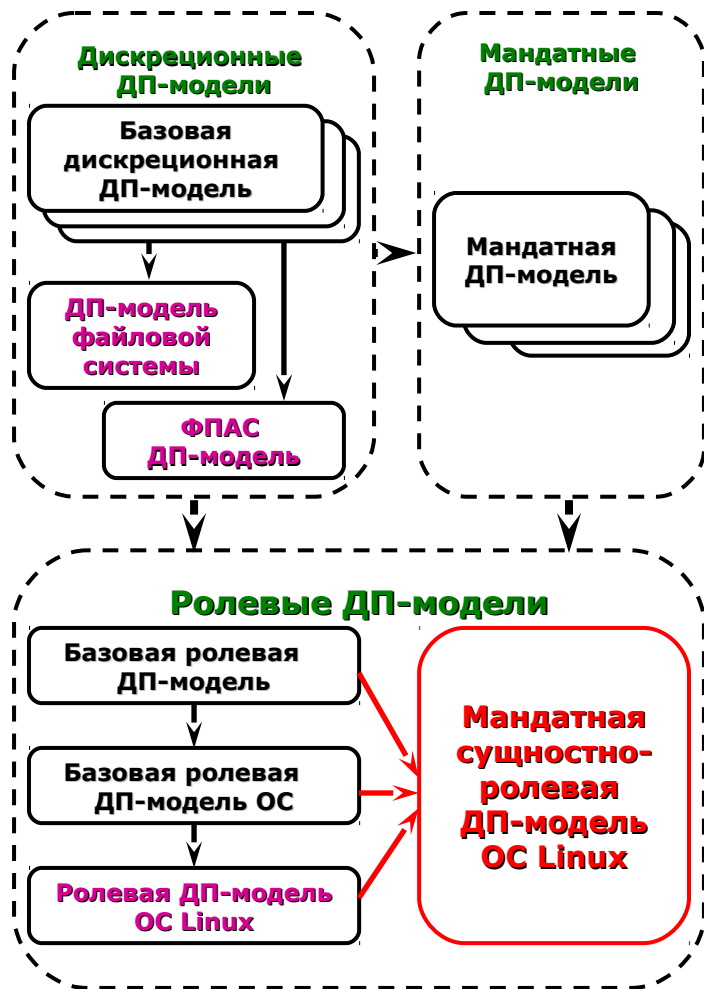

О представлении MРОСЛ ДП-модели в формализованной нотации Event-B (Rodin Platform)

д.т.н., доцент Девянин П.Н. (УМО ИБ),
к.ф.-м.н., доцент В.В. Кулямин,
д.ф.-м.н., профессор А.К. Петренко,
к.ф.-м.н. А.В. Хорошилов,
И.В. Щепетков (ИСП РАН)

Постановка задачи исследования

Семейство ДП-моделей



Отечественная защищенная ОС Astra Linux Special Edition

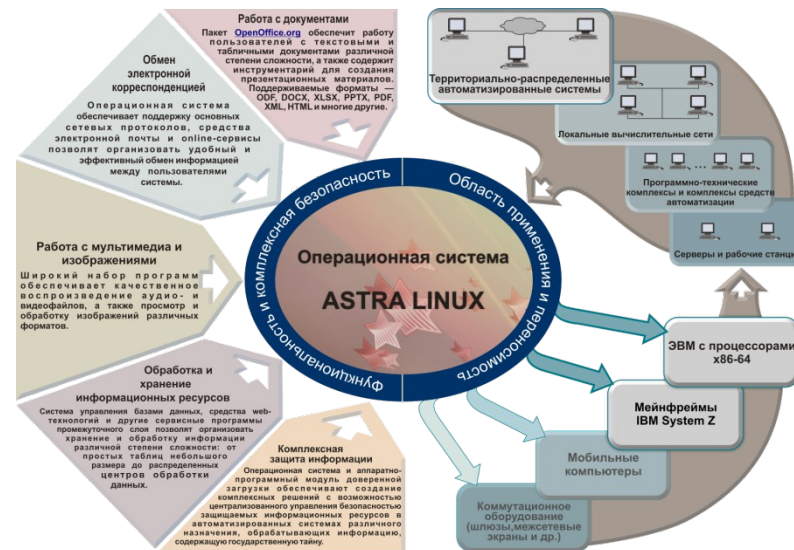


Схема решения задач исследования



Схема формирования МРОСЛ ДП-модели



Де-юре и де-факто правила преобразования состояний

de_facto_own: $S \rightarrow 2^S$ – функция фактического владения субъект-сессий субъект-сессиями.

de_facto_accesses: $S \rightarrow 2(E \cup R \cup AR) \times Ra$ – функция де-факто доступов субъект-сессий, при этом по определению в каждом состоянии системы **G** для каждой субъект-сессии $s \in S$ верно равенство:

de_facto_accesses(s) = {(e, α_a): существует $s' \in de_facto_own(s)$ и [либо $e \in E$ и (s', e, α_a) $\in A$], [либо $e \in R \cup AR$ и (s', e, α_a) $\in AA$]}.

Примеры де-юре правил

access_read (x, x', y)	x, x' $\in S, y \in E \cup R \cup AR$, существует $r \in R \cup AR: (x, r, read_e) \in AA$, [если $y \in E$, то $(y, read_e) \in PA(r)$ и либо $(execute_container(x, y) = true$ и $f_e(y) \leq f_e(x)$], либо $(x, downgrade_admin_role, read_e) \in AA$], [если $y \in R \cup AR$, то $(y, read_e) \in APA(r), i_e(y) \leq i_e(x)$, $Constraint_{t,u}(AA) = true$, (для $e \in E$] либо $(x, e, read_e) \in A$, либо $(x, e, write_e) \in A$], [либо $f_e(y) \leq f_e(x)$, либо $(x, downgrade_admin_role, read_e) \in AA$], [если $y \in R \cup AR$ и $i_e(y) = i_high$, то $(x', f_e(x)_l_entity, write_e) \in A$]	$S' = S, E' = E, APA' = APA, PA' = PA, user' = user, H_e' = H_e, F = F$, если $y \in E$, то $A' = A \cup \{(x, y, read_e)\}$, $AA' = AA$, если $y \in R \cup AR$, то $AA' = AA \cup \{(x, y, read_e)\}$, $A' = A$
create_hard_link (x, x', y, name, z)	x, x' $\in S, y \in O, z \in C, execute_container(x, y) = true, (x, z, write_e) \in A$, $name \in NAME \setminus \{\epsilon\}$, [либо $(f_e(y) = f_e(z) = f_e(x))$, если $CCR(y) = false$ или $CCR(y) = false$, то $(x, entities_admin_role, read_e) \in AA$], либо $(f_e(y) \leq f_e(z)$ и $(x, downgrade_admin_role, read_e) \in AA$], $i_e(y) \leq \min(i_e(z), i_e(x))$ [если $i_e(z) = i_high$, то $(x', f_e(x)_l_entity, write_e) \in A$]	$S' = S, E' = E, APA' = APA, PA' = PA, A' = A, AA' = AA, user' = user, F = F$, $entity_name(z, y) = name$, $H_e'(z) = H_e(z) \cup \{y\}$, для $e \in E \setminus \{z\}$ выполняется $H_e'(e) = H_e(e)$
create_first_session (x, x', u, y, z, zc, zi)	x, x' $\in S, u \in U, y \in E, z \notin S$, существует $r \in R \cup AR$ такая, что $(x, r, read_e) \in AA$ и $(y, execute) \in PA(r), execute_container(x, y) = true$, $f_e(y) \leq \min(f_e(x), f_e(u)), zc \leq f_e(u), zi \leq \min(i_e(u), i_e(y))$, [либо $f_e(x) \leq zc$, либо $(x, downgrade_admin_role, read_e) \in AA$], [для $e \in E$] либо $(x, e, read_e) \in A$, либо $(x, e, write_e) \in A$], [если $zi = i_high$, то $(x', f_e(x)_l_entity, write_e) \in A$]	$S' = S \cup \{z\}, E' = E, APA' = APA, A' = A, F = F$, $f_e'(z) = zc, i_e'(z) = zi, user'(z) = u$, для $s \in S$ $user'(s) = user(s), f_e'(s) = f_e(s), i_e'(s) = i_e(s)$, $AA' = AA \cup \{(z, u_admin_i_zi, read_e)\}, (z, u_c_zc_zi, write_e) \cup \{(z, u_e_f_f, read_e)\}$, где $I \leq zc$ и $I \leq zi$, $PA'(u_c_zc_zi) = PA(u_c_zc_zi) \cup \{(z, own_e)\}$, и для $r' \in R \setminus \{u_c_zc_zi\}$ выполняется $PA'(r') = PA(r')$, $[z] = fa(u, y), [z] = fp(u, y), H_e'(z) = \emptyset$, для $s \in S$ выполняется $H_e'(s) = H_e(s)$

Примеры де-факто правил

de_facto_op (s, op(x, x', ...))	$s \in N_S, x, x' \in de_facto_own(s)$, выполняются условия применения де-юре правила преобразования состояний $op(x, x', ...)$	Соответствуют результатам применения правила $op(x, x', ...)$
control (x, y, z)	$x \in N_S, y \in S, z \in E, x \neq y, z \in [y]$ и $(x, z, write_e) \in F$	$S' = S, E' = E, PA' = PA, A' = A, AA' = AA, user' = user, H_e' = H_e$, $de_facto_own'(x) = de_facto_own(x) \cup \{y\}$, $F = F \cup \{(x, y, write_e), (y, x, write_e)\}$
take_access_own (x, y, z)	$x \in N_S, y, z \in S, y \in de_facto_own(x), z \in de_facto_own(y)$	$S' = S, E' = E, PA' = PA, A' = A, AA' = AA, user' = user, H_e' = H_e$, $de_facto_own'(x) = de_facto_own(x) \cup \{z\}$, $F = F \cup \{(x, z, write_e), (z, x, write_e)\}$
flow_memory_access (x, y, α_e)	$x \in S, y \in E \setminus E_HOLE$, $(y, \alpha_e) \in de_facto_accesses(x)$, где $\alpha_e \in \{read_e, write_e\}$	$S' = S, E' = E, PA' = PA, A' = A, AA' = AA, user' = user, H_e' = H_e$, если $\alpha_e = read_e$, то $F = F \cup \{(y, x, write_e)\}$, если $\alpha_e = write_e$, то $F = F \cup \{(x, y, write_e)\}$
flow_time_access (x, y)	$x \in N_S \cup NF_S$, или $[y \in E \cup R \cup AR, (y, \alpha_e) \in de_facto_accesses(x)]$, где $\alpha_e \in R_e$, или $[y \in S$ и $y \in de_facto_own(x)]$	$S' = S, E' = E, PA' = PA, A' = A, AA' = AA, user' = user, H_e' = H_e$, если $y \in E \cup R \cup AR$ и $\alpha_e = write_e$, то $F = F \cup \{(y, x, write_e)\} \cup \{(x, e, write_e): e \in E \cup R \cup AR, x \neq e$ и или $y \leq e$, или $e \leq y\}$; если $y \in E \cup R \cup AR$ и $\alpha_e = read_e$, то $F = F \cup \{(y, x, write_e)\}$; если $y \in S$, то $F = F \cup \{(y, x, write_e)\} \cup \{(x, e, write_e): e \in S, x \neq e$ и или $y \leq e$, или $e \leq y\}$
post (x, y, z)	$x, z \in S, y \in E \cup S \cup R \cup AR, x \neq z$, $(x, y, \alpha_e) \in F$, где $\alpha_e \in \{write_e, write_e\}$, $[y \in E \cup R \cup AR$ и $(y, \beta_e) \in de_facto_accesses(z)$, где $\beta_e \in R_e$], или $[y \in S$ и $y \in de_facto_own(z)]$	$S' = S, E' = E, PA' = PA, A' = A, AA' = AA, user' = user, H_e' = H_e$, если $\alpha_e = write_e, \beta_e = read_e$, и $y \in E \setminus E_HOLE$, то $F = F \cup \{(x, z, write_e)\}$, иначе если $x, z \in N_S \cup NF_S$, то $F = F \cup \{(x, z, write_e)\}$, иначе $F = F$

Безопасность в смысле Белла-ЛаПадулы и мандатного контроля целостности

Определение. Пусть G_0 – безопасное начальное состояние системы $\Sigma(G^*, OP, G_0)$, и существует траектория без кооперации доверенных и недоверенных субъект-сессий $G_0 \vdash_{op1} G_1 \vdash_{op2} \dots \vdash_{opN} G_N$, где $N \geq 1$. В состоянии G_N произошло нарушение безопасности системы, когда в нем выполняется одно из условий, при этом они не выполняются в G_i траектории, где $0 \leq i < N$.

Условие 1 (нарушение безопасности в смысле мандатного контроля целостности). Существуют недоверенная субъект-сессия $x \in N_{SN}$ и доверенная субъект-сессия $y \in de_facto_own_N(x) \cap L_{SN}$ такие, что $i_{sN}(y) = i_high$.

Условие 2 (нарушение безопасности в смысле Белла-ЛаПадулы). Существует информационный поток по памяти $(x, y, write_m) \in F_N$: $x, y \in E_N$ и не верно неравенство $f_{eN}(x) \leq f_{eN}(y)$.

Условие 3 (нарушение безопасности в смысле контроля информационных потоков по времени). Существует информационный поток по времени $(x, y, write_e) \in F_N$ такой, что $x, y \in E_N$ и не верно $f_{eN}(x) \leq f_{eN}(y)$.

Теорема (базовая теорема безопасности – БТБ-ДП). Пусть G_0 – безопасное начальное состояние системы $\Sigma(G^*, OP, G_0)$. Пусть на всех траекториях системы без кооперации доверенных или недоверенных субъект-сессий $G_0 \vdash_{op1} G_1 \vdash_{op2} \dots \vdash_{opN} G_N$, где $N \geq 1$, и в каждом G_N для каждой $s \in S_N$ и сущности $e \in E_N$ выполняются следующие условия.

Условие 1 (корректность уровней конфиденциальности и целостности сущностей, функционально ассоциированных с субъект-сессиями). Если $e \in [s]$, то выполняются условия $i_{sN}(s) \leq i_{eN}(e)$ и $(f_{sN}(s) = f_{eN}(e))$ или $i_{eN}(e) = i_high$.

Условие 2 (корректность уровней конфиденциальности и целостности, а также прав доступа на чтение к сущностям, параметрически ассоциированным с субъект-сессиями). Если $e \in]s[$, то $f_{sN}(s) = f_{eN}(e)$ и для каждой роли или административной роли $r \in R_N \cup AR_N$ такой, что $(e, read_r) \in PA_N(r)$, выполняются условия $i_{sN}(s) \leq i_{eN}(e) \leq i_{rN}(r)$.

Условие 3 (функциональная и параметрическая корректность всех доверенных субъект-сессий относительно всех доверенных субъект-сессий и сущностей). Для всех доверенных $s \in L_{SN}$ верно $f_correct_N(s) = p_correct_N(s) = L_{SN} \times (E_N \cup S_N)$.

Условие 4 (абсолютная функциональная и параметрическая корректность субъект-сессии относительно всех сущностей и субъект-сессий с совпадающим уровнем конфиденциальности). Для всех $s \in S_N$ верно $\{s' \in S_N: f_{sN}(s') = f_{sN}(s)\} \times (E_N \cup S_N) \subset af_correct_N(s) = ap_correct_N(s)$.

Тогда система $\Sigma(G^*, OP, G_0)$ безопасна в смысле условий 1, 2 определения безопасности системы.

Обоснование адекватности.

Пример: правило `access_write(x, x', y)`

$x, x' \in S,$

$y \in E \cup R \cup AR,$

существует $r \in R \cup AR: (x, r, read_a) \in AA,$

[если $y \in E,$ то

$i_e(y) \leq i_s(x)$

и (либо `execute_container(x, y) = true`

и, если $y \in E_HOLE,$ то $f_s(x) \leq f_e(y),$

иначе $f_e(y) = f_s(x),$

либо $(x, downgrade_admin_role, read_a) \in AA),$

и $(y, write_e) \in PA(r),$

[если $y \in R \cup AR,$ то $(y, write_e) \in APA(r),$

$i_r(y) \leq i_s(x), Constraint_{AA}(AA') = true,$

(для $e \in]y[$ либо $(x, e, read_a) \in A,$ либо $(x, e,$

$write_e) \in A),$ (либо $f_r(y) = f_s(x),$

либо $(x, downgrade_admin_role, read_a) \in AA)],$

[если $(y \in E$ и $i_e(y) = i_high)$ или

$(y \in R \cup AR$ и $i_r(y) = i_high),$

то $(x', f_s(x)_i_entity, write_e) \in A]$

```
int pdp_permission(const PDP_0* s, const PDP_0* o, int mode)
{
    if( o->type & PDP_TYPE_EHOLE ) return 0;

    if( mode & R_OK ) {
        if( (s->lev < o->lev) || ( (s->cat & o->cat) != o->cat ) ) return -1;
    }

    if( mode & W_OK ) {
        if( (s->lev > o->lev) || (s->ilev < o->ilev) || ( (s->cat & o->cat) != s->cat ) ) return -1;
    }

    if( mode & X_OK ) {
        if( (s->lev < o->lev) || ( (s->cat & o->cat) != o->cat ) ) return -1;
    }

    return 0;

    mask &= (MAY_READ|MAY_WRITE|MAY_EXEC|MAY_APPEND);
    task_role = list_entry(next_task_role, struct role, list);
    inode_role = (struct inode_rback*) list_entry(next_inode_role, struct role, list);
    while(next_task_role != task_roles_list)
    {
        while(next_inode_role != inode_roles_list)
        {
            if(inode_role->role_seed > task_role->role_seed)
            {
                next_inode_role = next_inode_role->next;
                inode_role = (struct inode_rback*) list_entry(next_inode_role, struct role, list);
                continue;
            }
            if(task_role->role_seed == inode_role->role_seed)
            {
                ret = rback_may_access(inode_role->role_access, mask);
                if(0 == ret)
                    return ret;
                next_inode_role = next_inode_role->next;
                inode_role = (struct inode_rback*) list_entry(next_inode_role, struct role, list);
                continue;
            }
            if(inode_role->role_seed < task_role->role_seed)
                break;
            return ret;
        }
        next_task_role = next_task_role->next;
        task_role = list_entry(next_task_role, struct role, list);
    }
    return ret;
}
```

Обоснование адекватности реализации



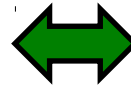
ИСПРАН

Правила МРОСЛ ДП-модели

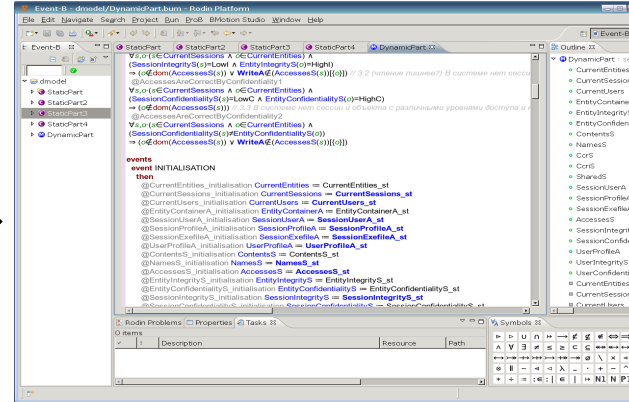
$access_read(x, x', y)$

$x, x' \in S, y \in E \cup R \cup AR$, существует $r \in R \cup AR$: $(x, r, read_a) \in AA$,
 [если $y \in E$, то $(y, read_a) \in PA(r)$ и либо $(execute_container(x, y) = true$ и $f_e(y) \leq f_s(x)$], либо $(x, downgrade_admin_role, read_a) \in AA$,
 [если $y \in R \cup AR$, то $(y, read_a) \in APA(r)$, $i_i(y) \leq i_s(x)$,
 $Constraint_{AA}(AA') = true$, (для $e \in y$) [либо $(x, e, read_a) \in A$, либо $(x, e, write_a) \in A$], (либо $f_i(y) \leq f_s(x)$], либо $(x, downgrade_admin_role, read_a) \in AA$],
 [если $y \in R \cup AR$ и $i_i(y) = i_high$, то $(x', f_s(x)_i_entity, write_a) \in A$]

$S' = S, E' = E, APA' = APA, PA' = PA$,
 $user' = user, H'_E = H_E, F' = F$,
 если $y \in E$, то $[A' = A \cup \{(x, y, read_a)\}]$,
 $AA' = AA$,
 если $y \in R \cup AR$, то
 $[AA' = AA \cup \{(x, y, read_a)\}, A' = A]$



Автоматизированная верификация МРОСЛ ДП-модели



Rodin (Event-B), Alloy



```

1 static int access_read(struct task_struct *subject, struct inode *entity)
2 {
3     struct task_security *subject_security;
4     struct inode_security *entity_security;
5     int ret = -EACCESS;
6
7     subject_security = get_task_security(subject);
8     entity_security = get_entity_security(entity);
9
10    if(!subject_security || !entity_security)
11        return ret;
12
13    if(is_role(entity)) //проверяем возможность получения доступа на чтение к сущности
14    {
15        ret = can_access(&subject_security->roles, &entity_security->list, MAY_READ, 0);
16        if(ret != 0)
17            ret = can_access(&subject_security->admin_roles, &entity_security->list, MAY_WRITE, 0);
18        if(ret == 0)
19        {
20            ret = execute_container(subject, entity);
21            if(ret != 0)
22                ret = is_downgrade_admin_role(&subject_security->admin_roles, MAY_READ);
23        }
24    }
25    else //проверяем возможность получения доступа на чтение к роли или административной роли
26    {
27        ret = can_admin_access(&subject_security->admin_roles, &entity_security->list, MAY_READ);
28    }
29    return ret;
  
```



```

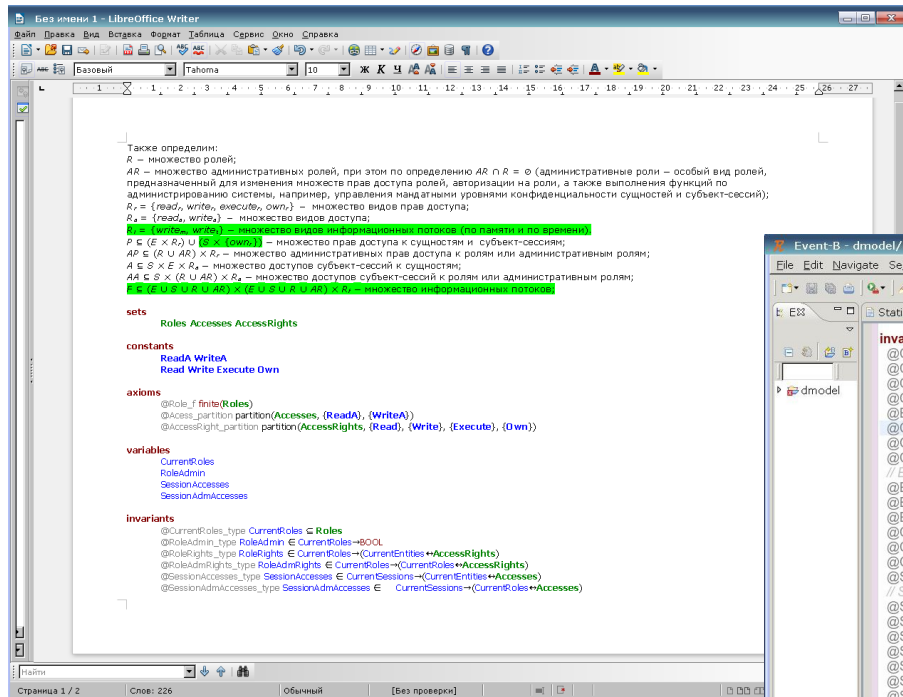
0 //B requires (valid(s) A (valid(o);
1 requires
2 (read_bit(mask) v write_bit(mask) v exec_bit(mask) v
3 writep_bit(mask);
4 terminates (true;
5 assigns (nothing;
6 behavior DENY;
7 assumes (read_bit(mask) v exec_bit(mask) v write_bit(mask);
8 ensures (result == 1;
9 //int mac_file;
10 behavior ALLOW;
11 assumes (read_bit(mask) v exec_bit(mask) v write_bit(mask);
12 ensures (ALLOW, result == 0);
13 behavior DEFAULT;
14 assumes (read_bit(mask) A exec_bit(mask) A write_bit(mask);
15 ensures (ALLOW, result == 0);
16 complete behaviors DEFAULT, ALLOW, DENY;
17 disjoint behaviors DEFAULT, ALLOW, DENY;
18 int mac_file.permission(parsec_mac_t const *, parsec_mac_label_t const *, mac_t mask);
19 int ret;
20 if (mask & 0x00000001) {
21     if (int i = 0; i < (unsigned int)0x00000001) {
22         if (int i = level < (int)0 - mac.level) {
23             ret = 1;
24             goto return_label;
25         }
26         else {
27             goto LAND;
28         }
29     }
30     else {
31         LAND; // internal
32         if (int i = 0; i < (unsigned int)0x00000001) {
33             ret = 1;
34         }
35     }
  
```

Frama-C, Why3

Программный код механизма управления доступом ОССН

Верификации кода на соответствие спецификациям, заданным на основе правил преобразования состояний

Переход от математической нотации к формализованной (*Event-B, Rodin Platform*)



Также определим:
 R – множество ролей;
 AR – множество административных ролей, при этом по определению $AR \cap R = \emptyset$ (административные роли – особый вид ролей, предназначенный для изменения множеств прав доступа ролей, авторизации на роли, а также выполнения функций по администрированию системы, например, управления мандатными уровнями конфиденциальности сущностей и субъект-сессий);
 $R_s = \{read, write, execute, own\}$ – множество видов прав доступа;
 $R_a = \{read, write\}$ – множество видов доступа.

$E \subseteq (E \cup O \cup C) \times (U \cup AR \cup R)$ – множество информационных потоков (по лантам и по времени);
 $P \subseteq (E \times R_s) \cup (E \times R_a)$ – множество прав доступа к сущностям и субъект-сессиям;
 $AP \subseteq (R \cup AR) \times R_s$ – множество административных прав доступа к ролям или административным ролям;
 $A \subseteq S \times E \times R_s$ – множество доступов субъект-сессий к сущностям;
 $AA \subseteq O \times (R \cup AR) \times R_a$ – множество доступов субъект-сессий к ролям или административным ролям;
 $I \subseteq (E \cup O \cup C \times U \cup AR) \times (E \cup O \cup C \cup R \cup AR) \times R_s$ – множество информационных потоков.

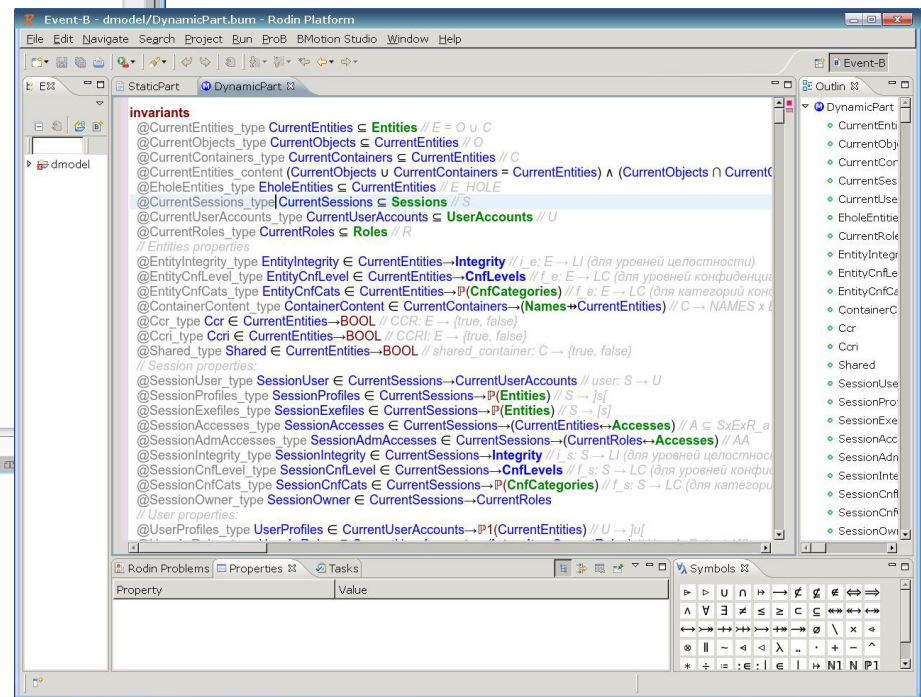
sets
Roles Accesses AccessRights

constants
Read WriteA
Read Write Execute Own

axioms
@Role_f finite(Roles)
@Access_partition partition(Accesses, (Read), (WriteA))
@AccessRight_partition partition(AccessRights, (Read), (Write), (Execute), (Own))

variables
CurrentRoles
RoleAdmin
SessionAccesses
SessionAdminAccesses

invariants
@CurrentRoles_type CurrentRoles \subseteq Roles
@RoleAdmin_type RoleAdmin \in CurrentRoles \rightarrow BOOL
@RoleRights_type RoleRights \in CurrentRoles \rightarrow (CurrentEntities \rightarrow AccessRights)
@RoleAdminRights_type RoleAdminRights \in CurrentRoles \rightarrow (CurrentRoles \rightarrow AccessRights)
@SessionAccesses_type SessionAccesses \in CurrentSessions \rightarrow (CurrentEntities \rightarrow Accesses)
@SessionAdminAccesses_type SessionAdminAccesses \in CurrentSessions \rightarrow (CurrentRoles \rightarrow Accesses)



```
invariants
@CurrentEntities_type CurrentEntities  $\subseteq$  Entities // E = O  $\cup$  C
@CurrentObjects_type CurrentObjects  $\subseteq$  CurrentEntities // O
@CurrentContainers_type CurrentContainers  $\subseteq$  CurrentEntities // C
@CurrentEntities_content (CurrentObjects  $\cup$  CurrentContainers = CurrentEntities)  $\wedge$  (CurrentObjects  $\cap$  CurrentContainers =  $\emptyset$ )
@EholeEntities_type EholeEntities  $\subseteq$  CurrentEntities // E_HOLE
@CurrentSessions_type CurrentSessions  $\subseteq$  Sessions // S
@CurrentUserAccounts_type CurrentUserAccounts  $\subseteq$  UserAccounts // U
@CurrentRoles_type CurrentRoles  $\subseteq$  Roles // R

// Entities properties
@EntityIntegrity_type EntityIntegrity  $\in$  CurrentEntities  $\rightarrow$  Integrity // i, e: E  $\rightarrow$  LI (для уровней целостности)
@EntityCnfLevel_type EntityCnfLevel  $\in$  CurrentEntities  $\rightarrow$  CnfLevels // f, e: E  $\rightarrow$  LC (для уровней конфиденциальности)
@EntityCnfCats_type EntityCnfCats  $\in$  CurrentEntities  $\rightarrow$  P(CnfCategories) // f, e: E  $\rightarrow$  LC (для категорий конфиденциальности)
@ContainerContent_type ContainerContent  $\in$  CurrentContainers  $\rightarrow$  (Names  $\rightarrow$  CurrentEntities) // C  $\rightarrow$  NAMES  $\times$  LI
@Ccr_type Ccr  $\in$  CurrentEntities  $\rightarrow$  BOOL // CCR: E  $\rightarrow$  {true, false}
@CcrI_type CcrI  $\in$  CurrentEntities  $\rightarrow$  BOOL // CCR_I: E  $\rightarrow$  {true, false}
@Shared_type Shared  $\in$  CurrentEntities  $\rightarrow$  BOOL // shared_container: C  $\rightarrow$  {true, false}

// Session properties:
@SessionUser_type SessionUser  $\in$  CurrentSessions  $\rightarrow$  CurrentUserAccounts // user: S  $\rightarrow$  U
@SessionProfiles_type SessionProfiles  $\in$  CurrentSessions  $\rightarrow$  P(Entities) // S  $\rightarrow$  {s}
@SessionExfiles_type SessionExfiles  $\in$  CurrentSessions  $\rightarrow$  P(Entities) // S  $\rightarrow$  {s}
@SessionAccesses_type SessionAccesses  $\in$  CurrentSessions  $\rightarrow$  (CurrentEntities  $\rightarrow$  Accesses) // A  $\subseteq$  S  $\times$  E  $\times$  R_s
@SessionAdminAccesses_type SessionAdminAccesses  $\in$  CurrentSessions  $\rightarrow$  (CurrentRoles  $\rightarrow$  Accesses) // AA
@SessionIntegrity_type SessionIntegrity  $\in$  CurrentSessions  $\rightarrow$  Integrity // i, s: S  $\rightarrow$  LI (для уровней целостности)
@SessionCnfLevel_type SessionCnfLevel  $\in$  CurrentSessions  $\rightarrow$  CnfLevels // f, s: S  $\rightarrow$  LC (для уровней конфиденциальности)
@SessionCnfCats_type SessionCnfCats  $\in$  CurrentSessions  $\rightarrow$  P(CnfCategories) // f, s: S  $\rightarrow$  LC (для категорий конфиденциальности)
@SessionOwner_type SessionOwner  $\in$  CurrentSessions  $\rightarrow$  CurrentRoles

// User properties:
@UserProfiles_type UserProfiles  $\in$  CurrentUserAccounts  $\rightarrow$  P(CurrentEntities) // U  $\rightarrow$  {u}
```

Пример задания имен и иерархии в сущностях

$entity_name: C \times E \rightarrow 2^{NAMES}$ – функция имен сущностей в составе сущностей-контейнеров.

В исходной математической нотации:

$H_E: E \rightarrow 2^E$ – функцию иерархии сущностей (сопоставляющую каждой сущности $e \in E$ множество сущностей $H_E(e) \subset E$, непосредственно в ней содержащихся), удовлетворяющую условиям:

- если сущность $e \in H_E(c)$, то $e < c$, при этом, если $e \in C$, то не существует сущности-контейнера $d \in C$ такой, что $e < d < c$;
- для любых сущностей $e_1, e_2 \in E$, $e_1 \neq e_2$ выполняется равенство $H_E(e_1) \cap H_E(e_2) \cap C = \emptyset$;
- если $o \in O$, то справедливо равенство $H_E(o) = \emptyset$.

В адаптированной математической нотации:

для любых контейнеров $c, c' \in C$ по определению выполняются условия:

- $|entity_name(c, c')| \leq 1$;
- если $c \neq c'$ и существует $c'' \in C$ такой, что $entity_name(c, c'') \neq \emptyset$, то $entity_name(c', c'') = \emptyset$;
- существует единственная сущность-«корневой контейнер» $ROOT \in C$ такая, что $entity_name(c, ROOT) = \emptyset$ и, если $c \neq ROOT$, то существует единственная последовательность контейнеров $c_1 = c, c_2, \dots, c_n = ROOT \in C$ такая, что $n \geq 2$ и $entity_name(c_i, c_{i-1}) \neq \emptyset$, где $1 < i \leq n$.

$H_E: E \rightarrow 2^E$, где для $e \in E$ выполняется $H_E(e) = \{e' \in E \mid entity_name(e, e') \neq \emptyset\}$.

Rodin Platform. Пример задания правила

The screenshot displays the Rodin Platform IDE interface. The main window shows a rule definition for the event `create_hard_link`. The rule is defined in the `DynamicPart` component of the `dmodel` package.

```
event create_hard_link
  any session object container // parameters
  name object_container

  where

    @grd1 object ∈ CurrentObjects // 3
    @grd2 session ∈ CurrentSessions // 1
    @grd3 container ∈ CurrentContainers // 4
    @grd4 name ∈ Names \ dom(ContainerContent(container)) // **
    @grd5 container → WriteA ∈ SessionAccesses(session) // 6
    @grd6 (EntityCnfLevel(object) = EntityCnfLevel(container) ∧ EntityCnfCats(object) = EntityCnfCats(container)
      ∧ EntityCnfLevel(object) = SessionCnfLevel(session) ∧ EntityCnfCats(object) = SessionCnfCats(session))
      ∨ (EntityCnfLevel(container) ≥ EntityCnfLevel(object) ∧ EntityCnfCats(object) ⊆ EntityCnfCats(container)
      ∧ DowngradeAR → ReadA ∈ SessionAdmAccesses(session)) // 8, 9, 10, 11
    @grd7 EntityIntegrity(container) ≥ SessionIntegrity(session) // 12
    @grd8 EntityIntegrity(container) ≥ EntityIntegrity(object) // 13
    @grd9 object_container ∈ dom(ContainerContent) ∧ object ∈ ran(ContainerContent(object_container)) // ?
    @grd10 ∃ S: S ⊆ CurrentContainers ∧ {y → x | x ∈ CurrentContainers ∧ y ∈ ran(ContainerContent(x))} [S] = S ∪ {object}
      ∧ (∀ o: o ∈ S ⇒ ((SessionCnfLevel(session) ≥ EntityCnfLevel(o)
      ∧ EntityCnfCats(o) ⊆ SessionCnfCats(session))
      ∨ Ccr(o) = FALSE))
      ∧ ((SessionIntegrity(session) ≥ EntityIntegrity(o))
      ∨ Ccri(o) = FALSE) ) // 5

  then
    @act1 ContainerContent := ContainerContent ∃ {container → (ContainerContent(container) ∪ {name → object})} //
  end
```

The interface includes a menu bar (File, Edit, Navigate, Search, Project, Run, ProB, BMotion Studio, Window, Help), a toolbar, a left sidebar with a project tree, a main editor window, a right sidebar with an outline, and a bottom status bar with tabs for Problems, Properties, Tasks, and Symbols.

Frama-C. Пример предусловия

```
File Project Analyses Help
Source file
mac_equ
mac_file_permiss
mac_label_cpy
mac_label_equ
mac_label_notg
parsec_mac_wri
read_bit
WP
Model... Typed
Script... (None
Provers... Alt-E
RTE Split
Invariants
Steps 0
Timeout 10
Slicing
Activate None
Enable
Libraries
Impact
Enable
Slicing after imp
Follow focus
Occurrence
Current var: None
Enable
Follow focus
Read Write
Value
0 slevel
main
Metrics
parsec.c
151
152 /*@
153 requires \valid(s) && \valid(o);
154 requires read_bit(mask) || write_bit(mask) || exec_bit(mask) || writeup_bit(mask);
155 terminates \true;
156 assigns \nothing;
157
158 behavior DENY:
159     assumes read_bit(mask) || exec_bit(mask) || write_bit(mask);
160     assumes !allow(s, o, mask);
161     ensures DENY: \result == -EPERM;
162
163 behavior ALLOW:
164     assumes read_bit(mask) || exec_bit(mask) || write_bit(mask);
165     assumes allow(s, o, mask);
166     //assumes read_bit(mask) ==> is_readable(s, o);
167     //assumes exec_bit(mask) ==> is_executable(s, o);
168     //assumes write_bit(mask) && (writeup_bit(mask) || (parsec_mac_write_up.counter != 0)) ==> is_w
169     //assumes write_bit(mask) && (!writeup_bit(mask) && (parsec_mac_write_up.counter == 0)) ==> is_w
170     ensures ALLOW: \result == 0;
171
172 behavior DEFAULT:
173     assumes !read_bit(mask) && !exec_bit(mask) && !write_bit(mask);
174     ensures ALLOW: \result ==0;
175
176 complete behaviors;
177 disjoint behaviors;
178 */
179 int mac_file_permission(const parsec_mac_t *s, const parsec_mac_label_t *o,
180                        opmask_t mask)
181 {
182     ASSERT(s);
183     ASSERT(o);
184
185     if (mask & MAY_READ) {
186         if ( (!(o->type & MAC_ATTR_IGNORER_LVL) &&
187              (s->level < o->mac.level)) ||
188              (!(o->type & MAC_ATTR_IGNORER_CAT) &&
189              (s->category & o->mac.category) != o->mac.category) )
190             return -EPERM;
191     }
192
193     if (mask & MAY_WRITE) {
194         int may_write_up = atomic_read(&parsec_mac_write_up);
195     }
196 }
Information Messages Console Properties WP Goals
Function 'mac_file_permission'
```

Спасибо за
внимание!