

Apple's Sandbox Guide

v0.2

Table of Contents

| | |
|---|-----------|
| 1 – Introduction | 3 |
| 2 – What are we talking about?..... | 3 |
| 3 – How can it be used or implemented? | 4 |
| 4 - Anatomy of a custom profile | 5 |
| 5 - Commands Reference | 6 |
| 5.1 – Actions | 6 |
| 5.2 – Operations | 6 |
| 5.3 – Filters | 27 |
| 5.4 – Modifiers | 31 |
| 5.5 - Other keywords..... | 32 |
| 6 – Special hardcoded cases | 32 |
| 7 – A sample profile for MS Outlook 2011 | 32 |

1 – Introduction

Apple's sandbox technology was introduced in Leopard version of Mac OS X, called Seatbelt and based on TrustedBSD MAC framework.

A few years have passed and documentation is still scarce. Dionysus Blazakis published a great paper¹ and presentation at Blackhat DC 2011, reversing the userland and kernel implementations of this feature. The other available public references are Apple's own sandbox profiles (located at `/usr/share/sandbox`) and some attempts by other users to create new profiles².

This document tries to close this gap by documenting the available operations and options available in this technology. I have tried to provide examples for all operations so it is easier to understand their impact.

This document is a work in progress and based on Snow Leopard v10.6.8. Apple still considers this technology as private and subject to changes.

It is very much possible that mistakes and wrong assumptions exist so all contributions and fixes are more than welcome! You can contact me at reverser@put.as or tweet @osxreverser.

The latest version is always available at <http://reverse.put.as>.

Enjoy,

fG!

2 – What are we talking about?

Using the definition from Apple's website:

“Sandboxing protects the system by limiting the kinds of operations an application can perform, such as opening documents or accessing the network. Sandboxing makes it more difficult for a security threat to take advantage of an issue in a specific application to affect the greater system.”

The implementation found in Mac OS X can limit the following type of operations:

- File: read, write, with many different granular operations
- IPC: Posix and SysV
- Mach
- Network: inbound, outbound
- Process: execution, fork
- Signals
- Sysctl
- System

It has a rich set of operations that can help to improve the security of applications and mitigate potential attacks, especially on network-enabled applications such as web browsers, Flash or applications that process potentially untrusted input such as pdf, word/excel/powerpoint

¹ https://media.blackhat.com/bh-dc-11/Blazakis/BlackHat_DC_2011_Blazakis_Apple_Sandbox-wp.pdf

² <https://github.com/s7ephen/OSX-Sandbox-Seatbelt-Profiles>

© 2011, fG!

documents, etc. Malware analysis and reverse engineering processes can also benefit from this technology.

3 – How can it be used or implemented?

There are two alternatives to use this feature (one is just a frontend for the other).

The first is to execute an application within a sandbox, using the command “sandbox-exec”. This is the best alternative for applying a sandbox to software you don’t have source code.

The other is to implement the sandbox feature inside your code or someone else’s code. The function “sandbox_init” will place the process into a sandbox using one of the pre-defined profiles below (they are also available to sandbox-exec, although with a different name).

These profiles are:

- kSBXProfileNoInternet : TCP/IP networking is prohibited.
- kSBXProfileNoNetwork : All sockets-based networking is prohibited.
- kSBXProfileNoWrite : File system writes are prohibited.
- kSBXProfileNoWriteExceptTemporary : File system writes are restricted to the temporary folder /var/tmp and the folder specified by theconfstr(3) configuration variable _CS_DARWIN_USER_TEMP_DIR.
- kSBXProfilePureComputation : All operating system services are prohibited.

Check the sandbox_init manpage for more information.

OS X Lion introduces Application Sandboxing³, a different way of applying sandboxing but with the same underlying technology.

Now let’s focus on sandbox-exec - the best alternative for most users.

The sandbox-exec supports the pre-defined profiles but also custom profiles. Custom profiles are written in SBPL – Sandbox Profile Language (a “Scheme embedded domain specific language” using Dion’s definition). Examples can be found at “/usr/share/sandbox”. These are used to sandbox some system daemons. The next chapters describe the different operations, filters and modifiers available to write custom profiles.

The syntax for sandbox-exec command is:

**sandbox-exec [-f profile-file] [-n profile-name] [-p profile-string] [-D key=value ...]
command [arguments ...]**

The -f switch should be used for loading custom profiles. Either you can use the absolute path to the profile or just the name of the profile, as long it is located at one of the these folders:

- /Library/Sandbox/Profiles
- /System/Library/Sandbox/Profiles
- /usr/share/sandbox

Using full path example:

```
$ sandbox-exec -f /usr/share/sandbox/bsd.sb /bin/lis
```

Using profile name example:

³
http://developer.apple.com/library/mac/#documentation/Security/Conceptual/CodeSigningGuide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40005929-CH1-SW1
© 2011, fgl

```
$ sandbox-exec -f bsd /bin/ls
where bsd.sb is located at /usr/share/sandbox
```

You can also use custom profiles writing from the input, using the `-p` switch. The example from Dion's paper:

```
$ sandbox-exec -p '
> (version 1)
> (allow default)
> (deny file-read-data
>     (regex #"^/private/tmp/dump\.c$"))
> '/bin/sh
```

The `-n` switch is used to load one of the pre-defined profiles. As previously pointed out, the profiles names are different from `sandbox_init`. Use the following table as reference.

| Sandbox_init | Sandbox-exec |
|-----------------------------------|---------------------------|
| kSBXProfileNoInternet | no-internet |
| kSBXProfileNoNetwork | no-network |
| kSBXProfileNoWriteExceptTemporary | no-write-except-temporary |
| kSBXProfileNoWrite | no-write |
| kSBXProfilePureComputation | pure-computation |

Example:

```
$ sandbox-exec -n no-internet ping www.google.com
PING www.l.google.com (209.85.148.106): 56 data bytes
ping: sendto: Operation not permitted
```

4 - Anatomy of a custom profile

A profile is composed of actions on operations, modifiers, filters, options and (optionally) comments. To simplify things I will call everything commands except for comments.

The core of custom profiles are operations, which are what you want to control and limit access to. Examples of operations are read files, write files, access a network port, send signals, etc. Most operations can have filters to improve granularity, others are binary (allow or deny, globally).

The first thing to be configured is the version of the SBPL. For now there's only version 1 so this should be common to all scripts.

Additionally you can configure the logging option, with the "debug" command. Two options are available, "all", which should log all operations (allowed or not), and "deny", which logs only denied operations. The option "all" doesn't seem to work (not implemented? requires a different log level?) but the "deny" option is very useful at the custom profile writing and debugging stage.

Other profiles can be included using the “import” command, for example a profile with common rules to be shared among daemons, which is what `bsd.sb` is for BSD daemons.

The default action can be configured either to deny or to allow. This will depend on the type of profile you are interested to achieve.

Comments should start with semicolon (;) and are valid until the end of the line.

5 - Commands Reference

All commands are enclosed into parenthesis. In each example, the “\$” symbol means command execution at the shell.

5.1 – Actions

There are two available actions, allow or deny.

Actions apply only to the operations defined below.

Syntax:

(action operation [filter modifiers])

Example:

- (deny default)

All operations will be denied unless explicitly allowed (default is an operation). This is a whitelist mode.

- (allow default)

All operations will be allowed unless explicitly denied. In this case we have a blacklist mode.

5.2 – Operations

As previously described, the sandbox supports different type of operations. Almost all operations have global and granular modes. Global means that the whole category of operation can be configured. For example, the “file*” operation will control all type of file related operations. But we can also be more granular and allow file reads and deny file writes (and even be a little more specific in these two operations).

The following table shows the global operations, including the ones without granular modes.

| | | | | | |
|---------|---------|---------|--------------|----------------------|----------|
| Default | File* | Ipc* | Mach* | Network* | Process* |
| Signal | Sysctl* | System* | Job-creation | Mach-per-user-lookup | |

Operations can have filters and modifiers. Modifiers apply to all operations (except the mach ones) while filters don't.

All the available operations are now described.

Default

Syntax: (action default [modifier])

Actions: allow deny

| | |
|------------|--------------------|
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Description:

As the name implies, this is the default action if no other operation matches. It doesn't matter where this operation is configured, either at the beginning or the end of the profile. The engine will only hit the default operation if no explicit match can be found. Searching for operations will stop when the first explicit match is hit. This means that a deny action followed by an allow action to the same operation and target will never trigger the allow action, it will always be denied.

Examples:

- (allow default)

If you wish to create a blacklist type of profile.

- (deny default)

To create a whitelist profile.

- (deny default (with no-log))

To create a whitelist profile without logging.

File*

| | |
|------------|------------------------------------|
| Syntax: | (action file* [filter] [modifier]) |
| Actions: | allow deny |
| Filters: | path file-mode |
| Modifiers: | send-signal no-log |

Description:

This operation will control file related operations such as reads, writes, extended attributes, etc.

Example(s):

- (deny file*)

This will deny all file related operations to any file.

- (deny file* (literal "/mach_kernel"))

This will deny all file related operations that have /mach_kernel as target.

File-chroot

| | |
|------------|--|
| Syntax: | (action file-chroot [filter] [modifier]) |
| Actions: | allow deny |
| Filters: | path file-mode |
| Modifiers: | send-signal no-log |

Description:

Control whether the target should be allowed or not to chroot() into the specified directory.

Example(s):

- (deny file-chroot (literal "/"))

```
# sandbox-exec -f ls2 /usr/sbin/chroot -g nobody / /bin/ls
```

```
chroot: /: Operation not permitted
```

Log output:

```
Sep 2 18:45:02 macbox sandboxd[40841]: chroot(40840) deny file-chroot /
```

File-iocctl

Syntax: (action file-iocctl [filter] [modifier])

Actions: allow deny

Filters: path file-mode

Modifiers: send-signal no-log

Description:

Determine whether the target can perform the iocctl operation.

Warning: Since iocctl data is opaque from the standpoint of the MAC framework, and since iocctls can affect many aspects of system operation, policies must exercise extreme care when implementing access control checks.

Example(s):

- (allow file-iocctl (literal "/dev/dtracehelper"))

File-read*

Syntax: (action file-read* [filter] [modifier])

Actions: allow deny

Filters: path file-mode

Modifiers: send-signal no-log

Description:

Controls all available read operations described below.

Example(s):

- (deny file-read* (literal "/mach_kernel"))

```
$ sandbox-exec -f ls2 cat /mach_kernel
```

```
cat: /mach_kernel: Operation not permitted
```

Log output:

```
Sep 2 00:13:12 macbox sandboxd[24486]: cat(24485) deny file-read-data /mach_kernel
```

```
$ sandbox-exec -f ls2 ls /mach_kernel
```

```
ls: /mach_kernel: Operation not permitted
```

Log output:

```
Sep 2 00:13:46 macbox sandboxd[24498]: ls(24504) deny file-read-metadata /mach_kernel
```

```
$ sandbox-exec -f ls2 xattr /mach_kernel
```


xattr: No such file: /mach_kernel

Log output:

Sep 2 00:13:38 macbox sandboxd[24498]: Python(24497) deny file-read-xattr /mach_kernel

Sep 2 00:13:38 macbox sandboxd[24498]: Python(24497) deny file-read-metadata
/mach_kernel

File-read-data

Syntax: (action file-read-data [filter] [modifier])

Actions: allow deny

Filters: path file-mode

Modifiers: send-signal no-log

Description:

Give or refuse read access to the contents of the target file.

Example(s):

- (deny file-read-data (literal "/mach_kernel"))

```
$ sandbox-exec -f ls2 ls /mach_kernel
```

```
/mach_kernel
```

```
$ sandbox-exec -f ls2 cat /mach_kernel
```

```
cat: /mach_kernel: Operation not permitted
```

Log output:

Sep 2 00:18:59 macbox sandboxd[24653]: cat(24652) deny file-read-data /mach_kernel

File-read-metadata

Syntax: (action file-read-metadata [filter] [modifier])

Actions: allow deny

Filters: path file-mode

Modifiers: send-signal no-log

Description:

Control read access to the files-system metadata. For example "ls" will not work against the target (if action is deny) while a "cat" will (because it is accessing the contents, not the metadata).

Example(s):

- (deny file-read-metadata (literal "/mach_kernel"))

```
$ cat /mach_kernel
```

```
????uZ$
```

```
$ sandbox-exec -f ls2 cat /mach_kernel
```

```
cat: /mach_kernel: Operation not permitted
```

Log output:

Sep 2 00:24:11 macbox sandboxd[24809]: ls(24808) deny file-read-metadata /mach_kernel

File-read-xattr

| | |
|------------|--|
| Syntax: | (action file-read-xattr [filter] [modifier]) |
| Actions: | allow deny |
| Filters: | path file-mode xattr |
| Modifiers: | send-signal no-log |

Description:

This operation will control read access to the file extended attributes.

Example(s):

- (deny file-read-xattr (literal “/mach_kernel”))

Result without sandbox:

```
$ xattr /mach_kernel  
com.apple.FinderInfo
```

Result with sandbox:

```
$ sandbox-exec -f ls2 xattr /mach_kernel  
xattr: [Errno 1] Operation not permitted: '/mach_kernel'
```

File-revoke

| | |
|------------|--|
| Syntax: | (action file-revoke [filter] [modifier]) |
| Actions: | allow deny |
| Filters: | path file-mode |
| Modifiers: | send-signal no-log |

Description:

Controls access to revoke().

Example(s):

N/A

File-write*

| | |
|------------|--|
| Syntax: | (action file-write* [filter] [modifier]) |
| Actions: | allow deny |
| Filters: | path file-mode |
| Modifiers: | send-signal no-log |

Description:

Controls all available write operations described below.

Example(s):

- (deny file-write* (literal “/test”))

```
$ sandbox-exec -f ls2 touch /test
```

touch: /test: Operation not permitted

Log output:

Sep 2 21:05:46 macbox sandboxd[45341]: touch(45340) deny file-write* /test

File-write-data

| | |
|------------|--|
| Syntax: | (action file-write-data [filter] [modifier]) |
| Actions: | allow deny |
| Filters: | path file-mode |
| Modifiers: | send-signal no-log |

Description:

Give or refuse write access to the contents of the target file.

Warning: this doesn't seem to work as expected if action is deny! File-read-data works as expected – content can't be read – but for some reason this one doesn't deny write contents to the target file (only file-write* works).

For example this works (data is written):

```
(allow file-write-data
  (literal "/private/tmp/test3")
)
(deny file-write* (literal "/private/tmp/test3"))
```

While this doesn't work (data is written when it shouldn't):

```
(deny file-write-data
  (literal "/private/tmp/test3")
)
(allow file-write* (literal "/private/tmp/test3"))
```

Or this also doesn't work (data is written when it shouldn't):

```
(allow default)
(deny file-write-data (literal "/private/tmp/test3"))
```

Example(s):

```
(deny file-write-data (literal "/private/tmp/test3"))
```

File-write-flags

| | |
|------------|---|
| Syntax: | (action file-write-flags [filter] [modifier]) |
| Actions: | allow deny |
| Filters: | path file-mode |
| Modifiers: | send-signal no-log |

Description:

Control access to file flags (check manpage for chflags).

Example(s):

- (deny file-write-flags (literal “/private/tmp/test”))

```
$ sandbox-exec -f ls2 chflags nohidden /tmp/test
```

```
chflags: /tmp/test: Operation not permitted
```

Log output:

```
Sep  2 19:29:59 macbox sandboxd[42198]: chflags(42197) deny file-write-flags  
/private/tmp/test
```

File-write-mode

Syntax: (action file-write-mode [filter] [modifier])

Actions: allow deny

Filters: path file-mode

Modifiers: send-signal no-log

Description:

Control access to file modes.

Example(s):

- (deny file-write-mode (literal “/private/tmp/test”))

```
$ sandbox-exec -f ls2 chmod 777 /tmp/test
```

```
chmod: Unable to change file mode on /tmp/test: Operation not permitted
```

Log output:

```
Sep  2 19:54:35 macbox sandboxd[43051]: chmod(43050) deny file-write-mode  
/private/tmp/test
```

File-write-mount

Syntax: (action file-write-mount [filter] [modifier])

Actions: allow deny

Filters: path file-mode

Modifiers: send-signal no-log

Description:

Access control check for mounting a file system.

Example(s):

N/A (tried different combinations and mount still works!)

File-write-owner

Syntax: (action file-write-owner [filter] [modifier])

Actions: allow deny

Filters: path file-mode

Modifiers: send-signal no-log

Description:

Control access to file ownership changes.

Example(s):

- (deny file-write-owner (literal "/private/tmp/test"))

```
# sandbox-exec -f ls2 chown nobody /tmp/test
```

```
chown: /tmp/test: Operation not permitted
```

Log output:

```
Sep  2 20:05:48 macbox sandboxd[43419]: chown(43418) deny file-write-owner  
/private/tmp/test
```

File-write-setugid

Syntax: (action file-write-setugid [filter] [modifier])

Actions: allow deny

Filters: path file-mode

Modifiers: send-signal no-log

Description:

Not implemented???

Example(s):

File-write-times

Syntax: (action file-write-times [filter] [modifier])

Actions: allow deny

Filters: path file-mode

Modifiers: send-signal no-log

Description:

Access timestamps?

“Determine whether the subject identified by the credential can set the passed access timestamps on the passed vnode.”

Example(s):

N/A

File-write-unmount

Syntax: (action file-write-unmount [filter] [modifier])

Actions: allow deny

Filters: path file-mode

| |
|-------------------------------|
| Modifiers: send-signal no-log |
|-------------------------------|

Description:

Access control check for unmounting a filesystem.

Example(s):

- (deny file-write-unmount (literal "/Volumes/Mac OS X Install ESD"))

```
# sandbox-exec -f ls2 umount /Volumes/Mac\ OS\ X\ Install\ ESD/  
umount: unmount(/Volumes/Mac OS X Install ESD): Operation not permitted
```

Log output:

```
Sep  2 20:21:19 macbox sandboxd[43908]: umount(43911) deny file-write-unmount  
/Volumes/Mac OS X Install ESD
```

File-write-xattr

| | |
|------------|---|
| Syntax: | (action file-write-xattr [filter] [modifier]) |
| Actions: | allow deny |
| Filters: | path file-mode xattr |
| Modifiers: | send-signal no-log |

Description:

This operation will control write access to the file extended attributes.

Example(s):

- (deny file-write-xattr (literal "/test"))

```
$ xattr -w test 123 /test  
$ xattr -l /test  
test: 123  
$ sandbox-exec -f ls2 xattr -w test2 123 /test  
xattr: [Errno 1] Operation not permitted: '/test'
```

Log output:

```
Sep  2 00:38:13 macbox sandboxd[25217]: Python(25216) deny file-write-xattr /test
```

Ipc*

| | |
|------------|--------------------------|
| Syntax: | (action ipc* [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Description:

This operation will IPC related operations described below.

Example(s):

lpc-posix*

| | |
|------------|--------------------------------|
| Syntax: | (action ipc-posix* [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Description:

This operation will IPC POSIX related operations described below.

Example(s):

lpc-posix-sem

| | |
|------------|-----------------------------------|
| Syntax: | (action ipc-posix-sem [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Description:

Controls access to POSIX semaphores (create, open, post, unlink, wait).

Example(s):

lpc-posix-shm

| | |
|------------|-----------------------------------|
| Syntax: | (action ipc-posix-shm [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Description:

Controls access to POSIX shared memory region (create, mmap, open, stat, truncate, unlink).

Example(s):

lpc-sysv*

| | |
|------------|-------------------------------|
| Syntax: | (action ipc-sysv* [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Description:

This operation will IPC POSIX related operations described below.

Example(s):

ipc-sysv-msg

| | |
|------------|----------------------------------|
| Syntax: | (action ipc-sysv-msg [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Definition:

Controls access to System V messages (enqueue, msgrcv, msgrmid, msqctl, msqget, msqrcv, msqsnd).

Example(s):

ipc-sysv-sem

| | |
|------------|----------------------------------|
| Syntax: | (action ipc-sysv-sem [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Definition:

Controls access to System V semaphores (semctl, semget, semop).

Example(s):

ipc-sysv-shm

| | |
|------------|----------------------------------|
| Syntax: | (action ipc-sysv-shm [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Definition:

Controls access to mapping System V shared memory (shmat, shmctl, shmdt, shmget).

Example(s):

Mach*

| | |
|------------|---------------------------|
| Syntax: | (action mach* [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Definition:

Example(s):

Mach-bootstrap

Syntax: (action mach-bootstrap [modifier])

Actions: allow deny

Filters: n/a

Modifiers: send-signal no-log

Definition:

Example(s):

Mach-lookup

Syntax: (action mach-lookup [modifier])

Actions: allow deny

Filters: n/a

Modifiers: send-signal no-log

Definition:

Example(s):

Mach-priv*

Syntax: (action mach-priv* [modifier])

Actions: allow deny

Filters: n/a

Modifiers: send-signal no-log

Definition:

Example(s):

Mach-priv-host-port

Syntax: (action mach-priv-host-port [modifier])

Actions: allow deny

Filters: n/a

Modifiers: send-signal no-log

Definition:

Example(s):

Mach-priv-task-port

| | |
|------------|---|
| Syntax: | (action mach-priv-task-port [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Definition:

Example(s):

Mach-task-name

| | |
|------------|------------------------------------|
| Syntax: | (action mach-task-name [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Definition:

Example(s):

Network*

| | |
|------------|---------------------------------------|
| Syntax: | (action network* [filter] [modifier]) |
| Actions: | allow deny |
| Filters: | network path file-mode |
| Modifiers: | send-signal no-log |

Definition:

Controls all available network operations described below.

Example(s):

- (deny network* (remote ip “*:80”))

```
$ sandbox-exec -f ls2 nc www.google.com 80
```

Log output:

```
Sep  2 21:12:00 macbox sandboxd[45542]: nc(45540) deny network-outbound 74.125.39.99:80
```

Network-inbound

| | |
|------------|--|
| Syntax: | (action network-inbound [filter] [modifier]) |
| Actions: | allow deny |
| Filters: | network path file-mode |
| Modifiers: | send-signal no-log |

Definition:

Controls network inbound operations.

“A socket has a queue for receiving incoming data. When a packet arrives on the wire, it eventually gets deposited into this queue, which the owner of the socket drains when they read from the socket's file descriptor.”

Example(s):

- (allow network-inbound (local ip4 "*:22))

Network-bind

Syntax: (action network-bind [filter] [modifier])

Actions: allow deny

Filters: network path file-mode

Modifiers: send-signal no-log

Definition:

Control access to socket bind().

Example(s):

- (deny network-bind (local ip "*:7890"))

```
$ sandbox-exec -f ls2 nc -l 7890
```

nc: Operation not permitted

Log output:

```
Sep 2 21:08:41 macbox sandboxd[45438]: nc(45437) deny network-bind 0.0.0.0:7890
```

Network-outbound

Syntax: (action network-outbound [filter] [modifier])

Actions: allow deny

Filters: network path file-mode

Modifiers: send-signal no-log

Definition:

Controls access to send data to the socket.

Example(s):

- (deny network-outbound)

This will deny any packets going out from the target application.

- (deny network-outbound (remote ip "*:80"))

```
$ sandbox-exec -f ls2 nc www.google.com 80
```

Log output:

```
Sep 2 22:29:03 macbox sandboxd[47760]: nc(47758) deny network-outbound
74.125.39.106:80
```

- (allow network-outbound (remote unix-socket (path-literal "/private/var/run/syslog")))

Allow access to the syslog unix socket.

Process*

| | |
|------------|------------------------------|
| Syntax: | (action process* [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Definition:

Controls all available process operations described below. One important detail is that no filters are available here but are on process-exec.

Example(s):

- (deny process*)

```
$ sandbox-exec -f ls2 ls
```

```
sandbox-exec: ls: Operation not permitted
```

Log output:

```
Sep 2 22:36:09 macbox sandboxd[47975]: sandbox-exec(47980) deny process-exec /bin/ls
```

Process-exec

| | |
|------------|---|
| Syntax: | (action process-exec [filter] [modifier]) |
| Actions: | allow deny |
| Filters: | path file-mode |
| Modifiers: | send-signal no-log no-sandbox |

Definition:

Control process execution.

Example(s):

- (deny process-exec (literal "/bin/ls"))

```
$ sandbox-exec -f ls2 /bin/ls
```

```
sandbox-exec: /bin/ls: Operation not permitted
```

```
$ sandbox-exec -f ls2 ls
```

```
sandbox-exec: ls: Operation not permitted
```

Log output:

```
Sep 2 01:16:57 macbox sandboxd[26360]: sandbox-exec(26359) deny process-exec /bin/ls
```

```
Sep 2 01:17:00 macbox sandboxd[26360]: sandbox-exec(26363) deny process-exec /bin/ls
```

Process-fork

| | |
|----------|----------------------------------|
| Syntax: | (action process-fork [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |

| |
|-------------------------------|
| Modifiers: send-signal no-log |
|-------------------------------|

Definition:

Control access to fork and vfork.

Example(s):

- (deny process-fork)

```
$ ./forktest
child!
parent!
$ sandbox-exec -f ls2 ./forktest
parent!
```

Log output:

```
Sep  2 01:23:52 macbox sandboxd[26677]: forktest(26676) deny process-fork
```

Signal

| |
|---|
| Syntax: (action signal [filter] [modifier]) |
| Actions: allow deny |
| Filters: signal |
| Modifiers: send-signal no-log |

Definition:

Control if program can send signals to itself, processes in the same group or all other processes.

Example(s):

- (deny signal (target others))

The sandboxed process will not be able to send signals to other processes.

```
$ sandbox-exec -f ls2 kill -ALRM 10229
kill: 10229: Operation not permitted
```

Log output:

```
Sep  2 10:45:01 macbox sandboxd[31416]: kill(31418) deny signal
```

Sysctl*

| |
|-------------------------------------|
| Syntax: (action sysctl* [modifier]) |
| Actions: allow deny |
| Filters: n/a |
| Modifiers: send-signal no-log |

Definition:

Control all access to sysctl() and its variants, sysctlbyname and sysctlnametomib.

Example(s):

- (deny sysctl*)

```
$ sandbox-exec -f ls2 sysctl debug
```

Log output:

```
Sep  2 01:33:50 macbox sandboxd[26952]: sysctl(26960) deny sysctl-read
# sandbox-exec -f ls2 sysctl -w debug.bpf_bufsize=1024
second level name bpf_bufsize in debug.bpf_bufsize is invalid
This happens because sysctl-read is also denied so it can't read the name.
```

Sysctl-read

| | |
|------------|---------------------------------|
| Syntax: | (action sysctl-read [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Definition:

Control read access to sysctl() and its variants, sysctlbyname and sysctlnametomib.

Example(s):

- (deny sysctl-read)

```
$ sandbox-exec -f ls2 sysctl debug
```

Log output:

```
Sep  2 01:40:01 macbox sandboxd[27171]: sysctl(27170) deny sysctl-read
# sandbox-exec -f ls2 sysctl -w debug.bpf_bufsize=1024
second level name bpf_bufsize in debug.bpf_bufsize is invalid
```

Sysctl-write

| | |
|------------|----------------------------------|
| Syntax: | (action sysctl-write [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Definition:

Control write access to sysctl() and its variants, sysctlbyname and sysctlnametomib.

Note: there seems to be a bug in this implementation (Snow Leopard at least), where a (deny sysctl-write) requires a (allow sysctl-read), even if we have a (allow default).

Test command:

```
# sandbox-exec -f ls2 sysctl -w debug.bpf_bufsize=1024
```

Test profile:

(version 1)

(debug all)

(allow default)

(deny sysctl-write)

But it works if written this way:

(version 1)

(debug all)

(allow default)

(deny sysctl-write)

(allow sysctl-read)

System*

Syntax: (action system* [modifier])

Actions: allow deny

Filters: n/a

Modifiers: send-signal no-log

Definition:

Controls all available system operations described below.

Example(s):

- (deny system*)

```
# sandbox-exec -f ls2 date 01212200
```

```
date: settimeofday (timeval): Operation not permitted
```

Log output:

```
Sep 2 22:49:30 macbox sandboxd[48428]: date(48435) deny system-set-time
```

System-acct

Syntax: (action system-acct [modifier])

Actions: allow deny

Filters: n/a

Modifiers: send-signal no-log

Description:

Determine whether the target should be allowed to enable accounting, based on its label and the label of the accounting log file. See acct(5) for more information.

Example(s):

- (allow system-acct)

System-audit

Syntax: (action system-audit [modifier])

Actions: allow deny

Filters: n/a

Modifiers: send-signal no-log

Description:

Determine whether the target can submit an audit record for inclusion in the audit log via the audit() system call.

Example(s):

- (allow system-audit)

System-fsctl

Syntax: (action process* [modifier])

Actions: allow deny

Filters: n/a

Modifiers: send-signal no-log

Description:

Control access to fsctl().

Warning: The fsctl() system call is directly analogous to ioctl(); since the associated data is opaque from the standpoint of the MAC framework and since these operations can affect many aspects of system operation, policies must exercise extreme care when implementing access control checks.

Example(s):

- (deny system-fsctl)

System-lcid

Syntax: (action system-lcid [modifier])

Actions: allow deny

Filters: n/a

Modifiers: send-signal no-log

Description:

Determine whether the target can relabel itself to the supplied new label (newlabel). This access control check is called when the mac_set_lctx/lcid system call is invoked. A user space application will supply a new value, the value will be internalized and provided in newlabel.

Example(s):

- (allow system-lcid)

System-mac-label

Syntax: (action system-mac-label [modifier])

Actions: allow deny

Filters: n/a

Modifiers: send-signal no-log

Description:

Determine whether the target can perform the `mac_set_fd` operation. The `mac_set_fd` operation is used to associate a MAC label with a file.

Example(s):

- (deny system-mac-label)

System-nfssvc

| | |
|------------|-----------------------------------|
| Syntax: | (action system-nfssvc [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Description:

Determine whether the target should be allowed to call `nfssrv(2)`.

Example(s):

- (allow system-nfssvc)

System-reboot

| | |
|------------|-----------------------------------|
| Syntax: | (action system-reboot [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Description:

Controls if target can reboot system.

Note: doesn't seem to work!

Example(s):

- (deny system-reboot)

System-set-time

| | |
|------------|-------------------------------------|
| Syntax: | (action system-set-time [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Description:

Controls access to the system clock.

Example(s):

- (deny system-set-time)

```
# sandbox-exec -f ls2 date 01212200
```

```
date: settimeofday (timeval): Operation not permitted
```

Log output:

Sep 2 22:49:30 macbox sandboxd[48428]: date(48435) deny system-set-time

System-socket

| | |
|------------|-----------------------------------|
| Syntax: | (action system-socket [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Description:

Control access to create (raw?) sockets.

Example(s):

- (deny system-socket)

System-swap

| | |
|------------|---------------------------------|
| Syntax: | (action system-swap [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Description:

Access control check for swap devices (swapon/swapoff).

Example(s):

- (allow system-swap)

System-write-bootstrap

| | |
|------------|--|
| Syntax: | (action system-write-bootstrap [modifier]) |
| Actions: | allow deny |
| Filters: | n/a |
| Modifiers: | send-signal no-log |

Description:

Not implemented???

Example(s):

Job-creation

| | |
|----------|---|
| Syntax: | (action job-creation [filter] [modifier]) |
| Actions: | allow deny |
| Filters: | path |

| |
|-------------------------------|
| Modifiers: send-signal no-log |
|-------------------------------|

Description:

Not implemented ???

Example(s):

Mach-per-user-lookup

| |
|--|
| Syntax: (action mach-per-user-lookup [modifier]) |
|--|

| |
|---------------------|
| Actions: allow deny |
|---------------------|

| |
|--------------|
| Filters: n/a |
|--------------|

| |
|-------------------------------|
| Modifiers: send-signal no-log |
|-------------------------------|

Description:

Example(s):

5.3 – Filters

Filters can be applied to the operations that support them, allowing better control and granularity. The filters can be path names, file names, IP addresses, extended attributes, file modes. Regular expressions are supported in some.

The following table resumes the existing filters:

| | | | | | |
|------|---------|-----------|-------|------|--------|
| path | network | file-mode | xattr | mach | signal |
|------|---------|-----------|-------|------|--------|

Anything included in square braces “[]” is optional.

Path

Description:

Match filenames or paths.

Three different modes are supported, regular expressions, literal, and subpath.

Symlinks are resolved so a path filter (literal or regex matching the beginning) to “/tmp/testfile” will fail because “/tmp” is a symbolic link to “/private/tmp”. In this case the correct filter should be “/private/tmp/testfile”.

1. Regular Expressions

Syntax:

(regex EXPRESSION)

Example(s):

- (allow file-read* (regex #”^/usr/lib/*”))

This will allow file reading access to all files available under /usr/lib/.

Multiple regular expressions are supported, so the operation can apply to multiple paths and/or files.

- (allow file-read*
 (regex
 #"^/usr/lib/*"
 #"^/dev/*"
 #"^/System/Library/Frameworks/*"
)
)

2. Literal

Syntax:

(literal PATH)

Example(s):

- (deny file-read* (literal "/dev"))

This will deny all file read access to /dev only, but everything else inside /dev isn't protected by this operation.

```
$ sandbox-exec -f ls2 ls /dev
```

```
ls: /dev: Operation not permitted
```

```
$ sandbox-exec -f ls2 ls /dev/dtrace
```

```
/dev/dtrace
```

3. Subpath

Syntax:

(subpath PATH)

Note: the PATH never ends with a slash (/).

Example(s):

- (deny file-read* (subpath "/dev"))

In this case, everything under /dev will be denied read access (including /dev itself).

Network

Description:

Filter by network protocol and source or destination.

Syntax:

(local ip | ip4 | ip6 | tcp | tcp4 | tcp6 | udp | udp4 | udp6 ["IP:PORT"])

(remote ip | ip4 | ip6 | tcp | tcp4 | tcp6 | udp | udp4 | udp6 ["IP:PORT"])

(remote unix | unix-socket [path-literal PATH])

The default "IP:PORT" is "*:*". The only valid input for IP is localhost or *, meaning that you can only filter by port.

The aliases "from", "to", and "unix-socket" can be used instead of "local", "remote", and "unix".

The ICMP protocol is included in the IP and UDP options.

Note:

In this case, PATH must be "path-literal" instead of "regex", "literal", or "subpath".

Example(s):

- (deny network* (remote ip "*:*))

Deny IP access to any remote host.

```
$ sandbox-exec -f ls2 ping www.google.com
PING www.l.google.com (74.125.39.147): 56 data bytes
ping: sendto: Operation not permitted
```

Log output:

```
Sep  2 11:00:17 macbox sandboxd[31870]: ping(31869) deny network-outbound
74.125.39.147:0
```

- (deny network* (remote tcp "*:~"))

Deny TCP access to any remote host.

```
$ sandbox-exec -f ls2 telnet www.google.com 80
Trying 74.125.39.147...
telnet: connect to address 74.125.39.147: Operation not permitted
```

Log output:

```
Sep  2 11:02:20 macbox sandboxd[31937]: telnet(31935) deny network-outbound
74.125.39.147:80
```

- (deny network* (local tcp "*:~"))

Deny TCP access to localhost ports.

```
$ telnet localhost 22
Trying 127.0.0.1...
telnet: connect to address 127.0.0.1: Connection refused
```

Log output:

```
Sep  2 11:04:49 macbox sandboxd[32011]: telnet(32010) deny network-outbound 127.0.0.1:22
```

- (allow network* (remote unix-socket (path-literal "/private/var/run/syslog")))

File-mode

Description:

Match file mode bits.

Syntax:

(file-mode #oFILEMODE)

where FILEMODE is composed of 4 bits.

Note: The match will be successful if each bit is equal or higher, meaning by this that a #o0644 will be successfully matched by a file with a mode of 0644, 0744, 0657, etc.

Example(s):

(file-mode #o0644)

Filter will match if target has permissions of 0644 (-rw-r--r-) or higher.

Xattr

Description:

Match the extended attribute name, not content.

Syntax:

(xattr REGEX)

Example(s):

- (deny file-write-xattr (xattr "test_xattr"))

Deny writing the extended attribute named "test_xattr" to any file.

```
$ xattr -w test_xattr aaaa /tmp/xattr
```

```
$ xattr -l /tmp/xattr
```

```
test_xattr: aaaa
```

```
$ sandbox-exec -f ls2 xattr -w test_xattr aaaa /tmp/xattr
```

```
xattr: [Errno 1] Operation not permitted: '/tmp/xattr'
```

Log output:

```
Sep  2 11:48:02 macbox sandboxd[33295]: Python(33294) deny file-write-xattr
/private/tmp/xattr
```

Mach

Description:

These are needed for things like getpwnam, hostname changes, & keychain.

Syntax:

(global-name REGEX|LITERAL)

(local-name REGEX|LITERAL)

Example(s):

- (allow mach-lookup


```
(global-name
    "com.apple.bsd.dirhelper"
    "com.apple.distributed_notifications.2"
  )
)
```

Signal

Description:**Syntax:**

(target self | pgrp | others)

where,

self: sandboxed process itself

pgrp: group processes ?

others: all processes

Example(s):

- (deny signal (target others))

The sandboxed process will not be able to send signals to other processes.

```
$ sandbox-exec -f ls2 kill -ALRM 10229
```

```
kill: 10229: Operation not permitted
```

Log output:

Sep 2 10:45:01 macbox sandboxd[31416]: kill(31418) deny signal

5.4 – Modifiers

There are three available modifiers, although one just applies to a single operation. The modifiers are send-signal, no-log, and no-sandbox. To use them you will need the keyword “with”.

1. Send-signal

Description:

The best description is found in Apple’s scripts:

“To help debugging, "with send-signal SIGFPE" will trigger a fake floating-point exception, which will crash the process and show the call stack leading to the offending operation.

For the shipping version "deny" is probably better because it vetoes the operation without killing the process.“

There is a special exception, where send-signal doesn’t apply to mach-* operations.

It can be applied to allow and deny actions.

Syntax:

(with send-signal SIGNAL)

Example(s):

- (deny file-read* (with send-signal SIGFPE))

The target binary will crash with a floating point exception when it tries to read any file.

```
$ sandbox-exec -f ls2 cat /tmp/test
```

Floating point exception

2. No-log

Description:

Do not log denied operations. Applies only to deny action.

Syntax:

(with no-log)

Example(s):

- (deny file-read* (subpath “/tmp”) (with no-log))

3. No-sandbox

Description:

Applies only to allow action and process-exec operation.

Syntax:

(with no-sandbox)

Example(s):

???

5.5 - Other keywords

▪ **require-any and require-all**

These are the keywords for logical OR and logical AND.

Syntax:

```
(require-any (filter1) (filter2) ...)
```

Example:

```
(deny file-read-data
  (require-all
    (file-mode #o0644)
    (subpath "/private")
    (literal "/private/tmp/test2"))
)
```

In this case, reading the contents of the file test2 located in /tmp will only be denied if it matches all the three filters (the subpath filter is somewhat redundant in this case).

```
$ sandbox-exec -f ls2 cat /tmp/test2
```

```
cat: /tmp/test2: Operation not permitted
```

Log output:

```
Sep  3 23:27:44 whq sandboxd[13401]: cat(13400) deny file-read-data /private/tmp/test2
```

```
$ chmod 0614 /tmp/test2
```

```
$ sandbox-exec -f ls2 cat /tmp/test2
```

```
aaaaaaaaaa
```

6 – Special hardcoded cases

The following special cases can be found inside the code:

- Allow mach-bootstrap if mach-lookup is ever allowed.
- Allow access to webdavfs_agent if file-read* is always allowed.
- Never allow a sandboxed process to open a launchd socket.c

7 – A sample profile for MS Outlook 2011