superuser

Practical (closer to the classical Linux one) way to use (automate) SUDO for CygWin

Asked 10 years, 1 month ago Modified 6 years, 4 months ago Viewed 5k times

11

Being able to use `sudo` commands in **CygWin is useful, and faster** than opening an elevated shell:

```
Luis@Kenobi /cygdrive/c/Users/Luis
$ net user /add TestUser
System error 5.
Access denied.

Luis@Kenobi /cygdrive/c/Users/Luis
$ sudo net user /add TestUser
Command completed successfully.
```

As shown above, you can run Windows commands/scripts too, just as Linux. For me is neat; **works on remote** (SSH) consoles and allows to **combine Windows/Linux** commands. So being able to execute administrative tasks is nearly a must.

But the SUDO for CygWin project has a behavior that could be **dangerous**: it works as a **server/client** architecture, in fact, a client (sudo) send requests of commands to a server (sudoserver.py) at internal (not listening outside the local computer) port 7070TCP, with **no user or permissions checking**, so anyone (even unprivileged users) logged in onto the computer **could execute admins** (CygWin or Windows) shell commands or scripts (CygWin or Windows, too).
The problem **gets worse** if you keep the suggested method of the author: registering "sudoserver.py" as a service, so it will keep permanently running.

So, to maintain the things a bit **more secure** (not totally), I do:
1.- Execute "sudoserver.py" on an admin shell.
2.- Execute my "sudo" commands on another CygWin shell.
3.- Close (Ctrl+C) "sudoserver.py" and the admin shell.

A bit **annoying**. I am workarounding it using a `.cmd` file with assigned **hotkey** that runs "sudoserver.py", and I am closing (manually) it after my administrative jobs, but **still far from the classic "sudo"** usability on Linux.

The great and practical way would be some method that:

1. **Auto-opens "sudoserver.py" requesting for UAC Elevation Prompt (or user/password).
2. **Closes it** after a while, so UAC requestion will not keep disturbing in case of several `sudo` commands executed sequentially.

Is there any way to **automate this**, at least partially?

windows command-line bash cygwin sudo

Share Improve this question Follow

edited Apr 28, 2014 at 23:50

asked Apr 13, 2014 at 22:34
Sopalajo de Arrierez
6,723 13 66 99

3 Answers

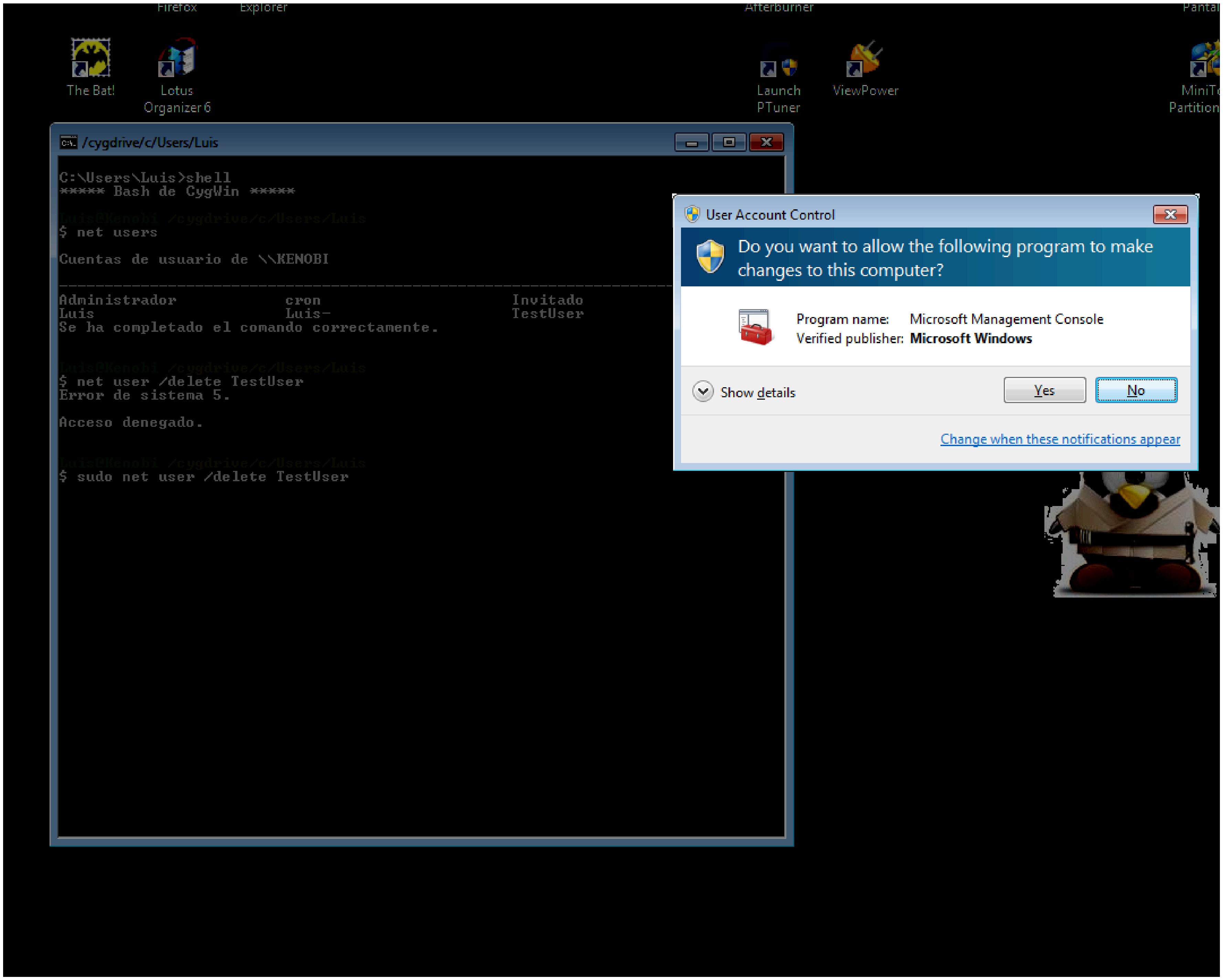
Sorted by: Highest score (default)

12

NOTE: This is mostly a **program** (shell script) I made, and I know this forum is more a question-answer site than a programs-introduction one. But I don't have any GitHub (or similar) account, nor have I had the time to research about the method of publishing an Open Source program to the community. So, as long as there is a risk that a working and useful program keeps unnoticed (even for months) to those who could enjoy it, and it would be sad to not share an already made program, I am going to **publish it here** for now. No problems for me if it the admins decide to remove this thread, I will understand. I hope to have worded the matter in a **question-answer** way enough to make it useful to this forum. If there are enough

interested users, I will do my best to dedicate some time to **continue the project** (after all my researchs, I have not found anything closest to this on Internet, but, well... I don't know if my script is valuable or it has been a waste of time).

I have programmed a **simple Linux shell script** that works (until now) on CygWin and helps (I hope) reducing the SUDO for CygWin time-attack interval. The program is named **TOUACExt** (acronym of "**TimeOut and UAC Extension**") and acts as a **wrapper for SUDO for CygWin** (required installed), and is really composed by a set of four `.sh` programs.



```
C:\Users\Luis>shell
***** Bash de CygWin *****

Luis@Kenobi /cygdrive/c/Users/Luis
$ net users

Cuentas de usuario de \\KENOBI

-------------------------------------------------------------------------------
Administrador            cron                     Invitado
Luis                     Luis-                    TestUser
Se ha completado el comando correctamente.


Luis@Kenobi /cygdrive/c/Users/Luis
$ net user /delete TestUser
Error de sistema 5.

Acceso denegado.


Luis@Kenobi /cygdrive/c/Users/Luis
$ sudo net user /delete TestUser
```

Do you want to allow the following program to make changes to this computer?

Program name: Microsoft Management Console
Verified publisher: **Microsoft Windows**

Features:

- **Comfortable usage**: By simulating the original sudo from Linux behavior, the UAC confirmation **request prompt only appears once** (multiple consecutive `sudo` commands will only generate one UAC request). As long as sudoserver.py keeps running (15 minutes default), there will be **no more UAC** requests.
- Privileged (Admin) users only **get UAC confirmation** request (*Yes/No*) on screen.
- Unprivileged (non-Admin) users **get an Administrator account/password** input screen.
- sudoserver.py keeps running, then **closes automatically** after predefined time (15 minutes) from the last sudo command execution.
- sudoserver.py does not close (keeps runing and will check again in 5 minutes) in case of **any instance of sudo** running.
- **Works remotely** (tested via SSH):
  - **Unprivileged users** can not start sudoserver.py remotely.
- Creates a (yet simple and not very readable) **log** at `/var/log/SUDOForCygWin/` .

Requirements (in CygWin):

- SUDO for CygWin.
- pgrep (at `procps` package).
- flock (at `util-linux` package).
- nohup (I think *installed by default* on CygWin, but not sure).

Assuming: - The two programs of the SUDO for CygWin project on the path suggested by the author:

```
/usr/local/bin/sudoserver.py
/usr/local/bin/sudo
```

TOUACExt have been **tested working** on Windows 7 SP1 and Windows XP SP3, but I don't know if it makes sense in using it on this last one.

Installation Instructions:

- Put this script (suggested name: `SUDOServer.cmd`) and create a shortcut (you can personalize its icon if you want) to it named `SUDOServer.lnk` (you must enable on this shortcut `Advanced Options --> Execute as Administrator`) **anywhere on your Windows path**, so `sudoserver.py` can be directly requested from Windows:

  ```
  c:\CygWin\bin\python2.7.exe /usr/local/bin/sudoserver.py
  ```

- Put the four .sh **scripts of TOUACExt** on the path, for example:

  ```
  /usr/local/bin/SUDO.sh
  /usr/local/bin/SUDOServer.sh
  /usr/local/bin/SUDOServerWatchDog.sh
  /usr/local/bin/SUDOServerWatchDogScheduler.sh
  ```

- **Rename the original** Python script from `sudo` to `sudo.py`:

  ```
  mv /usr/local/bin/sudo /usr/local/bin/sudo.py
  ```

  WARNING: The original "sudo" Python script must not remain anywhere in your path, or it could be executed instead.

- **Create this alias** (for example, manually or by editing your `~/.bashrc`):

  ```
  alias sudo='SUDO.sh'
  ```

Code for **SUDO.sh**:

```
#!/bin/bash

# ********** SUDO.sh v0.04a **********

# Variables:
# LockFile (will use a temporal one for now):
#lockfile=sudoserver-running.lck
LockFile=lockfile.lck

# Creating LogFile (if it does not exist):
mkdir /var/log/SUDOForCygWin 2>/dev/null
chmod 777 /var/log/SUDOForCygWin 2>/dev/null
LogFile=/var/log/SUDOForCygWin/$(date +%Y%m%d).log
exec 5>>$LogFile    # Redirector 5 will be the log file.
chmod 777 $LogFile >&5 2>&5 # Writable to anyone (for now).

# Start of the program
echo "========== Starting SUDO Server for CygWin ==========" >&5
echo $(date) >&5
```

```
# does the lock file exists as locked?
if [ $(flock -n $TMP/$LockFile echo>/dev/null;echo $?) -eq 0 ]
   then
    # The lock file is not locked.
    echo "LockFile not locked. Testing sudo access..." >&5
    if [ $(sudo.py vartemp=0>/dev/null 2>/dev/null;printf $?) -eq 0 ]
       then
        # Wooops. sudoserver.py is running without the lockfile. Better to
correct this.
        echo "LockFile not locked, but sudoserver.py seems to be running." >&5
        printf "Killing sudoserver.py...\n" >&5
        sudo.py kill $(sudo.py pgrep.exe -f -l sudoserver.p[y] | grep "pgrep" -v
| awk '{print $1}') >&5 2>&5
    fi
    # Starting SUDOServer.sh
    printf "Requesting SUDOServer start...\n" >&5
    nohup SUDOServer.sh >&5 2>&1&
    # Wait some time delay for UAC Prompt to start
    sleep 2
    timeout=$((SECONDS+10))
    # Has sudoserver.py already started?
    while [ $(flock -w 1 $TMP/$LockFile echo>/dev/null;printf $?) -eq 0 ] || [
$(tasklist | grep "consent.exe" -i>/dev/null;printf $?) -eq 0 ]
    do
        # No. We have to wait.
        # Waiting for SUDOServer.py to be running.
        printf "."
        if [ $SECONDS -ge $timeout ]
           then
            # sudoserver.py not responding. Aborting with errorlevel=3.
            printf "sudoserver.py not responding. Aborting.\n"
            exit 3
        fi
    done
    # Yes. sudoserver.py is up and running.
fi

printf "\n"
# Schedule (add) SUDOServer Watch Dog to Task Scheduler:
SUDOServerWatchDogScheduler.sh

# Invoke requested sudo command
sudo.py $@

#printf "ErrorLevel was: "$?
```

```
# ErrorLevel Codes:
# 3 --> timeout waiting for sudoserver.py to respond.
```

Code for **SUDOServer.sh**:

```
#!/bin/bash

# ********** SUDOServer.sh v0.04a **********

# Variables:
# LockFile (a temporal one for now):
#lockfile=sudoserver-running.lck
LockFile=lockfile.lck

# Check for other instances of sudoserver.py running
if [ $(flock -n $TMP/$LockFile echo>/dev/null;printf $?) -eq 0 ]
   then
    printf "Creating lockfile: "$TMP/$LockFile"\n"
    flock $TMP/$LockFile -c 'cmd /c SUDOServer'
    # The file has been unlocked. Send error level=2.
    exit 2
   else
    printf "The lockfile: "$TMP/$LockFile" is locked by another process.\n"
    printf "Exiting SUDOServer.sh"
fi

printf "SUDOServer.sh execution finished. Exiting."

# Exiting with no problems.
exit 0

# ErrorLevel Codes:
# 2 --> SUDOServer.lnk (maybe denial of UAC).
```

Code for **SUDOServerWatchDog.sh**:

```
#!/bin/bash

# ********** SUDOServerWatchDog.sh v0.04a **********

# Variables:
```

```
# LockFile (a temporal one for now):
#lockfile=sudoserver-running.lck
LockFile=lockfile.lck

# Redirecting to LogFile:
LogFile=/var/log/SUDOForCygWin/$(date +%Y%m%d).log
exec 5>>$LogFile
if [ $(stat $LogFile -c %a) -ne 777 ]
   then
    echo "Logfile "$LogFile" has incorrect permissions." >&5
    echo "Attemping to change permissions of "$LogFile >&5
    chmod 777 $LogFile >&5 2>&5
fi

# Remove Task Scheduler entry, if exists.
if [ $(schtasks.exe /query | grep "SUDOServerWatchDog" -i>/dev/null 2>&5;printf
$?) -eq 0 ]
   then
    sudo.py schtasks.exe /delete /tn "SUDOServerWatchDog" /f >&5 2>&5
fi

# Is sudoserver.py running?
if [ $(flock -n $TMP/$LockFile echo>/dev/null;printf $?) -eq 1 ] || [ $(sudo.py
vartemp=0>/dev/null 2>/dev/null;printf $?) -eq 0 ]
   then
    # Yes. sudoserver.py is running. So...
    printf "sudoserver.py detected running...\n" >&5
    # Is any instance of sudo running right now?
    if [ $(sudo.py pgrep -f -l "/usr/local/bin/sudo.py " | grep -v grep>/dev/null
2>&5;printf $?) -eq 0 ]
       then
        # Yes. sudo is running right now. So...
        printf "There are instances of sudo running.\n" >&5
        sudo.py schtasks /create /tn "SUDOServerWatchDog" /tr
"SUDOServerWatchDog" /sc minute /mo 5 /sd 10/10/2010 /ru "SYSTEM" >&5 2>&5
        printf "Will check again in 5 minutes. Adding Task.\n" >&5
       else
        # No. sudo is not running right now. So...
        # Kill sudoserver.py.
        printf "Closing sudoserver.py\n" >&5
        sudo.py kill $(sudo.py pgrep.exe -f -l sudoserver.p[y] | grep "pgrep" -v
| awk '{print $1}')
    fi
   else
    printf "sudoserver.py not running. Nothing to be done.\n" >&5
fi
```

Code for **SUDOServerWatchDogScheduler.sh**:

```
#!/bin/bash

# ********** SUDOWatchDogScheduler.sh v0.04a **********

# Check if WatchDog is already scheduled
if [ $(schtasks.exe /query | grep "SUDOServerWatchDog">/dev/null 2>&5;printf $?)
-eq 0 ]
   then
    # Yes. Remove it in order to create a new one.
        echo "Task SUDOServerWatchDog already existing." >&5
    echo "Removing task SUDOServerWatchDog..." >&5
    sudo.py schtasks.exe /delete /tn "SUDOServerWatchDog" /f >&5 2>&5
    if [ $? -eq 0 ]
       then
        # Task correctly deleted.
        echo "Task correctly removed." >&5
       else
        # Something failed in task creation. Report.
        echo "ERROR on deleting the SUDOServerWatchDog programmed task." >&5
    fi
fi
# Schedule new task for deletion.
echo "Adding new SUDOServerWatchDog task to trigger in 15 minutes." >&5
sudo.py schtasks /create /tn "SUDOServerWatchDog" /tr "SUDOServerWatchDog" /sc
minute /mo 15 /sd 10/10/2010 /ru "SYSTEM" >&5 2>&5
if [ $? -eq 0 ]
   then
    # Task correctly scheduled.
    echo "Task SUDOServerWatchDog correctly scheduled." >&5
   else
    # Something failed in task scheduling. Report.
    echo "ERROR on scheduling programmed task SUDOServerWatchDog." >&5
fi
```

Test the program from a CygWin Bash shell:

```
Luis@Kenobi ~
$ sudo ls -la
<UAC ELEVATION PROMPT APPEARS>
total 49
drwxr-xr-x+ 1 Luis  None     0 abr  7 02:23 .
```

```
drwxrwxrwt+ 1 Luis- None     0 abr  4 03:27 ..
-rw-------  1 Luis  None 13798 abr 14 00:31 .bash_history
-rwxr-xr-x  1 Luis  None  1494 mar  3 11:36 .bash_profile
-rwxr-xr-x  1 Luis  None  6260 abr  6 05:19 .bashrc
-rwxr-xr-x  1 Luis  None  1919 mar  3 11:36 .inputrc
-rw-------  1 Luis  None    35 abr  2 01:43 .lesshst
-rwxr-xr-x  1 Luis  None  1236 mar  3 11:36 .profile
drwx------+ 1 Luis  None     0 mar  8 01:49 .ssh
-rw-r--r--  1 Luis  None     7 mar  4 18:01 d:ppp.txt
-rw-r--r--  1 Luis  None    37 abr  7 02:23 my.log
```

NOTE2: These scripts are in **pre-beta** release, so they are still buggy and the code is not very clean. Anyway, in my tests with three different Windows 7 computers they seem to be working (mostly) OK.

Brief **explanation** of the program:

1. Due to the alias, when performing a sudo command **the SUDO.sh script** is invoked.
2. SUDO.sh **calls SUDOServer.sh**, opening (via `SUDOServer.lnk`) "sudoserver.py" if needed.
3. The **original sudo command** invoked by the user is executed.
4. Then SUDO.sh **calls SUDOServerWatchDogScheduler.sh**, that schedules SUDOServerWatchDog.sh for execution after the given time (15 minutes default) to close `sudoserver.py`.
5. After the predefined time, SUDOServerWatchDog.sh **closes sudoserver.py**. If there are any **instance of sudo running**, it programs itself for new execution after 5 minutes.

To Do:

- **Self installer** that creates all the .sh, .cmd and .lnk files automatically.
- Establish **lock file** to some other (it is at $TMP/lockfile.lck).
- Add a **configuration script** or .config file (for defaults in timeouts, file locations... etc).
- Add System account behavior (thanks, @Wyatt8740).
- ¿Change "flock" (internal locking SUDO mode) with "fuser" where appropriate?
- **Suggestions** accepted.

Reported **Bugs**:

- The bash shell keeps open even after inputing `exit` if `sudoserver.py` is running until it closes. Provisional workarounds are welcome.

I hope **someone will use** the long hours programming I have dedicated to TOUACExt.
Enhancements and corrections accepted.
Suggestions about **where should I go publishing** the code to stop nagging this forum accepted too ;-) .

Sorry for the long post. I don't have much free time, and this project was about disappear on my closet (maybe for years, who knows?).

Share Improve this answer Follow

edited Jan 5, 2016 at 0:21

answered Apr 13, 2014 at 22:34
Sopalajo de Arrierez
6,723 13 66 99

2 If you want feedback on your code, post it on codereview.stackexchange.com. (The usage notes and examples are good to have here) – Ben Voigt Apr 14, 2014 at 2:22

Thanks, @BenVoigt, I didn't know. Please a question: if I do, I think most of the post should be a duplicate of this answer. Will that be considered cross-posting? – Sopalajo de Arrierez Apr 14, 2014 at 9:36

1 Make sure to link them to each other. The harm in crossposting is that people duplicate effort. If they're linked, that's not such a problem – Ben Voigt Apr 14, 2014 at 15:17

This is a very good solution. If it did not require python, I would use it. I have a personal, deep loathing for python. It is a nice language, but for personal reasons more than anything, I dislike it. Still, since almost no one else hates python, and my hatred is irrational, I upvoted your solution, since it is closer to the real thing than mine. – Wyatt Ward Apr 30, 2014 at 2:26

1 Thanks, @CharlesRobertoCanato . Maybe you could give me the details at the chat, in order to solve it? Chat Room "TOUACExt - SuDo for Windows" : chat.stackexchange.com/rooms/56716/touacext-sudo-for-windows – Sopalajo de Arrierez Apr 7, 2017 at 12:12

1

SIMPLE sudo.bat (uses nircmd)

Nircmd can be downloaded here:
http://www.nirsoft.net/utils/nircmd.html

I downloaded nircmd and renamed `nircmdc.exe` to `nircmd.exe`, replacing the original `nircmd.exe`. I then moved it to `C:\windows\system32`.

I also made the following batch file to allow arguments to be passed to the script.

It should be said that I have disabled UAC on my machine, so I no longer need this script, but it DOES work as of windows 8. It works in cygwin just as well.

```
@echo off
if "%*" == "" goto error
nircmd elevate %*
goto thisiseof
:error
echo No arguments were given. Exiting.
:thisiseof
```

Share Improve this answer Follow

answered Apr 17, 2014 at 14:19

Wyatt Ward
1,189 10 14

A fancy solution. Short and easy. But why is it failing me in a simple `sudo.bat dir` ? An error windows tells "Windows can not find the file named dir". It seems to work with `sudo echo Hello` , but there is no console output. – Sopalajo de Arrierez Apr 17, 2014 at 14:38

A **small inconvenience** with this method is the continuous request for UAC prompt in consecutive commands. **TOUACExt solves** this, much like in classical Linux sudo executions. I have edited the features list to show it. – Sopalajo de Arrierez Apr 17, 2014 at 20:04

`dir` doesnt work because `dir` is not technically a program, but a built-in DOS command. while in linux, `ls` is a binary program, in DOS/windows, `dir` is handled by the interpreter itself (i.e. COMMAND.COM or cmd.exe). There is not a `dir.exe` anywhere for my program to run. But for cygwin, `sudo ls` should suffice. Even if you don't, doing `sudo cmd` or `sudo bash` or whatever should get you an 'Administrator' level prompt. Even 'Administrator' is below 'SYSTEM', though - for 'SYSTEM', use `nircmd.exe elevatecmd runassystem <program.exe>` . Also, I disable UAC on my machines :) – Wyatt Ward Apr 30, 2014 at 2:22

Also, I don't think `echo` is a program either in windows. It's part of COMMAND.COM/cmd.exe. For me though, a cygwin user with `ls.exe` and `echo.exe` it works fine. And I actually used it today for the first time in months to manage files on my brother's account without logging in as him (he managed to put every single program on his computer in his start menu's "startup" directory :P). Just logged into another and used `sudo.bat cmd` to get an administrator level prompt that would let me manage other user's files. – Wyatt Ward Apr 30, 2014 at 2:31

Your idea of the System account is good. I will add it to TOUCExt as an option. – Sopalajo de Arrierez Apr 30, 2014 at 9:36

1

Being unhappy with the available solution, I adopted nu774's script to add security and make it easier to setup and use. The project is available on Github

To use it, just download `cygwin-sudo.py` and run it via `python3 cygwin-sudo.py **yourcommand**` .

You can set up an alias for convenience:

```
alias sudo="python3 /path-to-cygwin-sudo/cygwin-sudo.py"
```

Share Improve this answer Follow

answered Jan 23, 2018 at 16:38

Chronial
231 2 8