DOI: 10.15514/ISPRAS-2025-37(2)-5

Проектирование и развитие механизма мандатного контроля целостности в операционной системе Astra Linux

[1] *П.Н. Девянин, ORCID: 0000-0003-2561-794X <pdevyanin@astralinux.ru>*
[1] *А.А. Старостин, ORCID: 0009-0008-6465-080X <astarostin@astralinux.ru>*
[1] *Д.С. Панов, ORCID: 0009-0001-6430-7947 <dpanov@astralinux.ru>*
[2] *С.В. Усачев, ORCID: 0009-0001-2147-2811 <usachev_semen@mail.ru>*
[1] *ООО «РусБИТех-Астра», Россия, 117105, г. Москва, Варшавское ш., д. 26.*
[2] *ООО «Яндекс», Россия, 119021, г. Москва, ул. Льва Толстого, д. 16.*

Аннотация. Механизм мандатного контроля целостности (МКЦ) – фундамент безопасности сертифицированной по высшим классам защиты и уровням доверия операционной системы (ОС) Astra Linux, обеспечивающий наряду с другими механизмами, включая замкнутую программную среду (ЗПС), защиту привилегированных процессов ОС, целостность исполняемых и конфигурационных системных файлов и каталогов ОС, а также пользовательских данных. Использование МКЦ направлено на защиту от вирусов (например, «шифровальщиков»), от эксплуатации многих типовых уязвимостей программного обеспечения (ПО) ОС семейства Linux, в том числе приводящих к атакам нарушителя с правами суперпользователя root. Научной основой реализации МКЦ в ОС Astra Linux является соответствующая критериям ГОСТ Р 59453.1-2021 мандатная сущностно-ролевая ДП-модель управления доступом и информационными потоками в ОС семейства Linux (МРОСЛ ДП-модель). При этом внедрение механизма МКЦ поверх штатного для ОС семейства Linux дискреционного управления доступом представляет существенные трудности, часто требует разработки технологий и сценариев согласованного с ним применения системного и прикладного ПО. Авторами проводятся исследования по проектированию, развитию и эффективному использованию МКЦ. Во-первых, это доработки МРОСЛ ДП-модели для теоретического описания механизма МКЦ с учетом вносимых в него изменений. Во-вторых, адаптированная к МКЦ технология контейнерной виртуализации, когда потенциально «опасное» ПО запускается в изолированных на промежуточных уровнях целостности (в сессиях администратора системы, работающего на максимальном уровне целостности) или отрицательных уровнях целостности (в сессиях непривилегированного пользователя, работающего на нулевом уровне целостности) контейнерах-«песочницах» (например, docker). В-третьих, технологии и сценарии непосредственного запуска прикладного ПО на промежуточных или отрицательных уровнях целостности с настройкой меню рабочего стола администратора системы или непривилегированного пользователя, соответственно. В-четвертых, утилита настройки МКЦ, выставляющая файлам и каталогам уровни целостности или специальные флаги на основе правил профилей LSM-модуля AppArmor.

Ключевые слова: операционная система; мандатный контроль целостности; МРОСЛ ДП-модель; контейнер; Astra Linux.

Для цитирования: Девянин П.Н., Старостин А.А., Панов Д.С., Усачев С.В. Проектирование и развитие механизма мандатного контроля целостности в ОС Astra Linux. Труды ИСП РАН, том 37, вып. 2, 2025, стр. 61–78. DOI: 10.15514/ISPRAS–2025–37(2)–5.

Design and development of mandatory integrity control in Astra Linux OS

[1] *P.N. Devyanin, ORCID: 0000-0003-2561-794X <pdevyanin@astralinux.ru>*
[1] *A.A. Starostin, ORCID: 0009-0008-6465-080X <astarostin@astralinux.ru>*
[1] *D.S. Panov, ORCID: 0009-0001-6430-7947 <dpanov@astralinux.ru>*
[2] *S.V. Usachev, ORCID: 0009-0001-2147-2811 <usachev_semen@mail.ru>*
[1] *RusBITech-Astra,*
26, Varshavskoe, Moscow, 117105, Russia.
[2] *Yandex,*
16, Leo Tolstoy St., Moscow, 119021, Russia.

Abstract. Mandatory integrity control (MIC) is the security foundation of the Astra Linux operating system (OS) certified for the highest protection classes and trust levels, which, along with other mechanisms, including a closed software environment, ensures protection of privileged OS processes, integrity of executable and configuration system files and OS directories, as well as user data. The use of MIC is aimed for protecting against viruses (for example, ransomware), from the exploitation of many typical vulnerabilities in the software of the Linux family OS, including those leading to attacks by the adversaries with superuser's root rights. The scientific basis for the implementation of MIC in the Astra Linux OS is the mandatory entity-role model of access and information flows security control in OS of Linux family (MROSL DP-model) that meets the criteria of GOST R 59453.1-2021. At the same time, the implementation of the MIC over the standard discretionary access control for the OS of Linux family presents significant difficulties and often requires the development of technologies and scenarios for the coordinated use of system and application software. In this regard, the authors conduct research on the design, development and effective use of MIC, a number of the results of which are devoted to this article. Firstly, there are modifications of the MROSL DP-model for the theoretical description of the MIC, including new features making for it. Secondly, adapting for MIC the container virtualization technology, when potentially "dangerous" software (for example, browsers) is launched at isolated intermediate integrity levels (in sessions of the system administrator with maximum integrity level) or negative integrity levels (in sessions of an unprivileged user with zero integrity level) in containers-sandboxes (for example, docker). Thirdly, technologies and scenarios for directly launching application software at intermediate or negative integrity levels with the configuration of the desktop menu of the system administrator or unprivileged user, respectively. Fourthly, the MIC configuration utility, which setting integrity levels or special flags for files and directories based on the rules of the AppArmor LSM module profiles.

Keywords: operating system; mandatory integrity control; MROSL DP-model; containers; Astra Linux.

For citation: Devyanin P.N., Starostin A.A., Panov D.S., Usachev S.V. Design and development of mandatory integrity control in Astra Linux OS. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 2, 2025. pp. 61-78 (in Russian). DOI: 10.15514/ISPRAS-2025-37(2)-5.

1. Введение

Управление доступом одна из основных функций безопасности [1], реализуемых средствами защиты информации (СЗИ) такими, как операционные системы (ОС), средства виртуализации, системы управления базами данных (СУБД) и др. В силу относительной простоты в большинстве ОС, особенно семейства Linux, применяется выполняющий эту функцию механизм дискреционного управления доступом [2-3], основанного на непосредственном и независимом друг от друга назначении прав доступа субъектам (например, процессам) к сущностям (объектам доступа, например, файлам или каталогам). Были решения, например, еще в 1976 г. в ОС Multics, использования более сложного механизма мандатного управления доступом [3-4], когда субъектам назначаются уровни доступа, сущностям – уровни конфиденциальности, и с использованием сравнения этих уровней принимаются решения о предоставлении доступов с конечной целью предотвратить создание информационных потоков (скрытых каналов) с высокого уровня

конфиденциальности на низкий [5]. Особенно часто в СУБД в качестве «промежуточного» по сложности реализации и достаточного гибкого механизма используется ролевое управление доступом [3, 6], отличающееся от дискреционного в первую очередь (помимо иерархии ролей, ограничений и др.) назначением прав доступа субъектам к сущностям не непосредственно, а через роли.

Во многом прорывным стало применение длительное время невостребованного механизма мандатного контроля целостности (МКЦ) [3, 7], близкого по технологиям реализации к мандатному управлению доступом. Он основан на присвоении субъектам и сущностям уровней целостности и использовании сравнения этих уровней для принятия решения о предоставлении доступов с целью – предотвратить получение субъектом с меньшим уровнем целостности управления другим субъектом с большим уровнем целостности. Таким образом задача МКЦ – защита привилегированных высокоцелостных компонент (субъектов или сущностей) СЗИ от их несанкционированного изменения или захвата управления над ними со стороны непривилегированных низкоцелостных субъектов. Внедрение в 2007 г. в ОС Microsoft Windows Vista механизма МКЦ совместно с механизмом контроля учётных записей (User Account Control – UAC) [8] показало его высокую эффективность для обеспечения целостности программной среды этой ОС, предотвращения заражения вирусами и внедрения программных закладок.

Именно поэтому более 10 лет назад было решено сделать МКЦ одним из основных механизмом защиты ОС Astra Linux (операционной системы специального назначения Astra Linux Special Edition) [9-10], согласованным с штатным для всех ОС семейства Linux механизмом дискреционного управления доступом, а также механизмом мандатного управления доступом, который к тому времени уже был внедрен в эту ОС. Эти механизмы защиты совместно с рядом других, например, замкнутой программной средой (ЗПС), реализуются в подсистеме безопасности PARSEC ОС Astra Linux.

Научной основой реализации перечисленных механизмов управления доступом в ОС Astra Linux является мандатная сущностно-ролевая ДП-модель управления доступом и информационными потоками в ОС семейства Linux (МРОСЛ ДП-модель) [3]. Данная модель полностью соответствует критериям, изложенным в ГОСТ Р 59453.1-2021 [1], и верифицирована с применением инструментальных средств [11] согласно рекомендациям ГОСТ Р 59453.2-2021 [12]. При этом модель имеет иерархическое представление, позволяющее описывать ее по уровням (слоям), где каждый нижний уровень модели представляет абстрактную систему, элементы которой не зависят от новых элементов, принадлежащих более высокому уровню модели, который, в свою очередь, наследует, а при необходимости корректирует или дополняет элементы нижнего уровня. В целом иерархическое представление модели состоит из восьми уровней – четырех уровней для моделирования управления доступом непосредственно в ОС (ролевого управления доступом, мандатного контроля целостности, мандатного управления доступом с информационными потоками по памяти, мандатного управления доступом с информационными потоками по времени) и четырех уровней для решения аналогичной задачи в штатной для ОС СУБД PostgreSQL. Такое разделение МРОСЛ ДП-модели на уровни соответствует тому, как она реализуется непосредственно в программном коде подсистемы безопасности PARSEC.

В контексте настоящей статьи наибольший интерес представляет второй уровень иерархического представления МРОСЛ ДП-модели, соответствующий механизму МКЦ. Этот уровень, как и вся модель регулярно модифицируется [13], чтобы, с одной стороны, отвечать находящейся в развитии теории информационной безопасности, с другой стороны, соответствовать появлению новых технологий применения механизмов защиты ОС Astra Linux. Такая регулярная модификация МРОСЛ ДП-модели полностью отвечает рекомендациям проекта национального стандарта ГОСТ Р «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке» [14].

Поскольку механизм МКЦ является фундаментом обеспечения безопасности ОС Astra Linux, то его дальнейшему проектированию, выработке эффективных сценариев его применения прилагаются значительные усилия специалистов «Группы Астра» (ООО «РусБИТех-Астра»), в том числе в этом участвуют авторы. Многие из этих сценариев стали возможными за счет реализации (начиная с релиза ОС Astra Linux 2021 г.) в механизме МКЦ достаточно «выразительных» уровней целостности, являющихся объединением неиерархических категорий целостности (маски 32 бит) и линейных уровней – знаковых чисел от -128 до 127 [10].

К первому направлению проектирования МКЦ относится адаптированная к нему технология контейнерной виртуализации, которая позволяет создавать для недоверенного потенциально «опасного» программного обеспечения (ПО) такого, как браузеры, своеобразные «песочницы», где оно изолируется от остального ПО. Вопросы разработки таких «песочниц» для потенциально «опасного» ПО исследовались в работе [15], посвященной анализу безопасности контейнерной виртуализации в ОС семейства Linux. При этом предлагалось изолировать «опасное» ПО с использованием штатных механизмов ОС таких, как namespaces и cgroups, а также seccomp [16]. Кроме того, в работе [17] исследовались возможности применения для изоляции ПО, имеющегося в пакете безопасности SELinux (LSM-модуле) механизма мандатного управления доступом, предотвращая с его помощью несанкционированный доступ к системным компонентам ОС. Вместе с тем наличие МКЦ в ОС Astra Linux позволяет повысить эффективность этих технологий за счет запуска «песочниц» на промежуточном неиерархическом или на отрицательных линейных уровнях целостности. В этом случае недоверенное ПО, даже если подвергнется атаке нарушителя или заражению вирусом, не будет представлять опасности для всей остальной системы.

Поскольку не все ПО возможно или эффективно запускать с использованием контейнерной виртуализации, следующим направлением исследований стала разработка технологий непосредственного запуска прикладного ПО на промежуточных или отрицательных уровнях целостности. Похожие технологии применяются в пакете безопасности SELinux, что снижает риск атак на ОС семейства Linux через эксплуатацию уязвимости ПО [18]. Кроме того, как уже было отмечено, МКЦ используется в ОС семейства Windows, где уровни целостности обеспечивают изоляцию процессов, файлов и каталогов, гарантируя, что процессы с низким уровнем целостности не могут модифицировать данные с более высоким уровнем целостности [8, 19]. Эти технологии исследуются в ряде работ, касающихся интеграции МКЦ с другими механизмами такими, как ролевое управление доступом, что может обеспечить гибкую многоуровневую защиту ОС [20]. Однако важным аспектом выполненного авторами исследования является учет специфики реализации механизма МКЦ в ОС Astra Linux, с целью разработки сценариев его применения, которые позволят оптимизировать работу ПО, анализируя влияние его применения на безопасность ОС в сессиях привилегированных и непривилегированных пользователей.

Также при проектировании и развитии механизма МКЦ в ОС Astra Linux целесообразно использовать существующие апробированные практики, повышающие эффективность управления доступом в ОС семейства Linux. Одна из них базируется на применении LSM-модуля AppArmor, реализующего более развитые чем штатные для ОС рассматриваемого семейства технологии дискреционного управления доступом [21]. Данный LSM-модуль позволяет назначать права доступа и ограничивать доступ к ресурсам ОС конкретному ПО посредством соответствующих ему профилей AppArmor. Непосредственное применение AppArmor в ОС Astra Linux затруднено в связи с практической невозможностью верификации этого механизма и обеспечения к нему доверия. Однако сами профили AppArmor могут быть полезны для настройки механизмов защиты ОС Astra Linux и в первую очередь МКЦ.

В целом развитие МРОСЛ ДП-модели, включая соответствующего МКЦ второго уровня ее иерархического представления, проектирование этого механизма управления доступом для

дальнейшего внедрения его результатов в ОС Astra Linux ведется в «Группе Астра» в рамках реализации единой методологии разработки безопасного системного ПО [22].

В связи с изложенным статья организована следующим образом. В следующем разделе рассматриваются основные новые элементы соответствующего МКЦ второго уровня иерархического представления МРОСЛ ДП-модели. В разделе 3 анализируются результаты проектирования адаптированной к МКЦ технологии контейнерной виртуализации. В разделе 4 излагаются технологии и сценарии непосредственного запуска прикладного ПО на промежуточных или отрицательных уровнях целостности. Раздел 5 посвящен настройке МКЦ на основе правил профилей LSM-модуля AppArmor. Заключение завершает статью, в нем подводятся итоги выполненного к настоящему времени теоретического и практического проектирования механизма МКЦ ОС Astra Linux, а также рассматриваются дальнейшие направления этих исследований.

2. Новые элементы соответствующего МКЦ второго уровня иерархического представления МРОСЛ ДП-модели

В предыдущем разделе было отмечено, что МРОСЛ ДП-модель является основой для реализации механизма управления доступом в подсистеме безопасности PARSEC ОС Astra Linux. При этом в контексте проводимого авторами исследования по переработке и развитию МКЦ именно в соответствующий ему второй уровень иерархического представления МРОСЛ ДП-модели вносится наибольшее число изменений. Часть этих изменений уже были представлены в [13]. Среди них наибольшее значение для повышения эффективности применения МКЦ имеют специальные метки (флаги) сущностей (файлов и каталогов), позволяющие повысить гибкость настройки этого механизма.

Рассмотрим их подробнее, для чего используем некоторые обозначения, заданные на первом (ролевого управления доступом) и втором уровнях иерархического представления МРОСЛ ДП-модели:

- U – конечное непустое множество учётных записей пользователей;
- S – конечное непустое множество субъектов (процессов) учётных записей пользователей;
- $E = O \cup C$ – конечное непустое множество сущностей, где O – множество сущностей-объектов (файлов), C – множество сущностей-контейнеров (каталогов). При этом следует обратить внимание, что в формальных моделях управления доступом под «контейнером», как правило, понимается составная сущность (например, каталог в файловой системе), состоящая из других сущностей-объектов или сущностей-контейнеров, к которым по отдельности возможно осуществление управления доступом. В контексте системного ПО «контейнер», как правило, основанная на виртуализации технология (например, Docker или Podman) изоляции процессов ОС и доступных им ресурсов. Поэтому далее, во избежание путаницы, в случаях, когда речь идет о каталогах файловой системы, используется термин «сущность-контейнер»;
- R – множество ролей;
- NR – множество запрещающих ролей;
- AR – множество административных ролей;
- (LI, ≤) – решётка уровней целостности, при этом заданы минимальный i_low = ⊗LI и максимальный i_high = ⊕LI элементы решётки, соответственно;
- i_u: U → LI – функция, задающая для каждой учётной записи пользователя её уровень целостности – максимальный разрешенный уровень целостности субъектов, функционирующих от её имени;

- i_e: E → LI – функция, задающая уровень целостности для каждой сущности;
- i_r: R ∪ NR ∪ AR → LI – функция, задающая для каждой роли, запрещающей роли или административной роли её уровень целостности;
- i_s: S → LI – функция, задающая для каждого субъекта его текущий уровень целостности.

Тогда соответствующие специальным меткам (флагам) сущностей и ролей функции задаются следующим образом:

- IRELAX: C ∪ R ∪ NR ∪ AR → {true, false} – функция, задающая способ получения доступа на запись к сущностям-контейнерам, ролям, запрещающим ролям и административным ролям с учетом или без учета их уровня целостности. Если доступ на запись к сущности-контейнеру или роли $c \in C \cup R \cup NR \cup AR$ разрешён субъекту $s \in S$ только с текущим уровнем целостности большим или равным уровня целостности этой сущности-контейнера ($i_s(s) \geq i_e(c)$) или роли ($i_s(s) \geq i_r(c)$), то по определению выполняется равенство IRELAX(c) = false, в противном случае выполняется равенство IRELAX(c) = true (например, такое значение соответствующего флага у каталога «/tmp», доступ на запись к которому необходимо обеспечить процессам с любым текущим уровнем целостности). При этом, даже если IRELAX(c) = true, то субъекту s разрешено переименовывать или удалять непосредственно содержащуюся в сущности-контейнере или роли с сущность или роль $e \in E \cup R \cup NR \cup AR$ с уровнем целостности не выше, чем текущий уровень целостности этого субъекта (соответственно, $i_s(s) \geq i_e(e)$ или $i_s(s) \geq i_r(e)$);
- SSI: E ∪ R ∪ NR ∪ AR → {true, false} – функция, задающая порядок получения доступа на чтение или использования права доступа на выполнение к сущностям, ролям, запрещающим ролям и административным ролям с учетом или без учета их уровня целостности. Если доступ на чтение или использование права доступа на выполнение к сущности или роли $e \in E \cup R \cup NR \cup AR$ разрешён субъекту только с текущим уровнем целостности большим или равным уровня целостности этой сущности ($i_s(s) \geq i_e(e)$) или роли ($i_s(s) \geq i_r(e)$), то по определению выполняется равенство SSI(e) = true (например, такое значение соответствующего флага целесообразно задать некоторым файлам с высоким уровнем целостности из каталогов «/etc» и «/dev»), в противном случае выполняется равенство SSI(e) = false;
- SILEV: O → {true, false} – функция, задающая порядок активизации субъекта из сущности-объекта с учетом уровней целостности этого объекта и уровня целостности учетной записи пользователя, от имени которой активизируется субъект (это позволяет процессам в сессии учетной записи пользователя с низким уровнем целостности запускать некоторые процессы, которым для выполнения их функций необходим высокий текущий уровень целостности; например, с помощью соответствующего флага помечается исполняемый файл консольной утилиты passwd, запускаемой для изменения пароля низкоцелостной учетной записи пользователя, образ которого хранится в обладающем высоким уровнем целостности файле «/ets/shadow»);
- IINH: C ∪ R ∪ NR ∪ AR → {true, false} – функция, определяющая способ задания уровня целостности при создании в сущностях-контейнерах, ролях, запрещающих ролях и административных ролях с наследованием или без наследования их уровня целостности. Если при создании в сущности-контейнере или в роли $c \in C \cup R \cup NR \cup AR$ необходимо наследование их уровня целостности (с учетом возможной их пометки с помощью функции IRELAX), то по определению выполняется равенство IINH(c) =

true, в противном случае, когда на создаваемые сущности или роли устанавливается минимальный уровень целостности i_low, выполняется равенство IINH(c) = false.

При этом в модели говорится, что если значение функции IRELAX, SSI, SILEV или IINH от сущности или какой-либо роли равно true, то эта сущность или роль помечена с помощью соответствующей функции.

Также в МРОСЛ ДП-модели по сравнению с ее предыдущими редакциями добавляются три специальные административные роли: chmac_role, inherit_integrity_role и setmac_role. Первая – chmac_role соответствует привилегии PARSEC_CAP_CHMAC ОС Astra Linux и позволяет обладающему ею как текущей субъекту понижать уровни целостности сущностей, когда они не выше текущего уровня целостности этого субъекта. Вторая – inherit_integrity_role соответствует привилегии PARSEC_CAP_INHERIT_INTEGRITY и обеспечивает субъекту возможность создавать в сущности-контейнере (каталоге) новые сущности с наследованием уровня целостности от этой сущности-контейнера, как если бы она была помечена с помощью функции IINH. Третья – setmac_role соответствует привилегии PARSEC_CAP_SETMAC и разрешает субъекту активизировать нового субъекта с текущим уровнем целостности меньшим, чем у активизирующего субъекта (по умолчанию их текущие уровни целостности должны быть равными).

Таким образом, перечисленные новые функции и роли применяются в модели как для уточнения порядка администрирования МКЦ, так и для задания порядка наследования уровня целостности от родительской сущности-контейнера (каталога) при создании в нем новой сущности (файла или подкаталога).

Так при администрировании МКЦ изменение значений функций IINH, SSI или IRELAX от сущности (функций IINH и IRELAX только от сущности-контейнера) разрешено субъекту с текущим уровнем целостности не ниже уровня целостности этой сущности. Понижение уровня целостности сущности разрешено только субъекту, обладающему текущим уровнем целостности не ниже уровня целостности этой сущности и обладающему текущими специальными административными ролями root_role (специальной ролью, соответствующей полномочиям суперпользователя root в ОС семейства Linux) и chmac_role (как аналогом привилегии PARSEC_CAP_CHMAC). При этом для повышения или назначения несравнимого уровня целостности этой сущности или изменения ее параметров, задаваемых функцией SILEV, дополнительно требуется, чтобы субъект был доверенным (имел максимальный текущий уровень целостности i_high) и обладал текущей специальной административной ролью admin_cap_role.

Механизм наследования уровней целостности в МРОСЛ ДП-модели описывается следующим образом. При создании в сущности-контейнере по умолчанию новая сущность получает минимальный уровень целостности i_low. Исключениями являются случаи, когда сущность-контейнер помечена с помощью функции IRELAX или IINH, или создание в ней осуществляется субъектом, обладающим текущей специальной административной ролью inherit_integrity_role. Если сущность-контейнер помечена с помощью функции IRELAX, то уровень целостности новой сущности устанавливается равным наибольшей нижней границе значений уровней целостности сущности-контейнера и текущего уровня целостности осуществляющего создание субъекта. Если сущность-контейнер не помечена с помощью функции IRELAX, но помечена с помощью функции IINH, или создание в ней осуществляется субъектом, обладающим текущей специальной административной ролью inherit_integrity_role, то новая сущность наследует уровень целостности от сущности-контейнера. Кроме того, если сущность-контейнер, в которой осуществляется создание, помечена IINH, то создаваемая сущность-контейнер также помечается IINH.

Еще одно нововведение в МРОСЛ ДП-модели коснулось механизма перемещения сущности, которому в ОС семейства Linux соответствует, например, применение консольной утилиты mv. Этот механизм, по сути, близок к механизмам переименования сущности, а также создания или

удаления «жестких» ссылок (hard link) на сущности-объекты. Поскольку явно он не был описан в предыдущих редакциях модели, а при его применении необходимо учитывать требования МКЦ, было решено учесть его особенности в правиле переименования сущности вида rename_entity(x, y, old_name, name, z, z'). Это правило позволяют субъекту x переименовать или переместить сущность y, входящую в состав сущности-контейнера z, в сущность-контейнер z' (при переименовании z и z' совпадают), к которым субъект x должен иметь доступы на запись и обладать либо текущей специальной административной ролью root_role, либо текущими ролями или административными ролями, имеющими к ним право доступа на выполнение execute, и одновременно не обладать текущими запрещающими ролями, имеющими к z и z' это право доступа. Поскольку на сущность-объект может быть несколько «жёстких» ссылок в одной сущности-контейнере и, следовательно, несколько имён, то в правило добавлен параметр, указывающей старое имя сущности, подлежащее замене. Также в правиле этого вида проверяется наличие у сущности y уровня целостности, не превосходящего текущего уровня целостности субъекта x. Выполнение этого условия для случая, когда сущность-контейнер z' помечена с помощью функции IRELAX, проверяется непосредственно, для остальных случаев обеспечивается за счёт наличия у x доступа на запись к сущности-контейнеру z', в состав которого будет входить y.

В результате рассмотренные в разделе новые элементы второго уровня иерархического представления МРОСЛ ДП-модели позволили в ее рамках спроектировать и проанализировать использующие МКЦ технологии, которые рассматриваются в последующих разделах.

3. Адаптированная к МКЦ технология контейнерной виртуализации

Как уже было отмечено, МРОСЛ ДП-модель, являющаяся основой механизма МКЦ ОС Astra Linux, достаточно выразительная, чтобы учитывать многие существенные особенности функционирования системного ПО, в том числе применяемые в нем технологии контейнерной виртуализации. Также она позволяет использовать при моделировании произвольную решётку уровней целостности, что в свою очередь дает возможность разрабатывать и внедрять в ОС Astra Linux базирующиеся на МКЦ новые технологии защиты.

Одной из них стала контейнерная виртуализация (в первую очередь технология Docker), которая в сочетании с МКЦ позволяет изолировать недоверенное ПО, снизить риски влияния уязвимостей в таком ПО на безопасность остального системного и прикладного ПО.

Функционирующее на уровне целостности, соответствующем выделенной неиерархической категории, недоверенное ПО (например, браузер, который обрабатывает самые разные потенциально «опасные» данные из сети Интернет), даже если подвергнется атаке нарушителя или заражению вирусом, не будет представлять опасности для доверенных высокоцелостных компонент ОС. На рис. 1 приведена схема данного примера, где недоверенное ПО запускается в контейнере-«песочнице» на неиерархическом уровне целостности «Виртуализация» 0x00000002 (в соответствующей маске бит ненулевой только второй бит), а доверенное ПО – на максимальном уровне целостности «Высокий», используемом в ОС Astra Linux по умолчанию – 0x0000003F (то есть соответствующее ему десятичное значение – 63).

Важно отметить, что такая технология успешно защищает системные компоненты, имеющее уровень целостности «Высокий» (0x0000003F), которые часто используют процессы от имени привилегированных учетных записей пользователей, например, администраторов ОС. Однако файлы, с которыми обычно работают процессы от имени непривилегированных учетных записей пользователей имеют уровень целостности «Низкий» (0x00000000), а на промежуточном неиерархическом уровне целостности «Виртуализация» – 0x00000002, и они остаются подвержены атакам, в том числе с использованием вирусов-«шифровальщиков». Появление таких сценариев атак совпало с дальнейшим развитием МКЦ в ОС Astra Linux релиза 2021 г., которое привнесло в решетку уровней целостности линейный (иерархический) уровень целостности. Этот уровень целостности стал указываться после неиерархических

категорий, например, так: 0x00000002:-128. Его использование позволило усовершенствовать адаптированную к МКЦ технологию запуска контейнеров в ОС Astra Linux для противодействия «межконтейнерным» атакам и для защиты от эксплуатации уязвимостей ПО в контейнерах, функционирующих в сессиях непривилегированных пользователей.

Так «межконтейнерные» атаки происходят, когда нарушитель использует уязвимость в одном контейнере для доступа к ресурсам или привилегиям другого контейнера, функционирующего на той же хостовой ОС. Ранее при запуске нескольких контейнеров на одном уровне целостности «Виртуализация» (0x00000002) в ОС Astra Linux один из контейнеров мог потенциально атаковать другой контейнер для захвата управления над ним или получения несанкционированного доступа к его данным или ресурсам (рис. 2).

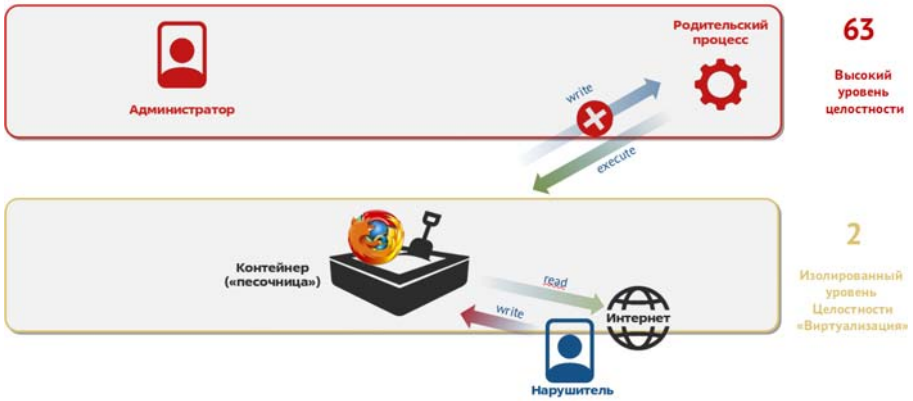

Рис. 1. Схема изоляция недоверенного ПО на выделенном неиерархическом уровне целостности.
Fig. 1. Scheme of isolating untrusted software at a dedicated non-hierarchical integrity level.

Рис. 2. Схема «межконтейнерной» атаки.
Fig. 2. Scheme of "inter-container" attack.

Применение линейных уровней целостности, например, -128 и -10, позволяет выполнить запуск этих контейнеров на разных уровнях, что защищает контейнер с большим уровнем целостности (на рис. 3 это 0x00000002:-10) от атаки из контейнера с меньшим уровнем целостности (на рис. 3 это 0x00000002:-128), так как при попытках получения соответствующего доступа на запись из второго контейнера это будет запрещено механизмом МКЦ. При этом изоляция второго контейнера от первого не осуществляется, то есть предполагается, что в первом контейнере функционирует менее «опасное» ПО, чем во втором.

Рис. 3. Схема защиты от «межконтейнерной» атаки.
Fig. 3. Protection scheme against "inter-container" attack.

Кроме того, с использованием реализованного в механизме МКЦ флага, соответствующего рассмотренной в предыдущем разделе функции IRELAX, возможна пометка общего каталога (например, «/home/<user>/Downloads»), смонтированного внутрь запущенных на разных уровнях целостности контейнерах. Это даст возможность таким контейнерам обмениваться данными как между собой, так и с другими процессами учетной записи пользователя, от имени которых они все функционируют.

Вместе с тем далеко не всегда нарушитель нацелен на захват управления над ОС в целом. Иногда ему достаточно зашифровать данные непривилегированного пользователя, чтобы достигнуть экономических целей своей атаки. Ранее при таком сценарии при запуске контейнера в сессии непривилегированного пользователя и реализации атаки вида «побег из контейнера» (на рис. 4 он имеет уровень целостности «Низкий» 0x00000000:0) нарушитель мог получить доступ ко всем данным (файлам или каталогам) соответствующего пользователя, так как они имеют тот же неиерархический уровень целостности «Низкий» (0x00000000:0).

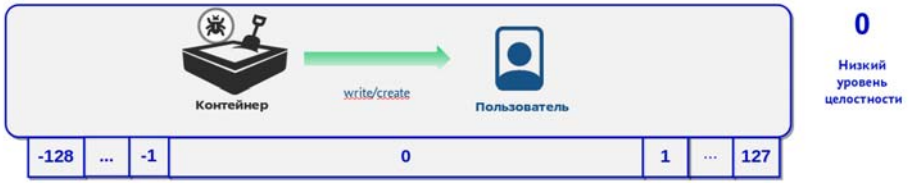

Рис. 4. Пример атаки из контейнера на уровне целостности «Низкий».
Fig. 4. Example of an attack from a container at «Low» integrity level.

Однако, если запускать контейнер, управление над которым может потенциально захватить нарушитель, на пониженном линейном уровне целостности (например, на 0x00000000:-128), то механизм МКЦ не позволит нарушителю получить доступ на запись к данным, находящимся на уровне целостности «Низкий» 0x00000000:0. Таким образом, существенно снижается риск успешного выполнения атак вирусов-«шифровальщиков» при захвате нарушителем контейнера, функционирующего от имени учетной записи непривилегированного пользователя.

Рассмотрим обратную ситуацию, когда надо защитить контейнер от других процессов, функционирующих от имени некоторой общей для них учетной записи пользователя. Ведь все чаще даже в сессиях непривилегированного пользователя в контейнерах запускается ПО, которое надо обеспечить дополнительной защитой (например, банковское ПО или ПО, устанавливаемое для доступа к государственным автоматизированным системам). Для этого с использованием механизма МКЦ предлагается запускать соответствующие «защищаемые» контейнеры на положительном линейном уровне целостности (на рис. 5 это 0x00000002:1).

Ранее положительный линейный уровень целостности не использовался, т.к. сам запуск процессов на таком уровне был невозможен. Прежде всего это было связано с тем, что согласно требованиям МКЦ [1], во-первых, уровень целостности любого подкаталога или файла, содержащегося в каталоге, должен быть не выше уровня целостности этого «родительского»

каталога (в том числе, корневого каталога «/», по умолчанию линейный уровень целостности которого был равен 0). Во-вторых, уровень целостности процесса должен быть не выше уровня целостности исполняемого файла, из которого он был запущен. Таким образом, линейный уровень целостности любого процесса не мог превосходить 0.

Исходя их этого, для запуска «защищаемых» контейнеров на положительном линейном уровне целостности потребуется назначение положительного линейного уровня целостности (например, 127) не только соответствующим образам этих контейнеров, но и всем находящимся выше их в иерархии файловой системы каталогов плоть до корневого каталога «/».

Подводя итог, рассмотренные технологии являются примером того, как важна при использовании контейнерной виртуализации основанная на применении механизма МКЦ многоуровневая защита. Следует отметить, что целью проведенного исследования было не обоснование преимущества контейнерной виртуализации для обеспечения безопасности функционирования различного ПО в ОС Astra Linux, а выработка технологии, позволяющей эффективно использовать для этого механизм МКЦ.

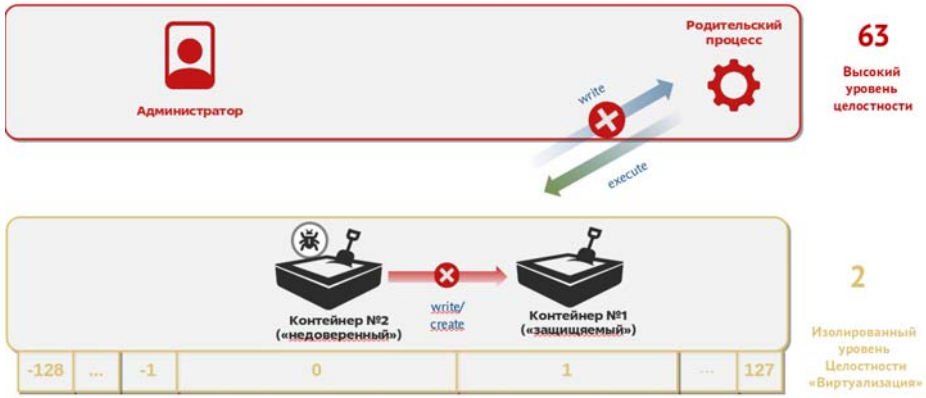

Рис. 5. Пример запуска «защищаемого» контейнера на положительном линейном уровне целостности.

Fig. 5. Example of running a "protected" container at a positive linear integrity level.

При этом можно отметить следующие достоинства данной технологии:

- Рассматриваемая технология базируется на использовании механизма МКЦ, который в отличие от многих других механизмов защиты (например, замкнутая программная среда или блокировка интерпретаторов) работает по умолчанию, начиная с основного (называемого «Усиленный» или «Воронеж») режима защиты ОС Astra Linux. При этом замкнутая программная среда или блокировка интерпретаторов ограничивают состав разрешенного для использования ПО, а МКЦ – нет;
- Контейнер позволяет упаковать в него прикладное ПО со всем необходимым окружением в максимально короткие сроки. При том, что одновременный запуск различного ПО вне контейнеров на различных уровнях целостности может привести к неразрешимым противоречиям при их доступе к общим ресурсам ОС (файлам, каталогам, сокетам и др.), так как может потребовать установки для них несовместимых параметров МКЦ;
- При использовании контейнерной виртуализации могут появляться дополнительные уязвимости, в том числе в составе компонент ядра ОС (хотя они встречаются достаточно редко). Однако для нарушителя проэксплуатировать эти уязвимости будет

гораздо сложнее, когда контейнер функционирует на отрицательном линейном уровне целостности. При этом с учетом изоляции контейнеров средствами ОС для выявления аномального «поведения» атакованных нарушителем контейнеров в перспективе возможно использование адаптированных к МКЦ методов машинного обучения.

4. Технологии и сценарии непосредственного запуска прикладного ПО на промежуточных или отрицательных уровнях целостности

На момент написания настоящей статьи, в ОС Astra Linux практически не используются промежуточные неиерархические уровни целостности, а линейный уровень целостности не применяется вовсе. Это негативно влияет на безопасность ОС, так как расширяет поверхность атаки, позволяя нарушителю эксплуатировать уязвимости конкретного прикладного ПО для распространения атаки на всю ОС. Поэтому рассмотренную в предыдущем разделе технологию запуска контейнеров на промежуточном неиерархическом или ненулевом линейном уровне целостности целесообразно адаптировать для запуска прикладного ПО непосредственно, то есть без применения контейнеров.

Основной сложностью при выработке технологий и сценариев запуска ПО на промежуточном неиерархическом или ненулевом линейном уровне целостности часто являлась некорректная работа ПО с механизмом МКЦ, что приводило к отказу от использования этих уровней целостности. В результате проведенного исследовании были выявлены типовые ошибки, возникающие при работе такого ПО, что позволило предложить способы их устранения.

Рассмотрим первый сценарий: запуск прикладного ПО на промежуточных неиерархических уровнях целостности. В рамках его исследования были проанализированы 14 популярных графических приложений (например, файловый менеджер fly-fm, медиаплеер vlc, браузеры chromium и firefox, графический редактор gimp и другие), которые запускались в ОС Astra Linux релиза 2024 г. (1.8.1) на уровне защиты «Воронеж». Тестирование этого ПО показало, что в целом оно корректно функционируют на промежуточных уровнях целостности без необходимости значительных доработок или настройки со стороны администратора ОС. Однако основной возникающей здесь трудностью стали ошибки доступа ПО к сессионной шине межпроцессного взаимодействия D-Bus. Подсистема безопасности PARSEC реализована таким образом, что ПО, запущенное на уровне целостности, отличном от уровня целостности, с которым был осуществлен вход пользователя, не имеет доступа к шине D-Bus. Это приводит, например, к невозможности записи в некоторые файлы, а также запуска другого ПО. Хотя иногда такое ПО, не имея доступа к шине D-Bus, частично обходит это ограничение, используя резервные механизмы. Например, вместо штатного требующего доступа к D-Bus меню «Сохранить как» в файловом менеджере fly-fm, такое ПО может задействовать аналогичные технологии из сред графических оболочек GNOME или KDE Plasma. Также ПО для своей дальнейшей успешной работы может потребовать первоначальной настройки или запуска на максимальном уровне целостности «Высокий» (по умолчанию 63), что усложняет автоматизацию его работы на других уровнях целостности, или ПО может потребовать предварительной пометки некоторых каталогов флагами МКЦ IINH или IRELAX.

Для обеспечения корректного взаимодействия, функционирующего на промежуточных неиерархических уровнях целостности ПО с шиной D-Bus, можно использовать одну из следующих технологий:

1. Запускать, если это возможно, процессы ПО с использованием утилиты dbus-launch, устанавливающей для ПО переменные среды, позволяющие ему подключиться к шине D-Bus.
2. Удалять переменную окружения DBUS_SESSION_BUS_ADDRESS для запускаемых процессов ПО, при отсутствии которой автоматически создается новая сессионная шина D-Bus, к которой ПО сможет подключиться.

3. Запускать несколько дополнительных сессионных шин – по одной для каждого из несравнимых неиерархических уровней целостности (а также одну для отрицательных уровней целостности). При этом нужную шину передавать запускаемому процессу ПО через переменную окружения DBUS_SESSION_BUS_ADDRESS.

Все три технологии допустимы для использования, однако в рамках настоящего исследования была применена именно третья, как показавшая совместимость с большинством тестируемого ПО. Тем не менее, ее нельзя считать полностью универсальной, и исследования по повышению ее эффективности будут продолжены.

Рассмотрим второй сценарий: запуск прикладного ПО на отрицательных линейных уровнях целостности. В отличие от неиерархического уровня целостности, применение отрицательных линейных уровней может быть полезно не столько в сессии «высокоцелостного» привилегированного администратора ОС, сколько в сессии «обычного» непривилегированного пользователя, процессы от имени учетной записи которого функционируют на нулевом неиерархическом уровне целостности. Аналогично контейнерам непосредственный запуск ПО на отрицательном линейном уровне целостности позволяет защитить данные непривилегированного пользователя от эксплуатации уязвимостей в ПО, например, заражения вирусами, в том числе вирусами-«шифровальщиками». Тем не менее такая изоляция процессов ПО создает сложности. Например, по умолчанию процесс, запущенный на отрицательном линейном уровне целостности, не имеет прав на запись к большинству файлов и каталогов, что может ограничить его функциональность.

В связи с этим в ходе исследования была разработана технология, позволяющая запускать прикладное ПО на отрицательных линейных уровнях целостности, обеспечивая при этом необходимые права доступа ПО к ресурсам ОС без угроз для безопасности остального ПО. Для этого, как и в случае с запуском ПО на промежуточных неиерархических уровнях целостности, используется пометка некоторых каталогов флагами МКЦ IINH или IRELAX, в том числе следующих подкаталогов домашних каталогов пользователей: «/home/<user>/.cache», «/home/<user>/.config», «/home/<user>/.config/session», «/home/<user>/.config/autostart», «/home/<user>/.local/share», «/home/<user>/.local/share/applications», «/home/<user>/Загрузки» и других подобных каталогов. Кроме того, потребовалось назначение минимального уровня целостности 0x00000000:-128 файлам «/dev/null», «/dev/zero» и ряду других файлов из каталога «/dev», доступ на запись к которым необходимо предоставлять практически всем процессам. И, как было отмечено ранее, было предложено запускать дополнительную сессионную D-Bus на отрицательном уровне целостности 0x00000000:-128.

В целом применение для обоих сценариев технологий настроек механизма МКЦ ОС Astra Linux позволило обеспечить функционирования выбранного для тестирования ПО в обеих сценариях его запуска: на промежуточном неиерархическом или отрицательном линейных уровнях целостности. Данные технологии также включают настройку рабочего стола администратора системы и непривилегированного пользователя для удобства автоматического запуска ими ПО на соответствующих уровнях целостности. При этом в дальнейшем еще потребуются исследования по улучшению этих технологий, в частности по интеграции сессионной шины D-Bus с механизмом МКЦ.

5. Настройка МКЦ на основе правил профилей LSM-модуля AppArmor

Как уже было отмечено, AppArmor – это LSM-модуль, реализующий дискреционное управление доступом, позволяющий назначать права доступа и ограничивать доступ к ресурсам ОС семейства Linux конкретному ПО посредством соответствующих ему профилей, загружаемых в этот модуль ядра ОС [21]. Каждый профиль AppArmor состоит из правил, большая часть из которых, как показали исследования, реализуемы механизмом МКЦ посредством его соответствующей настройки. Примерами реализуемых МКЦ правил профилей являются назначение права или, наоборот, запрет на чтение/запись/исполнение файлов

(правила вида «r», «w», «x» и другие). Имеются нереализуемые МКЦ правила, например, правила взаимодействия с сессионной шиной D-Bus (правила вида «dbus»), контроля ресурсов (правило вида «rlimit») и некоторые другие. Ряд из этих правил, которые нельзя реализовать механизмом МКЦ в общем случае (например, правила вида «mount», устанавливающие разрешения на монтирование каталогов друг на друга), можно выполнить для конкретных наиболее востребованных сценариев (например, механизм МКЦ запрещает монтирование каталога с меньшим уровнем целостности на каталог с большим или несравнимым уровнем целостности). Часть непосредственно нереализуемых механизмом МКЦ правил можно выполнить штатными средствами ОС. Например, с помощью ограничивающего для ПО доступные ему системные вызовы механизма seccomp можно реализовать правила на блокировку файлов (правила вида «k») или на разрешение отображения в памяти исполняемого файла (правила вида «m»).

Наиболее точно механизм МКЦ позволяет выразить правила профилей ПО, устанавливающих права доступа к файлам или каталогам в предположение, что это ПО будет функционировать на уровнях целостности меньших максимального уровня целостности «Высокий», то есть на промежуточных неиерархических, нулевом или отрицательных линейных уровнях целостности. Рассмотрим примеры, которые стали основой технологии применения профилей AppArmor для настройки механизма МКЦ:

- Правила, устанавливающие запрет на чтение к файлам или каталогам (вида «r»), могут быть реализованы назначением им максимального уровня целостности «Высокий» (например, 0x0000003F:0) и флага SSI;
- Правила, устанавливающие разрешения на создание, удаление, запись, дозапись в файл или каталог (вида «w» или «a»), могут быть реализованы установкой уровня целостности файла или каталога равным уровню целостности процессов того ПО, которому необходимо разрешить к ним доступ (это также позволяет запретить рассматриваемые доступы к файлам и каталогам процессам, имеющим меньшие уровни целостности);
- Правила, устанавливающие разрешения на исполнение файла (вида «x»), могут быть реализованы назначением ему уровня целостности не ниже уровня целостности процессов того ПО, которому необходимо разрешить доступ на исполнение к этому файлу;
- Правила, устанавливающие разрешения на изменение дискреционных прав доступа к файлам или каталогам (вида «chmod»), не могут быть непосредственно реализованы механизмом МКЦ, но этот механизм в целом позволяет запретить изменение этих прав к файлам или каталогам процессу, обладающему меньшим или несравнимым чем у них уровнем целостности.

В контексте рассматриваемой технологии также представляет интерес применение используемой для настройки AppArmor утилиты aa-logprof. Эта утилита в процессе работы тестируемого ПО анализирует сообщения, получаемые от находящегося в состоянии обучения (Complain) AppArmor, после чего предлагает правила для соответствующего ПО профиля, которые, с одной стороны, позволят ПО функционировать, а, с другой стороны, установят запреты и разрешения для безопасной работы этого ПО. Разработка аналога утилиты aa-logprof для настройки механизма МКЦ, позволит ускорить решение этой задачи и уже осуществляется авторами.

6. Заключение

В настоящей статье изложены результаты очередного этапа проектирования и развития ключевого для безопасности ОС Astra Linux механизма защиты МКЦ. При этом рассмотрены основные новые элементы соответствующего МКЦ второго уровня иерархического

представления МРОСЛ ДП-модели, на основе которой осуществляется разработка и обоснование безопасности всех механизмов управления доступом данной ОС. В первую очередь это функции для задания специальных флагов МКЦ, устанавливаемых на файлы или каталоги, а также роли, соответствующие используемым для управления механизмом МКЦ привилегиям ОС.

С практической точки зрения изложены технологии и сценарии применения адаптированных к МКЦ технологий контейнерной виртуализации или непосредственного запуска ПО на промежуточных неиерархических или отрицательных линейных уровнях целостности. Проанализированы возникающие здесь технические сложности (например, связанные с взаимодействием процессов ПО с сессионной шиной D-Bus), а также способы их преодоления. Также описана технология настройки механизма МКЦ на основе правил профилей LSM-модуля AppArmor. В целом эти технологии и сценарии позволяют повысить безопасность ОС, сократить потенциально доступную для нарушителя поверхность атаки, и обеспечить дополнительную защиту как процессов, функционирующих от имени учетных записей привилегированных администраторов ОС, так и «обычных» непривилегированных пользователей (например, защитить их данные от вирусов-«шифровальщиков»).

В дальнейшем планируется продолжить эти исследования, в том числе в направлении как внедрения разработанных технологий непосредственно в релиз ОС Astra Linux, так и развития механизма МКЦ для его применения в доменной инфраструктуре и сетевых файловых системах.

Список литературы / References

[1]. ГОСТ Р 59453.1-2021 «Защита информации. Формальная модель управления доступом. Часть 1. Общие положения». М.: Стандартинформ. 16 с. / GOST R 59453.1-2021 «Information protection. Formal access control model. Part 1. General principles», 2021 (in Russian).

[2]. Bishop M. Computer Security: Art and Science, 2nd edition. Pearson Education Inc., 2018, 1440 p.

[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.: ил. / P.N. Devyanin. Security models of computer systems. Control for access and information flows. Hotline-Telecom, 2020, 352 p. (in Russian).

[4]. Bell D.E., LaPadula L.J. Secure Computer Systems: Unified Exposition and Multics Interpretation. Bedford, Mass.: MITRE Corp., 1976. MTR-2997 Rev. 1.

[5]. ГОСТ Р 53113.1-2008 «Информационная технология. Защита информационных технологий и автоматизированных систем от угроз информационной безопасности, реализуемых с использованием скрытых каналов. Часть 1. Общие положения». М.: Стандартинформ. 12 с. / GOST R 53113.1-2008 «Information technology. Protection of information technologies and automated systems against security threats posed by use of covert channels. Part 1. General principles», 2008 (in Russian).

[6]. Sandhu R. Role-Based Access Control / Advanced in Computers. Academic Press, 1998. Vol. 46.

[7]. Biba K.J. Integrity Considerations for Secure Computer Systems. Bedford, Mass.: MITRE Corp., 1975. MTR-3153.

[8]. Conover M. Analysis of the Windows Vista security model / Technical Report, Symantec Corp., 2008, 18 p.

[9]. Операционная система специального назначения Astra Linux Special Edition. Доступно по ссылке: https://astragroup.ru/software-services/os/, 16.10.2024. / Astra Linux Special Edition operating system. Available at: https://astragroup.ru/software-services/os/, accessed 16.10.2024.

[10]. Девянин П.Н., Тележников В.Ю., Третьяков С.В. Основы безопасности операционной системы Astra Linux Special Edition. Управление доступом. Учебное пособие. М., Горячая линия – Телеком, 2022, 148 стр. / Devyanin P.N., Telezhnikov V.Y., Tret'yakov S.V. Astra Linux Special Edition security basics. Access control. Hotline-Telecom, 2022, 148 p. (in Russian).

[11]. Девянин П.Н., Леонова М.А. Приемы по доработке описания модели управления доступом ОССН Astra Linux Special Edition на формализованном языке метода Event-B для обеспечения ее автоматизированной верификации с применением инструментов Rodin и ProB // Прикладная дискретная математика. 2021. № 52. С. 83-96. / P. N. Devyanin, M. A. Leonova, "The techniques of formalization

of OS Astra Linux Special Edition access control model using Event-B formal method for verification using Rodin and ProB", Prikl. Diskr. Mat., 2021, no. 52, pp. 83–96 (In Russian).

[12]. ГОСТ Р 59453.2-2021 «Защита информации. Формальная модель управления доступом. Часть 2. Рекомендации по верификация формальной модели управления доступом». М.: Стандартинформ. 12 с./ GOST R 59453.2-2021 «Information protection. Formal access control model. Part 2. Recommendations on verification of formal access control model», 2021 (in Russian).

[13]. Девянин П.Н. Результаты переработки уровней ролевого управления доступом и мандатного контроля целостности формальной модели управления доступом ОС Astra Linux. Труды ИСП РАН, том 35, вып. 5, 2023, стр. 7-22 / Devyanin P.N. The results of reworking the levels of role-based access control and mandatory integrity control of the formal model of access control in Astra Linux. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 5, 2023, pp. 7-22 (in Russian).

[14]. Девянин П.Н. О разработке проекта национального стандарта ГОСТ Р «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке». Труды ИСП РАН, том 36, вып. 3, 2024, стр. 63-82 / Devyanin P.N. On the development of the draft standard GOST R "Information protection. Formal access control model. Trudy ISP RAN/Proc. ISP RAS, vol. 36, issue 3, 2024, pp. 63-82 (in Russian).

[15]. Wan Z., Lo D., Xia X., L. Cai. Practical and effective sandboxing for Linux containers / Empir Software Eng, 2019, Vol. 24, pp. 4034–4070.

[16]. N. Lopes, R. Martins, M.E. Correia, S. Serrano, F. Nunes. Container Hardening Through Automated Seccomp Profiling // In Proceedings of the 2020 6th International Workshop on Container Technologies and Container Clouds, 2020, pp. 31–36.

[17]. J.-A. Kabbe. Security analysis of Docker containers in a production environment // Norwegian University of Science and Technology, 2017, 91 p.

[18]. N. Li. Usable Mandatory Integrity Protection for Operating Systems // In proc of IEEE Symposium on Security and Privacy, 2007, pp.164-178.

[19]. Mandatory Integrity Control. Available at: https://learn.microsoft.com/en-us/windows/win32/secauthz/mandatory-integrity-control, accessed 16.10.2024.

[20]. H. Chen, N. Li, Z Mao. Analyzing and Comparing the Protection Quality of Security Enhanced Operating Systems // In proc of the Network and Distributed System Security Symposium, 2009, 16 p.

[21]. AppArmor Core Policy Reference. Available at: https://gitlab.com/apparmor/apparmor/-/wikis/AppArmor_Core_Policy_Reference, accessed 21.10.2024.

[22]. Девянин П.Н., Хорошилов А.В., Тележников В.Ю. Формирование методологии разработки безопасного системного программного обеспечения на примере операционных систем. Труды ИСП РАН, том 33, вып. 5, 2021, стр. 25-40 / Devyanin P.N., Telezhnikov V.Y., Khoroshilov V.V. Building a methodology for secure system software development on the example of operating systems. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 25-40 (in Russian).

Информация об авторах / Information about authors

Петр Николаевич ДЕВЯНИН – член-корреспондент Академии криптографии России, доктор технических наук, профессор, научный руководитель ООО "РусБИТех-Астра" («Группа Астра»). Область интересов: теория информационной безопасности, формальные модели безопасности компьютерных систем, разработка безопасного программного обеспечения, операционные системы семейства Linux.

Petr Nikolaevich DEVYANIN – Dr. Sci. (Tech.), corresponding member of Russian Academy of Cryptography, professor, scientific director in RusBITech-Astra (Astra Linux). Field of Interest: information security theory, formal security models of computer systems, secure software development, operating systems of Linux family.

Алексей Александрович СТАРОСТИН – старший инженер ООО "РусБИТех-Астра" («Группа Астра»). Область интересов: формальные модели безопасности компьютерных систем, искусственный интеллект, операционные системы семейства Linux.

Alexey Alexandrovich STAROSTIN – senior engineer in RusBITech-Astra (Astra Linux). Field of Interest: formal security models of computer systems, artificial intelligence, operating systems of Linux family.

Денис Сергеевич ПАНОВ – инженер ООО "РусБИТех-Астра" («Группа Астра»). Область интересов: формальные модели безопасности компьютерных систем, операционные системы семейства Linux, разработка безопасного программного обеспечения.

Denis Sergeevich PANOV – engineer in RusBITech-Astra (Astra Linux). Field of Interest: formal security models of computer systems, operating systems of Linux family, secure software development.

Семен Владимирович УСАЧЕВ – разработчик ООО «Яндекс». Область интересов: разработка безопасного программного обеспечения, операционные системы семейства Linux, бэкенд-разработка, теория компиляторов, формальные языки и грамматики.

Semen Vladimirovich USACHEV – developer in Yandex. Field of Interest: secure software development, operating systems of Linux family, backend development, compiler theory, formal languages and grammars.