» Issue43882



This issue tracker **has been migrated to GitHub**, and is currently **read-only**.
For more information, **see the GitHub FAQs in the Python's Developer Guide.**



This issue has been migrated to GitHub:
https://github.com/python/cpython/issues/88048

classification

Title:	[security] CVE-2022-0391: urllib.parse should sanitize urls containing ASCII newline and tabs.		
Type:	security	**Stage**:	commit review
Components:	Library (Lib)	**Versions**:	Python 3.11, Python 3.10, Python 3.9, Python 3.8, Python 3.7, Python 3.6

process

Status:	closed	**Resolution**:	fixed
Dependencies:		**Superseder**:	
Assigned To:	orsenthil	**Nosy List**:	Mike.Lissner, apollo13, felixxm, gregory.p.smith, lukasz.langa, mgorny, miss-islington, ned.deily, odd_bloke, orsenthil, pablogsal, sethmlarson, vstinner, xtreak
Priority:	high	**Keywords**:	patch

Created on **2021-04-18 19:37** by **orsenthil**, last changed **2022-04-11 14:59** by **admin**. This issue is now **closed**.

Pull Requests

URL	Status	Linked	Edit
PR 25595	merged	orsenthil, 2021-04-25 14:49	
PR 25725	merged	miss-islington, 2021-04-29 17:17	
PR 25726	merged	miss-islington, 2021-04-29 17:17	
PR 25727	closed	miss-islington, 2021-04-29 17:17	
PR 25728	closed	miss-islington, 2021-04-29 17:17	
PR 25853	merged	orsenthil, 2021-05-03 14:08	
PR 25921	merged	orsenthil, 2021-05-05 17:00	
PR 25923	merged	miss-islington, 2021-05-05 17:25	
PR 25924	merged	miss-islington, 2021-05-05 17:25	
PR 25936	merged	miss-islington, 2021-05-05 22:50	
PR 26267	merged	orsenthil, 2021-05-20 14:10	
PR 26268	merged	orsenthil, 2021-05-20 14:46	

PR 26275	merged	orsenthil, 2021-05-21 00:51
PR 26276	merged	orsenthil, 2021-05-21 01:33
PR 26277	merged	orsenthil, 2021-05-21 01:40

Messages (47)

msg391343 - (view) **Author: Senthil Kumaran (orsenthil) *** **Date: 2021-04-18 19:36**

A security issue was reported by Mike Lissner wherein an attacker was able to use `\r\n` in the url path, the urlparse method didn't sanitize and allowed those characters be present in the request.

```
> In [9]: from urllib.parse import urlsplit
> In [10]: urlsplit("java\nscript:alert('bad')")
> Out[10]: SplitResult(scheme='', netloc='', path="java\nscript:alert('bad')",
query='', fragment='')
```

Firefox and other browsers ignore newlines in the scheme. From
the browser console:

```
>> new URL("java\nscript:alert(bad)")
<< URL { href: "javascript:alert(bad)", origin: "null", protocol:
"javascript:", username: "", password: "", host: "", hostname: "", port: "",
pathname: "alert(bad)", search: ""
```

Mozilla Developers informed about the controlling specification for URLs is in fact defined by the "URL Spec"
from WHATWG which updates RFC 3986 and specifies that tabs and newlines should be stripped from the scheme.

See: **https://url.spec.whatwg.org/#concept-basic-url-parser**

That link defines an automaton for URL parsing. From that link, steps 2 and 3 of scheme parsing read:

If input contains any ASCII tab or newline, validation error.
3. Remove all ASCII tab or newline from input.

---

urlparse module behavior should be updated, and an ASCII tab or newline should be removed from the url (sanitized) before it is sent to the request, as WHATWG spec.

msg391352 - (view) **Author: Karthikeyan Singaravelan (xtreak) *** **Date: 2021-04-19 03:24**

See also a related issue to sanitise newline on other helper functions
https://bugs.python.org/issue30713

See also discussion and compatibility on disallowing control characters :
https://bugs.python.org/issue30458

msg391426 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2021-04-20 10:41**

See also **bpo-43883**.

msg391859 - (view) **Author: Senthil Kumaran (orsenthil) *** **Date: 2021-04-25 14:53**

I have added a PR to remove ascii newlines and tabs from URL input. It is as per the WHATWG spec.

However, I still like to research more and find out if this isn't introducing behavior that will break existing systems. It should also be aligned the decisions we have made with previous related bug reports.

Please review.

msg392334 - (view) **Author: Senthil Kumaran (orsenthil) *** **Date: 2021-04-29 17:16**

New changeset **76cd81d60310d65d01f9d7b48a8985d8ab89c8b4** by Senthil Kumaran in branch 'master':
~~**bpo-43882**~~ - urllib.parse should sanitize urls containing ASCII newline and tabs. (~~**GH-25595**~~)
https://github.com/python/cpython/commit/76cd81d60310d65d01f9d7b48a8985d8ab89c8b4

msg392338 - (view) **Author: Senthil Kumaran (orsenthil) *** **Date: 2021-04-29 17:57**

New changeset **491fde0161d5e527eeff8586dd3972d7d3a631a7** by Miss Islington (bot) in branch '3.9':
[3.9] ~~**bpo-43882**~~ - urllib.parse should sanitize urls containing ASCII newline and tabs. (~~**GH-25595**~~) (~~**GH-25725**~~)
https://github.com/python/cpython/commit/491fde0161d5e527eeff8586dd3972d7d3a631a7

msg392611 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2021-05-01 17:26**

I think there's still a flaw in the fixes implemented in 3.10 and 3.9 so far. We're closer, but probably not quite good enough yet.

why? We aren't stripping the newlines+tab early enough.

I think we need to do the stripping *right after* the _coerce_args(url, ...) call at the start of the function.

Otherwise we
  (1) are storing url variants with the bad characters in _parse_cache [a mere slowdown in the worst case as it'd just overflow the cache sooner]
  (2) are splitting the scheme off the URL prior to stripping. in 3.9+ there is a check for valid scheme characters, which will defer to the default scheme when found. The WHATWG basic url parsing has these characters stripped before any parts are split off though, so 'ht\rtps' - for example - would wind up as 'https' rather than our behavior so far of deferring to the default scheme.

I noticed this when reviewing the pending 3.8 PR as it made it more obvious due to the structure of the code and would've allowed characters through into query

and fragment in some cases.
https://github.com/python/cpython/pull/25726#pullrequestreview-649803605

msg392781 - (view) **Author: Łukasz Langa (lukasz.langa) *** **Date: 2021-05-03 09:10**

Good catch, Greg. Since it's not merged already, this change will miss 3.8.10 but as a security fix will be included in 3.8.11 later in the year.

The partial fix already landed in 3.9 will be released in 3.9.5 later today unless it's amended or reverted in a few hours.

msg392808 - (view) **Author: Senthil Kumaran (orsenthil) *** **Date: 2021-05-03 14:12**

Based on Greg's review comment, I have pushed the fix for 3.9, and 3.8

- [3.9] **https://github.com/python/cpython/pull/25853**
- [3.8] **https://github.com/python/cpython/pull/25726**

There is no need to hold off releases for these alone. If we get it merged before the release cut today, fine, otherwise, they will be in the next security fix.

msg392835 - (view) **Author: Senthil Kumaran (orsenthil) *** **Date: 2021-05-03 19:09**

New changeset **8a595744e696a0fb92dccc5d4e45da41571270a1** by Senthil Kumaran in branch '3.9':
[3.9] ~~**bpo-43882**~~ Remove the newline, and tab early. From query and fragments. (~~**#25853**~~)
https://github.com/python/cpython/commit/8a595744e696a0fb92dccc5d4e45da41571270a1

msg392873 - (view) **Author: Michał Górny (mgorny) *** **Date: 2021-05-04 10:57**

I hate to be the bearer of bad news but I've already found this change to be breaking tests of botocore and django. In both cases, the test failure is apparently because upstream used to reject URLs after finding newlines in the split components, and now they're silently stripped away.

Filed bugs:
https://github.com/boto/botocore/issues/2377
https://code.djangoproject.com/ticket/32713

Note that I'm not saying the change should be reverted.

msg392926 - (view) **Author: Seth Michael Larson (sethmlarson)** **Date: 2021-05-04 17:26**

Leaving a thought here, I'm highlighting that we're now implementing two different standards, RFC 3986 with hints of WHATWG-URL. There are pitfalls to doing so as now a strict URL parser for RFC 3986 (like the one used by urllib3/requests) will give different results compared to Python and thus opens up the door for SSRF vulnerabilities [1].

[1]: **https://www.blackhat.com/docs/us-17/thursday/us-17-Tsai-A-New-Era-Of-SSRF-Exploiting-URL-Parser-In-Trending-Programming-Languages.pdf**

msg392944 - (view) **Author: Mike Lissner (Mike.Lissner)** **Date: 2021-05-04 20:16**

I haven't watched that Blackhat presentation yet, but from the slides, it seems like the fix is to get all languages parsing URLs the same as the browsers. That's what @orsenthil has been doing here and plans to do in **https://bugs.python.org/issue43883**.

Should we get a bug filed with requests/urllib3 too? Seems like a good idea if it suffers from the same problems.

msg392971 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2021-05-05 02:19**

Both Django and Botocore issues appear to be in the category of: "depending on invalid data being passed through our urlsplit API so that they could look for it later" Not much sympathy. We never guaranteed we'd pass invalid data through. They're depending on an implementation detail (Hyrum's law). Invalid data causes other people who don't check for it problems. There is no valid solution on our end within the stdlib that won't frustrate somebody.

We chose to move towards safer (undoubtedly not perfect) by default.

Instead of the patches as you see them, we could've raised an exception. I'm sure that would also also have tripped up existing code depending on the undesirable behavior.

If one wants to reject invalid data as an application/library/framework, they need a validator. The Python stdlib does not provide a URL validation API. I'm not convinced we would even want to (though that could be something **issue43883** winds up providing) given how perilous that is to get right: Who's version of right? which set of standards? when and why? Conclusion: The web... such a mess.

msg392995 - (view) **Author: Michał Górny (mgorny) *** **Date: 2021-05-05 09:11**

In my opinion, raising an exception would have been safer.

Botocore and django do precisely what you say — provide a validator. To make this validator easier, they do the validation on splitted up URL parts.

I disagree with the premise that they were stupid to rely on invalid data being passed through. I could understand if the function started rejecting invalid data. But until now, you could reasonably assume that urlsplit()'s output would correspond to its input. Making the output 'sanitized' means that invalid input is converted into valid output. This goes against the principle of least surprise.

In the end, this opens us potential vulnerabilities in other packages. Imagine that something uses urlsplit() to perform the URL validation but uses the original URL elsewhere. By making the validation happen on a sanitized URL, you're effectively disarming the validator and letting bad URL through.

Security is not only about fixing potential problems with your package. It's also about considering the consequences to your users. In this case, the chosen solution may actually open more vulnerabilities that it fixes. What's even worse, you're actively harming security in projects that actually attempted to solve the same problem earlier.

msg393009 - (view) **Author: Florian Apolloner (apollo13)** **Date: 2021-05-05 13:52**

Thank you for the kind words Michał. We (Django) are exactly in the position that you describe. Our validation, at least for now has to stay strict, exactly to prevent fallout further down the road (see **https://github.com/django/django/pull/14349#pullrequestreview-652022529** for details).

Sure, we might have been a bit naive when relying on urllib.parse for parts of our validation routines, but this is why we have tests for this behavior. We can easily work around this fix and will issue a release shortly to prevent security issues for users on newer Python versions. But no matter how the Python code ends up in the long run, our validator (at least this specific class) cannot simply accept new URLs because a spec changed. We owe it to our users to keep in mind that relaxing the validation can cause other issues down the road.

msg393025 - (view) **Author: Łukasz Langa (lukasz.langa) *** **Date: 2021-05-05 17:25**

New changeset **515a7bc4e13645d0945b46a8e1d9102b918cd407** by Miss Islington (bot) in branch '3.8':
[3.8] ~~**bpo-43882**~~ - urllib.parse should sanitize urls containing ASCII newline and tabs. (~~**GH-25595**~~) (**#25726**)
https://github.com/python/cpython/commit/515a7bc4e13645d0945b46a8e1d9102b918cd407

msg393030 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2021-05-05 18:13**

Thanks Florian! Indeed, I'm glad you have tests for this. (I expect anyone writing their own validation code will have such tests)

Making urlsplit raise an exception where it never has before has other consequences:

In CPython's own test suite test_urllib fails as many of its tests for validation that these characters are either ignored or cause a specific http.client.InvalidURL error on urlopen() start failing. I draw no conclusions from that other than we'd need to rework some of those tests. It just reflects the state of our test suite and even inconsistent behavior between excluding the characters or erroring within the http.client code on them based on past CVEs.

Regardless, if people would people prefer to see urlsplit `raise SomeExistingException(f'Invalid characters in URL or scheme. url={url!r} scheme={scheme!r}')` in 3.9.6 and the security patch releases for other 3.x versions, evidence that it wouldn't cause alternate problems would be helpful.

I've kicked off tests at work on our huge codebase with both variants as a datapoint to see if that is informative or not.

If we went the exception route: SomeExistingException might make sense as `http.client.InvalidURL`, but that'd be a circular dependency (no big deal) and heavyweight import for urllib.parse to have. `urllib.error.URLError` could also make sense, but that's an OSError descendant and identifies itself as a urlopen error which would be surprising. `ValueError` is a reasonable fallback. But using that guarantees someone will wonder why it isn't one of the other two... As this is a bugfix, defining a new exception isn't an option.

We as a community currently lack a way for security patches to CPython to be tested against a broad swath of projects in advance of them appearing in a release. Once upon a time there were release candidates for patches releases that could serve this purpose...

msg393033 - (view) **Author: Mike Lissner (Mike.Lissner)** **Date: 2021-05-05 18:35**

> Instead of the patches as you see them, we could've raised an exception.

In my mind the definition of a valid URL is what browsers recognize. They're moving towards the WHATWG definition, and so too must we.

If we make python raise an exception when a URL has a newline in the scheme (e..g: "htt\np"), we'd be raising exceptions for *valid* URLs as browsers define them. That doesn't seem right at all to me. I'd be frustrated to have to catch such an exception, and I'd wonder how to pass through valid exceptions without urlparse raising something.

> Making the output 'sanitized' means that invalid input is converted into valid output. This goes against the principle of least surprise.

Well, not quite, right? The URLs this fixes *are* valid according to browsers. Browsers say these tabs and newlines are OK.

I agree though that there's an issue with the approach of stripping input in a way that affects output. That doesn't seem right.

I think the solution I'd favor (and I imagine what's coming in 43883) is to do this properly so that newlines are preserved in the output, but so that the scheme is also placed properly in the scheme attribute.

So instead of this (from the initial report):

> In [9]: from urllib.parse import urlsplit
> In [10]: urlsplit("java\nscript:alert('bad')")
> Out[10]: SplitResult(scheme='', netloc='', path="java\nscript:alert('bad')", query='', fragment='')

We get something like this:

> In [10]: urlsplit("java\nscript:alert('bad')")
> Out[10]: SplitResult(scheme='java\nscript', netloc='', path="alert('bad')", query='', fragment='')

In other words, keep the funky characters and parse properly.

msg393034 - (view) **Author: Mike Lissner (Mike.Lissner)** **Date: 2021-05-05 18:36**

> I'd wonder how to pass through valid exceptions without urlparse raising something.

Oops, meant to say "valid URLs", not valid exceptions, sorry.

msg393039 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2021-05-05 20:26**

Mike: There may be multiple ways to read that WHATWG recommendation? The linked to section is about implementing a state machine for parsing a URL into parts safely. But that may not imply that anything that passed through that state machine should be considered "valid". Just that this spec is able to make some sense out of otherwise messy data. Network adage: Be lenient in what you accept. I doubt anyone would want something producing URLs for consumption by something else to allow these in their output.

I have yet to read the _entire_ WHATWG spec from head to toe to try and better understand the context they had in mind.

I agree that it is unfortunate that the original URL may have these issues and go on to be (re)used in another context where it passes to something that might not treat it in the same manner. That's in some sense a flaw in our API design that we allow string based URLs to be used in APIs rather than require them to have gone through a parsing sanitization step into a instance of a URL object (for example). API design like that is clearly out of scope for this issue. :)

Regardless my gut feeling is that we continue with the existing fix that remove the characters as we've already started releasing. If we need to change our mind on how we've done that, so be it, we can, that'll show up in later patches.

msg393049 - (view) **Author: Senthil Kumaran (orsenthil) *** **Date: 2021-05-05 23:04**

New changeset **24f1d1a8a2c4aa58a606b4b6d5fa4305a3b91705** by Miss Islington (bot) in branch '3.10':
~~**bpo-43882**~~ Remove the newline, and tab early. From query and fragments. (~~**GH-25936**~~)
https://github.com/python/cpython/commit/24f1d1a8a2c4aa58a606b4b6d5fa4305a3b91705

msg393107 - (view) **Author: Ned Deily (ned.deily) *** **Date: 2021-05-06 16:52**

New changeset **f4dac7ec55477a6c5d965e594e74bd6bda786903** by Miss Islington (bot) in branch '3.7':
[3.7] ~~**bpo-43882**~~ - urllib.parse should sanitize urls containing ASCII newline and tabs. (~~**GH-25923**~~)
https://github.com/python/cpython/commit/f4dac7ec55477a6c5d965e594e74bd6bda786903

msg393108 - (view) **Author: Ned Deily (ned.deily) *** **Date: 2021-05-06 16:56**

New changeset **6c472d3a1d334d4eeb4a25eba7bf3b01611bf667** by Miss Islington (bot) in branch '3.6':
[3.6] ~~**bpo-43882**~~ - urllib.parse should sanitize urls containing ASCII newline and tabs (~~**GH-25924**~~)
https://github.com/python/cpython/commit/6c472d3a1d334d4eeb4a25eba7bf3b01611bf667

msg393136 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2021-05-06 19:14**

For completeness reference, the 'main' branch after the master->main rename also got fixed to check it early the same was as the release branches via:

https://github.com/python/cpython/commit/985ac016373403e8ad41f8d563c4355ffa8d49ff

our robot updating bug comments presumably didn't know about the master -> main rename yet so didn't leave a comment here.

msg393139 - (view) **Author: Daniel Watkins (odd_bloke) *** **Date: 2021-05-06 19:42**

Hey folks,

Thanks for all the work on this: I really appreciate the efforts to keep Python as secure as possible!

This change _is_ causing us problems in the cloud-init codebase, which thankfully have been caught by our testing in Ubuntu's development release. This is in a fairly deep part of the codebase, so apologies in advance for the detailed description.

TL;DR: cloud-init constructs mirror URLs and then sanitises them by replacing invalid characters with hyphens. With the fix for this bug, urlsplit silently removes (some of) those characters before we can replace them, modifying the output of our sanitisation code, and therefore meaning cloud-init will, albeit in fairly specific corner cases, configure different mirrors if run with a Python including this fix vs. one that precedes it.

cloud-init constructs mirror URLs based on applying cloud metadata to user-configured (or default) templates. As we're responsible for constructing these URLs, we also sanitise them before configuring the package manager to use them: specifically, we urlsplit to get the hostname, IDNA-encode (to handle non-ASCII input), replace any invalid URL characters with a "-", and then strip "-" off each part of the hostname (to handle leading/trailing invalid characters), then recombine the URL. The most common case for this is a cloud which specifies values for the variables used in the template with an underscore: **http://my_openstack_region.cloud.archive.ubuntu.com/ubuntu** causes Apache mirrors with the default "HTTPProtocolOptions Strict" configuration to reject all requests to them (as that's an invalid hostname). In contrast, **http://my-openstack-region.cloud.archive.ubuntu.com/ubuntu** *is* accepted, so is preferable. (This is important because *.cloud.archive.ubuntu.com exists so that local cloud admins can DNS "hijack" subdomains of it to point at internal servers: even though the Ubuntu mirrors don't reject underscored domains (any longer), this is a landmine waiting for any admins running their own mirrors.) For more background, see the bug where we figured this all out: **https://bugs.launchpad.net/cloud-init/+bug/1868232**

So, more concretely: if we consider a post-templated URL of **http://my**\topenstack\tregion.mirror.internal/ubuntu, cloud-init changes from rewriting that to my-openstack-region.mirror.internal (on < 3.9.5) to myopenstackregion.mirror.internal (on 3.9.5+): if, in this notional deployment, an apt mirror is running at (exactly) my-openstack-region.mirror.internal, then new instance deployments will start failing: they won't be able to install packages. This is the sort of breakage that we aim to avoid in cloud-init (because you just _know_ that everyone who deployed this cloud left NotionalCorp years ago, so fixing the configuration to remove these obviously-incorrect tabs is not necessarily trivial).

Given the current state of the fix here, it's not clear to me how we could (cleanly) achieve our desired behaviour. We could perform replacement of these characters before invoking `urlsplit` but that would then substitute these characters outside of only the hostname: this is also a change in behaviour. We could substitute those characters with magic strings, perform the split, and then replace them in the non-hostname parts with the original character and in the hostname with hyphens: we've obviously left "cleanly" behind at this point. Another option would be to monkeypatch _UNSAFE_URL_BYTES_TO_REMOVE to an empty list: again, not a solution I'd want to have to support across Python versions!

One solution that presents itself to me: add a `strip_insecure_characters: bool = True` parameter. We, in cloud-init, would pass this in as `False`, knowing that we're going to handle those ourselves. Of course, this does leave the door open for API users to keep the current insecure behaviour: if library code (either public or project-internal) were to default to `False`, then the situation is no better than today.

For our use case, at least, I think a more restricted solution would work: `url_replacement_char: str = ""`. We'd call `urlsplit(..., url_replacement_char="-")` and the rest of our code would work as it does today: from its POV, there were never these invalid chars in the first place.

Thanks once again for the work (and apologies for the wall of text)!

Dan

msg393142 - (view) **Author: Daniel Watkins (odd_bloke) *** **Date: 2021-05-06 19:50**

(Accidentally dropped Ned from nosy list; apologies!)

msg393144 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2021-05-06 20:02**

We try to not add a new parameter in a bugfix release as that can be difficult to use. That said, adding a new bool keyword only parameter to control this behavior seems feasible.

Unfortunately you already have to deal with the existence of 3.9.5 having the new behavior but not having a control. (maybe you resort to the global behavior change there by monkeypatching the list? removing tab is probably enough as I doubt you rely on newlines in your scenario?)

Code wanting to support versions before this patch _and pass that new parameter_ winds up needing to resort to inspect.signature(), or a call with the parameter, catch the NameError, and retry calling without it pattern. Not unheard of. And so long as most code doesn't ever need to do that trick, is fine. (it can be checked for at module import time rather than doing it on every call to save overhead if that matters)

meta: Marking the issue as open while we decide if we'll be doing something here.

msg393146 - (view) **Author: Mike Lissner (Mike.Lissner)** **Date: 2021-05-06 20:36**

> With the fix for this bug, urlsplit silently removes (some of) those characters before we can replace them, modifying the output of our sanitisation code

I don't have any good solutions for 3.9.5, but going forward, this feels like another example of why we should just do parsing right (the way browsers do). That'd maintain tabs and whatnot in your output, and it'd fix the security issue by putting `java\nscript` into the scheme attribute instead of the path.

> One solution that presents itself to me: add a `strip_insecure_characters: bool = True` parameter.

Doesn't this lose sight of what this tool is supposed to do? It's not supposed to have a good (new, correct) and a bad (old, obsolete) way of parsing. Woe unto whoever has to write the documentation for that parameter.

Also, I should reiterate that these aren't "insecure" characters so if we did have a parameter for this, it'd be more like `do_rfc_3986_parsing` or maybe `do_naive_parsing`. The chars aren't insecure in themselves. They're fine. Python just gets tripped up on them.

msg393149 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2021-05-06 20:57**

Of note: If we had chosen to raise a ValueError (or similar) for these characters by default, the cloud-init code would also fail to behave as intended today (based on what I see in **https://github.com/canonical/cloud-init/commit/c478d0bff412c67280dfe8f08568de733f9425a1**)

Recommendation for cloud-init - Do your hostname transformation early using as simple as possible logic. By virtue of accepting (and encouraging?) invalid characters and transforming them, what you have today that you call urlsplit on is more of a url template, not really a url. something like this:

```
if m := re.search(r'^(?P<scheme_colon_slashes>[^/:]+://|)(?P<hostname>[^/]+)(?P<rest>/.*)', url_template):
    start, hostname, end = m.groups()
    for transformation in transformations:
        ... fixup hostname ...
    url = f'{start}{hostname}{end}'
else:
    ... # doesn't look like a URL template
```

yes this simplicity would allow your transformations to apply to the :port number. you could simplify further by including the scheme_colon_slashes in the part transformed. as your values are coming from user written config files, do you need to care about invalid characters in those transforming into invalid in the scheme or port number - characters in the resulting url anyways?

after that, do you even need urlsplit at all in your `_apply_hostname_transformations_to_url()` function?

msg393150 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2021-05-06 21:04**

FWIW, if we were to add a parameter, I'd lean towards a name of "invalid_url_characters = None" defaulting to using what's in our private _UNSAFE_URL_BYTES_TO_REMOVE global when None but otherwise letting the user specify a sequence of characters.

msg393198 - (view) **Author: Ned Deily (ned.deily) *** **Date: 2021-05-07 18:18**

> Unfortunately you already have to deal with the existence of 3.9.5 having the new behavior but not having a control.

I have been holding off on 3.7.x and 3.6.x security releases pending resolutions of this and other open security issues. But based on the most recent discussions, my take is that it would be a disservice to anyone still using 3.7 or 3.6 to release with the most recent "partial" fix (~~GH-25924~~ / ~~GH-25924~~) if it going to cause breakage. So, rather than delay further, unless there is a new resolution or someone has a very persuasive argument against it, I am going to revert those last two PRs from 3.7 and 3.6 for the upcoming releases pending a less intrusive fix.

msg393203 - (view) **Author: Senthil Kumaran (orsenthil) *** **Date: 2021-05-07 18:51**

Hello All,

I think, the current striping of ASCII newline and tab is a _reasonable_ solution given it was a security issue.

It also follows the guidelines of "WHATWG" (Specifically Point 3)

> 2. If input contains any ASCII tab or newline, validation error.
> 3. Remove all ASCII tab or newline from input.

And as per WHATWG, "A validation error does not mean that the parser terminates. Termination of a parser is always stated explicitly, e.g., through a return statement."

I agree that terms used in spec vs representing it with library code may not be 1:1, but we tried to stay close and followed the existing behavior of widely used clients.

This is a fix, per a security report, and per an evolv{ed,ing} standard recommendation. When dealing with security fixes, there could be simple or more complex migration involvements.

My reading of the previous message was, even if we raised exception or gave as a parameter, it wont be any better for certain downstream users, as we let the security problem open, and have it only as opt-in fix.

With respect to control

The comment in the review - **https://github.com/python/cpython/pull/25595#pullrequestreview-647122723** was to make these characters available in module level parameters, so that if users prefer to override, they could patch it.

so a revert may not be necessary for the reason of lack of control.

In short, at this moment, I still feel that is reasonable fix at this moment for the problem report, and intention to move closer to whatwg spec.

msg393205 - (view) **Author: Ned Deily (ned.deily) *** **Date: 2021-05-07 19:12**

> My reading of the previous message was, even if we raised exception
> or gave as a parameter, it wont be any better for certain downstream
> users, as we let the security problem open, and have it only as opt-in fix.

Senthil, I am not sure which previous message you are referring to but, with regards to my comment about revert the recent fixes for 3.7 and 3.6 until the reported problems are resolved, I should add that, given the recent input from downstream users about the side effects, the only way we *should* proceed with the current changes is by including more information in a What's New entry and the NEWS blurb about that the implications to users are of these changes.

msg393207 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2021-05-07 19:15**

There is no less intrusive fix as far as I can see. I believe we're down to either stick with what we've done, or do nothing. It doesn't have to be the same choice in all release branches, being more conservative with changes the older the stable branch is okay. (ie: removing this from 3.6 and 3.7 seems fine even if more recent ones do otherwise)

Based on my testing, raising an exception is more intrusive to existing tests (which we can only ever hope is representative of code) than stripping. At least as exposed by running the changes through many tens of thousands of unittest suites at work.

ie: if we raise an exception, pandas.read_json() starts failing because that winds up using urlsplit in hopes of extracting the scheme and comparing that to known values as their method of deciding if something should be treated as a URL to data rather than data. Pandas would need to be fixed.

That urlsplit() API use pattern is repeated in various other pieces of code: urlsplit is not expected to raise an exception. The caller then has a conditional or two testing some parts of the urlsplit result to make a guess as to if something should be considered a URL or not. Doing code inspection, pandas included, this code pretty much always then goes on to pass the original url value off to some other library, be it urllib, or requests, or ...).

Consequences of that code inspection finding? With our existing character stripping change, new data is then allowed to pass through these urlsplit uses and be considered a URL. Which leads to some code sending the url with embedded \r\n\t chars on to other APIs - a concern expressed a couple of times above.

Even though urlsplit isn't itself a validation API, it gets used as an early step in peoples custom identification and validation attempts. So *any* change we make to it at all in any way breaks someones expectations, even if they probably shouldn't have had those expectations and aren't doing wise validation.

Treat this analysis as a sign that we should provide an explicit url validator

because almost everyone is doing it some form of wrong. (**issue43883**)

I did wonder if Mike's suggestion of removing the characters during processing, but leaving them in the final result in **https://bugs.python.org/issue43882#msg393033** is feasible as remediation for this? My gut feeling is that it isn't. It doesn't solve the problem of preventing the bad data from going where it shouldn't. Even if we happen to parse that example differently, the unwanted characters are still retained in other places they don't belong. Fundamantelly: We require people to make a different series of API call and choices in the end user code to **explicitly not use unvalidated inputs**. Our stdlib API surface can't satisfy that today and use of unvalidated data in wrong places is a broad software security antipattern theme.

msg393211 - (view) **Author: Senthil Kumaran (orsenthil) *** **Date: 2021-05-07 19:37**

Ned wrote:

> Senthil, I am not sure which previous message you are referring to but.

I meant, the messages from other developers who raised that change broke certain test cases.

Ned, but I got little concerned, if we planned to revert the change.

> the only way we *should* proceed with the current changes is by including more information in a What's New entry and the NEWS blurb about that the implications to users are of these changes.

I agree with completely. I will include an additional blurb for this change for security fix versions.

Greg wrote:

> There is no less intrusive fix as far as I can see. I believe we're down to either stick with what we've done, or do nothing.

Exactly my feeling too.

> It doesn't have to be the same choice in all release branches, being more conservative with changes the older the stable branch is okay. (ie: removing this from 3.6 and 3.7 seems fine even if more recent ones do otherwise)

I hadn't considered that. But it wont save much will be my opinion. The users will have to upgrade to supported versions anyway and it will break then. The problem is only pushed a little.

So, keeping it consistent seems alright to me. It is a little additional for everyone, but we seem to be doing it.

msg393997 - (view) **Author: Ned Deily (ned.deily) *** **Date: 2021-05-20 02:11**

> I will include an additional blurb for this change for security fix versions.

Ping. This issue is still blocking 3.7 and 3.6 security releases.

msg394056 - (view) **Author: Ned Deily (ned.deily) *** **Date: 2021-05-20 20:15**

New changeset **c723d5191110f99849f7b0944820f6c3cd5f7747** by Senthil Kumaran in branch '3.7':
[3.7] ~~**bpo-43882**~~ - Mention urllib.parse changes in Whats New section for 3.7.11 (~~**GH-26267**~~)
https://github.com/python/cpython/commit/c723d5191110f99849f7b0944820f6c3cd5f7747

msg394057 - (view) **Author: Ned Deily (ned.deily) *** **Date: 2021-05-20 20:16**

New changeset **6f743e7a4da904f61dfa84cc7d7385e4dcc79ac5** by Senthil Kumaran in branch '3.6':
[3.6] ~~**bpo-43882**~~ - Mention urllib.parse changes in Whats New section for 3.6.14 (~~**GH-26268**~~)
https://github.com/python/cpython/commit/6f743e7a4da904f61dfa84cc7d7385e4dcc79ac5

msg394058 - (view) **Author: Ned Deily (ned.deily) *** **Date: 2021-05-20 20:18**

Thanks, Senthil and Greg! The updates for 3.7 and 3.6 are now merged. Is there anything else that needs to be done for this issue or can it now be closed?

msg394062 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2021-05-20 20:33**

Lets get equivalent whatsnew text into the 3.8 and 3.9 and 3.10 branches before closing it.

msg394112 - (view) **Author: Senthil Kumaran (orsenthil) *** **Date: 2021-05-21 12:29**

New changeset **f14015adf52014c2345522fe32d43f15f001c986** by Senthil Kumaran in branch '3.10':
[3.10] ~~**bpo-43882**~~ - Mention urllib.parse changes in Whats new section. (~~**GH-26275**~~)
https://github.com/python/cpython/commit/f14015adf52014c2345522fe32d43f15f001c986

msg394113 - (view) **Author: Senthil Kumaran (orsenthil) *** **Date: 2021-05-21 12:30**

New changeset **0593ae84af9e0e8332644e7ed13d7fd8306c4e1a** by Senthil Kumaran in branch '3.9':
[3.9] ~~**bpo-43882**~~ - Mention urllib.parse changes in Whats new section. (~~**GH-26276**~~)
https://github.com/python/cpython/commit/0593ae84af9e0e8332644e7ed13d7fd8306c4e1a

msg396628 - (view) **Author: Łukasz Langa (lukasz.langa) *** **Date: 2021-06-28 10:05**

New changeset **634da2de88af06eb8c6ebdb90d8c00005847063d** by Senthil Kumaran in branch '3.8':
[3.8] ~~**bpo-43882**~~ - Mention urllib.parse changes in Whats new section. (**#26277**)
https://github.com/python/cpython/commit/634da2de88af06eb8c6ebdb90d8c00005847063d

msg412688 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2022-02-06 23:39**

CVE-2022-0391 has been assigned to this vulnerability.

msg412705 - (view) **Author: Mike Lissner (Mike.Lissner)** **Date: 2022-02-07 03:06**

Looks like that CVE isn't public yet.

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-0391

Any chance I can get access (I originally reported this vuln.). My email is **mike@free.law**, if it's possible and my email is needed.

Thanks!

msg412821 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2022-02-08 08:44**

> Looks like that CVE isn't public yet.
> **https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-0391**
> Any chance I can get access (I originally reported this vuln.).

Message from Gaurav Kamathe who requested the CVE:

"We've sent a request to MITRE to get this published and it'll be available on MITRE shortly."

History

Date	User	Action	Args
2022-04-11 14:59:44	admin	set	github: 88048
2022-02-09 11:40:17	felixxm	set	nosy: + **felixxm**
2022-02-08 08:44:47	vstinner	set	messages: + **msg412821**
2022-02-07 03:06:06	Mike.Lissner	set	messages: + **msg412705**
2022-02-06 23:39:40	vstinner	set	nosy: + **vstinner** messages: + **msg412688** title: [security] urllib.parse should sanitize urls containing ASCII newline and tabs. -> [security] CVE-2022-0391: urllib.parse should sanitize urls containing ASCII newline and tabs.
2021-06-28 10:05:35	lukasz.langa	set	messages: + **msg396628**
2021-06-02 01:26:09	gregory.p.smith	set	status: open -> closed resolution: fixed stage: patch review -> commit review
2021-05-21 12:30:08	orsenthil	set	messages: + **msg394113**
2021-05-21 12:29:36	orsenthil	set	messages: + **msg394112**
2021-05-21 01:40:01	orsenthil	set	pull_requests: + **pull_request24883**

2021-05-21 01:33:44	orsenthil	set	pull_requests: + **pull_request24882**
2021-05-21 00:51:10	orsenthil	set	stage: commit review -> patch review pull_requests: + **pull_request24881**
2021-05-20 20:33:55	gregory.p.smith	set	messages: + **msg394062**
2021-05-20 20:18:58	ned.deily	set	priority: release blocker -> high resolution: fixed -> (no value) messages: + **msg394058** stage: patch review -> commit review
2021-05-20 20:16:19	ned.deily	set	messages: + **msg394057**
2021-05-20 20:15:09	ned.deily	set	messages: + **msg394056**
2021-05-20 14:46:31	orsenthil	set	pull_requests: + **pull_request24872**
2021-05-20 14:10:59	orsenthil	set	stage: resolved -> patch review pull_requests: + **pull_request24871**
2021-05-20 02:11:52	ned.deily	set	priority: normal -> release blocker nosy: + **pablogsal** messages: + **msg393997**
2021-05-07 19:37:59	orsenthil	set	messages: + **msg393211**
2021-05-07 19:15:24	gregory.p.smith	set	messages: + **msg393207**
2021-05-07 19:12:23	ned.deily	set	messages: + **msg393205**
2021-05-07 18:51:59	orsenthil	set	messages: + **msg393203**
2021-05-07 18:18:36	ned.deily	set	messages: + **msg393198**
2021-05-06 21:04:43	gregory.p.smith	set	messages: + **msg393150**
2021-05-06 20:57:20	gregory.p.smith	set	messages: + **msg393149**
2021-05-06 20:36:08	Mike.Lissner	set	messages: + **msg393146**
2021-05-06 20:02:18	gregory.p.smith	set	status: closed -> open messages: + **msg393144**
2021-05-06 19:50:34	odd_bloke	set	nosy: + **ned.deily** messages: + **msg393142**
2021-05-06 19:42:25	odd_bloke	set	nosy: + **odd_bloke**, - **ned.deily** messages: + **msg393139**
2021-05-06 19:14:58	gregory.p.smith	set	messages: + **msg393136**
2021-05-06 17:11:39	ned.deily	set	status: open -> closed resolution: fixed stage: patch review -> resolved

2021-05-06 16:56:08	ned.deily	set	messages: + **msg393108**
2021-05-06 16:52:46	ned.deily	set	nosy: + **ned.deily** messages: + **msg393107**
2021-05-05 23:04:46	orsenthil	set	messages: + **msg393049**
2021-05-05 22:50:13	miss-islington	set	pull_requests: + **pull_request24603**
2021-05-05 20:26:06	gregory.p.smith	set	messages: + **msg393039**
2021-05-05 18:36:23	Mike.Lissner	set	messages: + **msg393034**
2021-05-05 18:35:05	Mike.Lissner	set	messages: + **msg393033**
2021-05-05 18:13:45	gregory.p.smith	set	messages: + **msg393030**
2021-05-05 17:25:51	miss-islington	set	pull_requests: + **pull_request24591**
2021-05-05 17:25:46	miss-islington	set	pull_requests: + **pull_request24590**
2021-05-05 17:25:37	lukasz.langa	set	messages: + **msg393025**
2021-05-05 17:00:32	orsenthil	set	pull_requests: + **pull_request24589**
2021-05-05 13:52:08	apollo13	set	nosy: + **apollo13** messages: + **msg393009**
2021-05-05 09:11:34	mgorny	set	messages: + **msg392995**
2021-05-05 02:19:35	gregory.p.smith	set	messages: + **msg392971** versions: + Python 3.11
2021-05-04 20:16:12	Mike.Lissner	set	messages: + **msg392944**
2021-05-04 17:26:45	sethmlarson	set	nosy: + **sethmlarson** messages: + **msg392926**
2021-05-04 10:57:39	mgorny	set	nosy: + **mgorny** messages: + **msg392873**
2021-05-03 19:09:07	orsenthil	set	messages: + **msg392835**
2021-05-03 14:12:08	orsenthil	set	messages: + **msg392808**
2021-05-03 14:08:59	orsenthil	set	pull_requests: + **pull_request24537**
2021-05-03 09:10:34	lukasz.langa	set	nosy: + **lukasz.langa** messages: + **msg392781**
2021-05-01 17:26:20	gregory.p.smith	set	messages: + **msg392611**
2021-04-29 17:57:46	orsenthil	set	messages: + **msg392338**
2021-04-29 17:17:30	miss-islington	set	pull_requests: + **pull_request24420**
2021-04-29 17:17:24	miss-islington	set	pull_requests: + **pull_request24419**

2021-04-29 17:17:18	miss-islington	set	pull_requests: + **pull_request24418**
2021-04-29 17:17:12	miss-islington	set	nosy: + **miss-islington** pull_requests: + **pull_request24417** stage: needs patch -> patch review
2021-04-29 17:16:55	orsenthil	set	messages: + **msg392334**
2021-04-27 14:21:46	vstinner	set	nosy: - **vstinner**
2021-04-25 14:53:44	orsenthil	set	messages: + **msg391859** stage: patch review -> needs patch
2021-04-25 14:49:59	orsenthil	set	keywords: + **patch** stage: needs patch -> patch review pull_requests: + **pull_request24315**
2021-04-20 10:41:07	vstinner	set	messages: + **msg391426**
2021-04-20 09:20:43	vstinner	set	components: + Library (Lib)
2021-04-20 09:20:34	vstinner	set	title: urllib.parse should sanitize urls containing ASCII newline and tabs. -> [security] urllib.parse should sanitize urls containing ASCII newline and tabs.
2021-04-19 20:24:54	Mike.Lissner	set	nosy: + **Mike.Lissner**
2021-04-19 03:24:07	xtreak	set	nosy: + **gregory.p.smith**, **vstinner**, **xtreak** messages: + **msg391352**
2021-04-18 19:37:00	orsenthil	create	