

Обеспечение доверия к системному ПО на примере ОС Astra Linux

Девянин П.Н., научный руководитель ГК Astra Linux

ОСНОВНЫЕ ФУНКЦИИ ВСТРОЕННЫХ СЗИ СЕРТИФИЦИРОВАННОЙ ПО 1 КЛАССУ ЗАЩИТЫ ОСН АСТРА LINUX SPECIAL EDITION

02

РЕЖИМ «КИОСК»

Белый список приложений

ДИСКРЕЦИОННОЕ УПРАВЛЕНИЕ ДОСТУПОМ

Защита информации одного
уровня конфиденциальности

МАНДАТНОЕ УПРАВЛЕНИЕ ДОСТУПОМ

Защита информации различных
уровней конфиденциальности

ЗАМКНУТАЯ ПРОГРАММНАЯ СРЕДА

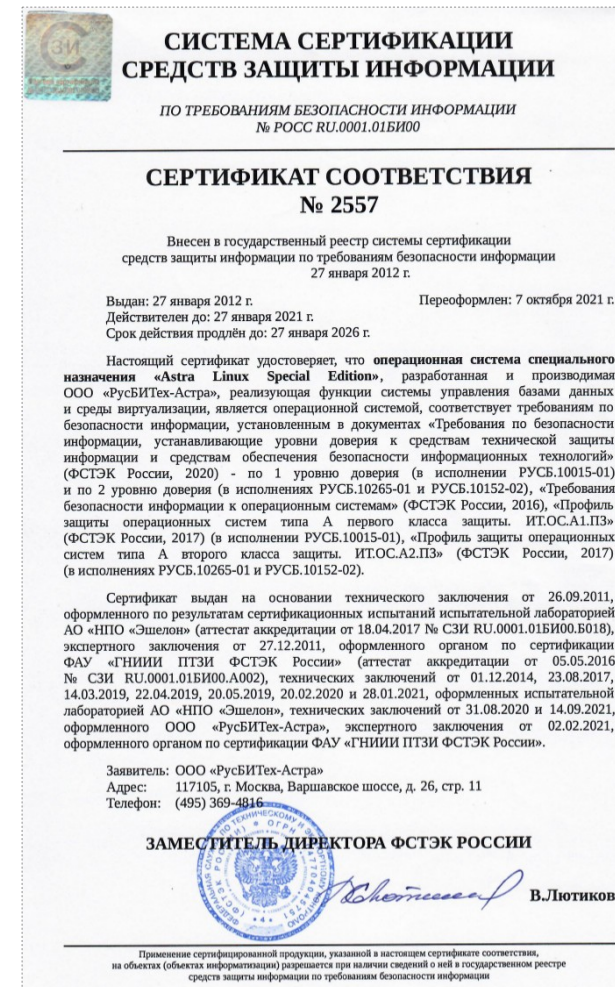
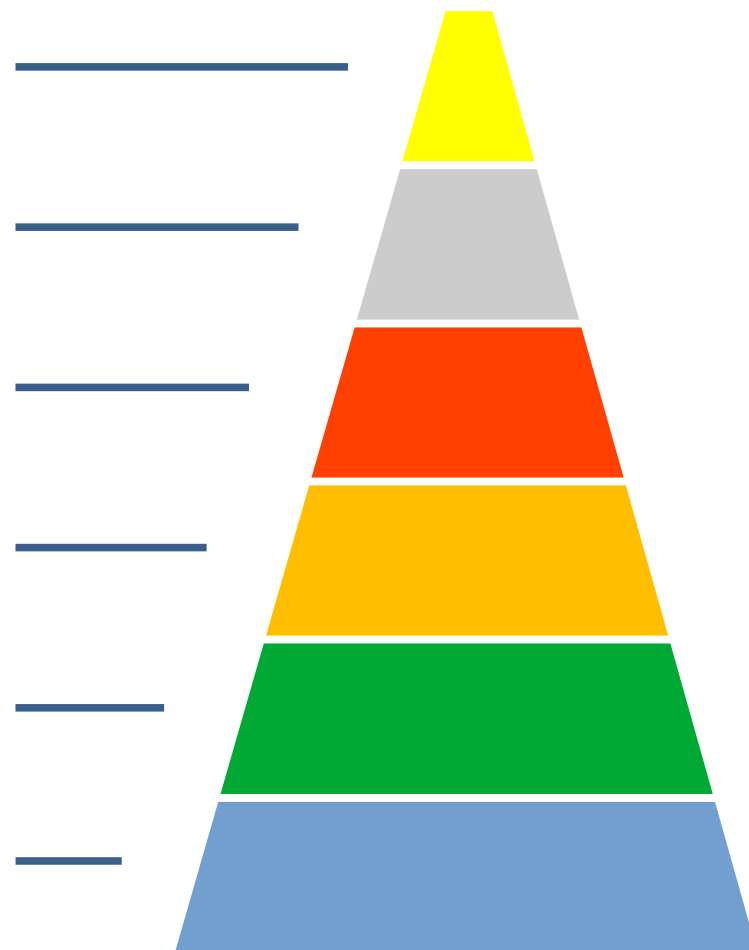
Защита приложений от подмены

МАНДАТНЫЙ КОНТРОЛЬ ЦЕЛОСТНОСТИ

Защита целостности программной среды,
в т.ч. от вирусов, закладок и типовых атак

ОПЕРАЦИОННАЯ СИСТЕМА

Настройка состава устанавливаемого ПО
и расширенный аудит событий безопасности



ТРЕБОВАНИЯ ПО БЕЗОПАСНОСТИ ИНФОРМАЦИИ, УСТАНОВЛИВАЮЩИЕ 6 УРОВНЕЙ ДОВЕРИЯ (ПРИКАЗ ФСТЭК РОССИИ ОТ 02.06.2020 № 76)

03

4 УРОВЕНЬ ДОВЕРИЯ (ОБЪЕКТЫ КИИ 1 КАТЕГОРИИ, ГИС 1 КЛАССА ЗАЩИЩЕННОСТИ)

- > модель безопасности, включая реализуемые политики управления доступом;
- > идентификация и анализ скрытых каналов по памяти.

3 УРОВЕНЬ ДОВЕРИЯ (ИС, В КОТОРЫХ ОБРАБАТЫВАЕТСЯ ИНФОРМАЦИЯ, СОДЕРЖАЩАЯ СЕКРЕТНЫЕ СВЕДЕНИЯ)

- > верификация модели безопасности с использованием инструментальных средств;
- > формальное (математическое) описание подсистем, реализующих функции безопасности;
- > идентификация и анализ скрытых каналов по времени.

2 УРОВЕНЬ ДОВЕРИЯ (ИС, В КОТОРЫХ ОБРАБАТЫВАЕТСЯ ИНФОРМАЦИЯ, СОДЕРЖАЩАЯ СОВЕРШЕННО СЕКРЕТНЫЕ СВЕДЕНИЯ)

- > формальное (математическое) описание модулей, реализующих функции безопасности.

1 УРОВЕНЬ ДОВЕРИЯ (ИС, В КОТОРЫХ ОБРАБАТЫВАЕТСЯ ИНФОРМАЦИЯ, СОДЕРЖАЩАЯ СВЕДЕНИЯ ОСОБОЙ ВАЖНОСТИ)

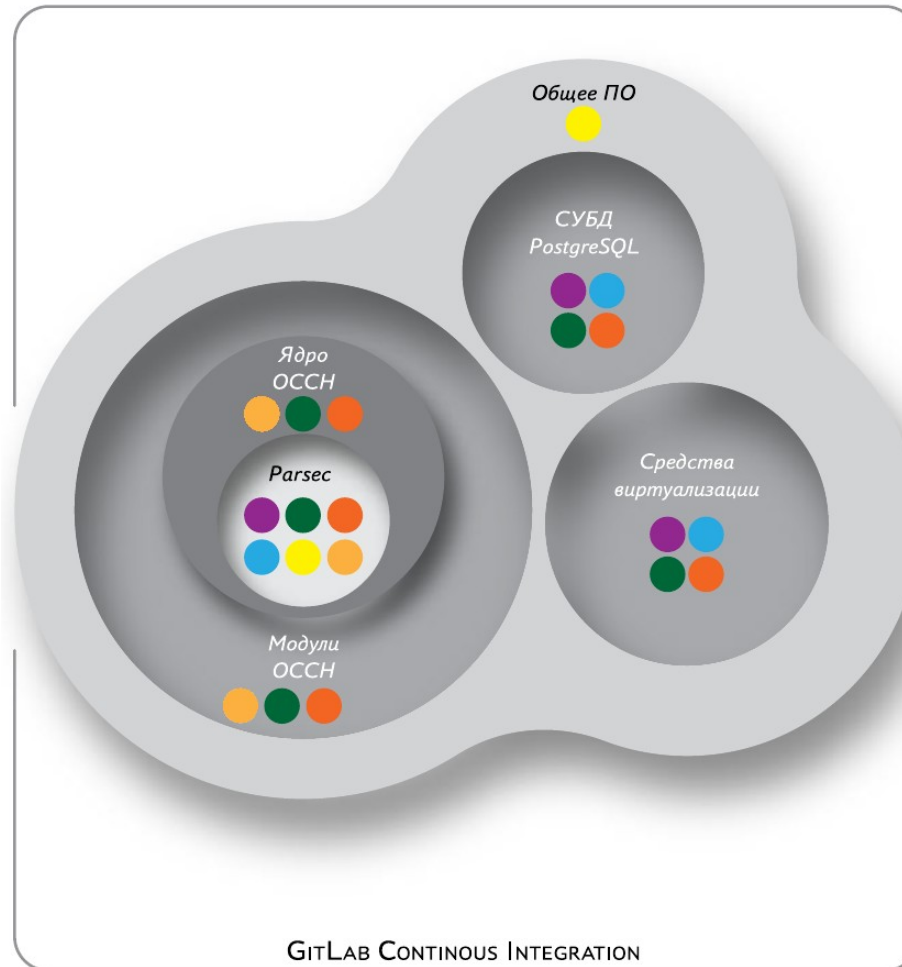
- > идентификация и анализ скрытых статистических каналов.

ВЕРИФИКАЦИЯ И АНАЛИЗ КОДА ОССН В СООТВЕТСТВИИ С МЕТОДИКОЙ ВЫЯВЛЕНИЯ УЯЗВИМОСТЕЙ И НЕДЕКЛАРИРОВАННЫХ ВОЗМОЖНОСТЕЙ

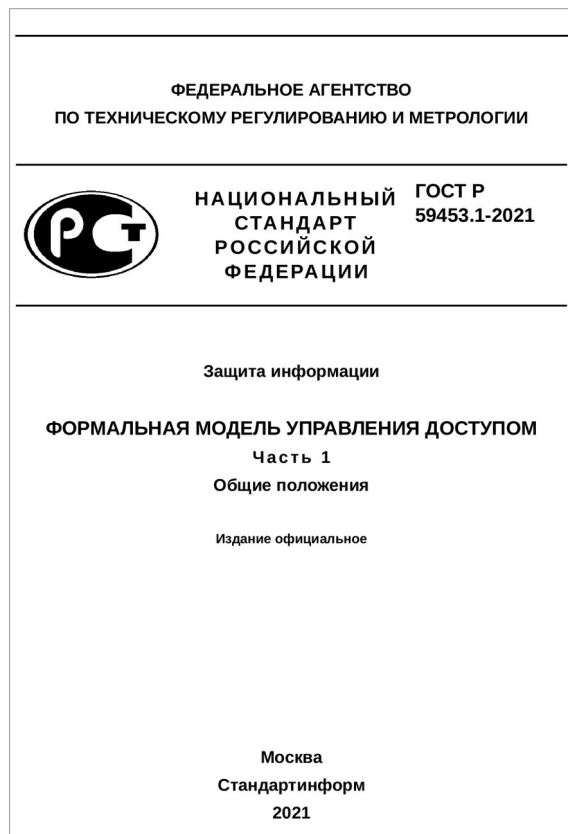
04

Для верификации программного кода компонент ОССН в зависимости от степени их важности для обеспечения безопасности (от основы **поверхности атаки** — модуля **PARSEC**) применяются:

- **Иерархическая МРОСЛ ДП-модель** — основа реализованных в ОССН мандатных управления доступом и контроля целостности;
- Комплекс инструментальных средств **верификации** МРОСЛ ДП-модели и её реализации в программном коде ОССН (Rodin, ProB, Frama-C);
- Инструментальные средства **статического анализа** программного кода (Svace, Clang Static Analyzer, cppcheck, АК-BC);
- Инструментальные средства **динамического анализа** (фаззинга) программного кода (Crusher, AFL, Syzkaller4Astra);
- Инструментальное средство **сбора трасс и анализа помеченных данных** (Блесна);
- Система **непрерывной разработки и интеграции** GitLab.

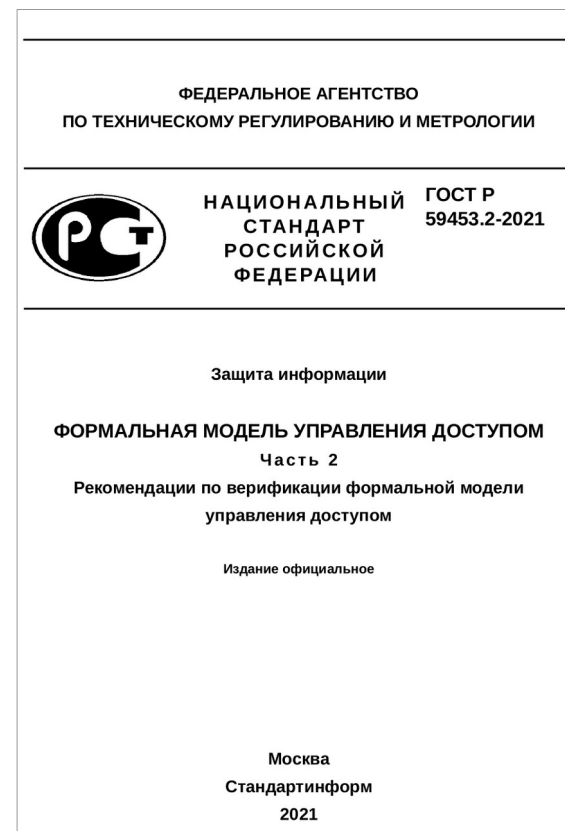


- МРОСЛ ДП-модель
- Rodin, ProB, Frama-C
- Svace, Clang SA, cppcheck, АК-BC
- Syzkaller4Astra
- Блесна
- Crusher, AFL



СОДЕРЖАНИЕ

1. Область применения
2. Нормативные ссылки
3. Термины и определения
4. Общие положения
5. Описание состояний в рамках формальной модели управления доступом
6. Описание правил перехода из состояний в состояния в рамках формальной модели управления доступом
7. Доказательство выполнения условий безопасности



СОДЕРЖАНИЕ

1. Область применения
 2. Нормативные ссылки
 3. Термины и определения
 4. Общие положения
 5. Выбор инструментальных средств верификации формальной модели управления доступом
 6. Формализованное (машиночитаемое) описание формальной модели управления доступом
 7. Верификация формализованного (машиночитаемого) описания формальной модели управления доступом
- Приложение А (справочное). Примеры перевода элементов математического описания формальной модели управления доступом в формализованное (машиночитаемое) описание

- > **формальная модель управления доступом:** Математическое или формализованное (машиночитаемое, пригодное для автоматизированной обработки) описание средства защиты информации и компонентов среды его функционирования, предоставление доступов между которыми регламентируется политиками управления доступом, реализуемыми этим средством защиты информации.

- > **объект доступа, ассоциированный функционально с субъектом доступа:** Объект доступа, содержание информации в котором влияет на функциональность субъекта доступа, и информационный поток по памяти к этому объекту от другого субъекта доступа позволяет второму субъекту доступа получить управление первым субъектом доступа.

- > **политика мандатного контроля целостности:** Политика управления доступом, при реализации которой задаются классификационные метки (уровни целостности): каждому объекту и субъекту доступа присваивается уровень целостности; субъект доступа может получить доступ к объекту доступа или другому субъекту доступа только в случае, когда выполняются следующие правила:
 - при получении доступа на запись к объекту доступа уровень целостности субъекта доступа должен быть не ниже уровня целостности объекта доступа;
 - доступ субъекта доступа к объекту или другому субъекту доступа не приводит к получению субъектом доступа управления некоторым субъектом доступа, уровень целостности которого не сравним или выше уровня целостности первого субъекта доступа.



ПРИМЕРЫ ОПРЕДЕЛЕНИЯ ЭЛЕМЕНТОВ МОДЕЛИ ДЛЯ УРОВНЯ 1.2 (РОЛЕВОЕ УПРАВЛЕНИЕ ДОСТУПОМ В СУБД POSTGRESQL)

08

DB_R – множество ролей СУБД;

$DB_ADMIN_PRIVILEGES = \{SUPERUSER, CREATEROLE, CREATEDB, LOGIN, REPLICATION, INHERIT\}$ –
множество административных привилегий СУБД;

$DB_AP: DB_R \rightarrow 2^{DB_ADMIN_PRIVILEGES}$ – функция административных привилегий ролей СУБД;

$db_login: S \rightarrow \{r \in DB_R : LOGIN \in DB_AP(r)\}$ – функция роли входа субъект-сессии в СУБД;

$db_inherit: DB_R \rightarrow 2^{DB_R}$ – функция наследования привилегий ролей к элементам СУБД

$db_with_admin_option: DB_R \rightarrow 2^{DB_R}$ – функция управления подчинённостью ролей в иерархии;

$DB_E = DB_O \cup DB_C$ – множество элементов СУБД, не являющихся ролями, где DB_O – множество
элементов-объектов СУБД, DB_C – множество элементов-контейнеров СУБД;

$DB_PRIVILEGES = \{SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER, USAGE, CREATE, CONNECT, TEMPORARY, TEMP, EXECUTE, OWN\}$ – множество видов привилегий СУБД;

$DB_E_E \subseteq DB_E$ – множество сущностей СУБД;

$db_path: DB_E_E \rightarrow 2^E$ – функция, ставящая в соответствие сущностям СУБД сущности ОССН, в которых
они содержатся;

$db_privileges: DB_R \rightarrow 2^{(DB_E \times DB_PRIVILEGES) \times DB_R}$ – функция привилегий к элементам СУБД ролей СУБД;

$A \subseteq S \times (E \cup DB_E_E) \times R_a$ – множество доступов субъект-сессий к сущностям или сущностям СУБД;

$AA \subseteq S \times (R \cup NR \cup AR \cup DB_R) \times R_a$ – множество доступов субъект-сессий к ролям, запрещающим ролям,
административным ролям или ролям СУБД.

ПРИМЕР УСЛОВИЙ БЕЗОПАСНОСТИ ДЛЯ УРОВНЯ 2.2 (МАНДАТНЫЙ КОНТРОЛЬ ЦЕЛОСТНОСТИ В СУБД POSTGRESQL)

09

Теорема т.Ц.01.БДЦ. 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 $N \geq 0$, и в каждом состоянии G_N для каждой субъект-сессии $s \in S_N$, сущности или сущности СУБД $e \in E_N \cup DB_E_E_N$ выполняются следующие условия.

Условие Ц.1. (корректность уровней целостности сущностей, функционально ассоциированных с субъект-сессиями) Если $e \in [s]$, то выполняется условие $i_{sN}(s) \leq i_{eN}(e)$.

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

Условие Ц.3.БДЦ. (функциональная и параметрическая корректность всех доверенных субъект-сессий относительно всех доверенных субъект-сессий и сущностей ОССН и СУБД) Для всех субъект-сессий $s \in S_N$ таких, что $i_low < i_{sN}(s)$, выполняются условия $\{s' \in S_N \mid i_low < i_{sN}(s') \leq i_{sN}(s)\} \times (E_N \cup DB_E_E_N \cup S_N) \subset f_correct_N(s)$, $\{s' \in S_N \mid i_low < i_{sN}(s') \leq i_{sN}(s)\} \times (E_N \cup S_N) \subset p_correct_N(s)$.

Условие БДЦ.4. (неизменность множества доверенных субъект-сессий СУБД) Множество доверенных субъект-сессий СУБД не меняется на траекториях функционирования системы: $DB_L_{S_N} = DB_L_{S_0}$. Для каждой субъект-сессии $x, y \in S_N$ выполняется, если $y \in DB_L_{S_0} \cap de_facto_own_N(x)$, то $x \in DB_L_{S_0}$.

Тогда на этих траекториях система $\Sigma(G^*, OP, G_0)$ безопасна в смысле мандатного контроля целостности.

ФРАГМЕНТ ЗАДАНИЯ В ФОРМАЛИЗОВАННОЙ НОТАЦИИ УРОВНЯ 4.2 (МАНДАТНОГО УПРАВЛЕНИЯ ДОСТУПОМ С ПОТОКАМИ ПО ВРЕМЕНИ В СУБД POSTGRESQL)

The screenshot displays the Event-B Explorer interface. The left sidebar shows a project tree with folders for '1-C-RBAC', '2-C-MIC', '3-C-MLS', '5-C-DBMS-RBAC', '6-C-DBMS-MIC', '7-C-DBMS-MLS', and a sub-folder '1-M-RBAC' containing models '2-M-MIC', '3-M-MLS-MFlows', '4-M-MLS-MTFlows', '5-M-DBMS-RBAC', '6-M-DBMS-MIC', '7-M-DBMS-MLS', and '8-M-DBMS-MTFlows'. The main editor shows the following formal specification:

```
R_DBEFlowsType: R_DBEFlows ∈ Roles → (DBEntities ↔ {2}) not theorem >
DBR_RFlowsType: DBR_RFlows ∈ DBRoles → (Roles ↔ {2}) not theorem >
DBR_DBRFlowsType: DBR_DBRFlows ∈ DBRoles → (DBRoles ↔ {2}) not theorem >
dbCheckRightETFtype: dbCheckRightETF ∈ Subjects × DBEntities × AccessRights → P(DBRoles) not theorem >
dbCheckRightR1TFtype: dbCheckRightR1TF ∈ Subjects × DBRoles × AccessRights → P({PostgresAdmRole, PublicRole}) not theorem >
dbCheckRightR2TFtype: dbCheckRightR2TF ∈ Subjects × DBRoles × {Read} → P(AdmRoles) not theorem >
dbExecuteContainerTFtype: dbExecuteContainerTF ∈ Subjects × DBEntities → BOOL not theorem >
dbCheckRightETFFunc1: ∀s,e,ar · s ∈ Subjects ∧ e ∈ DBEntities ∧ ar ∈ {Execute,Write} ⇒ dbCheckRightETF(s ⇒ e ⇒ ar) = dbCheckRightECnf(s ⇒ e ⇒ ar) not theorem >
dbCheckRightETFFunc2: ∀s,e · s ∈ Subjects ∧ e ∈ DBEntities ∧ DBObjects ⇒ dbCheckRightETF(s ⇒ e ⇒ Read) = dbCheckRightECnf(s ⇒ e ⇒ Read) not theorem >
dbCheckRightETFFunc3: ∀s,e,r · s ∈ Subjects ∧ e ∈ DBEntities ∧ DBContainers ∧ r ∈ DBRoles ⇒ (r ∈ dbCheckRightETF(s ⇒ e ⇒ Read) ⇔ r ∈ dbCheckRightECnf(s ⇒ e ⇒ Read)) not theorem >
dbCheckRightETFFunc4: ∀s,e,r · s ∈ Subjects ∧ e ∈ DBEntities ∧ r ∈ DBRoles ⇒ (r ∈ dbCheckRightETF(s ⇒ e ⇒ Own) ⇔ r ∈ dbCheckRightECnf(s ⇒ e ⇒ Own)) not theorem >
dbCheckRightR1TFFunc1: ∀s,e · s ∈ Subjects ∧ e ∈ DBRoles ⇒ dbCheckRightR1TF(s ⇒ e ⇒ Execute) = dbCheckRightR1Cnf(s ⇒ e ⇒ Execute) not theorem >
dbCheckRightR1TFFunc2: ∀s,e,r · s ∈ Subjects ∧ e ∈ DBRoles ∧ r ∈ {PostgresAdmRole, PublicRole} ⇒ (r ∈ dbCheckRightR1TF(s ⇒ e ⇒ Read) ⇔ r ∈ dbCheckRightR1Cnf(s ⇒ e ⇒ Read)) not theorem >
dbCheckRightR1TFFunc3: ∀s,e,ar,r · s ∈ Subjects ∧ e ∈ DBRoles ∧ ar ∈ {Write, Own} ∧ r ∈ {PostgresAdmRole, PublicRole} ⇒ (r ∈ dbCheckRightR1TF(s ⇒ e ⇒ Read) ⇔ r ∈ dbCheckRightR1Cnf(s ⇒ e ⇒ Read)) not theorem >
dbCheckRightR2TFFunc: ∀s,e,r · s ∈ Subjects ∧ e ∈ DBRoles ∧ r ∈ AdmRoles ⇒ (r ∈ dbCheckRightR2TF(s ⇒ e ⇒ Read) ⇔ r ∈ dbCheckRightR2Cnf(s ⇒ e ⇒ Read)) not theorem >
dbExecuteContainerTFFunc: ∀s,e · s ∈ Subjects ∧ e ∈ DBEntities ⇒ (dbExecuteContainerTF(s ⇒ e) = TRUE ⇔ e = DBCluster ∨ (e ≠ DBCluster ∧ (∃c · c ∈ DBContainers ∧ DBEntities ∧ c = dbParentE(e) ∧ (∀o · o ∈ dbEPath(c) ∧ DBEntities ⇒ dbCheckRightETF(s ⇒ o ⇒ Execute) ≠ ∅ ∧ (dbEntityCnf(o) ⊆ SubjectCnf(s) ∨ dbCCR_E(o) = FALSE ∨ PostgresAdmRole ⇒ ReadA ∈ DBSubjectAdmAccesses(s)))))) not theorem >
dbExecuteContainerLemma: ∀s,e · s ∈ Subjects ∧ e ∈ DBEntities ⇒ (dbExecuteContainerTF(s ⇒ e) = TRUE ⇔ dbExecuteContainerCnf(s ⇒ e)) not theorem >
DBSubjectAdmAccesses4: ∀s, r · s ∈ Subjects ∧ r ∈ DBRoles ∧ r ⇒ WriteA ∈ DBSubjectAdmAccesses(s) ⇒ dbRoleCnf(r) = SubjectCnf(s) ∨ PostgresAdmRole ⇒ ReadA ∈ DBSubjectAdmAccesses(s) not theorem >

dbEEFlows2: ∀x, y · x ∈ dom(dbEEFlows) ∧ y ⇒ 2 ∈ dbEEFlows(x) ∧ WithoutCoop = TRUE ⇒ EntityCnf(x) ⊆ EntityCnf(y) not theorem >Определение о.Ц.06.КП.КВ. Условие Ц.4.КП.КВ.
dbESFlows2: ∀x, y · x ∈ dom(dbESFlows) ∧ y ⇒ 2 ∈ dbESFlows(x) ∧ WithoutCoop = TRUE ⇒ EntityCnf(x) ⊆ SubjectCnf(y) not theorem >Определение о.Ц.06.КП.КВ. Условие Ц.4.КП.КВ.
dbERFlows2: ∀x, y · x ∈ dom(dbERFlows) ∧ y ⇒ 2 ∈ dbERFlows(x) ∧ WithoutCoop = TRUE ⇒ EntityCnf(x) ⊆ RoleCnf(y) not theorem >Определение о.Ц.06.КП.КВ. Условие Ц.4.КП.КВ.
dbSEFlows2: ∀x, y · x ∈ dom(dbSEFlows) ∧ y ⇒ 2 ∈ dbSEFlows(x) ∧ WithoutCoop = TRUE ⇒ SubjectCnf(x) ⊆ EntityCnf(y) not theorem >Определение о.Ц.06.КП.КВ. Условие Ц.4.КП.КВ.
dbSSFlows2: ∀x, y · x ∈ dom(dbSSFlows) ∧ y ⇒ 2 ∈ dbSSFlows(x) ∧ WithoutCoop = TRUE ⇒ SubjectCnf(x) ⊆ SubjectCnf(y) not theorem >Определение о.Ц.06.КП.КВ. Условие Ц.4.КП.КВ.
dbSRFlows2: ∀x, y · x ∈ dom(dbSRFlows) ∧ y ⇒ 2 ∈ dbSRFlows(x) ∧ WithoutCoop = TRUE ⇒ SubjectCnf(x) ⊆ RoleCnf(y) not theorem >Определение о.Ц.06.КП.КВ. Условие Ц.4.КП.КВ.
dbREFlows2: ∀x, y · x ∈ dom(dbREFlows) ∧ y ⇒ 2 ∈ dbREFlows(x) ∧ WithoutCoop = TRUE
```

ВЕРИФИКАЦИЯ МРОСЛ ДП-МОДЕЛИ (УРОВНЯ 2.2) В ФОРМАЛИЗОВАННОЙ НОТАЦИИ НА ЯЗЫКЕ МЕТОДА EVENT-B ИНСТРУМЕНТОМ RODIN

The screenshot displays the Rodin IDE interface with three main panels:

- Proof Tree (Left):** A hierarchical tree of proof goals. The root goal is $\forall s, e$. It branches into goal , ovr , remove , hyp , eh , ah , and goal . The tree shows various logical steps and sub-goals, with many nodes marked as verified (green checkmarks).
- Goal (Bottom Left):** A window showing the current goal being verified: $\forall s, e .$
 $seSubjects \wedge$
 $eeDBEntities \wedge$
 $e \mapsto WriteAe(DBSubjectAccesses\{subject \mapsto DBSubjectAccesses(subject)u\{dbEntityInt(e) \subseteq SubjectInt(s)$
- Event-B Explorer (Right):** A list of verified events. The selected event is `access_write_db_entity/DBSubjectAccesses1/INV`. Other events include `access_read_role/dbCheckRightR2IntType/INV`, `delete_access_entity/dbE5Flows1/INV`, and `access_read_db_entity/DBE_SFlowsType/INV`.

The interface also includes a **Search Hypothesis** window at the bottom left and a **Selected Hypotheses** window at the bottom center. The bottom right panel shows **Symbols**, **Progress**, and **Type Environment** tabs, with the message "No operations to display at this time."

ВЕРИФИКАЦИЯ МРОСЛ ДП-МОДЕЛИ (УРОВНЯ 1.1) В ФОРМАЛИЗОВАННОЙ НОТАЦИИ ПО МЕТОДУ ПРОВЕРКИ МОДЕЛЕЙ ИНСТРУМЕНТОМ PROB

The screenshot displays the Rodin tool interface for the verification of the 1-M-RBAC model. The interface is divided into several panes:

- Events:** A list of events with their parameters. Events like `rename_role`, `read_container_entity`, etc., are marked with a red circle, while `set_container_attr`, `create_first_subject`, and `create_subject` are marked with a green triangle.
- State:** A table showing the current state of the model. The table has columns for Name, Value, and Previous value.
- History:** A log of events, showing the sequence of operations performed during the verification process, such as `grant_rights`, `create_object`, and `access_write_entity`.
- Summary:** A green bar at the bottom indicates "Invariants ok" and "No event errors detected". A yellow bar below it shows "Max. nr. events reached".

Name	Value	Previous value
RoleAdmRights	{{(ARolesAR→{(ARole: {(ARolesAR→{(ARolesAR→Execute),(ARolesAR→Own),(Enti	
RoleName	{{(ARolesAR→Names2 {{(ARolesAR→Names22),(EntitiesAR→Names21),(NRolesAR	
RoleRights	{{(ARolesAR→∅),(Ent {{(ARolesAR→∅),(EntitiesAR→∅),(NRolesAR→∅),(RolesA	
Roles	{ARolesAR,EntitiesAR {ARolesAR,EntitiesAR,NRolesAR,RolesAR,SubjectsAR,Users,	
SParent	∅	∅
SharedC	{{(Root→TRUE}}	{{(Root→TRUE}}
SharedR	{{(ARolesAR→TRUE),(f {{(ARolesAR→TRUE),(EntitiesAR→TRUE),(NRolesAR→TRUE),(
SubjectAccesses	{{(SRoot→{(Root→Wri {{(SRoot→{(Root→WriteA)}}}}	
SubjectAdmAccesses	{{(SRoot→{(RolesU7→ {{(SRoot→{(RolesU7→ReadA),(RolesU7→WriteA),(RolesU11	
SubjectNOwners	{{(SRoot→∅}}	{{(SRoot→∅}}
SubjectOwner	{{(SRoot→{RolesU12};	{{(SRoot→{RolesU12}}
SubjectUser	{{(SRoot→RedUser}}	{{(SRoot→RedUser}}
Subjects	{SRoot}	{SRoot}
UserAccs	{LowUser,RedUser}	{LowUser,RedUser}
UserAdmRole	{{(LowUser→RolesU8; {{(LowUser→RolesU8),(RedUser→RolesU11}}	
UserOrdRole	{{(LowUser→RolesU9; {{(LowUser→RolesU9),(RedUser→RolesU12}}	

ВЕРИФИКАЦИЯ СПЕЦИФИКАЦИЙ НА ЯЗЫКЕ ACSL ПРОГРАММНОЙ РЕАЛИЗАЦИИ МРОСЛ ДП-МОДЕЛИ ИНСТРУМЕНТОМ FRAMA-C

The screenshot displays the FRAMA-C IDE interface. On the left, there is a sidebar with several sections: 'Context' (Unproved goals, All goals), 'Strategies' (Auto level 1, Compute, Inline, Inline all, Intro premises, Split, Split, prove, astraver), 'Provers' (Alt-Ergo (2.3.2), CVC4 (1.7), CVC4 (1.7 noBV), Coq (8.11.2), Eprover (2.4), Z3 (4.4.1), Z3 (4.4.1 noBV)), 'Tools' (Edit, Replay, Remove, Clean, Save, Fastest), and 'Proof monitoring' (Waiting: 0, Scheduled: 0, Running: 0, Interrupt).

The main area is divided into three panes. The top-left pane shows a 'Theories/Goals' tree with the following structure:

Theories/Goals	Status	Time
module_c	✓	0.48
pdp_lev_permission, safety	✓	0.03
1. Function pdp_lev_permissio	✓	0.03
Alt-Ergo (2.3.2)	✓	0.03 (steps: 146)
pdp_lev_permission, behaviors	✓	0.43
1. Function pdp_lev_permissio	✓	0.43
Alt-Ergo (2.3.2)	✓	0.43 (steps: 3164)
Axiomatic_ParsecTypesTheory	✓	0.02
1. Lemma MacLabelEQ	✓	0.02
Alt-Ergo (2.3.2)	✓	0.02 (steps: 79)

The top-right pane shows the 'Source code' editor with the following ACSL annotations and code:

```
220     getSecFlagCCNRI(m1) == getSecFlagCCNRI(m2));
221
222   }
223 */
224
225 /*@ axiomatic PermissionModeBits {
226   predicate read_bit(int mode) = (mode & 0x00000004);
227   predicate write_bit(int mode) = (mode & 0x00000002);
228   predicate exec_bit(int mode) = (mode & 0x00000001);
229   predicate create_bit(int mode) = (mode & 0x00000100);
230   predicate container_bit(int mode) = (mode & 0x08000);
231   predicate ignore_int_bit(int mode) = (mode & 0x40000);
232   predicate ignore_lev_bit(int mode) = (mode & 0x10000);
233   predicate ignore_cat_bit(int mode) = (mode & 0x20000);
234 }*/
235
236 /*@ axiomatic PdpLevPermission {
237   // Предикат, определяющий, даётся ли доступ субъекту slev к объекту olev.
238   // mode определяет, какой вид доступа запрошен субъектом.
239   predicate PdpLevPermission(MacLabelCnfLevel slev,
240                             MacLabelCnfLevel olev,
241                             int mode) =
242     ignore_lev_bit(mode) ||
243     (read_bit(mode) ==> slev >= olev) &&
244     (write_bit(mode) ==> slev == olev) &&
245     (exec_bit(mode) ==> slev >= olev);
246
247   // Логическая функция моделирует функцию pdp_lev_permission
248   logic integer pdp_lev_permission(MacLabelCnfLevel slev,
249                                   MacLabelCnfLevel olev,
250                                   int mode) =
251     PdpLevPermission(slev, olev, mode) ? 0 : -1;
252 }*/
253
254 /*@ assigns \nothing;
255   ensures \result == pdp_lev_permission((MacLabelCnfLevel) slev, (MacLabelCnfLevel) olev, mode);
256 */
257 int pdp_lev_permission(PDP_LEV_T slev, PDP_LEV_T olev, int mode)
258 {
259   if ((mode & 0x10000)) return 0;
260   if ((mode & 0x00000004) && (slev < olev)) return -1;
261   if ((mode & 0x00000002) && (slev != olev)) return -1;
262   if ((mode & 0x00000001) && (slev < olev)) return -1;
263
264   return 0;
265 }
```

СТАТИЧЕСКИЙ АНАЛИЗ КОДА ОССН С ИСПОЛЬЗОВАНИЕМ ИНСТРУМЕНТА CLANG STATIC ANALYZER

14

```
2788 static void
2789 _ignore_completion_names (names, name_func)
2790     char **names;
2791     sh_ignore_func_t *name_func;
2792 {
2793     char **newnames;
2794     int idx, nidx;
2795     char **oldnames;
2796
2797     9 -- 'oldnames' declared without an initial value --
2798     int oidx;
2799
2800     /* If there is only one completion, see if it is acceptable. If it is
2801     not, free it up. In any case, short-circuit and return. This is a
2802     special case because names[0] is not the prefix of the list of names
2803     if there is only one completion; it is the completion itself. */
2804     if (names[1] == (char *)0)
2805
2806         10 -- Assuming the condition is false --
2807
2808         11 -- Taking false branch --
2809     {
2810         if (force_ignore)
2811             if ((*name_func) (names[0]) == 0)
2812                 {
2813                     free (names[0]);
2814                     names[0] = (char *)NULL;
2815                 }
2816
2817     return;
2818 }
```

Bug Type	Quantity	Display?
All Bugs	239	<input type="checkbox"/>
API		
Argument with 'nonnull' attribute passed null	12	<input type="checkbox"/>
Dead store		
Dead assignment	152	<input type="checkbox"/>
Dead increment	3	<input type="checkbox"/>
Dead initialization	1	<input type="checkbox"/>
Logic error		
Assigned value is garbage or undefined	3	<input type="checkbox"/>
Branch condition evaluates to a garbage value	4	<input type="checkbox"/>
Dereference of null pointer	47	<input type="checkbox"/>

СТАТИЧЕСКИЙ АНАЛИЗ КОДА ОССН С ИСПОЛЬЗОВАНИЕМ ИНСТРУМЕНТА SVACE (ИСП РАН)

Checker	File	Line	Description
MEMORY_LEAK.EX	ecc.c	102	Dynamic memory referenced by 'r_' was allocated at mpiutil.c:65 by calling function 'mpi_alloc' at ecc.c:102
MEMORY_LEAK.EX	ecc.c	103	Dynamic memory referenced by 's_' was allocated at mpiutil.c:65 by calling function 'mpi_alloc' at ecc.c:103
MEMORY_LEAK.EX	ecc.c	103	Dynamic memory referenced by 's_' was allocated at mpiutil.c:65 by calling function 'mpi_alloc' at ecc.c:103
MEMORY_LEAK.EX	ecc.c	108	Dynamic memory referenced by 't2' was allocated at mpiutil.c:65 by calling function 'mpi_alloc' at ecc.c:108
MEMORY_LEAK.EX	ecc.c	108	Dynamic memory referenced by 't2' was allocated at mpiutil.c:65 by calling function 'mpi_alloc' at ecc.c:108
MEMORY_LEAK.EX	file.c	3096	Dynamic memory referenced by 'pages' was allocated at slab.h:629 by calling function 'kccalloc' at file.c:3096
MEMORY_LEAK.EX	file.c	3096	Dynamic memory referenced by 'pages' was allocated at slab.h:629 by calling function 'kccalloc' at file.c:3096
MEMORY_LEAK.EX	parsec_unit_pdp...	37	Dynamic memory referenced by 'raw' was allocated at pdp_common.c:595 by calling function 'pdp_get_raw' at parsec_unit_pdp_text.c:37 and lost at parsec_unit_pdp_text.c:44.
MEMORY_LEAK.EX	parsec_unit...	37	Dynamic memory referenced by 'raw' was allocated at pdp_common.c:595 by calling function 'pdp_get_raw' at parsec_unit_pdp_text.c:37
MEMORY_LEAK.EX	parsec_unit...	30	Dynamic memory referenced by 'l' was allocated at pdp_common.c:79 by calling function 'pdp_get_new_init_mac' at parsec_unit_pdp_text.c:30
MEMORY_LEAK.EX	parsec_unit...	30	Dynamic memory referenced by 'l' was allocated at pdp_common.c:79 by calling function 'pdp_get_new_init_mac' at parsec_unit_pdp_text.c:30
MEMORY_LEAK.EX	dsi_sig_verif...	256	Dynamic memory referenced by 'data' was allocated at mpicode.c:110 by calling function 'mpi_read_from' at dsi_sig_verif.c:256
MEMORY_LEAK.EX	dsi_sig_verif...	256	Dynamic memory referenced by 'data' was allocated at mpicode.c:110 by calling function 'mpi_read_from' at dsi_sig_verif.c:256
MEMORY_LEAK.EX	net.c	275	Dynamic memory referenced by 'opt' was allocated at net.c:73 by calling function 'parsec_ip_options_get' at net.c:275
MEMORY_LEAK.EX	net.c	275	Dynamic memory referenced by 'opt' was allocated at net.c:73 by calling function 'parsec_ip_options_get' at net.c:275

Role	Description
1. Role: create	1. Call of 'pdp_get_raw' parsec_unit_pdp_text.c:[37:14] 2. Call of 'pdp_get_raw_v1' pdp_common.c:[595:10] 3. Call of 'kzalloc' pdp_common.c:[568:7] 4. Created if kmalloc(...)!=0 slab.h:[690:9] 5. Call of 'kzalloc' slab.h:[690:9] 6. Created if __kmalloc(...)!=0 slab.h:[561:9] 7. Call of 'kzalloc' slab.h:[561:9]
2. Role: leaked	1. leak parsec_unit_pdp_text.c:[44:3] 2. l!= 0 pdp_common.c:[563:2] 3. l!= 0 pdp_common.c:[591:2] 4. l!= 0 parsec_unit_pdp_text.c:[32:6] 5. raw != 0 parsec_unit_pdp_text.c:[37:6] 6. pdp_raw_size(...)== size

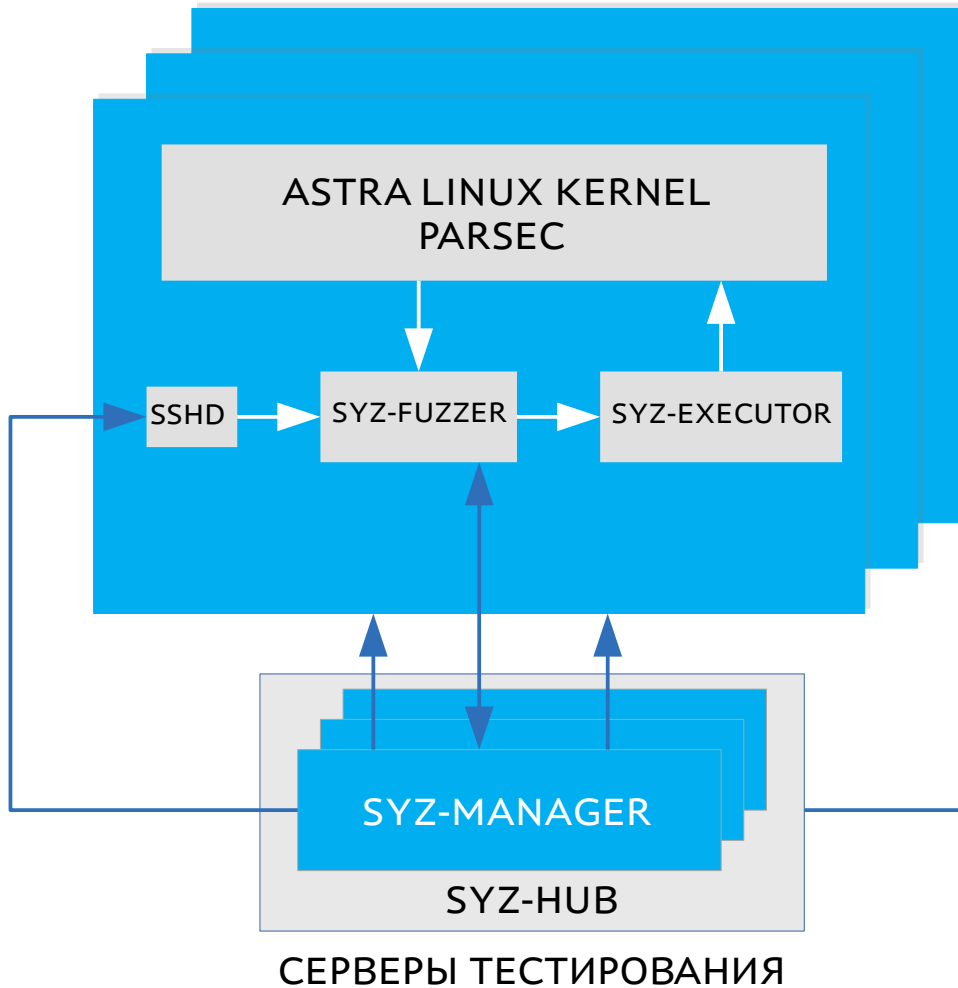

```
25 PDPL_T *l, *rawl;  
26 void *raw;  
27 size_t size;  
28  
29  
30 l = pdpl_get_new_init_mac(tls[i].lev, tls[i].ilev, 0, tls[i].cat,  
31 | | | | | tls[i].type, GFP_KERNEL);  
32 if (!l) {  
33 | D("pdp_get_new_init() error: %m");  
34 | return -1;  
35 }  
36
```

Undecided Unspecified Undecided MEMORY_LEAK.EX Dynamic memory referenced by 'raw' was allocated at pdp_common.c:595 by calling function 'pdp_get_raw' at parsec_unit_pdp_text.c:37 and lost at parsec_unit_pdp_text.c:44.

```
37 if (!(raw = pdpl_get_raw(l, &size, GFP_KERNEL))) {  
38 | D("pdp_get_raw() error");  
39 | return -1;  
40 }
```


ДИНАМИЧЕСКИЙ АНАЛИЗ КОДА ОССН С ИСПОЛЬЗОВАНИЕМ СТЕНДА SYZKALLER4ASTRA

ВИРТУАЛЬНЫЕ МАШИНЫ С ОССН



БАЗА ДАННЫХ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ

РАСЧЕТ ПОКРЫТИЯ КОДА

▼ parsec	60%	of 2096	
▼ linux-astra/parsec	61%	of 2071	
▼ dp		of 169	
pdp_common.c	82%	of 168	5454
pdp_common.h	100%	of 1	48
audit-file.c	37%	of 135	5454
audit-kernel.c	60%	of 22	
cap.c	80%	of 74	
cmdline.c	---	of 46	
crc16.c	---	of 5	
logfs.c	---	of 27	
net.c	48%	of 151	48
parsec-fs.c	85%	of 91	48
path.c	50%	of 6	
sec-audit.c	60%	of 220	5453
sec-audit.h	100%	of 1	
sec-hooks.c	51%	of 515	
security.c	72%	of 150	
security.h	100%	of 1	
syscalls.c	88%	of 217	
userconfine.c	43%	of 78	
xattr.c	73%	of 163	5454
parsec_elfrand.c	---	of 5	
parsec_gost89.c	---	of 20	
safesetid	---	of 78	

```
int pdp_lev_permission(PDP_LEV_T slev, PDP_LEV_T olev, int mode)
{
    if ((mode & LEGACY_IGNORE_LEV) return 0;
    if ((mode & R_OK) && (slev < olev) return -EACCES;
    if ((mode & W_OK) && (slev != olev) return -EACCES;
    if ((mode & X_OK) && (slev < olev) return -EACCES;
    return 0;
}

int pdp_cat_permission(PDP_CAT_T scat, PDP_CAT_T ocat, int mode)
{
    if( mode & LEGACY_IGNORE_CAT ) return 0;
    if( (mode&R_OK) && ( (scat & ocat) != ocat ) ) return -EACCES;
    if( (mode&W_OK) && (ocat != scat) ) return -EACCES;
    if( (mode&X_OK) && ( (scat & ocat) != ocat ) ) return -EACCES;
    return 0;
}

int pdpml_conf_permission(const PDPML_I *s, const PDPML_I *o, int mode)
{
    if( pdp_lev_permission(s->lev,o->lev,mode) || pdp_cat_permission(s->cat,o->cat,mode) ) {
        return -EACCES;
    }
    return 0;
}
```

CRASHES + CORPUS

РЕЗУЛЬТАТЫ ФАЗЗИНГ-ТЕСТИРОВАНИЯ ИНСТРУМЕНТАМИ CRUSHER (ИСП РАН), AFL И SYZKALLER4ASTRA МЕХАНИЗМОВ ЗАЩИТЫ ОССН

17

LCOV - code coverage report

Current view: [top level](#) - utils-pdp

Test: cov.info

Date: 2021-08-12 05:10:30

	Hit	Total	Coverage
Lines:	2409	3009	80.1 %
Functions:	127	163	77.9 %

Filename	Line Coverage	Functions
ls_pdp.c	 100.0 % 33 / 33	100.0 % 2 / 2
ls_proc.c	 67.4 % 698 / 1036	76.2 % 48 / 63
pdp-id.c	 90.1 % 64 / 71	100.0 % 2 / 2
pdp-ls.c	 86.3 % 1212 / 1404	74.7 % 56 / 75
pdpl-file.c	 82.9 % 189 / 228	100.0 % 7 / 7
pdpl-ps.c	 89.2 % 58 / 65	100.0 % 4 / 4
pdpl-user.c	 90.1 % 155 / 172	80.0 % 8 / 10

70% – покрытие модулей безопасности PARSEC по базовым блокам;

80% – покрытие модулей пользовательского пространства по строкам;

72 часа – среднее время устранения выявляемых ошибок в модулях безопасности.

АНАЛИЗ ПОМЕЧЕННЫХ ДАННЫХ ПРИ СБОРЕ ТРАСС ПРОГРАММ ИНСТРУМЕНТОМ БЛЕСНА (ИСП РАН)

Проект Инструменты

Стек вызовов

Размер стека вызовов: 9h

Вызов	Возврат	функция	Адрес	Модуль	Смещение
007613E9	?	неизвестная	00007FFF7DE...		
007640D0	?	неизвестная	00000000040...		
017D588C	034DA150	stub_403700	000000000040...		
017D588D	034DA150	pam_authenti...	00007FFF7FB...	libpam.so.0.84.2	35F0h
017D67AB	034D9FEC	func__3b20	00007FFF7FB...	libpam.so.0.84.2	3820h
01999008	0342A331	pam_sm_authen...	00007FFF7C6...	pam_unix.so	4050h
019FCA30	01BC71CE	stub__3630	00007FFF7C6...	pam_unix.so	3630h
019FCA31	01BC71CE	pam_get_auth...	00007FFF7FB...	libpam.so.0.84.2	5610h
019FCA33	01BC71CE	jmp__5040	00007FFF7FB...	libpam.so.0.84.2	5040h

Поиск утечек

Автоматический режим

Список задач

Ручной режим

Трек

Вердикт

✓ [01BC71C1 - 8E5F4268] v(0x42E090, 0xD)

Восстановить буфер Найти доступ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0	31	71	61	7A	78	73	77	32	21	40	71	77	65				1qazxsw2!@qwe

Восстановить буфер

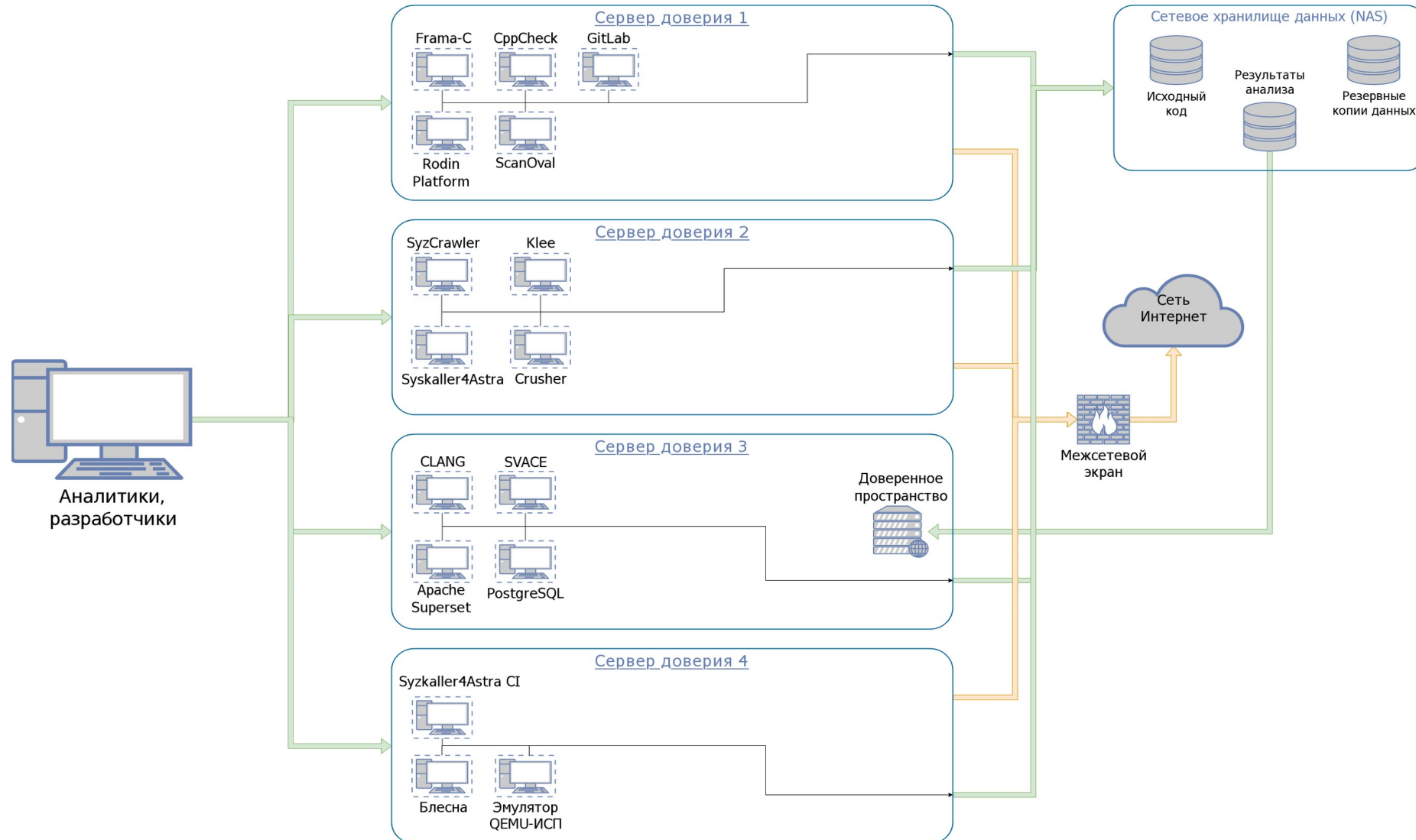
0%

Информ.

```
19FC5E0      func__7ffff7e48520
19FC69E      libpam.so.0.84.2      stub__3290
19FC69F      jmp__7ffff7e68020
19FC6C2      libpam.so.0.84.2      stub__3040
19FC6C3      func__jmp__7ffff7e4a9a0
19FC73B      pam_unix.so          stub__3610
19FC73C      func__jmp__7ffff7e4dda0
19FC741      stub__7ffff7de8130
19FC742      jmp__7ffff7e5e1e0
19FC789      stub__7ffff7de8308
19FC78A      func__jmp__7ffff7e4a350
19FC9CF      pam_unix.so          stub__3040
19FC9D0      func__jmp__7ffff7e4a9a0
19FCA31      pam_unix.so          stub__3630
19FCA32      libpam.so.0.84.2      pam_get_authtok
19FCA34      libpam.so.0.84.2      jmp__5040
19FCA4F      libpam.so.0.84.2      stub__30c0
19FCA50      libpam.so.0.84.2      jmp__30c6
19FCA63      func__7ffff7fe3a30
19FCA96      func__7ffff7fdf240
19FCB33      func__7ffff7fde750
19FCBDE      func__7ffff7fde5b0
19FCC0A      func__7ffff7feb10
19FCCBB      libpam.so.0.84.2      func__4f90
19FCCCB      libpam.so.0.84.2      stub__3110
19FCCCC      jmp__7ffff7e5e1e0
19FCCCE7      libpam.so.0.84.2      stub__3090
19FCCCE8      jmp__7ffff7e56ce0
19FCD11      libpam.so.0.84.2      stub__3090
19FCD12      jmp__7ffff7e56ce0
19FCD4A      libpam.so.0.84.2      stub__3100
```

Дерево вызовов

МАСШТАБИРУЕМАЯ СТРУКТУРА СТЕНДА ДОВЕРИЯ ДЛЯ ВЕРИФИКАЦИИ И АНАЛИЗА КОДА ОССН



The screenshot displays the ASTRA LINUX source code viewer interface. The main window shows the source code for the function `pdpml_get_new` in `pdp_common.c`. The code is as follows:

```
49 | PDPML_T* pdpml_get_new(gfp_t flags)
50 | {
51 |     PDPML_T *ml = kzalloc(sizeof(PDPML_T) + 1, flags);
52 |     if (!ml)
53 |         return ml;
54 |
55 |     *ml = PDPML_ZERO_INITIALIZER;
56 |     return ml;
57 | }
58 |
59 | PDPL_T* pdpl_get_new_init_mr(const PDPML_T *ml PDP_KRNL_ARG(gfp_t f))
60 | {
61 |     PDPL_T* l = _pdp_alloc(sizeof(PDPL_T), f);
62 |     if(!l) return NULL;
63 |
64 |     if(!ml) ml=pdpml_get_new(PDP_KRNL(f));
65 |
66 |     if(!ml) {
67 |         _pdp_free(l);
68 |     }
```

On the left, a file tree shows the current directory structure under `<<< dp`, with `pdp_common.c` selected. Below the code, a filter panel is visible with the following settings:

- Дата: Last quarter
- Количество результатов запроса: 1
- Дистрибутив: Astra Linux SE 1.7 (update 0)
- Название пакета: linux-astra-modules
- Средство: Syzkaller - фаззер уровня ядра ОС
- Критичность: 3. Средняя
- Версия пакета: 5.4.0-34.astra33+ci155

On the right, a list of error reports is shown:

- memory leak:51 (Syzkaller dynamic)
- memory leak:61 (Syzkaller dynamic)
- null-ptr-deref Write:129 (Syzkaller dynamic)
- core.NullDereference:592 (Clang)

Below the error reports is a "НАВИГАЦИЯ:" section with a horizontal bar chart.

Ошибка 93

(критичность: 4, аналоги к-во abs: 0)

Найдена в [linux-astra-modules-5.4.0/linux-astra/parsec/parsec_unit_pdpl_text.c:37](#) (уник. номер 104826714) [Вверх к описанию 93](#). Нет HTML для Svace

Dynamic memory referenced by 'raw' was allocated at pdpl_common.c:595 by calling function 'pdpl_get_raw' at parsec_unit_pdpl_text.c:37 and lost at parsec_unit_pdpl_text.c:44.

ЗАКЛЮЧЕНИЕ АНАЛИТИКА: В файле linux-astra/parsec/parsec_unit_pdpl.c:30 выполняется выделение памяти в pdpl_get_new_init_mac и выход из функции (return -1) в строке 39 в случае перехода в конструкцию if, при условии, что функция pdpl_get_raw возвращает NULL. В файле linux-astra/parsec/pdpl_common.c:599 вызывается функция _pdp_alloc, если память в ней не выделится, то функция pdpl_get_raw возвращает NULL.

Участок исследуемого кода:

```
17 struct PDP_TEST_LABEL_T tls[] = {
18     {0, 0, 0x00, 0x0, "0:0:0:0", 0, 14}, /*0 */
19     {0, 0xFFFF, 0x00, 0x0, "0:0:0:0", 0, 18}, /*1 */
20 };
22 const int tls_size = sizeof(tls) / sizeof(struct PDP_TEST_LABEL_T);
24 int __init test_pdpl_2from_raw(int _i)
25 {
26     PDPL_T *l, *rawl;
27     void *raw;
28     size_t size;
29     l = pdpl_get_new_init_mac(tls[_i].lev, tls[_i].ilev, 0, tls[_i].cat,
30                             tls[_i].type, GFP_KERNEL);
31     if (!l) {
32         D("pdpl_get_new_init() error: %m");
33         return -1;
34     }
35     if (!(raw = pdpl_get_raw(l, &size, GFP_KERNEL))) { // ----- Ошибка сработала здесь
36         D("pdpl_get_raw() error");
37         return -1;
38     }
39     if ((pdpl_raw_size(l) != size) || (size != tls[_i].raw_size)) {
```

РЕАЛИЗАЦИЯ НА ОСНОВЕ МРОСЛ ДП-МОДЕЛИ ВЕРИФИЦИРОВАННОГО МЕХАНИЗМА ЗАЩИТЫ ОСН АСТРА LINUX SPECIAL EDITION

22

Управление политикой безопасности - Режим эксперта

Файл Правка Настройки Помощь

астра

- Аудит
- Группы
- Замкнутая программная среда
- Мандатное разграничение доступа
- Мандатные атрибуты
- Мандатный контроль целостности
 - Режим эксперта
- Монитор безопасности
- Настройки безопасности
- Политики учетной записи
- Пользователи
- Привилегии
- Управление квотами
- Устройства и правила

Настройки мандатного контроля целостности

Управление Режим эксперта

Максимальный уровень целостности (текущий): 63 - Высокий

Максимальный уровень целостности (в загрузчике): 63 - Высокий

Целостность файловой системы (fs-ilev.conf) Сервисы

Включено частично. Запустите "set-fs-ilev status -v" чтобы увидеть подробности.

Отметить все элементы по умолчанию

Файловая система Редактирование конфига Исключения

Имя	Текущий уровень целостности	Уровень целостности в конфиге
/	63 - Высокий	
bin	63 - Высокий	
boot	63 - Высокий	
dev	63 - Высокий	
etc	63 - Высокий	
home	63 - Высокий	
lib	63 - Высокий	
lib32	63 - Высокий	
lib64	63 - Высокий	

Подсказка: используйте двойной щелчок

Свойства

dev

Общие Ярлык Дискреционные атрибуты Аудит Мандатная метка

Уровень конфиденциальности: Уровень_3

Уровень целостности: Высокий

Категории:

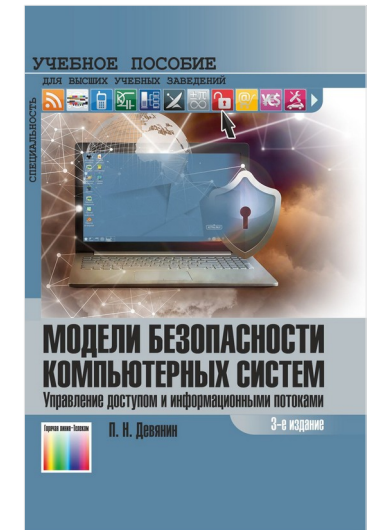
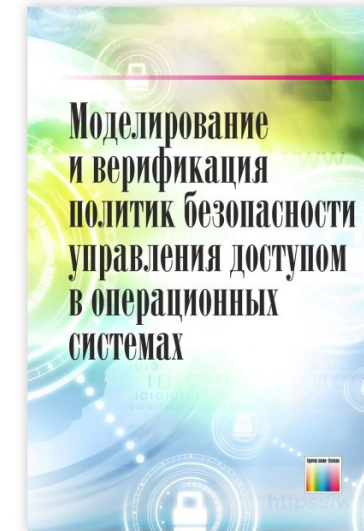
- 0:Категория_1
- 1:Категория_2
- 2:Неизвестная категория
- 3:Неизвестная категория
- 4:Неизвестная категория
- 5:Неизвестная категория

Спец. атрибуты:

- csnr

12:31 ВТ, 13 АПР

- > 1 | **БЕЗОПАСНОСТЬ ОПЕРАЦИОННОЙ СИСТЕМЫ СПЕЦИАЛЬНОГО НАЗНАЧЕНИЯ ASTRA LINUX SPECIAL EDITION.** Учебное пособие для вузов / П.В. Буренин, П.Н. Девянин, Е.В. Лебедеико и др.; Под редакцией доктора техн. наук, профессора П.Н. Девянина. 3-е издание, перераб. и доп. М.: Горячая линия – Телеком, 2019, 404 с.: ил.
- > 2 | Девянин П.Н., Ефремов Д.В., Кулямин В.В., Петренко А.К., Хорошилов А.В., Щепетков И.В. **МОДЕЛИРОВАНИЕ И ВЕРИФИКАЦИЯ ПОЛИТИК БЕЗОПАСНОСТИ УПРАВЛЕНИЯ ДОСТУПОМ В ОПЕРАЦИОННЫХ СИСТЕМАХ.** М.: Горячая линия – Телеком, 2019. 214 с.
- > 3 | Девянин П.Н. **МОДЕЛИ БЕЗОПАСНОСТИ КОМПЬЮТЕРНЫХ СИСТЕМ. УПРАВЛЕНИЕ ДОСТУПОМ И ИНФОРМАЦИОННЫМИ ПОТОКАМИ. УЧЕБНОЕ ПОСОБИЕ ДЛЯ ВУЗОВ.** 3-е изд., перераб. и доп. М.: Горячая линия – Телеком, 2020. 352 с.



Спасибо за внимание!