**Problem**

<https://roy.marples.name/git/dhcpcd>

auth.c in dhcpcd before 7.2.1 allowed attackers to infer secrets by performing latency attacks.

<https://www.tutorialspoint.com/c_standard_library/c_function_memcmp.htm>

The C library function **int memcmp(const void \*str1, const void \*str2, size\_t n))** compares the first **n** bytes of memory area **str1** and memory area **str2**.

**auth.c**

if (**memcmp**(d, &hmac\_code, dlen)) {

**Fix**

<https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=928056>

The fixed issues are (copied from upstream's announcement):

\* auth: Use **consttime\_memequal** to **avoid latency attack** consttime\_memequal is supplied if libc does not support it

dhcpcd >=6.2 <7.2.1 are vulnerable

\* DHCP: Fix a potential 1 byte read overflow with DHO\_OPTSOVERLOADED

dhcpcd >=4 <7.2.1 are vulnerable

\* DHCPv6: Fix a potential buffer overflow reading NA/TA addresses

dhcpcd >=7 <7.2.1 are vulnerable

**auth.c**

if (!**consttime\_memequal**(d, &hmac\_code, dlen)) {

<https://github.com/intel/linux-sgx/blob/master/sdk/tlibc/string/consttime_memequal.c>

int consttime\_memequal(const void \*b1, const void \*b2, size\_t len) {

const unsigned char \*c1 = b1, \*c2 = b2;

unsigned int res = 0;

while (len--)

res |= \*c1++ ^ \*c2++;

/\*

\* Map 0 to 1 and [1, 256) to 0 using only constant-time

\* arithmetic.

\*

\* This is not simply `!res' because although many CPUs support

\* branchless conditional moves and many compilers will take

\* advantage of them, certain compilers generate branches on

\* certain CPUs for `!res'.

\*/

return (1 & ((res - 1) >> 8));

}