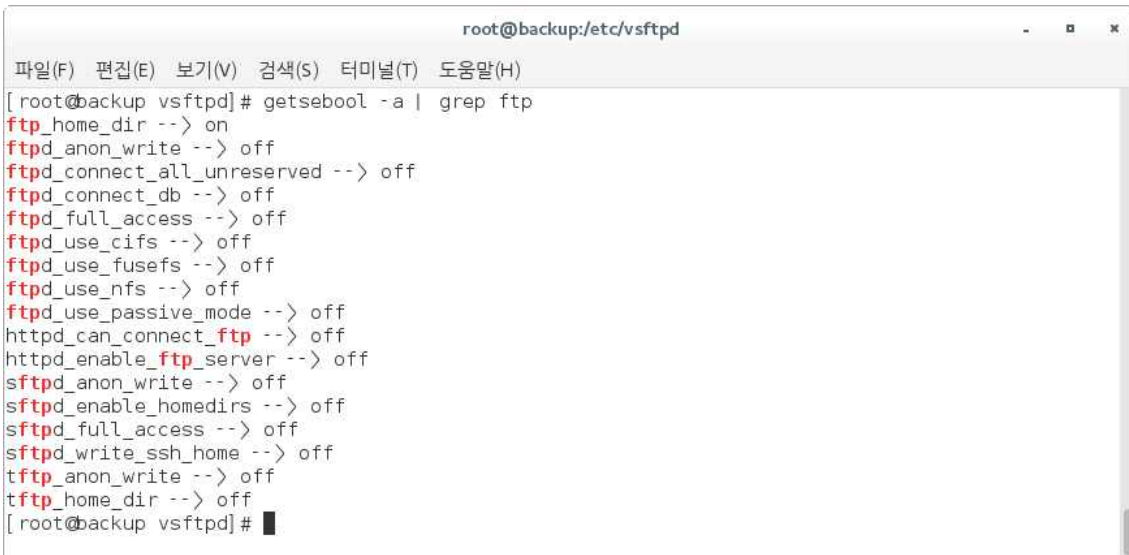
이 설정을 사용함으로써 chroot_list_file에 설정된 사용자를 제외한 모든 사용자들은 상위 디렉토리에 접근할 수 없다.

그리고 vsftpd.conf 파일에서는 ftp를 사용하는데 있어서 필요한 전송방식, 포트 설정등과 같은 여러 가지 설정을 할 수 있으며 추가적으로 보안방법에 대하여 추가가 가능하다.

```
# TLS/SSL 활성화
ssl_enable=YES (로그인 시 TLS를 사용)
allow_anon_ssl=NO
force_anon_logins_ssl=NO
force_anon_data_ssl=NO
force_local_data_ssl=NO
force_local_logins_ssl=YES
ssl_tlsv1=YES
ssl_ciphers=HIGH

# SSL 인증서/비밀키 위치 지정
rsa_cert_file=/etc/ssl/certs/vsftpd.pem
rsa_private_key_file=/etc/ssl/certs/vsftpd.pem
```

```
vi /etc/vsftpd/selinux
```



```
root@backup:/etc/vsftpd
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
[root@backup vsftpd]# getsebool -a | grep ftp
ftp_home_dir --> on
ftp_d_anon_write --> off
ftp_d_connect_all_unreserved --> off
ftp_d_connect_db --> off
ftp_d_full_access --> off
ftp_d_use_cifs --> off
ftp_d_use_fusefs --> off
ftp_d_use_nfs --> off
ftp_d_use_passive_mode --> off
httpd_can_connect_ftp --> off
httpd_enable_ftp_server --> off
sftp_d_anon_write --> off
sftp_d_enable_homedirs --> off
sftp_d_full_access --> off
sftp_d_write_ssh_home --> off
tftp_anon_write --> off
tftp_home_dir --> off
[root@backup vsftpd]#
```

```
getsebool -a | grep ftp
setsebool -P ftp_home_dir on
```

selinux에 의해 ftp 초기설정이 제한되어있기 때문에 위 명령을 사용하여 selinux의 옵션을 확인하고 사용제한을 허용한다. 그리고 설정파일 설정 후에는 vsftpd를 재가동 한다.

TLS 보안을 위한 인증서 생성

```
openssl req -x509 -days 365 -newkey rsa:2048 -nodes -keyout  
/etc/ssl/certs/vsftpd.pem -out /etc/ssl/certs/vsftpd.pem
```

```
Generating a 2048 bit RSA private key  
.....+++  
.....+++  
writing new private key to '/etc/ssl/certs/vsftpd.pem'  
-----  
You are about to be asked to enter information that will be incorporated  
into your certificate request.  
What you are about to enter is what is called a Distinguished Name or a DN.  
There are quite a few fields but you can leave some blank  
For some fields there will be a default value,  
If you enter '.', the field will be left blank.  
-----  
Country Name (2 letter code) [XX]:KR  
State or Province Name (full name) []:GY  
Locality Name (eg, city) [Default City]:GY  
Organization Name (eg, company) [Default Company Ltd]:JB  
Organizational Unit Name (eg, section) []:Security  
Common Name (eg, your name or your server's hostname) []:admin  
Email Address []:admin@naver.com
```

```
root@backup:/etc/ssl/certs  
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)  
[ root@backup certs] # ls  
Makefile  ca-bundle.crt  ca-bundle.trust.crt  make-dummy-cert  renew-dummy-cert  vsftpd.pem  
[ root@backup certs] #
```

/etc/ssl/certs에 인증서가 생성된 것을 확인할 수 있다.

ftp접속

인증이 필요함

 ftp://10.100.114.211 사용자명 및 암호 입력

사용자 이름:

암호:

ftp://10.100.114.211 로 접속을 시도하여 계정과 비밀번호를 입력하여 정상적으로 로그인을 하였을 때 데이터가 암호화 되는지 확인한다.

TLS 인증 사용 전 ftp패킷

No.	Time	Source	Destination	Protocol	Length	Info
90	25.306756000	192.168.210.1	192.168.210.153	FTP	66	Request: USER admin
92	25.306965000	192.168.210.153	192.168.210.1	FTP	88	Response: 331 Please specify the password.
94	25.307084000	192.168.210.1	192.168.210.153	FTP	69	Request: PASS 12341234
103	28.378896000	192.168.210.153	192.168.210.1	FTP	77	Response: 230 Login successful.
105	28.379722000	192.168.210.1	192.168.210.153	FTP	61	Request: CWD /
107	28.380033000	192.168.210.153	192.168.210.1	FTP	91	Response: 250 Directory successfully changed.

Internet Protocol Version 4, Src: 192.168.210.1 (192.168.210.1), Dst: 192.168.210.153 (192.168.210.153)

Transmission Control Protocol, Src Port: 8984 (8984), Dst Port: 21 (21), Seq: 13, Ack: 55, Len: 15

File Transfer Protocol (FTP)

PASS 12341234\r\n

Request command: PASS

Request arg: 12341234

0000 00 0c 29 b1 d0 e4 00 50 56 c0 00 08 08 00 45 00 ..)...P V....E.

0010 00 37 0b 30 40 00 80 06 c9 a4 c0 a8 d2 01 c0 a8 .7.0@...

0020 d2 99 23 18 00 15 c8 3d 76 bd 35 52 51 fc 50 18 ..#....= V.5RQ.P.

0030 40 1b c5 d4 00 00 50 41 53 53 20 31 32 33 34 31 @.....PA SS 12341

0040 32 33 34 0d 0aPA SS 12341

패킷에서 보이는 것처럼 사용자 계정(admin)과 비밀번호가 노출되는 것을 확인할 수 있다.

TLS 인증 사용 후 ftp패킷

No.	Time	Source	Destination	Proto	Length	Info
3148	638.975257903	10.100.114.211	59.18.35.50	TLSv1.2	281	Client Hello
3150	639.009325924	59.18.35.50	10.100.114.211	TLSv1.2	2114	Server Hello
3154	639.009488299	59.18.35.50	10.100.114.211	TLSv1.2	525	Certificate
3156	639.01238486	10.100.114.211	59.18.35.50	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Hello Request, Hello Request
3157	639.012457778	10.100.114.211	59.18.35.50	TLSv1.2	1514	Application Data, Application Data
3163	639.012514090	10.100.114.211	59.18.35.50	TLSv1.2	335	Application Data
3167	639.045075083	59.18.35.50	10.100.114.211	TLSv1.2	384	New Session Ticket, Change Cipher Spec, Hello Request, Hello Request, Application Data
3168	639.045100076	59.18.35.50	10.100.114.211	TLSv1.2	108	Application Data
3169	639.045218939	10.100.114.211	59.18.35.50	TLSv1.2	104	Application Data
3170	639.046082763	59.18.35.50	10.100.114.211	TLSv1.2	104	Application Data
3174	639.09999060	59.18.35.50	10.100.114.211	TLSv1.2	258	Application Data
3176	639.100293269	59.18.35.50	10.100.114.211	TLSv1.2	624	Application Data, Application Data
3178	639.100401624	10.100.114.211	59.18.35.50	TLSv1.2	112	Application Data
3182	639.102226838	10.100.114.211	59.18.35.50	TLSv1.2	228	Application Data
3185	639.136531261	59.18.35.50	10.100.114.211	TLSv1.2	433	Application Data, Application Data
3186	639.137084880	59.18.35.50	10.100.114.211	TLSv1.2	112	Application Data
3188	639.137509549	10.100.114.211	59.18.35.50	TLSv1.2	112	Application Data
3189	639.138200554	10.100.114.211	59.18.35.50	TLSv1.2	203	Application Data
3192	639.172576645	59.18.35.50	10.100.114.211	TLSv1.2	146	Application Data
3193	639.173396054	59.18.35.50	10.100.114.211	TLSv1.2	1759	Application Data, Application Data, Application Data

Frame 3192: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits) on interface 0

Ethernet II, Src: JuniperN_2a:48:01 (10:0e:7e:2a:48:01), Dst: AsustekC_54:cd:ac (14:dd:a9:54:cd:ac)

Internet Protocol Version 4, Src: 59.18.35.50 (59.18.35.50), Dst: 10.100.114.211 (10.100.114.211)

Version: 4

0000 14 dd a9 54 cd ac 10 0e 7e 2a 48 01 0e 00 45 00 ...T....-H...E.

0010 00 94 26 ad 00 00 38 06 80 4c 3b 12 23 32 0a 64 ..&...&...L...#2;d

0020 72 d3 01 bb 8c 82 cc ee 95 d8 f1 e5 c6 44 80 18 r.....D..

0030 01 a4 b0 6a 00 00 01 01 08 0a 73 99 4f b3 0e c1 ...j.....s.O...

0040 56 f4 17 03 03 00 4b 00 00 00 00 00 00 0a a9 V.....K.

0050 ef 3b 1d 89 9e b3 18 f2 7a 90 49 aa d9 62 a8 ba o;.....Z.I..b..

0060 9e 52 9d ae 19 c2 f5 ad 78 ee 28 29 3c ca a5 c8 .R.....x.()<...

0070 6a 62 aa 10 c0 bb bd eb ff 9b ef d8 03 16 70 a6 jb.....P.

0080 2f 56 68 cc cd f3 cf 64 15 a2 a8 2f 9e 2e cc 6b /Vh....d .../...k

0090 b8 c8 ..

인증 전과는 달리 사용자와 비밀번호에 대한 정보를 확인할 수가 없다.

ssh를 사용한 로그파일전송 데이터 패킷

No.	Time	Source	Destination	Proto	Length	Info
242	51.772392398	10.100.114.211	10.100.114.93	SSHv2	89	Encrypted request packet len=23
244	51.780267101	10.100.114.93	10.100.114.211	SSHv2	89	Encrypted response packet len=23
247	51.780940058	10.100.114.211	10.100.114.93	SSHv2	586	Client: Key Exchange Init
249	51.783062353	10.100.114.93	10.100.114.211	SSHv2	1706	Server: Key Exchange Init
251	51.785119837	10.100.114.211	10.100.114.93	SSHv2	114	Client: Diffie-Hellman Key Exchange Init
252	51.788427915	10.100.114.93	10.100.114.211	SSHv2	346	Server: New Keys
253	51.791209234	10.100.114.211	10.100.114.93	SSHv2	82	Client: New Keys
255	51.831562142	10.100.114.211	10.100.114.93	SSHv2	118	Encrypted request packet len=52
257	51.832005753	10.100.114.93	10.100.114.211	SSHv2	118	Encrypted response packet len=52
258	51.832067972	10.100.114.211	10.100.114.93	SSHv2	134	Encrypted request packet len=68
259	51.835215917	10.100.114.93	10.100.114.211	SSHv2	150	Encrypted response packet len=84
260	51.836667161	10.100.114.211	10.100.114.93	SSHv2	438	Encrypted request packet len=372
261	51.838113646	10.100.114.93	10.100.114.211	SSHv2	390	Encrypted response packet len=324
262	51.839680687	10.100.114.211	10.100.114.93	SSHv2	710	Encrypted request packet len=644
263	51.843568996	10.100.114.93	10.100.114.211	SSHv2	102	Encrypted response packet len=36
264	51.843654143	10.100.114.211	10.100.114.93	SSHv2	186	Encrypted request packet len=120
266	51.908945141	10.100.114.93	10.100.114.211	SSHv2	118	Encrypted response packet len=52
267	51.909038561	10.100.114.211	10.100.114.93	SSHv2	250	Encrypted request packet len=184
269	51.911831151	10.100.114.93	10.100.114.211	SSHv2	138	Encrypted response packet len=72
270	51.911889234	10.100.114.211	10.100.114.93	SSHv2	118	Encrypted request packet len=52

Frame 267: 250 bytes on wire (2000 bits), 250 bytes captured (2000 bits) on interface 0

Ethernet II, Src: AsustekC_54:cd:ac (14:dd:a9:54:cd:ac), Dst: AsustekC_54:d0:0d (14:dd:a9:54:d0:0d)

Internet Protocol Version 4, Src: 10.100.114.211 (10.100.114.211), Dst: 10.100.114.93 (10.100.114.93)

Version: 4

Header length: 20 bytes

Differentiated Services Field: 0x08 (DSCP: 0x02: Unknown DSCP; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))

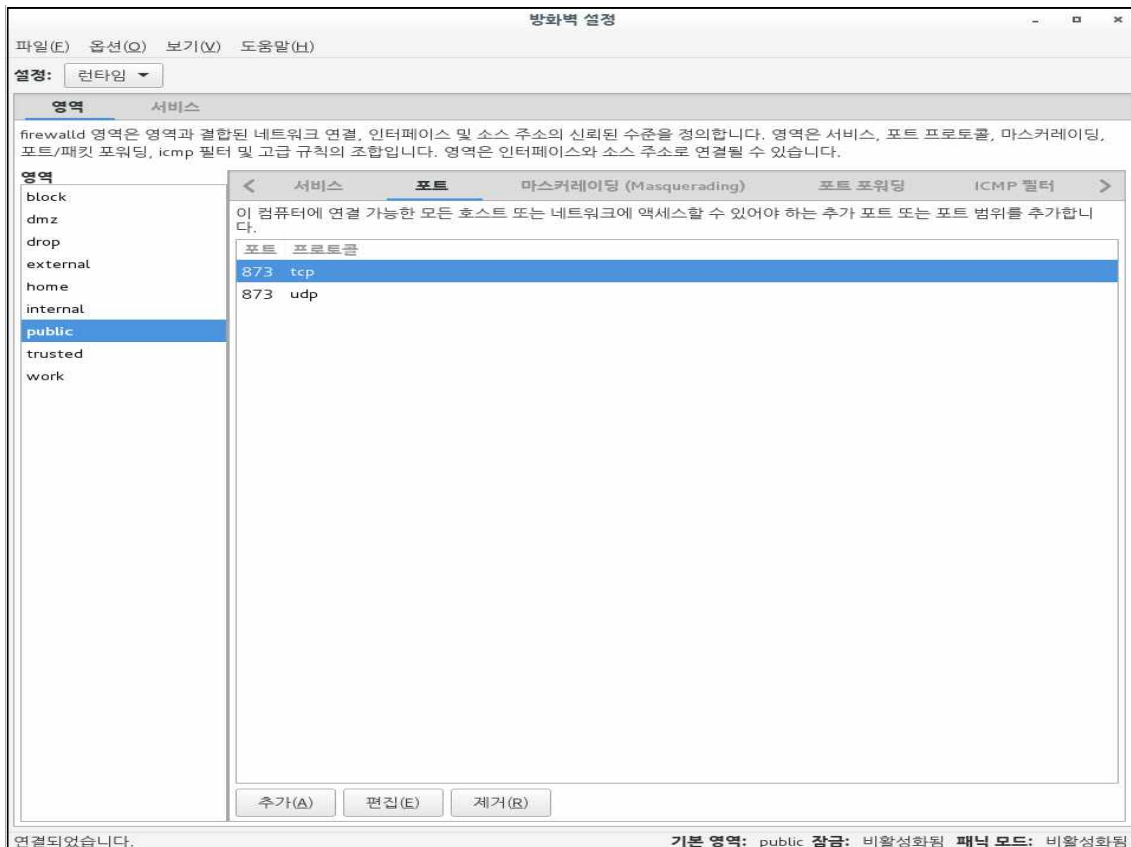
0000 10.. = Differentiated Services Codepoint: Unknown (0x02)

0000 14 dd a9 54 d0 0d 14 dd a9 54 cd ac 08 00 45 08 ...T...T...E.
0010 00 ec 13 69 40 00 04 06 2c a3 0a 64 72 d3 0a 64 ...i8.0...dr..d
0020 72 5d 87 c0 00 16 91 dc 80 c5 ad a2 40 70 80 18 r].....@..
0030 01 2b c4 83 00 00 01 01 08 0a 0e b8 61 17 90 bd .+.....8...
0040 5d dc 00 00 00 30 9e ac 77 8a 85 fc a2 e4 f6 18]....0..w.....
0050 a9 28 10 93 8e 41 80 7e 1a 41 14 cf 9e 8f 8a 80 .{...A~.A...>...
0060 d8 37 b5 6d c7 ee 40 36 7c ce 4a 91 64 a6 86 d2 .7.m.@6 l.j.d...
0070 a6 69 55 e1 33 f6 8c e6 9c 72 13 55 f3 72 41 61 .iU3...r.UrAa
0080 38 06 d3 22 cb 09 00 00 60 33 00 68 d0 d6 b6 8..*...3.h...
0090 35 1f ae 6d d4 69 9e 65 2f 76 d7 43 b9 54 f1 ae 5..m.i.e /v.C.T..
00a0 82 db 04 ce 18 8c a8 db c8 52 dd d5 e0 a4 81 43R.....C
00b0 01 47 12 33 ce 3e 04 83 3c d9 91 2b e0 42 e2 3d .G.3>..<.+B=...
00c0 b6 a4 51 93 96 89 b5 ee 6b 2b 88 9c 7f 80 24 61 ..Q.....k?...4a
00d0 d8 cc b6 fa b1 e7 d1 fc 7e d2 f7 ac 31 93 3c f3~...l.<...
00e0 93 87 a3 d6 16 04 d2 57 4a b9 3f b8 fb 84 36 0aW J.?...6.

일반적으로 ftp로 단순히 접속 후, 파일을 전송하게 되면 평문으로 전송되어 보안에 취약하다. 그러므로 위와 같은 방법으로 ftp접속을 하는데 있어서 인증서를 사용하여 계정의 정보가 노출되지 않도록 함으로써 데이터를 전송할 수 있다. 그리고 ftp 데이터 전송에는 22번 포트를 사용하는 SSH 통신을 이용하기 때문에 SSHv2로 암호화되어 데이터가 전송되는 것을 확인할 수 있다.

3.5 백업서버 구축

웬만한 Centos7 에는 Rsync가 깔려있다. 깔려있지 않으면 `yum -y install rsync` 로 다운 받으면 된다.



방화벽에 들어가서 rsync 포트번호인 873 번을 열어 준다.

`vi /etc/xinetd.d/rsync` 설정 파일 들어가서 `disalbe=yes`를 `no`로 수정해 준다.

```
service rsync
{
    disable=no
    socket_type = stream
    wait = no
    user = root
    server = /usr/bin/rsync
    server_args = -daemon
    log_on_failure += USERD
}
```

```
service rsync
{
    disable=no
    socket_type = stream
    wait = no
    user = root
    server = /usr/bin/rsync
    server_args = -daemon
    log_on_failure += USERD
}
```

vi /etc/xinetd.d/rsync 설정 파일 들어가서 disable=yes를 no로 수정해 준다.

```
service rsync
{
    disable=no
    socket_type = stream
    wait = no
    user = root
    server = /usr/bin/rsync
    server_args = -daemon
    log_on_failure += USERD
}
```

vi /etc/rsyncd.conf 설정 파일 들어가서 위와 같이 수정해준다.

```
[rsync]
path=/var/log/snort
comment = rsync
uid = root
gid = root
user chroot = yes
read only = yes
hosts allow = 10.100.114.93
max connections = 3
timeout = 600
```

rsyncd.conf 설정파일 정보	
path	데이터 경로
comment	코멘트
uid	사용자 아이디
gid	그룹 아이디
use chroot	yes
read only	yes
hosts allow	클라이언트 아이피
max connections	최대 접속자 수
timeout	타임아웃 값

```
systemctl restart xinetd
```

rsync 설정을 적용하기 위해 xinetd를 재시작 해준다.

```
rsync -avz [서버 IP주소]::[rsync 명] [저장할 디렉토리]
```

rsync 명령어	
-a	심볼릭 링크, 속성, 퍼미션, 소유권 등 보관
-v	자세한 정보 출력
-z	전송시 압축
-r	하위 디렉토리 포함
-e	ssh를 이용한 rsync 동기화
--delete	서버 동기화 후 rsync서버에서 파일이 삭제 되었으면 클라이언트도 대상 파일을 삭제

SSH Key 만들기

SSH Key를 통해서 서버에 접속 할 때 Unix 계열(리눅스, 맥)에서는 ssh-keygen이라는 프로그램을 이용하면 된다.

ssh-keygen 사용하기

아래와 같이 입력한다. -t rsa는 rsa라는 암호화 방식으로 키를 생성한다는 의미다.

이제 id_rsa.pub 파일을 리모트 서버의 \$HOME/.ssh/authorized_keys 파일에 추가해 줘야 한다. 아래의 그림을 보자.



SSH Client



SSH Server

id_rsa.pub
id_rsa

=

authorized_keys

SSH Server의 authorized_keys 의 내용이 SSH Client의 id_rsa.pub 파일과 같아야 한다. 그래서 ssh 접속을 할 때 id_rsa 파일과 authorized_keys 파일의 내용을 비교 할 수 있다.

```
root@server1 ~]# scp $HOME/.ssh/id_rsa.pub root@192.168.210.155:/root/.ssh/id_rsa.pub
root@192.168.210.155's password:
id_rsa.pub                                100% 394      0.4KB/s   00: 00
```

scp 명령어로 로컬 머신의 id_rsa.pub 파일을 리모트 머신의 홈디렉토리로 전송을 한다.

이제 원격 머신에서 전송한 id_rsa.pub 파일을 authorized_keys 파일에 추가한다. 내용을 교체하는 것이 아니라 추가하는 것이라는 점이다. 만약 리모트 머신으로 접속하는 여러개의 로컬 머신이 있다면 각각의 로컬 머신의 id_rsa.pub 파일을 authorized_keys에 추가해주면 된다.

```
cat $HOME/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

이렇게하면 SSH 로그인 비밀번호 없이 입력할 수 있게 되어 rsync 동기화를 간편하게 할 수 있게 된다.

```
* * * * * /backup/rsync.sh
```

rsync.sh 쉘을 매분마다 실행한다

```
vi /backup/rsync.sh
```

```
rsync -av -e ssh 10.100.114.93:/var/log/snort/day_log /home/admin
```

rsync.sh 쉘은 백업 디렉토리인 /home/admin에 원격서버인 10.100.114.93 에 접속해 /var/log/snort/day_log 안에 있는 디렉토리 자체를 동기화 한다.

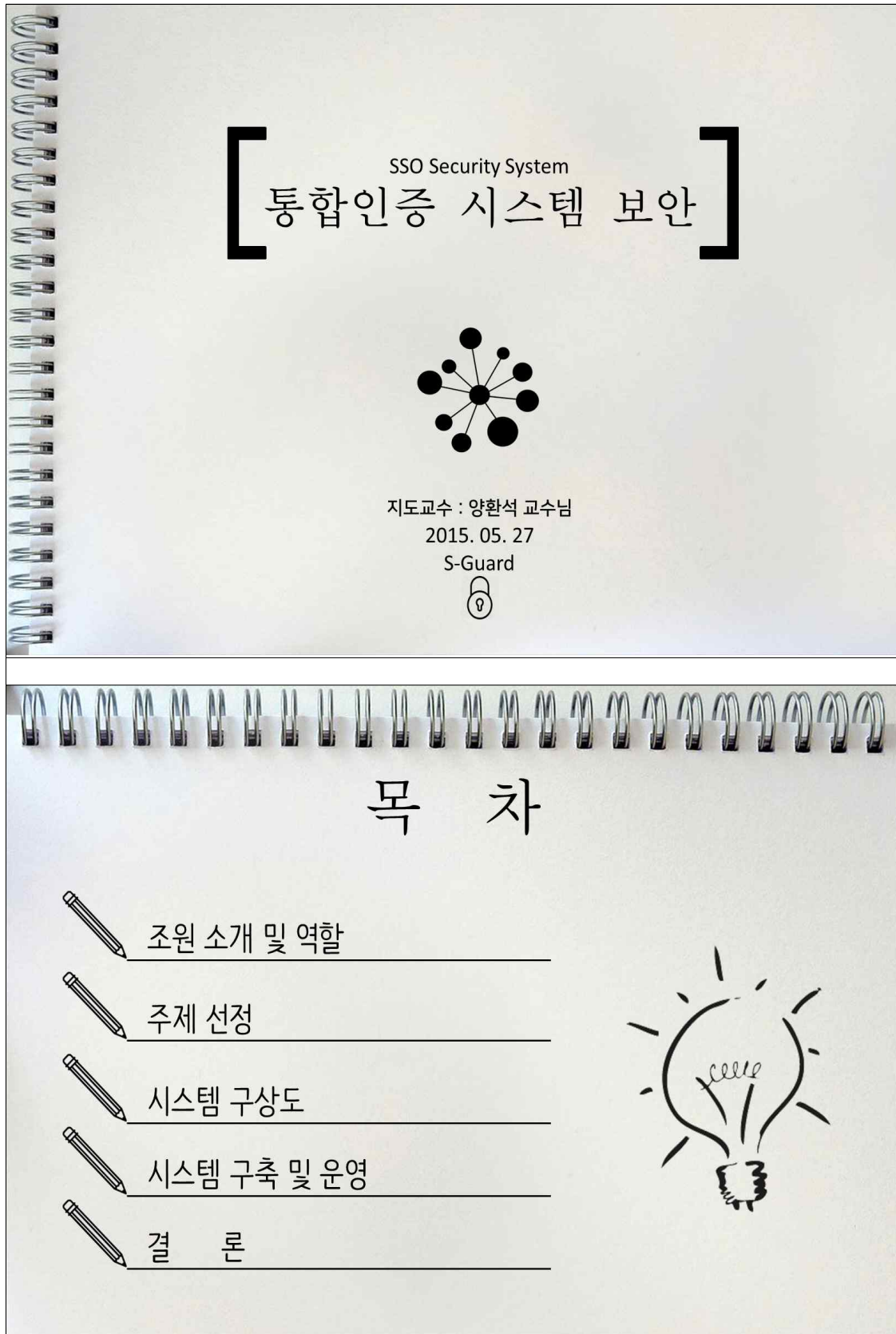
4. 결론

한 번의 로그인으로 여러 서비스를 이용할 수 있는 Single Sign On 시스템에는 비교적 취약한 단점이 존재한다. 의심되는 제3자가 원격접속 시도를 통하여 중앙관리 서버의 인증세션을 탈취할 경우 여러 사용자들의 인증정보(ID/PW)가 노출될 가능성이 있다. 그렇기 때문에 이번 프로젝트에서는 브릿지 설정을 통한 방화벽을 구축해서 허용된 IP대역을 제외한 곳으로부터 내부망을 안전하게 분리하여 혹시 모를 공격을 미연에 방지했으며, 방화벽 뒤에서는 침입탐지 시스템을 구축하여 방화벽을 우회한 외부로부터의 접근을 탐지하여 SSO서버로 접근하려는 해당 IP를 차단할 수 있도록 구현하였다. 그리고 침입탐지 시 발생하는 로그파일을 분 단위, 하루단위 보고서로 생성되어 백업서버에 백업되도록 하였고, 생성된 로그파일을 웹브라우저에서 ftp 서버로 접속하여 생성된 로그파일을 확인할 수 있게 하였다.

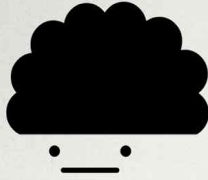
5. 참고 자료

웨일 소잉카(2014). 『가장 쉬운 리눅스 시스템 관리』. 서울: 비제이퍼블릭
우재남(2005). 『뇌를 자극하는 Redhat Fedora』. 서울 : 한빛미디어
카사노 히데마츠(2014). 『28일 동안 배우는 리눅스 서버 관리』. 서울 : 한빛미디어

6. 발표 자료



조원 소개



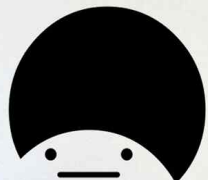
조 장 : 이유근

역 할 : 졸업작품 총괄 및 방화벽 구축



조 원 : 안 헤민

역 할 : SSO, FTP 서버 구축



조 원 : 이 종화

역 할 : IPS, 백업 서버 구축

주제 선정

주제 : 통합인증 시스템 보안

중앙관리서버가 외부공격으로부터 마비되거나 사용자 인증정보가 노출될 경우 심각한 위험을 초래

통합암호 'SSO'...단 한번의 편리함...그 뒤엔...한번에 돌릴 걱정

2013.04.19.

f i t

최근 데이터를 하나하나 쌓아두고 활용하는 빅데이터 경영이 화두로 떠오르
통합 관리하는 'SSO(Single Sign On)'가 각광받고 있다. 하지만 SSO의 통
기 한꺼번에 누출될 수 있어 기업들의 각별한 주의가 요구된다.

SSO는 단일 인증으로 한 시스템에서 연관 시스템에 저등으로 인증을 처리하
리저 계량만으로 다수의 서비스에 접근할 수 있는 셈이다. 사용자는 비밀번호
의 인력과 비용을 절감할 수 있다는 장점이 있다. 구글의 오픈 서비스나 세일
인 사례다.

SSO(싱글사인온)는 양면의 동전. 한번에 통하지만 한번에 당할 수도 있어

헤럴드POP

또 인증 정보를 재사용하거나 보안 검증을 받지 않은 암호 모듈을 사용하는 경우, 악티브X 컨트롤을 관례해서도
SSO의 취약점이 노출될 수 있다. 이에 전문가들은 SSO일수록 보안 기능을 더욱 촘촘히 갖춰야 한다고...

통합암호 'SSO'...단 한번의 편리함... 헤럴드경제



이러한 취약점을 보안하기 위해 방화벽을 구축한 뒤 허용되지 않는 IP는 내부망으로 접근하지 못하도록 관리하였고
만약, 방화벽을 우회해서 의심되는 IP가 들어온다면 침입탐지 시스템으로 탐지하여 차단할 수 있도록 구현

추진 일정

구분	2016년		
	3월	4월	5월
SSO 구축			
방화벽 구축			
IDS/IPS 구축			
웹 스크립트 작성			
백업/FTP서버			
시스템 점검 및 보완			

시스템 구상도

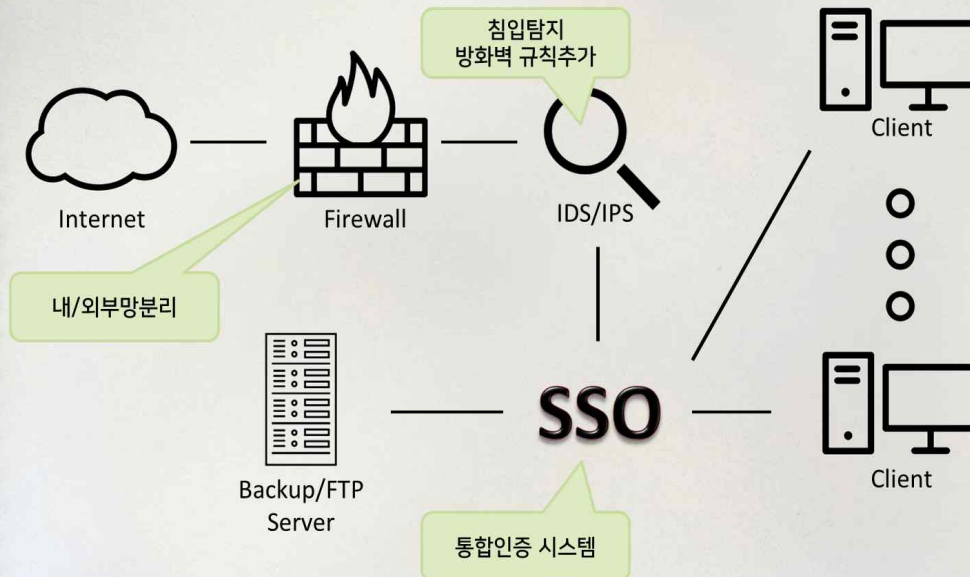
운영환경

운영 체제 : Centos 7, Fedora 12

인증 관리 서버 : LDAP, Kerberos 서버

응용 서버 : FTP, SSH 서버


시스템 구상도



시스템 구축 및 운영(1)

eth0=input

eth1=output



```

vi ifcfg-br0

DEVICE=br0
TYPE=Bridge
ONBOOT=yes
IPADDR=192.168.1.44(임의 값)
GATEWAY=192.168.0.254
NETMASK=255.255.255.0
DNS1=dns값
DNS2=dns값
                    
```

```

vi ifcfg-eth0

DEVICE=eth0
BOOTPROTO=static
ONBOOT=yes
BRIDGE=br0
TYPE=Ethernet
                    
```

```

vi ifcfg-eth1

DEVICE=eth1
BOOTPROTO=static
ONBOOT=yes
BRIDGE=br0
TYPE=Ethernet
                    
```

랜 카드를 2개를 장착한 뒤 br0, eth0, eth1에 설정을 적용

시스템 구축 및 운영(2)

```
[root@ss0 rules]# ls
brutessh.rules  ftpbrute.rules  nmap.rules  telnet.rules
brutetelnet.rules  hping.rules  ssh.rules
```

vi /etc/snort/rules/nmap.rules

```
alert tcp !10.100.114.0/24 any -> any any (msg:"SYN
Check"; flags:S; sid:1000051;)
alert tcp 10.100.114.0/24 any -> 10.100.114.0 any
(msg:"ACK Check"; flags:A; sid:1000052;)
alert tcp !10.100.114.0/24 any -> any any (msg:"PSH
Check"; flags:0; sid:1000053;)
```

root@ss0:~

스노트 실행

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

```
[root@ss0 ~]#
[root@ss0 ~]# snort -A fast -b -i enp3s0 -u snort -g snort -c /etc/snort/snort.conf
Running in IDS mode
```

각각의 룰을 설정한 뒤 IDS(snort)를 실행

시스템 구축 및 운영(3)

root@ss0:/etc/sysconfig

방화벽 규칙

```
# sample configuration for iptables service
# you can edit this manually or use system-config-firewall
# please do not ask us to add additional ports/services to this default configuration
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -s 10.100.114.250 -m state --state NEW -j DROP
-A INPUT -s 10.100.114.82 -m state --state NEW -j DROP
-A INPUT -s 10.200.100.249 -m state --state NEW -j DROP
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
-A INPUT -s 10.100.114.93 -m state --state NEW -j DROP
-A INPUT -p icmp -j ACCEPT
```

root@ss0:/etc/sysconfig

로그 생성

```
25/16:20:38:34.369111 ** [1:500001:0] ICMP Test ** [Priority: 0] [ICMP] 10.200.100.249 -> 10.100.114.93
25/16:20:38:35.976645 ** [1:500001:0] ICMP Test ** [Priority: 0] [ICMP] 10.100.114.93 -> 10.200.100.249
25/16:20:38:50.976675 ** [1:500001:0] ICMP Test ** [Priority: 0] [ICMP] 10.200.100.249 -> 10.100.114.93
25/16:20:38:51.977410 ** [1:500001:0] ICMP Test ** [Priority: 0] [ICMP] 10.100.114.93 -> 10.200.100.249
25/16:20:38:51.977443 ** [1:500001:0] ICMP Test ** [Priority: 0] [ICMP] 10.200.100.249 -> 10.200.100.249
25/16:20:39:07.963051 ** [1:1000056:0] Land Attack ** [Priority: 0] [UDP] 0.0.0.0:68 -> 255.255.255.255:67
25/16:20:39:44.758431 ** [1:500001:0] ICMP Test ** [Priority: 0] [ICMP] 10.200.100.249 -> 10.100.114.93
25/16:20:39:49.523836 ** [1:500001:0] ICMP Test ** [Priority: 0] [ICMP] 10.200.100.249 -> 10.100.114.93
25/16:20:39:54.546327 ** [1:500001:0] ICMP Test ** [Priority: 0] [ICMP] 10.200.100.249 -> 10.100.114.93
25/16:20:39:55.633211 ** [1:500001:0] ICMP Test ** [Priority: 0] [IPv6-ICMP] fe80::fd11:7ae1:82a7:8538 -> ff02::1:ff
25/16:20:39:55.633246 ** [1:500001:0] ICMP Test ** [Priority: 0] [IPv6-ICMP] fe80::fd11:7ae1:82a7:8538 -> ff02::1:ff
25/16:20:39:55.633749 ** [1:500001:0] ICMP Test ** [Priority: 0] [IPv6-ICMP] fe80::5dc1:32d9:3a51:a339 -> ff02::1:ff
25/16:20:39:55.634049 ** [1:500001:0] ICMP Test ** [Priority: 0] [IPv6-ICMP] fe80::9da3:1285:c7a3:b597 -> ff02::1:ff
25/16:20:39:59.540706 ** [1:500001:0] ICMP Test ** [Priority: 0] [ICMP] 10.200.100.249 -> 10.100.114.93
25/16:20:40:10.902672 ** [1:500001:0] ICMP Test ** [Priority: 0] [ICMP] 10.200.100.249 -> 10.100.114.93
25/16:20:40:15.522387 ** [1:500001:0] ICMP Test ** [Priority: 0] [ICMP] 10.200.100.249 -> 10.100.114.93
25/16:20:40:20.546903 ** [1:500001:0] ICMP Test ** [Priority: 0] [ICMP] 10.200.100.249 -> 10.100.114.93
25/16:20:40:25.542399 ** [1:500001:0] ICMP Test ** [Priority: 0] [ICMP] 10.200.100.249 -> 10.100.114.93
25/16:20:40:42.898570 ** [1:500001:0] ICMP Test ** [Priority: 0] [IPv6-ICMP] fe80::a1cc:8038:18fd:5932 -> ff02::1:ff
```

10.200.100.249(공격)에서 10.100.114.93(IDS)으로 icmp 패킷 통신
하지만 방화벽 규칙에 자동으로 추가가 되면 패킷이 들어오는 로그만 탐지

시스템 구축 및 운영(4)

터미널 1

```

root@ss0/etc
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
con.weekly      jvm             protocols      udisk2
fontab          jvm-common     pulse          unbound
pyttab          kdump.conf     purple         updatedb.conf
sh.cshrc        kernel         python         usb_modeswitch.conf
sh.login        krtb5.conf     qemu-ga       usb_modeswitch.d
aps             ksmtuned.conf  qemu-kvm      vconsole.conf
sshhelpers      ld.so.cache    radvd.conf    vmirc
bus-1          ld.so.conf     rc.d          vmware-tools
conf            ld.so.conf.d   rc.local      wgetrc
dfault          libaudit.conf  rc1.d         wpa_supplicant
dmod.d          liblberverbs.d rc2.d         wvdial.conf
fcp             libnl          rc3.d         xdg
feyna-server-service.conf libreport      rc4.d         xinetd.conf
fmasq.conf      libuser.conf   rc5.d         xinetd.conf
fmasq.d         locale.conf    rc6.d         xal
racut.conf.d    localtime      rdma          yum
zirc            login.defs     redhat-release yum.conf
zfsck.conf      logrotate.conf request-key.conf yum.repos.d
nscript.cfg     logrotate.d
environment
root@ss0/etc]# attack!!
root@ss0/etc]# attack!!

```

터미널 2

```

ss0@localhost/lib
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
공게           문서          비디오
ss0@ss0 ~]$ cd ..
ss0@ss0 home]$ cd
ss0@ss0 ~]$ ls
snortrules-snapshot-2980.tar.gz 다운로드 바탕화면 사진 음악
ss0@ss0 ~]$ cd /
ss0@ss0 /]$ ls
backup boot etc lib media opt root skyn sys tmp var
bin dev home lib64 mnt proc run srv test usr
ss0@ss0 /]$ cd lib
ss0@ss0 lib]$ ls
alsa games jvm modules sysctl.d
binfmt.d gcc jvm-common modules-load.d systemd
app grub jvm-exports mozilla tmpfiles.d
erda java jvm-private polkit-1 tuned
cups java-1.5.0 lbd python2.7 udev
debug java-1.6.0 kdump rpm udisk2
dracut java-1.7.0 kernel sendmail x86_64-redhat-linux6E
firewalld java-1.8.0 locale sendmail postfix yum-plugins
firmware java-ext modprobe.d sse2
ss0@ss0 lib]$ attack!!
attack!!

```

터미널 3

```

root@ss0~
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
05/16-21:40:42.907401 [++] [1:500001:0] ICMP Test [++] [Priority: 0] [IPV6-ICMP] fe80::a1cc:a03e:f8fb:5832
-> ff02::16
05/16-21:40:44.666322 [++] [1:500001:0] ICMP Test [++] [Priority: 0] [IPV6-ICMP] fe80::a1cc:a03e:f8fb:5832
-> ff02::16
05/16-21:40:44.669212 [++] [1:500001:0] ICMP Test [++] [Priority: 0] [IPV6-ICMP] fe80::a1cc:a03e:f8fb:5832
-> ff02::16
05/16-21:40:44.670194 [++] [1:500001:0] ICMP Test [++] [Priority: 0] [IPV6-ICMP] fe80::a1cc:a03e:f8fb:5832
-> ff02::16
05/16-21:40:44.670204 [++] [1:500001:0] ICMP Test [++] [Priority: 0] [IPV6-ICMP] fe80::a1cc:a03e:f8fb:5832
-> ff02::16
05/16-21:40:44.909150 [++] [1:500001:0] ICMP Test [++] [Priority: 0] [IPV6-ICMP] fe80::a1cc:a03e:f8fb:5832
-> ff02::16
05/16-21:40:49.249265 [++] [1:500001:0] ICMP Test [++] [Priority: 0] [IPV6-ICMP] fe80::cc2a:53e4:199f:6cbb
-> ff02::1:ff8a:39d8
05/16-21:40:49.249812 [++] [1:500001:0] ICMP Test [++] [Priority: 0] [IPV6-ICMP] fe80::fd6e:7706:fe8a:39d8
-> ff02::1:ff9f:6cbb
05/16-21:41:23.172930 [++] [1:500001:0] ICMP Test [++] [Priority: 0] [IPV6-ICMP] fe80::1070:c8e2:fac3:950b
-> ff02::1:ff51:a339
05/16-21:41:23.173439 [++] [1:500001:0] ICMP Test [++] [Priority: 0] [IPV6-ICMP]
root@ss0 ~]# attack!!
attack!!

```

침입이 탐지가 된다면 작업중인 모든 터미널에 alert라는 경고문구를 확인 가능

시스템 구축 및 운영(5)

SSO(Single Sign On)

하나의 계정으로 한 번 로그인하여 각각의 시스템 또는 인터넷 서비스마다 인증 절차를 밟지 않고 사용할 수 있는 기술

SSO가 필요한 이유

기업 내 다양한 응용시스템 도입 및 운용에 따라 ID/PW 관리가 복잡해짐에 따라서 중앙집중적인 ID관리 및 시스템 권한관리를 통한 업무의 표준화

구성요소

SSO인증서버, Kerberos 서버, LDAP

장점

인증과 계정 정보의 중앙관리 용이	ID, 패스워드에 대한 보안위험 예방
인증관리비용의 절감	Password 문의 감소

단점

단일 ID/PW 노출 시 기업전체의 보안위험이 되므로 사용자의 철저한 관리 필요
--

시스템 구축 및 운영(6)

Client에서 SSO 서버로 로그인 후 생성된 티켓

```

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
[client2@client2 ~]$ ssh joongbu1@client2
Could not create directory '/home/client2/.ssh'.
The authenticity of host 'client2 (192.168.25.156)' can't be established.
ECDSA key fingerprint is 38:60:18:02:09:4f:17:9d:bf:a1:86:35:30:42:6d:a4.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/client2/.ssh/known_hosts).
joongbu1@client2's password:
Last login: Sat May 14 20:25:18 2016 from client2
[joongbu1@client2 ~]$ klist
Ticket cache: KEYRING:persistent:1003:krb_ccache_tE1YmOI
Default principal: joongbu1@OONGBU.KVM

Valid starting      Expires            Service principal
2016-05-14T20:50:37  2016-05-15T20:50:37  krbtgt/OONGBU.KVM@OONGBU.KVM
[joongbu1@client2 ~]$
    
```

SSO서버에서 client 인증 시 생성되는 로그 확인

```

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
5월 15 21:29:12 sso krb5kdc[5763](info): TGS_REQ (4 etypes {18 17 16 23}) 10.100.114.129: LOOKING_UP_SERVER: authtime 0, host/client1.joongbu.kvm@OONGBU.KVM for nfs/sso@OONGBU.KVM, Server not found in Kerberos database
5월 15 21:29:12 sso krb5kdc[5763](info): TGS_REQ (6 etypes {18 17 16 23 25 26}) 10.100.114.129: LOOKING_UP_SERVER: authtime 0, host/client1.joongbu.kvm@OONGBU.KVM for nfs/sso@OONGBU.KVM, Server not found in Kerberos database
5월 15 21:29:12 sso krb5kdc[5763](info): TGS_REQ (4 etypes {18 17 16 23}) 10.100.114.129: LOOKING_UP_SERVER: authtime 0, host/client1.joongbu.kvm@OONGBU.KVM for nfs/sso@OONGBU.KVM, Server not found in Kerberos database
    
```

root@sso:~

```

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
May 15 21:44:59 sso sshd[26985]: Connection closed by 10.100.114.129 [preauth]
May 15 21:45:12 sso sshd[27108]: Accepted password for root from 10.100.114.129 port 59828 ssh2
May 15 21:45:12 sso sshd[27108]: pam_unix(sshd:session): session opened for user root by (uid=0)
May 15 21:46:07 sso sshd[27108]: Received disconnect from 10.100.114.129: 11: disconnected by user
May 15 21:46:07 sso sshd[27108]: pam_unix(sshd:session): session closed for user root
May 15 21:46:35 sso sshd[27503]: Accepted password for root from 10.100.114.129 port 59829 ssh2
May 15 21:46:35 sso sshd[27503]: pam_unix(sshd:session): session opened for user root by (uid=0)
    
```

시스템 구축 및 운영(7)

vi /etc/xinetd.d/rsync 설정 파일

```

service rsync
{
    disable=no
    socket_type = stream
    wait = no
    user = root
    server = /usr/bin/rsync
    server_args = -daemon
    log_on_failure += USERD }
    
```

vi /etc/rsyncd.conf 설정 파일

```

[rsync]
path=/var/log/snort
comment = rsync
uid = root
gid = root
user chroot = yes
read only = yes
hosts allow = 10.100.114.93
max connections = 3
timeout = 600
    
```

crontab -e

```

* * * * * rsync -av -e ssh 10.100.114.93:/var/log/snort/day_log/home/admin
    
```

rsync를 사용하기 위한 설정파일 수정

시스템 구축 및 운영(8)

날짜 별로 기록된 로그파일

```

root@backup:/home/admin/day_log
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

[root@backup day_log]# ls
2016.05.12 2016.05.13 2016.05.14 2016.05.15 2016
[root@backup day_log]#

```

일별 보고서

```

=====
침입탐지보고서-2016.05.12 (목)
=====
Team : S-Guard
=====
공격방법   시도횟수   위험도   최초발생시간   공격방향
Land Attack 422       0        05/12-15:40:59 10.100.114.129:996 -> 10.100.114.93:2049
PSH Check   6511      0        05/12-15:40:47 10.100.114.129:996 -> 10.100.114.93:2049
brute ssh   68        0        05/12-15:41:04 10.100.114.93:22 -> 10.100.114.211:59642

```

Backup 된 침입탐지로그

```

root@backup:/home/admin/day_log/2016.05.12
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

[root@backup 2016.05.12]# ls
log2016.05.12 report2016.05.12
[root@backup 2016.05.12]#

```

분 단위 로그

```

root@backup:/home/admin/day_log/2016.05.12
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

05/12-15:41:04.589381 [**] [1:1000052:0] PSH Check [**] [Priority: 0] [TCP] 10.100.114.211:59642 -> 10.100.114.93:22
05/12-15:41:47.624878 [**] [1:1000052:0] PSH Check [**] [Priority: 0] [TCP] 10.100.114.129:996 -> 10.100.114.93:2049
05/12-15:41:47.625476 [**] [1:1000052:0] PSH Check [**] [Priority: 0] [TCP] 10.100.114.129:996 -> 10.100.114.93:2049
05/12-15:42:03.715735 [**] [1:1000052:0] PSH Check [**] [Priority: 0] [TCP] 10.100.114.211:59643 -> 10.100.114.93:22

```

침입탐지 서버에서 백업서버로 로그백업

시스템 구축 및 운영(9)

웹 상에서 관리계정으로 로그인

ftp://10.100.114.211 사용자명 및 암호 입력

사용자 이름:

암호:

매일 생성되는 침입탐지 로그

ftp://10.100.114.211/day_log/ 목록

상위 디렉터리로 이동

이름	크기	최종 수정일
2016.05.12		2016년 05월 12일 23시 58분 00초
2016.05.13		2016년 05월 13일 23시 58분 00초
2016.05.14		2016년 05월 14일 23시 58분 00초
2016.05.15		2016년 05월 15일 23시 58분 00초
2016.05.16		2016년 05월 16일 00시 00분 00초

침입탐지 보고서 / 분 단위 생성 로그

ftp://10.100.114.211/day_log/2016.05.12/ 목록

상위 디렉터리로 이동

이름	크기	최종 수정일
파일: log2016.05.12	803 KB	2016년 05월 12일 23시 59분 00초
파일: report2016.05.12	1 KB	2016년 05월 12일 23시 58분 00초

시스템 구축 및 운영(10)

평문으로 정보가 노출되는 FTP 패킷

ssl인증서 생성으로 접속정보가 암호화된 패킷

로그파일 전송 시 SSH로 전송하여 암호화되는 패킷

결론

▶ 방화벽과 IDS를 결합해 통합인증시스템의 취약점을 보완

1. 방화벽 구축을 하여 특정 포트 및 IP를 차단
2. IDS/IPS 를 내부망에 구현함으로써 자동적으로 로그생성 및 IP를 차단
3. 백업서버에 동기화하여 중요정보를 보관

▶ 기대효과

통합인증시스템을 보완하여 보다 안전하게 이용이 가능



