

SSHswitch (u.blanxd.sshswitch) | version 1.1.0

Blanxd.H <blanxd.h@gmail.com> (reddit.com/u/blanxd)

July 2023

- NAME
- SYNOPSIS
- DESCRIPTION
- SUBCOMMANDS (Info requests)
- SUBCOMMANDS (OpenSSH settings)
- SUBCOMMANDS (Tool/Tweak settings)
- SUBCOMMANDS (Settings for Launchd)
- EXIT STATUS
- EXAMPLES
- COPYRIGHT (MIT/Expat)

NAME

SSHswitch - an iOS (jailbroken) setuid tool for configuring the OpenSSH server, and starting/stopping it.

SYNOPSIS

SSHswitch

```
SSHswitch on [port(s) \
    a[0|no|1|yes|-] \
    w[0|no|1|yes|-] \
    u[0|no|1|yes|-] \
    g[0|no|1|yes|-] \
    k[0|no|1|yes|-] \
    l[0|no|1|yes|-]]
```

```
SSHswitch off [port(s) \
    a[0|no|1|yes|-] \
    w[0|no|1|yes|-] \
    u[0|no|1|yes|-] \
    g[0|no|1|yes|-] \
    k[0|no|1|yes|-] \
    l[0|no|1|yes|-]]
```

```
SSHswitch s | status
```

```
SSHswitch p | port [awutglkefrnbv]
```

```
SSHswitch a | allowRoot
```

```
SSHswitch w | withPassword
```

```
SSHswitch u | usingKey
```

```
SSHswitch t | twoFactor
```

```
SSHswitch g | globalListener
```

```
SSHswitch i | info | ij
```

```
SSHswitch l | letEmLinger
```

```
SSHswitch k | kickEm
```

```
SSHswitch e | evenIfLocked
```

```
SSHswitch f | forbidIfLocked
```

```
SSHswitch r | runOnBoot
```

```
SSHswitch n | notOnBoot
```

```
SSHswitch b | bootAsToggled
```

```
SSHswitch d | defaultState
```

```
SSHswitch v | version | -v
```

```
SSHswitch -h
```

```
SSHswitch h | help | --help
```

DESCRIPTION

This tool is used internally by the GUI tweaks

- OpenSSH Settings (u.blanxd.OpenSshPort)
- OpenSSH CC Toggle (u.blanxd.OpenSshCC)
- OpenSSH Flipswitch (u.blanxd.OpenSshFlipswitch)

It sets OpenSSH options in `/etc/ssh/sshd_config` and `/Library/LaunchDaemons/com.openssh.sshd.plist`, it uses `launchctl` to start/stop the server and read its running status. It keeps the requested settings (its preferences) in `/var/mobile/Library/Preferences/u.blanxd.sshswitch` directory, and the package also tries to keep a backup of the original OpenSSH config files in there (before version 1.1.0 the settings were being kept in `/etc/ssh` as separate files, and there was a backup of the `launchd` plist right next to the original). Some of these settings are also directly read by the GUI toggle tools. In addition to the OpenSSH server settings these preferences toggle

- whether it kicks ssh sessions off the server (the *iDevice*) when it turns the server off.
- whether using the toggles is allowed while the device is locked.
- how it uses `launchctl` for starting/stopping the server (ie. whether ON or OFF are only temporary/runtime or permanent).

`com.openssh.sshd.plist` is being read and written to as a plain text XML file, so if anyone (any OpenSSH packager) ever makes it a binary or JSON plist then this tool might not be able to report the correct port numbers, nor the local/global listening status (but it wouldn't write to the file then either).

The SSH server itself runs as *root*, so it wouldn't be possible for the *mobile* user (the iOS GUI and apps) to start/stop this, that's why this tool uses `setuid 0`, and is installed as such, so it can be used by any user on the system.

When it successfully turns the server on or off, it emits a system-wide Darwin notification, which may be observed by any process. The name of the notification is “com.openssh.sshd/on” or “com.openssh.sshd/off”. If it's an internal `sshd` restart after changing some settings then there will be two of those, there are no exceptions.

If the OpenSSH server package is upgraded or reinstalled (while this tweak is present), it tries to re-backup the new config files and apply the custom settings again. For this it relies on the packaging system (`dpkg`) to tell when these files are being changed, to the maintenance script of this tweak, so it can rerun the tool using the saved settings.

All subcommands, although representing English words/expressions, are really only read from the beginning until a unique instruction can be determined, which mostly means the 1st letter. Two characters are needed for *on*, *off*, *ij*, *-v*, *-h*, *--help*.

All paths mentioned in this document might be preceded by “`/var/jb`” or some dynamic JB root path if it's being run on a rootless jailbreak (the tweak preferences are always kept in `/var/mobile/Library`, but the backup files' location depends on the rootlessness of the JB).

SUBCOMMANDS (Info requests)

SSHswitch

calling it without any args/subcommands simply exits with the return code 0 if OpenSSH is running, 1 if it isn't.

status | s

prints whether SSH is **on** or **off**

port | p [awutglkefrnbv]

prints the port(s) OpenSSH is currently configured to listen on, each one on a separate line.

It can be followed by any or all of the 5 letters described next and/or any or all of the tweak settings' or `launchd` settings' letters (in any order), in which case it also prints 'yes' or 'no' preceded by the corresponding letter, each on a separate line, and in this case the exit code reflects the running status of the server. They may be separated by spaces as separate args but may as well be written as one argument. This is the API output that the OpenSSH Settings tweak uses. Neither the ports nor the yes's/no's are guaranteed to be in any particular order. The corresponding tweak toggle preferences cancel each other, so it might not be necessary to query eg. both *l* and *k*.

allowRoot | a

prints whether root is allowed to log in (**yes** or **no**)

withPassword | **w**

prints whether it's allowed to login with a password (**yes** or **no**)

usingKey | **u**

prints whether it's allowed to login with a key (**yes** or **no**)

twoFactor | **t**

prints whether both the password and a key are required (**yes** or **no**)

globalListener | **g**

prints whether it's listening for connections from outside the localhost (**yes** or **no**)

info | **i** | **ij**

list active sessions, returns 1 if there aren't any. Adding 'j' makes it output the info in JSON format, without a newline.

defaultState | **d**

prints the startup state: `bootAsToggled` | `runOnBoot` | `notOnBoot`.

version | **v** | **-v**

prints the version of this tool.

-h

prints a short list of all the possible subcommands.

help | **h** | **--help**

prints a more comprehensive description (still less than in this manpage).

SUBCOMMANDS (OpenSSH settings)

All configuration concerning OpenSSH itself happens with the *on* or *off* subcommands. The optional arguments that follow the *on/off* determine if anything gets written to `/etc/ssh/sshd_config` and/or `/Library/LaunchDaemons/com.openssh.sshd.plist`, and whether it restarts/restops the server service. Their order doesn't matter and they may be repeated, the last one wins. Each of them, except for the ports, should start with a letter, possibly followed by *1*, *0*, *yes*, *no*, or *-* (a minus, which skips setting that option). The suffix *1/0/yes/no/-* determines which way it gets toggled, any *"-"* means *skip* and any *"0..."* or *"n..."* means *no*, anything else is a *yes* (including omitting the suffix). Before v.1.1.0 these didn't have the letter prefixes and they were positional (in the order they're described below), this still works for backward compatibility.

on | **off**

determines if the server stays on/off after the command has finished, regardless if any additional options follow.

Optional additional arguments after *on/off*:

[p]port[,port[...]] | **p-**

numeric port number(s) that the server needs to listen on, it may be prefixed with "p", but this isn't required. This tool only saves port 22 (the standard) and/or most numbers from 1001 through 65535 (a couple of high ports are known to be used by iOS itself which we avoid), there is no error indication if others are requested, but they don't get saved. This is just an arbitrary precaution I took so noone would make it listen on some port used by other known services. Up to 8 ports may be specified, separated by commas (no spaces). The *port* subcommand reads the result.

a | **a1** | **a0** | **ayes** | **ano** | **a-**

determines if root is allowed to log in (PermitRootLogin keyword in `sshd_config`). The *allowRoot* subcommand reads the result.

w | **w1** | **w0** | **wyes** | **wno** | **w-**

determines if it's allowed to log in using a password (PasswordAuthentication and/or KbdInteractiveAuthentication keywords in `sshd_config`). The *withPassword* subcommand reads the result.

u | u1 | u0 | uyes | uno | u-

determines if it's allowed to log in using a public key (PubkeyAuthentication keyword in sshd_config). The *usingKey* subcommand reads the result.

g | g1 | g0 | gyes | gno | g-

determines if connections are being listened for on all addresses (*1/yes/omit*) or localhost only (ListenAddress keywords in sshd_config and SockNodeName key in the launchd plist). The *globalListener* subcommand reads the result. This didn't exist before v.1.1.0.

k | k1 | k0 | kyes | kno | k-

l | l1 | l0 | lyes | lno | l-

this is not an OpenSSH setting. It makes us kick (or not) all possible (incoming) ssh/sftp/scp sessions off the server, even if the subcommand is *on*. They are opposites, *k* means Kick'Em nad *l* means Let'EmLinger. This option also overrides the *letEmLinger/kickEm* tweak preference setting (see below). In v.1.0.x this was the 5th possible toggle after *on/off*, only accepting 1/0 (and the 4th in some earlier versions of SSHonCC).

If both the *w* (withPassword) and *u* (usingKey) options are given 0|no then this doesn't make much sense, either Password or PubKey authentication should be allowed, else it's a lockout (unless using some more advanced authentication schemes, which are possible, but I'm not sure if anyone uses those in iOS). So this tool instead reverts these to 'yes', and additionally makes the server require both, *twoFactor* (using the AuthenticationMethods keyword in sshd_config). Ie. a user must have their pubkey in the authorized_keys file, and in addition they must provide the password when logging in. Ie. Extra Secure :)

Changing these settings may make the tool restart the server if it's issued while it's running. Definitely if the port(s) or the global/local listener get altered, but also for other settings if sshd isn't using launchd to listen and runs as a daemon on its own (a few jailbreaks have it like that).

If the OpenSSH server package is reinstalled or upgraded via a package manager (ie. dpkg), the config files /etc/ssh/sshd_config and /Library/LaunchDaemons/com.openssh.sshd.plist historically get overwritten, so the postinst script of this package tries to reapply the settings which this tool has changed.

SUBCOMMANDS (Tool/Tweak settings)

kickEm | k

makes us kick all possible (incoming) ssh/sftp/scp sessions off the server whenever an *off* subcommand is issued. This is the default when this tweak is installed. Use *letEmLinger* to revert this. It can be overridden with the *lyes* or *kno* option to the *off* subcommand.

letEmLinger | l

the opposite of the above *kickEm* subcommand, ie. the *off* subcommand would simply turn off the server and never mind if anyone is logged in. It can be overridden with the *kyes* or *lno* option to the *off* subcommand.

forbidIfLocked | f

this is used by the toggle tweaks. Makes them have no effect when the device is locked. This is the default for the u.blanxd.OpenSsh... series of toggles. Use *evenIfLocked* to revert this.

evenIfLocked | e

this is used by the toggle tweaks. The opposite of the above *forbidIfLocked*, makes them work even when the device is locked.

SUBCOMMANDS (Settings for Launchd)

runOnBoot | r

This is the default when OpenSSH gets installed ie. whenever the device gets re-jailbroken (or just rebooted in case the jb is untethered) then the ssh server starts up as well. Internally, the tool starts the server with 'launchctl load -w' and stops it with 'launchctl unload -F'.

notOnBoot | n

This makes the ssh server not start up when the device is re-jailbroken or rebooted. Internally, the tool starts the server with ‘launchctl load -F’ and stops it with ‘launchctl unload -w’.

bootAsToggled | b

This makes the ssh server either start or not, based on whichever state it was in before the restart/re-jailbreaking. Internally, the tool always starts and stops the server with ‘launchctl (un)load -w’.

Changing this setting may make the tool restart the server, depending on which way it got turned to while the server is running. Similarly it might get turned on for a millisecond if changing this while the server is off.

This setup might get overridden when the OpenSSH server gets re-installed or upgraded, because often the postinst script in the OpenSSH server package simply uses ‘launchctl load -w’ to load it up after any installation (but not all OpenSSH packages do that in case of upgrades). The postinst script of this package tries to automatically reapply the setting (if it has been altered).

EXIT STATUS

Any exit status ≥ 10 is an error in any case. One should really always check this, it only prints info to STDERR in case of unrecognized arguments (15) and launchctl errors (11-14,17, these also go to Apple System Log for reading with other tools).

Regular subcommands should return 0.

Without any subcommands, or in case there are extra letters after *port*, it returns 0 if the server is running, and 1 if it isn’t.

info and *ij* return 1 if there aren’t any sessions.

This program itself defines the following error codes:

10: setuid error
11-14: posix_spawn and its results errors while executing launchctl
15: unknown command line argument(s) given
16: failed reading /Library/LaunchDaemons/com.openssh.sshd.plist
17: cannot find launchctl binary
18: failed reading /etc/ssh/sshd_config
19-21: failed writing /etc/ssh/sshd_config
22-29: failed reading tweak prefs
30-40: failed writing tweak prefs
41-44: failed writing /Library/LaunchDaemons/com.openssh.sshd.plist

EXAMPLES

Most info request commands should be easy, they're just saying *yes* or *no*, and *on* or *off*. Showing some API type subcommands here.

```
iPhone:~ mobile$ # show the port(s)
iPhone:~ mobile$ SSHswitch port
10222
22
iPhone:~ mobile$ # turn off, save port 22 and allow password auth
iPhone:~ mobile$ SSHswitch off 22 wyes
iPhone:~ mobile$ # report ports and if password auth is allowed
iPhone:~ mobile$ SSHswitch p w
22
yes
iPhone:~ mobile$ # report all the OpenSSH and tweak settings this tool supports
iPhone:~ mobile$ SSHswitch port awutg ef kl rnb v
eno
fyes
```

```

kno
lyes
ryes
nno
bno
v1.1.0
22
gyes
ano
wyes
uyes
tno
iPhone:~ mobile$ # exit status shows if it's running
iPhone:~ mobile$ SSHswitch
iPhone:~ mobile$ echo $? # 1 means it's off, 0=running
1
iPhone:~ mobile$ # turning twoFactor on among other things
iPhone:~ mobile$ SSHswitch on 10222 a1 w0 u0 g1
iPhone:~ mobile$ SSHswitch p awutg
10222
gyes
ayes
wyes
uyes
tyes
iPhone:~ mobile$ # the p(ort) command with extra args returned 0 if it's running
iPhone:~ mobile$ echo $?
0
iPhone:~ mobile$ # kick everyone out without changing anything
iPhone:~ mobile$ SSHswitch on k
Connection to 10.8.9.14 closed.

```

COPYRIGHT (MIT/Expat)

Copyright (c) 2023-07 Blanxd.H blanxd.h@gmail.com

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.