# init-7-v2

#!/bin/bash  
export PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin  
  
# [配置备份目录]  
BACKUPDIR=/var/log/.backups  
if [ ! -d ${BACKUPDIR} ];then mkdir -vp ${BACKUPDIR}; fi  
  
# [配置记录目录]  
HISDIR=/var/log/.history  
if [ ! -d ${HISDIR} ];then mkdir -vp ${HISDIR}; fi  
  
# [用户设置]  
DefaultUser="app" # 系统创建的用户名称非root用户  
ROOTPASS=Z123.com # 密码建议12位以上且包含数字、大小写字母以及特殊字符。  
APPPASS=Z123.com  
  
log\_err() {  
 printf "[$(date +'%Y-%m-%dT%H:%M:%S')]: \033[31mERROR: $@ \033[0m\n"  
}  
  
log\_info() {  
 printf "[$(date +'%Y-%m-%dT%H:%M:%S')]: \033[32mINFO: $@ \033[0m\n"  
}  
  
log\_warning() {  
 printf "[$(date +'%Y-%m-%dT%H:%M:%S')]: \033[33mWARNING: $@ \033[0m\n"  
}  
  
RGB\_DANGER='\033[31;1m'  
RGB\_WAIT='\033[37;2m'  
RGB\_SUCCESS='\033[32m'  
RGB\_WARNING='\033[33;1m'  
RGB\_INFO='\033[36;1m'  
RGB\_END='\033[0m'  
  
EXECTIME=$(date +%Y%m%d-%m%S)  
CHECK\_CENTOS=$( cat /etc/redhat-release|sed -r 's/.\* ([0-9]+)\..\*/\1/' )  
PLATFORM=$(uname -i)  
CHECK\_RAM=$( cat /proc/meminfo | grep "MemTotal" | awk -F" " '{ram=$2/1000000}{printf("%.0f",ram)}' )  
  
sh\_path=$(cd `dirname $0`;pwd)  
LOGDIR=/var/log/init\_record  
if [ ! -d ${LOGDIR} ];then mkdir -vp ${LOGDIR}; fi  
  
DNSIP=("223.5.5.5" "223.6.6.6" "119.29.29.29")  
  
######start########  
tool\_info() {  
 echo -e "========================================================================================="  
 echo -e " Init CentOS 7 Script "  
 echo -e "========================================================================================="  
}  
  
check\_root(){  
 if [[ $EUID -ne 0 ]]; then  
 echo -e "${RGB\_DANGER}This script must be run as root!${RGB\_END}"  
 exit 1  
 fi  
}  
  
check\_os() {  
 if [ "${CHECK\_CENTOS}" != '7' -a ${PLATFORM} != "x86\_64" ]; then  
 echo -e "${RGB\_DANGER}This script must be run in CentOS 7 64bit!${RGB\_END}"  
 exit 1  
 fi  
}  
  
os\_dns(){  
 log\_name=os\_dns.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] dns配置....." |tee -a ${LOGDIR}/${log\_name}  
  
 # 系统DNS域名解析服务设置  
 cp -av /etc/resolv.conf ${BACKUPDIR}/resolv.conf.bak |tee -a ${LOGDIR}/${log\_name}  
 for dns in ${DNSIP[@]};do echo "nameserver ${dns}" >> /etc/resolv.conf;done   
  
 log\_info "[\*] dns配置 successful!" |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] dns已配置!"  
 fi  
}  
  
#CentOS 软件仓库镜像源配置&&初始化更新  
add\_repo(){  
 log\_name=add\_repo.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[\*] CentOS 软件仓库镜像源配置"  
 mkdir /etc/yum.repos.d/backup  
 mv -v /etc/yum.repos.d/\*.repo /etc/yum.repos.d/backup/ |tee -a ${LOGDIR}/${log\_name}  
 curl -o /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo |tee -a ${LOGDIR}/${log\_name}  
 curl -o /etc/yum.repos.d/CentOS-epel.repo http://mirrors.aliyun.com/repo/epel-7.repo |tee -a ${LOGDIR}/${log\_name}  
 sed -i "s#mirrors.cloud.aliyuncs.com#mirrors.aliyun.com#g" /etc/yum.repos.d/CentOS-Base.repo |tee -a ${LOGDIR}/${log\_name}  
 sed -i "s#mirrors.aliyuncs.com#mirrors.aliyun.com#g" /etc/yum.repos.d/CentOS-Base.repo |tee -a ${LOGDIR}/${log\_name}  
 rpm --import http://mirrors.aliyun.com/centos/RPM-GPG-KEY-CentOS-7 |tee -a ${LOGDIR}/${log\_name}  
 yum clean all |tee -a ${LOGDIR}/${log\_name}  
 yum makecache |tee -a ${LOGDIR}/${log\_name}  
 yum install deltarpm -y |tee -a ${LOGDIR}/${log\_name}  
 yum --exclude=kernel\* update -y && yum upgrade -y && yum -y install epel\* |tee -a ${LOGDIR}/${log\_name}  
 log\_info "[\*] CentOS 软件仓库镜像源配置successful!" |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] CentOS 软件仓库镜像源已配置!"  
 fi  
}  
  
# 安装常用的运维软件  
add\_softbase() {  
 log\_name=add\_softbase.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] 安装常用的运维软件 "  
 # 编译软件  
 yum install -y gcc gcc-c++ g++ make jq libpam-cracklib openssl-devel bzip2-devel |tee -a ${LOGDIR}/${log\_name}  
 # 常规软件  
 yum install -y nano vim git unzip wget ntpdate dos2unix net-tools policycoreutils-python |tee -a ${LOGDIR}/${log\_name}  
 yum install -y tree htop ncdu nload sysstat psmisc bash-completion fail2ban nfs-utils chrony |tee -a ${LOGDIR}/${log\_name}  
 yum install -y epel-release zip autoconf libtool-ltdl-devel gd freetype-devel libxml2-devel libjpeg-devel libpng-devel \  
curl-devel bison patch ncurses-devel sudo bzip2 mlocate flex lrzsz lsof setuptool system-config-firewall-tui ntp libaio-devel \  
nmap finger elinks tcl telnet libcurl openssl cpp binutils glibc-kernheaders glibc-common glibc-devel automake \  
libtool zlib-devel bash-completion-extras nc open-sshclients screen yum-utils device-mapper-persistent-data \  
lvm2 cronolog |tee -a ${LOGDIR}/${log\_name}  
 # 清空缓存和已下载安装的软件包  
 yum clean all |tee -a ${LOGDIR}/${log\_name}  
   
 log\_info "[\*] 安装常用的运维软件 successful!........." |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] 常用的运维软件已安装!"  
 fi   
}  
  
#CentOS 操作系统内核升级  
upgrade\_kernel() {  
 log\_name=upgrade\_kernel.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 cp -av /etc/grub2.cfg ${BACKUPDIR}/grub2.cfg.kernelupdate.bak |tee -a ${LOGDIR}/${log\_name}  
 log\_info "[\*] CentOS 操作系统内核升级(可选) "  
 rpm --import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org |tee -a ${LOGDIR}/${log\_name}  
 yum -y install https://www.elrepo.org/elrepo-release-7.el7.elrepo.noarch.rpm |tee -a ${LOGDIR}/${log\_name}  
 yum install -y yum-plugin-fastestmirror |tee -a ${LOGDIR}/${log\_name}  
 yum --disablerepo="\*" --enablerepo=elrepo-kernel repolist |tee -a ${LOGDIR}/${log\_name}  
 yum --disablerepo="\*" --enablerepo=elrepo-kernel list kernel\* |tee -a ${LOGDIR}/${log\_name}  
 # 内核安装，服务器里我们选择长期lt版本，安全稳定是我们最大的需求，除非有特殊的需求内核版本需求;  
 yum update -y --enablerepo=elrepo-kernel |tee -a ${LOGDIR}/${log\_name}  
 # 内核版本介绍, lt:longterm 的缩写长期维护版, ml:mainline 的缩写最新主线版本;  
 yum install -y --enablerepo=elrepo-kernel --skip-broken kernel-lt kernel-lt-devel kernel-lt-tools |tee -a ${LOGDIR}/${log\_name}  
 # yum -y --enablerepo=elrepo-kernel --skip-broken install kernel-ml.x86\_64 kernel-ml-devel.x86\_64 kernel-ml-tools.x86\_64  
 log\_warning "[\*] 当前 CentOS 操作系统可切换的内核内核版本" |tee -a ${LOGDIR}/${log\_name}  
 awk -F \' '$1=="menuentry " {print i++ " : " $2}' /etc/grub2.cfg |tee -a ${LOGDIR}/${log\_name}  
 sudo grub2-set-default 0 |tee -a ${LOGDIR}/${log\_name}  
 log\_info "[\*] 操作系统内核升级successful!,接下来重启！ " |tee -a ${LOGDIR}/${log\_name}  
 reboot  
 else  
 log\_info "[\*] 操作系统内核已升级!"  
 fi  
  
}  
  
#openssl升级到最新版  
upgrade\_openssl () {  
 log\_info "[-] update\_openssl Configuring..." |tee -a ${LOGDIR}/${log\_name}  
 yum -y install gcc gcc-c++ glibc make autoconf openssl openssl-devel pcre-devel pam-devel |tee -a ${LOGDIR}/${log\_name}  
 yum install -y zlib\* pam\* zlib\* |tee -a ${LOGDIR}/${log\_name}  
 mv /usr/bin/openssl{,.bak} |tee -a ${LOGDIR}/${log\_name}  
 mv /usr/include/openssl{,.bak} |tee -a ${LOGDIR}/${log\_name}  
 tar xf ${opensslVer}.tar.gz  
  
 ls /usr/local/openssl/bin/openssl |tee -a ${LOGDIR}/${log\_name}  
 ls /usr/local/openssl/include/openssl/ -d |tee -a ${LOGDIR}/${log\_name}  
 ln -sfv /usr/local/openssl/bin/openssl /usr/bin/openssl |tee -a ${LOGDIR}/${log\_name}  
 ln -sfv /usr/local/openssl/lib/libssl.so.1.1 /usr/lib64/libssl.so.1.1 |tee -a ${LOGDIR}/${log\_name}  
 ln -sfv /usr/local/openssl/lib/libcrypto.so.1.1 /usr/lib64/libcrypto.so.1.1 |tee -a ${LOGDIR}/${log\_name}  
 ln -sfv /usr/local/openssl/include/openssl /usr/include/openssl |tee -a ${LOGDIR}/${log\_name}  
 echo "/usr/local/openssl/lib" >> /etc/ld.so.conf   
 /sbin/ldconfig |tee -a ${LOGDIR}/${log\_name}  
 openssl version |tee -a ${LOGDIR}/${log\_name}  
 cd ${sh\_path}  
 log\_info "[\*] update\_openssl successful!" |tee -a ${LOGDIR}/${log\_name}  
}  
  
check\_openssl() {  
 log\_name=check\_openssl.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 while true;do  
 curl https://www.openssl.org/source/ > dwopenssl.log 2>&1  
 num=$(grep openssl-1.\*.tar.gz dwopenssl.log |wc -l)  
 if [ ${num} != 1 ];then   
 echo "未获取到最新版本，将继续尝试下载，请耐心等待！" |tee -a ${LOGDIR}/${log\_name}  
 else   
 echo "获取到最新版本,开始下载！" |tee -a ${LOGDIR}/${log\_name}  
 opensslVer=$(grep openssl-1.\*.tar.gz dwopenssl.log |awk -F'"' '{print $2}'|awk -F . '{print $1"."$2"."$3}')  
 while true;do  
 wget --no-check-certificate https://www.openssl.org/source/${opensslVer}.tar.gz|tee -a ${LOGDIR}/${log\_name}  
 if [ "$?" != 0 ];then  
 echo "未下载成功，将继续尝试下载，请耐心等待！" |tee -a ${LOGDIR}/${log\_name}  
 else   
 echo "下载成功,开始部署！" |tee -a ${LOGDIR}/${log\_name}  
 upgrade\_openssl  
 break  
 fi  
 done  
 break  
 fi  
 done  
 else  
 log\_info "[\*] update\_openssl Configuration already!...!"  
 fi  
}  
  
#openssh升级到最新版  
upgrade\_openssh () {  
 log\_info "[-] update\_openssh Configuring..." |tee -a ${LOGDIR}/${log\_name}  
 #yum -y install gcc gcc-c++ glibc make autoconf openssl openssl-devel pcre-devel pam-devel |tee -a ${LOGDIR}/${log\_name}  
 #yum install -y zlib\* pam\* zlib\* |tee -a ${LOGDIR}/${log\_name}  
 mv /etc/ssh{,.bak}  
 tar xf ${opensshVer}.tar.gz  
  
  
 #先备份  
 mv /usr/sbin/sshd{,.bak}   
 mv /usr/bin/ssh{,.bak}  
 mv /usr/bin/ssh-keygen{,.bak}  
 #拷贝文件  
 \cp -av /usr/local/openssh/bin/ssh /usr/bin/ssh |tee -a ${LOGDIR}/${log\_name}  
 \cp -av /usr/local/openssh/bin/ssh-keygen /usr/bin/ssh-keygen |tee -a ${LOGDIR}/${log\_name}  
 \cp -av /usr/local/openssh/sbin/sshd /usr/sbin/sshd |tee -a ${LOGDIR}/${log\_name}  
 #查看openssh当前版本  
 ssh -V |tee -a ${LOGDIR}/${log\_name}  
 #重新启动openssh服务  
 #先停止服务  
 systemctl disable sshd --now |tee -a ${LOGDIR}/${log\_name}  
 #备份文件  
 mv /usr/lib/systemd/system/sshd.service{,.bak}   
 #重载守护进程  
 systemctl daemon-reload |tee -a ${LOGDIR}/${log\_name}  
 #复制配置文件   
 cd ${sh\_path}/${opensshVer}  
 cp -av contrib/redhat/sshd.init /etc/init.d/sshd |tee -a ${LOGDIR}/${log\_name}  
 cp -av contrib/redhat/sshd.pam /etc/pam.d/sshd.pam |tee -a ${LOGDIR}/${log\_name}  
 #\cp -av /etc/ssh.bak/sshd\_config /etc/ssh/ |tee -a ${LOGDIR}/${log\_name}  
 grep ^[a-Z] /etc/ssh.bak/sshd\_config|egrep -v "Subsystem|AuthorizedKeysFile" >> /etc/ssh/sshd\_config  
 #添加设置  
 chkconfig --add sshd |tee -a ${LOGDIR}/${log\_name}  
 #重新开启服务  
 systemctl enable sshd --now |tee -a ${LOGDIR}/${log\_name}  
 systemctl restart sshd  
 cd ${sh\_path}  
 log\_info "[\*] update\_openssh successful!" |tee -a ${LOGDIR}/${log\_name}  
}  
  
check\_openssh(){  
 log\_name=check\_openssh.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 while true;do  
 curl https://mirrors.aliyun.com/openssh/portable/ > dwopenssh.log 2>&1  
 num=$(grep openssh-.\*tar.gz dwopenssh.log|grep -v asc|wc -l)  
 if [ ${num} -le 0 ];then   
 echo "未获取到最新版本，将继续尝试下载，请耐心等待！" |tee -a ${LOGDIR}/${log\_name}  
 else   
 echo "获取到最新版本,开始下载！" |tee -a ${LOGDIR}/${log\_name}  
 opensshVer=$(grep openssh-.\*tar.gz dwopenssh.log|grep -v asc|tail -n1|awk -F'"' '{print $4}'| awk -F . '{print $1"."$2}')  
 while true;do  
 wget --no-check-certificate https://mirrors.aliyun.com/openssh/portable/${opensshVer}.tar.gz |tee -a ${LOGDIR}/${log\_name}  
 if [ "$?" != 0 ];then  
 echo "未下载成功，将继续尝试下载，请耐心等待！" |tee -a ${LOGDIR}/${log\_name}  
 else   
 echo "下载成功,开始部署！" |tee -a ${LOGDIR}/${log\_name}  
 upgrade\_openssh  
 break  
 fi  
 done  
 break  
 fi  
 done  
 else  
 log\_info "[\*] update\_openssh Configuration already!...!"  
 fi  
}  
  
#部署jemalloc  
ist\_jemalloc () {  
 log\_info "[-] ist\_jemalloc Configuring..." |tee -a ${LOGDIR}/${log\_name}  
 tar xf jemalloc-${jemallocVer}.tar.bz2 && cd jemalloc-${jemallocVer}  
 ./autogen.sh  
 make && make install  
 ln -s /usr/local/lib/libjemalloc.so.2 /usr/lib64/libjemalloc.so.1  
 echo '/usr/local/lib' > /etc/ld.so.conf.d/local.conf  
 ldconfig  
 cd ${sh\_path}  
 log\_info "[\*] ist\_jemalloc successful!" |tee -a ${LOGDIR}/${log\_name}  
}  
  
check\_jemalloc () {  
 log\_name=ist\_jemalloc.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 if test -d /usr/local/include/jemalloc;then  
 blue "jemalloc已部署"  
 log\_info "[-] jemalloc已部署!" |tee -a ${LOGDIR}/${log\_name}  
 log\_info "[-] 设置jemalloc软链接!" |tee -a ${LOGDIR}/${log\_name}  
 ln -sfv /usr/local/lib/libjemalloc.so.2 /usr/lib64/libjemalloc.so.1  
 else  
 log\_info "[-] 安装 autogen autoconf!" |tee -a ${LOGDIR}/${log\_name}  
 yum -y install autogen autoconf  
 while true;do  
 log\_name=ist\_jemalloc.log  
 curl https://github.com/jemalloc/jemalloc/releases > dwjemalloc.log 2>&1  
 num=$(grep tar.bz2 dwjemalloc.log|wc -l)  
 if [ ${num} -le 0 ];then   
 echo "未获取到最新版本，将继续尝试下载，请耐心等待！" |tee -a ${LOGDIR}/${log\_name}  
 else   
 echo "获取到最新版本,开始下载！" |tee -a ${LOGDIR}/${log\_name}  
 jemallocVer=$(grep tar.bz2 dwjemalloc.log |head -n1|awk -F'"' '{print $2}'|awk -F / '{print $6}')  
 while true;do  
 wget --no-check-certificate https://github.com/jemalloc/jemalloc/releases/download/${jemallocVer}/jemalloc-${jemallocVer}.tar.bz2 |tee -a ${LOGDIR}/${log\_name}  
 if [ "$?" != 0 ];then  
 echo "未下载成功，将继续尝试下载，请耐心等待！" |tee -a ${LOGDIR}/${log\_name}  
 else   
 echo "下载成功,开始部署！" |tee -a ${LOGDIR}/${log\_name}  
 ist\_jemalloc  
 break  
 fi  
 done  
 break  
 fi  
 done  
 fi  
 else  
 log\_info "[\*] ist\_jemalloc Configuration already!...!"  
 fi  
}  
  
#部署docker  
ist\_docker(){  
 log\_name=ist\_docker.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] install docker " |tee -a ${LOGDIR}/${log\_name}  
 yum install -y yum-utils device-mapper-persistent-data lvm2 |tee -a ${LOGDIR}/${log\_name}  
 yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo |tee -a ${LOGDIR}/${log\_name}  
 yum makecache fast |tee -a ${LOGDIR}/${log\_name}  
 yum -y install docker-ce |tee -a ${LOGDIR}/${log\_name}  
 yum -y install docker-compose |tee -a ${LOGDIR}/${log\_name}  
 #配置加速  
 mkdir -p /etc/docker  
 sudo tee /etc/docker/daemon.json <<-'EOF'  
{  
 "registry-mirrors": ["https://lb3cvacp.mirror.aliyuncs.com"],  
 "exec-opts": ["native.cgroupdriver=systemd"],  
 "log-driver": "json-file",  
 "log-opts": {  
 "max-size": "100m"  
 },  
 "storage-driver": "overlay2",  
 "storage-opts": [ "overlay2.override\_kernel\_check=true"]  
}  
EOF  
 sudo systemctl daemon-reload |tee -a ${LOGDIR}/${log\_name}  
 systemctl start docker && systemctl enable docker && systemctl status docker |tee -a ${LOGDIR}/${log\_name}  
 log\_info "[\*] Docker successfully installed!........." |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] Docker is already installed!........."  
 fi  
}  
  
#swap配置  
new\_swap() {  
 log\_name=new\_swap.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 if [ "${CHECK\_RAM}" -le '2' ]; then  
 log\_info "[-] new\_swap Configuring... " |tee -a ${LOGDIR}/${log\_name}  
 dd if=/dev/zero of=/swapfile bs=1024 count=1048576 |tee -a ${LOGDIR}/${log\_name}  
 chmod 600 /swapfile |tee -a ${LOGDIR}/${log\_name}  
 mkswap /swapfile |tee -a ${LOGDIR}/${log\_name}  
 swapon /swapfile |tee -a ${LOGDIR}/${log\_name}  
 echo '/swapfile swap swap defaults 0 0' >> /etc/fstab  
 echo '# Swap' >> /etc/sysctl.conf  
 echo 'vm.swappiness = 10' >> /etc/sysctl.conf  
 sysctl -p|tee -a ${LOGDIR}/${log\_name}  
 sysctl -n vm.swappiness |tee -a ${LOGDIR}/${log\_name}  
 echo -e "\r${RGB\_SUCCESS}Configuration Success${RGB\_END}"  
 log\_info "[\*] Swap Configuration successfully!........." |tee -a ${LOGDIR}/${log\_name}  
 else  
 echo -e "${RGB\_SUCCESS}Skip, no configuration needed${RGB\_END}" |tee -a ${LOGDIR}/${log\_name}  
 fi  
 else  
 log\_info "[\*] Swap Configuration already!........."  
 fi  
}  
  
open\_bbr() {  
 log\_name=open\_bbr.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] open\_bbr Configuring... " |tee -a ${LOGDIR}/${log\_name}  
 echo -en "${RGB\_WAIT}Configuring...${RGB\_END}"  
 echo "# BBR" >> /etc/sysctl.conf  
 echo "net.core.default\_qdisc=fq" >> /etc/sysctl.conf  
 echo "net.ipv4.tcp\_congestion\_control=bbr" >> /etc/sysctl.conf  
 sysctl -p |tee -a ${LOGDIR}/${log\_name}  
 sysctl -n net.ipv4.tcp\_congestion\_control |tee -a ${LOGDIR}/${log\_name}  
 lsmod | grep bbr |tee -a ${LOGDIR}/${log\_name}  
 log\_info "[\*] open\_bbr Configuration successfully!........." |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] open\_bbr Configuration already!........."  
 fi  
}  
  
disable\_software() {  
 log\_name=disable\_software.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] disable\_software Configuring... " |tee -a ${LOGDIR}/${log\_name}  
 setenforce 0 |tee -a ${LOGDIR}/${log\_name}  
 sed -i 's/^SELINUX=.\*$/SELINUX=disabled/' /etc/selinux/config  
 systemctl disable firewalld.service |tee -a ${LOGDIR}/${log\_name}  
 systemctl stop firewalld.service |tee -a ${LOGDIR}/${log\_name}  
   
 systemctl stop NetworkManager |tee -a ${LOGDIR}/${log\_name}  
 systemctl disable NetworkManager |tee -a ${LOGDIR}/${log\_name}  
   
 systemctl stop postfix.service  
 systemctl disable postfix.service  
 log\_info "[\*] disable\_software successfully!........." |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] disable\_software already!........." |tee -a ${LOGDIR}/${log\_name}  
 fi  
}  
  
time\_zone() {  
 # (1) 时区设置东8区  
 log\_name=time\_zone.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] time\_zone Configuring... " |tee -a ${LOGDIR}/${log\_name}  
 rm -rf /etc/localtime |tee -a ${LOGDIR}/${log\_name}  
 ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime |tee -a ${LOGDIR}/${log\_name}  
 ls -ln /etc/localtime |tee -a ${LOGDIR}/${log\_name}  
 echo -e "\r${RGB\_SUCCESS}Configuration Success${RGB\_END}"  
 log\_info "[\*] time\_zone Configuring successfully!........." |tee -a ${LOGDIR}/${log\_name}  
 # (2) 时间同步软件安装  
 log\_info "[-] Ntp Time Configuring... " |tee -a ${LOGDIR}/${log\_name}  
 sudo tee /etc/chrony.conf << 'EOF'  
pool s1a.time.edu.cn iburst maxsources 1  
pool ntp.aliyun.com iburst maxsources 4  
keyfile /etc/chrony.keys  
driftfile /var/lib/chrony/chrony.drift  
logdir /var/log/chrony  
maxupdateskew 100.0  
rtcsync  
makestep 1.0 3  
EOF  
 systemctl enable chronyd && systemctl restart chronyd && systemctl status chronyd -l |tee -a ${LOGDIR}/${log\_name}  
  
 # 将当前的 UTC 时间写入硬件时钟 (硬件时间默认为UTC)  
 sudo timedatectl set-local-rtc 0 |tee -a ${LOGDIR}/${log\_name}  
 # 启用NTP时间同步：  
 timedatectl set-ntp yes |tee -a ${LOGDIR}/${log\_name}  
 # 时间服务器连接查看  
 chronyc tracking |tee -a ${LOGDIR}/${log\_name}  
 # 手动校准-强制更新时间  
 # chronyc -a makestep  
 # 硬件时钟(系统时钟同步硬件时钟 )  
 hwclock --systohc |tee -a ${LOGDIR}/${log\_name}  
 # 备用方案: 采用 ntpdate 进行时间同步 ntpdate s1a.time.edu.cn  
   
 # (3) 重启依赖于系统时间的服务  
 sudo systemctl restart rsyslog.service crond.service |tee -a ${LOGDIR}/${log\_name}  
   
 log\_info "[\*] Tie ntp time configuration modifiy successful! restarting chronyd rsyslog.service crond.service........."  
 timedatectl |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] Tie ntp time configuration modifiy already!........"  
 fi  
}  
  
os\_Security () {  
 log\_name=os\_Security.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] 删除潜在威胁文件... " |tee -a ${LOGDIR}/${log\_name}  
 find / -maxdepth 3 -name hosts.equiv | xargs rm -rf  
 find / -maxdepth 3 -name .netrc | xargs rm -rf  
 find / -maxdepth 3 -name .rhosts | xargs rm -rf  
 log\_info "[\*] 删除潜在威胁文件 successful!........" |tee -a ${LOGDIR}/${log\_name}  
   
 # (1) 系统用户及其终端核查配置  
 log\_info "[-] 锁定或者删除多余的系统账户以及创建低权限用户"  
 # cat /etc/passwd | cut -d ":" -f 1 | tr '\n' ' '  
 defaultuser=(root bin daemon adm lp sync shutdown halt mail operator games ftp nobody systemd-network dbus polkitd sshd postfix chrony ntp rpc rpcuser nfsnobody)  
 for i in $(cat /etc/passwd | cut -d ":" -f 1,7);do  
 flag=0; name=${i%%:\*}; terminal=${i##\*:}  
 if [[ "${terminal}" == "/bin/bash" || "${terminal}" == "/bin/sh" ]];then  
 log\_warning "${i} 用户，shell终端为 /bin/bash 或者 /bin/sh"  
 fi  
 for j in ${defaultuser[@]};do  
 if [[ "${name}" == "${j}" ]];then  
 flag=1  
 break;  
 fi  
 done  
 if [[ $flag -eq 0 ]];then  
 log\_warning "${i} 非默认用户"  
 fi  
 done  
 cp -a /etc/shadow ${BACKUPDIR}/shadow-${EXECTIME}.bak  
 #锁定与设备运行、维护等工作无关的账号  
 log\_info "[\*] 锁定与设备运行、维护等工作无关的账号"  
  
 log\_info "[\*] 锁定帐号完成"  
   
 # (2) 用户密码设置和口令策略设置  
 log\_info "[-] 配置满足策略的root管理员密码 "  
 echo "root:${ROOTPASS}" | chpasswd  
   
 log\_info "[-] 配置满足策略的app普通用户密码(根据需求配置)"  
 groupadd application  
 useradd -m -s /bin/bash -c "application primary user" -g application app   
 echo "app:${APPPASS}" | chpasswd  
   
 #log\_info "[-] 强制用户在下次登录时更改密码 "  
 #chage -d 0 -m 0 -M 90 -W 15 root && passwd --expire root   
 #chage -d 0 -m 0 -M 90 -W 15 app && passwd --expire app  
 #chage -d 0 -m 0 -M 90 -W 15 ${DefaultUser} && passwd --expire ${DefaultUser}  
  
 else  
 log\_info "[\*] os\_Security already!........"  
 fi  
}  
  
function AccountPolicy(){  
 log\_name=AccountPolicy.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 # 设置口令长度最小值和密码复杂度策略  
 log\_info "[-] 设置口令长度最小值和密码复杂度策略" |tee -a ${LOGDIR}/${log\_name}  
 #大写字母、小写字母、数字、特殊字符 4选3，密码最小长度为8，登陆尝试三次，可自行修改  
 # 修改system-auth  
 egrep -q "^\s\*password\s\*(requisite|required)\s\*pam\_cracklib.so.\*$" /etc/pam.d/system-auth && sed -ri "s/^\s\*password\s\*(requisite|required)\s\*pam\_cracklib.so.\*$/\password requisite pam\_cracklib.so try\_first\_pass retry=3 minlen=8 dcredit=-1 ocredit=-1 lcredit=-1/" /etc/pam.d/system-auth || echo "password requisite pam\_cracklib.so try\_first\_pass retry=3 minlen=8 dcredit=-1 ocredit=-1 lcredit=-1" >> /etc/pam.d/system-auth  
 # 修改password-auth  
 egrep -q "^\s\*password\s\*(requisite|required)\s\*pam\_cracklib.so.\*$" /etc/pam.d/password-auth && sed -ri "s/^\s\*password\s\*(requisite|required)\s\*pam\_cracklib.so.\*$/\password requisite pam\_cracklib.so try\_first\_pass retry=3 minlen=8 dcredit=-1 ocredit=-1 lcredit=-1/" /etc/pam.d/password-auth || echo "password requisite pam\_cracklib.so try\_first\_pass retry=3 minlen=8 dcredit=-1 ocredit=-1 lcredit=-1" >> /etc/pam.d/password-auth  
 # 修改login.defs  
 egrep -q "^\s\*PASS\_MIN\_LEN\s+\S\*(\s\*#.\*)?\s\*$" /etc/login.defs && sed -ri "s/^(\s\*)PASS\_MIN\_LEN\s+\S\*(\s\*#.\*)?\s\*$/\PASS\_MIN\_LEN 12/" /etc/login.defs || echo "PASS\_MIN\_LEN 12" >> /etc/login.defs  
   
   
 # 设置口令生存周期  
 #log\_info "[-] 设置口令生存周期" |tee -a ${LOGDIR}/${log\_name}  
 #口令生成周期最小0天最大180天预警7前天  
 #egrep -q "^\s\*PASS\_MAX\_DAYS\s+\S\*(\s\*#.\*)?\s\*$" /etc/login.defs && sed -ri "s/^(\s\*)PASS\_MAX\_DAYS\s+\S\*(\s\*#.\*)?\s\*$/\PASS\_MAX\_DAYS 180/" /etc/login.defs || echo "PASS\_MAX\_DAYS 180" >> /etc/login.defs  
 #egrep -q "^\s\*PASS\_MIN\_DAYS\s+\S\*(\s\*#.\*)?\s\*$" /etc/login.defs && sed -ri "s/^(\s\*)PASS\_MIN\_DAYS\s+\S\*(\s\*#.\*)?\s\*$/\PASS\_MIN\_DAYS 14/" /etc/login.defs || echo "PASS\_MIN\_DAYS 0" >> /etc/login.defs  
 #egrep -q "^\s\*PASS\_WARN\_AGE\s+\S\*(\s\*#.\*)?\s\*$" /etc/login.defs && sed -ri "s/^(\s\*)PASS\_WARN\_AGE\s+\S\*(\s\*#.\*)?\s\*$/\PASS\_WARN\_AGE 14/" /etc/login.defs || echo "PASS\_WARN\_AGE 7" >> /etc/login.defs  
   
 # 密码重复使用次数限制  
 #log\_info "[-] 密码重复使用次数限制" |tee -a ${LOGDIR}/${log\_name}  
 #记住3次已使用的密码  
 #if [[ ! -f "/etc/security/opasswd" || "$(ls -l /etc/security/opasswd | egrep -c '\-rw\-\-\-\-\-\-\-')" != "1" ]];then  
 # 手动创建/etc/security/opasswd，解决首次登录修改密码时提示"passwd: Authentication token manipulation error"  
 #mv /etc/security/opasswd /etc/security/opasswd.old > /dev/null 2>&1  
 #touch /etc/security/opasswd  
 #chown root:root /etc/security/opasswd  
 #chmod +600 /etc/security/opasswd  
 #fi  
 # 修改system-auth  
 #egrep -q "^\s\*password\s\*sufficient\s\*pam\_unix.so.\*$" /etc/pam.d/system-auth && sed -ri "s/^\s\*password\s\*sufficient\s\*pam\_unix.so.\*$/\password sufficient pam\_unix.so sha512 shadow nullok try\_first\_pass use\_authtok remember=3/" /etc/pam.d/system-auth || echo "password sufficient pam\_unix.so sha512 shadow nullok try\_first\_pass use\_authtok remember=3" >> /etc/pam.d/system-auth  
 # 修改password-auth  
 #egrep -q "^\s\*password\s\*sufficient\s\*pam\_unix.so.\*$" /etc/pam.d/password-auth && sed -ri "s/^\s\*password\s\*sufficient\s\*pam\_unix.so.\*$/\password sufficient pam\_unix.so sha512 shadow nullok try\_first\_pass use\_authtok remember=3/" /etc/pam.d/password-auth || echo "password sufficient pam\_unix.so sha512 shadow nullok try\_first\_pass use\_authtok remember=3" >> /etc/pam.d/password-auth  
   
   
 # 用户认证失败次数限制  
 log\_info "[-] 用户认证失败次数限制" |tee -a ${LOGDIR}/${log\_name}  
 #连续登录失败5次锁定帐号5分钟  
 sed -ri "/^\s\*auth\s+required\s+pam\_tally2.so\s+.+(\s\*#.\*)?\s\*$/d" /etc/pam.d/sshd /etc/pam.d/login /etc/pam.d/system-auth /etc/pam.d/password-auth  
 sed -ri '1a auth required pam\_tally2.so deny=5 unlock\_time=300 even\_deny\_root root\_unlock\_time=30' /etc/pam.d/sshd /etc/pam.d/login /etc/pam.d/system-auth /etc/pam.d/password-auth  
 egrep -q "^\s\*account\s+required\s+pam\_tally2.so\s\*(\s\*#.\*)?\s\*$" /etc/pam.d/sshd || sed -ri '/^password\s+.+(\s\*#.\*)?\s\*$/i\account required pam\_tally2.so' /etc/pam.d/sshd  
 egrep -q "^\s\*account\s+required\s+pam\_tally2.so\s\*(\s\*#.\*)?\s\*$" /etc/pam.d/login || sed -ri '/^password\s+.+(\s\*#.\*)?\s\*$/i\account required pam\_tally2.so' /etc/pam.d/login  
 egrep -q "^\s\*account\s+required\s+pam\_tally2.so\s\*(\s\*#.\*)?\s\*$" /etc/pam.d/system-auth || sed -ri '/^account\s+required\s+pam\_permit.so\s\*(\s\*#.\*)?\s\*$/a\account required pam\_tally2.so' /etc/pam.d/system-auth  
 egrep -q "^\s\*account\s+required\s+pam\_tally2.so\s\*(\s\*#.\*)?\s\*$" /etc/pam.d/password-auth || sed -ri '/^account\s+required\s+pam\_permit.so\s\*(\s\*#.\*)?\s\*$/a\account required pam\_tally2.so' /etc/pam.d/password-auth  
 log\_info "[\*] AccountPolicy successful configuration!........" |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] AccountPolicy already configuration!........"  
 fi   
}  
  
  
custom\_profile() {  
 log\_name=custom\_profile.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] custom profile configuration... " |tee -a ${LOGDIR}/${log\_name}  
 echo -en "${RGB\_WAIT}Configuring...${RGB\_END}"  
 cat > /etc/profile.d/centos7init.sh << EOF  
alias vi=vim  
alias tf='tail -f'  
alias md5='md5sum'  
GREP\_OPTIONS="--color=auto"  
alias l='ls -AFhlt'  
alias grep='grep --color'  
alias egrep='egrep --color'  
alias fgrep='fgrep --color'  
export HISTTIMEFORMAT="[%Y-%m-%d %H:%M:%S]: "  
PS1='[\[\e[1;35m\]\u\[\e[1;33m\]@\[\e[1;32m\]\h \[\e[1;0m\]\e[4m`pwd`\e[m\e[1;37m \[\e[1;0m\]\[\e[1;34m\]\t\[\e[1;0m\]]\n\[\e[1;31m\]\$\[\e[0m\] '  
EOF  
  
cat >> /root/.vimrc << EOF  
set tabstop=4  
set shiftwidth=4  
set expandtab  
set cursorline  
set mouse=a  
set selection=exclusive  
set selectmode=mouse,key  
syntax on  
set autoindent  
set number  
EOF  
 cat /etc/profile.d/centos7init.sh |tee -a ${LOGDIR}/${log\_name}  
 log\_info "[\*] custom profile successful configuration!........" |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] custom profile already configuration!........"  
 fi  
}  
  
  
  
adjust\_ulimit() {  
 log\_name=adjust\_ulimit.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] adjust ulimit Configuring... " |tee -a ${LOGDIR}/${log\_name}  
 sed -i '/^# End of file/,$d' /etc/security/limits.conf  
 cat >> /etc/security/limits.conf <<EOF  
# End of file  
\* soft core unlimited  
\* hard core unlimited  
\* soft nproc 1000000  
\* hard nproc 1000000  
\* soft nofile 1000000  
\* hard nofile 1000000  
root soft core unlimited  
root hard core unlimited  
root soft nproc 1000000  
root hard nproc 1000000  
root soft nofile 1000000  
root hard nofile 1000000  
EOF  
 cat /etc/security/limits.conf |tee -a ${LOGDIR}/${log\_name}  
 log\_info "[\*] adjust\_ulimit Configuration successful!........" |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] adjust\_ulimit already Configuration!........"  
 fi  
}  
  
kernel\_optimum() {  
 log\_name=kernel\_optimum.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] kernel optimum Configuring... " |tee -a ${LOGDIR}/${log\_name}  
 [ ! -e "/etc/sysctl.conf\_bak" ] && /bin/mv /etc/sysctl.conf{,\_bak}  
 cat > /etc/sysctl.conf << EOF  
#决定检查过期多久邻居条目  
net.ipv4.neigh.default.gc\_stale\_time=120  
#使用 arp\_announce / arp\_ignore 解决 ARP 映射问题  
net.ipv4.conf.default.arp\_announce = 2  
net.ipv4.conf.all.arp\_announce=2  
net.ipv4.conf.lo.arp\_announce=2  
  
# 开启恶意 icmp 错误消息保护  
net.ipv4.icmp\_ignore\_bogus\_error\_responses = 1  
  
#启用 timewait 快速回收  
net.ipv4.tcp\_tw\_recycle = 1  
  
# Controls source route verification  
#开启反向路径过滤  
net.ipv4.conf.all.rp\_filter = 1  
net.ipv4.conf.default.rp\_filter = 1  
  
net.ipv4.ip\_nonlocal\_bind = 1  
net.ipv4.ip\_forward = 1  
# 确保无人能修改路由表  
net.ipv4.conf.all.accept\_redirects = 0  
net.ipv4.conf.default.accept\_redirects = 0  
net.ipv4.conf.all.secure\_redirects = 0  
net.ipv4.conf.default.secure\_redirects = 0  
#处理无源路由的包  
net.ipv4.conf.all.accept\_source\_route = 0   
net.ipv4.conf.default.accept\_source\_route = 0  
# 避免放大攻击  
net.ipv4.icmp\_echo\_ignore\_broadcasts = 1  
net.ipv4.conf.all.log\_martians = 1   
net.ipv4.conf.default.log\_martians = 1  
net.ipv4.conf.all.promote\_secondaries = 1  
net.ipv4.conf.default.promote\_secondaries = 1  
  
# Controls the use of TCP syncookies  
  
#core 文件名中添加 pid 作为扩展名  
kernel.core\_uses\_pid = 1  
# Number of pid\_max  
kernel.pid\_max = 1000000  
# 开启 SYN 洪水攻击保护  
net.ipv4.tcp\_syncookies = 1  
  
# Controls the maximum size of a message, in bytes  
# Controls the default maxmimum size of a mesage queue  
# Controls the maximum shared segment size, in bytes  
# Controls the maximum number of shared memory segments, in pages  
#修改消息队列长度  
kernel.msgmnb = 65536  
kernel.msgmax = 65536  
#设置最大内存共享段大小 bytes  
kernel.shmmax = 68719476736  
kernel.shmall = 4294967296  
#关闭 sysrq 功能  
kernel.sysrq = 1  
  
kernel.softlockup\_panic = 1  
kernel.printk = 5  
  
# TCP kernel paramater  
net.ipv4.tcp\_mem = 94500000 915000000 927000000  
net.ipv4.tcp\_rmem = 4096 87380 4194304  
net.ipv4.tcp\_wmem = 4096 16384 4194304  
net.ipv4.tcp\_window\_scaling = 1  
net.ipv4.tcp\_sack = 1  
  
# Socket buffer  
net.core.wmem\_default = 8388608  
net.core.rmem\_default = 8388608  
net.core.rmem\_max = 16777216  
net.core.wmem\_max = 16777216  
#每个网络接口接收数据包的速率比内核处理这些包的速率快时，允许送到队列的数据包的最大数目  
net.core.netdev\_max\_backlog = 32768  
net.core.somaxconn = 65535  
net.core.optmem\_max = 81920  
  
# TCP conn  
#未收到客户端确认信息的连接请求的最大值  
net.ipv4.tcp\_max\_syn\_backlog = 262144  
#内核放弃建立连接之前发送 SYN 包的数量  
net.ipv4.tcp\_syn\_retries = 1  
net.ipv4.tcp\_retries1 = 3  
net.ipv4.tcp\_retries2 = 15  
  
# TCP conn reuse  
net.ipv4.tcp\_timestamps = 0  
#开启重用。允许将 TIME-WAIT sockets 重新用于新的 TCP 连接  
net.ipv4.tcp\_tw\_reuse = 1  
net.ipv4.tcp\_fin\_timeout = 5  
net.ipv4.tcp\_max\_tw\_buckets = 7000  
#限制仅仅是为了防止简单的 DoS 攻击  
net.ipv4.tcp\_max\_orphans = 3276800  
#内核放弃建立连接之前发送 SYNACK 包的数量  
net.ipv4.tcp\_synack\_retries = 1  
  
# keepalive conn  
net.ipv4.tcp\_keepalive\_time = 300  
net.ipv4.tcp\_keepalive\_intvl = 30  
net.ipv4.tcp\_keepalive\_probes = 3  
#允许系统打开的端口范围  
net.ipv4.ip\_local\_port\_range = 1024 65535  
  
net.ipv6.neigh.default.gc\_thresh3 = 4096  
net.ipv4.neigh.default.gc\_thresh3 = 4096  
  
#修改防火墙表大小，默认 65536  
net.netfilter.nf\_conntrack\_max=655350  
net.netfilter.nf\_conntrack\_tcp\_timeout\_established=1200  
EOF  
 sysctl -p |tee -a ${LOGDIR}/${log\_name}  
 cat /etc/sysctl.conf |tee -a ${LOGDIR}/${log\_name}  
 log\_info "[\*] kernel optimum Configuration successful!........" |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] kernel optimum already Configuration!........"  
 fi  
}  
  
#open\_ipv6 （可选，这里不开启）  
open\_ipv6() {  
 log\_name=open\_ipv6.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] open ipv6 Configuring... " |tee -a ${LOGDIR}/${log\_name}  
 echo -en "${RGB\_WAIT}Configuring...${RGB\_END}"  
 echo '# IPV6' >> /etc/sysctl.conf  
 echo 'net.ipv6.conf.all.disable\_ipv6=0' >> /etc/sysctl.conf  
 echo 'net.ipv6.conf.default.disable\_ipv6=0' >> /etc/sysctl.conf  
 echo 'net.ipv6.conf.lo.disable\_ipv6=0' >> /etc/sysctl.conf  
 sysctl -p |tee -a ${LOGDIR}/${log\_name}  
 cat /etc/sysctl.conf |tee -a ${LOGDIR}/${log\_name}  
 echo -e "\r${RGB\_SUCCESS}Configuration Success${RGB\_END}"  
 log\_info "[\*] open\_ipv6 Configuration successful!........" |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] open\_ipv6 already Configuration!........"  
 fi  
}   
  
  
remove\_users() {  
 log\_name=remove\_users.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] remove users Configuring... " |tee -a ${LOGDIR}/${log\_name}  
 for u in adm lp sync shutdown halt mail operator games ftp news uucp gopher  
 do  
 userdel ${u} |tee -a ${LOGDIR}/${log\_name}  
 done  
 cut -d : -f 1 /etc/passwd |tee -a ${LOGDIR}/${log\_name}  
 for g in adm lp mail news uucp games gopher mailnull floppy dip pppusers popusers slipusers daemon  
 do  
 groupdel ${g} |tee -a ${LOGDIR}/${log\_name}  
 done  
 cat /etc/group |tee -a ${LOGDIR}/${log\_name}  
 log\_info "[\*] remove users successful!........" |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] remove users already Configuration!........"  
 fi  
}  
  
sys\_permissions() {  
 log\_name=sys\_permissions.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] sys permissions Configuring... " |tee -a ${LOGDIR}/${log\_name}  
 chmod 644 /etc/passwd |tee -a ${LOGDIR}/${log\_name}  
 chmod 644 /etc/group |tee -a ${LOGDIR}/${log\_name}  
 chmod 000 /etc/shadow |tee -a ${LOGDIR}/${log\_name}  
 chmod 000 /etc/gshadow |tee -a ${LOGDIR}/${log\_name}  
 ls -la /etc/passwd |tee -a ${LOGDIR}/${log\_name}  
 ls -la /etc/group |tee -a ${LOGDIR}/${log\_name}  
 ls -la /etc/shadow |tee -a ${LOGDIR}/${log\_name}  
 ls -la /etc/gshadow |tee -a ${LOGDIR}/${log\_name}  
 log\_info "[\*] sys permissions successful!........" |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] sys permissions already Configuration!........"  
 fi  
}  
  
change\_useradd() {  
 log\_name=change\_useradd.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] change\_useradd Configuring... " |tee -a ${LOGDIR}/${log\_name}  
 sed -i 's/^INACTIVE.\*$/INACTIVE=180/' /etc/default/useradd  
 cat /etc/default/useradd |tee -a ${LOGDIR}/${log\_name}  
 log\_info "[\*] change\_useradd successful!........" |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] change\_useradd already Configuration!........"  
 fi  
}  
  
#SSHD 服务安全加固设置以及网络登陆Banner设置  
sec\_ssh() {  
 log\_name=sec\_ssh.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] sec\_ssh Configuring... " |tee -a ${LOGDIR}/${log\_name}  
   
 # 严格模式  
 sudo egrep -q "^\s\*StrictModes\s+.+$" /etc/ssh/sshd\_config && sed -ri "s/^(#)?\s\*StrictModes\s+.+$/StrictModes yes/" /etc/ssh/sshd\_config || echo "StrictModes yes" >> /etc/ssh/sshd\_config  
   
 # 默认的监听端口更改  
 if [ -e ${SSHPORT} ];then export SSHPORT=22;fi  
 sudo egrep -q "^\s\*Port\s+.+$" /etc/ssh/sshd\_config && sed -ri "s/^(#)?\s\*Port\s+.+$/Port ${SSHPORT}/" /etc/ssh/sshd\_config || echo "Port ${SSHPORT}" >> /etc/ssh/sshd\_config  
   
 # 禁用X11转发以及端口转发  
 sudo egrep -q "^\s\*X11Forwarding\s+.+$" /etc/ssh/sshd\_config && sed -ri "s/^(#)?\s\*X11Forwarding\s+.+$/X11Forwarding no/" /etc/ssh/sshd\_config ||echo "X11Forwarding no" >> /etc/ssh/sshd\_config  
 sudo egrep -q "^\s\*X11UseLocalhost\s+.+$" /etc/ssh/sshd\_config && sed -ri "s/^(#)?\s\*X11UseLocalhost\s+.+$/X11UseLocalhost yes/" /etc/ssh/sshd\_config || echo "X11UseLocalhost yes" >> /etc/ssh/sshd\_config  
 sudo egrep -q "^\s\*AllowTcpForwarding\s+.+$" /etc/ssh/sshd\_config && sed -ri "s/^(#)?\s\*AllowTcpForwarding\s+.+$/AllowTcpForwarding no/" /etc/ssh/sshd\_config || echo "AllowTcpForwarding no" >> /etc/ssh/sshd\_config  
 sudo egrep -q "^\s\*AllowAgentForwarding\s+.+$" /etc/ssh/sshd\_config && sed -ri "s/^(#)?\s\*AllowAgentForwarding\s+.+$/AllowAgentForwarding no/" /etc/ssh/sshd\_config || echo "AllowAgentForwarding no" >> /etc/ssh/sshd\_config  
   
 # 关闭禁用用户的 .rhosts 文件 ~/.ssh/.rhosts 来做为认证: 缺省IgnoreRhosts yes   
 egrep -q "^(#)?\s\*IgnoreRhosts\s+.+$" /etc/ssh/sshd\_config && sed -ri "s/^(#)?\s\*IgnoreRhosts\s+.+$/IgnoreRhosts yes/" /etc/ssh/sshd\_config || echo "IgnoreRhosts yes" >> /etc/ssh/sshd\_config  
   
 # 禁止root远程登录（推荐配置-根据需求配置）  
 egrep -q "^\s\*PermitRootLogin\s+.+$" /etc/ssh/sshd\_config && sed -ri "s/^\s\*PermitRootLogin\s+.+$/PermitRootLogin no/" /etc/ssh/sshd\_config || echo "PermitRootLogin no" >> /etc/ssh/sshd\_config  
   
   
 sed -i 's/^#UseDNS.\*$/UseDNS no/' /etc/ssh/sshd\_config  
 sed -i 's/GSSAPIAuthentication.\*/GSSAPIAuthentication no/' /etc/ssh/sshd\_config  
 sed -i 's/^#LoginGraceTime.\*$/LoginGraceTime 60/' /etc/ssh/sshd\_config  
 sed -i 's/^#PermitEmptyPasswords.\*$/PermitEmptyPasswords no/' /etc/ssh/sshd\_config  
 sed -i 's/^#PubkeyAuthentication.\*$/PubkeyAuthentication yes/' /etc/ssh/sshd\_config  
 sed -i 's/^#MaxAuthTries.\*$/MaxAuthTries 3/' /etc/ssh/sshd\_config  
 sed -i "s/#ClientAliveInterval 0/ClientAliveInterval 30/g" /etc/ssh/sshd\_config   
 sed -i "s/#ClientAliveCountMax 3/ClientAliveCountMax 3/g" /etc/ssh/sshd\_config  
   
 # 登陆前后欢迎提示设置  
 egrep -q "^\s\*(banner|Banner)\s+\W+.\*$" /etc/ssh/sshd\_config && sed -ri "s/^\s\*(banner|Banner)\s+\W+.\*$/Banner \/etc\/issue/" /etc/ssh/sshd\_config || \  
 echo "Banner /etc/issue" >> /etc/ssh/sshd\_config  
 log\_info "[-] 远程SSH登录前后提示警告Banner设置" |tee -a ${LOGDIR}/${log\_name}  
 # SSH登录前后提示警告Banner设置  
 sudo tee /etc/issue <<'EOF'  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* [ 安全登陆 (Security Login) ] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Authorized only. All activity will be monitored and reported.By Security Center.  
EOF  
# SSH登录后提示Banner  
# 艺术字B格: http://www.network-science.de/ascii/  
sudo tee /etc/motd <<'EOF'  
################## [ 安全运维 (Security Operation) ] ####################  
 \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_\_\_  
 | | | |\_\_\_\_\_\_ | |\_\_\_\_\_| | | |\_\_\_\_\_] |\_\_\_\_\_\_  
 | | | |\_\_\_\_\_\_ | | | |\_\_\_\_\_| | \_\_\_\_\_\_|  
Login success. Please execute the commands and operation data after carefully!  
EOF  
 systemctl restart sshd.service |tee -a ${LOGDIR}/${log\_name}  
 cat /etc/ssh/sshd\_config |tee -a ${LOGDIR}/${log\_name}  
 echo -e "\r${RGB\_SUCCESS}Configuration Success${RGB\_END}"  
  
 # (5) 用户远程登录失败次数与终端超时设置   
 log\_info "[-] 用户远程连续登录失败10次锁定帐号5分钟包括root账号" |tee -a ${LOGDIR}/${log\_name}  
 cp /etc/pam.d/sshd ${BACKUPDIR}/sshd.bak  
 cp /etc/pam.d/login ${BACKUPDIR}/login.bak  
   
 # 远程登陆  
 sed -ri "/^\s\*auth\s+required\s+pam\_tally2.so\s+.+(\s\*#.\*)?\s\*$/d" /etc/pam.d/sshd   
 sed -ri '2a auth required pam\_tally2.so deny=10 unlock\_time=300 even\_deny\_root root\_unlock\_time=300' /etc/pam.d/sshd   
 # 宿主机控制台登陆(可选)  
 # sed -ri "/^\s\*auth\s+required\s+pam\_tally2.so\s+.+(\s\*#.\*)?\s\*$/d" /etc/pam.d/login  
 # sed -ri '2a auth required pam\_tally2.so deny=10 unlock\_time=300 even\_deny\_root root\_unlock\_time=300' /etc/pam.d/login  
   
 log\_info "[-] 设置登录超时时间为10分钟 "  
 egrep -q "^\s\*(export|)\s\*TMOUT\S\w+.\*$" /etc/profile && sed -ri "s/^\s\*(export|)\s\*TMOUT.\S\w+.\*$/export TMOUT=600\nreadonly TMOUT/" /etc/profile || echo -e "export TMOUT=600\nreadonly TMOUT" >> /etc/profile  
   
 egrep -q "^\s\*.\*ClientAliveInterval\s\w+.\*$" /etc/ssh/sshd\_config && sed -ri "s/^\s\*.\*ClientAliveInterval\s\w+.\*$/ClientAliveInterval 600/" /etc/ssh/sshd\_config || echo "ClientAliveInterval 600" >> /etc/ssh/sshd\_config  
   
 # (6) 切换用户日志记录和切换命令更改名称为SU  
 log\_info "[-] 切换用户日志记录和切换命令更改名称为SU " |tee -a ${LOGDIR}/${log\_name}  
 cp -a /etc/rsyslog.conf ${BACKUPDIR}/rsyslog.conf-${EXECTIME}.bak  
 egrep -q "^\s\*authpriv\.\\*\s+.+$" /etc/rsyslog.conf && sed -ri "s/^\s\*authpriv\.\\*\s+.+$/authpriv.\* \/var\/log\/secure/" /etc/rsyslog.conf || echo "authpriv.\* /var/log/secure" >> /etc/rsyslog.conf  
 egrep -q "^(\s\*)SULOG\_FILE\s+\S\*(\s\*#.\*)?\s\*$" /etc/login.defs && sed -ri "s/^(\s\*)SULOG\_FILE\s+\S\*(\s\*#.\*)?\s\*$/\SULOG\_FILE \/var\/log\/.history\/sulog/" /etc/login.defs || echo "SULOG\_FILE /var/log/.history/sulog" >> /etc/login.defs  
 egrep -q "^\s\*SU\_NAME\s+\S\*(\s\*#.\*)?\s\*$" /etc/login.defs && sed -ri "s/^(\s\*)SU\_NAME\s+\S\*(\s\*#.\*)?\s\*$/\SU\_NAME SU/" /etc/login.defs || echo "SU\_NAME SU" >> /etc/login.defs  
 mkdir -vp /usr/local/bin  
 cp /usr/bin/su ${BACKUPDIR}/su.bak  
 mv /usr/bin/su /usr/bin/SU  
 chmod 777 /var/log/.history   
 chattr -R +a /var/log/.history   
 chattr +a /var/log/.backups  
 log\_info "[\*] 切换用户日志记录和切换命令更改名称为SU successful!........" |tee -a ${LOGDIR}/${log\_name}  
   
 # (7) 记录安全事件日志  
 log\_info "[-] 记录安全事件日志" |tee -a ${LOGDIR}/${log\_name}  
 touch /var/log/.history/adm&>/dev/null; chmod 755 /var/log/.history/adm  
 semanage fcontext -a -t security\_t '/var/log/.history/adm'  
 restorecon -v '/var/log/.history/adm'&>/dev/null  
 egrep -q "^\s\*\\*\.err;kern.debug;daemon.notice\s+.+$" /etc/rsyslog.conf && sed "s/^\s\*\\*\.err;kern.debug;daemon.notice\s+.+$/\*.err;kern.debug;daemon.notice \/var\/log\/.history\/adm/" /etc/rsyslog.conf || echo "\*.err;kern.debug;daemon.notice /var/log/.history/adm" >> /etc/rsyslog.conf  
 log\_info "[\*] sec\_ssh successful!........" |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] sec\_ssh already Configuration!........"  
 fi  
   
}  
  
cmd\_audit(){  
 log\_name=cmd\_audit.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] 行为审计日志记录 Configuring" |tee -a ${LOGDIR}/${log\_name}  
 FILENAME="/var/log/usermonitor.log"  
 PATHNAME="/etc/profile"  
 FINDNAME="HISTORY\_FILE"  
 if [[ ! -f ${FILENAME} ]]  
 then  
 #创建行为审计日志文件  
 touch ${FILENAME}   
 #将日志文件的所有者改为权限低的用户NOBODY  
 chown nobody:nobody ${FILENAME}   
 #赋予所有用户对日志文件写的权限  
 chmod 002 ${FILENAME}  
 #使所有用户对日志文件只有追加权限  
 chattr +a ${FILENAME}  
 fi   
  
 if [[ `cat ${PATHNAME} | grep ${FINDNAME} | wc -l` < 1 ]]  
 then  
 cat >> ${PATHNAME} <<"EOF"  
export HISTORY\_FILE=/var/log/usermonitor.log  
export PROMPT\_COMMAND='{ date "+%y-%m-%d %T ##### $(who am i |awk "{print \$1\" \"\$2\" \"\$5}") #### $(id|awk "{print \$1}") #### $(history 1 | { read x cmd; echo "$cmd"; })"; } >> ${HISTORY\_FILE}'  
EOF  
 source ${PATHNAME}  
 log\_info "[\*] 行为审计日志记录 successful Configuration!........" |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] 行为审计日志记录 already Configuration!........"   
 fi  
 else  
 log\_info "[\*] 行为审计日志记录 already Configuration!........"   
 fi  
}  
  
  
#操作系统安全运维设置相关脚本  
os\_Operation () {  
 log\_name=os\_Operation.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] 操作系统安全运维设置相关脚本" |tee -a ${LOGDIR}/${log\_name}  
   
 # (0) 禁用ctrl+alt+del组合键对系统重启 (必须要配置,避坑)  
 log\_info "[-] 禁用控制台ctrl+alt+del组合键重启"  
 mv /usr/lib/systemd/system/ctrl-alt-del.target ${BACKUPDIR}/ctrl-alt-del.target-${EXECTIME}.bak  
   
 # (1) 设置文件删除回收站别名  
 log\_info "[-] 设置文件删除回收站别名(防止误删文件) "  
 sudo tee -a /etc/profile.d/alias.sh <<'EOF'  
# User specific aliases and functions  
# 删除回收站  
# find ~/.trash -delete  
# 删除空目录  
# find ~/.trash -type d -delete  
alias rm="sh /usr/local/bin/remove.sh"  
EOF  
 sudo tee /usr/local/bin/remove.sh <<'EOF'  
#!/bin/sh  
# 定义回收站文件夹目录.trash  
trash="/.trash"  
deltime=$(date +%Y%m%d-%H-%M-%S)  
TRASH\_DIR="${HOME}${trash}/${deltime}"  
# 建立回收站目录当不存在的时候  
if [ ! -e ${TRASH\_DIR} ];then  
 mkdir -p ${TRASH\_DIR}  
fi  
for i in $\*;do  
 if [ "$i" = "-rf" ];then continue;fi  
 # 防止误操作  
 if [ "$i" = "/" ];then echo '# Danger delete command, Not delete / directory!';exit -1;fi  
 #定义秒时间戳  
 STAMP=$(date +%s)  
 #得到文件名称(非文件夹)，参考man basename  
 fileName=$(basename $i)  
 #将输入的参数，对应文件mv至.trash目录，文件后缀，为当前的时间戳  
 mv $i ${TRASH\_DIR}/${fileName}.${STAMP}  
done  
EOF  
 sudo chmod +775 /usr/local/bin/remove.sh /etc/profile.d/alias.sh /etc/profile.d/history-record.sh  
 sudo chmod a+x /usr/local/bin/remove.sh /etc/profile.d/alias.sh /etc/profile.d/history-record.sh  
 source /etc/profile.d/alias.sh /etc/profile.d/history-record.sh  
 log\_info "[\*] 操作系统安全运维设置相关脚本 successful Configuration!........" |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] 操作系统安全运维设置相关脚本 already Configuration!........"   
 fi  
  
}  
  
lockout\_policy() {  
 log\_name=lockout\_policy.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] lockout\_policy Configuring...." |tee -a ${LOGDIR}/${log\_name}  
 [ ! -e "/etc/pam.d/system-auth\_bak" ] && /bin/mv /etc/pam.d/system-auth{,\_bak}  
 cat > /etc/pam.d/system-auth << EOF  
auth required pam\_env.so  
auth required pam\_faillock.so preauth silent audit deny=3 unlock\_time=300  
auth required pam\_faildelay.so delay=2000000  
auth [default=1 ignore=ignore success=ok] pam\_succeed\_if.so uid >= 1000 quiet  
auth [default=1 ignore=ignore success=ok] pam\_localuser.so  
auth sufficient pam\_unix.so nullok try\_first\_pass  
auth [default=die] pam\_faillock.so authfail audit deny=3 unlock\_time=300  
auth requisite pam\_succeed\_if.so uid >= 1000 quiet\_success  
auth sufficient pam\_sss.so forward\_pass  
auth required pam\_deny.so  
  
account required pam\_unix.so  
account sufficient pam\_localuser.so  
account sufficient pam\_succeed\_if.so uid < 1000 quiet  
account [default=bad success=ok user\_unknown=ignore] pam\_sss.so  
account required pam\_permit.so  
account required pam\_faillock.so  
  
password requisite pam\_pwquality.so try\_first\_pass local\_users\_only  
password sufficient pam\_unix.so sha512 shadow nullok try\_first\_pass use\_authtok  
password sufficient pam\_sss.so use\_authtok  
password required pam\_deny.so  
  
session optional pam\_keyinit.so revoke  
session required pam\_limits.so  
-session optional pam\_systemd.so  
session [success=1 default=ignore] pam\_succeed\_if.so service in crond quiet use\_uid  
session required pam\_unix.so  
session optional pam\_sss.so  
EOF  
 [ ! -e "/etc/pam.d/password-auth\_bak" ] && /bin/mv /etc/pam.d/password-auth{,\_bak}  
 cat > /etc/pam.d/password-auth << EOF  
auth required pam\_env.so  
auth required pam\_faillock.so preauth silent audit deny=3 unlock\_time=300  
auth required pam\_faildelay.so delay=2000000  
auth [default=1 ignore=ignore success=ok] pam\_succeed\_if.so uid >= 1000 quiet  
auth [default=1 ignore=ignore success=ok] pam\_localuser.so  
auth sufficient pam\_unix.so nullok try\_first\_pass  
auth [default=die] pam\_faillock.so authfail audit deny=3 unlock\_time=300  
auth requisite pam\_succeed\_if.so uid >= 1000 quiet\_success  
auth sufficient pam\_sss.so forward\_pass  
auth required pam\_deny.so  
  
account required pam\_unix.so  
account sufficient pam\_localuser.so  
account sufficient pam\_succeed\_if.so uid < 1000 quiet  
account [default=bad success=ok user\_unknown=ignore] pam\_sss.so  
account required pam\_permit.so  
account required pam\_faillock.so  
  
password requisite pam\_pwquality.so try\_first\_pass local\_users\_only  
password sufficient pam\_unix.so sha512 shadow nullok try\_first\_pass use\_authtok  
password sufficient pam\_sss.so use\_authtok  
password required pam\_deny.so  
  
session optional pam\_keyinit.so revoke  
session required pam\_limits.so  
-session optional pam\_systemd.so  
session [success=1 default=ignore] pam\_succeed\_if.so service in crond quiet use\_uid  
session required pam\_unix.so  
session optional pam\_sss.so  
EOF  
 systemctl restart sshd.service |tee -a ${LOGDIR}/${log\_name}  
 cat /etc/pam.d/system-auth |tee -a ${LOGDIR}/${log\_name}  
 cat /etc/pam.d/password-auth |tee -a ${LOGDIR}/${log\_name}  
 log\_info "[\*] lockout\_policy successful Configuration!........" |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] lockout\_policy already Configuration!........"   
 fi  
}  
  
#开机启动优化  
boot\_optimum () {  
 log\_name=boot\_optimum.log  
 succ\_num=$(grep successful ${LOGDIR}/${log\_name}|wc -l)  
 if [ ! -f ${LOGDIR}/${log\_name} -o ${succ\_num} -eq 0 ];then  
 log\_info "[-] boot\_optimum Configuring...." |tee -a ${LOGDIR}/${log\_name}  
 for i in acpid gpm lvm2-monitor anacron haldaemon mcstrans oddjobd setroubleshoot atd halt mdmonitor pand single auditd hidd mdmpd pcscd smartd hplip messagebus avahi-daemon ip6tables microcode\_ctl snmptrapd avahi-dnsconfd ipmi multipathd psacct bluetooth rawdevices svnserve conman irda netconsole rdisc cpuspeed irqbalance netfs readahead\_early tcsd iscsi netplugd readahead\_later winbind cups iscsid restorecond wpa\_supplicant cups-config-daemon kdump NetworkManager rpcgssd dnsmasq killall nfs rpcidmapd ypbind dund krb524 nfslock rpcsvcgssd yum-updatesd firstboot lm\_sensors nscd saslauthd  
 do  
 chkconfig $i off > /dev/null 2>&1  
 done  
 log\_info "[\*] boot\_optimum successful Configuration!........" |tee -a ${LOGDIR}/${log\_name}  
 else  
 log\_info "[\*] boot\_optimum already Configuration!........"   
 fi  
}  
  
reboot\_os() {  
 echo -e "\n${RGB\_WARNING}Please restart the server and see if the services start up fine.${RGB\_END}"  
 echo -en "${RGB\_WARNING}Do you want to restart OS ? [y/n]: ${RGB\_END}"  
 while :; do  
 read REBOOT\_STATUS  
 if [[ ! "${REBOOT\_STATUS}" =~ ^[y,n]$ ]]; then  
 echo -en "${RGB\_DANGER}Input error, please only input 'y' or 'n': ${RGB\_END}"  
 else  
 break  
 fi  
 done  
 [ "${REBOOT\_STATUS}" == 'y' ] && reboot  
}  
  
#菜单  
menu1()  
{  
 clear  
 cat <<EOF  
----------------------------------------  
|\*\*\*\* 欢迎使用cetnos7初始化脚本 \*\*\*\*|  
----------------------------------------  
1. 基础配置  
2. 内核升级  
3. 系统优化  
4. openssl&openssh升级  
5. 部署docker  
6. 部署jemalloc  
7. 退出  
EOF  
 read -p "please enter your choice[1-7]:" num1  
}  
  
#控制函数  
main()  
{  
 menu1  
 case $num1 in  
 1)  
 tool\_info  
 check\_root  
 check\_os  
 os\_dns  
 add\_repo  
 add\_softbase  
 ;;  
 2)  
 upgrade\_kernel  
 ;;  
 3)  
 new\_swap  
 open\_bbr  
 disable\_software  
 time\_zone  
 os\_Security  
 AccountPolicy  
 custom\_profile  
 adjust\_ulimit  
 kernel\_optimum  
 remove\_users  
 sys\_permissions  
 change\_useradd  
 sec\_ssh  
 cmd\_audit  
 os\_Operation  
 lockout\_policy  
 boot\_optimum  
 reboot\_os  
 ;;  
 4)  
 check\_openssl  
 check\_openssh  
 ;;  
 5)  
 ist\_docker  
 ;;  
 6)  
 check\_jemalloc  
 ;;  
 7)  
 exit  
 ;;  
 \*)  
 echo 'Err:Please select a number from [1-7].'  
 sleep 3  
 main  
 ;;  
 esac  
}  
  
clear  
main "$\*"