**Problem**

The crypto\_report\_one function in crypto/crypto\_user.c in the report API in the crypto user configuration API in the Linux kernel through 3.8.2 uses an incorrect length value during a copy operation, which allows local users to obtain sensitive information from kernel memory by leveraging the CAP\_NET\_ADMIN capability.

1. The structures used for the netlink based crypto algorithm report API are located on the stack. As **snprintf() does not fill the remainder of the buffer with null bytes**, those stack bytes will be **disclosed to users of the API**. Switch to strncpy() to fix this.
2. crypto\_report\_one() does not initialize all field of struct crypto\_user\_alg. Fix this to fix the **heap info leak**.
3. For the module name we should copy only as many bytes as module\_name() returns -- not as much as the destination buffer could hold. But the current code does not and therefore it copies random data from behind the end of the module name, as the module name is always shorter than CRYPTO\_MAX\_ALG\_NAME.
4. Also switch to use strncpy() to copy the algorithm's name and driver\_name. They are strings, after all (memcpy).

La documentation est assez claire sur le sujet, le problème viens principalement de snprintf et la copie d’octets du nom de module qui se base sur le le tampon de destination et non module\_name().

Parameters: int snprintf(char \*str, size\_t size, const char \*format, …);

\*str : is a buffer.

size : is the maximum number of bytes (characters) that will be written to the buffer.

format : C string that contains a format string that follows the same specifications as format in printf

… : the optional ( …) arguments are just the string formats like (“%d”, myint) as seen in printf.

**ablkcipher.c**

**snprintf**(rblkcipher.type, **CRYPTO\_MAX\_ALG\_NAME**, "%s", "ablkcipher");

**snprintf**(rblkcipher.geniv, **CRYPTO\_MAX\_ALG\_NAME**, "%s", alg->cra\_ablkcipher.geniv ?: "<default>");

**snprintf**(rblkcipher.type, **CRYPTO\_MAX\_ALG\_NAME**, "%s", "givcipher");

**snprintf**(rblkcipher.geniv, **CRYPTO\_MAX\_ALG\_NAME**, "%s", alg->cra\_ablkcipher.geniv ?: "<built-in>");

**aead.c**

**snprintf**(raead.type, **CRYPTO\_MAX\_ALG\_NAME**, "%s", "aead");

**snprintf**(raead.geniv, **CRYPTO\_MAX\_ALG\_NAME**, "%s", aead->geniv ?: "<built-in>");

**snprintf**(raead.type, **CRYPTO\_MAX\_ALG\_NAME**, "%s", "nivaead");

**snprintf**(raead.geniv, **CRYPTO\_MAX\_ALG\_NAME**, "%s", aead->geniv);

**ahash.c**

**snprintf**(rhash.type, **CRYPTO\_MAX\_ALG\_NAME**, "%s", "ahash");

**blkcipher.c**

**snprintf**(rblkcipher.type, **CRYPTO\_MAX\_ALG\_NAME**, "%s", "blkcipher");

**snprintf**(rblkcipher.geniv, **CRYPTO\_MAX\_ALG\_NAME**, "%s", alg->cra\_blkcipher.geniv ?: "<default>");

**crypto\_user.c** :

**snprintf**(rcipher.type, **CRYPTO\_MAX\_ALG\_NAME**, "%s", "cipher");

**snprintf**(rcomp.type, **CRYPTO\_MAX\_ALG\_NAME**, "%s", "compression");

**snprintf**(rl.type, **CRYPTO\_MAX\_ALG\_NAME**, "%s", "larval");

**memcpy**(&ualg->cru\_name, &alg->cra\_name, sizeof(ualg->cru\_name));

**memcpy**(&ualg->cru\_driver\_name, &alg->cra\_driver\_name, sizeof(ualg->cru\_driver\_name));

**memcpy**(&ualg->cru\_module\_name, module\_name(alg->cra\_module), **CRYPTO\_MAX\_ALG\_NAME**);

**pcompress.c**

**snprintf**(rpcomp.type, **CRYPTO\_MAX\_ALG\_NAME**, "%s", "pcomp");

**rng.c**

**snprintf**(rrng.type, **CRYPTO\_MAX\_ALG\_NAME**, "%s", "rng");

**shash.c**

**snprintf**(rhash.type, **CRYPTO\_MAX\_ALG\_NAME**, "%s", "shash");

**Fix**

Parameters : char \*strncpy(char \*dest, const char \*src, size\_t n)

dest − This is the pointer to the destination array where the content is to be copied.

src − This is the string to be copied.

n − The number of characters to be copied from source.

**ablkcipher.c**

**strncpy**(rblkcipher.type, "ablkcipher", **sizeof(rblkcipher.type)**);

**strncpy**(rblkcipher.geniv, alg->cra\_ablkcipher.geniv ?: "<default>", **sizeof(rblkcipher.geniv)**);

**strncpy**(rblkcipher.type, "givcipher", **sizeof(rblkcipher.type)**);

**strncpy**(rblkcipher.geniv, alg->cra\_ablkcipher.geniv ?: "<built-in>", **sizeof(rblkcipher.geniv)**);

**aead.c**

**strncpy(**raead.type, "aead", sizeof(raead.type));

**strncpy**(raead.geniv, aead->geniv ?: "<built-in>", sizeof(raead.geniv));

**strncpy**(raead.type, "nivaead", sizeof(raead.type));

**strncpy**(raead.geniv, aead->geniv, sizeof(raead.geniv));

**ahash.c**

**strncpy**(rhash.type, "ahash", **sizeof(rhash.type)**);

**blkcipher.c**

**strncpy**(rblkcipher.type, "blkcipher", **sizeof(rblkcipher.type)**);

**strncpy**(rblkcipher.geniv, alg->cra\_blkcipher.geniv ?: "<default>", **sizeof(rblkcipher.geniv)**);

**crypto\_user.c** :

**strncpy**(rcipher.type, "cipher", **sizeof(rcipher.type)**);

**strncpy**(rcomp.type, "compression", **sizeof(rcomp.type)**);

**strncpy**(ualg->cru\_name, alg->cra\_name, sizeof(ualg->cru\_name));

**strncpy**(ualg->cru\_driver\_name, alg->cra\_driver\_name, sizeof(ualg->cru\_driver\_name));

**strncpy**(ualg->cru\_module\_name, module\_name(alg->cra\_module), **sizeof(ualg->cru\_module\_name)**);

ualg->cru\_type = 0; **(fix heap info leak)**

ualg->cru\_mask = 0; **(fix heap info leak)**

ualg->cru\_flags = alg->cra\_flags;

ualg->cru\_refcnt = atomic\_read(&alg->cra\_refcnt);

**strncpy**(rl.type, "larval", **sizeof(rl.type)**);

**pcompress.c**

**strncpy**(rpcomp.type, "pcomp", **sizeof(rpcomp.type)**);

**rng.c**

**strncpy**(rrng.type, "rng", **sizeof(rrng.type)**);

**shash.c**

**strncpy**(rhash.type, "shash", **sizeof(rhash.type)**);