S.A.R.A. Documentation

Release 0.2

Salvatore Mesoraca

May 30, 2018

Contents

CHAPTER 1

Introduction

S.A.R.A. (S.A.R.A. is Another Recursive Acronym) is a stacked Linux Security Module that aims to collect heterogeneous security measures, providing a common interface to manage them. As of today it consists of one submodule:

- WX Protection

The kernel-space part is complemented by its user-space counterpart: *saractl*[1]. A test suite for WX Protection, called *sara-test*[3], is also available. You can also visit the *official home page of S.A.R.A.*[4].

At the time of writing S.A.R.A. has been proposed for upstreaming, but it's still out of tree.

1.1 S.A.R.A.'s Submodules

1.1.1 WX Protection

WX Protection aims to improve user-space programs security by applying:

- *W^X enforcement*
- *W!->X (once writable never executable) mprotect restriction*
- *Executable MMAP prevention*

All of the above features can be enabled or disabled both system wide or on a per executable basis through the use of configuration files managed by *saractl*[1].

It is important to note that some programs may have issues working with WX Protection. In particular:

- **W^X enforcement** will cause problems to any programs that needs memory pages mapped both as writable and executable at the same time e.g. programs with executable stack markings in the *PT_GNU_STACK* segment.

[1] saractl
[3] sara-test
[4] Homepage

- **W!->X mprotect restriction** will cause problems to any program that needs to generate executable code at run time or to modify executable pages e.g. programs with a *JIT* compiler built-in or linked against a *non-PIC* library.
- **Executable MMAP prevention** can work only with programs that have at least partial *RELRO* support. It's disabled automatically for programs that lack this feature. It will cause problems to any program that uses *dlopen* or tries to do an executable mmap. Unfortunately this feature is the one that could create most problems and should be enabled only after careful evaluation.

To extend the scope of the above features, despite the issues that they may cause, they are complemented by */proc/PID/attr/sara/wxprot interface* and *Trampoline emulation*. It's also possible to override the centralized configuration via *Extended filesystem attributes*.

At the moment, WX Protection (unless specified otherwise) should work on any architecture supporting the NX bit, including, but not limited to: *x86_64*, *x86_32* (with PAE), *ARM* and *ARM64*.

Parts of WX Protection are inspired by some of the features available in PaX.

W^X enforcement

W^X means that a program can't have a page of memory that is marked, at the same time, writable and executable. This also allow to detect many bad behaviours that make life much more easy for attackers. Programs running with this feature enabled will be more difficult to exploit in the case they are affected by some vulnerabilities, because the attacker will be forced to make more steps in order to exploit them. This feature also blocks accesses to /proc/*/mem files that would allow to write the current process read-only memory, bypassing any protection.

W!->X (once writable never executable) mprotect restriction

"Once writable never executable" means that any page that could have been marked as writable in the past won't ever be allowed to be marked (e.g. via an mprotect syscall) as executable. This goes on the same track as W^X, but is much stricter and prevents the runtime creation of new executable code in memory. Obviously, this feature does not prevent a program from creating a new file and *mmapping* it as executable, however, it will be way more difficult for attackers to exploit vulnerabilities if this feature is enabled.

Executable MMAP prevention

This feature prevents the creation of new executable mmaps after the dynamic libraries have been loaded. When used in combination with **W!->X mprotect restriction** this feature will completely prevent the creation of new executable code from the current thread. Obviously, this feature does not prevent cases in which an attacker uses an *execve* to start a completely new program. This kind of restriction, if needed, can be applied using one of the other LSM that focuses on MAC. Please be aware that this feature can break many programs and so it should be enabled after careful evaluation.

/proc/PID/attr/sara/wxprot interface

The *procattr* interface can be used by a thread to discover which WX Protection features are enabled and/or to tighten them: protection can't be softened via procattr. The interface is simple: it's a text file with an hexadecimal number in it representing enabled features (more information can be found in the *Flags values* section). Via this interface it is also possible to perform a complete memory scan to remove the write permission from pages that are both writable and executable.

Protections that prevent the runtime creation of executable code can be troublesome for all those programs that actually need to do it e.g. programs shipping with a JIT compiler built-in. This feature can be use to run the JIT compiler with few restrictions while enforcing full WX Protection in the rest of the program.

The preferred way to access this interface is via *libsara*[2]. If you don't want it as a dependency, you can just statically link it in your project or copy/paste parts of it. To make things simpler *libsara* is the only part of S.A.R.A. released under *CC0 - No Rights Reserved* license.

Extended filesystem attributes

When this functionality is enabled, it's possible to override WX Protection flags set in the main configuration via extended attributes, even when S.A.R.A.'s configuration is in "locked" mode. If the user namespace is also enabled, its attributes will override settings configured via the security namespace. The xattrs currently in use are:

- security.sara.wxprot
- user.sara.wxprot

They can be manually set to the desired value as a decimal, hexadecimal or octal number. When this functionality is enabled, S.A.R.A. can be easily used without the help of its userspace tools. Though the preferred way to change these attributes is *sara-xattr* which is part of *saractl*[1].

Trampoline emulation

Some programs need to generate part of their code at runtime. Luckily enough, in some cases they only generate well-known code sequences (the *trampolines*) that can be easily recognized and emulated by the kernel. This way WX Protection can still be active, so a potential attacker won't be able to generate arbitrary sequences of code, but just those that are explicitly allowed. This is not ideal, but it's still better than having WX Protection completely disabled.

In particular S.A.R.A. is able to recognize trampolines used by GCC for nested C functions and libffi's trampolines. This feature is available only on *x86_32* and *x86_64*.

Flags values

Flags are represented as a 16 bit unsigned integer in which every bit indicates the status of a given feature:

Feature	Value
W!->X Heap	0x0001
W!->X Stack	0x0002
W!->X Other memory	0x0004
W^X	0x0008
Don't enforce, just complain	0x0010
Be Verbose	0x0020
Executable MMAP prevention	0x0040
Force W^X on setprocattr	0x0080
Trampoline emulation	0x0100
Children will inherit flags	0x0200

1.2 Bugs

Please report any issue to the relevant issue tracker:

- saractl
- libsara

[2] libsara

- sara-test
- kernel

CHAPTER 2

Install

As of today S.A.R.A. it's still out of tree, so, if you want to use it, you'll need to use a patched kernel image or patch the kernel yourself.

2.1 Generic Instructions

The stable patch releases can be found here. The releases are signed with the following PGP key: `0xD7286260BBF31719A2759FA485F0580B9DACBE6E`. To apply the patches change the working directory to that of your kernel sources and enter the usual patch command:

```
$ cd kernel-sources/
$ patch -p1 < ../sara.patch
```

where `../sara.patch` is the path to the uncompressed and PGP-verified patch.

To install *saractl* you can use `pip`:

```
$ pip install saractl
```

Or you can manually download the latest archive from here, check its PGP signature, extract it and install it by simply running:

```
$ python3 setup.py install
```

You we'll need at least Python 3.4. Some saractl's functionalities depend on `pyelftools`, `pythonprctl` and `pyxattr`. You may want to install them too, but they are optional. Alternatively you can download the sources tarball from here too.

To install the other userspace tools just download the latest GitHub releases from the links below and follow the instructions in their README file. Please, always remeber to check PGP signatures.

- libsara
- sara-test

CHAPTER 3

Kernel Configuration

CONFIG_SECURITY_SARA - Enable S.A.R.A.

This selects S.A.R.A. LSM, which aims to collect heterogeneous security measures providing a common interface to manage them. This LSM will always be stacked with the selected primary LSM and other stacked LSMs.

CONFIG_SECURITY_SARA_DEFAULT_DISABLED - S.A.R.A. will be disabled at boot

If you say Y here, S.A.R.A. will not be enabled at startup. You can override this option at boot time via "sara.enabled=[1|0]" kernel parameter or via user-space utilities. This option is useful for distro kernels.

CONFIG_SECURITY_SARA_NO_RUNTIME_ENABLE - S.A.R.A. can be turn on only at boot time

By enabling this option it won't be possible to turn on S.A.R.A. at runtime via user-space utilities. However it can still be turned on at boot time via the "sara.enabled=1" kernel parameter. This option is functionally equivalent to "sara.enabled=0" kernel parameter. This option is useful for distro kernels.

CONFIG_SECURITY_SARA_WXPROT - WX Protection: W^X and W!->X protections

WX Protection aims to improve user-space programs security by applying:

- W^X memory restriction
- W!->X (once writable never executable) mprotect restriction
- Executable MMAP prevention

See *WX Protection*.

Default action for W^X and W!->X protections

Choose the default behaviour of WX Protection when no config rule matches or no rule is loaded.

CONFIG_SECURITY_SARA_WXPROT_DEFAULT_FLAGS_ALL_COMPLAIN_VERBOSE - Protections enabled but not enforced

All features enabled except "Executable MMAP prevention", verbose reporting, but no actual enforce: it just complains. Its numeric value is 0x3f. See *Flags values*.

CONFIG_SECURITY_SARA_WXPROT_DEFAULT_FLAGS_ALL_ENFORCE_VERBOSE - Full protection, verbose

All features enabled except "Executable MMAP prevention". The enabled features will be enforced with verbose reporting. Its numeric value is 0x2f. See *Flags values*.

CONFIG_SECURITY_SARA_WXPROT_DEFAULT_FLAGS_ALL_ENFORCE - Full protection, quiet

All features enabled except "Executable MMAP prevention". The enabled features will be enforced quietly. Its numeric value is 0xf. See *Flags values*.

CONFIG_SECURITY_SARA_WXPROT_DEFAULT_FLAGS_NONE - No protection at all

All features disabled. Its numeric value is 0. See *Flags values*.

CONFIG_SECURITY_SARA_WXPROT_EMUTRAMP - Enable emulation for some types of trampolines

Some programs and libraries need to execute special small code snippets from non-executable memory pages. Most notable examples are the GCC and libffi trampolines. This features make it possible to execute those trampolines even if they reside in non-executable memory pages. This features need to be enabled on a per-executable basis via user-space utilities. See *Trampoline emulation*.

CONFIG_SECURITY_SARA_WXPROT_XATTRS_ENABLED - xattrs support enabled by default

If you say Y here it will be possible to override WX protection configuration via extended attributes in the security namespace. Even when S.A.R.A.'s configuration has been locked. See *Extended filesystem attributes*.

CONFIG_CONFIG_SECURITY_SARA_WXPROT_XATTRS_USER - 'user' namespace xattrs support enabled by default

If you say Y here it will be possible to override WX protection configuration via extended attributes in the user namespace. Even when S.A.R.A.'s configuration has been locked. See *Extended filesystem attributes*.

CONFIG_SECURITY_SARA_WXPROT_DISABLED - WX protection will be disabled at boot

If you say Y here WX protection won't be enabled at startup. You can override this option via user-space utilities or at boot time via "sara.wxprot_enabled=[0|1]" kernel parameter.

CHAPTER 4

Kernel Parameters

sara.enabled= Disable or enable S.A.R.A. at boot time.

If disabled this way, S.A.R.A. can’t be enabled again.

Format: { “0” | “1” }

See *Kernel Configuration*

0 – disable.

1 – enable.

Default value is set via kernel config option.

sara.wxprot_enabled= Disable or enable S.A.R.A. WX Protection at boot time.

Format: { “0” | “1” }

See *Kernel Configuration*

0 – disable.

1 – enable.

Default value is set via kernel config option.

sara.wxprot_default_flags= Set S.A.R.A. WX Protection default flags.

Format: <integer>

See *Flags values*

Default value is set via kernel config option.

sara.wxprot_xattrs_enabled= Enable support for security xattrs.

Format: { “0” | “1” }

See *Kernel Configuration*

0 – disable.

1 – enable.

Default value is set via kernel config option.

sara.wxprot_xattrs_user= Enable support for user xattrs.

Format: { "0" | "1" }

See *Kernel Configuration*

0 – disable.

1 – enable.

Default value is set via kernel config option.

CHAPTER 5

Userspace Tools Manpages

5.1 saractl manual page

5.1.1 Synopsys

saractl [*options*] <*command*> [*command-specific options* . . .]

5.1.2 Description

`saractl` is the userspace utility that manages S.A.R.A. LSM's configurations.

5.1.3 Commands

load Load configurations. If a config is already present and up to date it won't be loaded again (-s is ignored).

startup Load configurations for the first time at boot (-s is ignored).

enable Enable S.A.R.A.

disable Disable S.A.R.A.

status Get S.A.R.A. status.

lock Prevent changing the config until next reboot.

config_to_file Generate various binary formats to import the configuration without saractl (-s is ignored).

test Run some self-tests.

5.1.4 Options

-h, --help	show this help message and exit

-v, --verbose Be verbose. For extra verbosity use multiple -v.

-q, --quiet Suppress any output.

-V, --version show program's version number and exit

-c CONFIG_DIR, --config-dir CONFIG_DIR Specify config directory. Defaults to "/etc/sara"

-S SECURITYFS, --securityfs SECURITYFS The mount point of the securityfs. Defaults to "/sys/kernel/security".

-s {main,wxprot}, –submodule {main,wxprot} Select the submodule you want to work with. Available submodules: wxprot. Defaults to "main".

-f, --force Force reload even if the config is already up to date (to use only after the *load* command).

-F {binary,sh,c}, –output-format {binary,sh,c} Select the desired output format. Available formats: "binary", "sh" and "c". Defaults to "binary" (to use only after the *config_to_file* command).

-o OUTPUT, --output OUTPUT Output file or directory. Defaults to "./output/" directory for "binary" format, "./output.sh" file for "sh" format and "./output.c" file for "c" format (to use only after the *config_to_file* command).

5.1.5 Examples

Reload the config:

```
saractl load
```

Create a stand alone shell script to load the config:

```
saractl config_to_file -F sh -o ./script.sh
```

5.1.6 Config file

The main configuration file for saractl can be found in */etc/sara/main.conf*. These are the available options:

```
sara_enabled=1                          # enable S.A.R.A. LSM

sara_locked=0                           # lock S.A.R.A. config
                                        # when it has been loaded

wxprot_enabled=1                        # enable WX Protections

wxprot_emutramp_missing_default=none    # default option to use
                                        # when emutramp is not
                                        # supported.
                                        # It can be set to "none"
                                        # or "mprotect".

wxprot_xattr_enabled=0                  # enable security XATTRs
                                        # support

wxprot_xattr_user_allowed=0             # enable user XATTRs support
```

5.1.7 Bugs

If you find any bugs, please report them at <https://github.com/smeso/saractl/issues>

5.1.8 See also

`sara(7)`, `wxprot.conf(5)`, `sara-xattr(8)` and <https://sara.smeso.it>

5.2 wxprot.conf manual page

5.2.1 Description

This man page describes the format of S.A.R.A.'s WX Protections configuration files. See `sara(7)` for an overview of S.A.R.A. or `saractl(8)` for an overview of the program used to manage S.A.R.A's configuration.

5.2.2 Format

The configuration format is line oriented. Comments starts with #, inline comments are supported. Every line is made up of two parts seperated by a whitespace. The first part is the file path, in case it contains a whitespace itself, the string can be enclosed in double quotes or escaped. The path can be terminated with a '*' to make it match every path that starts with the chosen prefix. The second part is the flags list. It's a case in-sensitive and comma separated list of the flags that need to be enabled. It can include whitespaces before or after the commas. Files in the */etc/sara/wxprot.conf.d/* directory are read in lexycografical order and merged together at the end of */etc/sara/wxprot.conf* as if they were a single big file. In general, lines order doesn't matter, the rule with the most specific path has precedence. In case of multiple entries with *exactly* the same path, the first one has precedence and others are discarded.

Flags

WXORX Prevents any page of memory from being marked as both writable and executable at the same time.

STACK Prevents any page of memory in the stack from becoming executable if it could have been written in the past. (Depends on WXORX)

HEAP Prevents any page of memory in the heap from becoming executable if it could have been written in the past. (Depends on WXORX)

OTHER Prevents any other page of memory from becoming executable if it could have been written in the past. (Depends on WXORX)

MPROTECT Enables WXORX, STACK, HEAP and OTHER

MMAP Prevents new executable mmap after the dynamic libraries have been loaded. (Depends on OTHER)

FULL Enables MPROTECT and MMAP.

EMUTRAMP Enables trampoline emulation, if trampoline emulation is missing, it's changed to whatever is set in "wxprot_emutramp_missing_default". (Depends on MPROTECT and conflicts with any other EMUTRAMP*)

EMUTRAMP_OR_MPROTECT Like EMUTRAMP but, if trampoline emulation is missing, it's changed to MPROTECT. (Depends on MPROTECT and conflicts with any other EMUTRAMP*)

EMUTRAMP_OR_NONE Like EMUTRAMP but, if trampoline emulation is missing, all the flags are replaced with NONE. (Depends on MPROTECT and conflicts with any other EMUTRAMP*)

VERBOSE Verbosely report every violation. (Depends on WXORX)

COMPLAIN Don't actually block anything. If VERBOSE is enabled too S.A.R.A will reports violations. (Depends on WXORX)

TRANSFER Child tasks will inherit this task's flags despite what is written in the configuration.

NONE Disables everything.

5.2.3 Examples

Enable full reporting, without enforcement, to any executable under /bin/:

```
/bin/* FULL,COMPLAIN,VERBOSE
```

Enable MPROTECT with verbose reporting on everything:

```
* MPROTECT,VERBOSE
```

5.2.4 See also

sara(7), *saractl(8)*, *sara-xattr(8)* and <https://sara.smeso.it>

5.3 sara-xattr manual page

5.3.1 Synopsys

sara-xattr [*options*] <*command*> <*filename*> [*flags*]

5.3.2 Description

`sara-xattr` is the userspace utility that manages S.A.R.A. LSM's extended attributes. The optional *flags* field is used only with the *set* command. Flags format is the same of *`wxprot.conf(5)`*'s config files.

5.3.3 Commands

set Set specified xattr on a file.

get get xattr from a file

del delete xattr from a file

5.3.4 Options

-h, --help	show this help message and exit
-v, --verbose	Be verbose. For extra verbosity use multiple -v.
-q, --quiet	Suppress any output.
-V, --version	show program's version number and exit
-c CONFIG_DIR, --config-dir CONFIG_DIR	Specify config directory. Defaults to "/etc/sara"
-S SECURITYFS, --securityfs SECURITYFS	The mount point of the securityfs. Defaults to "/sys/kernel/security".
-s {wxprot}, –submodule {wxprot}	Select the submodule you want to work with. Available submodules: wxprot.
-u, --user	Use user xattr namespace instead of the security namespace.
-b, --both	Use both user and security xattr namespace.

5.3.5 Examples

Turn off WX Protections for an executable:

```
sara-xattr -s wxprot set /usr/bin/program none
```

Turn on all WX Protections for an executable:

```
sara-xattr -s wxprot set /usr/bin/program full,verbose
```

5.3.6 Bugs

If you find any bugs, please report them at <https://github.com/smeso/saractl/issues>

5.3.7 See also

sara(7), *wxprot.conf(5)*, *saractl(8)*, *xattr(7)* and <https://sara.smeso.it>

5.4 sara-test manual page

5.4.1 Synopsys

sara-test

5.4.2 Description

sara-test is a regression test tool for S.A.R.A. When S.A.R.A. is disabled the output should look like this:

```
These tests should pass even with SARA disabled:
              wx_mappings:      OK
             nx_shellcode:      OK
     fake_trampoline_heap:      OK
 gcc_trampolines_working1:      OK
 gcc_trampolines_working2:      OK
        shm_mode_mprotect:      OK

These tests should pass with SARA fully enabled:
             anon_mmap_wx:      VULNERABLE
             file_mmap_wx:      VULNERABLE
     gnu_executable_stack:      VULNERABLE
            heap_mprotect:      VULNERABLE
           stack_mprotect:      VULNERABLE
       anon_mmap_mprotect:      VULNERABLE
       file_mmap_mprotect:      VULNERABLE
                 shm_wxorx:      VULNERABLE
      shm_permissive_mode:      VULNERABLE
         shm_mode_change1:      VULNERABLE
         shm_mode_change2:      VULNERABLE
            text_mprotect:      VULNERABLE
             bss_mprotect:      VULNERABLE
            data_mprotect:      VULNERABLE
                mmap_exec:      VULNERABLE
           proc_mem_write:      VULNERABLE
                 transfer:      ERROR
         fake_trampolines:      ERROR

Tests for procattr interface:
         correct_settings:      VULNERABLE
procattr tests disabled
```

When S.A.R.A. is enabled the output should look like this:

```
These tests should pass even with SARA disabled:
              wx_mappings:      OK
             nx_shellcode:      OK
     fake_trampoline_heap:      OK
 gcc_trampolines_working1:      OK
 gcc_trampolines_working2:      OK
        shm_mode_mprotect:      OK

These tests should pass with SARA fully enabled:
             anon_mmap_wx:      OK
             file_mmap_wx:      OK
     gnu_executable_stack:      OK
            heap_mprotect:      OK
           stack_mprotect:      OK
       anon_mmap_mprotect:      OK
       file_mmap_mprotect:      OK
                shm_wxorx:      OK
      shm_permissive_mode:      OK
         shm_mode_change1:      OK
         shm_mode_change2:      OK
```

(continues on next page)

(continued from previous page)

```
            text_mprotect:      OK
             bss_mprotect:      OK
            data_mprotect:      OK
                mmap_exec:      OK
           proc_mem_write:      OK
                 transfer:      OK
         fake_trampolines:      OK

Tests for procattr interface:
         correct_settings:      OK
         verbosity_change:      OK
          complain_change:      OK
     full_change_no_force:      OK
               force_wxorx:      OK
```

Please note that, in case you are running on an architecture that lacks trampoline emulation, the results of the following tests will be `NOT AVAILABLE`:

- fake_trampoline_heap
- gcc_trampolines_working1
- gcc_trampolines_working2
- fake_trampolines

If you get any result different from what is listed above please open an issue at <https://github.com/smeso/sara-test/issues>.

5.4.3 Bugs

If you find any bugs, please report them at <https://github.com/smeso/sara-test/issues>

5.4.4 See also

sara(7), *wxprot.conf(5)*, *saractl(8)* and <https://sara.smeso.it>

5.5 sara.h manual page

5.5.1 Synopsys

#include <sara.h>

5.5.2 Description

libsara can be used to query or change S.A.R.A.'s WX Protections flags. Please remember that protection can't be softened via this interface and you will only be allowed to add or remove a flag if and only if that action won't make the thread less secure. The VERBOSE flag can't be changed and any attempt to do this will be silently ignored. Any other violation will result in an error. Please always check the return value of `set_wxprot_self_flags`, `add_wxprot_self_flags` and `rm_wxprot_self_flags`. A thread can only query its own flags, unless it has CAP_MAC_ADMIN. Also, it can only change its own flags, regardless of the capabilities it has. Be aware that, while the flags change will only affect the current thread, the changes forced by `SARA_FORCE_WXORX` will affect the whole process. For more information please see *sara(7)*.

The <sara.h> header shall define the following:

```
SARA_ERROR
SARA_HEAP
SARA_STACK
SARA_OTHER
SARA_WXORX
SARA_COMPLAIN
SARA_VERBOSE
SARA_MMAP
SARA_FORCE_WXORX
SARA_EMUTRAMP
SARA_TRANSFER
SARA_NONE
SARA_MPROTECT
SARA_FULL
```

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```c
int set_wxprot_self_flags(uint16_t flags);
int add_wxprot_self_flags(uint16_t flags);
int rm_wxprot_self_flags(uint16_t flags);
uint16_t get_wxprot_flags(pid_t pid);
uint16_t get_wxprot_self_flags(void);
int is_emutramp_active(void);
```

5.5.3 See also

sara(7), *wxprot.conf(5)*, *saractl(8)*, *sara-xattr(8)* and <https://sara.smeso.it>