New issue

Jump to bottom

[libtpms 0.7] TPM2_CreatePrimary creates prime numbers with 32 zero bits #183

Closed **fishilico** opened this issue on Feb 17, 2021 · 8 comments

fishilico commented on Feb 17, 2021

Describe the bug
When running `tpm2_createprimary` on swtpm with libtpms 0.7.4 to generate an RSA-2048 key, the modulus of the key always contains many zeros in its high bits. By extracting the prime factors from file holding the persistent TPM state, they always have 32 bits set to zero. For example:

- Modulus `n =` `0xc8f8f77a0000bacc2f18e6a43f600d9990b9079654f85b23da5b26896a7cab24d52ea3dcae0bc754b2cf0aefd3ac036459d4a2bd1ec1c25d56f7822a633c1bd29ad54f4ec7a7e3e8d64697c737283c4b77ec01c9eb0dd36371dfb7fd160b1b6f946db24f0eda15997e488f7427c87f173d5d3baecf98d0d6832a97d28c3da99835cf22a83c39d6dd61f880422541b80d958a0f8756f804a65571c7937768717648deb6f8ccc8343a42a2b672ecd4e523f0ded37822db4b1af769416e4dcdb3d55ad4c7b78f47904a47f1418bc59295c2b2a6e2ea9f3c40caaf1f23a79ed18aab4bfb9754676b47759e705d5ab1ba8fd96be01ac7e475485a4577e105d6dc5e4f`
- First prime number `p =` `0xdd7500000000a998d562fd94dd501fbaedbd693474333472bab4a5889a021b5ae79d4bf98c87d2509831103be119297abdc09e315a66bf1918a2d27a19ed805b01ea87f01e433c14bb6dde779bb70de6878b0697b9f2d1b064fe48bd06c83d11d3170d99a7f1637d2c10f0e00a0f04bd21ce3e02a2fa8803fd8fe5d9512b60b9`
- Second prime number `q =` `0x0xe8520000000026049bfad92bbdea021f996acd0c55557fd42d16acb73fbd94cd908a636fc5da4e635707f7aa1e2f3041f144755520094b6ac322dfd57f5f3b98b679725a7bd33373fd67a11f2d51bbef4cf486bf60db2c89093bfcbc229f640be853f04d98b27aa0b5a613d5edd4350d8c90e7189a4097f0703f4f88f0556347`

(Observe the eight `0` hexdigits after the 4 first hexdigits of the prime numbers).

This is due to a bug in function `RsaAdjustPrimeCandidate`

libtpms/src/tpm2/crypto/openssl/CryptPrime.c
Lines 304 to 318 in `2452a24`

```
304   LIB_EXPORT void
305   RsaAdjustPrimeCandidate(
306                           bigNum          prime
307                           )
308   {
309       UINT16  highBytes;
310       crypt_uword_t       *msw = &prime->d[prime->size - 1];
311   #define MASK (MAX_CRYPT_UWORD >> (RADIX_BITS - 16))
312       highBytes = *msw >> (RADIX_BITS - 16);
313       // This is fixed point arithmetic on 16-bit values
314       highBytes = ((UINT32)highBytes * (UINT32)0x4AFB) >> 16;
315       highBytes += 0xB505;
```

The issue is that on 64-bit systems, `MASK` is not `0x0000ffffffffffff` but `0x000000000000ffff` when `RADIX_BITS` is 64, so only the 16 lowest bits of `*msw` are kept instead of the 48 lowest bits. More precisely, in `#define MASK (MAX_CRYPT_UWORD >> (RADIX_BITS - 16))`, the shift operand should have been `16`.

This bug is present in TCG specification (https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-4-Supporting-Routines-01.38-code.pdf section `10.2.15.7 AdjustPrimeCandiate()`). This specification was updated and the current version does not have this bug (https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part4_SuppRoutines_code_pub.pdf section `10.2.14.1.6 AdjustPrimeCandiate()`). This new version was implemented in April 2020 in `625171b` (branch `master`) but no release of `libtpms` includes the new version yet.

Therefore I have three questions:

- Could `RsaAdjustPrimeCandidate` be fixed in branch `stable-0.7.0` so that generating new RSA keys do not use prime numbers with many zeros, on 64-bit systems?
- If no, could a comment be added which clearly state that this prime number generator generates prime numbers that have 32 bits always set to zero and that the TCG already fixed this issue in a newer version of "Trusted Platform Module Library Family "2.0" Specification - Part 4: Routines - Code"?
- When will the next release of `libtpms` (version `0.8` ?) occur?

To Reproduce
Steps to reproduce the behavior:

1. On Arch Linux on an x86-64 system, install `swtpm`, `tpm2-tools` and `tpm2-abrmd`.
2. Launch `swtpm` in TPM2-mode and `tpm2-abrmd`, for example with:

```
swtpm socket --tpm2 --server port=2321 --daemon --ctrl type=tcp,port=2322 --flags not-need-init --tpmstate dir=/tmp
tpm2-abrmd --allow-root --tcti swtpm:host=127.0.0.1,port=2321
```

3. Generate a persistent RSA key on the software TPM, for example with:

```
export TPM2TOOLS_TCTI=tabrmd:bus_type=system
tpm2_createprimary -c /tmp/context.out -g sha256 -G rsa
tpm2_evictcontrol -c /tmp/context.out 0x81000001
```

4. Analyze the content of `/tmp/tpm2-00.permall` to retrieve the modulus and the prime numbers of the generated RSA key.

Expected behavior
`tpm2_createprimary` should create an RSA key with prime numbers which really look random, instead of with 32 bits always set to zero.

Desktop (please complete the following information):

- OS: Arch Linux on an x86-64 CPU (64-bit system)

Versions of relevant components

- libtpms: 0.7.4
- swtpm: 0.5.2

fishilico changed the title ~~[libtpms 0.7] TPM2_CreatePrimary creates prime numbers with 32 zeros bits~~ [libtpms 0.7] TPM2_CreatePrimary creates prime numbers with 32 zero bits on Feb 17, 2021

stefanberger commented on Feb 17, 2021 Owner

I am aware of this issue and cannot solve it for 0.7.x. Users may have encrypted data with this key and we cannot just fix the key creation algorithm and create different keys derived from seeds that then cannot decrypt the data anymore. The created keys have to be the same. So the solution is more complicated than just fixing the key creation algorithm. It involves tracking the 'age' of the seed and only switch to the new key creation algorithm when new seeds are created and creating it with the broken algorithm for as long as the old seed is used. Backporting the fixes would disturb the way the TPM 2 state is written and so I don't want to do that - no backports of fixes that change the state that is being written out otherwise we may loose the upgrade path.
The issue is solved for the 0.8.0 release (Commit `c1f7bf5` in master activates the fix.) but this version is on hold due to issue #147 related to Linux not being able to deal with 3072 bit keys - or at least the fix that I provided for that has to be distributed first to distros before we can use it

stefanberger commented on Feb 17, 2021 • edited Owner

> If no, could a comment be added which clearly state that this prime number generator generates prime numbers that have 32 bits always set to zero and that the TCG already fixed this issue in a newer version of "Trusted Platform Module Library Family "2.0" Specification - Part 4: Routines - Code"?

Where do you want me to add a comment?

fishilico commented on Feb 17, 2021 Author

Thanks for your quick reply. I understand that 0.7.x cannot be easily modified without disturbing things and I agree with your approach.

> Where do you want me to add a comment?

I was thinking of modifying (in `stable-0.7.0`)

libtpms/src/tpm2/crypto/openssl/CryptPrime.c
Line 311 in `2452a24`

```
311      #define MASK (MAX_CRYPT_UWORD >> (RADIX_BITS - 16))
```

to:

```
/* This computation is known to be buggy on 64-bit systems, as MASK=0xffff
 * instead of 0x0000ffffffffffff and this introduces 32 zero bits in the
 * adjusted prime number. But fixing this would modify keys which were
 * previously generated, so keep it this way for now.
 * This was fixed in the current version of "Trusted Platform Module Library
 * Family 2.0 Specification - Part 4: Routines - Code" (from TCG) and this issue
 * will be fixed for newly-generated prime numbers for the version of libtpms
 * which will come after 0.7.x.
 */
#define MASK (MAX_CRYPT_UWORD >> (RADIX_BITS - 16))
```

For issue #147, the last comment seems to indicate that the stable versions of the Linux kernel were patched and that the issue was fixed. Why is this comment still open? (It would be helpful to add a sentence about what is now blocking this issue)

stefanberger commented on Feb 17, 2021 Owner

The primary reason is that I wanted to give it some time for the fix the be propagated through distros and people have had a chance to update their systems. The bad keys won't go away for existing VMs but will be fixed primarily for new ones.

I am going to make a 0.7.5 release now and added this comment to the CHANGES file:

```
version 0.7.5
  - Note: The TPM 2 implementation returns 2048 bit keys with ~1984 bit
          strength due to a bug in the TPM 2 key creation algo that cannot
          easily be fixed. The bug is in RsaAjustPrimeCandidate, which is
          called before the prime number check.
```

Ok, so let me update the code as well with your additional text.

stefanberger commented on Feb 17, 2021 Owner

I added this patch now to the PR including your Reported-by: line: `0e3d5d0`

stefanberger commented on Feb 17, 2021 Owner

There's another unfortunate problem with 3072 bit RSA key support. If someone was to use an older version of a Linux distro where tools may not support the larger size of the context then they are stuck. I believe the Intel tools did have some issues with larger sized key contexts at some point. Downgrading libtpms from 0.8 to 0.7.y would not work then. So this is a tricky issue and I may have to modify libtpms 0.8 in such a way that the older short context is returned for 2048 bit RSA keys... I wished I didn't have to change this code.

salahcoronya mentioned this issue on Feb 24, 2021

app-crypt/libtpms: Bump to 0.8.2 gentoo/gentoo#19630

Closed

stefanberger commented on Feb 24, 2021 Owner

I just released v0.8.0 that should resolve the prime number issue for new instances of swtpm or when a user changes the seeds, possibly using a vFirmware menu in case a VM is used. The user has to be aware that keys that depended on those seeds or data/keys that were encrypted with those keys, will be lost then.

fishilico commented on Mar 1, 2021 Author

Thanks! I confirm that libtpms v0.8.1 works nicely (on Arch Linux on an x86-64 computer), and that I successfully upgraded from 0.7.5:

- `tpm2_createprimary -g sha256 -G rsa -C e` , `tpm2_createprimary -g sha256 -G rsa -C o` and `tpm2_createprimary -g sha256 -G rsa -C p` continued to produce the same RSA keys (which were issued from prime numbers with 32 zero bits), as expected.
- After changing the seeds (with `tpm2_changeeps` and `tpm2_changepps`) the keys changed, as expected.
- The persistent TPM state saved `EPSeedCompatLevel = 1` and `PPSeedCompatLevel = 1` , as expected.

Moreover for my use-case, not being able to downgrade from 0.8.x to 0.7.y is not an issue.

So from my perspective, the issue is fixed and you did an excellent work for this. Thanks again!

fishilico closed this as completed on Mar 1, 2021

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants