





☆ Starred by 4 users

Owner:

lukasza@chromium.org

CC:

tommycli@chromium.org
adetaylor@chromium.org
tgu...@chromium.org
clamy@chromium.org
 jdonnelly@chromium.org
tedc...@chromium.org
est...@chromium.org
 creis@chromium.org
pkasting@chromium.org
lukasza@chromium.org
alex...@chromium.org
arthu...@chromium.org
mea...@chromium.org
 nasko@chromium.org
 mgiuca@chromium.org

Status:

Fixed (Closed)

Components:

[UI>Browser>Omnibox](#)
[UI>Browser>Navigation](#)
[UI>Security>UrlFormatting](#)

Modified:

Dec 30, 2020

Backlog-Rank:

Editors:

EstimatedDays:

NextAction:

OS:

[Android](#)

Pri:

1

Type:

[Bug-Security](#)

Hotlist-Merge-Review
reward-5000
Security_Impact-Stable
Security_Severity-Medium
allpublic
reward-inprocess
CVE_description-submitted
Target-85
M-85
FoundIn-B4
Release-0-M86
merge-merged-4240

Issue 1116280: Self-XSS / Crash via window.open and delayed navigation

Reported by [herre...@gmail.com](#) on Thu, Aug 13, 2020, 10:26 PM EDT

🔗 Code

VULNERABILITY DETAILS

It is possible to trick Chrome's URL parser and open an empty new tab with an arbitrary scheme that will be executed after the user navigates to it.

In the case of the attack that will be described, a javascript scheme with an arbitrary payload was used - which leads to javascript execution in an arbitrary domain if well-timed with a delayed navigation, but the file:// or chrome:// schemes could be used as well.

A crash (953e347b-60ee-4ff2-9dc1-01661ec0cc9f) also happens when the user tries to send the page URL to their mobile phone - I didn't check whether this has any security implication due to lack of skillz.

The attack works as follows:

1. The user goes to the attacker's website and clicks on a link.
2. A new empty window opens with a javascript scheme and a prepared payload created to deceive the user.
3. The user clicks on the omnibox and tries to reload the page.
4. At the same time step 3 is happening, hopefully after the victim already clicked on the omnibox, the window is redirected by the attacker's script to a Google page.
5. When the victim finally clicks on the button to reload the page, the javascript ends up being executed on any arbitrary origin chosen by the attacker. In this example, [accounts.google.com](#)

Because I think there is no way for the attacker to know when the victim clicked on the omnibox, an arbitrary value was chosen to decide when the redirect should happen (in the PoC, I am using the arbitrary value of 1500ms). Given this, the attack won't always be successful. In the wild, however, the attacker could tweak the time by analyzing which value gives the best rate of success.

I also abuse the fact that when the omnibox is focused, the URL is elided to the left, allowing the javascript scheme to be hidden and a fake Google URL to be put in its place. In the case of this PoC I am using <https://google.com/xyz/webpage/13333337> (for perfect spoofing this URL will need to be dynamically generated by the attacker according to the victim's screen size - in this attack, I am not doing that, so it will probably be a bit misaligned).

When the attack is successful, arbitrary script is executed on an arbitrary origin. This issue is similar to [bug-850834](#).

I have also attached a video simulating the attack.

VERSION

Chrome Version: 84.0.4147.125 (Official Build)

REPRODUCTION CASE

To reproduce the Self-XSS attack follow the steps below:

1. Open <https://lbherreragithub.io/lab/chrome-selfxss/index.html>
2. Click on the link.
3. Click on the omnibox and then click on the "Go/Enter" button on the virtual keyboard.
4. You should see an alert displaying "accounts.google.com".

To reproduce the crash follow the steps below:

1. Make sure you are using Chrome for Desktop.
2. Open <https://lbherreragithub.io/lab/chrome-selfxss/crash.html> and click on the link.
3. In the new window that was opened click on the "Send to your devices" icon located in the top-right side of the omnibox.
4. Select your phone.

5. Your browser should crash.

CREDIT INFORMATION

Reporter credit: Luan Herrera (@lbherrer_)

selfxss.mp4

2.3 MB [View](#) [Download](#)



Comment 1 by [vakh@chromium.org](#) on Fri, Aug 14, 2020, 5:47 AM EDT Project Member

Labels: Needs-Feedback

Thanks for the report.
To help me triage this further, could you please attach the hosted files directly to the bug?

Comment 2 by [vakh@chromium.org](#) on Fri, Aug 14, 2020, 5:48 AM EDT Project Member

Components: UI>Browser>Omnibox UI>Browser>Navigation UI>Security>UriFormatting

Comment 3 by [herre...@gmail.com](#) on Fri, Aug 14, 2020, 12:16 PM EDT

#1, sure.

crash.html

379 bytes [View](#) [Download](#)

selfxss.html

613 bytes [View](#) [Download](#)

Comment 4 by [sheriffbot](#) on Fri, Aug 14, 2020, 12:18 PM EDT Project Member

Cc: [vakh@chromium.org](#)

Labels: -Needs-Feedback

Thank you for providing more feedback. Adding the requester to the cc list.

For more details visit <https://www.chromium.org/issue-tracking/autotriage> - Your friendly Sheriffbot

Comment 5 by [herre...@gmail.com](#) on Thu, Aug 20, 2020, 2:52 PM EDT

Friendly ping!

Comment 6 by [creis@chromium.org](#) on Mon, Aug 24, 2020, 5:27 PM EDT Project Member

Owner: [creis@chromium.org](#)

Ah, given that this is related to [issue-959834](#), I can try to take a look this week.

Comment 7 by [creis@chromium.org](#) on Tue, Aug 25, 2020, 1:25 AM EDT Project Member

Status: Assigned (was: Unconfirmed)

Cc: [nasko@chromium.org](#) [pkasting@chromium.org](#) [arthur...@chromium.org](#)

Labels: Needs-Bisect Security_Severity-Medium Security_Impact-Stable FoundIn-84 OS-Android OS-Chrome OS-Linux OS-Mac OS-Windows Pri-1

I'll post a few observations so far, but I don't have access to my development box at the moment so I can't look too closely just yet.

The attack looks extremely similar to the one from [issue-959834](#), apparently only differing in an extra colon in the link URL. (That colon appears to be necessary for the URL to be executable later; I suspect something in the URL formatting logic has changed in the meantime.) Since the impact is similar, I'll tentatively mark this as the same Medium severity. I've only tested on ChromeOS, but the video shows Android, and I suspect it affects all desktop platforms as well.

The original fix in [r597304](#) (71.0.3572.0) discarded the pending entry when it failed to navigate, and it included a `ChromeNavigationBrowserTest.ClearInvalidPendingURLOnFail` test to verify that. That test doesn't appear to be disabled, but the fix seems to have regressed. I'm curious why the test would still be passing.

I'll also note that the fix was moved from `NavigatorImpl::DiscardPendingEntryIfNeeded` to `NavigationControllerImpl::PendingEntryRefDeleted` in [arthursonzogni@'s r703713](#) (79.0.3937.0). I'm hoping that didn't regress the fix (and that the test would have caught it if so), but again, I'm not sure why the test is passing today.

FWIW, I was hoping to land a more comprehensive fix that prevented navigations to invalid URLs (<https://chromium-review.googlesource.com/c/chromium/src/+1257863>), but I never got it to pass some failing tests on Android WebView and Fuchsia (see <https://crbug.com/959834#c64>).

A few next steps:

- 1) Someone could bisect to see what revision after 71.0.3572.0 the original fix stopped working (assuming that the original fix was effective against this near-duplicate in 71.0.3572.0 itself).
- 2) Someone with a debugger could check to see if we're getting to `NavigationControllerImpl::PendingEntryRefDeleted` and why we aren't discarding the entry, assuming it has an invalid GURL.
- 3) Someone could repro the crash and report the crash ID, to confirm it's not exploitable. (The crash ID listed in [comment 0](#) isn't the right format-- we need the server ID shown on `chrome://crashes`.)

Hopefully I can get access to my development box again shortly to help.

Comment 8 by [herre...@gmail.com](#) on Tue, Aug 25, 2020, 2:10 AM EDT

#7: Oops, here's the crash ID in the right format: `crash/455e825b74f5505f`

Comment 9 by [arthursonzogni@google.com](#) on Tue, Aug 25, 2020, 3:41 AM EDT Project Member

Cc: [clamy@chromium.org](#)

Labels: -OS-Linux -OS-Windows -OS-Chrome -OS-Mac

The POC:

```
...
<script>
const redirect = () => {
  setTimeout(() => {
    open("https://accounts.google.com/error", "win");
  }, 1500);
}
</script>

<a href="o.o:@javascript:://google.com/error/%0Aalert(document.domain)//https://google.com/xyz/webpage/13333333" target="win" onclick="redirect();">Click here to go to
https://google.com</a>
...
```

What happens:

- 1) Open a new window with an URL rewritten to `javascript:uri` (This must not happen).
- 2) After a timeout, navigate the new window to a different origin.
- 3) If the user was editing the URL on Android in between (1) and (2), then the URL is kept. If they press enter, then it will be executed on the new origin.

I suppose this can only be exploitable on Android. On desktop, the URL is not kept after (2).

I started:

```
...
./tools/bisect-builds.py -g 599034 -b 768962 --verify-range --use-local-cache --archive linux64
...
```

I have a poor internet connexion (only today), it take times to download new revision. I will get the results later during the day.

At least, I can confirm there was a regression in the omnibox:

- On chrome M71, the URL is rewritten about:blank.
- On Chrome M84, the URL is rewritten into a javascript-URL.

Comment 10 by [arthursonzogni@google.com](#) on Tue, Aug 25, 2020, 4:10 AM EDT Project Member

`./tools/bisect-builds.py -g 599034 -b 768962 --verify-range --use-local-cache --archive linux64`

Downloading list of known revisions...

Loaded revisions 41523-794497 from `/home/arthursonzogni/chromium/src/tools/bisect-builds-cache.json`

Downloading revision 599034...

Received 110225059 of 110225059 bytes, 100.00%

Trying revision 599034...

Revision 599034 is `[[good](bad)(retry)(unknown)(s)tdout(q)uit]: g`

Downloading revision 768959...

Trying revision 768959...

Revision 768959 is `[[good](bad)(retry)(unknown)(s)tdout(q)uit]: b`

Downloading revision 683105...

Bisecting range [599034 (good), 768959 (bad)], roughly 16 steps left.

Trying revision 683105...

Revision 683105 is `[[good](bad)(retry)(unknown)(s)tdout(q)uit]: g`

Downloading revision 725171...

Received 126837745 of 126837745 bytes, 100.00%

Bisecting range [683105 (good), 768959 (bad)], roughly 15 steps left.

Trying [revision 725171](#)...

[Revision 725171](#) is [(g)ood/(b)ad/(r)etry/(u)nkknown/(s)tdout/(q)uit]: b

Downloading [revision 703527](#)...

Received 123370611 of 123370611 bytes, 100.00%

Bisecting range [683105 (good), 725171 (bad)], roughly 14 steps left.

Trying [revision 703527](#)...

[Revision 703527](#) is [(g)ood/(b)ad/(r)etry/(u)nkknown/(s)tdout/(q)uit]: g

Downloading [revision 714775](#)...

Bisecting range [703527 (good), 725171 (bad)], roughly 13 steps left.

Trying [revision 714775](#)...

[Revision 714775](#) is [(g)ood/(b)ad/(r)etry/(u)nkknown/(s)tdout/(q)uit]: g

Downloading [revision 720137](#)...

Received 126775856 of 126775856 bytes, 100.00%

Bisecting range [714775 (good), 725171 (bad)], roughly 12 steps left.

Trying [revision 720137](#)...

[Revision 720137](#) is [(g)ood/(b)ad/(r)etry/(u)nkknown/(s)tdout/(q)uit]: g

Downloading [revision 722782](#)...

Received 126949111 of 126949111 bytes, 100.00%

Bisecting range [720137 (good), 725171 (bad)], roughly 11 steps left.

Trying [revision 722782](#)...

[Revision 722782](#) is [(g)ood/(b)ad/(r)etry/(u)nkknown/(s)tdout/(q)uit]: g

Downloading [revision 724138](#)...

Received 127157117 of 127157117 bytes, 100.00%

Bisecting range [722782 (good), 725171 (bad)], roughly 10 steps left.

Trying [revision 724138](#)...

[Revision 724138](#) is [(g)ood/(b)ad/(r)etry/(u)nkknown/(s)tdout/(q)uit]: g

Downloading [revision 724759](#)...

Received 126773765 of 126773765 bytes, 100.00%

Bisecting range [724138 (good), 725171 (bad)], roughly 9 steps left.

Trying [revision 724759](#)...

[Revision 724759](#) is [(g)ood/(b)ad/(r)etry/(u)nkknown/(s)tdout/(q)uit]: b

Downloading [revision 724459](#)...

Received 126751833 of 126751833 bytes, 100.00%

Bisecting range [724138 (good), 724759 (bad)], roughly 8 steps left.

Trying [revision 724459](#)...

[Revision 724459](#) is [(g)ood/(b)ad/(r)etry/(u)nkknown/(s)tdout/(q)uit]: g

Downloading [revision 724579](#)...

Received 126773985 of 126773985 bytes, 100.00%

Bisecting range [724459 (good), 724759 (bad)], roughly 7 steps left.

Trying [revision 724579](#)...

[Revision 724579](#) is [(g)ood/(b)ad/(r)etry/(u)nkknown/(s)tdout/(q)uit]: g

Downloading [revision 724656](#)...

Received 126764683 of 126764683 bytes, 100.00%

Bisecting range [724579 (good), 724759 (bad)], roughly 6 steps left.

Trying [revision 724656](#)...

[Revision 724656](#) is [(g)ood/(b)ad/(r)etry/(u)nkknown/(s)tdout/(q)uit]: g

Downloading [revision 724706](#)...

Received 126770550 of 126770550 bytes, 100.00%

Bisecting range [724656 (good), 724759 (bad)], roughly 5 steps left.

Trying [revision 724706](#)...

[Revision 724706](#) is [(g)ood/(b)ad/(r)etry/(u)nkknown/(s)tdout/(q)uit]: b

Downloading [revision 724677](#)...

Received 126762738 of 126762738 bytes, 100.00%

Bisecting range [724656 (good), 724706 (bad)], roughly 4 steps left.

Trying [revision 724677](#)...

[Revision 724677](#) is [(g)ood/(b)ad/(r)etry/(u)nkknown/(s)tdout/(q)uit]: g

Downloading [revision 724691](#)...

Received 126764089 of 126764089 bytes, 100.00%

Bisecting range [724677 (good), 724706 (bad)], roughly 3 steps left.

Trying [revision 724691](#)...

[Revision 724691](#) is [(g)ood/(b)ad/(r)etry/(u)nkknown/(s)tdout/(q)uit]: g

Downloading [revision 724699](#)...

Received 126769488 of 126769488 bytes, 100.00%

Bisecting range [724691 (good), 724706 (bad)], roughly 2 steps left.

Trying [revision 724699](#)...

[Revision 724699](#) is [(g)ood/(b)ad/(r)etry/(u)nkknown/(s)tdout/(q)uit]: g

Downloading [revision 724704](#)...

Received 126770223 of 126770223 bytes, 100.00%

Bisecting range [724699 (good), 724706 (bad)], roughly 2 steps left.

Trying [revision 724704](#)...

[Revision 724704](#) is [(g)ood/(b)ad/(r)etry/(u)nkknown/(s)tdout/(q)uit]: b

You are probably looking for a change made after 724699 (known good), but no later than 724704 (first known bad).

CHANGELOG URL:

<https://chromium.googlesource.com/chromium/src/+log/63a1921cf9791a21f1ea20aba347c0eb7af182a..c54c29c8d778075fd829526501d4826a16790456>

Comment 11 by [arthursonzogni@google.com](#) on Tue, Aug 25, 2020, 4:14 AM EDT Project Member

Owner: lukasza@chromium.org
Cc: creis@chromium.org

The bisection from [comment 10](#) gives 5 possible patches:
...

c54c29c Make sure that empty port doesn't lead to invalidating a valid GURL. by Lukasz Anforowicz · 9 months ago
f7bc361 Added check for SSN and OTP for password manager by Maria Kazinova · 9 months ago
61b8929 Convert Chromium Mac Desktop Reachable ObjC Instance Variable Names From Trailing to Leading Underscore by Robert Liao · 9 months ago
97d32e3 [iOS][EG2] Enabling AccountsTableTests by Jérôme Lebel · 9 months ago
51a7fe45 [build] Test without pointer-compression on one builder by Michael Achenbach · 9 months ago
...

The first is clearly the one we are looking for:
...

Make sure that empty port doesn't lead to invalidating a valid GURL.

Before this CL, url_formatter::FixupURL could "fix" a valid GURL with empty port into an invalid GURL. This CL fixes this.

[Bug-1030346](#)

Change-Id: I93d530d8263b9755fafcc814d99820cf1958eadf
Reviewed-on: <https://chromium-review-googlesource.com/c/chromium/src/+1964640>
Commit-Queue: Łukasz Anforowicz <lukasza@chromium.org>
Reviewed-by: Alex Moshchuk <alexmos@chromium.org>
Reviewed-by: Peter Kasting <pkasting@chromium.org>
Cr-Commit-Position: refs/heads/master@{#724704}

chrome/browser/chrome_navigation_browsertest.cc

components/url_formatter/url_fixer.cc
components/url_formatter/url_fixer_unittest.cc
...

It also explain why the new POC require an extra colon.

[Comment 12](#) by [arthursonzogni@google.com](#) on Tue, Aug 25, 2020, 4:15 AM EDT Project Member
Cc: alex...@chromium.org

[Comment 13](#) by [arthursonzogni@google.com](#) on Tue, Aug 25, 2020, 9:05 AM EDT Project Member
lukasza@ is OOO.
I think we should revert the patch + add a regression test. Then we need to merge it back to M86.

I will make the revert.

[Comment 14](#) by [arthu...@chromium.org](#) on Tue, Aug 25, 2020, 12:26 PM EDT Project Member
The revert will be there:
<https://chromium-review.googlesource.com/c/chromium/src/+2375332>

[Comment 15](#) by [arthu...@chromium.org](#) on Tue, Aug 25, 2020, 1:09 PM EDT Project Member
I am not fully convinced about the revert. I had to disable one test and update the some expectations.

I will be OOO tomorrow. creis@ / nasko@, feel free to land the revert if you are happy with it. Or fix this another way.

[Comment 16](#) by [sheriffbot](#) on Tue, Aug 25, 2020, 2:14 PM EDT Project Member
Labels: Target-85 M-85
Setting milestone and target because of Security_Impact=Stable and medium severity.
For more details visit <https://www.chromium.org/issue-tracking/autotriage> - Your friendly Sheriffbot

[Comment 17](#) by [creis@chromium.org](#) on Tue, Aug 25, 2020, 8:32 PM EDT Project Member
Thanks Arthur! That's a big help. I can confirm that the new variation on the attack URL is treated as a valid URL in PendingRefDeleted, causing it to be left in the address bar. The ClearInvalidPendingURLOnFail test wasn't exercising the relevant part of the URL formatter, which explains why it is still passing.

I'm also nervous about jumping straight to reverting [r724704](#) (which landed in 81.0.3996.0), without understanding what it was needed for or what effect that will have. I'm hopeful we can look a bit closer before landing something, though I agree that we probably don't want to wait until next week. I'll try to look again soon now that I have a debug build working.

Oh, and the crash aspect of this isn't concerning-- it's a simple null deref of GetLastCommittedEntry of a WebContents that doesn't have one. That's tracked in issue 1034863, and I'll give it a ping.

[Comment 18](#) by [creis@chromium.org](#) on Tue, Aug 25, 2020, 8:34 PM EDT Project Member
Cc: tgu...@chromium.org

[Comment 19](#) by [sheriffbot](#) on Fri, Aug 28, 2020, 1:37 PM EDT Project Member
lukasza: Uh oh! This issue still open and hasn't been updated in the last 14 days. This is a serious vulnerability, and we want to ensure that there's progress. Could you please leave an update with the current status and any potential blockers?

If you're not the right owner for this issue, could you please remove yourself as soon as possible or help us find the right one?

If the issue is fixed or you can't reproduce it, please close the bug. If you've started working on a fix, please set the status to Started.

Thanks for your time! To disable nags, add the Disable-Nags label.

For more details visit <https://www.chromium.org/issue-tracking/autotriage> - Your friendly Sheriffbot

[Comment 20](#) by [creis@chromium.org](#) on Fri, Aug 28, 2020, 5:38 PM EDT Project Member
I'm not sure why the nag email kicked in, since we've been posting updates this week. At any rate, I haven't had time to take a closer look at lukasza's fix, so getting his thoughts when he returns on Monday seems like the best plan.

[Comment 21](#) by [lukasza@chromium.org](#) on Mon, Aug 31, 2020, 2:59 PM EDT Project Member
Cc: tedc...@chromium.org mea...@chromium.org
pkasting@, could you PTAL at the notes and questions below?

RE: pkasting@: (in another issue) <https://crbug.com/950924#e43>: I'm confident you won't get a javascript: URL unless we navigated to an invalid URL.

Well, in the repro from this bug (see <https://crbug.com/2385921/2>), we seem to get a "valid" URL in NavigationControllerImpl::PendingEntryRefDeleted and yet the Omnibox ends up displaying a javascript URL:

NavigationControllerImpl::PendingEntryRefDeleted: pending_entry->GetURL().is_valid() = 1; pending_entry->GetURL().possibly_invalid_spec() = [http://o.o/javascript/google.com/error/%0Aalert\(document.domain\)//https://google.com/xyz/webpage/13333337](http://o.o/javascript/google.com/error/%0Aalert(document.domain)//https://google.com/xyz/webpage/13333337); should_preserve_entry = 1

Value of: omnibox_text
Expected: doesn't start with "javascript."
Actual: "javascript://google.com/error/%0Aalert(document.domain)//https://google.com/xyz/webpage/13333337"

I understand why Chrome might want to "fix up" a URL entered by a user into the omnibox. I don't understand why Chrome "fixes up" a URL coming from a renderer-initiated navigation. I assume that this security bug would go away if Chrome only "fixed up" URLs of browser-initiated navigations. Is this something that we have considered in the past?

I support meacer@s statement from <https://crbug.com/950924#e6>: "the page should never be able to put a javascript: URL in the omnibox". So, if we need to "fix up" URLs coming from renderer-initiated navigations, then I would also vote for using StripJavaScriptSchemas to make sure "fixing upping" doesn't accidentally turn something into a javascript URL.

I note that in [issue 950924](#), the reporter said that the exploit works because "a redirect doesn't remove focus from the omnibox nor resets the URL automatically (unlike happens in the Desktop version of Chrome)". OTOH, it seems that this fix direction cannot be pursued due to tedchoc@s concerns about text input usability on mobile (e.g. see <https://crbug.com/950924#e9>).

RE: Reverting [r724704](#)

AFAIU the revert wouldn't have any obvious negative security consequences (mostly negative perf consequences). Still, I'd rather focus on the long-term fix for the current bug (given that the repro requires user interactions, the repro is racey, the fix for [issue-960024](#) took half a year anyway).

[Comment 22](#) by [pkasting@chromium.org](#) on Mon, Aug 31, 2020, 4:10 PM EDT Project Member

It looks to me like running fixup was added to solve [bug-440820](#).

I'm suspicious about whether fixup was truly necessary there. Maybe GURL canonicalization should take care of this? Fixup is a heavy hammer to wield and I tend to agree it shouldn't be used on renderer URLs.

[Comment 23](#) by [lukasza@chromium.org](#) on Tue, Sep 1, 2020, 5:52 PM EDT Project Member

I looked closer and it seems to me that url_formatter::FixupURL is innocent - there are no cases where this function gets as input something that doesn't start with "javascript:" and return something that does start with "javascript:".

So, I think we need to figure out how the Omnibox gets populated with a text that starts with "javascript:". AFAICT, this is happening in the following callstack:

```
views::TextfieldModel::SetText()
views::Textfield::SetTextWithoutCaretBoundsChangeNotification()
OmniboxViewViews::SetTextAndSelectedRanges()
OmniboxViewViews::SetWindowTextAndCaretPos()
OmniboxEditModel::Revert() <- this calls GetPermanentDisplayText which just returns |display_text_|
OmniboxView::RevertAll()
OmniboxViewViews::Update()
LocationBarView::Update()
ToolBarView::Update()
Browser::UpdateToolBar()
Browser::ScheduleUIUpdate()
Browser::NavigationStateChanged()
content::WebContentsImpl::NotifyNavigationStateChanged()
content::Navigator::GetNavigationEntryForRendererInitiatedNavigation()
content::Navigator::OnBeginNavigation()
content::RenderFrameHostImpl::BeginNavigation()
```

OmniboxEditModel::display_text_ is populated here:

```
OmniboxEditModel::ResetDisplayTexts()
OmniboxViewViews::Update()
LocationBarView::Update()
ToolBarView::Update()
Browser::UpdateToolBar()
Browser::ScheduleUIUpdate()
Browser::NavigationStateChanged()
content::WebContentsImpl::NotifyNavigationStateChanged()
content::Navigator::GetNavigationEntryForRendererInitiatedNavigation()
content::Navigator::OnBeginNavigation()
content::RenderFrameHostImpl::BeginNavigation()
```

It seems to me that LocationBarModel incorrectly chops off "http://" and leaves "javascript:" as the starting substring - this is what OmniboxEditModel::ResetDisplayTexts sees:

```
location_bar_model->GetURL() = http://o.o@javascript://foo.com%0Aalert(document.domain)
location_bar_model->GetURLForDisplay() = javascript://foo.com%0Aalert(document.domain)
location_bar_model->GetFormattedFullURL() = javascript://foo.com%0Aalert(document.domain);
```

You can see that the //content layer returns a more-or-less benign url: [http://o.o@javascript://foo.com%0Aalert\(document.domain\)](http://o.o@javascript://foo.com%0Aalert(document.domain))

OTOH, GetURLForDisplay and GetFormattedFullURL transform the benign URL into a javascript: URL.

PS. Note that I've run the test in interactive/painting mode and I have manually tested/verified that after focusing the omnibox, the text being edited also starts with "javascript:". In other words, this is not just a matter of chopping of "http:" for display purposes - the chopped off URL is also used when editing the omnibox.

[Comment 24](#) by [lukasza@chromium.org](#) on Tue, Sep 1, 2020, 6:00 PM EDT Project Member

Owner: [tommycli@chromium.org](#)

Cc: [est...@chromium.org](#) [lukasza@chromium.org](#)

[tommycli@](#), based on [#c23](#) it seems to me that the core issue is in LocationBarModel. Would you agree with this assessment? Could you PTAL at the bug (since it seems that you've landed majority of CLs in location_bar_model.h in 2018-2020)?

(also CC-ing [estark@](#) who also has landed some CLs in location_bar_model.h and might be able to help from Chrome Security / Enamel side)

I hope that <https://crrev.com/c/2385921> is a good starting point for taking over the investigation:

*) It contains a regression test that can be used to repro the problem (OTOH, I think the test changes there are not going in the right direction - unlike the old URL/test, the new URL/test should get a pending entry; so, maybe rather than tweaking ChromeNavigationBrowserTest.ClearInvalidPendingURLOnFail we should instead add a new test).

*) It has some ad-hoc printf-style logging that I used to arrive at the conclusions in [#c23](#)

[Comment 25](#) by [sheriffbot](#) on Fri, Sep 11, 2020, 1:37 PM EDT Project Member

[tommycli](#): Uh oh! This issue still open and hasn't been updated in the last 28 days. This is a serious vulnerability, and we want to ensure that there's progress. Could you please leave an update with the current status and any potential blockers?

If you're not the right owner for this issue, could you please remove yourself as soon as possible or help us find the right one?

If the issue is fixed or you can't reproduce it, please close the bug. If you've started working on a fix, please set the status to Started.

Thanks for your time! To disable nags, add the Disable-Nags label.

For more details visit <https://www.chromium.org/issue-tracking/autotriage> - Your friendly Sheriffbot

[Comment 26](#) by [tommycli@chromium.org](#) on Mon, Sep 14, 2020, 4:18 PM EDT Project Member

Cc: [jdonnelly@chromium.org](#)

[Comment 27](#) by [vakh@chromium.org](#) on Mon, Sep 14, 2020, 5:37 PM EDT Project Member

Cc: [-vakh@chromium.org](#)

[Comment 28](#) by [tommycli@chromium.org](#) on Tue, Sep 15, 2020, 6:34 PM EDT Project Member

These malicious URLs seem like invalid GURLs. (Please correct me if this assumption is false! If it's a valid assumption, we should add some unit tests in gurl_unittest to validate this hypothesis.)

Putting on my security / defense in depth hat, that I think there's multiple things we should do here.

1. url_formatter tries to format invalid GURLs. That's by design, and we shouldn't change that. We should document that in the API header though. (You currently have to read the implementation to learn this).

2. In LocationBarModelDelegate, we can do either one or both of:

- We need to change ShouldDisplayURL() to return false if the GURL is not valid. If this happens, omnibox will show about:blank for invalid GURLs.
- Or, we need to change ShouldPreventElision() to return true if GURL is invalid. In that case this will prevent the elisions from changing it to a javascript: URL.

I'd lean towards doing both, but I'm still a LITTLE bit confused because of my next bullet point.

3. It's actually surprising to me that LocationBarModelDelegate can return an invalid GURL. This is implemented in chrome_location_bar_model_delegate.cc, and queries content::NavigationEntry. We had assumed that the GetVirtualURL() / GURL() methods there would always return something valid. Is that a bad assumption? Should we change this assumption too?

Comment 29 by [pkasting@chromium.org](#) on Tue, Sep 15, 2020, 7:23 PM EDT Project Member

I'm not certain the URLs are invalid. We certainly shouldn't allow navigation to an invalid GURL, but I think that gets restricted way before this.

Based on [comment 23](#), I think this really is a formatter bug. I think the code in components/url_formatter/url_formatter.cc:FormatUrlWithAdjustments() that sets and uses [strip_scheme] needs to avoid stripping the scheme in this case, similar to how we avoid stripping it for "ftp._____" hosts. I'd have to step through this code to be certain, and to know precisely the right fix.

Comment 30 by [tommycli@chromium.org](#) on Tue, Sep 15, 2020, 7:25 PM EDT Project Member

Hi lukasza:

I've been running the example browsertest you sent, which navigate to:
o.o:@javascript:://foo.com%0Aalert(document.domain);

and I'm getting logs like this:

--

```
[2462611:2462611:0915/161749.211189:ERROR:url_fixer.cc(628)] FixupURL; text = o.o:@javascript:://foo.com%0Aalert(document.domain); desired_tid = ; result.is_valid() = 0; result = http://o.o:@javascript://foo.com%0Aalert(document.domain)
```

```
[2462611:2462611:0915/161749.211225:ERROR:url_fixer.cc(628)] FixupURL; text = http://o.o:@javascript://foo.com%0Aalert(document.domain); desired_tid = ; result.is_valid() = 1; result = http://o.o:@javascript://foo.com%0Aalert(document.domain)
```

```
[2462611:2462611:0915/161749.211637:ERROR:omnibox_edit_model.cc(282)] OmniboxEditModel::ResetDisplayTexts; location_bar_model->GetURL() = http://o.o:@javascript://foo.com%0Aalert(document.domain); url is valid: 0; location_bar_model->GetURLForDisplay() = javascript://foo.com%0Aalert(document.domain); location_bar_model->GetFormattedFullURL() = javascript://foo.com%0Aalert(document.domain)
```

--

It looks like to me that the URL is indeed invalid, and the second FixupURL call makes it valid.

But the LocationBarModel is receiving the result of the FIRST FixupURL call, which is still invalid. The URL in the second FixupURL, I don't know where it goes, but the omnibox is not receiving that.

--

Are there any VALID GURLs that get transformed to javascript: URLs by FormatURL? If so that's a big problem, and we need to update the formatter.

But it SEEMS like to me that the root cause is that the LocationBarModel is receiving an invalid GURL. Can we guarantee that we only get valid GURLs from NavigationEntry?

Thanks.

Comment 31 by [lukasza@chromium.org](#) on Tue, Sep 15, 2020, 8:16 PM EDT Project Member

Owner: lukasza@chromium.org

RE: [#c30](#): tommycli@:

Thanks! Something fishy is indeed going on at the NavigationEntry level. In particular, I note that when NavigationControllerImpl::PendingEntryRefDeleted is called, then GetURL() is valid and GetVirtualURL() is invalid (previously I only looked at GetURL and incorrectly concluded that URLs at the NavigationEntry level are valid):

```
pending_entry_->GetURL()      = http://o.o:@javascript//foo.com%0Aalert(document.domain);
pending_entry_->GetURL().is_valid() = 1;
pending_entry_->GetVirtualURL() = http://o.o:@javascript://foo.com%0Aalert(document.domain);
pending_entry_->GetVirtualURL().is_valid() = 0
```

The URLs differ only by presence of the colon after "javascript:

```
http://o.o:@javascript//foo.com%0Aalert(document.domain);
http://o.o:@javascript://foo.com%0Aalert(document.domain);
```

Comment 32 by [pkasting@chromium.org](#) on Wed, Sep 16, 2020, 3:39 PM EDT Project Member

Cc: tommycli@chromium.org

Comment 33 by [tommycli@chromium.org](#) on Wed, Sep 16, 2020, 3:46 PM EDT Project Member

Lukas / Peter - thanks for your investigations and thanks for [c#30](#).

I'm sent out a CL to you that adds some CHECKs in LocationBarModel to try to "wall off" the problem to a smaller and smaller section of the code.

I think we are getting closer.

Comment 34 by [lukasza@chromium.org](#) on Fri, Sep 18, 2020, 12:46 PM EDT Project Member

I think it is interesting what happens in RewriteUrlForNavigation in //content/browser/renderer_host/navigation_controller_impl.cc [1]:

```
void RewriteUrlForNavigation(const GURL& original_url,
                           BrowserContext* browser_context,
                           GURL* url_to_load,
                           GURL* virtual_url,
                           bool* reverse_on_redirect) {
// #1 *****
// Fix up the given URL before letting it be rewritten, so that any minor
// cleanup (e.g., removing leading dots) will not lead to a virtual URL.
*virtual_url = original_url;
BrowserURLHandlerImpl::GetInstance()->FixupURLBeforeRewrite(virtual_url,
                                                            browser_context);
// #2 *****

// Allow the browser URL handler to rewrite the URL. This will, for example,
// remove "view-source:" from the beginning of the URL to get the URL that
```

```
// will actually be loaded. This real URL won't be shown to the user, just
// used internally.
*url_to_load = *virtual_url;
BrowserURLHandlerImpl::GetInstance()->RewriteURLIfNecessary(
    url_to_load, browser_context, reverse_on_redirect);
// #3 *****
}
```

The |original_url| is valid,
then |original_url| gets rewritten by FixupURLBeforeRewrite into an invalid URL and stored in |virtual_url|
then |virtual_url| is copied to |url_to_load| and rewritten by RewriteURLIfNecessary into a different valid URL....

Output from ad-hoc logging:

```
RewriteUrlForNavigation #1 (input arguments);
original_url = o.o:@javascript:://foo.com%0Aalert(document.domain);
original_url.is_valid() = 1

RewriteUrlForNavigation #2 (after FixupURLBeforeRewrite);
*virtual_url = http://o.o:@javascript//foo.com%0Aalert(document.domain);
virtual_url->is_valid() = 0

RewriteUrlForNavigation #3 (after RewriteURLIfNecessary);
*url_to_load = http://o.o:@javascript//foo.com%0Aalert(document.domain);
url_to_load->is_valid() = 1;
*virtual_url = http://o.o:@javascript//foo.com%0Aalert(document.domain);
virtual_url->is_valid() = 0
```

I think it is a bug that:

- A) FixupURLBeforeRewrite can turn a valid URL into an invalid one
- B) RewriteURLIfNecessary works with invalid URLs - I think that for invalid URLs it should return early and avoid mutating the URL

[1]
https://source.chromium.org/chromium/chromium/src/+master:content/browser/renderer_host/navigation_controller_impl.cc;l=354;drc=e92c33da084dde92a91b2ccc4b35032b95195c6c

Comment 35 by lukasza@chromium.org on Fri, Sep 18, 2020, 7:00 PM EDT Project Member

Status: Started (was: Assigned)

I think that we can fix this bug, by making sure that RewriteURLIfNecessary doesn't rewrite invalid URLs. This will help ensure that NavigationEntry::GetURL and GetVirtualURL are consistent (not one of them valid and the other invalid) which in turn will help ensure that NavigationControllerImpl::PendingEntryRefDeleted (introduced in the previous fix at [r597304](#)) rejects such NavigationEntry. WIP CL is at <https://crrev.com/c/2385921>.

I've also tried to investigate why sometimes FixupURLBeforeRewrite can turn a valid URL into an invalid one. I've opened issue 1130091 to track this investigation going forward. Addressing this is not strictly required for fixing the security bug at hand (I think <https://crrev.com/c/2385921> should be sufficient).

Comment 36 by [bugdroid](#) on Tue, Sep 22, 2020, 6:31 PM EDT Project Member

The following revision refers to this bug:

<https://chromium.googlesource.com/chromium/src.git/+0a131d1daca17c2266d1dd909d72bbc44f4ac130>

commit [0a131d1daca17c2266d1dd909d72bbc44f4ac130](#)

Author: Lukasz Anforowicz <lukasza@chromium.org>

Date: Tue Sep 22 22:28:13 2020

Avoid fixing/rewriting/mutating invalid URLs in RewriteURLIfNecessary.

This CL changes BrowserURLHandlerImpl::RewriteURLIfNecessary so that it returns early (and doesn't mutate the |url| in the in-out argument) if |url| is invalid. This helps avoid scenarios where RewriteUrlForNavigation (in navigation_controller_impl.cc) ends up generating a NavigationEntry with an invalid virtual URL that (accidentally/incorrectly) gets rewritten into a valid URL.

~~Bug-1446280~~

Change-Id: I114cf8cd9459b6931ae659f62a100679b994d5e

Reviewed-on: <https://chromium-review.googlesource.com/c/chromium/src/+2385921>

Reviewed-by: Alex Moshchuk <alexmos@chromium.org>

Commit-Queue: Łukasz Anforowicz <lukasza@chromium.org>

Cr-Commit-Position: refs/heads/master@{#809537}

[modify] https://crrev.com/0a131d1daca17c2266d1dd909d72bbc44f4ac130/chrome/browser/chrome_navigation_browsertest.cc

[modify] https://crrev.com/0a131d1daca17c2266d1dd909d72bbc44f4ac130/content/browser/browser_url_handler_impl.cc

Comment 37 by lukasza@chromium.org on Wed, Sep 23, 2020, 12:57 PM EDT Project Member

Status: Fixed (was: Started)

[r809537](#) from [#c36](#) initially landed in 87.0.4272.0. AFAIU this CL should fix this bug / make the attack not possible anymore. [herrera...@](#), could you please confirm that the attack doesn't work on the latest Chrome Canary?

Comment 38 by lukasza@chromium.org on Wed, Sep 23, 2020, 12:57 PM EDT Project Member

Labels: -Needs-Bisect

Comment 39 by [sheriffbot](#) on Wed, Sep 23, 2020, 3:09 PM EDT Project Member

Labels: -Restrict-View-SecurityTeam Restrict-View-SecurityNotify

Comment 40 by adetaylor@google.com on Mon, Sep 28, 2020, 12:00 AM EDT Project Member

Labels: reward-topanel

Comment 41 by [sheriffbot](#) on Mon, Sep 28, 2020, 3:34 PM EDT Project Member

Labels: Merge-Request-86

Requesting merge to beta M86 because latest trunk commit (809537) appears to be after beta branch point (800218).

For more details visit <https://www.chromium.org/issue-tracking/autotriage> - Your friendly Sheriffbot

Comment 42 by [sheriffbot](#) on Mon, Sep 28, 2020, 3:39 PM EDT Project Member

Labels: -Merge-Request-86 Hotlist-Merge-Review Merge-Review-86

This bug requires manual review. We are only 7 days from stable.
Before a merge request will be considered, the following information is required to be added to this bug:

1. Does your merge fit within the Merge Decision Guidelines?
- Chrome: https://chromium.googlesource.com/chromium/src.git/+master/docs/process/merge_request.md#when-to-request-a-merge
- Chrome OS: <https://goto.google.com/cros-release-branch-merge-guidelines>
2. Links to the CLs you are requesting to merge.
3. Has the change landed and been verified on ToT?
4. Does this change need to be merged into other active release branches (M-1, M+1)?
5. Why are these changes required in this milestone after branch?
6. Is this a new feature?
7. If it is a new feature, is it behind a flag using finch?

Chrome OS Only:

8. Was the change reviewed and approved by the Eng Prod Representative? See Eng Prod ownership by component: <http://go/cros-engprodcomponents>

Please contact the milestone owner if you have questions.

Owners: govind@ (Android), bindusuvama@ (iOS), geohsu@ (ChromeOS), pbommana@ (Desktop)

For more details visit <https://www.chromium.org/issue-tracking/autotriage> - Your friendly Sheriffbot

Comment 43 by gov...@chromium.org on Mon, Sep 28, 2020, 5:02 PM EDT Project Member

Cc: adetaylor@chromium.org

+adetaylor@ (Security TPM) for M86 merge review. Thank you.

Note: We're cutting M86 Stable RC tomorrow, Tuesday noon.

Comment 44 by adetaylor@google.com on Mon, Sep 28, 2020, 6:23 PM EDT Project Member

Labels: -Merge-Review-86 Merge-Approved-86

lukasza@ this seems like a simple fix with no stability implications. I'm assuming there are also no compatibility implications on the basis that this function should never have to deal with an invalid URL in legitimate use cases. As such I'm approving merge to M86, branch 4240. Per #c43 there's some urgency to merge. Thanks!

Comment 45 by lukasza@chromium.org on Mon, Sep 28, 2020, 7:20 PM EDT Project Member

Thanks govind@ and adetaylor@. M86 merge is in progress here: <https://chromium-review.googlesource.com/c/chromium/src/+2436782>

Comment 46 by bugdroid on Mon, Sep 28, 2020, 11:23 PM EDT Project Member

Labels: -merge-approved-86 merge-merged-4240 merge-merged-86

The following revision refers to this bug:

<https://chromium.googlesource.com/chromium/src.git/+513d5dbbf0d2c6098d947b175a71fa353ec84840>

commit 513d5dbbf0d2c6098d947b175a71fa353ec84840

Author: Lukasz Anforowicz <lukasza@chromium.org>

Date: Tue Sep 29 03:22:15 2020

[M86] Avoid fixing/rewriting/etc invalid URLs in RewriteURLIfNecessary.

This CL changes BrowserURLHandlerImpl::RewriteURLIfNecessary so that it returns early (and doesn't mutate the [url] in the in-out argument) if [url] is invalid. This helps avoid scenarios where RewriteUrlForNavigation (in navigation_controller_impl.cc) ends up generating a NavigationEntry with an invalid virtual URL that (accidentally/incorrectly) gets rewritten into a valid URL.

(cherry picked from commit 0a131d1daca17c2266d1dd909d72bbo44f4ac130)

~~Bug-1446299~~

Change-Id: I114cf8cd9459b6931ae659f62a100679b994d5e

Reviewed-on: <https://chromium-review.googlesource.com/c/chromium/src/+2385921>

Reviewed-by: Alex Moshchuk <alexmos@chromium.org>

Commit-Queue: Lukasz Anforowicz <lukasza@chromium.org>

Cr-Original-Commit-Position: refs/heads/master@{#809537}

Reviewed-on: <https://chromium-review.googlesource.com/c/chromium/src/+2436782>

Reviewed-by: Lukasz Anforowicz <lukasza@chromium.org>

Cr-Commit-Position: refs/branch-heads/4240@{#1067}

Cr-Branched-From: f297677702651916bbf65e59c0d4bbd4ce57d1ee-refs/heads/master@{#800218}

[modify] https://crrev.com/513d5dbbf0d2c6098d947b175a71fa353ec84840/chrome/browser/chrome_navigation_browsertest.cc

[modify] https://crrev.com/513d5dbbf0d2c6098d947b175a71fa353ec84840/content/browser/browser_url_handler_impl.cc

Comment 47 by adetaylor@google.com on Wed, Sep 30, 2020, 6:49 PM EDT Project Member

Labels: -reward-topanel reward-unpaid reward-5000

*** Boilerplate reminders! ***

Please do NOT publicly disclose details until a fix has been released to all our users. Early public disclosure may cancel the provisional reward. Also, please be considerate about disclosure when the bug affects a core library that may be used by other products. Please do NOT share this information with third parties who are not directly involved in fixing the bug. Doing so may cancel the provisional reward. Please be honest if you have already disclosed anything publicly or to third parties. Lastly, we understand that some of you are not interested in money. We offer the option to donate your reward to an eligible charity. If you prefer this option, let us know and we will also match your donation - subject to our discretion. Any rewards that are unclaimed after 12 months will be donated to a charity of our choosing.

Please contact security-vrp@chromium.org with any questions.

Comment 48 by adetaylor@google.com on Wed, Sep 30, 2020, 7:00 PM EDT Project Member

Congratulations! The VRP panel has decided to award \$5000 for this bug.

Comment 49 by adetaylor@google.com on Thu, Oct 1, 2020, 2:32 PM EDT Project Member

Labels: -reward-unpaid reward-inprocess

Comment 50 by adetaylor@google.com on Thu, Oct 1, 2020, 3:47 PM EDT Project Member

Labels: Release-0-M86

Comment 51 by adetaylor@google.com on Mon, Oct 5, 2020, 1:00 AM EDT Project Member

Labels: CVE-2020-15978 CVE_description-missing

Comment 52 by adetaylor@google.com on Mon, Nov 2, 2020, 9:15 PM EST Project Member

Labels: -CVE_description-missing CVE_description-submitted

Comment 53 by sheriffbot on Wed, Dec 30, 2020, 1:50 PM EST Project Member

Labels: -Restrict-View-SecurityNotify allpublic

This bug has been closed for more than 14 weeks. Removing security view restrictions.

For more details visit <https://www.chromium.org/issue-tracking/autotriage> - Your friendly Sheriffbot