Check if/how we should use NetworkManager's new MAC address spoofing capabilities

<table>
<thead>
<tr>
<th>Status:</th>
<th>Confirmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Category:</td>
<td>Spoof MAC</td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
<tr>
<td>Feature Branch:</td>
<td>wip/feature/11293-network-manager-spoof-mac</td>
</tr>
<tr>
<td>Type of work:</td>
<td>Research</td>
</tr>
<tr>
<td>Blueprint:</td>
<td></td>
</tr>
<tr>
<td>Start date:</td>
<td>03/31/2016</td>
</tr>
<tr>
<td>Due date:</td>
<td></td>
</tr>
<tr>
<td>% Done:</td>
<td>10%</td>
</tr>
<tr>
<td>Estimated time:</td>
<td>0.00 hour</td>
</tr>
<tr>
<td>Spent time:</td>
<td>0.00 hour</td>
</tr>
<tr>
<td>Starter:</td>
<td></td>
</tr>
<tr>
<td>Affected tool:</td>
<td></td>
</tr>
</tbody>
</table>

**Description**

As https://blogs.gnome.org/lkundrak/2016/01/18/networkmanger-and-tracking-protection-in-wi-fi-networks/ sums up, NM 1.2 + wpasupplicant 2.3 allow (opt-in) to randomize MAC address used for scanning for Wi-Fi networks (that we've rejected in #7380), and also to spoof the MAC address used for connecting to all Wi-Fi networks (needs to be verified, the blog post is unclear; previous NM versions allowed to configure a spoofed MAC address per network). NM 1.4 improves this further (announce blog post, NEWS entry).

Does this work on all hardware? The code suggests that this might rely on per-device capabilities.

What shall we do about it?

**Related issues:**

- Related to Tails - Feature #6453: Protect against fingerprinting via active Wi-Fi networks probing added
- Related to Tails - Feature #11856: Update design doc: NetworkManager 1.4+ ran...
- Related to Tails - Bug #11931: MAC Spoofing is broken in Stretch
- Related to Tails - Feature #10491: Redesign the network configuration and sta...
- Related to Tails - Bug #17115: MAC spoofing can fail and disable a network ad...
- Related to Tails - Bug #17128: Investigate whether internal network adapters ...
- Blocked by Tails - Bug #10289: Tails based on Debian Stretch

**Associated revisions**

Revision 73e43275 - 11/16/2016 05:09 PM - intrigeri

Configure NetworkManager to not touch MAC addresses (refs: #11931).

Its default behaviour on Debian Stretch is to reset the MAC address to the permanent one, and we did not make up our mind yet wrt. replacing our custom MAC spoofing system with NM's own one (refs: #11293).

**History**

#1 - 03/31/2016 09:39 AM - intrigeri
- Related to Feature #6453: Protect against fingerprinting via active Wi-Fi networks probing added

#2 - 08/26/2016 11:07 AM - intrigeri
- Description updated

#3 - 08/26/2016 11:28 AM - intrigeri

git grep mac.address in NM's Git repo helped me find the relevant settings:

- ethernet.cloned-mac-address (and ethernet.generate-mac-address-mask)
- wifi.cloned-mac-address (and wifi.generate-mac-address-mask), that deprecates wifi.mac-address-randomization
• wifi.scan-rand-mac-address (and wifi.scan-generate-mac-address-mask), that defaults to "yes" in NM 1.4

To look closer:

• source: git grep cloned.mac.address
• doc: NetworkManager.conf(5)

We have automated tests for all relevant scenarios, so I'm tempted to disable our own MAC address randomization, build a test ISO with NM's MAC randomization enabled, and see how it fares.

#4 - 08/26/2016 11:51 AM - intrigeri

#5 - 08/26/2016 11:52 AM - intrigeri
- Assignee set to anonym
- QA Check set to Info Needed

#6 - 08/28/2016 03:31 AM - intrigeri

$ cat /etc/NetworkManager/conf.d/spoof-mac.conf
[connection]
eternet.cloned-mac-address=random
    wifi.cloned-mac-address=random

... seems to work. It only affects newly created connections though.

#7 - 08/28/2016 03:33 AM - intrigeri
- Target version deleted (Tails_3.0)

(The only thing we really need to do for 3.0 is checking that the new NetworkManager's default settings (i.e. spoof MAC for scanning Wi-Fi) do not break our own MAC spoofing system. If this breaks then our test suite will tell us, and I'll file a dedicated ticket about it, and then we'll see how we can / want to fix it... which can probably be done without changing our MAC spoofing system entirely => dropping target version.)

#8 - 08/28/2016 03:47 AM - intrigeri
- Blocked by Bug #10289: Tails based on Debian Stretch added

#9 - 09/28/2016 08:07 AM - anonym
- Assignee changed from anonym to intrigeri

intrigeri wrote:
We have automated tests for all relevant scenarios, so I'm tempted to disable our own MAC address randomization, build a test ISO with NM's MAC randomization enabled, and see how it fares.

Note that several of our tests (the failure modes) are depending on the MAC spoofing implementation, in our case that macchanger is used (in a particular way). But the other two scenarios should work.

As for this ticket, we at least want wifi.scan-rand-mac-address -- in our design page we list the lack of this in our current implementation as a limitation. Any way, I'd need more details of how that feature works vs the randomisation done by NM when connecting to a network before I feel like I can make a decision.

Any way, I don't want to put any effort on getting this into Tails <3.0, which I think you'll agree with.

---

#10 - 10/02/2016 11:37 AM - intrigeri
- Related to Feature #11856: Update design doc: NetworkManager 1.4+ randomizes the MAC address used for scanning for Wi-Fi networks added

#11 - 10/02/2016 11:39 AM - intrigeri
- Assignee changed from intrigeri to anonym

As for this ticket, we at least want wifi.scan-rand-mac-address -- in our design page we list the lack of this in our current implementation as a limitation.

Agreed. It's now the default in current Stretch, so all that we need to do on this side is double-checking that it works, and updating our design doc for 3.0 ⇒ created #11856 to track it.

  Any way, I'd need more details of how that feature works vs the randomisation done by NM when connecting to a network before I feel like I can make a decision.

journalctl -u NetworkManager | grep -w MAC on my system seems to indicate that NM sets a random MAC address for scanning, and before connecting to a network it sets it to whatever is configured (random, permanent, or per-network persistent). Does this answer your question?

  Any way, I don't want to put any effort on getting this into Tails <3.0, which I think you'll agree with.

ACK. I'd like to make a decision (and implement it) in time for 3.0 but it's not a blocker.
intrigeri wrote:

Any way, I'd need more details of how that feature works vs the randomisation done by NM when connecting to a network before I feel like I can make a decision.

`journalctl -u NetworkManager | grep -w MAC` on my system seems to indicate that NM sets a random MAC address for scanning, and before connecting to a network it sets it to whatever is configured (random, permanent, or per-network persistent). Does this answer your question?

Yes, that sounds promising, so let's make this happen!

Any way, I don't want to put any effort on getting this into Tails <3.0, which I think you'll agree with.

ACK. I'd like to make a decision (and implement it) in time for 3.0 but it's not a blocker.

Let's aim for this, then!
With this branch, MAC spoofing scenarios we run on Jenkins pass, i.e. the MAC is spoofed and no leak is detected. Of course, "MAC address spoofing is disabled" fails since the branch always randomizes the MAC address, regardless of what was chosen in the Greeter.

#16 - 01/02/2017 07:57 PM - intrigeri
- Assignee deleted (intrigeri)
- Target version deleted (Tails_3.0)

anonym wrote:

Note that several of our tests (the failure modes) are depending on the MAC spoofing implementation, in our case that macchanger is used (in a particular way).

My quest to find the code that changes the MAC address in NM started (almost) at nm_platform_link_set_address:

- called from _hw_addr_set (not sure what happens exactly if it fails, BTW; _hw_addr_set will log and return an error, but then?)
- defined in src/platform/nm-platform.c, calls link_set_address (src/platform/nm-linux-platform.c), which calls NLA_PUT from libnl3 and waits for its reply with do_change_link_result.

So, if we want to test (either quickly, right now, or in the test suite, forever) what happens in the failure modes, we need to monkeypatch one of these functions (LD_PRELOAD, anyone?), or to somehow block MAC changing at a lower level (kernel? QEMU? libvirt?). Suddenly it feels harder than I would hope, and seriously outside of my comfort zone.

So for now I'm dropping this from the 3.0 target. If skilled people, who are not busy with higher-prio 3.0 tasks already, want to give a hand, I'd be delighted. Otherwise it'll be postponed to whenever #10491 is a thing, if its implementation requires per-connection MAC spoofing settings.

#17 - 01/02/2017 07:57 PM - intrigeri
- Related to Feature #10491: Redesign the network configuration and startup added

#18 - 02/01/2017 10:45 PM - intrigeri
- Feature Branch changed from feature/11293-network-manager-spoof-mac to wip/feature/11293-network-manager-spoof-mac

#19 - 10/07/2019 07:36 AM - intrigeri
- Related to Bug #17115: MAC spoofing can fail and disable a network adapter when waking up from suspend added

#20 - 10/07/2019 07:36 AM - intrigeri
- Related to Bug #17128: Investigate whether internal network adapters consistently use their previously spoofed MAC address after resuming from suspend added

#21 - 10/07/2019 08:00 AM - intrigeri
Does this work on all hardware? The code suggests that this might rely on per-device capabilities.

I think that's a more important question, at this point, than if/how we could adjust our test suite for the change this ticket is about (which is where we got blocked last time).

The current implementation of MAC spoofing in Tails tries to randomize the MAC address of every network adapter. If that fails, it fails closed and lets the user know, so they can consciously decide to restart and disable MAC spoofing, or buy a network adapter that supports MAC spoofing, or something. The resulting UX is not great but at least we're trying our best to meet our design goals (which in many cases have become user expectations: given how often users ask about such things on XMPP, it's clear that some of them do care).

IIRC NetworkManager has a list of devices (or kernel modules, or something) that are known to not support MAC spoofing — to be verified. If that's indeed the case, what does it do with those devices, if MAC address randomization is enabled? Will it just let the use the permanent MAC address, i.e. fail open? Same question when NM fails to spoof a MAC address. And finally, if the current NM behavior is not OK, how hard would it be to have it optionally do what we want?

---

#22 - 10/07/2019 08:07 AM - intrigeri

intrigeri wrote:

IIRC NetworkManager has a list of devices (or kernel modules, or something) that are known to not support MAC spoofing — to be verified.

Confirmed: on my sid system, wpasupplicant ships a /usr/lib/NetworkManager/conf.d/no-mac-addr-change.conf config file, which disables MAC address randomization for a bunch of drivers (rtl8723bs, rtl8189es, r8188eu, 8188eu, eagle_sdio, wl).