Bug 568803 (CVE-2021-34430) - Vulnerability in TinyDTLS

Status: CLOSED MOVED

Alias: CVE-2021-34430

Product: Community
Component: Vulnerability Reports (show other bugs)
Version: unspecified
Hardware: PC Linux

Importance: P3 normal (vote)
Target Milestone: ---
Assignee: Security vulnerabilitied reported against Eclipse projects
QA Contact:

URL:
Whiteboard:
Keywords: security

Depends on:
Blocks:

Reported: 2020-11-13 14:49 EST by Wayne Beaton ✓ ECA
Modified: 2021-12-23 06:46 EST (History)
CC List: 3 users (show)

See Also:

Attachments		
Writeup and PoC of the Vuln (5.50 KB, application/zip) 2020-11-17 06:16 EST, Ruben Gonzalez ✓ ECA	*no flags*	Details
Add an attachment (proposed patch, testcase, etc.)		View All

Note
You need to log in before you can comment on or make changes to this bug.

Wayne Beaton ✓ ECA 2020-11-13 14:49:30 EST Description

From the Security Team inbox (abridged):

The original poster included attachments which we have not opened. I will check to see if we can get the original poster to join the conversation here.

--

Tinydtls uses the default pseudo random number generator for the affected systems.

I will explain the bug based on POSIX. TinyDTLS uses '/dev/urandom' as a source of entropy to seed a PRNG using srand(entropy). I assume, this is done in response to a complaint about unsafe cookies on the tinydtls mailing list[1]. Afterwards it uses the libc's rand() function to obtain pseudo random bits in the function 'dtls_prng':

```
int dtls_prng(unsigned char *buf, size_t len) {
  while (len--)
    *buf++ = rand() & 0xFF;
  return 1;
}
```

It is not sufficient to seed a PRNG with entropy. In all major C standard libraries, the rand() function outputs states of a simple PRNG, such as a linear congruence generator.

As the name says, all states are in a linear relationship. Which is why one state (output) is usually enough to compute all future outputs (and preceding ones). Even if the output is only partially available (as in the function 'dtls_prng'), few outputted states are enough to recover the internal PRNG state.

The direct relationship between outputs is a property all major PRNGs have (such as the mersenne twister or LSFRs).

Also, the default internal state size of PRNGs in most standard libraries (e.g. glibc or dietlibc) is only around 32 bits.

Cryptographically secure pseudo-random number generator (CSPRNG) do not have this linear property, an output does not give hints about the internal state.

However, such a CSPRNG is not used in tinydtls.

My exploit works as follows:

- Observe a DTLS handshake. Included (and publicly visible) is the value server_random, which functions as a nonce.

- Since the server_random value gets its entropy from 28 calls to the 'dtls_prng' function, its bytes correspond to outputs of the PRNG.

- The secret scalar used for the elliptic curve computation is then obtained by calling 'dtls_prng' 32 times. Note that obtaining this secret scalar on one side of the connection is enough to compromise the entire communication, since all derived symmetric keys are based on it, see [2].

- As an attacker: Recover the internal state of the PRNG using the publicly known server_random value.

- Then the attacker just sets the state of his PRNG to this recovered internal state

- By calling rand() 32 times on his PRNG, the attacker obtains the secret scalar (key) used by the server

The same attack works equivalently for the client side. Meaning that it is enough if one side of the connection is using tinydtls.

Attached is the exploit code with a README file, explaining how to use it. There is also a network dump included, against which the attack can be tested.

Please note that my exploit just brute forces through all possible PRNG states on a current glibc. This takes about 0.014 seconds on my CPU for the current master branch.

HOWEVER, making the seed value (initial state of the PRNG) wider, as it is in the develop branch does not fix the problem! The state still leaks out through the server_random/client_random.

The only solution would be to either include a CSPRNG in tinydtls or replace dtls_prng with always reading bytes out of '/dev/urandom' (if available).

I think this flaw calls for a CVE to inform possible users.

I found out about this bug because a coworker uses tinydtls in an embedded project.

Wayne Beaton ✓ ECA 2020-11-13 14:57:10 EST Comment 1

```
Note that the handbook has some content regarding how we handing vulnerability
reports.

https://www.eclipse.org/projects/handbook/#vulnerability
```

Ruben Gonzalez ✓ ECA 2020-11-17 06:16:54 EST Comment 2

```
Created attachment 284785 [details]
Writeup and PoC of the Vuln
```

Ruben Gonzalez ✓ ECA 2020-11-17 06:17:21 EST Comment 3

```
Hi, I'm the reporter and I just added the attachment to this thread.
```

Wayne Beaton ✓ ECA 2021-07-06 17:59:39 EDT Comment 4

```
I've assigned CVE-2021-34430. The CVE has been reported to the central authority.
```

Frederic Gurr ✓ ECA 2021-12-23 06:46:46 EST Comment 5

```
This issue has been migrated to
https://gitlab.eclipse.org/eclipsefdn/helpdesk/-/issues/540.
```

Format For Printing - XML - Clone This Bug - Top of page