Talos Vulnerability Report

TALOS-2022-1510

DD-WRT httpd unescape memory corruption vulnerability

JULY 27, 2022

CVE NUMBER

CVE-2022-27631

Summary

A memory corruption vulnerability exists in the httpd unescape functionality of DD-WRT Revision 32270 - Revision 48599. A specially-crafted HTTP request can lead to memory corruption. An attacker can send a network request to trigger this vulnerability.

Tested Versions

DD-WRT Revision 32270 - Revision 48599

Product URLs

DD-WRT - https://dd-wrt.com/

CVSSv3 Score

5.3 - CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N

CWE

CWE-20 - Improper Input Validation

Details

DD-WRT is a Linux-based firmware for embedded systems. This open-source firmware offers several functionalities like: VPN integrations, ease of configuration through web browser, WLAN supports, etc.

The DD-WRT's httpd component has a file named `cgi.c` that contains CGI helper functions. One of these functions is `unescape`:

```
static void unescape(char *s)
{
    unsigned int c;

    while ((s = strpbrk(s, "%+"))) {
[1]
        /* Parse %xx */
        if (*s == '%') {
            sscanf(s + 1, "%02x", &c);
[2]
            *s++ = (char)c;
[3]
            strncpy(s, s + 2, strlen(s) + 1);
[4]
        }
        /* Space is special */
        else if (*s == '+')
            *s++ = ' ';
    }
}
```

This function take as argument a string. If URL-encoded, this function will decode it. At `[1]`, there is a loop that takes the next `%` or `+` in the string. If a `%` is found, then at `[2]`, the two characters following it are converted from hex values to a single character. At `[3]` the converted character replaces the `%` character and the string pointer advance. At `[4]` the string after the already-parsed URL-encoded character is moved left by two positions. This will replace the parsed characters. A string like "A…B%41%42" would go through the following steps:

```
|A|...|B|%|4|1|%|4|2|NULL|                    at    [1]/[2]
|A|...|B|A|4|1|%|4|2|NULL|                    after [3]
|A|...|B|A|%|4|2|NULL|NULL|NULL|              after [4]
```

Eventually, after the second iteration of the loop, we would end up like this:

```
|A|...|B|A|B|NULL|NULL|NULL|NULL|NULL|           after [4]
```

The `unescape` function assumes, wrongly, that after a `%` there are always at least two characters. If this is not the case, the instruction at `[4]` would cause an out-of bounds read and could cause also an out-of-bounds write. Let's take for instance the following string "A…B%a"; this would go through the following steps:

```
|A|...|B|%|a|NULL|Q|Q|Q|Q|Q|                    at    [1]/[2]
|A|...|B|\n|a|NULL|Q|Q|Q|Q|Q|                    after [3]
            ^ s points to 'a'                   after [3]
```

Assuming that after the string there is other data, in this scenario the string "QQQQQ", `s+2`, at `[4]`, will point to the first `Q`. So after `[4]` the string will look like:

```
|A|...|B|\n|a|NULL|Q|Q|Q|Q|Q|                    after [3]
                  ^ s+2 point to the first Q    at [4]
|A|...|B|\n|Q|Q|Q|Q|Q|Q|Q|                      after [4]
```

The result would be the string "A…B\nQQQQQQQ…".

Timeline

2022-04-11 - Initial vendor contact
2022-04-27 - Vendor Disclosure
2022-05-06 - Talos - Follow-up Sent
2022-06-07 - Talos - Follow-up Sent
2022-07-14 - Talos - Follow-up Sent
2022-07-27 - Public Release

CREDIT

Discovered by Francesco Benvenuto of Cisco Talos.
