DOI 10.36622/VSTU.2022.25.4.004
УДК 004.8

РАСШИРЕННАЯ МОДЕЛЬ ОТКРЫТЫХ СИСТЕМ (ЧАСТЬ 3)

К.А. Бугайский, Б.О. Дерябин, К.В. Табаков, Е.С. Храмченкова, С.О. Цепенда

В третьей части статьи разработана плоскость администрирования, которая заявлена в составе, но не описана в референсной модели открытых систем OSR/RM. Плоскость администрирования рассмотрена в парадигме систем управления, то есть как состоящая из объектов управления, субъектов управления и механизмов управления. Объекты управления представлены как наборы параметров – конфигурации, описывающие связи и функционирование аппаратной и программной частей вычислительной системы. При описании субъектов управления показана связь между пользователями системы, их учетными записями и соответствующими пространствами прав пользователя. Механизмы управления рассмотрены как с точки зрения внутренней структуры, так и с точки зрения выполняемых ими операций. Описание плоскости администрирования выполнено с использованием терминологического и сущностного аппарата базовой плоскости модели, разработанной в предыдущих статьях серии. В рамках разработки плоскости администрирования предложены операторы, отображающие взаимосвязи субъекта, объекта и механизма управления.

Ключевые слова: открытые системы, информационная безопасность, модель, информационная система

Введение

Расширенная модель открытых систем, как и базовая модель открытых систем OSE/RM, предполагает наличие в своем составе плоскости администрирования [1]. Согласно определениям, приведенных в [2, 3], при разработке плоскости администрирования термин «администрирование» можно трактовать как процесс управления, направленный на поддержание вычислительной системы (далее – ВС) в рабочем состоянии и связанный с управлением взаимодействием и состоянием ресурсов по всем уровням архитектуры открытой системы. Согласно [4, 5] управление взаимодействием и состоянием ресурсов опирается на взаимосвязанные функциональные и физические характеристики ресурсов, установленные в данных о конфигурации ресурсов, то есть совокупности значений параметров, определяющих работу ресурсов.

Таким образом, на основании определений данных в [3, 6, 7], плоскость администрирования расширенной модели открытых систем (далее – модель, РМОС) должна обеспечивать моделирование процессов формирования элементов конфигурации, управления изменением этих элементов, связей между ними и подготовку отчетности об этих действиях. Дадим следующие определения [4 – 7].

1. Под элементом конфигурации будем понимать совокупность программных сущностей (далее – ПС) базовой плоскости РМОС. Данная совокупность ПС образует взаимосвязанную группу, обеспечивает выделенную функциональность при работе в составе ВС и управление ее конфигурацией осуществляется независимого от других элементов конфигурации.
2. Под конфигурацией будем понимать набор поименованных параметров и их значений, описывающих функциональные и физические характеристики ПС из состава элемента конфигурации.

Для обеспечения общности подхода, разработку плоскости администрирования модели будем представлять как структуру механизмов управления объектом со стороны субъекта управления или отношение

субъект => механизм => объект.

Элемент конфигурации

Элемент конфигурации (далее – ЭК) будем рассматривать как абстракцию над элементами базовой плоскости РМОС [1]. При описании базовой плоскости модели было предложено [8] рассматривать ее как набор слоев, отображающих выполнение со стороны ПС одной из операций по обращению к физическому носителю ИЕ

$$OPS = \{op^{mem}, op^{proc}, op^{file}, op^{net}, op^{io}\},$$

где op^{proc} – операции с процессором и системной шиной;

op^{mem} – операции с энергозависимой памятью (ОЗУ);

op^{file} – операции с энергонезависимой память (диски);

op^{io} – операции с устройствами ввода-вывода;

op^{net} – операции с устройствами сетевого обмена данными.

Каждый слой базовой плоскости модели представляет собой граф G^{op}, $op \in OPS$, являющийся деревом и имеющим в качестве корня монитор обращений. Мониторы обращений слоев базовой плоскости M^{op} обеспечивают выполнение той или иной операции с ИЕ за счет входящих в их состав драйверов и диспетчеров.

В соответствии с [1] обозначим множество ПС как $AE = \{a_1, \dots, a_i\}$ и множество ИЕ как $IE = \{e_1, \dots, e_j\}$ из состава ВС, а подмножества $AE^{op} \subseteq AE$, $IE^{op} \subseteq IE$ как участвующие в выполнении операций

$$OPS = \{op^{mem}, op^{proc}, op^{file}, op^{net}, op^{io}\}.$$

Тогда взаимодействие ПС и ИЕ в процессе обработки может быть представлен как:

$$P = \cup_{OPS} P^{op},\ P^{op}[AE^{op} \overset{op}{\Rightarrow} M^{op} \overset{op}{\Rightarrow} IE^{op}],\ op \in OPS. \quad (1)$$

В выражении (1) оператор объединения обозначает выполнение операций P^{op} в определенном порядке или с учетом приоритетности операций для ПС. Квадратные скобки используются для обозначения оператора «содержит» или «состоит из». Двойная стрелка соответствует выполнению операции. На рис. 1 показано взаимодействие для одного типа операций OPS.

В рамках модели все последовательности операций P^{op} относятся ко всем уровням и слоям базовой плоскости и формируются посредством уровня *kernel* [8]. Из этого можно сделать следующее предположение относительно плоскости администрирования.

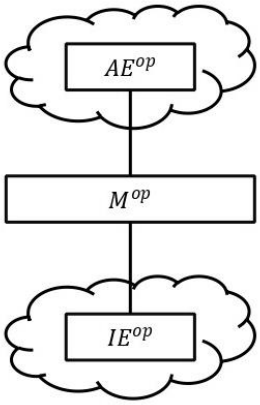

Рис. 1. Взаимодействие ПС и ИЕ для типа операций

П1. Монитор обращений и входящие в его состав драйверы и диспетчеры слоев HW и OW [1, 8] можно рассматривать как элемент конфигурации ядра ВС или ЭК типа *kernel*. Тогда, плоскость администрирования модели состоит, как минимум, из ЭК типа *kernel* для последовательностей операций

$$P^{mem}, P^{proc}, P^{file}, P^{net}, P^{io},$$

что можно обозначить как

$$CE^{mem}, CE^{proc}, CE^{file}, CE^{net}, CE^{io}$$

соответственно. Эти ЭК образуют множество

$$CE^{op} = \{CE^{mem}, CE^{proc}, CE^{file}, CE^{net}, CE^{io}\}.$$

Рассмотрим в качестве примера работу браузера, что предполагает связи с мониторами обращений базовой плоскости модели, обеспечивающие:

– сетевые операции (запросы и получение страниц от сервера);

– операции с памятью (парсинг страниц и изменение структуры DOM);

– операции с файлами (кеширование страниц);

– операции с процессами и потоками (распараллеливание процессов получения и обработки страниц);

– операции с устройствами ввода-вывода (дисплей и клавиатура как минимум).

Таким образом, браузер взаимодействует со всеми мониторами обращения и функционирование ПС, входящих в состав приложения «браузер», определяется предоставленными со стороны соответствующих ЭК типа *kernel* ресурсами и возможностью их использования. При этом состав браузера (число ПС и связи между ними) определяется разработчиком. Порядок и правила использования браузером ресурсов ВС также определяется разработчиком, что дает основания определить браузер как ЭК плоскости администрирования.

Аналогично, другие наборы ПС общесистемного или прикладного характера, относящиеся к слоям OW, MW, AW базовой плоскости модели являются ЭК плоскости администрирования. Определим такие ЭК как функциональные и обозначим их как множество CE^{app}.

Еще один тип ЭК вводится в модель на основании рассмотрения субъекта управления.

Субъект управления

Прежде всего следует подчеркнуть принципиальную особенность ВС: все пользователи, обрабатывающие информацию в ВС, являются физическими лицами, находятся вне ВС и на этом основании не могут быть включены в состав модели. Назовем таких пользователей *акторами*, которые представляют собой множество

$$Act = \{actor_1, \dots, actor_i\}.$$

В модели необходимо обеспечить отображение акторов на ЭК плоскости администрирования модели. Единственный способ обеспечить такое отображение в рамках ВС – это связать определенные ЭК с определенной учетной записью ВС или *аккаунтом* [3 – 5]. При этом разработчиков (*vendor*) не будем включать в число акторов и аккаунтов модели. При описании работы ВС обычно говорят об обработке данных актором – представленным учетной записью – с помощью программ из состава ВС. Таким образом можно говорить об отношении

$$актор => аккаунт => ЭК => ИЕ.$$

В модели принято, что в качестве учетной записи могут выступать не только акторы, но и отдельные ПС как процессы ВС. В качестве обоснования такого подхода можно привести следующие принципы построения современных ВС [4, 5, 9]:

– считаются лучшими практиками реализация общесистемных и прикладных сервисов по принципу «один сервис - одна учетная запись»;

– на АРМ работа прикладных программ идет в пространстве учетной записи пользователя userspace;

– современные технологии построения микросервисов [5] на основе виртуализации и контейнеризации построены на разделении пространств имен аккаунтов namespace.

Аккаунты ВС в модели образуют множество

$$Acc = \{account_1, \dots, account_j\}$$

и представлены следующими типами:

– *kernel* – встроенная уникальная учетная запись определяющая работу ЭК уровня ядра операционной системы или монитора обращений базовой плоскости модели;

– *system* – системные сервисы, как правило несколько встроенных аккаунтов ВС (подмножество Acc), обеспечивающих работу общесистемных сервисов;

– *service* – как правило несколько встроенных аккаунтов ВС (подмножество Acc) обеспечивающих работу прикладных сервисов;

– *root* – встроенная уникальная учетная запись определяющая выполнение функций субъектом – администратором ВС;

– *user* – как правило несколько аккаунтов ВС (подмножество Acc) обеспечивающих работу пользователей ВС.

Состав типов и количество аккаунтов каждого типа определяются архитектурой

конкретной ВС. Соотношение типов аккаунтов и слоев базовой плоскости модели можно представить в виде зон разграничения как показано на рис. 2.

В соответствии с этим соотношением в дальнейшем для обозначения комбинаций слоев базовой плоскости модели будем использовать типы аккаунтов как уровни плоскости администрирования.

Для выполнения задач обработки информации актор должен ассоциировать себя с одним из аккаунтов ВС, что в рамках модели можно обозначить как функцию *IniAcc(actor)*. Это позволяет в рамках модели говорить, что актор в ВС представлен или описывается аккаунтом, то есть, $Acc[\![Act]\!]$. Здесь двойные квадратные скобки обозначают оператор описания.

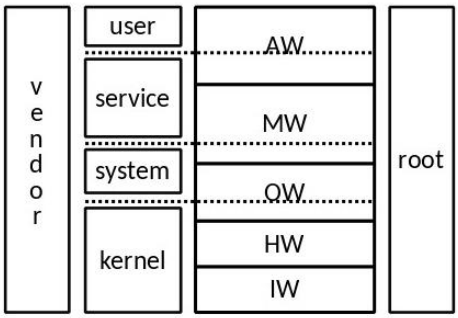

Рис. 2. Зоны разграничения типов аккаунтов

В рамках модели «описание» базируется на следующих предположениях.

П2. Разработчик задает в виде определенных алгоритмов и их параметров порядок и правила функционирования тех или иных сущностей модели (ПС или функций, например, *IniAcc(actor)*).

П3. В дальнейшем входные и выходные параметры реализованных алгоритмов в ПС или функций модели будем определять как параметры конфигурации $B = \{b_1, \dots, b_i\}$ той или иной сущности модели.

Таким образом, под «описанием» понимаем не только список правил, полностью определяющих возможности описываемой сущности, но и присвоение входящих в эти правила параметрам значений по умолчанию или диапазона значений.

Функция *IniAcc(actor)* в современных ВС реализуется встроенными механизмами, которые условно можно обозначить как:

– init – для аккаунтов типа *kernel*, *system*, и части *service*;

– login – для аккаунтов типа *root*, *user* и части *service.*

Результатом работы функции *IniAcc(actor)* является создание оболочки аккаунта – *account shell (*далее – *AS)*, то есть

$$as=IniAcc(actor)[init \vee login].$$

Здесь и далее квадратными скобками будем обозначать выражение «состоит из набора». Как будет показано в дальнейшем, набор параметров отличается от множества тем, что в набор могут входить параметры с одинаковыми именами, но разными значениями.

Оболочка аккаунта *as[GUI,CLI,utils]* состоит из наборов ПС относящихся как правило к слоям OW и MW базовой плоскости и образующих:

– *GUI* – графический интерфейс актора, присущего аккаунтам типа $root$, $user$ и частично для аккаунтов типа $service$;

– *CLI* – интерфейс командной строки актора присущего аккаунтам всех типов;

– *utils* – набор утилит обеспечивающих выполнение тех или иных функций операционной системы.

В рамках модели примем, что ПС, участвующие в формировании аккаунтов различных типов и составляющие *init*, *login*, *GUI*, *CLI*, *utils* $(init, login, GUI) \subset AE$, образуют ЭК слоя UW базовой плоскости модели $CE^{UW}[init, login, GUI, CLI, utils]$. Состав ЭК слоя UW базовой плоскости модели определяются разработчиком $vendor[\![CE^{UW}]\!]$. Структура ЭК слоя UW и ее соотношение с типами аккаунтов приведены на рис. 3.

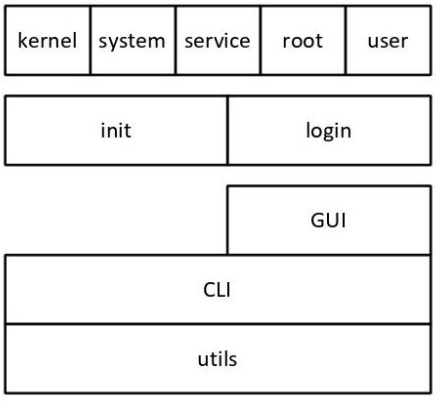

Рис. 3. Структура ЭК слоя UW модели

Отметим, что, как правило, все ПС из состава ЭК (подмножества AE^{UW}) и формирующие оболочку аккаунта, являются общими для всех аккаунтов, существующих в ВС независимо от их типа. Это позволяет в рамках модели говорить, что оболочка аккаунта описывает аккаунт $AS[\![Acc]\!]$.

В общем виде описание субъекта управления плоскости администрирования может быть представлено как $[\![AS[\![Acc[\![Act]\!]]\!]]\!]$.

Современные ВС допускают соотношение между *actor* и *account* как «многие ко многим». То есть, один актор может использовать несколько аккаунтов или один аккаунт может использоваться несколькими акторами. Представление субъекта управления в виде $[\![AS[\![Acc[\![Act]\!]]\!]]\!]$ предполагает, что разным сочетаниям «актор-аккаунт» могут ставиться в соответствие разные типы оболочки аккаунта. Обозначим типы оболочек, образующие множество *AS и* соответствующие типам аккаунтов как:

– *ask* – встроенная уникальная учетная запись типа *kernel*;

– *asy* – учетная запись типа *system*;

– *ass* – учетная запись типа *service*;

– *asr* – встроенная уникальная учетная запись типа *root*;

– *asu* – учетная запись типа *user*.

Поскольку в модели принято, что оболочка аккаунта является элементом конфигурации CE^{UW} состоящим из единого набора ПС, то различия между типами оболочек аккаунтов (а значит и аккаунтов) могут проявляться только в наборах параметров конфигурации $B = \{b_1, \dots, b_i\}$ AS. Можно говорить, что $B^{ask}[\![ask]\!]$, $B^{asy}[\![asy]\!]$, $B^{ass}[\![ass]\!]$, $B^{asr}[\![asr]\!]$, $B^{asu}[\![asu]\!]$. Тогда слой UW базовой плоскости модели может быть представлен как

$$UW[CE^{UW}, B^{ask}, B^{asy}, B^{ass,}B^{asr}, B^{asu}]. \quad (2)$$

Аккаунты пользователей ВС образуют множество $AS = \{ask, asy, ass, asr, asu\}$, соответственно можно говорить и о множестве параметров конфигураций аккаунтов $B^{AS} = \{B^{ask}, B^{asy}, B^{ass,}B^{asr}, B^{asu}\}$. Состав и допустимые значения параметров конфигурации оболочек аккаунтов – общие для всех типов аккаунтов – определяются разработчиком $vendor[\![B^{AS}]\!]$.

Значения параметров конфигурации оболочек различных аккаунтов одного типа могут различаться. Как правило, ВС содержат предопределенные шаблоны (templates) оболочек с параметрами по умолчанию для создания, в том числе автоматического, необходимых аккаунтов того или иного типа.

Основная цель формирования и функционирования аккаунта заключается в обеспечении обработки ИЕ с помощью определенных ПС в интересах конкретного актора. То есть, актору должны быть доступны определенные наборы ПС (образующие ЭК) и ИЕ ассоциированные с оболочкой используемого им аккаунта

$$CE^{as} = SelAcc(CE), IE^{as} = SelAcc'(IE).$$

Тогда аккаунт можно представить как

account[*IniAcc*(*actor*),*SelAcc*(*CE*),*SelAcc'*(*IE*)]
или $account[as, CE^{as}, IE^{as}]$.

Функции $SelAcc$ и $SelAcc'$ отображают ЭК (наборы ПС) и ИЕ из базовой плоскости модели на аккаунты плоскости администрирования. То есть, возможности актора в ВС в виде доступных ему ЭК и ИЕ (CE^{as}, IE^{as}) определяются типом используемого аккаунта.

С другой стороны, как правило, оболочкам аккаунтов типа *kernel* и *root* доступны для выполнения операций обработки информации полные множества ПС и ИЕ, а остальным (*system, service и user*) – только определенные подмножества этих множеств. Таким образом, можно говорить, что оболочки аккаунтов являются основой для формирования в интересах аккаунта контейнеров, содержащих различные комбинации ЭК ВС. На примере браузера отметим, что один и тот же ЭК типа CE^{app}, может использоваться в контейнерах различных учетных записей ВС с разными параметрами.

Важно отметить, что ЭК множества CE^{op} образуют отдельный контейнер ядра ВС, который обозначим как окружение аккаунта – *shell environment* (далее – *SE*), которое определяет порядок и правила исполнения

оболочки аккаунта и содержащихся в ней ПС и ИЕ с точки зрения допустимого использования ресурсов ВС

Обозначим типы окружения аккаунтов:

– *sek* – встроенная уникальная учетная запись типа *kernel*;

– *sey* – учетная запись типа *system*;

– *ses* – учетная запись типа *service*;

– *ser* – встроенная уникальная учетная запись типа *root*;

– *seu* – учетная запись типа *user*.

Тогда можно записать принципы формирования окружения аккаунтов как:

$$(sek \vee ser)[(CE^{op} \cap CE^{app}), IE] \quad (3)$$
$$(sey \vee ses \vee seu)[CE^{as}, IE^{as}] \quad (4)$$

Эти выражения позволяют считать, что окружение аккаунта полностью определяет возможности аккаунта по обработке информации посредством оболочки аккаунта или $SE[\![AS]\!]$ и $account[SE, as]$.

Таким образом, если субъектом управления является пользователь ВС, представленный как оболочка аккаунта, то в качестве объекта управления целесообразно рассматривать параметры окружения аккаунта.

Объект управления

В [1] было предложено параметр представлять как пару «ключ=значение» $b = (name, value)$. В предыдущих разделах было показано, что состав параметров, описывающих ЭК и окружения аккаунта *b(name)* определяются разработчиками соответствующих аппаратных и программных компонент ВС $vendor[\![B[\![CE]\!]]\!]$. Важно отметить, что состав *b(name)* постоянен и неизменен в течение всего жизненного цикла той или иной компоненты ВС. Другое важное замечание, что значение $value$ параметра определяется разработчиком, но может изменяться в течение жизненного цикла компоненты. Поэтому параметр в модели будем представлять как $b[name \coloneqq value]$, где символ $\coloneqq$ означает оператор присваивания. Заметим, что параметры, входящие в конфигурации разных аккаунтов, могут совпадать по имени, но иметь различные значения $b_i(name) = b_j(name)$, $b_j(value) \neq b_j(value)$. Поэтому в модели введен оператор «содержит». Применительно к параметрам оператор «содержит» будет означать список параметров не зависимо от значений, то есть использование именованной части *b(name)*. В то время как оператор «описывает» будет означать существование определенных для данной компоненты значений *b(value)*.

На основании предыдущих разделов соотношения между ПС, ЭК и окружением аккаунта представляющих из себя множества *AE*, *CE* и *SE* можно представить в виде $account[SE[CE[AE]]]$. В соответствии с [4, 5, 9, 10] с точки зрения описания параметров функционирования отдельные ПС (например, login, утилиты типа dir, ls) могут рассматриваться как отдельные ЭК типа CE^{app}, поэтому в дальнейшем будем говорить о параметрах применительно к ЭК.

В модели определены следующие типы параметров, являющиеся базовыми для определения функционирования ПС.

К1. Обозначим порядок и правила использования ресурсов аппаратной части ВС (слой $HW = \{CPB, RAM, DD, DIO, NIC\}$ [1] базовой плоскости модели) как параметры ресурсов B^{HW}. Объем предоставляемых ВС ресурсов, порядок и правила их использования определяются разработчиком, то есть можно говорить об описании параметров аппаратной части со стороны разработчика $vendor[\![B^{HW}]\!]$. Порядок и правила использования ресурсов аппаратной части ВС со стороны ПС слоев OW, MW, AW базовой плоскости модели определяются ЭК типа *kernel* [1, 4, 9]. В соответствии с предположениями П1, П3 можно утверждать, что эти правила описывают выполнение соответствующих операций $B^{HW}[\![P^{op}]\!]$. Следовательно, каждый ЭК типа *kernel* плоскости администрирования может быть представлен как набор параметров $B^{op}[b_1, \dots, b_k]$, соответствующего типу операции P^{op}, используемый всеми ПС из состава ВС. То есть, с точки зрения использования ресурсов, можно записать $CE^{op}[B^{op}]$. Это позволяет ввести в модель функцию $B^{op} = SetHW(B^{HW})$, которая обеспечивает формирование доступных значений ресурса аппаратной части для того или иного типа операций.

К2. Поскольку общесистемные и прикладные ЭК разрабатываются с учетом аппаратных характеристик уровня HW базовой плоскости, то можно положить, что $B^{ce} = SelOP(B^{op}),\ \ ce \in CE$, где $SelOP$– функция выбора параметров определяющих использование ресурсов со стороны конкретного ЭК $B^{op}[\![ce]\!]$, $vendor[\![B^{ce}]\!]$.

К3. Взаимодействие ЭК и ИЕ согласно [1, 8] обеспечивается механизмами коммуникаций (далее – МК). МК представляют из себя программно реализуемые в ЭК take- и grant- функции, которые задаются разработчиком и опираются в работе на использование параметров, определяющих каналы, структуры данных и соглашения (протоколов) обмена. МК представляет из себя множество $IC = \{ic_1, \dots, ic_i\}$ и сочетания различных МК могут использоваться при выполнении разных последовательностей операций со стороны ПС. Тогда $vendor[\![B^{ic}]\!]$, $B^{ic}[\![ic]\!]$, $ic \in IC$.

К4. Как показано в [8], кроме использования ресурсов ВС (реализации операций из состава множества OPS), ЭК процессе обработки информации непосредственно взаимодействуют между собой на основе реализованных программных интерфейсов, то есть реализуют операции типа op^{api}. Соответственно, можно говорить $vendor[\![B^{api}]\!]$, $B^{api}[\![ce]\!]$.

К5. С точки зрения защиты информации можно выделить в ЭК реализацию операций типа op^{aaa} [8], обеспечивающих функции идентификации, аутентификации и авторизации в пространстве прав пользователей ВС. Тогда соответствующие параметры конфигурации $vendor[\![B^{aaa}]\!]$, $B^{aaa}[\![ce]\!]$.

На основании пунктов К1 – К5 сделаем следующие предположения.

П4. В [4,5, 8, 9] показано, что:

– ИЕ можно соотнести с типами операций *OPS;*

– окружение пользователя (SE) должно рассматриваться как пространство прав пользователя;

– все ЭК, входящие в состав того или иного *SE*, наделяются правами доступа в рамках именного этого окружения аккаунта ко всем типам ИЕ, кроме тех, которые являются данными пользователя или созданы им (например, документы).

Таким образом, параметры операций op^{aaa} (предположение П3) необходимо разделить на два типа: описывающие работу ЭК с ИЕ документов пользователя, которые обозначим как B^{dfa} и все остальные, обозначение которых оставим как B^{aaa}.

Тогда пункт К5 можно переформулировать следующим образом: $vendor[\![B^{aaa}]\!]$, $vendor[\![B^{dfa}]\!]$, $(B^{aaa} \vee B^{dfa})[\![ce]\!]$.

П5. Параметры как минимум типов B^{HW} и B^{op}, а также B^{aaa} и B^{dfa} предполагают наличие иерархии в их структуре, соотнесенной с иерархией *account => SE => CE => a*.

Приведенный выше перечень параметров не является исчерпывающим и может меняться или дополняться в зависимости от архитектуры ВС.

Вместе с тем перечень типов параметров показывает, что как ЭК, так и окружение аккаунта в полной мере могут быть описаны только сочетанием параметров различных типов, то есть конфигурацией. В соответствии с [8] обозначим операции, не входящие в множество OPS, как множество $OPA = \{op^{aaa}, op^{api}\}$. В соответствии с П3, определим параметры для функций

$$SCE = CE^{as} = SelAcc(CE)$$

и

$$SIE = IE^{as} = SelAcc'(IE)$$

как

$$B^{CE},\ vendor[\![B^{CE}]\!],\ B^{CE}[\![CE]\!].$$

Конфигурация для ЭК может быть представлена как

$$C^{ce}[\bigcup_{OPS} B^{op} + \bigcup_{IC} B^{IC} + \bigcup_{OPA} B^{api} + \bigcup_{OPA} B^{aaa} + \bigcup_{OPA} B^{dfa}][\![ce]\!],\ ce \in CE \quad (5)$$

Разница в использовании символов объединения и сложения в выражениях для конфигураций заключается в следующем:

– объединение параметров предполагает их упорядочивание, то есть функционирование ПС в аккаунте будет зависеть от порядка следования параметров;

– сложение обозначает аддитивность при объединении различных типов параметров в общую конфигурацию.

Конфигурация для окружения аккаунта с учетом пунктов П4, П5 и выражений (3, 4) может быть представлена как

$$C^{se}[\cup_{OPS} B^{op} + \cup_{ACE} B^{aaa} + \cup_{SIE} B^{dfa} + B^{CE} + B^{AS} + \cup_{ACE} C^{ce}][\![se]\!],\ se \in SE \qquad (6)$$

Отметим, в частности, что наличие в выражении для конфигурации окружения аккаунта параметров B^{op} и B^{aaa} позволяет описывать влияние встраиваемых в состав ВС программных средств защиты информации, которые обеспечивают дополнительное управление аппаратными и программными ресурсами и их взаимодействием.

Окружение учетных записей типа *kernel* и *root* в рамках модели содержат параметры [4, 5, 10]:

– аппаратной части ВС $(sek \vee ser)[\cup B^{HW} \wedge\cup B^{op}]$,

– описывающие все ИЕ $(sek \vee ser)[\cup B^{aaa}[\![IE]\!] \wedge\cup B^{dfa}][\![IE]\!]$,

– описывающие СЕ, в части касающейся $(sek \vee ser)[\cup B^{*}[\![CE^{op}]\!]]$ и $(ser)[\cup B^{*}[\![CE^{*}]\!]]$.

Механизмы управления

В соответствии с [3, 5, 6] в рамках модели будем рассматривать роль и место механизмов управления плоскости администрирования (далее – МУ) не затрагивая вопросы устройства и функционирования собственно МУ. В частности потому, что этим вопросам посвящено значительное количество источников (см., например, [1, 2, 4, 5, 8 – 11] и литературу там же).

Обобщая, можно сказать, что приведенные в предыдущих разделах функции модели так или иначе описывают МУ, в том числе выполняемые МУ операции с параметрами конфигураций. При этом важно учесть, что собственно список параметров и их допустимые значения задаются разработчиками аппаратной и программной части ВС – $vendor[\![B^{*}]\!]$. Это означает, что каждый разработчик предоставляет в том или ином виде допустимые значения параметров, которые обозначим как *tmp*. На основании описания объекта управления целесообразно каждому типу параметров конфигурации поставить в соответствие отдельный МУ. Кроме того, выражения (5, 6) показывают, что МУ для отдельных типов параметров могут дублироваться на уровне окружения аккаунта и ЭК. Например, ЭК типа *kernel* формируют допустимые параметры ресурсов ВС для каждого из аккаунтов, в пределах окружения которого может происходить перераспределение выделенных ресурсов среди ЭК типа CE^{app}, то есть общесистемного и прикладного программного обеспечения используемого данным аккаунтом.

Аналогично можно сказать про МУ для параметров типа B^{aaa} и B^{dfa}, если принять во внимание, что один и тот же ЭК может использоваться разными аккаунтами для выполнения различающихся операций по обработке совпадающих (для разных аккаунтов) или нет ИЕ. Таким образом, МУ соответствует типу ЭК или окружения аккаунта.

Рассмотрим случай, когда пользователь запускает на исполнение новую программу. Поскольку до этого момента исполняемый файл программы выступал в качестве ИЕ, то запуск его на исполнение, то есть создание нового процесса ВС, равносильно созданию нового ЭК в окружении данного аккаунта.

Собственные действия МУ на этом примере можно определить следующим образом:

Д1 – клонирование параметров конфигурации,

Д2 – установка необходимых значений параметров из числа или диапазона допустимых,

Д3 – контроль текущих значений на допустимость,

Д4 – обработка ошибок в случае выхода значений параметра за допустимые пределы или значения.

Отдельно следует рассмотреть шаг Д2. С одной стороны, все параметры можно разделить на два типа – когда при установке выбирается одно из предопределенных значений параметра (например, тип протокола обмена B^{ic}) или когда устанавливается допустимый диапазон

значений (например объем доступной памяти B^{mem}). Второй тип, в частности, предполагает, что

$$B^{mem}[\![se]\!] = \sum_{ACE} B^{mem}[\![ce]\!].$$

Для шага Д1 характерным является пример создание нового пользователя со стандартными правами. Как следует из (2), это действие предполагает полное клонирование параметров типа B^{asu} и B^{CE} для создание новых оболочки и окружения аккаунта при этом ЭК CE^{UW} остается неизменным.

Шаги Д2 – Д4 могут иметь различную алгоритмическую реализацию за счет статического (тип протокола) или динамического (используемая память) характера значений параметра конфигурации.

Обозначим МУ как *D*. Если рассматривать МУ как функцию, то его можно представить в следующем виде

$$D: domD \xrightarrow{operation} ranD,$$

где в качестве входа МУ может быть шаблон коэффициента конфигурации $T([\![CE]\!] \vee [\![SE]\!])$, в качестве выхода – параметры конфигурации $B([\![CE]\!] \vee [\![SE]\!])$. Также возможны сочетания $B[\![SE]\!] \to B[\![CE]\!]$ и $B[\![CE_i]\!] \to B[\![CE_j]\!]$. Пусть $X = \{T([\![CE]\!] \vee [\![SE]\!]), B([\![CE]\!] \vee [\![SE]\!])\}$, $Y = \{B([\![CE]\!] \vee [\![SE]\!])\}$. В модели для МУ принят следующий перечень операций (operation):

Clone – клонирование параметра $b[name := value]\ [\![X_i]\!] \xrightarrow{clone} b[name := value]\ [\![Y_i]\!]$.

Mode – установка диапазона значений параметра $b[name := value]\ [\![X_i]\!] \xrightarrow{mode} b[name := (\alpha value + \beta)]\ [\![Y_i]\!]$, $0 < \alpha < 1, 0 \le \beta < value$.

List – выбор значения параметра $b[name := \{v_1, \dots, v_i, \dots, v_n\}]\ [\![X_i]\!] \xrightarrow{list} b[name := v_i]\ [\![Y_i]\!]$.

Range – упорядочивание значений параметра $b[name := \{v_1, \dots, v_i, \dots, v_n\}]\ [\![X_i]\!] \xrightarrow{list} b[name := (\{v_1, \dots, v_i, \dots, v_n\}, \prec]\ [\![Y_i]\!]$.

Select – выбор параметра $\{b_1, \dots, b_i, \dots, b_n\}[\![X_i]\!] \xrightarrow{select} b_i[\![Y_i]\!]$.

Таким образом МУ представляет собой

$$D[X, operation][\![Y]\!].$$

Указанные шаги работы МУ позволяют выделить в нем установочную – *set* и исполнительную – *mon* компоненты. Кроме того, шаг Д4 предполагает наличие в МУ регистратора, обеспечивающего журналирование событий и ошибок – *log*. При этом журналирование может вестись локально или централизовано, что предполагает наличие соответствующих ИЕ – *loge* и дополнительной ПС – *loga* в случае централизации логов.

Шаг Д2 также предполагает наличие дополнительной ПС – *ed*, выполняющей функции редактирования. Структура МУ приведена на рис. 4.

Если обозначить компоненты МУ как

$$node = \{set, mon, tmp, loga, loge, ed\},$$

то МУ $D(node)[X, operation][\![Y]\!]$.

Как следует из описания объекта и субъекта управления, хранилище шаблонов, исполнительная и установочная части одного МУ могут быть разнесены по разным типам аккаунтов. Что позволяет ввести понятия владельца и пользователя компонент МУ и, соответственно конфигураций. В качестве примера приведем МУ для ЭК типа CE^{app}

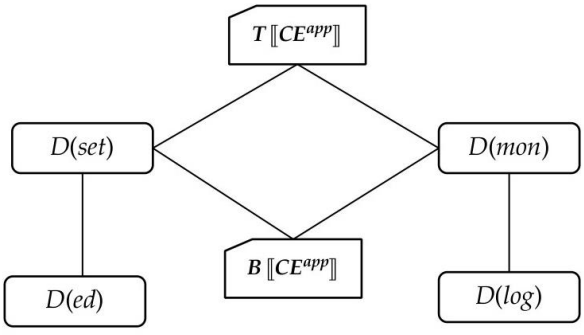

Рис. 4. Структура МУ

В операционных системах типа Linux шаблоны параметров хранятся в *usr/share*, текущие параметры в */etc* на системном уровне и */home* на пользовательском уровне,

и в качестве редактора может выступать любой текстовый редактор, а в операционных системах типа Windows практически все конфигурации и их параметры (включая шаблоны) хранятся в специализированной базе данных для работы с которой предоставляется специальная ПС (regedit). Если учесть, что компоненты МУ представляют собой ПС, то возможно проведение исследований его работы в методах и определениях базовой плоскости модели [1, 8]. Пример такого подхода приведен на рис. 5.

Следует подчеркнуть, что рис. 5 хорошо иллюстрирует тот факт, что любая конфигурация является по сути ИЕ расположенном на физическом носителе (будь то диск или сегмент оперативной памяти). Следовательно, управление конфигурациями использует все механизмы, включая МК, описанные в базовой плоскости модели [8], что позволяет в дальнейшем описывать процессы обработки информации и управления обработкой на одном языке.

Отметим, что наличие возможности журналирования в МУ служит обеспечению обратной связи между объектом и субъектом управления.

Следует подчеркнуть, что с учетом выражений (5), (6) исполнительная часть МУ должна быть в той или иной мере реализована в виде алгоритма во всех ЭК ВС. Выражения (3), (4) и положения [4, 5, 9, 10] позволяют сделать следующие предположения относительно расположения МУ на плоскости администрирования

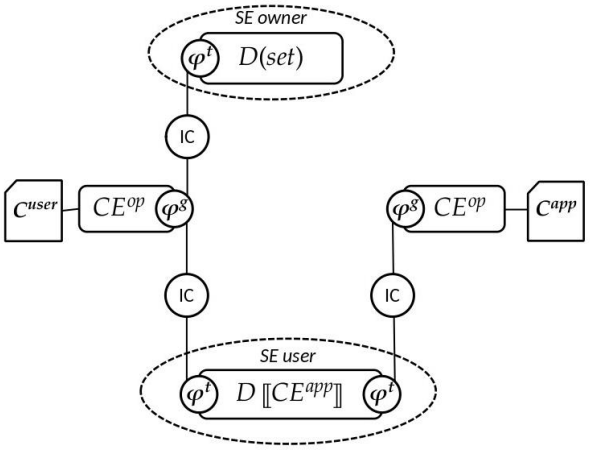

Рис. 5. МУ на базовой плоскости

П6. Оболочка аккаунта должна содержать МУ, прежде всего в части управления параметрами типа

$$B^{aaa}\ as[GUI,CLI,utils,D[\![B^{aaa}]\!]].$$

П7. Современные ВС как правило используют централизованные схемы управления параметрами типа B^{aaa} и B^{dfa}, что предполагает наличие в составе ВС ЭК типа CE^{aaa} и CE^{dfa}.

П8. Целесообразно любой МУ модели рассматривать как распределенную структуру, работающую на разных уровнях как с точки зрения слоев базовой плоскости, так и с точки зрения необходимых и достаточных прав пользователей ВС.

Заключение

Разработка в настоящей статье плоскости администрирования закрывает пробел, допущенный разработчиками референсной модели открытых систем OSE/RM, разработанной в 1996 году. В последующие годы усилия разработчиков по стандартизации открытых систем были направлены на развитие стандартов POSIX. Однако референсная модель сохраняет до настоящего времени потенциал с точки зрения моделирования в интересах защиты информации процессов в вычислительных системах на уровне отдельных программных и информационных сущностей таких систем. Данный подход приобретает особую актуальность в свете широкого применения при построении информационных систем принципов виртуализации и микросервисной архитектуры [9, 11]. Разрабатываемая расширенная модель открытых систем позволяет рассматривать в качестве программных и информационных сущностей вычислительных систем как отдельные программы, так и архитектурные единицы типа микросервисов или контейнеров. Разработанная плоскость администрирования предоставляет определенный уровень абстракции и язык описания для рассмотрения вопросов функционирования информационных систем с точки зрения теории управления в терминах субъект управления – пользователь системы, объект

управления – наборы разнородных параметров в виде конфигураций, описывающие работу и взаимодействие компонент программной и аппаратной части системы. Разработанный подход к описанию механизмов управления конфигурациями позволяет с хорошим уровнем абстракции описывать процессы обеспечения жизненного цикла конфигураций, состоящих из разнородных по смыслу и структуре параметров. А также учитывать распределенный по различным узлам вычислительной системы характер применения конфигураций и механизмов управления ими.

Список литературы

1. Бугайский К.А. Расширенная модель открытых систем (Часть1) / К.А. Бугайский, Д.С. Бирин, Б.О., Дерябин, С.О. Цепенда // Информация и безопасность. 2022. Т. 25. № 2 С. 169-178.

2. Воройский Ф. С. Информатика. Новый систематизированный толковый словарь-справочник (Введение в современные информационные и телекоммуникационные технологии в терминах и фактах). 3-е изд., перераб. и доп. / Воройский Ф. С. М: ФИЗМАТЛИТ, 2011. 760 с.

3. ГОСТ Р ИСО/МЭК ТО 10032–2007 Эталонная модель управления данными. М.: Стандартинформ, 2009.

4. Лимончелли Т. «Практика системного и сетевого администрирования» - 3-е издание пер. с англ. / Т. Лимончелли, К. Хоган, С. Чейлап – М: издательский дом Вильямс, 2018. 1104 с.

5. Информационная безопасность систем организационного управления : теорет. основы : монография : в 2 т. / Н. А. Кузнецов [и др.] ; под ред. Н. А. Кузнецова, В. В. Кульбы ; Рос. акад. наук, Ин-т проблем передачи информ. М.: Наука, 2006.

6. ГОСТ Р ИСО 10007-2007 Менеджмент организации. Руководящие указания по управлению конфигурацией. М.: Стандартинформ, 2008.

7. Р 50.1.031–2001: Информационные технологии поддержки жизненного цикла продукции. Терминологический словарь. Часть1. Стадии жизненного цикла продукции. М: ИПК Изд-во стандартов, 2011.

8. Бугайский К.А. Расширенная модель открытых систем (Часть2) / К.А. Бугайский, Д.С. Бирин, Б.О., Дерябин, С.О. Цепенда // Информация и безопасность. 2022. Т. 25. № 3. С. 321- 330.

9. Маркелов А.А. Введение в технологии контейнеров и Kubernetes / А.А. Маркелов. М: ДМКпресс 2019. 194 с.

10. Столяров А.В. Программирование: введение в профессию в 3 т. / А.В. Столяров. М: ДМКпресс, 2021.

11. Калашников А.О. Инфраструктура как код: формируется новая реальность информационной безопасности / А.О. Калашников, К.А. Бугайский // Информация и безопасность. 2019. Т. 22. № 4. С. 495-506.

Федеральное государственное бюджетное учреждение науки
Институт проблем управления им. В.А. Трапезникова РАН
V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences

Поступила в редакцию 3.11.2022

Информация об авторах

Бугайский Константин Алексеевич – ИПУ РАН, младший научный сотрудник, e-mail: kabuga@ipu.ru
Дерябин Богдан Олегович – ИПУ РАН, младший научный сотрудник, e-mail: baga_d@mail.ru
Табаков Кирилл Викторович – ИПУ РАН, младший научный сотрудник, e-mail: tabakov2002@mail.ru
Храмченкова Екатерина Сергеевна – ИПУ РАН, младший научный сотрудник, e-mail: hramchenkovaes@yandex.ru
Цепенда Сергей Олегович – ИПУ РАН, младший научный сотрудник, e-mail: tsependе.s@gmail.com

EXTENDED MODEL OF OPEN SYSTEMS (PART 3)

K.A. Bugayskiy, B.O. Deryabin, K.V. Tabakov, E.S. Hramchenkova S.O. Tsependa

In the third part of the article, the administration plane is developed, which is stated in the composition, but not described in the reference model of open OSR/RM systems. The administration plane is considered in the paradigm of management systems, that is, as consisting of management objects, management subjects and management mechanisms. The control objects are presented as sets of configuration parameters describing the connections and functioning of the hardware and software parts of the computing system. When describing the management entities, the relationship between the system users, their accounts and the corresponding user rights spaces is shown. The management mechanisms are considered both from the point of view of the internal structure and from the point of view of the operations performed by them. The description of the administration plane is carried out using the terminological and essential apparatus of the base plane of the model developed in the previous articles of the series. As part of the development of the administration plane, operators are proposed that display the relationship of the subject, object and control mechanism.

Keywords: open systems, information security, model, information system.

Submitted 3.11.2022

Information about the authors

Konstantin A. Bugayskiy – junior researcher, V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, e-mail: kabuga@ipu.ru

Bogdan O. Deryabin – junior researcher, V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, e-mail: baga_d@mail.ru

Kirill V. Tabakov – junior researcher, V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, e-mail: tabakov2002@mail.ru

Elena S. Hramchenkova – junior researcher, V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, e-mail: hramchenkovaes@yandex.ru

Sergey O. Tsependa – junior researcher, V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, e-mail: tsependе.s@gmail.com