Tails - Bug #15460
Test suite broken with Java 9+ so we need to replace Sikuli

03/27/2018 02:33 PM - intrigeri

Status: Resolved
Priority: High
Assignee: anonym
Category: Test suite
Target version: Tails_4.5
Feature Branch: test/15460-replace-sikuli-force-all-tests, https://salsa.debian.org/tails-team/tails/merge_requests/43
Type of work: Code
Blueprint:

Description
Sikuli is not in Buster/testing/sid anymore, because of at least:

- https://bugs.debian.org/874146

OTOH, as mentioned by anonym on #15953#note-10: "It's worth noting that we don't use much of Sikuli. As long as we have an image matching primitive that returns the coordinates of the match, implementing the various find(), wait() etc we need is pretty easy. And something like xdotool (preferably something that will work in Wayland... if that is even possible?) can do the rest (mouse and keyboard interaction)".

And indeed, for example OpenQA supports Wayland, so even if we stick to our current testing framework, we could possibly steal some ideas and code from there wrt. the image matching primitives and interaction.

Subtasks:

<table>
<thead>
<tr>
<th>Bug</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>#17293</td>
<td>Install python-opencv and imagemagick on isotesters</td>
<td>Resolved</td>
</tr>
</tbody>
</table>

Related issues:

<table>
<thead>
<tr>
<th>Related to</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tails</td>
<td>Test suite produces obsoletion warnings with R...</td>
<td>Resolved</td>
</tr>
<tr>
<td>Tails</td>
<td>Test suite has defunct processes after sikuli ...</td>
<td>Resolved</td>
</tr>
<tr>
<td>Tails</td>
<td>Make our test suite survive changes in the surround...</td>
<td>Resolved 09/14/2018</td>
</tr>
<tr>
<td>Tails</td>
<td>Core work: Foundations Team</td>
<td>Confirmed</td>
</tr>
<tr>
<td>Tails</td>
<td>Uninstall libsikulixapi-java on isotesters</td>
<td>Confirmed</td>
</tr>
<tr>
<td>Tails</td>
<td>Refresh our patches to apply on top of thunder...</td>
<td>Resolved</td>
</tr>
<tr>
<td>Tails</td>
<td>Add Buster support to the automated test suite</td>
<td>Resolved</td>
</tr>
</tbody>
</table>

Associated revisions

Revision 8dad6f60 - 12/03/2019 02:30 PM - anonym
Test suite: add OpenCV module (to replace Sikuli).

Basically it is a wrapper around OpenCV's matchTemplate() which can be used for image matching, just like Sikuli's find() etc. Since Ruby doesn't have any (working) OpenCV bindings we resort to calling a Python script.

Currently it's not used, but stay tuned!

Refs: #15460
We are about to remove Sikuli, so we need an alternative.

Refs: #15460

Test suite: replace Sikuli with mix of xdotool and OpenCV.

Removed functionality: --retry-find --fuzzy-image-matching

Refs: #15460

Test suite: bump some images.

After bumping just these (objectively outdated) images I can run a large part of the test suite, indicating that OpenCV and Sikuli perform very similarly. Yay!

Refs: #15460

Test suite: add OpenCV module (to replace Sikuli).

Basically it is a wrapper around OpenCV's matchTemplate() which can be used for image matching, just like Sikuli's find() etc. Since Ruby doesn't have any (working) OpenCV bindings we resort to calling a Python script.

Currently it's not used, but stay tuned!

Refs: #15460

Test suite: add screenshot-method based on ImageMagick.

We are about to remove Sikuli, so we need an alternative.

Refs: #15460
Revision 15c71004 - 12/03/2019 08:50 PM - anonym
Test suite: replace Sikuli with mix of xdotool and OpenCV.
Removed functionality: --retry-find --fuzzy-image-matching
Refs: #15460

Revision 9d9549d3 - 12/03/2019 08:50 PM - anonym
Test suite: bump some images.
After bumping just these (objectively outdated) images I can run a
large part of the test suite, indicating that OpenCV and Sikuli
perform very similarly. Yay!
Refs: #15460

Revision 2ddc8b2c - 12/11/2019 03:57 PM - anonym
Test suite: add OpenCV module (to replace Sikuli).
Basically it is a wrapper around OpenCV's matchTemplate() which can be
used for image matching, just like Sikuli's find() etc. Since Ruby
doesn't have any (working) OpenCV bindings we resort to calling a
Python script.
Currently it's not used, but stay tuned!
Refs: #15460

Revision 50119f8a - 12/11/2019 03:57 PM - anonym
Test suite: add screenshot-method based on ImageMagick.
We are about to remove Sikuli, so we need an alternative.
Refs: #15460

Revision ecd5e1e5 - 12/11/2019 03:57 PM - anonym
Test suite: replace Sikuli with mix of xdotool and OpenCV.
Removed functionality: --retry-find --fuzzy-image-matching
Refs: #15460
Test suite: bump some images.

After bumping just these (objectively outdated) images I can run a large part of the test suite, indicating that OpenCV and Sikuli perform very similarly. Yay!

Refs: #15460

Test suite: add OpenCV module (to replace Sikuli).

Basically it is a wrapper around OpenCV's matchTemplate() which can be used for image matching, just like Sikuli's find() etc. Since Ruby doesn't have any (working) OpenCV bindings we resort to calling a Python script.

Currently it's not used, but stay tuned!

Refs: #15460

Test suite: add screenshot-method based on ImageMagick.

We are about to remove Sikuli, so we need an alternative.

Refs: #15460

Test suite: replace Sikuli with mix of xdotool and OpenCV.

Removed functionality: --retry-find --fuzzy-image-matching

Refs: #15460

Test suite: bump some images.

After bumping just these (objectively outdated) images I can run a large part of the test suite, indicating that OpenCV and Sikuli perform very similarly. Yay!

Refs: #15460
Revision bcf8196 - 01/15/2020 12:52 PM - anonym

Test suite: add OpenCV module (to replace Sikuli).

Basically it is a wrapper around OpenCV's matchTemplate() which can be used for image matching, just like Sikuli's find() etc. Since Ruby doesn't have any (working) OpenCV bindings we resort to calling a Python script.

Currently it's not used, but stay tuned!

Refs: #15460

---

Revision 75bab5db - 01/15/2020 12:52 PM - anonym

Test suite: add screenshot-method based on ImageMagick.

We are about to remove Sikuli, so we need an alternative.

Refs: #15460

---

Revision 91674552 - 01/15/2020 08:52 PM - anonym

Test suite: replace Sikuli with mix of xdotool and OpenCV.

Sikuli is broken with Java >= 9 and its Debian maintainer has given up. So we need something else.

Here we implement something that is almost a drop-in-replacement for the Sikuli features we use (although the interface for keyboard interaction has changed, resulting in tons of small changes under features/step_declarations in this commit). It uses OpenCV for (fuzzy) image matching, and xdotool for keyboard/mouse interaction. We didn't bother with most of the Sikuli class hierarchy, settling with just equivalents of Screen and Match -- we drop Pattern since we don't do OCR but just image matching; we drop Region because so far we have always operated on the whole screen (if we care about a subsection we can just use findAll() and check which matches fulfills the constraints).

This commit removes some functionality: both --retry-find and --fuzzy-image-matching are gone. These were mainly used for "image bumping", and I do have plans for an improved feature replacing these.

Refs: #15460
Revision 9715518c - 01/15/2020 08:52 PM - anonym
Test suite: bump some images.
After bumping just these (objectively outdated) images I can run a
large part of the test suite, indicating that OpenCV and Sikuli
perform very similarly. Yay!
Refs: #15460

Revision 70c77a4a - 01/16/2020 10:36 AM - anonym
Test suite: add --image-bumping-mode option.
This is a superior solution to the --retry-find option that were lost
when migrating from Sikuli to OpenCV.
Refs: #15460

Revision 73ee63cc - 01/16/2020 10:36 AM - anonym
Test suite: add automatic image bumping.
This is an improved version of the --fuzzy-image-matching option that
were lost when migrating from Sikuli to OpenCV.
Refs: #15460

Revision 6f826654 - 03/22/2020 06:18 AM - intrigeri
Merge branch 'test/15460-replace-sikuli-force-all-tests' into devel (refs: #15460)

History
#1 - 03/29/2018 08:31 AM - anonym
- Status changed from Confirmed to In Progress
- % Done changed from 0 to 10

intrigeri wrote:

At least:
- [https://bugs.debian.org/874146](https://bugs.debian.org/874146)

So this one we have a workaround for (copy-paster from the Debian bug):

```bash
mkdir -p /usr/lib/jvm/jre/lib/amd64/server/
ln -s /usr/lib/jvm/java-9-openjdk-amd64/lib/server/libjvm.so /usr/lib/jvm/jre/lib/amd64/server/libjvm.so
```
This one requires packaging SikuliX 1.1.2 for Debian, so using Java 9 is still out of the question...

... so let's just Java 8 for now:

```bash
# When Java 8 is dropped from unstable, set to "stable" instead
DIST=unstable
sudo apt install openjdk-8-j{re,dk}{,-headless}/{DIST}
sudo update-java-alternatives -s java-1.8.0-openjdk-amd64
```

---

**#3 - 04/02/2018 10:06 AM - intrigeri**

Actually, regarding rjb at least: this could be a great opportunity for you (anonym) to get involved a bit in Debian: you're one of the few people who understand anything about this Java+Ruby thing and it might be more pleasurable for you to submit a fix than to find someone else to do it.

**#4 - 04/13/2018 11:36 AM - intrigeri**

- Blocks Feature #13241: Core work: Test suite maintenance added

**#5 - 04/13/2018 11:36 AM - intrigeri**

- Assignee deleted (anonym)

**#6 - 04/13/2018 03:31 PM - intrigeri**

- Assignee set to lamby
- Target version changed from Tails_3.7 to Tails_3.8

Wrt. rjb Lunar won't mind help and will not have time to work on this any time soon => NMU / team upload welcome.

**#7 - 04/13/2018 03:33 PM - intrigeri**

- Estimated time set to 4.00 h

**#8 - 04/15/2018 07:36 PM - lamby**

- Subject changed from Test suite is broken with Java 9 to ruby-rjb test suite broken with Java 9

I've fixed the FTBFS and testsuite failures here:

https://bugs.debian.org/874146#28
What are the next steps here?

#9 - 04/19/2018 10:13 AM - lamby

lamby wrote:

I've fixed the FTBFS and testsuite failures here:

https://bugs.debian.org/874146#28

What are the next steps here?

Should I be pushing this fix in Debian for an easier "merge" or shall I upload to Tails? :)

#10 - 04/22/2018 06:49 AM - lamby

Just let me know :)

#11 - 04/27/2018 12:55 PM - intrigeri

I've fixed the FTBFS and testsuite failures here:

\o/

What are the next steps here?

Announce your intent to NMU, wait a bit (I'll ask Lunar if he's fine with it) and then upload :)

#12 - 04/27/2018 12:56 PM - intrigeri

Should I be pushing this fix in Debian for an easier "merge" or shall I upload to Tails? :)

05/06/2020
This package is only required for Tails developers who run the test suite on their sid system, so uploading to Debian makes much more sense here than uploading to a Tails-only repo.

#13 - 04/27/2018 12:56 PM - intrigeri

Just let me know :)

Pro tip: when asking someone's input here, assign to them and set "QA Check" to "Info Needed" :)

#14 - 04/27/2018 02:10 PM - lamby

Announce your intent to NMU

Thanks. Done here: https://bugs.debian.org/874146#35

Pro tip: when asking someone's input here, assign to them and set "QA Check" to "Info Needed" :)

Thanks! I guess I didn't know whom to assign to really, but I should have assigned to someone at the very least! :p

#15 - 04/27/2018 02:53 PM - intrigeri

I guess I didn't know whom to assign to really, but I should have assigned to someone at the very least! :p

Generally, for Foundations Team work, unless someone else (most likely anonym) is explicitly your team-mate/mentor, assigning to me is best.

#16 - 04/29/2018 07:06 PM - lamby
ruby-rjb 1.5.5-2 uploaded that fixes this :) 

#17 - 04/30/2018 08:25 AM - intrigeri
- Subject changed from ruby-rjb test suite broken with Java 9 to Test suite broken with Java 9
- Status changed from 11 to In Progress
- % Done changed from 10 to 20
- Type of work changed from Code to Debian

Thanks lamby!

So half of the problem (the part you committed to work on) is presumably fixed.

Now, two things.

First, I'm a bit confused. With Java 9 and this updated ruby-rjb (up-to-date sid) I still see the RuntimeError: Constants DL and Fiddle is not defined.

```bash
$ ./run_test_suite --view --iso ~/ftp/iso/tails/tails-amd64-3.6.2/tails-amd64-3.6.2.iso
Virtual X framebuffer started on display :1
VNC server running on: localhost:5900
Constants DL and Fiddle is not defined. (RuntimeError)
/usr/lib/ruby/vendor_ruby/rjb.rb:79:in `load'
/usr/lib/ruby/vendor_ruby/rjb.rb:79:in `load'
/srv/tails/git/features/support/helpers/sikuli_helper.rb:5:in `<top (required)>'
/usr/lib/ruby/vendor_ruby/cucumber/rb_support/rb_language.rb:96:in `load'
/usr/lib/ruby/vendor_ruby/cucumber/runtime/support_code.rb:142:in `load_file'
/usr/lib/ruby/vendor_ruby/cucumber/runtime/support_code.rb:84:in `block in load_files!'
/usr/lib/ruby/vendor_ruby/cucumber/runtime/support_code.rb:83:in `each'
/usr/lib/ruby/vendor_ruby/cucumber/runtime/support_code.rb:83:in `load_files!'
/usr/lib/ruby/vendor_ruby/cucumber/runtime.rb:253:in `load_step_definitions'
/usr/lib/ruby/vendor_ruby/cucumber/runtime.rb:61:in `run!'
/usr/lib/ruby/vendor_ruby/cucumber/cli/main.rb:32:in `execute!'
/bin/cucumber:??:in `<main>'
```

And indeed, the autopkgtests fail in just the same way: [https://ci.debian.net/data/autopkgtest/testing/amd64/r/ruby-rjb/225136/log.gz](https://ci.debian.net/data/autopkgtest/testing/amd64/r/ruby-rjb/225136/log.gz). So I wonder what's going on: it seems that lamby's patch fixed the problem for the package build but not at runtime.

But if I export JAVA_HOME=/usr/lib/jvm/java-9-openjdk-amd64 then rjb does load (confirmed by adding a few puts) but then SikuliX aborts the process:

```bash
./run_test_suite --view --iso ~/ftp/iso/tails/tails-amd64-3.6.2/tails-amd64-3.6.2.iso
Virtual X framebuffer started on display :1
VNC server running on: localhost:5900
[error] RunTimeINIT: *** terminating: Java version must be 1.7 or later!
```

... which is the symptom of the second problem I'm summing up below.

So it looks like rjb now needs JAVA_HOME to be set, otherwise it fails to load; OK, why not, we can do this (actually we used to). I suspect the autopkgtests should do the same.

Secondly, the remaining problem is that SikuliX 1.1.1 does not support Java 9. Apparently SikuliX 1.1.2 should support it: the [Changelog version 1.1.2](https://ci.debian.net/data/autopkgtest/testing/amd64/r/ruby-rjb/225136/log.gz) from March 10th 2018 reads "most JAVA 9 problems are fixed" => I've reported [https://bugs.debian.org/897215](https://bugs.debian.org/897215). Next step here is to test if SikuliX 1.1.2 fixes this problem and if it does, get it into Debian... which might be non-trivial given the process seems quite involved and upstream has switched to another Git repo again. Let's give the maintainers a couple weeks to answer but if they don't I'll check with lamby whether this bonus task fit into the 4h budget (and if not, how much more is needed).

In passing, I've tried anonym's workaround ([#15460#note-1](https://ci.debian.net/data/autopkgtest/testing/amd64/r/ruby-rjb/225136/log.gz)) + export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64 + sudo ln -s /usr/lib/jvm/java-8-openjdk-amd64/jre/lib/amd64/server /usr/lib/jvm/java-8-openjdk-amd64/jre/lib/server (to cope with the change lamby uploaded), and I see the Constants DL and Fiddle is not defined. (RuntimeError) error. So it looks like our workaround is not working anymore.
Since I fixed the FTBFS, the toolchain has changed so the build is now failing:

https://gist.github.com/lamby/9bd4d19faeaad9eb17ae9f98a0eb5bd0f/raw (attached as ruby-rjb_1.5.5-2_amd64.build)

I just had a quick poke (looks like an OpenJDK9 vs OpenJDK10 issue wrt http://openjdk.java.net/jeps/313) but I'm going to be hitting my 4h on this issue alas. Go longer? :)

(I've filed the FTBFS as https://bugs.debian.org/897664)

Can you please fix the ruby-rjb FTBFS in Debian (and submit a PR upstream)? I guess that 2 hours should be enough to replace the single call to javah. BTW, older build logs might be useful: there's been warnings in there for a while telling us that javah was deprecated and suggesting how to replace it :)

Sure thing!

- Fixed in Debian, uploaded as 1.5.5-2
- Forwarded upstream here: https://github.com/arton/rjb/pull/63
- Also forwarded https://github.com/arton/rjb/pull/62

I meant 1.5.5-3 here, sorry
Great! So we're good wrt. ruby-rjb. Chris, please tell me how much of the 6 hours you've used and I'll validate your work in our accounting :)

The next steps (SikuliX) are documented on #15460#note-17. Let's wait a couple more weeks (until our next FT meeting) to give the maintainers the chance to address https://bugs.debian.org/897215 and then we'll see.

---

Please tell me how much of the 6 hours you've used and I'll validate your work in our accounting :)

Any objection if we do that in one "batch" with the other tickets to keep our overheads low? :)

OK, let's try!

---

intrigeri wrote:

The next steps (SikuliX) are documented on #15460#note-17. Let's wait a couple more weeks (until our next FT meeting) to give the maintainers the chance to address https://bugs.debian.org/897215 and then we'll see.

I have:

- Updated the severity of https://bugs.debian.org/897215 to serious.
- Asked the Java maintainers whether they would like me to update it.
Noting RM bug #897333.

WIP SikuliX 1.1.2 packaging (broken at runtime according to the former maintainer): https://salsa.debian.org/java-team/sikuli

- Priority changed from Normal to Elevated

- Assignee changed from intrigeri to lamby
- QA Check set to Info Needed

Dear Chris,

sorry I forgot to raise this topic at the end of our FT meeting (while I had mentioned it earlier). I'd like us to assess the SikuliX situation so we can make a decision for Tails (and maybe for Debian). To start with I'd like to have an idea of:

- How far is the current WIP packaging of 1.1.2 from working on testing/sid (according to pini@d.o it "is completely broken at runtime") in a X.Org environment? Maybe 1.1.3 fixes some of that?
- What would the minimal maintenance cost of SikuliX in Debian would look like? I guess that importing upstream 1.1.3 could give an idea of the mess pini@d.o is talking of.
- What's the deal with Wayland? Note that Tails does not use it yet: #12213. pini@d.o says that he has "tested version 1.1.1 currently in unstable and testing and it is broken as well, probably because of the migration to Wayland". But I'm running GNOME on Wayland on sid and SikuliX 1.1.1 worked just fine to run the Tails test suite before Java 9 became the default; I suspect that pini@d.o mixed up "SikuliX 1.1.1 does not support Java 9" with "Sikuli is broken with Wayland" or similar. But I've not tested Sikuli to exercise an OS that runs Wayland so I dunno, it may very well be that pini@d.o is correct! It would be interesting to know if we're using any part of Sikuli that rely on X.Org and will break on Wayland. I'm not even sure how exactly we wire Sikuli with the VM under test.

Upstream publishes no Git tag so I think I start to understand what Gilles is talking of…

Once we have this info it'll be easier to decide whether we take over maintenance of SikuliX in Debian, or we start maintaining it for Tails only (in case the previous option is too costly/crazy and we can live with a cheap crappy version for Tails), or we research alternatives to Sikuli.

Are you interested to look into this? Do you think you can answer these questions in 8 hours? (I would say just try, report back shortly before running out of time, and how much progress you'll have made in 8h will already teach us something.) IIRC you did not want more work in June, so: can you do this in July? (Rationale: I'd like us to make a decision in time for the Buster freeze :)

ACK all these difficult questions!
Are you interested to look into this? Do you think you can answer these questions in 8 hours?

As I'm sure you can understand at this point, I simply can't tell without jumping into packaging 1.1.3. :) Will indeed just try and report back if/when running out of time. I should find time for this in June.

My gut tells me we should look into alternatives given all these problems we know about.

(pinging bug back and forth for visibility)

#33 - 05/29/2018 05:50 PM - lamby
- Assignee changed from intrigeri to lamby

#34 - 05/29/2018 05:59 PM - intrigeri
- Estimated time changed from 6.00 h to 14.00 h
- QA Check deleted (Info Needed)

Great! (updating total budget => 6 + 8 = 14)

#35 - 06/26/2018 04:28 PM - intrigeri
- Target version changed from Tails_3.8 to Tails_3.9

#36 - 06/27/2018 05:15 PM - lamby
During summit we roughly scheduled this as being targetted for Dec/Jan 2018/2019.

#37 - 08/30/2018 04:49 AM - intrigeri
- Target version changed from Tails_3.9 to Tails_3.12

lamby wrote:

During summit we roughly scheduled this as being targetted for Dec/Jan 2018/2019.

Right, updating target version accordingly ;)
Note that Debian will likely be shipping buster with OpenJDK 11.

Priority changed from Elevated to High
- Target version changed from Tails_3.14 to 2019

lamby wrote:

https://salsa.debian.org/java-team/sikuli/commit/9d14622bb62e7b16e13d55672f8a308820d08c3d

lamby wrote:
My gut tells me we should look into alternatives given all these problems we know about.

Agreed, let’s put the “make Sikuli work in Debian” on the back burner for now, and instead look into alternatives. I’ve collected some ideas in the ticket description.

#54 - 10/02/2019 11:32 AM - anonym
- File opencv-match.py added

A few days ago I stumbled upon some (Python) code that used OpenCV’s matchTemplate() which suites perfectly as an “image matching primitive”. You get coordinates and a similarity score for the match, so we can still match with different fuzziness. It only finds one match, though, so we cannot do something like Sikuli’s findAll() but we are currently not using any of them (although we have before), but I don’t think this is a blocker.

Inspired by that code I quickly wrote the attached script for a quick check on how it performs for our images (if you try it, setting the DEBUG env var will show the match in a popup which is handy): I booted Tails, took a screenshot of the desktop and could successfully match the expected images e.g. TorStatusUsable.png and GnomeApplicationsMenu.png. A nice start! For more evaluation I suggest integrating something like this script into our automated test suite so it replicates each match Sikuli attempts and saves the matches (the original image with a red square for the match) so they can be manually inspected easily after the full run is completed.

So, that would let us know if OpenCV’s matching can be a drop-in replacement for Sikuli’s. Unfortunately, however, there’s only an unmaintained Ruby gem for OpenCV (and it only supports exactly OpenCV version 2.4.12 which isn’t even in Debian). But now that we have no reason for a Java-to-Ruby bridge, why not introduce a Python-to-Ruby bridge? :D Of course it exists but it is also unmaintained, so I mention this mostly to immediately rule out this approach.

So if we want to use OpenCV I suppose we need to call it as a subprocess. In fact, the script as-is would serve for that purpose. A potential issue is that the script has two heavy dependencies, cv2 and numpy, which take almost 300ms to import on my pretty powerful CPU (i7-7700HQ @ 2.8 GHz) which I worry is so much that it might interfere. So perhaps we make it a “server” that we leave running and communicate with over e.g. a unix socket. It will work, but it is a bit ugly...

#55 - 10/02/2019 05:03 PM - intrigeri

A few days ago I stumbled upon some (Python) code that used OpenCV’s matchTemplate() which suites perfectly as an “image matching primitive”.

Great news, I’m glad you’ve been looking into this. It’s exciting!

So if we want to use OpenCV I suppose we need to call it as a subprocess. In fact, the script as-is would serve for that purpose. A potential issue is that the script has two heavy dependencies, cv2 and numpy, which take almost 300ms to import on my pretty powerful CPU (i7-7700HQ @ 2.8 GHz) which I worry is so much that it might interfere. So perhaps we make it a “server” that we leave running and communicate with over e.g. a unix socket. It will work, but it is a bit ugly...
Interesting. I think that at this point, I'm less worried about the potential ugliness than about reliability: it's one more thing that manages its own state and can get stuck or desynchronized from the callers. The remote shell historical fragility (now thankfully a thing of the past!) comes to mind here.

Wrt. strategy and allocation of our resources, with my "person who manages the FT's budget" hat on: I'm a bit concerned about the costs that come with a NIH solution. I would like us to first look into how OpenQA does images matching, as suggested in the ticket description; and later, fallback on this brand new approach, solve the problems that come with it, and maintain it, if and only if the code that a number of major distros use does not work for us. For example, I suspect that OpenQA's implementation does not affect any of this load time issue. Now, of course, it may be that we quickly realize that the OpenQA implementation is very much non-reusable and we give up on it after spending 1-2h on it; but at this point we don't know this.

---

intrigeri wrote:

A few days ago I stumbled upon some (Python) code that used OpenCV's matchTemplate() which suites perfectly as an "image matching primitive".

Great news, I'm glad you've been looking into this. It's exciting!

So if we want to use OpenCV I suppose we need to call it as a subprocess. In fact, the script as-is would serve for that purpose. A potential issue is that the script has two heavy dependencies, cv2 and numpy, which take almost 300ms to import on my pretty powerful CPU (i7-7700HQ @ 2.8 GHz) which I worry is so much that it might interfere. So perhaps we make it a "server" that we leave running and communicate with over e.g. a unix socket. It will work, but it is a bit ugly...

Interesting. I think that at this point, I'm less worried about the potential ugliness than about reliability:

Ack. Let's consider this a braindump and forget it for now!

Wrt. strategy and allocation of our resources, with my "person who manages the FT's budget" hat on: I'm a bit concerned about the costs that come with a NIH solution.

Yes, given the history with Sikuli (it was the reason for us initially using JRuby...) this is also my first priority. The thing is, I have (in a very relaxed manner) looked for Sikuli alternatives for years, and OpenCV has been the basically the only answer. Luckily it is a stable project in terms of maintenance, AFAICT! :)

Also, I suspect that the amount of work needed for integrating any matching primitive into our test suite is dwarfed by the work needed for the machinery to around that is shared by all such solutions, e.g. keyboard/mouse stuff, and probably rewriting minimal versions of a few of Sikuli's classes, like Screen and Match, since we are pretty invested in them and they are pretty great abstractions any way. In other words, the NIH-part is small in comparison to the whole work. And it will be compartmentalized so it's little work replacing it with another matching primitive.

I would like us to first look into how OpenQA does images matching, as suggested in the ticket description; and later, fallback on this brand new approach, solve the problems that come with it, and maintain it, if and only if the code that a number of major distros use does not work for us. For example, I suspect that OpenQA's implementation does not affect this load time issue. Now, of course, it may be that we quickly realize that the OpenQA implementation is very much non-reusable and we give up on it after spending 1-2h on it; but at this point we don't know this.
OpenQA also employs a straightforward application of OpenCV's matchTemplate(). :) At least we can look for inspiration in their choice of parameters.

I spent another 20 minute searching for image matching alternatives in Ruby-land, and while I can find some things for computer vision in general, none of them provide what we want (fuzzy image matching). I suspect that if more time is needed on this search, anything that comes up will be pretty obscure and hence likely worse in the end.

BTW, everything we need for the OpenCV python solution is available in Debian (python3-opencv python3-numpy python3-pil) so all-in-all it's starting to looks like pretty ok to me.

#57 - 10/03/2019 05:09 AM - intrigeri

Wrt. strategy and allocation of our resources, with my "person who manages the FT's budget" hat on: I'm a bit concerned about the costs that come with a NIH solution.

Yes, given the history with Sikuli (it was the reason for us initially using JRuby...) this is also my first priority.

Glad we're on the same page :)

In other words, the NIH-part is small in comparison to the whole work.

Okay. Without having thought about it much, I'm inclined to blindly trust your assessment here.

OpenQA also employs a straightforward application of OpenCV's matchTemplate(). :) At least we can look for inspiration in their choice of parameters.

OK!

I spent another 20 minute searching for image matching alternatives in Ruby-land, and while I can find some things for computer vision in general, none of them provide what we want (fuzzy image matching). I suspect that if more time is needed on this search, anything that comes up will be pretty obscure and hence likely worse in the end.

It's not 100% obvious to me that fuzzy image matching should be an absolute requirement here.

On the one hand, I do see some benefits in it. For example, since you added this feature, we've used it for at least 2 batch-updates: `1f02ddea0211b4f0c0b59a689f9af67154b83, 39f7d4baed5ea5aad2db07e477c4307bc7ed4d70`. That's not nothing. OTOH, it's also not huge. IIRC we expected to get much more benefit out of this feature: you implemented it initially when we were playing with the idea of basing Tails on Debian testing, which would have caused much more churn in images, and in turn would have made fuzzy matching a huge time saver.

So, if there's a way to use Ruby, avoiding the complexity and potential fragility that come from bridging Ruby & Python somehow, at the cost of losing fuzzy matching, then IMO we should seriously consider this option.

Note, in passing, that a bug in our fuzzy matching feature (#17029) has made me, and possibly others, waste substantial amounts of time for 2 years, which greatly outweighed the time we saved thanks to batch-updates so far: I started suspecting something was fishy since ~1 year ago (`3ac908f6bc9a54ac325bb529af02c48f2a89b676ca29`) but only tracked down & fixed the problem recently. But that's mostly off-topic because we'll learn from it and ensure that our next implementation of fuzzy matching won't have such problems :)

BTW, everything we need for the OpenCV python solution is available in Debian (python3-opencv python3-numpy python3-pil) so all-in-all it's starting to looks like pretty ok to me.
It's great to have this option if we have to, or decide to, fallback to Python!

#58 - 10/03/2019 09:53 AM - anonym
intrigeri wrote:

I spent another 20 minute searching for image matching alternatives in Ruby-land, and while I can find some things for computer vision in general, none of them provide what we want (fuzzy image matching). I suspect that if more time is needed on this search, anything that comes up will be extremely obscure and hence likely worse in the end.

It's not 100% obvious to me that fuzzy image matching should be an absolute requirement here.

Sikuli's image matching is always fuzzy, and we use a default required similarity of 0.9 (sikuli_settings.MinSimilarity = 0.9), that is what I'm talking about. The test suite's --fuzzy-image-matching means that we will try with a couple lower similarity values (i.e. "more fuzz") to get a candidate.

So at the moment we do not know if a pixel-by-pixel perfect matcher is good enough because we have never done that but I guess we could bump the default required similarity to 1.0 and do a full run to get an idea. My intuition is that it will be bad, and I fear it could introduce new fun problems like image compression changing pixels ⇒ false negatives.

That said, I haven't seen any image matcher in Ruby-land except OpenCV. But I guess it would be pretty easy to implement (convert to bitmaps, slide the candidate picture around until all pixels match (success), or all possibilities are exhausted (failure)) and maintain ourselves, so that is not out of the question.

On the one hand, I do see some benefits in it. For example, since you added this feature, we've used it for at least 2 batch-updates: 1692cbaf211b44f0f7cfbdf59a689ff9a67164abb, 331a7c4ba9f6a5ac3db07ea477f64307bd7de4d70. That's not nothing. OTOH, it's also not huge.

If we expected to get much more benefit out of this feature: you implemented it initially when we were playing with the idea of basing Tails on Debian testing, which would have caused much more churn in images, and in turn would have made fuzzy matching a huge time saver.

To be clear, I would happily sacrifice --fuzzy-image-matching for a native Ruby solution!

So, if there's a way to use Ruby, avoiding the complexity and potential fragility that come from bridging Ruby & Python somehow, at the cost of losing fuzzy matching, then IMO we should seriously consider this option.

Let's seriously consider implementing our own simple pixel-by-pixel equality matcher then!

Note, in passing, that a bug in our fuzzy matching feature (#17029) has made me, and possibly others, waste substantial amounts of time for 2 years, which greatly outweighed the time we saved thanks to batch-updates so far: I started suspecting something was fishy since ~1 year ago (3ac90f2ed9a544ce235bf7e520aa6c3a50208c81, 670bd5f6fcb4b301599e2cf81a4fee89b67dca2a) but only tracked down & fixed the problem recently.

Eek. Your theory that the default similarity is forgotten in favor of the last explicitly used one could be true. The hack around the comment "Due to bugs in rjb we cannot re-throw Java exceptions" also looks suspicious. Any way...

But that's mostly off-topic because we'll learn from it and ensure that our next implementation of fuzzy matching won't have such problems :)

Indeed. It's worth noting that it is because rjb does an imperfect job at integrating Java into Ruby that implementing the hooks needed for --retry-find and --fuzzy-image-matching have been this error prone. So many hacks... it will be great to get rid of all that mess!
... and I've just noticed that the very part of our test suite I'm working on today does not behave as intended, precisely because Sikuli matches an image which is not the one we're waiting for. So on my branch for #17056 I've (locally) trying to bump SIKULI_MIN_SIMILARITY to 0.95 to avoid this problem, which might actually fix the painful bug I'm after.

Well, I'm getting sick of Sikuli (boom bada boom tchak, then the rest of the "Sick of Sikuli" song ;)

- With SIKULI_MIN_SIMILARITY = 0.99, Sikuli still matches an image that is not (exactly) on screen AFAICT
- With SIKULI_MIN_SIMILARITY = 1.0, Sikuli does not match an image that is exactly on screen AFAICT (updated it from a failure screenshot I got immediately before)

This suggests that either pixel-perfect image matching won't work for us, or that Sikuli's matching algorithm significantly differs from why my eyes and brain match.

At this point, I'm not sure we can actually check if pixel-perfect image matching can work for us merely by setting SIKULI_MIN_SIMILARITY = 1.0, because I'm losing the faith I had in the fact it would produce the result I understand it should produce :/

---

#60 - 10/08/2019 01:19 PM - intrigeri

Hi @anonym,

(I've sent this comment 5 days ago but just received the rejection message from Redmine, which disliked the facepalm emoji I had included.)

anonym wrote:

intrigeri wrote:

It's not 100% obvious to me that fuzzy image matching should be an absolute requirement here.

Sikuli's image matching is always fuzzy, and and we use a default required similarity of 0.9 (sikuli_settings.MinSimilarity = 0.9), that is what I'm talking about.

Oops, I had forgotten this.
Looks like my misunderstanding started an interesting discussion and will make us consider another option, so it hopefully it was not only wasted time ;)

So at the moment we do not know if a pixel-by-pixel perfect matcher is good enough because we have never done that

Absolutely.

but I guess we could bump the default required similarity to 1.0 and do a full run to get an idea.
I'd be very curious to see the results of such an experiment. Some of our images may be identical across test suite runs but still currently match only thanks to the current 0.9 number; so we might need to update them to be pixel-perfect; and then the 2nd run and following will give us the answer we want here.

and I fear it could introduce new fun problems like image compression changing pixels ⇒ false negatives.

Just curious: where is image compression involved?

To be clear, I would happily sacrifice --fuzzy-image-matching for a native Ruby solution!

OK, case closed then.

So, if there's a way to use Ruby, avoiding the complexity and potential fragility that come from bridging Ruby & Python somehow, at the cost of losing fuzzy matching, then IMO we should seriously consider this option.

Let's seriously consider implementing our own simple pixel-by-pixel equality matcher then!

If I understand correctly that the choice is indeed "a shiny new pile of hacks to bridge Python & Ruby, so that we can use OpenCV" vs. "our own simple pixel-by-pixel equality matcher in Ruby", then yes, I'm leaning towards the latter. Of course, we first need to validate the hypothesis that pixel-perfect matches can work for us: as you said, at this point we don't know.
intrigeri wrote:

This suggests that either pixel-perfect image matching won't work for us, or that Sikuli's matching algorithm significantly differs from why my eyes and brain match.
At this point, I'm not sure we can actually check if pixel-perfect image matching can work for us merely by setting SIKULI_MIN_SIMILARITY = 1.0, because I'm losing the faith I had in the fact it would produce the result I understand it should produce :/

Yeah, this was a bad idea from me, sorry! Related: OpenCV's matchTemplate() fails with 1.0, so it can only be fuzzy.

Let's seriously consider implementing our own simple pixel-by-pixel equality matcher then!

If I understand correctly that the choice is indeed "a shiny new pile of hacks to bridge Python & Ruby, so that we can use OpenCV" vs. "our own simple pixel-by-pixel equality matcher in Ruby", then yes, I'm leaning towards the latter. Of course, we first need to validate the hypothesis that pixel-perfect matches can work for us: as you said, at this point we don't know.

I propose this: we put in the work of the machinery needed around the image matching primitive (i.e. remove Sikuli, re-implement some basic versions of classes like Screen, Match, including methods like wait(), findAny() etc (our primitive only gives us the equivalent of exists(), something like @findfailed_hook() etc). Since we want to move away from Sikuli it seems like this is work we must do any way. Once we have all that we have done most of the work (according to my estimation, at least), so it makes sense to do the extra work to evaluate both OpenCV's matchTemplate() and a homegrown pixel-perfect matcher.

Thoughts?

(Also, assigning to me now that we seem to have given up on Sikuli and lamby won't work on this.)
I propose this: we put in the work of the machinery needed around the image matching primitive (i.e. remove Sikuli, re-implement some basic versions of classes like Screen, Match, including methods like wait(), findAny() etc (our primitive only gives us the equivalent of exists(), something like @findfailed_hook() etc). Since we want to move away from Sikuli it seems like this is work we must do any way. Once we have all that we have done most of the work (according to my estimation, at least), so it makes sense to do the extra work to evaluate both OpenCV's matchTemplate() and a homegrown pixel-perfect matcher.

Thoughts?

Sounds good to me.

(Also, assigning to me now that we seem to have given up on Sikuli and lamby won't work on this.)

This sounds fair enough to me, given lamby clearly expressed he'd rather not to work directly on our test suite framework, and the current proposal will happen there.

@lamby, I'm sure we can find more exciting Tails work for you than this :)

---

#63 - 12/05/2019 08:06 PM - intrigeri
- Blocks Bug #17308: Uninstall libsikulixapi-java on isotesters added

#64 - 12/16/2019 12:20 PM - anonym
- Target version changed from 2019 to Tails_4.3
- Feature Branch set to test/15460-replace-sikuli (and +force-all-tests)

#65 - 01/31/2020 10:01 AM - anonym
- Subject changed from Test suite broken with Java 9+ to Test suite broken with Java 9+ so we need to replace Sikuli

#66 - 01/31/2020 11:35 AM - anonym
As of fcc34ce this branch is starting to look pretty good!

Currently there are three branches running this on Jenkins:

- test/15460-replace-sikuli obviously
- test/15460-replace-sikuli-force-all-tests which does not run all tests due to my mistake of not using a +. But it's good to get even more results with @fragile, so whatever. :)  
- test/15460-replace-sikuli-foo-force-all-tests to actually test everything.

Let's go through the failed scenarios for all the runs that skip @fragile tests:

05/06/2020
Installing Tails with GNOME Disks from a USB image: failed due to lack of space.
The system partition is updated when booting from a USB drive where a Tails USB image was copied: failed because it depends on the previous scenario's success.
The shipped Tails OpenPGP keys are up-to-date: failing for legit reasons.
Downloading files with the Tor Browser: This one is very curious! Looking at the failure screenshot reveals that the wrong home page is loaded -- we expect "/home/testing/" but get just the "/home/" version intended for stable releases. This would make it a bug in Tails, not the test suite, but OTOH I never see this issue in the stable branch's test runs. WTF?

The shipped Tails OpenPGP keys are up-to-date: same as above.
The Tor Browser directory is usable: another instance of the wrong home page as in the "Downloading files with the Tor Browser" run above.

These Additional Software scenarios fails all the time on branches, so not an issue for this branch:
- I set up Additional Software when installing a package without persistent partition and the package is installed next time I start Tails
- My Additional Software list is configurable through a GUI or through notifications when I install or remove packages with APT or Synaptic
- Recovering in offline mode after Additional Software previously failed to upgrade and then succeed to upgrade when online
- I am notified when Additional Software fails to install a package
The shipped Tails OpenPGP keys are up-to-date: same as above.
The Tor Browser should not have any plugins enabled: yet another instance of wrong page home.

The same four Additional Software scenarios again.
The shipped Tails OpenPGP keys are up-to-date: same as above
The following scenarios have unrelated "The system is not fully running yet" failures because console-setup.service failed to load. These scenarios restore from the same snapshot, usb-install-with-persistence-logged-in, so I guess some random error relating to console-setup.service happened before the snapshot was taken, thus poisoning the scenarios that restore from it and later run the Tor is ready step (which contains the systemd unit sanity check, which is a bit ugly IMHO, but well).
- Upgrading an initial Tails installation with an incremental upgrade
- Upgrading a Tails that has several SquashFS deltas present with an incremental upgrade
- Upgrading a Tails whose signing key is outdated
- Writing files to a read/write-enabled persistent partition
- Creating and using a persistent NetworkManager connection
- Using a persistent Pidgin configuration

The shipped Tails OpenPGP keys are up-to-date: same as above.

The same four Additional Software scenarios again.
The shipped Tails OpenPGP keys are up-to-date: same as above.
Only the expected add-ons are installed: Looking at the video it seems that the Ad-ons manager lags a bit when loading, so the "Extensions" tab that we want to press is moved in the time between the image match and the click, so we end up clicking the "Recommendations" tab → failure. It could also be that the OpenCV approach is faster than Sikuli. :) Any way, this should be an easy fix.

Conclusion and next steps
IMHO the test results finally look pretty great! It has been difficult and tedious (reading logs, logs, logs...) work and has taken a lot of time (~45 h in total), because I also did work towards making the test suite run onbuster/current sid, and had to deal with really subtle issues like 7fe2123 and afc1638 (this latter commit I have hopes that it will reduce lost key presses (and thus stability) over Sikuli!). Also 70c777d and 73ee63c might have been premature investments, but I'm sure it will pay off in the future (when it would take 2x as much time to implement these compared to now, when all this in my hot cache :)).

So I think the next steps are:
- Fix the WTF with the Tor Browser loads the startup page step, where the wrong home page is loaded in Tor Browser like we were testing a stable release. (Filed as #17462)
- Fix the simple fragility issue in Only the expected add-ons are installed.
- Update the docs. Probably look for more occurrences of "sikuli" in both code and code comments and update them.
- Also look at the force-all-tests runs. I need more results from fec34eb (or later) before I can do that, but from my initial look at the results so far I can tell that it looks promising!
- Get an initial review of this monster branch (33 files changed, 801 insertions, 561 deletions). @intrigeri, I think you are the most suitable for this task. What do you think?
Here's some more analysis for non-@fragile runs:

From now on I will ignore the following:

- The same four Additional Software scenarios again.
- The shipped Tails OpenPGP keys are up-to-date: same as above.
- Failures due to the Tor Browser opening the wrong home page.

**test_Tails_ISO_test-15460-replace-sikuli run 61**

Only unrelated failures.

**test_Tails_ISO_test-15460-replace-sikuli run 62**

- Writing a Tails iso hybrid to a USB drive and booting it, then installing Tails on top of it using Tails Installer, and it still boots: The failure is that Xorg fails to start or similar; there's just a black screen when we are waiting for Tails Greeter. Unrelated.

**test_Tails_ISO_test-15460-replace-sikuli run 63**

Only unrelated failures.

**test_Tails_ISO_test-15460-replace-sikuli-force-all-tests run 61**

- These fail for the same (unrelated) reasons as they failed in test_Tails_ISO_test-15460-replace-sikuli run 61 above:
  - Installing Tails with GNOME Disks from a USB image
  - The system partition is updated when booting from a USB drive where a Tails USB image was copied

**test_Tails_ISO_test-15460-replace-sikuli-force-all-tests run 62**

Only unrelated failures.

**test_Tails_ISO_test-15460-replace-sikuli-force-all-tests run 63**

Only unrelated failures.

**Conclusion**

No problems at all with this branch in these six runs, yay!
Here’s the analysis of the full runs:


I’m ignoring the same scenarios as I did before. Here I’m simplifying a bit by lumping all runs together; we list each scenario only once, together with how many times it failed (which runs are listed in [square brackets]).

- **Syncing OpenPGP keys using Seahorse should work and be done over Tor.:** 6 [15, 16, 17, 18, 19, 20]
  - Unrelated, because this is broken in Tails.
- **Syncing OpenPGP keys using Seahorse started from the OpenPGP Applet should work and be done over Tor.:** 6 [15, 16, 17, 18, 19, 20]
  - Same as the previous scenario.
- **VirtualBox guest modules are available:** 2 [18, 19]
  - Unrelated. It’s some odd virt-viewer bug that I never have seen (here’s the video if you are interested).
- **check all PO files:** 1 [15]
  - Unrelated, since this check is unaffected by this branch. FWIW it looks like a i18nsppector bug.
- **The Additional Software dpkg hook notices when persistence is locked down while installing a package:** 1 [17]
  - Unrelated. I looked carefully at the video + debug.log an concluded that all mouse/keyboard/screen interaction (that this branch changes) succeeds; it’s some command run via the Remote Shell that fails, for whatever reason.
- **Symmetric encryption and decryption using OpenPGP Applet:** 1 [18]
  - This is quite similar to the failure in the **Only the expected add-ons are installed** scenario for Thunderbird, above. Our OpenCV-based image matching is so fast that when clicking OK in the pin entry we manage to take the screenshot before the pin entry prompt has disappeared, leading to error.

**Conclusion**

So the full runs look great too. Yay!

**Todo:**

- Make pin entry handling robust.
Fix the WTF with the Tor Browser loads the startup page step, where the wrong home page is loaded in Tor Browser like we were testing a stable release.

If this is not specific to this branch, then I'd rather track it on a dedicated ticket.

Update the docs. Probably look for more occurrences of "sikuli" in both code and code comments and update them.

ACK.

Get an initial review of this monster branch (33 files changed, 801 insertions, 561 deletions). @intrigeri, I think you are the most suitable for this task. What do you think?

Sure. What kind of timeline do you have in mind?

---

#70 - 02/05/2020 10:19 AM - anonym

intrigeri wrote:

- Fix the WTF with the Tor Browser loads the startup page step, where the wrong home page is loaded in Tor Browser like we were testing a stable release.

If this is not specific to this branch, then I'd rather track it on a dedicated ticket.

Agreed. I was initially unable to find this failure on any other branch so I wasn't ready to rule this branch out as the culprit, however, after doing some grep@ing on @jenkins.lizard I found other examples (e.g. this stable run has three instances).

Filed as #17462.

- Get an initial review of this monster branch (33 files changed, 801 insertions, 561 deletions). @intrigeri, I think you are the most suitable for this task. What do you think?

Sure. What kind of timeline do you have in mind?
There certainly is no emergency, so take it after 4.3 when things are slower. It would be cool if we could get this into 4.4!

#71 - 02/11/2020 03:25 PM - anonym
- Target version changed from Tails_4.3 to Tails_4.4

#72 - 02/13/2020 01:14 PM - anonym
- Status changed from In Progress to Needs Validation
- Assignee changed from anonym to intrigeri
- Type of work changed from Research to Code

I have now fixed everything I could think of. Enjoy the review! :)

#73 - 02/16/2020 05:26 PM - intrigeri
- Blocked by Bug #17481: Refresh our patches to apply on top of thunderbird 1.68.5.0-1-deb10u1 added

#74 - 02/16/2020 05:36 PM - intrigeri
- Feature Branch changed from test/15460-replace-sikuli (and +force-all-tests) to test/15460-replace-sikuli, test/15460-replace-sikuli-foo+force-all-tests

(Fix feature branch value, after reading the comment that made me understood I was looking at the wrong thing.)

#75 - 02/16/2020 05:59 PM - intrigeri
- test/15460-replace-sikuli-force-all-tests which does not run all tests due to my mistake of not using a +.

IMO that's not a mistake of yours, but rather a UX bug, so I've just fixed it so the system behaves like you would understandably expect it to: https://git.tails.boum.org/puppet-tails/commit/?id=189b63bc87e0901dc423061340ae1f09c4db068d :) \n
#76 - 02/16/2020 06:09 PM - intrigeri
- Feature Branch changed from test/15460-replace-sikuli, test/15460-replace-sikuli-foo+force-all-tests to test/15460-replace-sikuli-force-all-tests, https://salsa.debian.org/tails-team/tails/merge_requests/43

I have now fixed everything I could think of. Enjoy the review! :) 

Ooh yeah!

For the most part, I'm very satisfied by the last few runs on Jenkins. The only concern I have is:

- On our other branches, IIRC every time some Additional Software scenarios fail, it's always the same set of 4 scenarios.
On this branch, sometimes only 1 scenario fails, at the "I update APT using Synaptic" step.

This makes me wonder if the case when only the "I update APT using Synaptic" step fails is a regression, i.e. a new failure mode, different from the case when the infamous 4 scenarios fail. Did you already take a closer look? Or did you classify this under the broader "Additional Software scenarios are fragile" category?

I did not check and I can be remembering it wrongly, but IIRC the case when 4 scenarios fail is a different failure mode.

This is a pretty large branch ("39 files changed, 886 insertions(+), 631 deletions(-)") so I'd rather do the code review on GitLab:
https://salsa.debian.org/tails-team/tails/merge_requests/43

The only thing that changes for you is the place where we'll discuss the diff. If you have any follow-up commits to push, push them as usual to the canonical tails.git, and our infra will automatically propagate them to GitLab (with weird authorship of push actions but oh well, it'll be solved in a couple months once we use our own GitLab).

I'll use the test/15460-replace-sikuli-force-all-tests branch for that merge request (MR), so a far as GitLab is concerned, that's the one that matters. See you there :)

---

#77 - 02/16/2020 07:43 PM - intrigeri
- Status changed from Needs Validation to In Progress
- Assignee changed from intrigeri to anonym

I'm done with the code review. Spoiler alert: mostly code style comments from me. Great work!

I'm running this on my local Jenkins and will report back if anything goes wrong.

I did not run this on my own sid system yet but I intend to do so during the next (and likely final) round of review.

---

#78 - 02/17/2020 08:13 AM - intrigeri
- File 00_16_51_Using_obfs4_pluggable_transports.mkv added
- File 01_42_30_My_Additional_Software_list_is_configurable_through_a_GUI_or_through_notifications_when_I_install_or_remove_packages_with_APT_or_Synaptic.mkv added

Here is a summary of 2 full test suite runs on my the fastest workers available in my local Jenkins; they're probably the fastest CPUs (Intel Xeon E-2134) on which this code was ever run so far, which may trigger different behavior wrt. racy scenarios and gives us an opportunity to identify more potential problems before we merge this branch :) 

Erroneous typed text

I've seen "Using obfs4 pluggable transports" fail once because incorrect chars were typed into the Tor Launcher bridges input text area. We tried to type this:

```
00:16:10.685829350: Keyboard: typing: obfs4 10.2.1.1:9938 6B746184B92C974DE27909B82214959F920FCD cert=nn1+KR TpfDfvng4GvHjD6Tl2UHRhx5t.suhrx6.jCb+FHNbXHzUuaz2Ynw64AFYwSs9yGEg lat-mode=0
00:16:19.681299207: Keyboard: pressing: Return
```

05/06/2020
... but in the attached video you'll see that instead of "RP?+Fjz", we typed "RP?+(FJZ", i.e. it's as if the shift key got stuck pressed and not released after typing "P".

I don't remember seeing this with Sikuli but I did not check.

**I update APT using Synaptic**

The "I update APT using Synaptic" step failed in "My Additional Software list is configurable through a GUI or [...]" in each of my two runs. It looks like a false positive to me: Tails did what we want but the test suite failed to notice it. I suspect the new implementation does something faster or slower which affects our chances to win/lose a race.

I wonder if it's the same problem in the test suite runs on lizard that have only 1 Additional Software scenario failing, which I reported as a possible sign of a regression yesterday.

I don't remember seeing this with Sikuli but I did not check.

Here's the debug log (stripping some of the 10 identical retries) and I'm attaching the corresponding video:

```
01:16:45.604254999: Screen: found GnomeActivitiesOverview.png at (357, 68)
01:16:45.604456714: Keyboard: typing: S
01:16:47.666015875: Keyboard: typing: ynpatic Package Manager
01:16:51.072207074: Keyboard: pressing: ctrl+Return
01:16:51.073392347: Screen: waiting for PolicyKitAuthPrompt.png
01:16:51.977461477: Screen: found PolicyKitAuthPrompt.png at (320, 274)
01:16:51.977660034: Keyboard: typing: asdf
01:16:52.221698315: Keyboard: pressing: Return
01:16:52.221855987: Screen: waiting for PolicyKitAuthPrompt.png to vanish
01:16:52.607428326: Screen: found PolicyKitAuthPrompt.png at (320, 274)
01:16:54.608783479: Screen: PolicyKitAuthPrompt.png has vanished
01:16:54.608966939: executing Python as amnesia:
import dogtail.config
import dogtail.tree
import dogtail.predicate
import dogtail.rawinput
dogtail.config.logDebugToFile = False
dogtail.config.logDebugToStdOut = False
dogtail.config.blinkOnActions = True
dogtail.config.searchShowingOnly = True
node227 = dogtail.tree.root.application('synaptic')
01:16:54.776623678: execution complete
01:16:54.776782840: executing Python as amnesia: node228 = node227.child('Synaptic Package Manager', roleName='frame', recursive=False)
01:17:06.762882840: execution complete
01:17:06.762949501: retry_action: Tor operation failed with exception: Test::Unit::AssertionFailedError: <RuntimError> exception expected but was <Dogtail::Failure>{<The Dogtail script raised: SearchError: child of [application | synaptic]: child with roleName='dialog'>}.
When I start Synaptic
```
try_for: attempt 1 (s elapsed of 10s)...
Remote shell: calling as root: pidof -x -o '%PPID' synaptic
Remote shell: call returned: [1, "", ""]
try_for: success!

Remote shell: calling as root: pidof -x -o '%PPID' synaptic
Remote shell: call returned: [0, "", ""]

Screen: waiting for GnomeApplicationsMenu.png
Screen: found GnomeApplicationsMenu.png at (60, 15)

Remote shell: calling as amnesia: xdotool key Super
Remote shell: call returned: [0, "", ""]

Screen: waiting for GnomeActivitiesOverview.png
Screen: found GnomeActivitiesOverview.png at (357, 68)

Keyboard: typing: S

Keyboard: typing: synaptic Package Manager
Keyboard: pressing: ctrl+Return

Screen: waiting for PolicyKitAuthPrompt.png
Screen: found PolicyKitAuthPrompt.png at (320, 274)
Keyboard: typing: asdf
Keyboard: pressing: Return
Screen: waiting for PolicyKitAuthPrompt.png to vanish
Screen: found PolicyKitAuthPrompt.png at (320, 274)

node231 = dogtail.tree.root.application('synaptic')

node231.button('Reload').click()
import dogtail.config
import dogtail.tree
import dogtail.predicate
import dogtail.rawinput
dogtail.config.logDebugToFile = False
dogtail.config.logDebugToStdOut = False
dogtail.config.blinkOnActions = True
dogtail.config.searchShowingOnly = True
node267 = dogtail.tree.root.application('synaptic')
node268 = node267.child('Synaptic Package Manager', roleName='frame', recursive=False)
node269 = node267.button('Reload')
node270 = node267.child(roleName='dialog', recursive=False)
And I update APT using Synaptic: # features/step_definitions/apt.rb:143

Tor operation failed (despite retrying 10 times) with
Test::Unit::AssertionFailedError: <RuntimeError> exception expected but was
<Dogtail::Failure(<The Dogtail script raised: SearchError: child of [application | synaptic]: child with roleNam
e='dialog'>)>. (MaxRetriesFailure)
 features/support/helpers/misc_helpers.rb:173:in `rescue in block in retry_action'
 features/support/helpers/misc_helpers.rb:156:in `block in retry_action'
 features/support/helpers/misc_helpers.rb:155:in `loop'
 features/support/helpers/misc_helpers.rb:155:in `retry_action'
 features/support/helpers/misc_helpers.rb:144:in `retry_tor'
 features/step_definitions/apt.rb:148:in `^I update APT using Synaptic$'
 features/additional_software_packages.feature:53:in `And I update APT using Synaptic'
Thanks so much for the review and tests! I'll answer to the other issues later, and for now focus on:

intrigeri wrote:

**Erroneous typed text**

I've seen "Using obfs4 pluggable transports" fail once because incorrect chars were typed into the Tor Launcher bridges input text area. We tried to type this:

```
[...]
```

... but in the attached video you'll see that instead of "RP/+fjz", we typed "RP?+(FJZ", i.e. it's as if the shift key got stuck pressed and not released after typing "P".

With my slower CPU I could reproduce this by dropping the sleep 0.060. I then investigated deep into the libvirt/qemu sources, and got the idea that a non-zero holdtime might be the problem, and it indeed seems to fix this. So c792956 fixes that (and a related thing). @intrigei, can you stress test this on the Xeon? I used this step to be able to do it with more focus:

```ruby
Then /^I stress test the keyboard$/ do
  s = 'obfs4 10.2.1.1:9940 FD092E44D9BB74A94DE6B5C82E01431975CE8133 cert=o+cnEnxPwPRIDs43MjWksm+2k983BaRF/+9fjzv9DocN5MgSJE7mL/VGpU2O1HTXQaTw iat-mode=0'
  loop do
    tmp = $vm.execute_successfully('mktemp', user: LIVE_USER).stdout.chomp
    step "I run "echo '#{s}' > #{tmp}" in GNOME Terminal"
    loop do
      break unless $vm.file_empty?(tmp)
    end
    assert_equal(s, $vm.file_content(tmp).chomp)
  end
  $vm.execute_successfully("rm #{tmp}")
end
```

I don't remember seeing this with Sikuli but I did not check.

I have definitely seen it (same scenario!) but couldn't find anything on Redmine. :S
So c792956 fixes that (and a related thing). @intrigei, can you stress test this on the Xeon? I used this step to be able to do it with more focus:

Please push that commit and I'll happily do so :)

Gah! Done!

Unfortunately, stress-testing failed:

```
12:08:51   Scenario: Stress testing keyboard input via libvirt   # features/test.feature:4
12:09:01   Given I have started Tails from DVD without network and logged in   # features/step_definitions/snapshots.rb:170
12:09:01   And I wait 10 seconds   # features/step_definitions/compon_steps.rb:863
12:09:01   Slept for 10 seconds
12:09:43   Then I stress test the keyboard   # features/step_definitions/compon_steps.rb:1096
12:09:43   <>obfs4 10.2.1.1:9940 FD092E44D9BB74A94DE6B5CB2E01431975CE8133 cert=o+cNenxPwPRIDs43MjWcksm+2k9
```

```
12:09:43   S8BaRP/+9fjzv9DooN5MqSJE7mL/VGpU20IHTJXQuaTtw iat-mode=0" > expected but was
12:09:43   <>obfs4 10.2.1.1:9940 FD092E44D(BB74A4E6B5CB2E01431975CE8133 cert=o+cNenxPwPRIDs43MjWcksm+2k9
```

```
12:09:43   BaRP/+9fjzv9DooN5MqSJE7mL/VGpU20IHTJXQuaTtw iat-mode=0"
```

```
12:09:43   diff:
12:09:43   - <>obfs4 10.2.1.1:9940 FD092E44D9BB74A94DE6B5CB2E01431975CE8133 cert=o+cNenxPwPRIDs43MjWcksm+2k9
```

So c792956 fixes that (and a related thing). @intrigei, can you stress test this on the Xeon? I used this step to be able to do it with more focus:
Should I try bumping the default opts[:holdtime] value?

I've run the stress-test step twice with opts[:holdtime] ||= 0.040 on the Xeon and in both cases, it has run just fine until the job run hit the 30 minutes inactivity timeout my local Jenkins uses. This suggests that indeed, a higher hold time makes a difference — yeah!

I've now added a puts in the loop so that Jenkins does not believe the job run is inactive ⇒ I can stress-test it more (easily).

Then, if 0.040 proves to be robust in this context, and for some reason we think that value is too high, I could bisect the [0.010, 0.040] interval to find out what's the minimal value we can use.
intrigeri wrote:

Should I try bumping the default opts[:holdtime] value?

I've run the stress-test step twice with opts[:holdtime] ||= 0.040 on the Xeon and in both cases, it has run just fine until the job run hit the 30 minutes inactivity timeout my local Jenkins uses. This suggests that indeed, a higher hold time makes a difference — yeah!

Awesome! \o/

I've now added a puts in the loop so that Jenkins does not believe the job run is inactive ⇒ I can stress-test it more (easily).

I'm curious, @intrigeri, did you see any failure?

Then, if 0.040 proves to be robust in this context, and for some reason we think that value is too high, I could bisect the [0.010, 0.040] interval to find out what's the minimal value we can use.

I know too little about this domain to dare analyzing this further. FWIW, I had a quick look at what my hold times seems to be with xev and 40ms looks realistic. So unless you uncovered problems with the additional stress testing I'm happy with this number. If so, feel free to commit that change!

#85 - 02/27/2020 10:41 AM - intrigeri

Hi,

sorry for the delay: as you'll see below, it was significantly more complicated for me than providing a simple yes/no answer.

I've run the stress-test step twice with opts[:holdtime] ||= 0.040 on the Xeon and in both cases, it has run just fine until the job run hit the 30 minutes inactivity timeout my local Jenkins uses. This suggests that indeed, a higher hold time makes a difference — yeah!

Awesome! \o/
Unfortunately, my "it has run just fine" was entirely wrong: I reached this conclusion because I saw no error in the console output, until Jenkins timed out. But after closer inspection, I've established that the test suite was stuck in the break unless $vm.file_empty?(tmp) loop: due to buggy key presses, the file was not created. I've now replaced that loop with a try_for that has a timeout, so I get easier to understand output from my next stress test runs.

I'm wondering if your own successful stress test results are affected by the same problem. In doubt, I recommend adding an info_log() to the loop, that will tell on the console output when a new iteration is started. If your own tests were indeed erroneously reported as successful, perhaps you could rerun them, figure out something that works for you, report back about your tests, and then I'll try myself to see if there's any difference?

I've now added a puts in the loop so that Jenkins does not believe the job run is inactive ⇒ I can stress-test it more (easily).

I'm curious, @intrigeri, did you see any failure?

All these combinations always fail here in the 2nd or 3rd iteration of the stress test:

- delay = 0.060, holdtime = 0.010
- delay = 0.060, holdtime = 0.040
- delay = 0.100, holdtime = 0.050

In all cases, shift gets stuck and some key presses are lost. The first set of tests is the one that behaves best. I guess it makes sense that increasing holdtime without increasing delay accordingly is bound to fail.

While waiting for your answer above & guidance, I'll run tests again, this time increasing delay (0.120) but with holdtime = 0.010.

I did not check if that's worse than before c792956cc4f5d98099fe04245c5e45df57cc2cd4 yet.
3rd iteration fails: shift gets stuck and some key presses are lost. I'll run tests with delay = 0.500, holdtime = 0.010. If that fails too, then I don't know what I could/should test next.

#87 - 02/27/2020 01:39 PM - intrigeri

While waiting for your answer above & guidance, I'll run tests again, this time increasing delay (0.120) but with holdtime = 0.010.

3rd iteration fails: shift gets stuck and some key presses are lost. I'll run tests with delay = 0.500, holdtime = 0.010.

Same problem, sadly.

If that fails too, then I don't know what I could/should test next.

That's where I'm at: I'm now officially lost ⇒ please advice :)

#88 - 03/04/2020 12:17 PM - anonym

@intrigeri wrote

delay = 0.500, holdtime = 0.010

With that kind of delay I'm tempted to think we're not experiencing some sort of timing issue as I originally thought, but rather some "discrete" bug where events are lost/reordered/mixed/whatever. Still, for me setting a holdtime (even 0.001) seemed like an improvement, which is weird, because holdtime = 0.010 happens to be QEMU's default (i.e. when you set holdtime = 0). I am curious what the results would be with delay = 0.500, holdtime = 0.250, just in case the problem is with a too short holdtime. Could you test this? Oh, and please include the assertion errors -- with enough of them I'm hoping to see some pattern.

Also, which version of QEMU are you using? If you are on 2.8 (from Stretch, which I guess you are) there's been quite a few changes to how key events are dealt with, so I'd like you to try 4.2. If it helps, you could use my stretch-backport which I have uploaded to my home on misc.lizard for your convenience.
I am curious what the results would be with delay = 0.500, holdtime = 0.250, just in case the problem is with a too short holdtime. Could you test this?

Will do, hopefully by the end of the week!

Also, which version of QEMU are you using? If you are on 2.8 (from Stretch, which I guess you are)

Yes, that's what I'm using (same as on lizard).

there's been quite a few changes to how key events are dealt with, so I'd like you to try 4.2. If it helps, you could use my stretch-backport which I have uploaded to my home on misc.lizard for your convenience.

Thanks! I'll try my best to do this by the end of the week too.

I am curious what the results would be with delay = 0.500, holdtime = 0.250, just in case the problem is with a too short holdtime.

I've run this 3 times. Every time it failed in the first iteration, with File '/tmp/tmp.POhiXkJKt' was not created (Timeout::Error) after the 20s timeout of the try_for loop I've set up. Each screenshot shows missing chars in the destination filename and the terminal cursor is still at the end of the command line, as if Return had not been pressed.
Hi,

I'd like you to try 4.2. If it helps, you could use my stretch-backport which I have uploaded to my home on misc.lizard for your convenience.

Thanks! I'll try my best to do this by the end of the week too.

Unfortunately, this looks like a backport for Buster (the version number suggested that but I tried anyway): at least qemu-system-x86_4.2-3+deb10_amd64.deb has a few versioned dependencies that are not available on Stretch, so I cannot test this.

#92 - 03/06/2020 04:47 PM - anonym
intrigeri wrote:

Hi,

I'd like you to try 4.2. If it helps, you could use my stretch-backport which I have uploaded to my home on misc.lizard for your convenience.

Thanks! I'll try my best to do this by the end of the week too.

Unfortunately, this looks like a backport for Buster (the version number suggested that but I tried anyway): at least qemu-system-x86_4.2-3+deb10_amd64.deb has a few versioned dependencies that are not available on Stretch, so I cannot test this.

Sorry, now I feel like a fool! :S Indeed, it's a buster-backport, sorry for wasting your time! :(  
I gave a shot backporting to stretch (just dropping deps that are missing in stretch) but I ended up with lots of unrelated issues around includes, and while I fixed a few, new ones just kept popping up so I gave up. So I guess this test is blocked by the upgrade of our isotesters to Buster.
hey,

anonym wrote:

intrigeri wrote:

Sorry, now I feel like a fool! :S Indeed, it's a buster-backport, sorry for wasting your time! :( 

No worries, it really did not take me that much time! :)

I gave a shot backporting to stretch (just dropping deps that are missing in stretch) but I ended up with lots of unrelated issues around includes, and while I fixed a few, new ones just kept popping up so I gave up.

OK, let's give up the idea of backporting QEMU 4.x to Stretch. I'm not surprised it can be quite tricky.

So I guess this test is blocked by the upgrade of our isotesters to Buster.

This worries me a little. It could be that I'm overestimating the risk×impact of the key press fragility issue, and it could be that it affects nobody so far in practice except my local Jenkins, in which case we can probably merge this branch once it has passed the rest of its review process, then I immediately upgrade my isotesters to Buster, and if the problem is still there, we'll need to do something about it quickly.

But we have little data to tell how large that risk×impact actually is, so this does not feel super confidence inspiring to me.

In order to gain more confidence into the hypothesis that this fragility problem is fixed by QEMU 4.x, I think the cheapest option we have is this:

1. backup the (Stretch) disk of one of my local super fast isotesters
2. upgrade that isotester to Buster
3. upgrade to QEMU 4.2 using your backport
4. stress-test key presses on this branch and report back here
5. backup the (Buster) disk of that isotester, so I can easily go back to step 4
6. revert that isotester to Stretch so it can run the test suite for our other branches

In theory it should be rather straightforward. I'm going to start this right now, let's see what unexpected difficulties I'll face in practice ;)

Meanwhile, on your side, you're not blocked by this test: there's still my review to process ;)

05/06/2020
Hi,

intrigeri wrote:

In order to gain more confidence into the hypothesis that this fragility problem is fixed by QEMU 4.x, I think the cheapest option we have is this:

1. backup the (Stretch) disk of one of my local super fast isotesters
2. upgrade that isotester to Buster
3. upgrade to QEMU 4.2 using your backport
4. stress-test key presses on this branch and report back here

Good news!

My isotester (Xeon) upgraded to Buster, with your QEMU 1:4.2-3+deb10 backport, has stress-tested keyboard input for 1 hour with no failure, using the settings you've put on the branch (delay = 0.060, holdtime = 0.010). Woohoo! :)

Given stuff broke super early in the same VM on Stretch with an older QEMU, I'm now confident that the keyboard input thing is not a problem... as long as all potentially impacted systems that run our test suite are upgraded to Buster + a QEMU 4.2 backport soon after this branch is merged. Given the systems and people impacted, this raises some interesting deployment timeline and coordination issues, which IMO we need to consider carefully — but perhaps high-latency async' communication is not the best way to have this conversation.

I've created an isotester-buster suite in our custom APT repo, where we can upload a QEMU 4.2+ backport for Buster. Whenever you want to prepare and upload it, please ensure you use a correct version number for backports (hint: dch --bpo), instead of one that's greater than the version in testing/sid.

Our packaging repo for QEMU is there:

```
git clone tails@git.tails.boum.org:qemu
```

---

FYI, while I had a Buster isotester ready to try stuff, I've run the full test suite on this branch. The only failure was the same I reported a few weeks ago:

**I update APT using Synaptic**

The "I update APT using Synaptic" step failed in "My Additional Software list is configurable through a GUI or [...]" in each of my two runs. It looks like a false positive to me: Tails did what we want but the test suite failed to notice it. I suspect the new implementation does something faster or slower which affects our chances to win/lose a race.
I wonder if it's the same problem in the test suite runs on lizard that have only 1 Additional Software scenario failing, which I reported as a possible sign of a regression yesterday.

#97 - 03/10/2020 01:50 PM - anonym
- Status changed from In Progress to Needs Validation
- Assignee changed from anonym to intrigeri

Thanks so much for your testing! Good news indeed!

intrimer wrote:

[...all potentially impacted systems that run our test suite [must be] upgraded to Buster + a QEMU 4.2 backport soon after this branch is merged. Given the systems and people impacted, this raises some interesting deployment timeline and coordination issues, which IMO we need to consider carefully — but perhaps high-latency async’ communication is not the best way to have this conversation.

Well, I'm not sure we have to coordinate this like you suggest: after all, this branch has been running without any (?) instances of messed up keyboard events on Jenkins and on my system (except when I ran the stress test) so it's essentially only relevant for you Xeon monster. What do you think?

FYI, while I had a Buster isotester ready to try stuff, I've run the full test suite on this branch. The only failure was the same I reported a few weeks ago:

I update APT using Synaptic

More good news that this was the only failure, because I found the fix! See this part of the log:

01:17:49.867714792: retry_action: Tor operation failed with exception: Test::Unit::AssertionFailedError: <RuntimeError> exception expected but was <Docktail::Failure(<The Dogtail script raised: SearchError: child of [application | synaptic]: child with roleName='dialog'>)>.

[...]

01:22:06.320960584: Forcing new Tor circuit...

In 3eca3b26ef95b186d8d99fc33ed41a0d031d4553 I moved from RuntimeError to Dogtail::Failure for Dogtail errors, so we were expecting the wrong exception, which now is fixed in c4b0a785807b78e5b04dc68a588050d29e47ac79.

I've also fixed all issues that you identified in the GitLab MR, except one which I think we should wait with.

So, except the deployment discussion, I think this is merge ready!
Hi!

anonym wrote:

intrigeri wrote:

[...] all potentially impacted systems that run our test suite [must be] upgraded to Buster + a QEMU 4.2 backport soon after this branch is merged. Given the systems and people impacted, this raises some interesting deployment timeline and coordination issues, which IMO we need to consider carefully — but perhaps high-latency async' communication is not the best way to have this conversation.

Well, I'm not sure we have to coordinate this like you suggest: after all, this branch has been running without any (?) instances of messed up keyboard events on Jenkins and on my system (except when I ran the stress test) so it's essentially only relevant for you Xeon monster. What do you think?

I hear what you're saying, so I tried to identify more precisely where my fear actually lies vs. where I'm confident enough to merge this without coordination with upgrades to Buster.

I'm fine with not coordinating this with any of:

- Jenkins isotesters' upgrade to Buster: we already have reassuring data on this front
- developers: AFAICT, with 1 exception, they all run Buster or newer already

My only significant concern is the impact on kibi: as part of his primary RM role, he often needs to re-run failed tests locally. I'd rather not see this situation happen: we merge this, and next time he attempts to re-run tests during a release process, we discover that his system is affected by this problem too.

I think that the odds of seeing this happen are very low. But it seems easy and cheap to gain more confidence upfront, at a more relaxed time: we could ask kibi to run the full test suite from this branch a few times overnight.

IMO, at the end of the day, what matters most here is not how you or I feel about it: it's how kibi feels about it. He explicitly requested to be in the loop ahead of merging, for big changes that could impact his work. I think this qualifies. He may answer "let's not bother, this looks unlikely enough to break and I know I can deal with it if needed", or "thanks for asking, indeed I'd rather try it before it's merged", or something else. But IMO we should ask him.

Does this make sense to you?

If it may help, I could sum up the problem to kibi and ask him myself.

---

I update APT using Synaptic

More good news that this was the only failure, because I found the fix!

Great!

I've also fixed all issues that you identified in the GitLab MR, except one which I think we should wait with.

Confirmed!

So, except the deployment discussion, I think this is merge ready!

I'm super happy we've reached that point!!!

I think 3 things are missing:
the deployment discussion (see above)
- a QEMU 4.2 backport for Buster in our custom APT repo (for snapshots support in nested environments + the keyboard input problem): we could discuss at length whose job it is to prepare a first version, but at the end of the day, based on what happened for QEMU/Stretch, chances are that I'm going to be the one who maintains it, so I can as well take care of this from day 1 → I'll do that unless you tell me you can, and want to, take responsibility for maintaining this backport until we stop supporting Buster as a platform to run our test suite
- me testing this on my sid system

Since most of it is on my plate, I'll keep this ticket assigned to me :)
My main worry would be to not properly monitor for when I have to prepare a new release. How would you do it? Is there some nice way to get notified in this precise situation (e.g. getting an email precisely when a security fix has been released for $PACKAGE in $DEBIAN_DISTRIBUTION)?

I get the impression that you prefer to do this from day 1 rather than risking to suddenly get a new responsibility on day N in case I fail to maintain the backport -- is this correct? If so I think it's probably best you take it from now.

- me testing this on my sid system

Since most of it is on my plate, I'll keep this ticket assigned to me :)

Ok!

---

#100 - 03/12/2020 10:19 AM - anonym

@CyrilBrulebois, I have a request! I'll try my best to summarize everything in this post so you can avoid the lengthy discussion on this ticket.

Could you please try to run the full test suite from this branch, test/15460-replace-sikuli-force-all-tests (ignore the other similarly named branches) on the exact setup you use when testing releases? The reason is that we want to make sure we don't break your setup, which is critical during release testing when you are the RM. So ideally you'd run it, say, three times so we get a few data points to reason about.

One issue we have seen is that with QEMU 2.8 (i.e. Stretch's, which I am assuming your test setup is using) there can be issues with keyboard input (lost key strokes; shift-modifier being applied for more key strokes than it should). But who knows, perhaps your system will exhibit yet another failure mode? So we feel it's safest to confirm it works well enough on your system before merging this.

---

#101 - 03/13/2020 06:35 AM - CyrilBrulebois

Hey @anonym; +1 on the all-in-one-post-summary. :)  

As a data point, I had a single test to run locally, so as long as the CI is running the tests successfully enough, I can probably live with glitches on my system until we move to buster or something; or start thinking about upgrading the system to buster. Or run tests on a nearby machine I would upgrade to buster.

I think the important part is mostly the CI (as in Jenkins setup); whatever is needed, I can accomodate (bonus points if you confirm that doesn't need to be my Tails-trusted-system).

---

#102 - 03/14/2020 06:12 PM - intrigeri

Hi anonym,
anonym wrote:

intrigeri wrote:

- a QEMU 4.2 backport for Buster in our custom APT repo (for snapshots support in nested environments + the keyboard input problem): we could discuss at length whose job it is to prepare a first version, but at the end of the day, based on what happened for QEMU/Stretch, chances are that I'm going to be the one who maintains it, so I can as well take care of this from day 1 → I'll do that unless you tell me you can, and want to, take responsibility for maintaining this backport until we stop supporting Buster as a platform to run our test suite.

My main worry would be to not properly monitor for when I have to prepare a new release. How would you do it?

So far, whenever there's a DSA against QEMU, either I do the package update immediately, or I create an issue on Redmine. I get the notification from https://lists.debian.org/debian-security-announce/.

And if we still support oldstable past its official security support, once it's become supported by the LTS group, then s/DSA/DLA/ and there's a dedicated mailing list for them.

Is there some nice way to get notified in this precise situation (e.g. getting an email precisely when a security fix has been released for $PACKAGE in $DEBIAN_DISTRIBUTION)?

I don't know. I've verified that https://tracker.debian.org/ has no specific option to track only security uploads.

I get the impression that you prefer to do this from day 1 rather than risking to suddenly get a new responsibility on day N in case I fail to maintain the backport -- is this correct?

You guessed correctly that I sometimes would say that, and I appreciate you took it into account. This being said, it's not the case here, because the amount of work is so small.

My preference for doing it from day 1 myself if I am going to maintain the thing rather comes from:

- If I'm going to maintain it, I prefer to pick my preferred workflow/style rather than have to adjust myself (or the packaging) to whatever you will have started with.
- If I'm going to maintain it, it's kind of a waste of effort that you learn to do it in a way that's good enough for our infra (for some undefined version of "good enough", which is surely part of the problem). If you're interested in learning (and then I'm very happy to mentor and review), IMO it's better to learn how to do the long-term thing, rather than a one-shot backport.

Is this info sufficient for you to decide whether you want to commit to this? If not, please ask :)

05/06/2020
This is not going to happen because x11vnc won't work on my system:

14/03/2020 19:42:41 passing arg to libvncserver: localhost
14/03/2020 19:42:41 x11vnc version: 0.9.16 lastmod: 2019-01-05 pid: 71747
14/03/2020 19:42:41 Wayland display server detected.
14/03/2020 19:42:41 Wayland sessions are as of now only supported via -rawfb and the bundled deskshot utility.
Exiting.'

I propose we don't block on this here (it's not a regression introduced by this branch) but instead:

- document it (GNOME on Wayland is the default desktop on Buster, so not exactly a corner case :)
- file a ticket about making our test suite run in a Wayland desktop

Makes sense?

Note that as part of trying this, I spotted a few problems in the doc → see the MR on Salsa.

I'm done with the bits I committed to do so far, so reassigning to anonym.

intrigeri wrote:

- file a ticket about making our test suite run in a Wayland desktop

I've mentioned it on #17457. We'll see if we need a dedicated ticket once we cross that bridge :)
intrigeri wrote:

I'll try running this on my system without --view.

It works! It took 265 minutes and I've seen only 1 failure (presumably caused by my unstable Internet connection; that scenario passed after retrying). Woohoo 😃

And regarding your additional test requests:

- I confirm that failure screenshots are correctly saved.
- I confirm that --capture-all correctly saved 1 video when I run 1 scenario.

I'll do the QEMU backport and maintain it.

Uploaded 4.2-3~bpo10+0.tails1 to our isotester-buster APT suite.
I quickly gave up preparing the backport from Vcs-Git (it does not build out-of-the-box and I was not willing to deal with submodules matters) so I did it from the sid source package. This implies I had to upload my backport's source package since it won't be in Git.

Apart of that, two things:

- I see dozens of zombie "[import] <defunct>" and "[python3] <defunct>" processes left behind while the test suite is running. They go away once the test suite run completes though. Maybe some process management is missing? I did not check how this affects memory usage.

I see some Ruby warnings on my up-to-date sid (Ruby 2.7):

```
features/support/helpers/vm_helper.rb:433: warning: Using the last argument as keyword parameters is deprecated; maybe ** should be added to the call
features/support/helpers/remote_shell.rb:79: warning: The called method 'initialize' is defined here
features/support/helpers/misc_helpers.rb:356: warning: constant ::Fixnum is deprecated
```
I guess these are not regressions brought by this branch, but the goal here is to support systems more recent than Stretch, so it could make sense to fix these while we're at it. Not a merge blocker IMO, as long as it's fixed soon enough to avoid warning fatigue :)

#111 - 03/21/2020 06:49 PM - intrigeri
- Assignee changed from anonym to intrigeri

Given the timing for the upcoming freeze vs. anonym's week-end, to ensure this goes in 4.5~rc1, I will:

1. merge devel (that now has #6560 & friends) into this branch, resolve conflicts
2. run the test suite locally on the resulting branch
3. merge into devel if happy
4. send this back to anonym to deal with the few remaining issues (either in time for the freeze or as a freeze exception before 4.5)

#112 - 03/22/2020 06:20 AM - intrigeri
- Assignee changed from intrigeri to anonym

intrigeri wrote:

Given the timing for the upcoming freeze vs. anonym's week-end, to ensure this goes in 4.5~rc1, I will:

1. merge devel (that now has #6560 & friends) into this branch, resolve conflicts
2. run the test suite locally on the resulting branch
3. merge into devel if happy
4. send this back to anonym to deal with the few remaining issues (either in time for the freeze or as a freeze exception before 4.5)

Done!

#113 - 03/23/2020 02:45 PM - anonym
- Status changed from In Progress to Needs Validation
- Assignee changed from anonym to intrigeri

intrigeri wrote:

I'll do the QEMU backport and maintain it.

anonym will:

- fix the doc issues mentioned on the MR
- document the Wayland limitation, so that nobody wastes time setting everything up before they discover it won't work as they want
- detect the Wayland situation and advise to not use --view (assuming it does indeed work without --view)

All this is now resolved in these commits that I pushed straight to devel:
950fa6ae37 Test suite: adapt setup instructions for #15460.
9592d3f8e3 Test suite: document that --view/--vnc-server-only requires X11.
99b0557ed0 run_test_suite: --view/--vnc-server-only are only supported on x11.
ca7d987d24 run_test_suite: don't print usage on error.

Please review!

#114 - 03/25/2020 09:13 AM - intrigeri
- Status changed from Needs Validation to In Progress
- Assignee changed from intrigeri to anonym

Hi,

LGTM, thanks!

On Bullseye and newer, why do we install python-jabberbot and python-potr from Stretch, as opposed to Buster? The latter would be more future-proof vs. the upcoming Stretch EOL.

Meanwhile, since we last met, I did the tests I had committed to, and reported on #15460#note-110. Could you please take a look at my findings and process them in whatever way you see fit?

#115 - 03/26/2020 08:42 AM - intrigeri
- Related to Bug #17552: Test suite produces obsoletion warnings with Ruby 2.7 added

#116 - 04/07/2020 05:05 PM - CyrilBrulebois
- Target version changed from Tails_4.5 to Tails_4.6

#117 - 04/12/2020 01:20 PM - intrigeri
- Related to Bug #17551: Test suite has defunct processes after sikuli removal added

#118 - 04/12/2020 01:21 PM - intrigeri
- Status changed from In Progress to Resolved
- Target version changed from Tails_4.6 to Tails_4.5

Files

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>ruby-rjb_1_5_5_2_amd64.build</td>
<td>18 KB</td>
<td>05/03/2018</td>
<td>lamby</td>
</tr>
<tr>
<td>opencv-match.py</td>
<td>1003 Bytes</td>
<td>10/02/2019</td>
<td>anonym</td>
</tr>
<tr>
<td>00_16_51_Using_obfs4_pluggable_transports.mkv</td>
<td>405 KB</td>
<td>02/17/2020</td>
<td>intrigeri</td>
</tr>
<tr>
<td>01_42_30_My_Additional_Software_list_is_configurable_through_a_GUI_or_through_notifications_when_i_install_or_remove_packages_with_APT_or_Synaptic.mkv</td>
<td>4.62 MB</td>
<td>02/17/2020</td>
<td>intrigeri</td>
</tr>
</tbody>
</table>