**Problem**

The random number generator in the Crypto application before 2.0.2.2, and SSH before 2.0.5, as used in the Erlang/OTP ssh library before R14B03, uses predictable seeds based on the current time, which makes it easier for remote attackers to guess DSA host and SSH session keys.

Text, letter

Description automatically generated

**ssh\_connection\_handler.erl**

nit([Role, Manager, Socket, SshOpts]) ->

**{A,B,C} = erlang:now(),**

**random:seed(A, B, C),**

{NumVsn, StrVsn} = ssh\_transport:versions(Role, SshOpts),

ssh\_bits:install\_messages(ssh\_transport:transport\_messages(NumVsn)),

**ssh\_bits.erl**

irandom(0, \_Top, \_Bottom) ->

0;

irandom(Bits, Top, Bottom) ->

Bytes = (Bits+7) div 8,

Skip = (8-(Bits rem 8)) rem 8,

TMask = case Top of

0 -> 0;

1 -> 16#80;

2 -> 16#c0

end,

BMask = case Bottom of

0 -> 0;

1 -> (1 bsl Skip)

end,

<<X:Bits/big-unsigned-integer, \_:Skip>> = **random(Bytes, TMask, BMask)**,

X.

**random(N)** ->

**random(N, 0, 0).**

**random(N, TMask, BMask)** ->

list\_to\_binary(rnd(N, TMask, BMask)).

%% random/3

%% random(Bytes, TopMask, BotMask)

%% where

%% Bytes is the number of bytes to generate

%% TopMask is bitwised or'ed to the first byte

%% BotMask is bitwised or'ed to the last byte

%%

rnd(0, \_TMask, \_BMask) ->

[];

rnd(1, TMask, BMask) ->

[(rand8() bor TMask) bor BMask];

rnd(N, TMask, BMask) ->

[(rand8() bor TMask) | rnd\_n(N-1, BMask)].

rnd\_n(1, BMask) ->

[rand8() bor BMask];

rnd\_n(I, BMask) ->

[rand8() | rnd\_n(I-1, BMask)].

rand8() ->

(rand32() bsr 8) band 16#ff.

rand32() ->

random:uniform(16#100000000) -1.

**Fix**

La fonction random se basais sur le seed du temps (**random:seed(A, B, C), {A,B,C} = erlang:now()**).

Dans le fix, elle se base sur une fonction ajouté dans le fichier **crypto.c**. C’est précisément ceci qui a régler le problème (la fonction *random* est utilisé dans *irandom* aussi). En conclusion, c’est précisément le fichier **ssh\_connection\_handler.erl** qui causait la vulnérabilité dans le fichier **ssh\_bits.erl**. Maintenant, le fichier **ssh\_bits.erl** utilise le fichier **crypto.c** pour l’entropy.

**ssh\_connection\_handler.erl**

nit([Role, Manager, Socket, SshOpts]) ->

**(REMOVED {A,B,C} = erlang:now(), random:seed(A, B, C))**

{NumVsn, StrVsn} = ssh\_transport:versions(Role, SshOpts),

ssh\_bits:install\_messages(ssh\_transport:transport\_messages(NumVsn)),

**ssh\_bits.erl**

irandom(Bits, Top, Bottom) when is\_integer(Top),

0 =< Top, Top =< 2 ->

crypto:erlint(**crypto:strong\_rand\_mpint(Bits, Top - 1, Bottom)**).

random(N) ->

**crypto:strong\_rand\_bytes(N).**

**crypto.c**

static ERL\_NIF\_TERM strong\_rand\_bytes\_nif(ErlNifEnv\* env, int argc, const ERL\_NIF\_TERM argv[]);

static ERL\_NIF\_TERM strong\_rand\_mpint\_nif(ErlNifEnv\* env, int argc, const ERL\_NIF\_TERM argv[]);

{"strong\_rand\_bytes\_nif", 1, strong\_rand\_bytes\_nif},

{"strong\_rand\_mpint\_nif", 3, strong\_rand\_mpint\_nif},

static ERL\_NIF\_TERM strong\_rand\_bytes\_nif(ErlNifEnv\* env, int argc, const ERL\_NIF\_TERM argv[])

**{/\* (Bytes) \*/**

**unsigned bytes;**

**unsigned char\* data;**

**ERL\_NIF\_TERM ret;**

**if (!enif\_get\_uint(env, argv[0], &bytes)) {**

**return enif\_make\_badarg(env);**

**}**

**data = enif\_make\_new\_binary(env, bytes, &ret);**

**if ( RAND\_bytes(data, bytes) != 1) {**

**return atom\_false;**

**}**

**ERL\_VALGRIND\_MAKE\_MEM\_DEFINED(data, bytes);**

**return ret;**

**}**

Text, email

Description automatically generated

static ERL\_NIF\_TERM strong\_rand\_mpint\_nif(ErlNifEnv\* env, int argc, const ERL\_NIF\_TERM argv[])

**{/\* (Bytes, TopMask, BottomMask) \*/**

**unsigned bits;**

**BIGNUM \*bn\_rand;**

**int top, bottom;**

**unsigned char\* data;**

**unsigned dlen;**

**ERL\_NIF\_TERM ret;**

**if (!enif\_get\_uint(env, argv[0], &bits)**

**|| !enif\_get\_int(env, argv[1], &top)**

**|| !enif\_get\_int(env, argv[2], &bottom)) {**

**return enif\_make\_badarg(env);**

**}**

**if (! (top == -1 || top == 0 || top == 1) ) {**

**return enif\_make\_badarg(env);**

**}**

**if (! (bottom == 0 || bottom == 1) ) {**

**return enif\_make\_badarg(env);**

**}**

**bn\_rand = BN\_new();**

**if (! bn\_rand ) {**

**return enif\_make\_badarg(env);**

**}**

**/\* Get a (bits) bit random number \*/**

**if (!BN\_rand(bn\_rand, bits, top, bottom)) {**

**ret = atom\_false;**

**}**

**else {**

**/\* Copy the bignum into an erlang mpint binary. \*/**

**dlen = BN\_num\_bytes(bn\_rand);**

**data = enif\_make\_new\_binary(env, dlen+4, &ret);**

**put\_int32(data, dlen);**

**BN\_bn2bin(bn\_rand, data+4);**

**ERL\_VALGRIND\_MAKE\_MEM\_DEFINED(data+4, dlen);**

**}**

**BN\_free(bn\_rand);**

**return ret;**

**}**

Text, letter

Description automatically generated