SELinux (outils et bibliothèques) vs analyse de code

Nicolas Iooss

Rump SSTIC 2017

SELINUX ?

- Contrôle d'accès obligatoire [1], module de sécurité Linux [2]
- "iptables pour des processus" (politique, restriction d'accès, etc.)
- Contextes avec des étiquettes, types, roles, niveaux, classes, contraintes, etc.

1. Mandatory Access Control (MAC)
2. Linux Security Module (LSM)

SELinux ?

- Contrôle d'accès obligatoire [1], module de sécurité Linux [2]
- "iptables pour des processus" (politique, restriction d'accès, etc.)
- Contextes avec des étiquettes, types, roles, niveaux, classes, contraintes, etc.

- Code :
  - noyau (Linux : security/selinux/)
  - utilisateur (libsepol, libselinux, setools, etc.)
- Compilateurs de politiques : des parseurs en C !
  - source .te/.if/.fc ou .cil
  - formats binaires .mod, .pp, policy
  - checkpolicy, checkmodule, semodule, secilc, hll/pp, secil2conf, etc.

1. Mandatory Access Control (MAC)
2. Linux Security Module (LSM)

Vérifications ?

- Tests de non-régression [3] (make test)
- Drapeaux d'attention [4] (-Wall -Wextra -Werror -Wformat=2)
- Compilateur pédant [5]
- Analyseur statique [6] (LLVM scan-build)
- Désinfectants du compilateur [7] (ASan, UBSan, etc.)
- Frelatage [8] (American Fuzzy Lop)

---

3. Tests
4. Warning flags
5. clang -Weverything
6. scan-build
7. Sanitizers
8. Fuzzing

Drapeaux d'attention

2014 :

seusers_local.c : In function ‘semanage_seuser_modify_local’ :
seusers_local.c :122 :6 : error : ‘rc’ may be used uninitialized in this function [-Werror=maybe-uninitialized]

Drapeaux d'attention

2014 :

seusers_local.c : In function ‘semanage_seuser_modify_local’ :
seusers_local.c :122 :6 : error : ‘rc’ may be used uninitialized in this function [-Werror=maybe-uninitialized]

```
int semanage_seuser_modify_local(/*...*/) {
  /* ... */
  if (semanage_seuser_clone(handle, data, &new) < 0) {
    goto err;
  }
  /* ... */
err:
  return rc;
}
```

Quelques bogues trouvés et corrigés récemment

Saurez-vous trouver les bogues[9] dans les extraits suivants ?

9. bugs

Trouvez le bogue (1)

```
int semanage_get_active_modules(/* ... */) {
    /* ... */
cleanup:
  semanage_list_destroy(&list);
  for (i = 0; i < all_modinfos_len; i++)
    semanage_module_info_destroy(sh, &all_modinfos[i]);

  free(all_modinfos);
  if (status != 0) {
    for (i = 0; i < j; j++) {
      semanage_module_info_destroy(sh, &(*modinfo)[i]);
    }
    free(*modinfo);
  }
  return status;
}
```

Trouvez le bogue (1)

Janvier 2017 (merci clang)

```
--- a/libsemanage/src/semanage_store.c
+++ b/libsemanage/src/semanage_store.c
@@ -1158,7 +1158,7 @@ cleanup:
     free(all_modinfos);

     if (status != 0) {
-        for (i = 0; i < j; j++) {
+        for (i = 0; i < j; i++) {
             semanage_module_info_destroy(
                 sh, &(*modinfo)[i]);
         }
         free(*modinfo);
```

Trouvez le bogue (2)

```
int role_set_expand(role_set_t * x, ebitmap_t * r,
    policydb_t * out, policydb_t * base,
    uint32_t * rolemap) {
  unsigned int i;
  policydb_t *p = out;

  ebitmap_init(r);

  if (x->flags & ROLE_STAR) {
    for (i = 0; i < p->p_roles.nprim++; i++)
      if (ebitmap_set_bit(r, i, 1))
        return -1;
    return 0;
  }
  /* ... */
```

Trouvez le bogue (2)

Novembre 2016 (merci American Fuzzy Lop)

```
--- a/libsepol/src/expand.c
+++ b/libsepol/src/expand.c
@@ -2424,7 +2424,7 @@
      ebitmap_init(r);

      if (x->flags & ROLE_STAR) {
-       for (i = 0; i < p->p_roles.nprim++; i++)
+       for (i = 0; i < p->p_roles.nprim; i++)
          if (ebitmap_set_bit(r, i, 1))
            return -1;
        return 0;
```

Trouvez le bogue (3)

```
int avrule_ioctl_ranges(
    struct av_ioctl_range_list **rangelist) {
  struct av_ioctl_range_list *rangehead;
  uint8_t omit;
  /* read in ranges to include and omit */
  if (avrule_read_ioctls(&rangehead))
    return -1;
  omit = rangehead->omit;
  if (rangehead == NULL) {
    yyerror("error processing ioctl commands");
    return -1;
  }
  /* sort and merge the input ioctls */
  if (avrule_sort_ioctls(&rangehead))
    return -1;
```

Trouvez le bogue (3)

Mars 2017

```
--- a/checkpolicy/policy_define.c
+++ b/checkpolicy/policy_define.c
@@ -1924,11 +1924,11 @@
    /* read in ranges to include and omit */
    if (avrule_read_ioctls(&rangehead))
      return -1;
-   omit = rangehead->omit;
    if (rangehead == NULL) {
      yyerror("error processing ioctl commands");
      return -1;
    }
+   omit = rangehead->omit;
    /* sort and merge the input ioctls */
    if (avrule_sort_ioctls(&rangehead))
      return -1;
```

Exécution de pointeur NULL!

Novembre 2016

```
--- a/libsepol/src/module_to_cil.c
+++ b/libsepol/src/module_to_cil.c
@@ -3530,6 +3530,9 @@
    struct avrule_decl *decl = stack_peek(decl_stack);

    for (args.sym_index = 0; args.sym_index < SYM_NUM;
         args.sym_index++) {
+     if (func_to_cil[args.sym_index] == NULL) {
+       continue;
+     }
      rc = hashtab_map(decl->symtab[args.sym_index].table
                       additive_scopes_to_cil_map, &args);
      if (rc != 0) {
        goto exit;
```

LLVM SCAN-BUILD (CLANG STATIC ANALYZE)

https ://cloud.iooss.fr/vrac/rump2017.html#EndPath

```
270 static inline void cil_reset_context(struct cil_context *context)
271 {
272         if (context->range_str == NULL) {
                    7 ← Access to field 'range_str' results in a dereference of a null pointer (loaded from variable 'context')
273                 cil_reset_levelrange(context->range);
274         }
275 }
276
277 static void cil_reset_sidcontext(struct cil_sidcontext *sidcontext)
278 {
279         if (sidcontext->context_str == NULL) {
280                 cil_reset_context(sidcontext->context);
281         }
282 }
283
284 static void cil_reset_filecon(struct cil_filecon *filecon)
285 {
286         if (filecon->context_str == NULL && filecon->context != NULL) {
287                 cil_reset_context(filecon->context);
288         }
289 }
290
291 static void cil_reset_ibpkeycon(struct cil_ibpkeycon *ibpkeycon)
292 {
293         if (!ibpkeycon->context)
                    3 ← Assuming pointer value is null →
                    4 ← Taking true branch →
```

ET AUSSI...

- Utilisation après libération [10], libération double [11], etc.
- Fuites de mémoire allouée [12]
- ...

10. Use after free
11. Double free
12. Memory leaks

Conclusion

SELinux est génial pour tester des outils d'analyse de code ![13]

13. SELinux is great !

Merci de votre attention

https://github.com/SELinuxProject/SELinux/commits?author=fishilico