CVE-2016-9424

Description: An issue was discovered in the Tatsuya Kinoshita w3m fork before 0.5.3-31. w3m doesn't properly validate the value of **tag attribute**, which allows remote attackers to cause a **denial of service** (**heap buffer overflow** crash) and possibly execute arbitrary code via a **crafted HTML page**.

References:
https://github.com/tats/w3m/blob/master/ChangeLog; https://github.com/tats/w3m/issues/12

Related CWE: CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer

Codes: Entry points:"HTML tag attribute", "crafted HTML page (input data)"; **Targets:**"heap buffer"; **Mechanisms:** "...".

W3M Github Changelog

2016-08-17 Tatsuya Kinoshita tats@debian.org>
....
* file.c, form.c:
Prevent negative **array index** for selectnumber and textareanumber.
Bug-Debian: https://github.com/tats/w3m/issues/12 [CVE-2016-9424]
....

Codes: Entry points: "..."; **Targets:** "array access"; **Mechanisms:** "...".

W3M Github Issues 2020-01-31-2

heap out of bound write due to negative array index 12:
How to reproduce:
$ echo -e ' <table><>\x00 >**<select_int selectnumber=90000000000>'** | $./w3m -Ttext/html -dump > /dev/null$ $Segmentation fault$
$ echo -e ' <table<>\x00><**select_int selectnumber=-90000>'** | $./w3m -Ttext/html -dump > /dev/null$ $Segmentation fault$
Here, selectnumber could be negative, or positive but overflows to negative.
The corresponding code snippet:
6033 **if (parsedtag_get_value(tag, ATTR_SELECTNUMBER, &n_select)**
6034 && n_select <max_select) {
6035 **select_option[n_select].first** = NULL;
n_select is the selectnumber mentioned above. It will crash at line 6035.
Similar code pattern at line 6015:
if (parsedtag_get_value(tag, ATTR_TEXTAREANUMBER,&n_textarea)
&& n_textarea <max_textarea) {
textarea_str[n_textarea] = Strnew();
this is found by afl-fuzz

Codes: Entry points:"tag attribute ", "functions that get tag attribute"; **Targets:**"array access", "write memory"; **Mechanisms:**"...".

CWE-119

Description: The software performs operations on a **memory buffer**, but it can read from or **write to a memory location** that is outside of the intended boundary of the buffer.

Example:
.....
The following example asks a user for an offset into an array to select an item.
Example Language: C
int main (int argc, char **argv) {
char *items[] = "boat", "car", "truck", "train";
int index = GetUntrustedOffset();
printf("You selected %s", **items[index-1]**);
}
The programmer allows the user to specify which element in the list to select, however an attacker can provide an **out-of-bounds offset**, resulting in a buffer over-read (CWE-126).
.....

Codes: Entry points: "..."; **Targets:** "memory buffer", "array access", "write memory statement"; **Mechanisms:** "...".