Suricata - Feature #2931

Perform privdrop without libcap-ng support

04/11/2019 12:14 PM - Emmanuel Roullit

Status: New
Priority: Normal
Assignee: Emmanuel Roullit
Category: Target version: TBD
Effort: Label:
Difficulty:

Description

Some platforms might not have libcap-ng support which disables the "run-as" configuration option and leaves the user with no other options but to run suricata with elevated privileges.

We created a changeset which allows to drop privileges despite the lack of libcap-ng support.

It lets the main thread perform initialization with the needed elevated privileges and drops them, by using setresuid(2) and setresgid(2), right before the SuricataMainLoop() is entered.

There is a caveat, which has been documented in the changeset, RUNMODE_AFP_DEV requires elevated privileges to allow the packet acquisition threads to reopen an AF_PACKET socket.

To guard against this, suricata will not start and inform the user libcap-ng support is required when the following requirements are met:
- libcap-ng support is disabled
- uid or gid change requested
- RUNMODE_AFP_DEV main run mode detected

Does other runmodes such as RUNMODE_PFRING and RUNMODE_NAPATECH would require similar guard as well?

Related issues:

Related to Feature #276: Libcap support for dropping privileges

History

#1 - 04/11/2019 12:18 PM - Emmanuel Roullit
Github PR available at https://github.com/OISF/suricata/pull/3793

#2 - 04/11/2019 06:21 PM - Victor Julien
- Related to Feature #276: Libcap support for dropping privileges added

#3 - 04/11/2019 06:26 PM - Victor Julien
I think a cleaner option would be to disable the whole reconnect/reopen feature in af-packet when we know it can't work. We can determine this at start up, so we might have a message at initialization of af-packet.

#4 - 04/12/2019 08:01 AM - Eric Leblond
Agree with Victor on reconnect. It is really unlikely it happens on standard ethernet interface so we could disable it in this case.

But question, is there Linux distribution without libcap-ng available ?

#5 - 04/12/2019 08:17 AM - Victor Julien
Right now libcap-ng is not mandatory on Linux, so we need to think about how to handle the case where its not available. Perhaps we can simply make it mandatory for Linux.

#6 - 04/12/2019 08:19 AM - Emmanuel Roullit
I am not aware of a specific Linux distro having this situation. So test this case on my machine, I forced LIBCAP_NG to "no" like this:

diff --git a/configure.ac b/configure.ac
[31x786]@@ -1711,7 +1711,7 @@

 LDFLAGS="${LDFLAGS} -L${with_libcap_ng_libraries}" 
 fi

- AC_CHECK_HEADER(cap-ng.h,,LIBCAP_NG="no")
+ AC_CHECK_HEADER(cap-ng.h,LIBCAP_NG="no",LIBCAP_NG="no")
    if test "$LIBCAP_NG" != "no"; then
        LIBCAP_NG=""
        AC_CHECK_LIB(cap-ng,capng_clear,,LIBCAP_NG="no")

#7 - 04/16/2019 02:40 PM - Emmanuel Roullit
A v2 Pull request has been published: https://github.com/OISF/suricata/pull/3800

#8 - 09/26/2019 09:11 AM - Victor Julien
Out of curiosity: have you considered doing a libcap implementation or supporting libcap-ng on the platforms you care about?

#9 - 11/07/2019 11:02 PM - Jason Ish
Victor Julien wrote:

Out of curiosity: have you considered doing a libcap implementation or supporting libcap-ng on the platforms you care about?

I'm not sure if we'll seen libcap/libcap-ng on other platforms. OpenBSD has pledge support, so I wonder if thats good enough for them. Would be nice to know if FreeBSD has something as well that would be more preferred than this approach in the above PR.

It would be nice to have some resolution here. I'm not against a lowest common denominator approach that may be limited in what it can do.

For example, how far could we get without ever elevating privileges again? This is not something we'd have to worry about on Linux with libpcap-ng. The idea is to keep the lowest common denominator as simple as possible.