» Issue30458



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/74643

classification

Title:	[security][CVE-2019-9740][CVE-2019-9947] HTTP Header Injection (follow-up of CVE-2016-5699)		
Type:	security	**Stage**:	resolved
Components:	Library (Lib)	**Versions**:	Python 3.9, Python 3.8, Python 3.7, Python 3.6, Python 3.5, Python 2.7

process

Status:	closed	**Resolution**:	fixed
Dependencies:		**Superseder**:	
Assigned To:		**Nosy List**:	benjamin.peterson, cstratak, gregory.p.smith, jaraco, larry, lukasz.langa, martin.panter, miss-islington, ned.deily, orange, rschiron, serhiy.storchaka, vstinner, ware, xiang.zhang, xtreak
Priority:	deferred blocker	**Keywords**:	patch, security_issue

Created on **2017-05-24 15:01** by **orange**, last changed **2022-04-11 14:58** by **admin**. This issue is now **closed**.

Pull Requests

URL	Status	Linked	Edit
PR 12755	merged	gregory.p.smith, 2019-04-10 09:11	
PR 13032	merged	xtreak, 2019-05-01 06:50	
PR 13044	merged	gregory.p.smith, 2019-05-01 20:10	
PR 13154	merged	hroncok, 2019-05-07 14:32	
PR 13155	merged	hroncok, 2019-05-07 14:50	
PR 13207	merged	hroncok, 2019-05-08 18:41	
PR 13315	merged	vstinner, 2019-05-14 15:04	
PR 12524	closed	push0ebp, 2019-05-29 12:14	
PR 11768	closed	matrixise, 2019-05-29 12:15	
PR 13771	closed	python-dev, 2019-06-03 06:15	
PR 16321	closed	jaraco, 2019-09-21 08:55	

Messages (55)

msg294360 - (view) **Author: Orange (orange)** **Date: 2017-05-24 15:01**

Hi, the patch in CVE-2016-5699 can be broke by an addition space.
http://www.cvedetails.com/cve/CVE-2016-5699/
https://hg.python.org/cpython/rev/bf3e1c9b80e9
https://hg.python.org/cpython/rev/1c45047c5102

```
import urllib, urllib2

urllib.urlopen('http://127.0.0.1\r\n\x20hihi\r\n :11211')
urllib2.urlopen('http://127.0.0.1\r\n\x20hihi\r\n :11211')
```

msg295026 - (view) **Author: Xiang Zhang (xiang.zhang) *** **Date: 2017-06-02 15:36**

Looking at the code and the previous issue ~~#22928~~, CRLF immediately followed by a tab or space (obs-fold: CRLF 1*(SP / HTAB)) is a valid part of a header value so the regex deliberately ignore them.

So it looks right to me the url given doesn't raise the same exception as the url without spaces, though the given url seems malformed.

msg295067 - (view) **Author: Martin Panter (martin.panter) *** **Date: 2017-06-03 07:01**

You can also inject proper HTTP header fields (or do multiple requests) if you omit the space after the CRLF:

```
urlopen("http://localhost:8000/ HTTP/1.1\r\nHEADER: INJECTED\r\nIgnore:")

Data sent to the server:
>>> server = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)
>>> server.bind(("localhost", 8000))
>>> server.listen()
>>> [conn, addr] = server.accept()
>>> pprint(conn.recv(300).splitlines(keepends=True))
[b'GET / HTTP/1.1\r\n',
 b'HEADER: INJECTED\r\n',
 b'Ignore: HTTP/1.1\r\n',
 b'Accept-Encoding: identity\r\n',
 b'User-Agent: Python-urllib/3.5\r\n',
 b'Connection: close\r\n',
 b'Host: localhost:8000\r\n',
 b'\r\n']
```

Issue 14826 is already open about how “urlopen” handles spaces, and there is a patch in **Issue 13359** that proposes to also encode newline characters. But if the CRLF or header injection is a security problem, then 2.7 etc could be changed to raise an exception (like ~~**Issue 22928**~~), or to do percent encoding.

msg306981 - (view) **Author: Martin Panter (martin.panter) *** **Date: 2017-11-26 01:04**

Actually, the CRLF + space can be injected via percent encoding, so just dealing with literal CRLFs and spaces wouldn’t be enough. You would have to validate the hostname after it is decoded.

```
urlopen("http://127.0.0.1%0D%0A%20SLAVEOF . . . :6379/")

>>> pprint(conn.recv(300).splitlines(keepends=True))
[b'GET / HTTP/1.1\r\n',
 b'Accept-Encoding: identity\r\n',
 b'Host: 127.0.0.1\r\n',
 b' SLAVEOF . . . :6379\r\n',
 b'Connection: close\r\n',
 b'User-Agent: Python-urllib/2.7\r\n',
 b'\r\n']
```

msg337970 - (view) **Author: Karthikeyan Singaravelan (xtreak) *** **Date: 2019-03-15 06:15**

See also **https://bugs.python.org/issue36276** for a similar report. I think it's better to raise an error instead of encoding CRLF characters in URL similar to headers.

I feel either of the issue and more preferably ~~**issue36276**~~ closed as a duplicate of this one. Copy of **msg337968** with reference to details about similar report in golang :

For reference an exact report on golang repo : **https://github.com/golang/go/issues/30794** . This seemed to have been fixed in latest golang release 1.12 and commit **https://github.com/golang/go/commit/829c5df58694b3345cb5ea41206783c8ccf5c3ca** . The commit introduces a check for CTL characters and throws an error for URLs something similar to Python does for headers now at **bf3e1c9b80e9**.

```
func isCTL(r rune) bool {
        return r < ' ' || 0x7f <= r && r <= 0x9f
}

if strings.IndexFunc(ruri, isCTL) != -1 {
        return errors.New("net/http: can't write control character in Request.URL")
}
```

So below program used to work before go 1.12 setting a key on Redis but now it throws error :

```
package main

import "fmt"
import "net/http"

func main() {
        resp, err := http.Get("http://127.0.0.1:6379?\r\nSET test
failure12\r\n:8080/test/?test=a")
        fmt.Println(resp)
        fmt.Println(err)
}


➜  go version
go version go1.12 darwin/amd64
➜  go run urllib_vulnerability.go
<nil>
```

parse **http://127.0.0.1:6379**?
SET test failure12
:8080/test/?test=a: net/url: invalid control character in URL

Looking more into the commit there seemed to be a solution towards escaping characters with **https://github.com/golang/go/issues/22907** . The fix seemed to have broke Google's internal tests [0] and hence reverted to have the above commit where only CTL characters were checked and raises an error. I think this is a tricky bug upon reading code reviews in the golang repo that has around 2-3 reports with a fix committed to be reverted later for a more conservative fix and the issue was reopened to target go 1.13 .

Thanks a lot for the report @ragdoll.guo

[0] **https://go-review.googlesource.com/c/go/+/159157/2#message-39c6be13a192bf760f6318ac641b432a6ab8fdc8**

msg339754 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-04-09 14:28**

The CVE-2019-9740 has been assigned to the ~~**bpo-36276**~~:

* **https://nvd.nist.gov/vuln/detail/CVE-2019-9740**
* **https://bugzilla.redhat.com/show_bug.cgi?id=1692984**

... which has been marked as a duplicate of this issue.

msg339840 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2019-04-10 09:22**

Martin claimed "Actually, the CRLF + space can be injected via percent encoding"

I am unable to reproduce that behavior using urllib.request.urlopen() or urllib.request.URLopener.open() in my master/3.8 tree.

msg339846 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-04-10 10:36**

Oh, I didn't recall that this issue (this class of security vulnerabilities) has a so old history. I found *A LOT* of similar open issues. Here are my notes. Maybe most open issues should be closed as duplicate of this one to clarify the status of urllib in Python? :-)

Emails:

* 2019: **https://mail.python.org/pipermail/python-dev/2019-April/157014.html**
* 2017: **https://mail.python.org/pipermail/python-dev/2017-July/148699.html**

Open issues:

* 2011, **bpo-13359**: "urllib2 doesn't escape spaces in http requests"
  Not marked as a security issue.
* 2012, **bpo-14826**: "urlopen URL with unescaped space"
  Fix using quote(self.__original, safe="%/:=&?~#+!$,;'@()*[]|")
  ... and the changed has then be reverted because it broke buildbots.
  Still open.
* 2013, **bpo-17322**: "urllib.request add_header() currently allows trailing spaces (and other weird stuff)"
  Not marked as a security issue.

* 2014, ~~bpo-22928~~: "HTTP header injection in urrlib2/urllib/httplib/http.client (CVE-2016-5699)"
  Marked as fixed, but user Orange explained in the first comment of in
  ~~bpo-30458~~ that the fix is incomplete.
* 2017, ~~bpo-30458~~: "[CVE-2019-9740][security] CRLF Injection in httplib" (this issue)
* 2017, ~~bpo-32085~~: "[Security] A New Era of SSRF - Exploiting URL Parser in Trending Programming Languages!"
* 2019, ~~bpo-35906~~: "[CVE-2019-9947] Header Injection in urllib" (another CVE!)

Closed issues:

* 2004, ~~bpo-918368~~: "urllib doesn't correct server returned urls" (urllib)
  FIXED BY: commit **7c2867fcb1ade429a41e030585332ea26e3f60e1**
  Fix: fullurl = quote(fullurl, safe="%/:=&?~#+!$,;'@()*[]")
* 2005, ~~bpo-1353433~~: "Http redirection error in urllib2.py" (urllib2)
  FIXED BY: commit **ddb84d7c69addc5d5e2ab3e327260d97b52af3a7**
  Fix: newurl = newurl.replace(' ', '%20')
* 2005, ~~bpo-1153027~~: "http_error_302() crashes with 'HTTP/1.1 400 Bad Request"
  FIXED BY: commit **690ce9b353bc0a86d0886470adbaa50e813de3b8** (**Lib/urllib/request.py**)
  Fix: fullurl = quote(fullurl, safe="%/:=&?~#+!$,;'@()*[]")
* ~~bpo-29606~~: "urllib FTP protocol stream injection"
  Duplicate of ~~bpo-30119~~.
* ~~bpo-30119~~: "(ftplib) A remote attacker could possibly attack by containing the newline characters"
  FIXED BY: commmit **8c2d4cf092c5f0335e7982392a33927579c4d512**
  Fix: reject "\r" and "\n" in FTP.putline() (**Lib/ftplib.py**)
* ~~bpo-36276~~: "[CVE-2019-9740] Python urllib CRLF injection vulnerability"
  Closed as duplicate of ~~bpo-30458~~

Rejected pull requests:

* **https://github.com/python/cpython/pull/1216/files**
  ~~bpo-29606~~: Reject "\n" in ftp_open() of **Lib/urllib/request.py**
* **https://github.com/python/cpython/pull/2800/files**
  ~~bpo-29606~~: Reject "\n" in ftp_open() and open_ftp() of **Lib/urllib/request.py**
* **https://github.com/python/cpython/pull/2301/files**
  ~~bpo-30713~~: The splittype(), splitport() and splithost() functions of the urllib.parse module now reject URLs which contain a newline character.
* **https://github.com/python/cpython/pull/2303/files**
  ~~bpo-30713~~: The splittype(), splitport() and splithost() functions of the urllib.parse module now reject URLs which contain a newline character, but splittype() accepts newlines after the type.

msg339848 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-04-10 10:43**

> * 2019, ~~bpo-35906~~: "[CVE-2019-9947] Header Injection in urllib" (another CVE!)

Gregory P. Smith just marked ~~bpo-35906~~ as a duplicate of this issue. Copy of his **msg339842**:

"""
my fix proposed in ~~issue30458~~ fixes this issue.

i do not think this one deserved its own CVE; at least **https://nvd.nist.gov/vuln/detail/CVE-2019-9947**'s current text also points to the

other one.
"""

Until the status of CVE-2019-9947 is clarified, I added CVE-2019-9947 in the title of this issue to help to better track all CVEs :-)

Did someone contact the CVE organization to do something with CVE-2019-9947?

msg339850 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-04-10 10:45**

> The CVE-2019-9740 has been assigned to the ~~**bpo-36276**~~

I don't know how CVE are assigned. Since this issue started with "the patch in CVE-2016-5699 can be broke by an addition space", would it make sense to reuse CVE-2016-5699 rather than using a new CVE?

msg339851 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-04-10 10:48**

> Closed issues:

I forgot:

* 2017, ~~**bpo-30713**~~: "Reject newline character (U+000A) in URLs in urllib.parse"
  Rejected: the ftplib vulnerabilty has been fixed by ~~**bpo-30119**~~
  with commmit **8c2d4cf092c5f0335e7982392a33927579c4d512**.

msg339852 - (view) **Author: Karthikeyan Singaravelan (xtreak) *** **Date: 2019-04-10 10:59**

As @gregory.p.smith noted in GitHub [0] this fixes only protocol level bugs. There are some parsing ambiguities in urllib that are potential security issues still to be fixed.

issue20271 - urllib.urlparse('**http://benign.com**\[attacker.com]') returns attacker.com as hostname . A slightly related issue **https://bugs.python.org/issue20271**
issue35748 - urllib.urlparse(r'**http://spam**\eggs!cheese&**aardvark@evil.com**') returns evil.com as hostname
issue23505 - Urlparse insufficient validation leads to open redirect
issue33661 - urllib may leak sensitive HTTP headers to a third-party web site (Redirecting from https to http might also pass some headers in plain text. This behavior was changed in requests, golang, Curl that had their own respective CVEs)

As a fun side note this vulnerability was used by one of our own tests as a feature from 2012 to test another security issue (~~**issue14001**~~) [1] :)

[0] **https://github.com/python/cpython/pull/12755#issuecomment-481599611**
[1] **https://github.com/python/cpython/pull/12755#issuecomment-481618741**

msg339853 - (view) **Author: Martin Panter (martin.panter) *** **Date: 2019-04-10 11:28**

Gregory, I haven’t tried recent Python code, but I expect the problem with percent decoding is still there. If you did try my example, what results did you see? Be aware that these techniques only work if the OS co-operates and connects to localhost when you give it the longer host string. At the moment I have glibc 2.26 on x86-64 Linux.

```
In the Python 3 master branch, the percent-encoding should be decoded in
“urllib.request.Request._parse”:

def _parse(self):
    ...
    self.host, self.selector = _splithost(rest)
    if self.host:
        self.host = unquote(self.host)

Then in “AbstractHTTPHandler.do_request_” the decoded host string becomes the
“Host” header field value, without any encoding:

def do_request_(self, request):
    host = request.host
    ...
    sel_host = host
    ...
    if not request.has_header('Host'):
        request.add_unredirected_header('Host', sel_host)

Perhaps one solution to both my version and Orange’s original version is to encode
the “Host” header field value properly. This might also apply to the “http.client”
code.
```

msg339857 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-04-10 12:32**

~~**bpo-36276**~~ has been marked as a duplicate of this issue.

According to the following message, urllib3 is also vulnerable to HTTP Header Injection:
https://bugs.python.org/issue36276#msg337837

Copy of Alvin Chang's **msg337837**:

```
"""
I am also seeing the same issue with urllib3

import urllib3

pool_manager = urllib3.PoolManager()

host = "localhost:7777?a=1 HTTP/1.1\r\nX-injected: header\r\nTEST: 123"
url = "http://" + host + ":8080/test/?test=a"

try:
    info = pool_manager.request('GET', url).info()
    print(info)
except Exception:
    pass

nc -l localhost 7777
GET /?a=1 HTTP/1.1
X-injected: header
TEST: 123:8080/test/?test=a HTTP/1.1
Host: localhost:7777
```

```
Accept-Encoding: identity
"""
```

msg339858 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-04-10 12:35**

```
> According to the following message, urllib3 is also vulnerable to HTTP Header
Injection: (...)

And the issue has been reported to urllib3:
https://github.com/urllib3/urllib3/issues/1553

Copy of the first message:

"""
At https://bugs.python.org/issue36276 there's an issue in Python's urllib that an
attacker controlling the request parameter can inject headers by injecting CR/LF
chars.

A commenter mentions that the same bug is present in urllib3:
https://bugs.python.org/issue36276#msg337837

So reporting it here to make sure it gets attention.
"""
```

msg339861 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-04-10 13:03**

```
Since this issue has a long history and previously attempts to fix it failed, it
seems like the Internet is a black or white world, more like a scale of gray...
*Maybe* we need to provide a way to allow to pass junk characters in an URL?
(disable URL validation)

Idea: add an optional parameter to urllib, httplib, maybe also ftplib, to allow
arbitrary "invalid" URLs / FTP commands. It would be a parameter *per request*, not
a global option.

I don't propose to have a global configuration option like an environment variable,
urllib attribute or something else. A global option would be hard to control and
would impact just too much code.

My PEP 433 has been rejected because of the sys.setdefaultcloexec(cloexec: bool)
function which allowed to change globally the behavior of Python. The PEP 446 has
been accepted with no *global* option to opt-in for the old behavior, but only
"local" *per file descriptor*: os.set_inheritable(fd, inheritable).
```

msg339884 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2019-04-10 19:31**

```
> *Maybe* we need to provide a way to allow to pass junk characters in an URL?
(disable URL validation)

We should not do this in our http protocol stack code.  Anyone who _wants_ that is
already intentionally violating the http protocol which defeats the entire purpose
of our library and the parameter named "url".

Will this break something in the world other than our own test_xmlrpc test?
Probably.  Do they have a right to complain about it?  Not one we need listen to.
Such code is doing something that was clearly an abuse of the API.  The parameter
```

was named url not raw_data_to_stuff_subversively_into_the_binary_protocol. Its intent was clear.

msg339894 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-04-10 22:05**

> Will this break something in the world other than our own test_xmlrpc test?
Probably. Do they have a right to complain about it? Not one we need listen to.

I understand. But. Can we consider that for old Python versions like Python 2.7 and 3.5?

This change will be applied to all supported Python versions.

I recall that when Python 2.7 started to validate TLS certificate, the change broke some applications. Are these applications badly written? Yes! But well, "it worked well before". Sometimes, when you work in a private network, the security matters less, whereas it might be very expensive to fix a legacy application. At Red Hat, we developed a solution to let customers to opt-out from this fix (to no validate TLS certificates), because it is just too expensive for customers to fix their legacy code but they would like to be able to upgrade RHEL.

One option to not validate URLs is to downgrade Python, but I'm not sure that it's the best compromise :-/

msg340405 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-04-17 15:32**

It seems like a change has been pushed into urllib3 to fix this issue, but that there is an issue with international URLs and that maybe RFC 3986 should be updated.

RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax" (January 2005)
https://www.ietf.org/rfc/rfc3986.txt

"Without **~~#1531~~** or IRI support in rfc3986 releasing master in it's current state will break backwards compatibility with international URLs."

https://github.com/urllib3/urllib3/issues/1553#issuecomment-474046652

=> where 1531 means **https://github.com/urllib3/urllib3/pull/1531**

"wave Hi! I've noticed that CVE-2019-11236 has been assigned to the CRLF injection issue described here. It seems that the library has been patched in GitHub, but no new release has been made to pypi. Will a new release containing the fix be made to pypi soon? Based on @theacodes comment it seems like a release was going to be made, but I also see her status has her perhaps unavailable. Is someone else perhaps able to cut a new release into pypi?"

https://github.com/urllib3/urllib3/issues/1553#issuecomment-484113222

msg340407 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-04-17 15:35**

"wave Hi! I've noticed that CVE-2019-11236 has been assigned to the CRLF injection issue described here. It seems that the library has been patched in GitHub, but no new release has been made to pypi. (...)"

This urllib3 change:

https://github.com/urllib3/urllib3/commit/0aa3e24fcd75f1bb59ab159e9f8adb44055b2271

urllib3 now vendors a copy of the rfc3986 library:

https://pypi.org/project/rfc3986/

msg340408 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-04-17 15:40**

> urllib3 now vendors a copy of the rfc3986 library:
> **https://pypi.org/project/rfc3986/**

There are multiple Python projects to validate URI:

* **https://github.com/python-hyper/rfc3986/** -> **https://pypi.org/project/rfc3986/**
* **https://github.com/dgerber/rfc3987** -> **https://pypi.org/project/rfc3987/** (the name is confusing: the library implements the RFC 3986, not the RFC 3987)
* **https://github.com/tkem/uritools/** -> **https://pypi.org/project/uritools/**

msg341174 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2019-05-01 02:12**

New changeset **c4e671eec20dfcb29b18596a89ef075f826c9f96** by Gregory P. Smith in branch 'master':
~~**bpo-30458**~~: Disallow control chars in http URLs. (~~**GH-12755**~~)
https://github.com/python/cpython/commit/c4e671eec20dfcb29b18596a89ef075f826c9f96

msg341175 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2019-05-01 02:18**

backports to older releases will need to be done manually and take care depending on how much of a concern tightening the existing abusive lenient behavior of the http.client API to enforce what characters are allowed in URLs is to stable releases.

I question if this is _really_ worthy of a "security" tag and a CVE (thus its non-high ranking)... it is a bug in the calling program if it blindly uses untrusted data as a URL. What this issue addresses is that we catch that more often and raise an error; a good thing to do for sure, but the stdlib should be the last line of defense.

msg341176 - (view) **Author: Karthikeyan Singaravelan (xtreak) *** **Date: 2019-05-01 03:30**

This causes buildbot failure (AMD64 FreeBSD 10-STABLE Non-Debug 3.x and AMD64 Debian root 3.x). I tried debugging and it's reproducible on my mac machine that has python not built with ssl and not reproducible on Ubuntu machine built with ssl.

The failed tests use https scheme and as I can see from the file there is one another test (test_cafile_and_context) which is skipped and has skip test if ssl is absent @unittest.skipUnless(ssl, "ssl module required"). It seems perhaps wrapping these machines don't have ssl built with skip test might help. Since primary CI has ssl built it would have been caught.

On trying to add a print statement for lookup variable at **https://github.com/python/cpython/blob/c4e671eec20dfcb29b18596a89ef075f826c9f96/Lib/urllib/request.py#L485** I can see the below output where httpshandler was not defined for machines without built ssl. HTTPSConnection was not present as part of http.client due to import ssl causing ImportError.

```
Ubuntu with ssl

{'unknown': [<urllib.request.UnknownHandler object at 0x7f855d2fe7e8>], 'http':
[<urllib.request.HTTPHandler object at 0x7f855d2fe878>], 'ftp':
[<urllib.request.FTPHandler object at 0x7f855d2fe6c8>], 'file':
[<urllib.request.FileHandler object at 0x7f855d2fe908>], 'data':
[<urllib.request.DataHandler object at 0x7f855d2fe998>], 'https':
[<urllib.request.HTTPSHandler object at 0x7f855d2fea28>]}

Mac without ssl (https handler missing causing unknown to be taken up for the test)

{'unknown': [<urllib.request.UnknownHandler object at 0x108824c68>], 'http':
[<urllib.request.HTTPHandler object at 0x108824cb0>], 'ftp':
[<urllib.request.FTPHandler object at 0x108824cf8>], 'file':
[<urllib.request.FileHandler object at 0x108824d88>], 'data':
[<urllib.request.DataHandler object at 0x108824e60>]}

Gregory, I can create a PR with below patch if my analysis right to see if it helps
or you can try a buildbot-custom branch to see if this works with buildbots since
my PR won't have any effect on primary CI which have ssl built version of Python. I
am not sure I have privileges to trigger a custom buildbot run. I tested it on my
Mac without ssl and it has no failures since the tests are skipped.

diff --git a/Lib/test/test_urllib.py b/Lib/test/test_urllib.py
index e87c85b928..c5b23f935b 100644
--- a/Lib/test/test_urllib.py
+++ b/Lib/test/test_urllib.py
@@ -329,6 +329,7 @@ class urlopen_HttpTests(unittest.TestCase, FakeHTTPMixin,
FakeFTPMixin):
         finally:
             self.unfakehttp()

+    @unittest.skipUnless(ssl, "ssl module required")
     def test_url_with_control_char_rejected(self):
         for char_no in list(range(0, 0x21)) + [0x7f]:
             char = chr(char_no)
@@ -354,6 +355,7 @@ class urlopen_HttpTests(unittest.TestCase, FakeHTTPMixin,
FakeFTPMixin):
             finally:
                 self.unfakehttp()

+    @unittest.skipUnless(ssl, "ssl module required")
     def test_url_with_newline_header_injection_rejected(self):
         self.fakehttp(b"HTTP/1.1 200 OK\r\n\r\nHello.")
         host = "localhost:7777?a=1 HTTP/1.1\r\nX-injected: header\r\nTEST: 123"
```

msg341178 - (view) **Author: Karthikeyan Singaravelan (xtreak) *** **Date: 2019-05-01 03:59**

```
Sorry, I will toggle back the issue status. Not sure why bpo didn't warn in this
case.
```

msg341192 - (view) **Author: miss-islington (miss-islington)** **Date: 2019-05-01 12:00**

```
New changeset 2fc936ed24cf04ed32f6015a8aa78c8ea40da66b by Miss Islington (bot)
(Xtreak) in branch 'master':
```

~~bpo-30458~~: Disable https related urllib tests on a build without ssl (~~GH-13032~~)
https://github.com/python/cpython/commit/2fc936ed24cf04ed32f6015a8aa78c8ea40da66b

msg341234 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2019-05-01 20:39**

New changeset b7378d77289c911ca6a0c0afaf513879002df7d5 by Gregory P. Smith in branch 'master':
~~bpo-30458~~: Use InvalidURL instead of ValueError. (~~GH-13044~~)
https://github.com/python/cpython/commit/b7378d77289c911ca6a0c0afaf513879002df7d5

msg341286 - (view) **Author: Karthikeyan Singaravelan (xtreak) *** **Date: 2019-05-02 16:58**

IMO it does qualify as a security issue. In case of urllib to be lenient and can be exploited it's good to document like tarfile and xml modules that have a warning about untrusted data potentially causing issues and perhaps link to a url validator that adheres to RFC in pypi. I would expect stdlib to handle this but in case it's not handled due to backwards compatibility and potential regressions a warning could be made about the same in the docs noting down the responsibility of the functions and that they are not always safe against malicious data.

msg341290 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2019-05-02 17:35**

A note from the urllib3 fixes to this: They chose to go the route of auto-%-encoding the offending characters in URLs instead. I do not think the stdlib should do this.

One thing to note though is that they claim URLs with spaces embedded in them are apparently somewhat common in the world, we might want to relax our check to not include space (\x20) in the rejected characters for that reason.

A space alone cannot be used for injection. Someone could append an incorrect HTTP protocol version to a request using it " HTTP/1.0" but that would be followed by the actual " HTTP/x.y" generated by our library which at that point is up to the server to parse and or reject as odd. Without the ability to inject \r\n the headers to go with the protocol cannot be modified; so a change in protocol version could at most alter how some headers may be treated. Worst case: they upgrade/downgrade the http version in a non-pedantic server - i believe this to be low impact (feel free to prove me wrong with a working example against a common server). Best case: The server rejects the unparseable request or considers their " HTTP/1.0" to be part of their URL path.

In a world where unescaped spaces in URLs are common, some servers _might_ already take the strategy of splitting only on the first and last spaces in the request line anyways, considering everything in the middle to be the url with embedded spaces.

msg341291 - (view) **Author: Karthikeyan Singaravelan (xtreak) *** **Date: 2019-05-02 17:47**

> One thing to note though is that they claim URLs with spaces embedded in them are apparently somewhat common in the world, we might want to relax our check to not include space (\x20) in the rejected characters for that reason.

Guess I missed it in the PR discussion and read your comment [0] now about the change from golang's fix that excluded space as a problematic character. Is it worth documenting this change somewhere like a versionchanged directive in http.client?

[0] **https://github.com/python/cpython/pull/12755#discussion_r279888496**

Thanks for the details.

msg341724 - (view) **Author: Miro Hrončok (hroncok) *** **Date: 2019-05-07 13:42**

I'll work on 3.7 backport.

msg341750 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2019-05-07 15:28**

New changeset **7e200e0763f5b71c199aaf98bd5588f291585619** by Gregory P. Smith (Miro Hrončok) in branch '3.7':
~~**bpo-30458**~~: Disallow control chars in http URLs. (~~**GH-12755**~~) (~~**GH-13154**~~)
https://github.com/python/cpython/commit/7e200e0763f5b71c199aaf98bd5588f291585619

msg341906 - (view) **Author: Ned Deily (ned.deily) *** **Date: 2019-05-08 16:33**

New changeset **c50d437e942d4c4c45c8cd76329b05340c02eb31** by Ned Deily (Miro Hrončok) in branch '3.6':
~~**bpo-30458**~~: Disallow control chars in http URLs. (~~**GH-12755**~~) (~~**GH-13155**~~)
https://github.com/python/cpython/commit/c50d437e942d4c4c45c8cd76329b05340c02eb31

msg341932 - (view) **Author: Charalampos Stratakis (cstratak) *** **Date: 2019-05-08 19:25**

A small clarification on the differences of those two CVE's.

CVE-2019-9740: CLRF sequences are not properly handled in python built-in modules urllib/urllib2 in the query part of the url parameter of urlopen() function

CVE-2019-9947: CLRF sequences are not properly handled in python built-in modules urllib/urllib2 in the path part of the url parameter of urlopen() function

msg342470 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-05-14 15:09**

I backported the fix from Python 3.7 to Python 2.7: ~~**PR 13315**~~.

Please review it carefully, I had to make multiple changes to adapt the fix to Python 2:

* non-ASCII characters are explicitly rejected
* urllib doesn't reject control characters: they are quoted properly, so I addapted test_urllib
* urllib2 doesn't quote the URL and so reject control characters, I added tests to test_urllib2
* I replaced http.client with httplib
* I replaced urllib.request with urllib or urllib2

msg343045 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-05-21 13:12**

New changeset **bb8071a4cae5ab3fe321481dd3d73662ffb26052** by Victor Stinner in branch '2.7':
~~**bpo-30458**~~: Disallow control chars in http URLs (~~**GH-12755**~~) (~~**GH-13154**~~) (~~**GH-13315**~~)
https://github.com/python/cpython/commit/bb8071a4cae5ab3fe321481dd3d73662ffb26052

msg343104 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2019-05-21 21:53**

Assigning to Larry to decide if he wants to merge that PR into 3.5 or not.

msg344826 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-06-06 16:06**

Note for myself: Python 2 urllib.urlopen(url) always quotes the URL and so is not vulnerable to HTTP Header Injection (at least, not to this issue ;-)).

msg347282 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-07-04 14:02**

The commit **b7378d77289c911ca6a0c0afaf513879002df7d5** is incomplete: it doesn't seem to check for control characters in the "host" part of the URL, only in the "path" part of the URL. Example:

```
---
try:
    from urllib import request as urllib_request
except ImportError:
    import urllib2 as urllib_request
import socket
def bug(*args):
    raise Exception(args)
# urlopen() must not call create_connection()
socket.create_connection = bug
urllib_request.urlopen('http://127.0.0.1\r\n\x20hihi\r\n :11211')
---
```

The URL comes from the first message of this issue:
https://bugs.python.org/issue30458#msg294360

Development branches 2.7 and master produce a similar output:

```
---
Traceback (most recent call last):
 ...
Exception: (('127.0.0.1\r\n hihi\r\n ', 11211), ..., None)
---
```

So urllib2/urllib.request actually does a real network connection (DNS query), whereas it should reject control characters in the "host" part of the URL.

***

A second problem comes into the game. Some C libraries like glibc strip the end of the hostname (strip at the first newline character) and so HTTP Header injection is still possible is this case:
https://bugzilla.redhat.com/show_bug.cgi?id=1673465

***

According to the RFC 3986, the "host" grammar doesn't allow any control character, it looks like:

```
   host          = IP-literal / IPv4address / reg-name

   ALPHA (letters)
   DIGIT (decimal digits)
```

```
   unreserved    = ALPHA / DIGIT / "-" / "." / "_" / "~"
      pct-encoded = "%" HEXDIG HEXDIG
      sub-delims  = "!" / "$" / "&" / "'" / "(" / ")"
                  / "*" / "+" / "," / ";" / "="
   reg-name      = *( unreserved / pct-encoded / sub-delims )

   IP-literal    = "[" ( IPv6address / IPvFuture  ) "]"
   IPvFuture     = "v" 1*HEXDIG "." 1*( unreserved / sub-delims / ":" )
   IPv6address   =                            6( h16 ":" ) ls32
                 /                       "::" 5( h16 ":" ) ls32
                 / [               h16 ] "::" 4( h16 ":" ) ls32
                 / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
                 / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
                 / [ *3( h16 ":" ) h16 ] "::"    h16 ":"   ls32
                 / [ *4( h16 ":" ) h16 ] "::"              ls32
                 / [ *5( h16 ":" ) h16 ] "::"              h16
                 / [ *6( h16 ":" ) h16 ] "::"
   h16           = 1*4HEXDIG
   ls32          = ( h16 ":" h16 ) / IPv4address
   IPv4address   = dec-octet "." dec-octet "." dec-octet "." dec-octet
```

msg347285 - (view) **Author: Karthikeyan Singaravelan (xtreak) *** **Date: 2019-07-04 15:31**

Okay, the url variable against which the regex check is made is not the full url but the path. The HTTPConnection class sets self.host [0] in the constructor which is used to send the Host header. Perhaps the regex check could be done for the host too given the path check is already done in the previous commit. With that the reported host also throws a http.client.InvalidURL exception.

> A second problem comes into the game. Some C libraries like glibc strip the end of the hostname (strip at the first newline character) and so HTTP Header injection is still possible is this case: **https://bugzilla.redhat.com/show_bug.cgi?id=1673465**

The bug link raises permission error. Does fixing the host part fix this issue too since there won't be any socket connection made? Is it possible to have a Python reproducer of this issue?

[0]
https://github.com/python/cpython/blob/7f41c8e0dd237d1f3f0a1d2ba2f3ee4e4bd400a7/Lib/http/client.py#L829

msg347290 - (view) **Author: Riccardo Schirone (rschiron)** **Date: 2019-07-04 17:04**

> > A second problem comes into the game. Some C libraries like glibc strip the end of the hostname (strip at the first newline character) and so HTTP Header injection is still possible is this case: **https://bugzilla.redhat.com/show_bug.cgi?id=1673465**

> The bug link raises permission error. Does fixing the host part fix this issue too since there won't be any socket connection made? Is it possible to have a Python reproducer of this issue?

I think this was supposed to refer to CVE-2016-10739 (**https://bugzilla.redhat.com/show_bug.cgi?id=1347549**)

msg347897 - (view) **Author: Larry Hastings (larry) *** **Date: 2019-07-14 09:07**

New changeset **afe3a4975cf93c97e5d6eb8800e48f368011d37a** by larryhastings (Miro Hrončok) in branch '3.5':
~~**bpo-30458**~~: Disallow control chars in http URLs. (~~**GH-12755**~~) (~~**#13207**~~)
https://github.com/python/cpython/commit/afe3a4975cf93c97e5d6eb8800e48f368011d37a

msg350003 - (view) **Author: Riccardo Schirone (rschiron)** **Date: 2019-08-20 13:30**

Will the flaw outlined in **https://bugs.python.org/issue30458#msg347282** be fixed in python itself? If so, I think a CVE for python should be requested to MITRE (I can request one, in that case).

Moreover, does it make sense to create a new bug to track the new issue? This bug already references 3 CVEs and it would probably just create more confusion to reference a 4th. What do you think?

msg350028 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2019-08-20 17:53**

I'm not a fan of CVE numbers in general, people have been creating too many of those. But that also means I just don't care if someone does. Having a CVE entry is not a way to claim something is important.

This issue is still open and can be used to track dealing with the host.

msg352451 - (view) **Author: Jason R. Coombs (jaraco) *** **Date: 2019-09-14 20:38**

This change caused a regression or two captured in ~~**issue36274**~~. Essentially, by blocking invalid requests, it's now not possible for a system intentionally to generate invalid requests for testing purposes. As these point releases of Python start making it into the wild, the impact of this change will likely increase.

I think this patch was applied at too low a level. That is, instead of protecting the user inputs, the change protects the programmer's inputs.

I mention this here so those interested can follow the mitigation work happening in ~~**issue36274**~~.

msg352596 - (view) **Author: Ned Deily (ned.deily) *** **Date: 2019-09-17 04:58**

If I understand Jason's message correctly, the changes for ~~**Issue30458**~~ introduced a regression in 3.7.4 and will introduce the same regression in other branches as they are released, including 3.5.8 whose rc1 is now in testing. 3.7.5rc1 is scheduled to be tagged later today. Is this regression serious enough that we should hold 3.7.5 and/or 3.5.8 for a fix? If so, there should probably be a separate issue for it unless it is necessarily intertwined with the resolution of ~~**Issue36274**~~.

I'm provisionally setting the status of this issue to "release blocker".

msg352727 - (view) **Author: Larry Hastings (larry) *** **Date: 2019-09-18 13:32**

Should we open a separate issue to track fixing the regression?

msg352731 - (view) **Author: Jason R. Coombs (jaraco) *** **Date: 2019-09-18 14:37**

> Should we open a separate issue to track (fixing) the regression?

Yes, I think so. The ticket I referenced mainly addresses an incompatibility that was introduced with Python 3.0, so is much less urgent than the one introduced more recently, so I believe it deserves a proper, independent description and discussion. I'll gladly file that ticket, tonight most likely.

msg352751 - (view) **Author: Jason R. Coombs (jaraco) *** **Date: 2019-09-18 18:44**

I've created ~~issue38216~~ to address the (perceived) regression.

msg352760 - (view) **Author: Ned Deily (ned.deily) *** **Date: 2019-09-18 22:30**

With the breaking out of the portential and/or actual regression (e.g. invalid requests can no longer be crafted) into ~~Issue38216~~, itself a potential release blocker, we are still left here with the as-yet unresolved issue identified above in **msg34728** (e.g. not checking for control characters in the "host" part of the URL, only the "path" part). Since this also affects so many branches/releases and has external components (CVE's, third-party impacts), it probably would have made sense to break it out into a separate issue (and maybe it still does). But since this problem has been present for many releases (apparently), I would rather not further hold the 3.7.5 release for a resolution (though that would be a good thing) so I'm going to change the priority for the moment to "deferred blocker".

But we need someone (preferably a core dev already involved) to take charge of this and push it to a resolution. Thanks for everyone's help so far!

msg355246 - (view) **Author: Riccardo Schirone (rschiron)** **Date: 2019-10-23 18:34**

CVE-2019-18348 has been assigned to the issue explained in **https://bugs.python.org/issue30458#msg347282** . Maybe a separate bug for it would be better though. CVE-2019-18348 is about injecting CRLF in HTTP requests through the *host* part of a URL.

msg355261 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2019-10-23 19:48**

Can you please open a separate issue for CVE-2019-18348? It is easier to track that way.

(META: In general I think the CVE process is being abused and that these really did not deserve that treatment. **https://lwn.net/Articles/801157/** is good reading and food for thought.)

msg355298 - (view) **Author: Riccardo Schirone (rschiron)** **Date: 2019-10-24 08:12**

I have created **https://bugs.python.org/issue38576** to address CVE-2019-18348.

@gregory.p.smith if you have particular complains about these CVEs feel free to let me know (even privately). I think the security impact of these flaws is: an application that relies on urlopen/HTTPConnection/etc. where either the query part, the path part or the host part are user-controlled, could be exploited to send unintended HTTP headers to other hosts (maybe services that would not be directly reachable by the user).

FYI, there were some good replies to that CVE talk, one of which is **https://grsecurity.net/reports_of_cves_death_greatly_exaggerated** .

msg357988 - (view) **Author: Ned Deily (ned.deily) *** **Date: 2019-12-07 20:45**

What is the status of this issue? Now that ~~**Issue38576**~~ has been opened to cover the host address part, can this issue be closed or downgraded? Should ~~**Issue38576**~~ be a release blocker?

msg358050 - (view) **Author: Gregory P. Smith (gregory.p.smith) *** **Date: 2019-12-09 03:10**

i believe new work will be done via the new issue. marking this closed. if there is something not covered by ~~**issue38576**~~ that remains, please open a new issue for it. new discussion on this long issue is easy to get lost in.

History

Date	User	Action	Args
2022-04-11 14:58:46	admin	set	github: 74643
2019-12-09 03:10:03	gregory.p.smith	set	status: open -> closed resolution: fixed messages: + **msg358050** stage: patch review -> resolved
2019-12-07 20:45:10	ned.deily	set	messages: + **msg357988**
2019-10-24 08:12:29	rschiron	set	messages: + **msg355298**
2019-10-23 19:48:10	gregory.p.smith	set	messages: + **msg355261**
2019-10-23 18:34:37	rschiron	set	messages: + **msg355246**
2019-09-21 08:55:08	jaraco	set	stage: needs patch -> patch review pull_requests: + **pull_request15900**
2019-09-18 22:30:07	ned.deily	set	priority: release blocker -> deferred blocker messages: + **msg352760**
2019-09-18 18:44:54	jaraco	set	messages: + **msg352751**
2019-09-18 14:37:37	jaraco	set	messages: + **msg352731**
2019-09-18 13:32:02	larry	set	messages: + **msg352727**
2019-09-17 04:58:53	ned.deily	set	priority: normal -> release blocker nosy: + **lukasz.langa**, **benjamin.peterson**, **ned.deily** messages: + **msg352596**
2019-09-14 20:38:54	jaraco	set	nosy: + **jaraco** messages: + **msg352451**
2019-08-20 17:53:52	gregory.p.smith	set	keywords: + **security_issue** assignee: **larry** ->
messages: + **msg350028** stage: patch review -> needs patch

2019-08-20 13:30:47	rschiron	set	messages: + **msg350003**
2019-07-14 09:07:14	larry	set	messages: + **msg347897**
2019-07-04 17:04:56	rschiron	set	nosy: + **rschiron** messages: + **msg347290**
2019-07-04 15:31:37	xtreak	set	messages: + **msg347285**
2019-07-04 14:02:42	vstinner	set	messages: + **msg347282** versions: + Python 2.7, Python 3.6, Python 3.7, Python 3.8, Python 3.9
2019-06-06 16:06:09	vstinner	set	messages: + **msg344826**
2019-06-03 06:15:52	python-dev	set	pull_requests: + **pull_request13655**
2019-05-29 12:15:07	matrixise	set	pull_requests: + **pull_request13546**
2019-05-29 12:14:46	push0ebp	set	pull_requests: + **pull_request13545**
2019-05-21 21:53:01	gregory.p.smith	set	assignee: **larry** messages: + **msg343104** nosy: + **larry**
2019-05-21 21:52:07	gregory.p.smith	set	versions: - Python 3.6, Python 3.7
2019-05-21 21:51:47	gregory.p.smith	set	versions: - Python 2.7
2019-05-21 13:12:39	vstinner	set	messages: + **msg343045**
2019-05-14 15:09:30	vstinner	set	messages: + **msg342470**
2019-05-14 15:04:51	vstinner	set	pull_requests: + **pull_request13225**
2019-05-08 19:25:57	cstratak	set	nosy: + **cstratak**, - **ned.deily**, **koobs**, **hroncok** messages: + **msg341932** versions: + Python 3.7
2019-05-08 18:41:16	hroncok	set	pull_requests: + **pull_request13118**
2019-05-08 16:33:29	ned.deily	set	nosy: + **ned.deily** messages: + **msg341906**
2019-05-07 15:29:37	gregory.p.smith	set	versions: - Python 3.7
2019-05-07 15:28:51	gregory.p.smith	set	messages: + **msg341750**
2019-05-07 14:50:11	hroncok	set	pull_requests: + **pull_request13072**
2019-05-07 14:32:29	hroncok	set	pull_requests: + **pull_request13071**
2019-05-07 14:03:04	koobs	set	nosy: + **koobs**
2019-05-07 13:42:24	hroncok	set	nosy: + **hroncok** messages: + **msg341724**
2019-05-02 17:47:42	xtreak	set	messages: + **msg341291**

2019-05-02 17:35:01	gregory.p.smith	set	messages: + **msg341290**
2019-05-02 16:58:20	xtreak	set	messages: + **msg341286**
2019-05-01 20:39:26	gregory.p.smith	set	messages: + **msg341234**
2019-05-01 20:10:45	gregory.p.smith	set	stage: backport needed -> patch review pull_requests: + **pull_request12964**
2019-05-01 12:02:40	gregory.p.smith	set	stage: patch review -> backport needed
2019-05-01 12:00:11	miss-islington	set	nosy: + **miss-islington** messages: + **msg341192**
2019-05-01 06:50:54	xtreak	set	stage: backport needed -> patch review pull_requests: + **pull_request12953**
2019-05-01 03:59:58	xtreak	set	assignee: **gregory.p.smith** -> (no value) stage: patch review -> backport needed messages: + **msg341178** versions: - Python 3.8
2019-05-01 03:30:25	xtreak	set	assignee: **gregory.p.smith** stage: backport needed -> patch review messages: + **msg341176** versions: + Python 3.8
2019-05-01 02:18:51	gregory.p.smith	set	assignee: **gregory.p.smith** -> (no value) stage: patch review -> backport needed
2019-05-01 02:18:09	gregory.p.smith	set	messages: + **msg341175** versions: - Python 3.8
2019-05-01 02:12:37	gregory.p.smith	set	messages: + **msg341174**
2019-04-17 15:40:34	vstinner	set	messages: + **msg340408**
2019-04-17 15:35:49	vstinner	set	messages: + **msg340407**
2019-04-17 15:32:37	vstinner	set	messages: + **msg340405**
2019-04-10 22:05:48	vstinner	set	messages: + **msg339894**
2019-04-10 19:31:06	gregory.p.smith	set	messages: + **msg339884**
2019-04-10 13:03:09	vstinner	set	messages: + **msg339861**
2019-04-10 12:35:01	vstinner	set	messages: + **msg339858**
2019-04-10 12:32:58	vstinner	set	messages: + **msg339857**
2019-04-10 11:28:11	martin.panter	set	messages: + **msg339853**
2019-04-10 10:59:20	xtreak	set	messages: + **msg339852**
2019-04-10 10:50:15	vstinner	set	title: [CVE-2019-9740][CVE-2019-9947][security] CRLF Injection in httplib -> [security][CVE-2019-9740][CVE-2019-9947] HTTP Header Injection (follow-up of CVE-2016-5699)
2019-04-10 10:48:21	vstinner	set	messages: + **msg339851**

2019-04-10 10:45:12	vstinner	set	messages: + **msg339850**
2019-04-10 10:43:18	vstinner	set	messages: + **msg339848** title: [CVE-2019-9740][security] CRLF Injection in httplib -> [CVE-2019-9740][CVE-2019-9947][security] CRLF Injection in httplib
2019-04-10 10:36:30	vstinner	set	versions: + Python 3.5, Python 3.6, Python 3.7, Python 3.8
2019-04-10 10:36:15	vstinner	set	messages: + **msg339846**
2019-04-10 09:32:36	gregory.p.smith	link	**issue35906 superseder**
2019-04-10 09:23:06	gregory.p.smith	set	assignee: **gregory.p.smith**
2019-04-10 09:22:34	gregory.p.smith	set	nosy: + **gregory.p.smith** messages: + **msg339840**
2019-04-10 09:11:30	gregory.p.smith	set	keywords: + **patch** stage: patch review pull_requests: + **pull_request12688**
2019-04-09 15:42:49	ware	set	nosy: + **ware**
2019-04-09 14:28:59	vstinner	set	nosy: + **vstinner** messages: + **msg339754** title: CRLF Injection in httplib -> [CVE-2019-9740][security] CRLF Injection in httplib
2019-03-17 08:52:59	xtreak	link	**issue36276 superseder**
2019-03-15 06:15:37	xtreak	set	nosy: + **xtreak** messages: + **msg337970**
2017-11-26 01:04:36	martin.panter	set	messages: + **msg306981**
2017-11-26 01:00:28	martin.panter	link	**issue32085 dependencies**
2017-11-26 00:03:14	martin.panter	set	type: security
2017-06-03 07:01:33	martin.panter	set	messages: + **msg295067**
2017-06-02 15:36:20	xiang.zhang	set	nosy: + **xiang.zhang**, **serhiy.storchaka**, **martin.panter** messages: + **msg295026**
2017-05-24 15:01:31	orange	create	