Sec Bug #79465 OOB Read in urldecode()

Submitted: 2020-04-10 16:00 UTC **Modified:** 2020-04-14 04:10 UTC
From: bigshaq at wearehackerone dot com **Assigned:**
Status: Closed **Package:** *URL Functions
PHP Version: Irrelevant **OS:** Any
Private report: No **CVE-ID:** 2020-7067

View | Add Comment | Developer | Edit

[2020-04-10 16:00 UTC] bigshaq at wearehackerone dot com

```
Description:
------------
If  ``CHARSET_EBCDIC`` is defined (usually, on systems with EBCDIC encoding support), an Out-of-Bounds read can occur
using a malformed url-encoded string.


[C_SNIPPET]
PHPAPI size_t php_url_decode(char *str, size_t len)
{
        char *dest = str;
        char *data = str;
/*...more code...*/

#ifndef CHARSET_EBCDIC
                        *dest = (char) php_htoi(data + 1);
#else
                        *dest = os_toebcdic[(char) php_htoi(data + 1)]; // <--- here
#endif

/* ... more code ... */
[/C_SNIPPET]

* ``os_toebcdic[256]`` is an array(or an "encoding map", i assume) used for decoding purposes.
* To convert the url-encoded string input into actual hex values, PHP uses ``php_htoi()`` and then convert the result
into a signed byte(char).
* This signed number is then provided as an index to the ``os_toebcdic[]`` array.
* There will be no OOB Read *after* the buffer because the max value of a byte is 0xff (=256), which is the same size
as ``os_toebcdic[]``
* However, the casting (mentioned in the second bullet) is done to a ``char`` type and not an ``unsigned char``, which
means that we can insert negative hex values to leak values that are found in the memory BEFORE the array.



if we run:
[PHP_SNIPPET]
<?
urldecode('%xfd'); //0xfd == -3, It could also be 0x80 for bigger OOB (which is -128 in dec)
?>
[/PHP_SNIPPET]


we can look at the casting in dynamic analysis:

[GDB_SNIPPET]
gdb-peda$ call php_htoi(data+1)
$42 = 0xfd

gdb-peda$ p/d (char)$42
$43 = -3
[/GDB_SNIPPET]

Note: I did not **completely** verify it because CHARSET_EBCDIC is not supported on my system. So I hope the gdb
snippet demonstrates the concept well enough.
I'm 99% sure that it's a valid bug because: if you look at it, it's pretty straight forward. There's a casting to a
signed byte and there are no bounds checking/validations. I usually won't report without a fully working PoC on my
system but on this case i still think it's worth looking into it (I tried to enable EBCDIC support in my system but
didn't find any info on the internet about how to do it).
Please let me know if you can verify.

Thanks!


Test script:
---------------
<?
urldecode('%xfd'); //0xfd == -3, It could also be 0x80 for bigger OOB (which is -128 in dec)
?>

Expected result:
----------------
copy the 253rd (unsigned 0xfd) index of os_toebcdic

Actual result:
--------------
copy the -3rd (signed 0xfd) byte before os_toebcdic
```

Patches

CVE-2020-7067 (last revision 2021-03-22 03:42 UTC by 1552630135 at qq dot com)

Add a Patch

Pull Requests

Add a Pull Request

History

All | Comments | Changes | Git/SVN commits | Related reports

[2020-04-10 16:23 UTC] bigshaq at wearehackerone dot com

There's a mistake in my payload/test script, it should be without the 'x' character after the precent(%) character.

Here's a new one:
[PHP_SNIPPET]
<?php
urldecode('%fd');
?>
[/PHP_SNIPPET]

[2020-04-11 10:05 UTC] cmb@php.net

> […] then convert the result into a signed byte(char).

char is not necessarily signed. Whether it is signed or unsigned
is actually implementation defined.

[2020-04-11 11:43 UTC] bigshaq at wearehackerone dot com

Hi @cmb, thanks for the quick response.
Yes, I know that it depends on the implementation...

I wrote that char is signed by default as if it was "obvious" because any common implementation (that respect itself)
support both signed and unsigned char by default(x86 GNU/Linux, Microsoft Windows and so on)

Yes, there might be some cases when char is unsigned by default and the OOB-Read won't work but those are edge cases
imho.

[2020-04-13 05:01 UTC] stas@php.net

-CVE-ID:
+CVE-ID: 2020-7067

[2020-04-14 04:06 UTC] stas@php.net

I have no way to test it and I am not 100% sure it even compiles now, but I guess it doesn't hurt to put unsigned char
there...

[2020-04-14 04:10 UTC] stas@php.net

Automatic comment on behalf of stas
Revision: http://git.php.net/?p=php-src.git;a=commit;h=9d6bf8221b05f86ce5875832f0f646c4c1f218be
Log: Fix bug #79465 - use unsigneds as indexes.

[2020-04-14 04:10 UTC] stas@php.net

-Status: Open
+Status: Closed

[2021-03-22 03:42 UTC] 1552630135 at qq dot com

The following patch has been added/updated:

Patch Name: CVE-2020-7067
Revision: 1616384567
URL: https://bugs.php.net/patch-display.php?bug=79465&patch=CVE-2020-7067&revision=1616384567