

SSO를 이용한 원격 서버관리 시스템

팀 명 : S.M (Server Manager)

지도 교수 : 양 환 석 교수님

팀 장 : 임현섭

팀 원 : 홍석현 장석재 남현정

2016. 05

중부대학교 정보보호학과

목 차

1. 서론	2
2. 관련연구	
2.1 SSO	2
2.2 LDAP	4
2.3 Kerberos	6
3. 본론	
3.1 SSO 구성 (LDAP + Kerberos)	8
3.1.1 Kerberos 설치 및 구현	
3.1.2 LDAP 서버 클라이언트 설치 및 설정	
3.2 Web 서버 구축 및 설정	16
3.2.1 DB 구축	
3.2.2 shellinabox 설치	
3.3 NoVNC 설치	25
3.4 Shell Script 구성	27
3.4.1 Main서버	
3.4.2 Second서버	
3.4.3 Third서버	
4. 결론	53
5. 참고 자료	53
6. 발표 자료	54

1. 서론

일반적인 서버관리는 관리자의 ID/PW를 가지고 접근하여 관리한다.

관리자가 여러 시스템을 관리하게 되는 경우 많은 ID/PW가 필요하게 됩니다.

또한 관리자의 관리 영역이 회사에서 상주 할 수 없기 때문에 즉각 대응할 수 없게 됩니다. 이것은 관리적인 효율이 떨어질 뿐만 아니라 늦은 대응으로 피해가 누적될 수 있습니다.

따라서 여러 대의 서버 시스템을 관리하기 위하여 SSO(Single-Sign-On) 통합인증 방식을 통하여 각각의 시스템에 로그인 정보를 저장하여 사용하는 방식이 아닌 통합적인 인증 정보로 시스템에 자동 로그인을 통하여 이전의 서버 관리의 불편함을 좀 더 손쉽게 관리하려는 목적이 있습니다.

보안이 필요한 환경에서 통합인증을 도입하는 경우 여러 응용 프로그램의 로그인 처리가 간소화되어 편리성을 도모할 수 있다.

그러므로 통합인증 방식을 사용하는 SSO를 통한 원격 관리 시스템을 구현하였습니다.

2. 관련연구

2.1 SSO

SSO란 Single Sign On의 약자로 여러 개의 사이트에서 한번의 로그인으로 여러 가지 다른 사이트들을 자동적으로 접속하여 이용하는 방법을 말합니다. 일반적으로 서로 다른 시스템, 서로 다른 사이트에서는 각각의 사용자 정보를 관리하게 됩니다. 하지만 필요에 의해서 각각의 사용자 정보를 연동하여 사용해야 할 경우가 생깁니다. 이때 하나의 사용자 정보를 기반으로 여러 시스템을 하나로 개발하기에는 어려움이 따르겠죠? 따라서 각각의 정보를 그대로 두고 통합인증을 사용하게 됩니다. 이때 각각의 시스템에 로그인할 때 통합인증 정보가 있는지 확인하고 통합인증정보가 있을 경우 타 시스템에서 자동으로 로그인 가능하도록 처리하고, 없을 때는 로그인 하면서 통합인증정보를 생성하여 다른 시스템에서 참조 가능하도록 하는 것입니다.

최근 회사들이 그룹화 되거나 대형화가 되어 여러 사이트들을 통합 관리하는 경우 SSO를 사용하게 됩니다. 이때 통합인증 SSO를 사용하게 되면, 관리자는 하나의 아이디로 모든 고객을 통합관리 할 수 있게 되기에 각각의 사이트 아이디를 관리할 필요가 없게 되고 기존 사용자는 정보변경 없이 하나의 사이트에 되어 있다면, 다

른 모든 사이트에 별도로 가입하지 않고 로그인 할 수 있게 되는 겁니다.

SSO(싱글사인온)의 도입효과로 관리의 투명성과 신뢰성을 높이고 비용을 절감하고 효율성을 높일 수 있습니다.

- 사용자 ID/Password 관리 효율 증가
- 별도 로그인 없이 다른 시스템 이용
- 윈도우 자격 증명과 같은 인증 지원
- 관리자의 로그인/ 종료/ 재접속을 위한 재입력 감소
- 사용자 접속정보에 대한 리포팅 기능 제공

2.2 LDAP (Lightweight Directory Access Protocol)

LDAP

경량 디렉터리 액세스 프로토콜(영어: Lightweight Directory Access Protocol; LDAP)은 TCP/IP 위에서 디렉터리 서비스를 조회하고 수정하는 응용 프로토콜이며 디렉터리는 논리, 계급 방식 속에서 조직화된 비슷한 특성을 가진 객체들의 모임이다. 많은 서버들 사이에 분포될 수 있으며 각 서버는 전체 디렉터리의 사본을 가질 수 있고 그 내용이 주기적으로 동기화 된다.

X.500을 근거로 한 디렉터리 데이터베이스에 접속하기 위한 통신 규약이다. 미국 미시간 대학에서 개발되었으며 디렉터리 정보의 등록, 갱신, 삭제와 검색 등을 실행할 수 있다. 운영 체제(OS)나 그룹웨어 제품들이 지원해 주고 있다. RFC 2251에 규정된 버전3이 최신판이며, 통신망을 이용한 이용자 메일 주소나 이용자의 정보를 검색하는 데 주로 사용된다. LDAP 서버에는 넷스케이프 디렉터리 서버와 같은 전용 서버 제품도 있다.

디렉토리 서비스

디렉토리는 데이터베이스와 유사하지만 더욱 설명적이고 속성에 기포한 정보를 갖고 있다. 디렉토리 내의 정보는 일반적으로 쓰기보다는 읽기 작업에 더욱 빈번히 이용된다.

따라서 디렉토리는 통상적으로 정규 데이터베이스들이 다량의 복잡한 갱신을 위해 사용하는 복잡한 처리(transaction) 또는 롤백 계획(프로그램에 따라 바로 전의 체크포인트로 돌아가기, roll-back)을 수행하지 않는다. 디렉토리는 다량의 색인(lookup) 또는 검색 연산에 대해 빠르게 응답하기 위해 조정된다. 디렉토리는 응답 시간을 감소시키는 반면에 가용성과 신뢰성을 증대시키기 위해 정보를 널리 복제하면 무방하다.

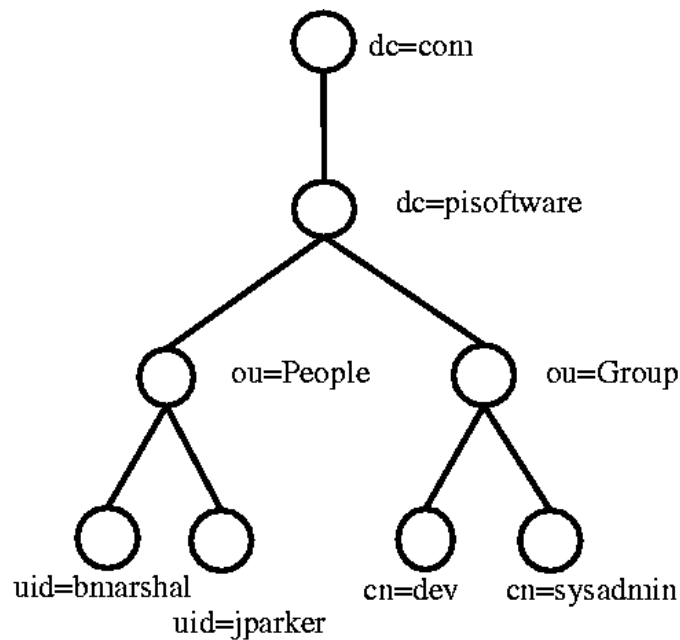
디렉토리 서비스를 제공하는 많은 다른 방법이 있다. 각각의 방법들은 다양한 종류의 정보가 디렉토리에 저장되는 것을 허용하며 정보가 어떻게 참조, 질의 및 갱신될 수 있는지 또는 허가받지 않은 액세스로부터 어떻게 보호되는지 등에 대한 여러 가지 요건을 둔다.

어떤 디렉토리 서비스는 제한된 상황(단독 머신에서 finger서비스 등)에 대해서 서비스를 제공해 지역적인 반면, 어떤 서비스는 더욱 넓은 상황에 대해서 서비스를 제공하여 전체적이다.

LDAP 디렉토리 서비스는 클라이언트-서버 모델에 기초하는데 하나 또는 그 이상의 LDAP 서버들이 LDAP 디렉토리 트리 또는 백엔드(backend) 데이터베이스를 구성하는 자료를 갖고 있다.

LDAP 클라이언트는 LDAP 서버에 연결해 질의하며, 서버가 응답 또는 클라이언트가 더 많은 정보를 얻을 수 있는 포인터(일반적으로 다른 LDAP서버)를 갖고 응답한다. 클라이언트는 어떤 LDAP 서버에 연결하던지 간에 동일한 디렉토리 구조를 본다. 한 LDAP 서버에 보내지는 이름은 다른 LDAP에 있을 수 있는 동일한 엔트리를 참조하며 이것이 LDAP와 같은 전체적인 디렉토리 서비스의 중요한 특징이다.

이 시스템에서 쓰인 LDAP Server는 계정들을 관리하는 디렉토리 서버에 해당하는 역할을 해준다.



[그림 2] LDAP 엔트리

[그림 2]는 LDAP에서의 간단한 디렉토리의 구조의 예이다.

이러한 LDAP 디렉토리 트리 구조를 특별히 DIT(Directory Information Tree)라고 부른다. LDAP에서 하나의 데이터를 나타낸다. dc: 도메인 컨트롤러(domain controller), ou: 조직편성(organization unit)

2.2 kerberos

Kerberos는 MIT의 Athena Project의 일환으로 개발된 인증 서비스 시스템으로 신뢰하는 제 3의 컴퓨터가 서비스를 이용하려는 클라이언트의 사용자를 인증함으로써 가능해진다. 인증을 통해 서버는 클라이언트의 사용자가 올바른 사용자 인지를 확인하고 비밀통신이 가능해진다.

종래의 공유 비밀 키 암호를 사용하여 신뢰할 수 있는 제 3자 인증 서비스로서 인증을 수행한다. 또한 호스트 오퍼레이팅 시스템의 인증에 의존하지 않고 호스트 주소에 대한 신뢰를 기초로 하지 않으며 네트워크 상의 모든 호스트의 실제 보안을 요구하지 않는다. 네트워크에 돌아다니는 패킷을 언제든지 읽고 수정하고 삽입할 수 있다는 가정 하에서 프린시펄의 ID를 검증할 수 있는 수단을 제공한다.

kerberos 서비스의 클라이언트는 주체로 식별된다. 주체는 KDC가 티켓을 지정할 수 있는 고유 ID이다. 영역은 도메인과 유사한 논리적 그룹, 동일한 마스터 KDC아래에 있는 시스템 그룹을 정의한다. 일부 영역은 계층형으로서 한 영역이 다른 영역의 슈퍼 세트이다. 또 다른 영역은 비계층형으로서 두 영역간의 매칭을 정의해야 한다.

Kerberos 서비스의 특징은 영역간 인증을 허용한다는 점이다. 이 Kerberos 기능을 영역간 인증이라고 한다. 각 영역에는 주체 데이터베이스의 마스터 복사본을 유지 관리 하는 서버가 있어야 한다. 이서버를 마스터 KDC서버라고 한다. 또한 각 영역에는 주체 데이터베이스의 복제 복사본을 포함하는 슬레이브 KDC서버도 한개이상 있어야 한다. 마스터 KDC서버와 슬레이브 KDC서버모두 인증을 설정하는데 사용되는 티켓을 만든다. 이 서버는 Kerberos화된 서비스에 대한 액세스를 제공한다.

"티켓"(ticket)을 기반으로 동작하는 컴퓨터 네트워크 인증 암호화 프로토콜로서 비 보안 네트워크에서 통신하는 노드가 보안 방식으로 다른 노드에 대해 식별할 수 있게 허용한다.

클라이언트 서버 모델을 목적으로 개발되었으며 사용자와 서버가 서로 식별할 수 있는 상호 인증(양방향 인증)을 제공한다. 커버로스 프로토콜의 메시지는 도청과 재전송 공격으로부터 보호되며 대칭 키 암호로 빌드되고 TTP(신뢰된 서드 파티)를 요구한다.

또한 특정 인증 구간에서 비대칭 키 암호 방식을 이용함으로써 선택적으로 공개 키 암호 방식을 사용할 수 있다.

3. 본론



[그림 1] SSO(single-sign-on) 구현 시스템 구성도

3.1 SSO 구성 (LDAP + Kerberos) 구축환경 : CentOS 7

3.1.1 Kerberos 설치 및 구현

(Main)

명령어 yum을 이용하여 커버로스 설치

```
yum -y install Krb5-server
```

vi /etc/krb5.conf에 들어가서 주석 제거 후 원하는 SSO 도메인 네임 변경

```
[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 = SSO.COM
default_ccache_name = KEYRING:persistent:%{uid}

[realms]
SSO.COM = {
  kdc = main.sso.com
  admin_server = main.sso.com
}

[domain_realm]
.sso.com = SSO.COM
sso.com = SSO.COM
```

vi /var/kerberos/krb5kdc/kdc.conf 입력 후

원하는 도메인 이름으로 변경 후 저장

```
[kdcdefaults]
kdc_ports = 88
kdc_tcp_ports = 88

[realms]
SSO.COM = {
  #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_enctypes = 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 입력 후

원하는 도메인 이름으로 변경 후 저장

```
/admin@SSO.COM *
```


`kdb5_util create -s -r 원하는 도메인 이름 ` 입력후 사용할 비밀번호 입력

```
[root@main ~]# kdb5_util create -s -r ***
```

<방화벽 및 데몬 시작>

systemctl enable krb5kdc / systemctl start krb5kdc

systemctl enable kadmind / systemctl start kadmind

firewall-cmd --permanent --add-service=kerberos

firewall-cmd --reload

<kadmind을 이용한 계정 설정>

kadmind.local

addprinc root/admin

패스워드 입력 생성완료

<계정 PW 키 생성>

addprinc -randkey host/2번째 서버계정.도메인.com

addprinc -randkey host/3번째 서버계정.도메인.com

<계정 정보를 /tmp 저장>

ktadd -k /tmp/2번째 계정.keytab host/2번째계정.도메인.com

ktadd -k /tmp/3번째 계정.keytab host/3번째계정.도메인.com

<계정 정보 SSO 등록할 원격 서버로 전달> - 미리 /etc/hosts 에 등록

scp /etc/krb5.conf /tmp/2번째 계정.keytab 2번째 계정:/tmp/
root PW 입력

scp /etc/krb5.conf /tmp/3번째 계정.keytab 3번째 계정:/tmp/
root PW 입력

(Second)

ls /tmp/ 파일확인

yum -y install pam_krb5 krb5-workstation 설치 -> 커버로스 클라이언트 설치

Main에서 보낸 파일 /etc로 교체

Wcp /tmp/krb5.conf /etc/

ktutil 명령어 이용하여 커버로스에 정보 등록

rkt /tmp/2번째 서버계정.keytab wkt /etc/krb5.keytab quit

(Third)

ls /tmp/ 파일확인

yum -y install pam_krb5 krb5-workstation 설치 -> 커버로스 클라이언트 설치

Main에서 보낸 파일 /etc로 교체

Wcp /tmp/krb5.conf /etc/

ktutil 명령어 이용하여 커버로스에 정보 등록

rkt /tmp/2번째 서버계정.keytab wkt /etc/krb5.keytab quit

3.1.2 LDAP 서버 클라이언트 설치 및 설정

(main)

LDAP 서버 및 클라이언트 migrationtools 설치

yum -y install openldap-servers openldap-clients migrationtools

<디렉토리 DB 파일 복사>

cp /usr/share/openldap-servers/DB_CONFIG.example /var/lib/ldap/DB_CONFIG

<소유권 변경>

chown -R ldap. /var/lib/ldap

<slapd PW 설정>

slappasswd 입력후 비밀번호 설정 한 값 복사

cd /etc/openldap/slapd.d/cnW=config

vi olcDatabaseW=W{0W}config.ldif에서 마지막줄 olcRootPW : 복사한 값 추가

```
# AUTO-GENERATED FILE - DO NOT EDIT!! Use ldapmodify.
# CRC32 aed629cd
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: 631a3072-9f1a-1035-87d6-fd64b08a634b
creatorsName: cn=config
createTimestamp: 20160425101532Z
entryCSN: 20160425101532.129155Z#000000#000#000000
modifiersName: cn=config
modifyTimestamp: 20160425101532Z
olcRootPW: {SSHA}Ge5JYM9/OGknMJEaXu8gD0zvliOItoHUT
```

vi olcDatabaseW=W{2W}hdb.ldif에서 기본 셋팅된 dc 값 자신값으로 변경

olcRootDN 아래 olcRootPW 복사한 값 추가

마지막 줄에 olcAccess: {0}to attrs=userPassword by self write

by dn.base="cn=Manager, dc=원하는 도메인 ,dc=com" write by anonymous auth
by * none

olcAccess: {1}to * by dn.base="Manager, dc=원하는 도메인 ,dc=com" write by
self write by * read

```
! AUTO-GENERATED FILE - DO NOT EDIT!! Use ldapmodify.
# CRC32 60781b06
dn: olcDatabase={2}hdb
objectClass: olcDatabaseConfig
objectClass: olcHdbConfig
olcDatabase: {2}hdb
olcDbDirectory: /var/lib/ldap
olcSuffix: dc=sso,dc=com
olcRootDN: cn=Manager,dc=sso,dc=com
olcRootPW: {SSHA}Ge5JYM9/OGknMJEaXu8gD0zvliOIohUT
olcDbIndex: objectClass eq,pres
olcDbIndex: ou,cn,mail,surname,givenname eq,pres,sub
structuralObjectClass: olcHdbConfig
entryUUID: 631dedde-9f1a-1035-87d8-fd64b08a634b
creatorsName: cn=config
createTimestamp: 20160425101532Z
entryCSN: 20160425101532.153664z#000000#000#000000
modifiersName: cn=config
modifyTimestamp: 20160425101532Z
olcAccess: {0}to attrs=userPassword by self write by dn.base="cn=Manager,dc=sso,dc=com" write by anonymous
auth by * none
olcAccess: {1}to * by dn.base="cn=Manager,dc=sso,dc=com" write by self by write by * read
```

vi olcDatabaseW=W{1W}monitor.ldif 원하는 도메인 dc 값 변경

```
! AUTO-GENERATED FILE - DO NOT EDIT!! Use ldapmodify.
# CRC32 2965b062
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=sso,dc=com" read by * none
structuralObjectClass: olcDatabaseConfig
entryUUID: 631de9ec-9f1a-1035-87d7-fd64b08a634b
creatorsName: cn=config
createTimestamp: 20160425101532Z
entryCSN: 20160425101532.153563z#000000#000#000000
modifiersName: cn=config
modifyTimestamp: 20160425101532Z
```

<데몬 실행 및 방화벽 설정>

systemctl enable slapd / systemctl start slapd / netstat -nltp 확인

firewall-cmd --permanent --add-service=ldap

firewall-cmd reload

ls -l /etc/openldap/schema 확인

<ldap에 설정파일 등록>

ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/cosine.ldif

ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/nis.ldif

ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/inetorgperson.ldif

```
base.ldif 생성
dn: dc=도메인,dc=도메인
objectClass: dcObject
objectClass: organizationalUnit
ou: 도메인
o : dc의 도메인

dn: ou=People dc=도메인,dc=도메인
objectClass: organizationalUnit
ou: People

dn: ou=Group dc=도메인,dc=도메인
objectClass: organizationalUnit
ou: Group
```

<LDAP에 등록하기>

```
ldapadd -x -D cn=Manager,dc=도메인,dc=도메인 -W -f base.ldif
```

<LDAP 확인하기>

```
ldapsearch -x -D cn=Manager,dc=도메인,dc=도메인 -W -b dc=도메인,dc=도메인
```

<LDAP에서 사용할 사용자 생성하기>

```
cd /usr/share/migrationtools/
```

```
vi migrate_common.ph에서 DEFAULT 값 변경 하고 스키마 값 1로 변경
```

```
Default DNS domain
DEFAULT_MAIL_DOMAIN = "sso.com";

Default base
DEFAULT_BASE = "dc=sso,dc=com";

Turn this on for inetLocalMailReceipient
sendmail support; add the following to
sendmail.mc (thanks to Petr@Kristof.CZ):

$EXTENDED_SCHEMA = 1;
```

```
grep 추가한 사용자 /etc/passwd > /tmp/users
```

```
grep 추가한 사용자 /etc/group > /tmp/groups
```

<LDAP에 맞는 파일로 설정 변경>

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

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

<LDAP에 사용자 추가>

```
ldapadd -x -D cn=Manager,dc=도메인,dc=도메인 -W -f /tmp/groups.ldif
```

```
ldapadd -x -D cn=Manager,dc=도메인,dc=도메인 -W -f /tmp/users.ldif
```

```
objectClass: organizationalUnit
ou: People

# Group, sso.com
dn: ou=Group,dc=sso,dc=com
objectClass: organizationalUnit
ou: Group

# kkk1, Group, sso.com
dn: cn=kkk1,ou=Group,dc=sso,dc=com
objectClass: posixGroup
objectClass: top
cn: kkk1
userPassword:: e2NyeXB0fXg=
gidNumber: 1001

# kkk2, Group, sso.com
dn: cn=kkk2,ou=Group,dc=sso,dc=com
objectClass: posixGroup
objectClass: top
cn: kkk2
userPassword:: e2NyeXB0fXg=
gidNumber: 1002

# kkk1, People, sso.com
dn: uid=kkk1,ou=People,dc=sso,dc=com
uid: kkk1
cn: kkk1
sn: kkk1
mail: kkk1@sso.com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword:: e2NyeXB0fSEh
shadowLastChange: 16916
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 1001
gidNumber: 1001
homeDirectory: /home/kkk1
```

<Main에 nfs 유틸 설치>

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

```
vi /etc/exports --> /home *(rw,sync)
```

```
systemctl enable rpcbind / systemctl start rpcbind
```

```
systemctl enable nfs-server / systemctl nfs-server
```

```
firewall-cmd --permanent --add-service nfs
```

```
firewall-cmd --reload
```

이후 showmount -e 확인

```
[root@main migrationtools]# showmount -e
Export list for main:
/home *
```

(Second / Third)

yum -y install nss-pam-ldapd

authconfig-tui 설정 ldap 체크 커버로트 체크 ldap의 서버 계정이름 입력후 종료

설정 완료후

vi /etc/nsswitch.conf 파일에서 passwd, group에 ldap 등록 확인후

getent passwd 등록한 계정 확인 (사용자 정보 가져오기)

```
[root@second ~]# getent passwd kkk1
kkk1:x:1001:1001:kkk1:/home/kkk1:/bin/bash
[root@second ~]# id kkk1
uid=1001(kkk1) gid=1001(kkk1) groups=1001(kkk1)
```

이후 vi /etc/auto.master

마지막 줄에 /home /etc/auto.autofs --timeout=600

vi /etc/auto.autofs 에 들어가서

* ldap서버계정:/home/&

systemctl enable autofs / systemctl start autofs

vi /etc/ssh/ssh_config 들어가서 값 변경

```
# Site-wide defaults for some commonly used options.
# list of available options, their meanings and
# ssh_config(5) man page.

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
```

vi /etc/ssh/sshd_config 값 변경


```
# GSSAPI options
GSSAPIAuthentication yes
GSSAPICleanupCredentials yes
#GSSAPIStrictAcceptorCheck yes
#GSSAPIKeyExchange no
#GSSAPIEnablek5users no
```

systemctl reload sshd --> sshd 데몬 재실행

LDAP 서버에서 다시 kadmin.local로 사용자 다시 등록

kadmin.local

addprinc user1

ldap서버 패스워드 입력

결과 확인

second / third 서버에서 접속 확인

```
[root@second ~]# ssh kkk1@second
kkk1@second's password:
Last login: Sat May 14 18:51:34 2016 from main
[kkk1@second ~]$
```

위와 같이 접속이 되면 LDAP과 Kerberos 인증 성공

또한 klist 명령어로 커버로스 티켓팅 확인

```
[kkk1@second ~]$ klist
Ticket cache: KEYRING:persistent:1001:krb_ccache_CpQRDNJ
Default principal: kkk1@SSO.COM

Valid starting          Expires                Service principal
2016-05-14T20:41:36    2016-05-15T20:41:36  krbtgt/SSO.COM@SSO.COM
```

<Main에서의 확인>

```
[root@main home]# pwd
/home
[root@main home]# ls
kkk1 kkk2 main test1
[root@main home]#
```

Main의 /home 디렉토리에 kkk1 kkk2 디렉토리가 생성

kkk1 <-> second

kkk2 <-> third

각각의 디렉토리로 접근하여 이용 가능

3.2 Web 서버 구축 및 설정

yum -y install httpd 설치
systemctl enable httpd / systemctl start httpd

php를 사용하기 위해서 php 설치
yum -y install php php-mysql

또한 보안을 강화 하기 위하여 https 추가로 설정
vi /etc/httpd/conf/httpd.conf



```
root@localhost:~#  
#  
#Listen 12.34.56.78:80  
Listen 80  
Listen 8080  
#  
# Dynamic Shared Object (DSO) Support  
#  
# To be able to use the functionality of a module which was built as a DSO you  
# have to place corresponding 'LoadModule' lines at this location so the  
# directives contained in it are actually available _before_ they are used.  
# Statically compiled modules (those listed by 'httpd -l') do not need  
# to be loaded here.  
#  
# Example:  
# LoadModule foo_module modules/mod_foo.so  
#  
# Include conf.modules.d/*.conf  
#  
# Virtual hosts  
# Include conf/extra/httpd-vhosts.conf  
#  
-- INSERT --
```

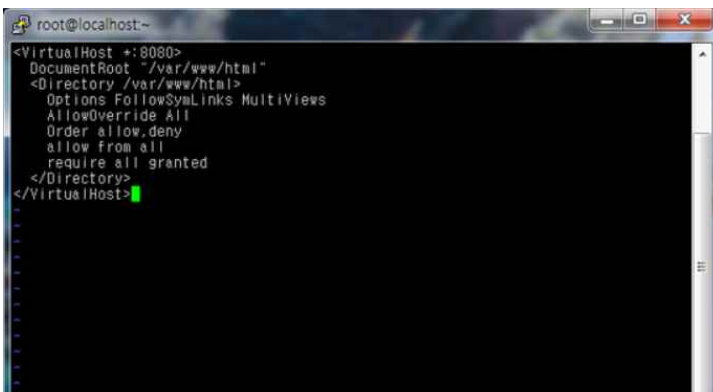
- Listen 8080

httpd.conf에 Listen 부분을 찾아서 8080포트/8443포트 추가

httpd.conf 마지막부분에 추가

Virtual hosts

Include conf/extra/httpd-vhosts.conf



```
root@localhost:~#  
<VirtualHost *:8080>  
  DocumentRoot "/var/www/html"  
  <Directory /var/www/html>  
    Options FollowSymLinks MultiViews  
    AllowOverride All  
    Order allow,deny  
    allow from all  
    require all granted  
  </Directory>  
</VirtualHost>  
#
```

mkdir /etc/httpd/conf/extra extra 폴더 생성

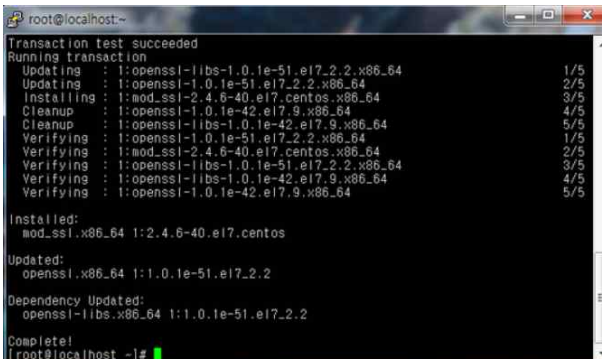
vi /etc/httpd/conf/extra/httpd-vhosts.conf

httpd-vhosts.conf 생성 후 아래 내용을 추가

```
<VirtualHost *:8080>
    DocumentRoot "/var/www/html"
    <Directory /var/www/html>
        Options FollowSymLinks MultiViews
        AllowOverride All
        Order allow,deny
        allow from all
        require all granted
    </Directory>
</VirtualHost>

<VirtualHost *:8443>
    DocumentRoot "/var/www/html"
    SSLEngine on
    SSLCertificateFile /etc/httpd/ssl/server.crt
    SSLCertificateKeyFile /etc/httpd/ssl/server.key
    <Directory /var/www/html>
        Options FollowSymLinks MultiViews
        AllowOverride All
        Order allow,deny
        allow from all
        require all granted
    </Directory>
</VirtualHost>
```

<SSL 설치>

A terminal window titled 'root@localhost:~' showing the output of the command 'yum install mod_ssl openssl'. The output displays the transaction test success, the running transaction, and the progress of installing and updating packages. The packages listed are mod_ssl-2.4.6-40.el7.centos.x86_64 and openssl-1.0.1e-51.el7.2.2.x86_64. The transaction is completed successfully.

```
Transaction test succeeded
Running transaction
  Updating : 1:openssl-libs-1.0.1e-51.el7.2.2.x86_64 1/5
  Updating : 1:openssl-1.0.1e-51.el7.2.2.x86_64 2/5
  Installing : 1:mod_ssl-2.4.6-40.el7.centos.x86_64 3/5
  Cleanup : 1:openssl-1.0.1e-42.el7.9.x86_64 4/5
  Cleanup : 1:openssl-libs-1.0.1e-42.el7.9.x86_64 5/5
  Verifying : 1:openssl-1.0.1e-51.el7.2.2.x86_64 1/5
  Verifying : 1:mod_ssl-2.4.6-40.el7.centos.x86_64 2/5
  Verifying : 1:openssl-libs-1.0.1e-51.el7.2.2.x86_64 3/5
  Verifying : 1:openssl-libs-1.0.1e-42.el7.9.x86_64 4/5
  Verifying : 1:openssl-1.0.1e-42.el7.9.x86_64 5/5

Installed:
  mod_ssl.x86_64 1:2.4.6-40.el7.centos

Updated:
  openssl.x86_64 1:1.0.1e-51.el7.2.2

Dependency Updated:
  openssl-libs.x86_64 1:1.0.1e-51.el7.2.2

Complete!
[root@localhost ~]#
```


yum install mod_ssl openssl

yum으로 openssl 설치

mkdir /etc/httpd/ssl; cd /etc/httpd/ssl

httpd폴더에 ssl폴더를 생성하고 해당위치로 이동

<인증서 생성>



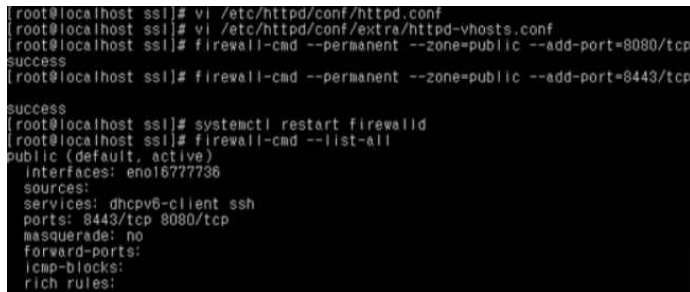
```
root@localhost:/etc/httpd/ssl# openssl genrsa -out server.key 2048
root@localhost/ssl# openssl req -new -key server.key -out server.csr
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
[root@localhost/ssl]# openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
Getting Private key
[root@localhost/ssl]#
```

openssl genrsa -out server.key 2048

openssl req -new -key server.key -out server.csr

openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt

<방화벽 등록 및 데몬 실행>



```
[root@localhost/ssl]# vi /etc/httpd/conf/httpd.conf
[root@localhost/ssl]# vi /etc/httpd/conf/extra/httpd-vhosts.conf
[root@localhost/ssl]# firewall-cmd --permanent --zone=public --add-port=8080/tcp
success
[root@localhost/ssl]# firewall-cmd --permanent --zone=public --add-port=8443/tcp
success
[root@localhost/ssl]# systemctl restart firewall
[root@localhost/ssl]# firewall-cmd --list-all
public (default, active)
  interfaces: eno16777736
  sources:
  services: dhcpv6-client ssh
  ports: 8443/tcp 8080/tcp
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

firewall-cmd --permanent --zone=public --add-port=8080/tcp

firewall-cmd --permanent --zone=public --add-port=8443/tcp

방화벽에 8080포트, 8443포트를 추가해준다.

systemctl restart firewall

방화벽 재부팅

firewall-cmd --list-all

방화벽 정책 확인

systemctl restart httpd

3.2.1 DB 구축

Maria_DB 사용

```
yum -y install mariadb-server mariadb  
systemctl start mariadb / systemctl enable mariadb
```

설치 완료

<DB 구성>

create database

```
MariaDB [(none)]> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| member |  
| mysql |  
| performance_schema |  
+-----+  
4 rows in set (0.06 sec)
```

create tables; (테이블 sql 구문)

```
create table users (idx int not null auto_increment primary key, id char(20),name  
char(20),pw char(20));
```

```
MariaDB [member]> show tables;  
+-----+  
| Tables_in_member |  
+-----+  
| users |  
+-----+  
1 row in set (0.00 sec)
```

desc users;

```
MariaDB [member]> desc users;  
+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+  
| idx | int(11) | NO | PRI | NULL | auto_increment |  
| id | char(20) | YES | | NULL | |  
| name | char(20) | YES | | NULL | |  
| pw | char(20) | YES | | NULL | |  
+-----+  
4 rows in set (0.07 sec)
```

select * from users;

```
MariaDB [member]> select * from users;  
+-----+  
| idx | id | name | pw |  
+-----+  
| 1 | admin | ADMINISTRATOR | |  
+-----+  
1 row in set (0.00 sec)
```

<WEB site 화면>



← → ↻ 🏠 | 인증서 오류 10.100.114.72 | ☆ | ≡ | 📄 | 🔍 | ⌵

☆ How to Install [PHP DB] MySQL 마녀를 사랑한 남이 Android SDK 기 1FreeHosting.com LINE Plus

관리자 Login

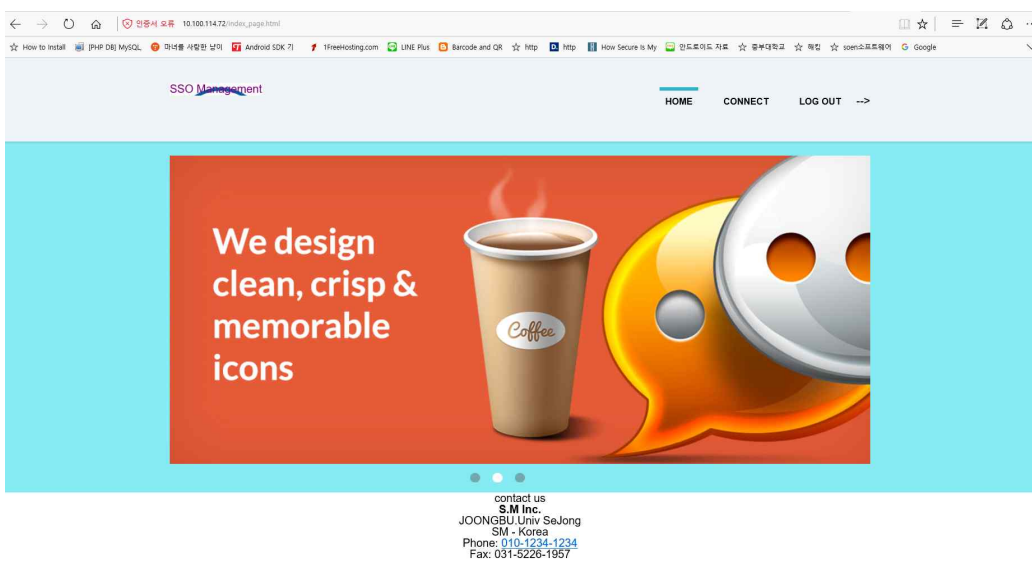
ID :

PW :

Login

© Copyright 2016 - JoongBu.Univ ?>

<로그인 화면>



<main 화면>

<Web site 소스>

(index.php)

<?php?>

<div >

<center><h1>관리자 Login</h1></center>

<hr></hr>

</div>

```

<center><br><br><br>
<form name="login_form" action="login_check.php" method="post">
  ID : <input type="text" name="id" width='10px'> <br><br>
  PW : <input type="password" name="pw"> <br><br>
  <input type="submit" name="login" value="Login">
</form>
</center>
<p id="copyright" align="center"> &copy; Copyright 2016 - JoongBu.Univ ?>

```

```

<index_page.html>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>SSO Management</title>
  <link rel="stylesheet" type="text/css" href="stylesheets/reset.css" />
  <link rel="stylesheet" type="text/css" href="stylesheets/main.css" />
</head>
<body>
  <div id="header">
    <div class="container">
      <h1><a href="index_page.html">SSO Management</a></h1>
      <div id="main_menu">
        <ul>
          <li class="first_list"><a href="index_page.html"
class="main_menu_first main_current">home</a></li>
          <li class="first_list">
            <a href="https://10.100.114.72:4200"
class="main_menu_first">Connect</a>
          </li>
          <li class="first_list"><a href="logout.php" class="main_menu_first"
onclick="popup()">Log out</a></li>
        </ul>
      </div> <!-- END #main_menu -->
    </div> <!-- END .container -->
  </div> <!-- END #header -->

```

```

<div id="main_content">
  <div id="slideshow_area">
    <div class="container">
      <div id="slideshow_container">
        <ul>
          <li></li>
        </ul>
        <div id="slideshow_pagination">
          <ul>
            <li><a href="#"></a></li>
            <li><a href="#" class="current"></a></li>
            <li><a href="#"></a></li>
          </ul>
        </div> <!-- END #slideshow_pagination -->
      </div> <!-- END #slideshow_container -->
    </div> <!-- END .container -->
  </div> <!-- END #slideshow_area -->
  <div class="container" align="center">

    <h2>contact us</h2>
    <p><span class="bold_text">S.M Inc.</span>
    <br />
    JOONGBU.Univ SeJong
    <br />
    SM - Korea
    <br />
    Phone: 010-1234-1234
    <br />
    Fax: 031-5226-1957</p>
    <center><h1>&copy;Copyright2016-JoongBu.Univ </h1>
    </center>

    <br />
  </div> <!-- END #footer_about -->
  <div id="footer_connect" class="footer_info" align="center">
    <h4>connect with us</h4>

    <ul>

```

```

        <li> <ahref="#"id="facebook"
                    title="Facebook">Facebook</a> </li>
        <li> <ahref="#"id="linkedin"
                    title="LinkedIn">LinkedIn</a> </li>
        <li> <a href="#" id="skype" title="Skype">Skype</a> </li>
    </ul>

```

```

    </div> <!-- END #footer -->
</body>
</html>

```

```

<login_check>
<?php

```

```

include ("connect.php");

```

```

$id = $_POST['id'];
$pw = $_POST['pw'];

```

```

$query = "select * from users where id ='$id' and pw ='$pw'";
$result=mysqli_query($con, $query);
$row = mysqli_fetch_array($result);

```

```

if($id==$row['id']&& $pw==$row['pw'] && $id!=""&& $pw!=""){
    echo "<script>location.href='index_page.html';</script>";
}
else{
    echo "<script>window.alert('아이디와 비밀번호를 확인해주시오.');"</script>";
    echo "<script>location.href='index.php';</script>";
}
?>

```

```

<logout.php>
<?php

```

```

if($_SESSION['id']!=null){
    session_destroy();

```

```

}
echo "<script>window.alert('로그아웃되었습니다.');

```

3.2.2 shellinabox 설치

```

yum -y install epel-release shellinabox
vi /etc/sysconfig/shellinaboxd

```

```

# Shell in a box daemon configuration
# For details see shellinaboxd man page

# Basic options
USER=shellinabox
GROUP=shellinabox
CERTDIR=/var/lib/shellinabox
PORT=4200
#OPTS="--disable-ssl-menu -s /:LOGIN"
OPTS=" -s /:SSH"

# Additional examples with custom options:

# Fancy configuration with right-click menu choice for black-on-white:
# OPTS="--user-css Normal:#black-on-white.css,Reverse:-white-on-black.css --disabl
e-ssl-menu -s /:LOGIN"

# Simple configuration for running it as an SSH console with SSL disabled:
# OPTS="-t -s /:SSH:host.example.com"

```

<데몬 시작>

```
systemctl restart shellinaboxd / systemctl enable shellinaboxd
```


3.3 NoVNC 설치

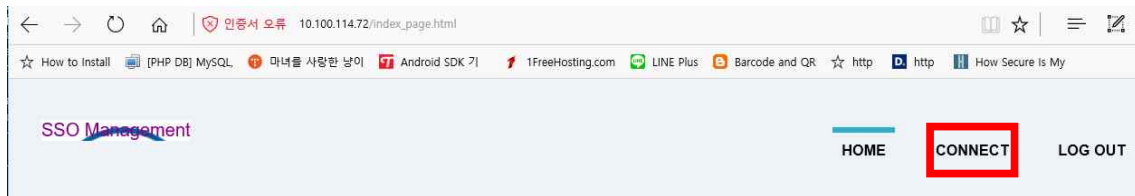
wget <http://github.com/kanaka/noVNC/zipball/master>

unzip master

cd master

yum -y groupinstall "GNOME Desktop"

vncserver :1 이후 pw 입력



connect 클릭

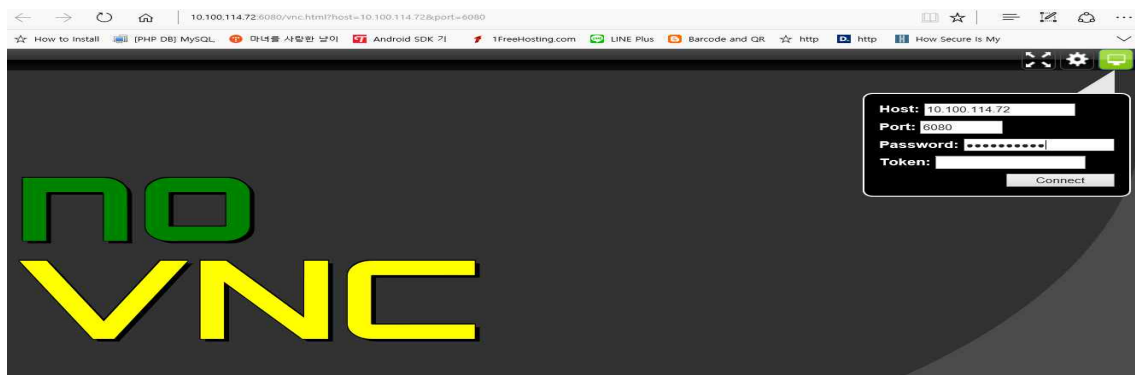
```
[root@main ~]# ./start.sh
Using installed websockify at /usr/bin/websockify
Starting webserver and WebSockets proxy on port 6080
WebSocket server settings:
- Listen on :6080
- Flash security policy server
- Web server. Web root: /root/noVNC
- SSL/TLS support
- proxying from :6080 to 10.100.114.72:5901
```

Navigate to this URL:

<http://10.100.114.72:6080/vnc.html?host=10.100.114.72&port=6080>

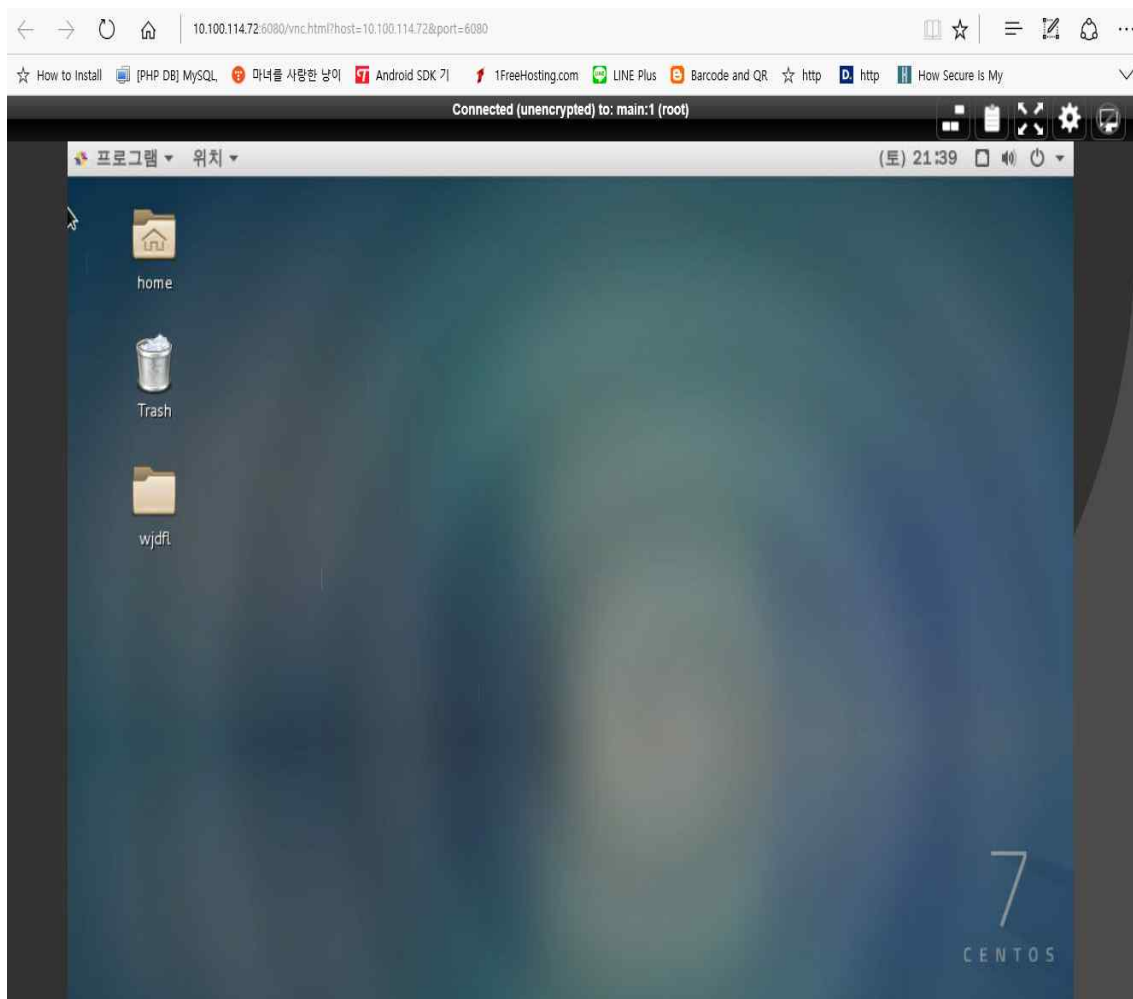
Press Ctrl-C to exit

주소 클릭



PW 입력후 SSO LDAP 서버로 접속

NoVNC 접속 화면



3.4 Shell Script 구성

3.4.1 Main서버

(main.sh)

```

=====
SSO MANAGEMENT MENU
=====
|          1.mornitoring
|          2.control
|          3.exit
=====

```

(main 화면)

(main.sh shell)

```
#!/bin/bash
clear

{= `tty | cut -f 4 -d "`'`
}= `expr $K + 1`

echo "
echo "                                "
echo "                                SSO MANAGEMENT MENU                                "
echo "===== "
echo " |                               1.mornitoring                               | "
echo " |                               2.control                                   | "
echo " |                               3.exit                                     | "
echo "===== "
```

```
    echo -n "SELECT MENU : "
read num

case $num in

1)

gnome-terminal
sleep 0.3
sh /home/test1/SSO/monitor.sh &
sh /home/test1/SSO/main.sh >> /dev/pts/$B ;;

2)

sh /home/test1/SSO/control.sh ;;

3)

exit;;

esac
```

(mornitor.sh)

SYSTEM MONITORING LIST

	main	ser1	ser2
1) cpu	17.3%	0.9%	6.3%
2) mem	41.62%	17.92%	30.07%
3) traffic	749 KiB/s	129 KiB/s	140 KiB/s

(mornitor.sh shell)

(mornitorsh 는 sso, kkk1, kkk2에 기록되어져있는 cpu.mem.traffic.txt파일을 수시로 가져와 모니터링 쉘 안에 띄우고 모든 값의 사용량이 40이 넘을시에 등록되어있는 bird266@naver.com(관리자 이메일) 로 경고메일을 보내는 기능을 한다.)

```
#!/bin/bash
clear
alert_main(){
    cpu=`tail -1 /home/test1/SS0/cpu_status.txt`
    if [ $(echo "$cpu > 40" | bc) -eq 1 ]; then
        notify-send "SERVER_MAIN Alert" "please lower CPU usage"
        echo "System cpu checking!!!!" | mail -s "SystemAlert" bird266@naver.com
    fi
}
alert_second(){
    cpu_2=`tail -1 /home/kkk1/cpu_status2.txt`
    if [ $(echo "$cpu_2 > 40" | bc) -eq 1 ]; then
        notify-send "SERVER2 Alert" "please lower CPU usage"
        echo "System cpu checking!!!!" | mail -s "SystemAlert" bird266@naver.com
    fi
}
alert_third(){
    cpu_3=`tail -1 /home/kkk2/cpu_status3.txt`
    if [ $(echo "$cpu_3 > 40" | bc) -eq 1 ]; then
        notify-send "SERVER3 Alert" "please lower CPU usage"
        echo "System cpu checking!!!!" | mail -s "SystemAlert" bird266@naver.com
    fi
}
cpu_main() {
    c_1=`tail -1 /home/test1/SS0/cpu_status.txt`
}
cpu_second(){
    c_2=`tail -1 /home/kkk1/cpu_status2.txt`
}
cpu_third(){
    c_3=`tail -1 /home/kkk2/cpu_status3.txt`
}
mem_main(){
    m_1=`tail -1 /home/test1/SS0/mem_status.txt`
}
mem_second(){
    m_2=`tail -1 /home/kkk1/memory2.txt`
}
mem_third(){
    m_3=`tail -1 /home/kkk2/memory3.txt`
}
trattic_main() {
    t_1=`tail -1 /home/test1/SS0/rx3.txt`
}
trattic_second() {
    t_2=`tail -1 /home/kkk1/rx3.txt`
}
trattic_third() {
```

```

}
while [ : ]
do
sleep 1
clear
cpu_main
cpu_second
cpu_third
mem_main
mem_second
mem_third
trattic_main
trattic_second
trattic_third
alert_main
alert_second
alert_third
echo "
echo "
echo "
echo "
echo "
echo "
echo "
echo "
echo "
echo "
echo "
echo "
echo "
echo "
echo "
echo "
echo "
done

```

SYSTEM MONITORING LIST				
	main	ser1	ser2	
1)cpu	\$c_1%	\$c_2%	\$c_3%	"
2)mem	\$m_1	\$m_2	\$m_3	"
3)traffic	\$t_1 \$t_2 \$t_3			"

(cpu.sh)

(main 의 cpu값을 txt파일안에 실시간으로 기록해 놓는다)

```
18.0  
17.9  
18.6  
17.9  
17.4  
17.3  
18.1  
18.0  
17.5  
17.3  
17.0  
17.1  
17.4  
18.5  
17.9  
17.1  
17.2  
17.5
```

(cpu.sh shell)

```
#!/bin/bash
```

```
while [ : ]
```

```
do
```

```
#A="`tail -1 out.txt`"
```

```
tail -1 out.txt | cut -f 3 -d " " | cut -f 1 -d "%" >>  
/home/test1/SSO/cpu_status.txt
```

```
sleep 3
```

```
done
```

(cpu.pl)

```
2016-05-12 21:55:46 32.6%us 38.7%sy 28.7%id
2016-05-12 22:02:15 32.7%us 37.7%sy 29.6%id
2016-05-12 22:02:16 34.7%us 38.8%sy 26.5%id
2016-05-12 22:02:17 33.5%us 39.1%sy 27.5%id
2016-05-12 22:02:18 32.6%us 37.9%sy 29.5%id
2016-05-12 22:02:19 33.1%us 38.7%sy 28.2%id
2016-05-12 22:02:20 31.7%us 37.8%sy 30.4%id
2016-05-12 22:02:21 31.8%us 38.4%sy 29.9%id
2016-05-12 22:02:22 33.5%us 38.1%sy 28.4%id
2016-05-12 22:02:23 33.2%us 38.5%sy 28.3%id
2016-05-12 22:02:24 33.2%us 39.6%sy 27.2%id
2016-05-12 22:02:25 33.6%us 37.8%sy 28.6%id
2016-05-12 22:02:26 32.9%us 38.8%sy 28.2%id
2016-05-12 22:02:27 32.2%us 37.8%sy 30.0%id
2016-05-12 22:02:28 32.5%us 39.7%sy 27.7%id
2016-05-12 22:02:29 32.4%us 38.5%sy 29.2%id
2016-05-12 22:02:30 32.8%us 38.1%sy 29.0%id
2016-05-12 22:02:31 34.0%us 37.9%sy 28.1%id
2016-05-12 22:02:42 32.7%us 39.5%sy 27.9%id
2016-05-12 22:02:43 32.0%us 38.2%sy 29.8%id
2016-05-12 22:02:44 32.4%us 38.1%sy 29.6%id
2016-05-12 22:02:45 32.3%us 39.1%sy 28.6%id
2016-05-12 22:02:46 32.7%us 38.5%sy 28.9%id
```

(cpu.pl shell)

```
#!/usr/bin/perl
use strict;
# 변수 선언
my $blsFirst = 1;
my ( $iUser, $iSystem, $idle, $iSum );
my ( $iOldUser, $iOldSystem, $iOldIdle );
my @arrList = ();
my @arrTime;
my $strText;
while(1)
{
```



```

# 현재 시간을 구한다.
@arrTime = localtime(time);
# /proc/stat 파일에서 최상위 CPU 정보를 가져와서 현재의 CPU 사
용량을 구한다.
open(FILE, "/proc/stat") or die "/proc/stat open error: $!";
while(<FILE>)
{
    @arrList = split( /\s+/, $_ );
    if( $blsFirst == 1 )
    {
        $iOldUser = $arrList[1];
        $iOldSystem = $arrList[3];
        $iOldIdle = $arrList[4];
        $blsFirst = 0;
        last;
    }
    $iUser = $arrList[1] - $iOldUser;
    $iSystem = $arrList[3] - $iOldSystem;
    $iIdle = $arrList[4] - $iOldIdle;
    $iOldUser = $arrList[1];
    $iOldSystem = $arrList[3];
    $iOldIdle = $arrList[4];
    $iSum = $iUser + $iSystem + $iIdle;
    # CPU 사용량의 퍼센트 단위로 계산한다.
    $iUser = ( $iUser * 100 ) / $iSum;
    $iSystem = ( $iSystem * 100 ) / $iSum;
    $iIdle = ( $iIdle * 100 ) / $iSum;
    $strText = sprintf( "%04d-%02d-%02d %02d:%02d:%02d %.1f%%us
%.1f%%sy %.1f%%idWn"
        , $arrTime[5] + 1900, $arrTime[4] + 1, $arrTime[3]
        , $arrTime[2], $arrTime[1], $arrTime[0]

```

```

    , $iUser, $iSystem, $idle );
# print "$strText";
# system("print $strText > top.txt");
open FH, ">>", "/home/kkk1/out.txt" or die "$!Wn";
print FH "$strText";
close FH;
#open FH, ">>", "/home/kkk1/cpu_bak.txt" or die "$!Wn";
#print FH "$strText";
#close FH;
    last;
}
close(FILE);
sleep(1);
}

```

(memory.sh)

(main 의 mem값을 txt파일안에 실시간으로 기록해 놓는다)

```

49. 16%
49. 17%
49. 18%
49. 18%
49. 19%
49. 19%
49. 20%
49. 20%
49. 20%
49. 19%
49. 18%
49. 18%
49. 18%
49. 19%
49. 20%
49. 20%
49. 21%
49. 21%
49. 21%
49. 21%
49. 21%
49. 23%
49. 24%

```

(memory.sh shell)

```
#!/bin/bash
```

```
while [ : ]; do
```

```
    mem=$(free | grep Mem | awk '{printf "%.2f% ",$3/$2*100}')
```

```
    echo $mem >> /home/test1/SSO/mem_status.txt
```

```
done
```

(traffic.sh)

```
16: 13: 35 : 353 / 259 : 5141181315 / 2679118529
16: 13: 37 : 441 / 310 : 5141271798 / 2679185073
16: 13: 39 : 392 / 235 : 5141384934 / 2679264621
16: 13: 41 : 349 / 219 : 5141485348 / 2679324889
16: 13: 43 : 284 / 142 : 5141574932 / 2679381133
16: 13: 45 : 357 / 192 : 5141647864 / 2679417571
16: 13: 47 : 443 / 237 : 5141739398 / 2679466885
16: 13: 49 : 392 / 201 : 5141852870 / 2679527605
16: 13: 51 : 570 / 283 : 5141953684 / 2679579645
16: 13: 53 : 524 / 240 : 5142099689 / 2679652177
16: 13: 55 : 433 / 201 : 5142234239 / 2679714245
16: 13: 57 : 410 / 182 : 5142345177 / 2679765885
16: 13: 59 : 425 / 183 : 5142450889 / 2679813401
16: 14: 01 : 572 / 263 : 5142560187 / 2679860451
16: 14: 03 : 723 / 288 : 5142706842 / 2679927940
16: 14: 05 : 789 / 296 : 5142891940 / 2680001690
16: 14: 07 : 665 / 248 : 5143094640 / 2680077996
16: 14: 09 : 730 / 260 : 5143265102 / 2680141560
16: 14: 11 : 740 / 258 : 5143452222 / 2680208172
16: 14: 13 : 747 / 262 : 5143642012 / 2680274750
16: 14: 15 : 789 / 268 : 5143833300 / 2680341824
16: 14: 17 : 423 / 272 : 5144035399 / 2680410618
16: 14: 19 : 394 / 297 : 5144143745 / 2680480384
```

(traffic.sh shell)

```
#!/bin/bash
if [ "$1" == "" ] ; then delay=2 ; else delay=$1 ; fi
# echo " 시간 : 트래픽(수신/송신) : Data(수신량/송신량)"
## 수신=(kbit/sec) / 송신=(kbit/sec)
while ( true ) ; do
    rx1=`grep eno1 /proc/net/dev | awk '{print $2}'`
    tx1=`grep eno1 /proc/net/dev | awk '{print $10}'`
    sleep $delay
    rx2=`grep eno1 /proc/net/dev | awk '{print $2}'`
    tx2=`grep eno1 /proc/net/dev | awk '{print $10}'`
    # 1024/8 == 128
    rx3=$(((rx2-rx1)/128/delay))
    tx3=$(((tx2-tx1)/128/delay))
    echo "`date '+%k:%M:%S'` : $rx3 / $tx3 : $rx1 / $tx1 "
    echo $rx3 KiB/s >> /home/test1/SSO/rx3.txt
    echo $tx3 KiB/s >> /home/test1/SSO/tx3.txt
done
```

(control menu)

(main control menu)

```
=====
                        SERVER CONTROL
=====
|
|      1. DISK USAGE
|      2. S1_CPU/Memory USAGE
|      3. S2_CPU/Memory USAGE
|      4. Command control
|      5. Exit
|
=====
```

1.DISK USAGE

```
=====
SELECT MENU : 1
-----DISK USAGE-----
Root size 50G Root avail 44G Root Use %13%
Home size 1.8T Home avail 1.8T Home Use %1%
Boot size 494M Boot avail 336M Boot Use %32%
-----
■
```

(서버의 디스크 사용값을 가져와 보여준다.)

2.S1_CPU/memory USAGE

[MEM USAGE]				
USER	PID	%MEM	TIME	COMMAND
root	23389	10.9	00: 07: 49	firefox
root	3593	6.1	00: 09: 42	gnome-shell
root	1625	2.3	00: 48: 29	Xorg
root	3699	1.3	00: 00: 00	evolution-calen
root	3676	1.1	00: 00: 07	nautilus
root	4096	1.0	00: 08: 47	gnome-terminal-
root	2347	0.7	00: 00: 00	file-roller
[CPU USAGE]				
USER	PID	%CPU	TIME	COMMAND
root	23389	3.9	00: 07: 49	firefox
root	4096	2.4	00: 08: 47	gnome-terminal-
root	3699	0.0	00: 00: 00	evolution-calen
root	3676	0.0	00: 00: 07	nautilus
root	3593	2.7	00: 09: 42	gnome-shell
root	2347	0.0	00: 00: 00	file-roller
root	1625	13.1	00: 48: 29	Xorg

(second서버에서 사용중인 프로세서중 cpu.mem 사용량이 높은 프로세서 순으로 정렬하여 보여준다.)

3.S2_CPU/memory USAGE

[MEM USAGE]

USER	PID	%MEM	TIME	COMMAND
root	11270	18.0	00:11:39	firefox
root	3524	7.7	00:05:47	gnome-shell
root	3927	2.6	00:00:00	evolution-calen
root	3608	1.8	00:00:06	nautilus
root	4028	1.7	00:01:32	gnome-terminal-
root	1430	1.6	00:03:27	Xorg
root	3562	1.5	00:00:25	caribou

[CPU USAGE]

USER	PID	%CPU	TIME	COMMAND
root	11270	5.4	00:11:39	firefox
root	4028	0.4	00:01:32	gnome-terminal-
root	3927	0.0	00:00:00	evolution-calen
root	3608	0.0	00:00:06	nautilus
root	3562	0.1	00:00:25	caribou
root	3524	1.6	00:05:47	gnome-shell
root	1430	0.9	00:03:27	Xorg

(third서버에서 사용중인 프로세서중 cpu.mem 사용량이 높은 프로세서 순으로 정렬하여 보여준다.)

4.command control

(command control 메뉴에서는 second,third 서버에 사용할 수 있는 명령어를 차단/해제 할 수 있다.)

```
=====
1. server1
2. server2
3. exit
=====
번호를 선택하시오 : █

=====
1. 차단할명령어
2. 해제할명령어
3. exit
=====
번호를 선택하시오 : 1
차단할 명령어를 입력하세요 : rpm
rpm 명령어차단 Success █
```

(second server 에서 rpm 명령어 사용을 차단하였다.)


```
[kkk1@second ~]$ rpm
-bash: /usr/bin/rpm: 허가 거부
```

(second server에서 rpm 명령어를 사용하자 '허가 거부' 라고 뜨며 사용이 차단된다.)

(서버)

```
=====
1. 차단할 명령어
2. 해제할 명령어
3. exit
=====
번호를 선택하시오 : 2
# 해제할 명령어를 입력하시오 : rpm
rpm 명령어해제 Success █
```

(차단한 명령어를 다시 해제 할 수 있다.)

3.4.2 Second서버

(cpu.sh)

```
24.6  
26.9  
25.1  
25.7  
26.1  
24.9  
26.3  
25.1  
25.2  
25.9  
26.1  
25.7  
24.5  
26.2  
24.9  
25.7  
25.4  
25.6  
26.0  
25.5  
26.3  
25.5  
25.6  
"cpu2.txt" [읽기 전용] 61917L, 274256C
```

(cpu.sh shell)

```
#!/bin/bash
```

```
while [ : ]
```

```
do
```

```
#A=`tail -1 /home/kkk1/out.txt`
```

```
tail -1 /home/kkk1/out.txt | cut -f 3 -d " " | cut -f 1 -d "%" >>  
/home/kkk1/cpu_status2.txt
```

(cpu.pl shell)

```
#!/usr/bin/perl
use strict;
# 변수 선언
my $blsFirst = 1;
my ( $iUser, $iSystem, $iIdle, $iSum );
my ( $iOldUser, $iOldSystem, $iOldIdle );
my @arrList = ();
my @arrTime;
my $strText;
while(1)
{
    # 현재 시간을 구한다.
    @arrTime = localtime(time);
    # /proc/stat 파일에서 최상위 CPU 정보를 가져와서 현재의 CPU 사
    용량을 구한다.
    open(FILE,"/proc/stat") or die "/proc/stat open error: $!";
    while(<FILE>)
    {
        @arrList = split( /\s+/, $_ );
        if( $blsFirst == 1 )
        {
            $iOldUser = $arrList[1];
            $iOldSystem = $arrList[3];
            $iOldIdle = $arrList[4];
            $blsFirst = 0;
            last;
        }
        $iUser = $arrList[1] - $iOldUser;
```

```

$iSystem = $arrList[3] - $iOldSystem;
$iIdle = $arrList[4] - $iOldIdle;
$iOldUser = $arrList[1];
$iOldSystem = $arrList[3];
$iOldIdle = $arrList[4];
$iSum = $iUser + $iSystem + $iIdle;
# CPU 사용량의 퍼센트 단위로 계산한다.
$iUser = ( $iUser * 100 ) / $iSum;
$iSystem = ( $iSystem * 100 ) / $iSum;
$iIdle = ( $iIdle * 100 ) / $iSum;
$strText = sprintf( "%04d-%02d-%02d %02d:%02d:%02d %.1f%%us
%.1f%%sy %.1f%%idWn"
    , $arrTime[5] + 1900, $arrTime[4] + 1, $arrTime[3]
    , $arrTime[2], $arrTime[1], $arrTime[0]
    , $iUser, $iSystem, $iIdle );
# print "$strText";
# system("print $strText > top.txt");
open FH, ">>", "/home/kkk1/out.txt" or die "$!Wn";
print FH "$strText";
close FH;
#open FH, ">>", "/home/kkk1/cpu_bak.txt" or die "$!Wn";
#print FH "$strText";
#close FH;
    last;
}
close(FILE);
sleep(1);
}

sleep 2
done

```

(memory.sh)

```
46.62%  
46.59%  
46.61%  
46.60%  
46.62%  
46.65%  
46.64%  
46.66%  
46.65%  
46.61%  
46.65%  
46.65%  
46.67%  
46.67%  
46.66%  
46.64%  
46.65%  
46.63%  
46.64%  
46.62%  
46.63%  
46.64%  
46.62%  
"memory2.txt" [읽기 전용] 4239256L, 296
```

(memory.txt)

#!/bin/bash

while [:]; do

mem=\$(free | grep Mem | awk '{printf "%.2f% ",\$3/\$2*100}')

echo \$mem >> /home/kkk1/memory2.txt

done

(traffic.sh)

```
15: 54: 22 : 116 KiB/s / 114 KiB/s
15: 54: 23 : 122 KiB/s / 120 KiB/s
15: 54: 24 : 150 KiB/s / 167 KiB/s
15: 54: 26 : 128 KiB/s / 127 KiB/s
15: 54: 27 : 152 KiB/s / 170 KiB/s
15: 54: 28 : 134 KiB/s / 128 KiB/s
15: 54: 29 : 129 KiB/s / 144 KiB/s
15: 54: 30 : 136 KiB/s / 129 KiB/s
15: 54: 31 : 116 KiB/s / 131 KiB/s
15: 54: 33 : 119 KiB/s / 119 KiB/s
15: 54: 34 : 147 KiB/s / 156 KiB/s
15: 54: 35 : 129 KiB/s / 129 KiB/s
15: 54: 36 : 134 KiB/s / 137 KiB/s
15: 54: 37 : 135 KiB/s / 158 KiB/s
15: 54: 39 : 129 KiB/s / 129 KiB/s
15: 54: 40 : 120 KiB/s / 141 KiB/s
15: 54: 41 : 122 KiB/s / 128 KiB/s
15: 54: 42 : 130 KiB/s / 129 KiB/s
15: 54: 43 : 142 KiB/s / 160 KiB/s
15: 54: 45 : 123 KiB/s / 133 KiB/s
15: 54: 46 : 132 KiB/s / 136 KiB/s
15: 54: 47 : 146 KiB/s / 153 KiB/s
15: 54: 48 : 139 KiB/s / 155 KiB/s
```

(traffic.sh shell)

```
#!/bin/bash
```

```
if [ "$1" == "" ] ; then delay=1 ; else delay=$1 ; fi
```

```
echo " 시간 : 트래픽(수신/송신)"
```

```
while ( true ) ; do
```

```
rx1=`grep enp0s25 /proc/net/dev | awk '{print $2}'`
```

```
tx1=`grep enp0s25 /proc/net/dev | awk '{print $10}'`
```

```
sleep $delay
```

```
rx2=`grep enp0s25 /proc/net/dev | awk '{print $2}'`
```

```
tx2=`grep enp0s25 /proc/net/dev | awk '{print $10}'`
```

```
rx3=$(((rx2-rx1)/128/delay))
```

```

tx3=$((tx2-tx1)/128/delay))
echo "`date '+%k:%M:%S`" : $rx3 KiB/s / $tx3 KiB/s "
    echo $rx3 KiB/s >> /home/kkk1/rx3.txt
    echo $tx3 KiB/s >> /home/kkk1/tx3.txt
done

```

3.4.3 Third서버

(cpu.sh)

```

5.6
4.2
6.1
5.3
5.2
4.7
5.3
5.9
4.8
8.8
4.7
4.8
5.1
4.5
4.6
5.5
4.0
4.4
5.2
5.2
8.1
9.3
12.8
"

```

(cpu.sh shell)

```
#!/bin/bash
while [ : ]
do
#A="tail -1 out.txt"
tail -1 tail -1 /home/kkk2/out.txt | cut -f 3 -d " " | cut -f 1 -d "%"
>> /home/kkk2/cpu3.txt
sleep 2
done
```

(cpu.pl shell)

```
#!/usr/bin/perl
use strict;
# 변수 선언
my $blsFirst = 1;
my ( $iUser, $iSystem, $iIdle, $iSum );
my ( $iOldUser, $iOldSystem, $iOldIdle );
my @arrList = ();
my @arrTime;
my $strText;
while(1)
{
# 현재 시간을 구한다.
@arrTime = localtime(time);
# /proc/stat 파일에서 최상위 CPU 정보를 가져와서 현재의 CPU 사
용량을 구한다.
open(FILE,"/proc/stat") or die "/proc/stat open error: $!";
while(<FILE>)
```



```

{
    @arrList = split( /Ws+/, $_ );
    if( $blsFirst == 1 )
    {
        $iOldUser = $arrList[1];
        $iOldSystem = $arrList[3];
        $iOldIdle = $arrList[4];
        $blsFirst = 0;
        last;
    }
    $iUser = $arrList[1] - $iOldUser;
    $iSystem = $arrList[3] - $iOldSystem;
    $iIdle = $arrList[4] - $iOldIdle;
    $iOldUser = $arrList[1];
    $iOldSystem = $arrList[3];
    $iOldIdle = $arrList[4];
    $iSum = $iUser + $iSystem + $iIdle;
    # CPU 사용량의 퍼센트 단위로 계산한다.
    $iUser = ( $iUser * 100 ) / $iSum;
    $iSystem = ( $iSystem * 100 ) / $iSum;
    $iIdle = ( $iIdle * 100 ) / $iSum;
    $strText = sprintf( "%04d-%02d-%02d %02d:%02d:%02d %.1f%%us
%.1f%%sy %.1f%%idWn"
        , $arrTime[5] + 1900, $arrTime[4] + 1, $arrTime[3]
        , $arrTime[2], $arrTime[1], $arrTime[0]
        , $iUser, $iSystem, $iIdle );
    # print "$strText";
    # system("print $strText > top.txt");
    open FH, ">>", "/home/kkk2/out.txt" or die "$!Wn";
    print FH "$strText";
    close FH;

```

```

open FH, ">>", "/home/kkk2/cpu_bak.txt" or die "$!Wn";
print FH "$strText";
close FH;
    last;
}
close(FILE);
sleep(1);
}

```

(memory.sh)

```

34. 18%
34. 19%
34. 20%
34. 19%
34. 18%
34. 19%
34. 19%
34. 18%
34. 19%
34. 19%
34. 19%
34. 19%
34. 18%
34. 18%
34. 19%
34. 19%
34. 19%
34. 18%
34. 18%
34. 18%
34. 18%
34. 18%
34. 19%
"memory3.txt" 2797521L, 19420191C

```

(memory.sh shell)

```
#!/bin/bash
```

```
while [ : ]; do
```

```
    mem=$(free | grep "Mem" | awk '{printf "%.2f%",$3/$2*100}')
    echo $mem >> /home/kkk2/memory3.txt
```

```
done
```

(traffic.sh)

```
15:59:39 : 109 KiB/s / 546 KiB/s
15:59:40 : 117 KiB/s / 589 KiB/s
15:59:42 : 126 KiB/s / 640 KiB/s
15:59:43 : 124 KiB/s / 295 KiB/s
15:59:44 : 132 KiB/s / 179 KiB/s
15:59:45 : 187 KiB/s / 197 KiB/s
15:59:46 : 113 KiB/s / 181 KiB/s
15:59:48 : 105 KiB/s / 160 KiB/s
15:59:49 : 171 KiB/s / 178 KiB/s
15:59:51 : 62 KiB/s / 126 KiB/s
15:59:52 : 113 KiB/s / 204 KiB/s
15:59:53 : 125 KiB/s / 254 KiB/s
15:59:54 : 148 KiB/s / 281 KiB/s
15:59:55 : 122 KiB/s / 282 KiB/s
15:59:57 : 109 KiB/s / 289 KiB/s
15:59:58 : 74 KiB/s / 197 KiB/s
15:59:59 : 114 KiB/s / 305 KiB/s
16:00:01 : 117 KiB/s / 337 KiB/s
16:00:02 : 78 KiB/s / 231 KiB/s
16:00:03 : 85 KiB/s / 255 KiB/s
16:00:05 : 86 KiB/s / 248 KiB/s
16:00:06 : 118 KiB/s / 366 KiB/s
16:00:07 : 125 KiB/s / 392 KiB/s
```

(traffic.sh shell)

```
#!/bin/bash
```

```
if [ "$1" == "" ] ; then delay=1 ; else delay=$1 ; fi
```

```
echo " 시간 : 트래픽(수신/송신)"
```

```
while ( true ) ; do
```

```
rx1=`grep enp3s0 /proc/net/dev | awk '{print $2}'`
```

```
tx1=`grep enp3s0 /proc/net/dev | awk '{print $10}'`
```

```
sleep $delay
```

```
rx2=`grep enp3s0 /proc/net/dev | awk '{print $2}'`
```

```
tx2=`grep enp3s0 /proc/net/dev | awk '{print $10}'`
```

```
# 1024/8 == 128
```

```
rx3=$(((rx2-rx1)/128/delay))
```

```
tx3=$(((tx2-tx1)/128/delay))
```

```
echo "`date '+%k:%M:%S'` : $rx3 KiB/s / $tx3 KiB/s"
```

```
echo $rx3 KiB/s >> /home/kkk2/rx3.txt
```

```
echo $tx3 KiB/s >> /home/kkk2/tx3.txt
```

```
done
```

4. 결론

정보 시스템 다양화에 따라 1인 평균 ID와 PW를 5개 이상 보유하고 있으며 개인정보를 수정 할 때 가진 각각의 계정들을 수정하기가 번거롭고 개인정보가 유출될 때 언제 어디서 어떤 계정이 유출 됐는지 알 수 없기 때문에 효율적인 관리를 위해 SSO시스템이 필히 요구된다.

다른 아이디와 암호 조합으로 인한 암호 피곤을 줄일 수 있으며, 같은 아이디마다 암호를 다시 입력하는 시간을 줄일 수 있어 헬프데스크 비용을 줄일 수 있는 장점을 가지고 있다.

리눅스 운영체제에서 SSO인증 체제 구현을 하기 위해서 사용자, LDAP(디렉토리 서비스), Kerberos인증, Token송/수신 수행을 이용한다. 단일 ID Password만 사용함으로써 SSO의 유용성이 입증됨으로 보안관리가 용이하며 비용절감 등 보안관리 기능 향상, 다양한 인터넷 환경에 대응하는 표준 보안 인프라 체계를 기대한다.

5. 참고 자료

[1]UNIX/Linux 시스템 관리자를 위한 쉘 스크립트 활용 가이드 -정해주 (비판복스)

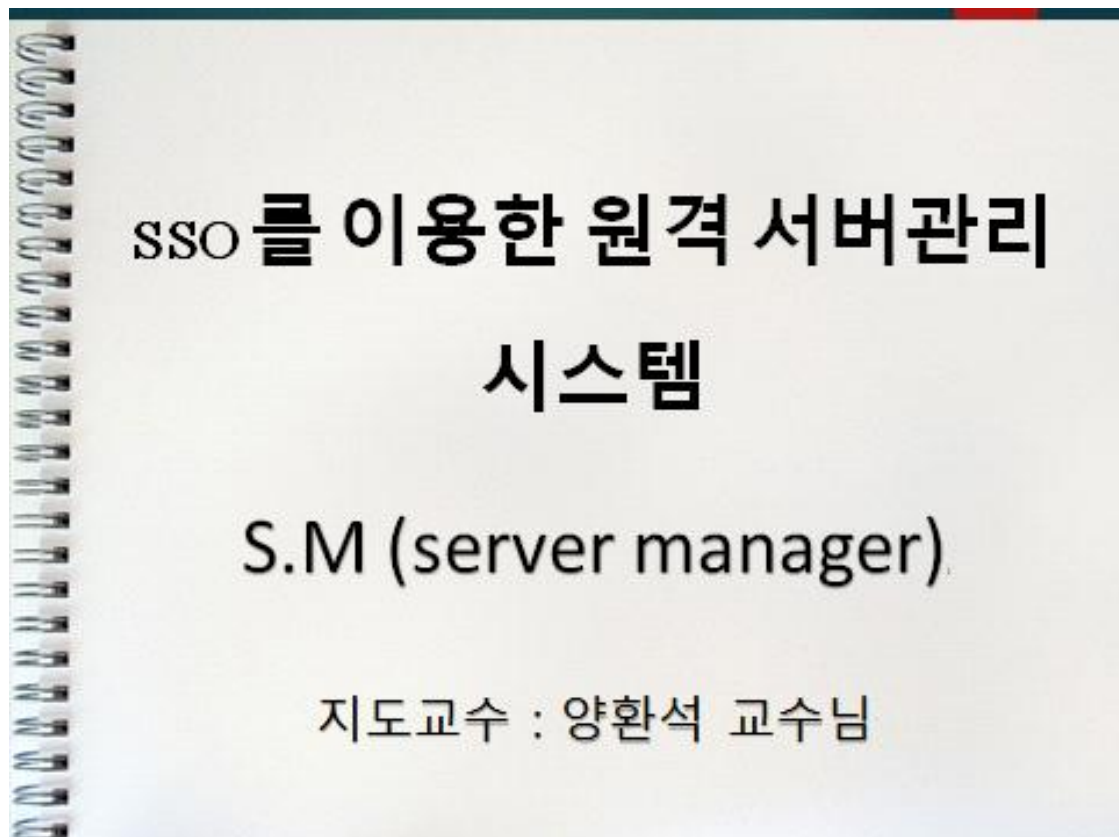
[2]Linux centos 기초에서 활용까지 -이지선. (EHAN MEDIA)

[3]웹 접근성 웹 표준 완벽 가이드

LADP/Kerberos - 짐 대처 (에이콘 출판)

https://wiki.debian.org/LDAP/Kerberos#LDAP_.2B-_Kerberos

6. 발표 자료



INDEX	①	조언 소개
	②	추진 경과
	③	주제 선정
	④	시스템 구성
	⑤	결론

조원 소개



조장 (이혁서)
장 민 보

- 커트론 메인 구성
- 셀 스크립트



조원 (홍석현)
원 오 기 인

- 프로젝트 총괄
- 시스템 구축



조원 (장서재)
원 오 기

- 자료조사
- 셀 스크립트



조원 (남여정)
원 민 오

- 자료조사
- 셀 스크립트

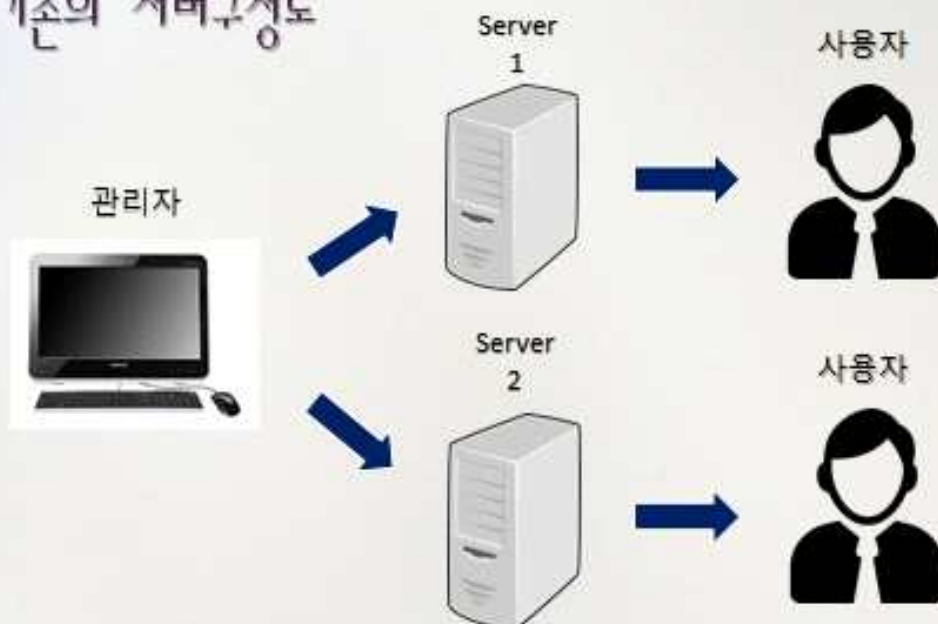
추진 경과

추진 경과

추진일정 수행내용	1~3월				2~4월				5월			
	1	2	3	4	1	2	3	4	1	2	3	4
SSO 구성 및 웹 사이트구현	○	○	○	○	○							
웹 스크립트구현				○	○	○	○	○				
시스템 점검 및 오류개선								○	○	○		
오류 및 마무리										○	○	○

주제 선정 이유

기존의 서버구성도



SSO란 무엇인가?



SSO(Single sign on) : 단 한번의 인증
과정으로 여러 컴퓨터 상의 자원을 이용 가능하게
하는 인증 기능.

시스템 구성

SSO 를 도입한 서버구성도



시스템 구현 - 웹 기반 로그인 화면

관리자 Login

ID : admin

PW : [masked]

Login

© Copyright 2016 - JoongBu.Univ

시스템 구현 - 로그인 진행



메인 페이지 화면



원격 로그인 - 웹 ssh

시스템 구현 – 웹 ssh → noVNC 접속

```
[root@localhost kanaka-noVNC-b403cb9]# ./utils/launch.sh --vnc 10.100.114.72:5901
Using installed websockify at /usr/bin/websockify
Starting webserver and WebSockets proxy on port 6080
WebSocket server settings:
- Listen on :6080
- Flash security policy server
- Web server. Web root: /root/kanaka-noVNC-b403cb9
- SSL/TLS support
- proxying from :6080 to 10.100.114.72:5901
```

Navigate to this URL:

<http://localhost.localdomain:6080/vnc.html?host=localhost.localdomain&port=6080>

Press Ctrl-C to exit



원격 웹 접속 URL

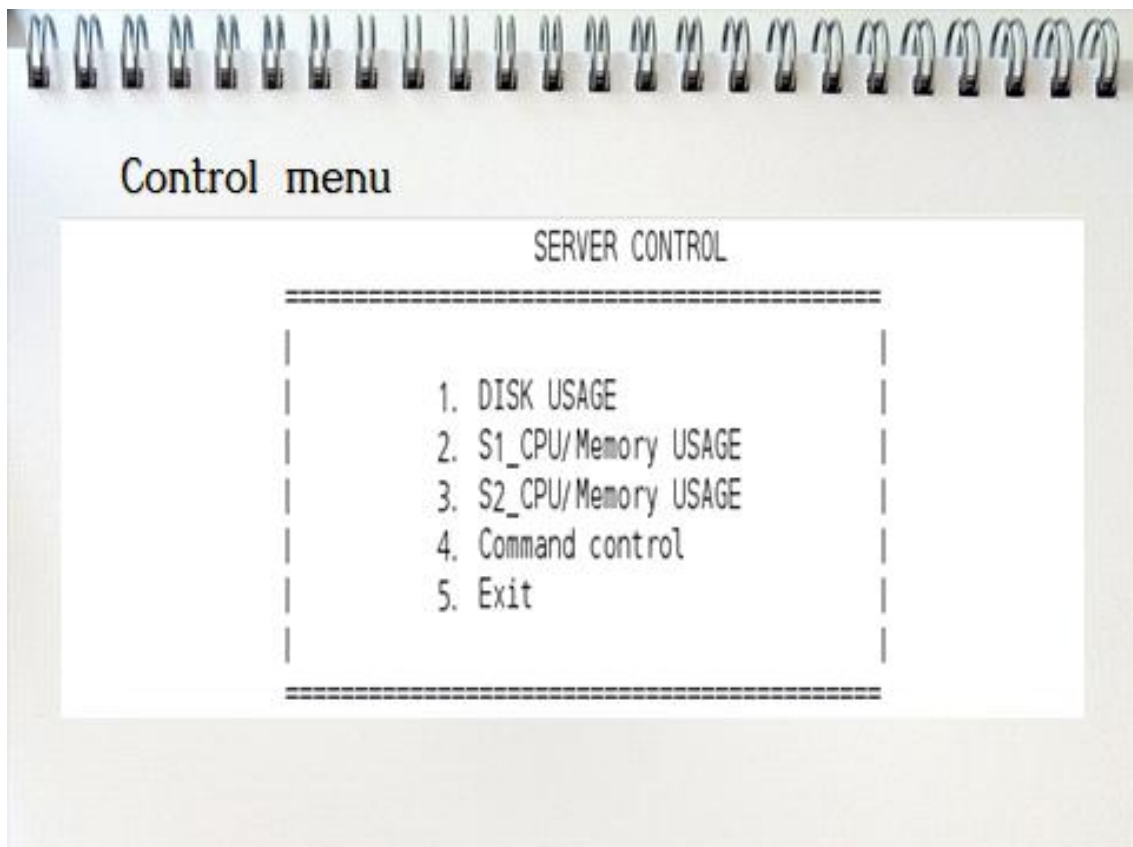
시스템 구현 – noVNC 로그인



시스템 구현 – noVNC



Main 서버 접속



Control menu

1. DISK USAGE

SELECT MENU : 1

-----DISK USAGE-----

Root size 50G Root avail 44G Root Use %13%

Home size 1.8T Home avail 1.8T Home Use %1%

Boot size 494M Boot avail 336M Boot Use %32%

■

Control menu

2. S1_CPU/Memory USAGE

[MEM USAGE]

USER	PID	%MEM	TIME	COMMAND
root	23389	10.9	00:07:49	firefox
root	3593	6.1	00:09:42	gnome-shell
root	1625	2.3	00:48:29	Xorg
root	3699	1.3	00:00:00	evolution-calen
root	3676	1.1	00:00:07	nautilus
root	4096	1.0	00:08:47	gnome-terminal-
root	2347	0.7	00:00:00	file-roller

[CPU USAGE]

USER	PID	%CPU	TIME	COMMAND
root	23389	3.9	00:07:49	firefox
root	4096	2.4	00:08:47	gnome-terminal-
root	3699	0.0	00:00:00	evolution-calen
root	3676	0.0	00:00:07	nautilus
root	3593	2.7	00:09:42	gnome-shell
root	2347	0.0	00:00:00	file-roller

3. S2_CPU/Memory USAGE

[MEM USAGE]

USER	PID	%MEM	TIME	COMMAND
root	11270	18.0	00:11:39	firefox
root	3524	7.7	00:05:47	gnome-shell
root	3927	2.6	00:00:00	evolution-calen
root	3608	1.8	00:00:06	nautilus
root	4028	1.7	00:01:32	gnome-terminal-
root	1430	1.6	00:03:27	Xorg
root	3562	1.5	00:00:25	caribou

[CPU USAGE]

USER	PID	%CPU	TIME	COMMAND
root	11270	5.4	00:11:39	firefox
root	4028	0.4	00:01:32	gnome-terminal-
root	3927	0.0	00:00:00	evolution-calen
root	3608	0.0	00:00:06	nautilus
root	3562	0.1	00:00:25	caribou
root	3524	1.6	00:05:47	gnome-shell
root	1430	0.9	00:03:27	Xorg

Control menu

4. Command control

```
=====
1.server1
2.server2
3.exit
=====
번호를 선택하십시오 : █
```

```
=====
1.차단명령어
2.해제명령어
3.exit
=====
번호를 선택하십시오 : 1
차단명령어를 입력하십시오 : rpm
rpm명령어 차단 Success █
```

결론

→ 사용자가 사이트에 접속하기 위하여 아이디와 패스워드는 계인정보를 각 사이트마다
일일이 기록해야 하던 것을 **한 번의 작업으로 끝나므로 불편함이 없다**

→ 관리자는 언제 어디서든 관리 서버에 문제가 생겼을때 웹으로 원격 서버 관리 시
스템에 접속해 서버들의 에러등에 **즉각적으로 대처 할수있다**

→ 하나의 계정으로 다른 서버에 접속할 아이디와 패스워드를 하나하나
외워야 하는 부담을 줄이고, 보다 쉽게 각 **서버들을 원격으로 관리 및 제어할
수 있도록 해준다**

THANK YOU