

## Appendix A

This is the supplementary content for SRG-APP-000295-DB-000305. Name script `no_login.sh` and place in the `/var/lib/pgsql` directory.

```
#!/bin/bash
# Lock: Switch roles to NOLOGIN, keep track of the roles that are switched
# Restore: Switch roles to LOGIN, only the roles found in .restore_login

lock_login=false
restore_login=false
current_dir=$(pwd)
restore_file="${current_dir}/.restore_login"

#####
# Get Options - database_name, help, lock user logins, restore logins
#####
while getopts ":d:hlr" opt
do
    case ${opt?} in
        d) database_name=${OPTARG?} ;;
        h) echo "$0 -d <database_name> -l (lock) |-r (restore)"; exit 0;;
        l) lock_login=true ;;
        r) restore_login=true ;;
        *) echo "Invalid argument: $0 -d <database_name> -l (lock) |-r
(restore)"; exit 1;;
    esac
done

#####
# Make sure an option is set
#####
if ( ! ${lock_login?} && ! ${restore_login?} )
then
    echo "Must choose lock or restore login access"
    echo "$0 -d <database_name> -l|-r"
    exit 1
fi

#####
# No options or both options detected, error out
#####
if [ $# -ne 3 ] || ( ${lock_login?} && ${restore_login?} )
then
    echo "Must choose lock or restore login access"
    echo "$0 -d <database_name> -l|-r"
    exit 1
fi
```

```

fi

#####
# Keep track of users that we disable/restore so we don't restore
# access to users that have been locked already by DBA
#####
role_array=()

if [[ ! -f ${restore_file?} ]]
then
    touch ${restore_file?}
    chmod 660 ${restore_file?}
else
    readarray -t role_array < ${restore_file?}
fi

#####
# Lock detected but restore file has users - need to restore access
# first so we don't lose track of users
#####
if (( ${lock_login?} && ${#role_array[@]?} > 0 ))
then
    echo "Lock triggered but roles are already disabled.  Restore access then
relock:"
    echo "$0 -d <database_name> -r"
    exit 1
fi

#####
# Lock users from login and disconnect their current session
#####
if ${lock_login?}
then
    roles=($(psql -d ${database_name?} -A -t -q -c "SELECT rolname, rolcanlogin
FROM pg_roles"))
    if [[ $? != 0 ]]
    then
        exit 1
    fi

    for role in ${roles[@]?}
    do
        rolname=$(echo ${role?} | awk -F'|' '{print $1}')
        rolcanlogin=$(echo ${role?} | awk -F'|' '{print $2}')
        if [[ ${rolname?} != 'postgres' ]]
        then
            if [[ ${rolcanlogin?} == 't' ]]
            then
                psql -q -c "ALTER ROLE ${rolname?} NOLOGIN"
                if [[ $? != 0 ]]

```

```

        then
            echo "Error altering role.  Exiting.."
            exit 1
        fi
        echo ${rolname?} >> ${restore_file?}

        psql -q -c "SELECT pg_terminate_backend(pid) FROM
pg_stat_activity WHERE username='${rolname?}'" >& /dev/null
        if [[ $? != 0 ]]
        then
            echo "Error terminating role cyrrebt session.  Exiting.."
            exit 1
        fi
    fi
fi
done
echo "Lock success"
#####
# Restore users login
#####
else
    if (( ${#role_array[@]?} <= 0 ))
    then
        echo "Nothing to restore"
        exit 0
    fi

    for rolname in ${role_array[@]?}
    do
        psql -q -c "ALTER ROLE ${rolname?} LOGIN"
        if [[ $? != 0 ]]
        then
            echo "Error altering role.  Exiting.."
            exit 1
        fi
    done
    > ${restore_file?}
    echo "Restore success"
fi

exit 0

```