Параметрическое управление кибербезопасностью UNIX-систем

М. О. Калинин

Ввод в эксплуатацию и обслуживание защищенной киберсистемы связаны с решением таких проблем, как распределение функций администрирования безопасности между субъектами и использование механизмов защиты на всех уровнях системы. Одна из самых масштабируемых сред — операционная система (ОС) UNIX — применяется в настоящее время практически во всех информационных сферах: от создания среды функционирования настольных приложений до развертывания базовой системы крупнейших вычислительных центров. Строгость системной архитектуры, ориентированной на обеспечение надежности работы и кибербезопасности, и ее открытость позволяют эксплуатировать ОС UNIX при построении защищенных киберсистем в качестве основы, обеспечивающей необходимый уровень защиты и реализацию политики безопасности при сохранении совместимости с существующими технологиями обработки информации. Однако многокомпонентность и сложность взаимосвязей структурных элементов любой современной защищенной операционной среды, в т. ч. ОС UNIX, зачастую не позволяют достаточно эффективно противостоять нарушениям безопасности и обеспечивать гарантированную защиту при обработке, хранении и передаче информации.

Действия по администрированию безопасности ОС крупных киберсистем и комплексов требует непрерывного контроля, анализа и управления параметрами безопасности, выявления и ликвидации уязвимостей безопасности, что в силу ограниченности человеческих возможностей не всегда результативно, о чем свидетельствует статистика киберпреступлений. В связи с этим возникает необходимость в построении систем управления безопасностью, которые были бы способны на начальном этапе эксплуатации контролируемой ОС выполнять несложные операции по первоначальной настройке параметров безопасности, а в дальнейшем контролировать текущие системные со-

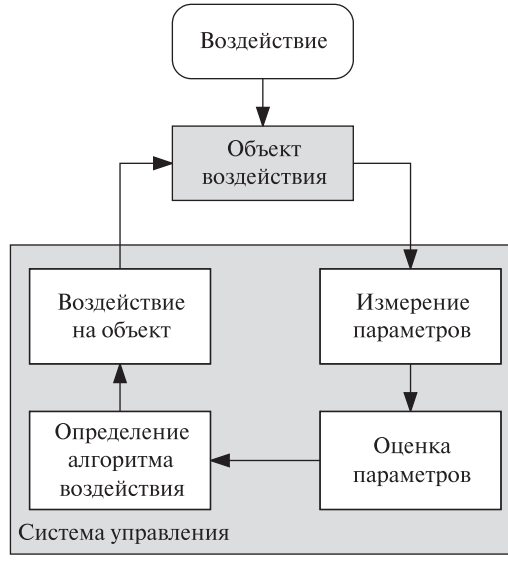

Рис. 1. Общая схема системы управления

стояния безопасности и адаптировать их к негативным воздействиям, оказываемым на систему. Для решения данной задачи автором предложен подход к параметрическому управлению кибербезопасностью информационных систем, построенных на базе ОС UNIX.

Решения задачи управления применимы к аспектам кибербезопасности за счет общности формального описания поведения систем, динамики происходящих процессов и алгоритмов выработки управляющих воздействий. При этом система управления кибербезопасностью функционирует по схеме *с обратной связью* (по замкнутому контуру), т. е. использует информацию о текущем состоянии параметров объекта воздействия (рис. 1) [1–3].

В приведенной схеме (рис. 1) происходит измерение значений заданного множества параметров объекта воздействия. Затем полученные параметры оцениваются по критериям, заложенным в систему управления. После этого в зависимости от полученных оценок происходит определение алгоритма воздействия, который затем применяется к объекту воздействия. Для механизмов безопасности ОС UNIX в качестве объекта воздействия будем рассматривать множество параметров безопасности данной ОС. Состав данного множества определяется разновидностью модели безопасности, реализованной

Каталог	Владелец			Группа			Остальные		
Открытие	Чтение	Запись	Исполнение	Чтение	Запись	Исполнение	Чтение	Запись	Исполнение
d (sticky)	**r**	**w**	**x (suid)**	**r**	**w**	**x (sgid)**	**r**	**w**	**x**

Рис. 2. Маска битов доступа в ОС UNIX

в ОС, и применением дополнительных средств контроля и управления доступом. В ОС UNIX традиционной применяется дискреционная модель безопасности [4].

Суть дискреционной модели безопасности заключается в разграничении доступа между субъектами и объектами. При этом субъектом называется активная сущность, которая может инициировать запросы ресурсов и использовать их для выполнения каких-либо заданий, а объектом — пассивная сущность, используемая для хранения или получения информации [5]. При исполнении субъектами операций происходит взаимодействие субъектов и объектов (доступ), в результате чего происходит перенос информации между ними. Разграничение доступа реализуется посредством установки прав доступа субъектам на объекты и соблюдения правил контроля и управления доступа, заданных в виде матрицы доступа. Права доступа в ОС UNIX задаются с помощью механизма битов доступа, устанавливаемых на объект [6]. Типичными правами доступа в ОС UNIX являются чтение (read), запись (write) и исполнение (execute). Маска битов доступа приведена на рис. 2. Биты доступа задают права трех категорий пользователей: владельца объекта (owner), членов группы владельца (group) и остальных субъектов (other) [6].

Помимо стандартных битов в ОС UNIX существует механизм, позволяющий пользователям запускать процессы от имени других пользователей, если одному пользователю необходимо на время предоставлять права другого (например, суперпользователя). Для этого применяется бит подмены идентификатора пользователя (suid-бит) или группы (sgid-бит). Этот бит используется совместно с битом исполнения (x) для обычных файлов. С целью предотвращения удаления файла посторонним пользователем в общедоступном каталоге применяется sticky-бит. Для каталогов его смысл заключается в том, что удалить или переименовать файл может только владелец файла. В обычном случае,

без sticky-бита, возможность операций над файлами определяется правом записи (w) на каталоги [6].

Таким образом, в ОС UNIX основное управление кибербезопасностью осуществляется через воздействие на биты доступа. При этом традиционно исходят из того, что к системе можно получить несанкционированный доступ путем различных проникновений. В таком случае для обеспечения кибербезопасности системы требуется постоянно проводимое исследование уязвимостей системы и расстановка битов доступа.

Предлагаемый автором альтернативный подход состоит в том, чтобы обеспечивать безопасность системы динамически в требуемых условиях безопасного функционирования. Под термином «условия безопасного функционирования» понимается совокупность правил, выполнение которых свидетельствует о работоспособности и безопасности системы. Применение таких условий требует предварительного их задания с учетом правил контроля и управления доступом, реализованных в модели безопасности ОС. Например, для ОС семейства Windows условия безопасного функционирования заданы разработчиком данной ОС и представлены в виде так называемых шаблонов безопасности. Для ОС UNIX такие нормы не определены, поэтому на основе открытого алгоритма работы системы контроля и управления доступом сформулируем общие условия безопасного функционирования для ОС UNIX.

Первостепенной задачей обеспечения безопасности ОС UNIX является защита объектов, доступных суперпользователю. Например, все системные файлы принадлежат суперпользователю, включая системные скрипты и утилиты, и злоумышленник, получив права суперпользователя и изменив маску прав доступа, может получить неограниченный доступ к системе. Поэтому основное условие безопасного функционирования состоит в том, что права записи для файлов, владельцем которых является суперпользователь, должны быть сняты.

Suid-механизм позволяет предоставлять функции суперпользователя другим пользователям, например, для исполнения системных утилит. Наличие файлов с suid-битом (suid-файлов) содержит потенциальную угрозу безопасности системы, поэтому следующее условие безопасного функционирования состоит в том, что бит подмены идентификатора на системных скриптах должен быть снят и права записи на файлах с установленным suid-битом должны быть запрещены.

Следующий аспект безопасности ОС UNIX заключается в защите каталогов. Если на каталог установлено право записи, то любой файл внутри этого каталога можно удалить, даже если на сам файл такое право не установлено. Таким образом, права на каталог и на файлы в нем должны быть согласованы. Условие безопасного функционирования состоит в том, что если на каталог установлен бит записи, то такой же бит должен стоять на каждом файле, находящемся в данном каталоге. Если же не на всех файлах такое разрешение установлено, то тогда бит записи с каталога должен быть снят.

Условие безопасного функционирования для защиты пользовательских данных состоит в том, что только владелец должен иметь право записи для своего домашнего каталога и файлов, хранящихся в нем.

На системные журналы (logs) и на каталог /proc должны быть установлены только права чтения. В этих каталогах содержится различная системная информация, например информация о процессах выполнения программ, данные о процессоре, информация о RAM, список параметров, передаваемых ядру при загрузке. Для каталога /dev также должны быть установлены только право чтения, так как в нем хранятся файлы устройств, изменения в которых могут привести к сбоям в работе системы.

Перечисленные условия безопасного функционирования распределим по степени критичности нарушения, возникающего в случае невыполнения соответствующего условия (табл. 1, наиболее критичное нарушение — № 1).

Таблица 1

Условия безопасного функционирования

Степень критичности нарушения	**Условие безопасного функционирования**
1	Запрет записи для системных файлов
2	Запрет suid-бита для системных скриптов
3	Запрет записи для всех suid-файлов
4	Только чтение для каталогов /proc и /dev
5	Соответствие прав для каталогов и его файлов
6	Защита домашних каталогов
7	Только чтение для системных журналов

Рассмотрим процедуру выработки управляющего воздействия на множество параметров безопасности ОС UNIX на основе предложенных условий безопасного функционироваң. Допустим, изначально система находилась в состоянии *S,* которое будем считать безопасным. В каждый последующий момент времени система находится в некотором состоянии *S′,* которое необходимо оценить согласно условиям безопасного функционирования и при необходимости адаптировать к обнаруженным нарушениям. Определим для каждого условия безопасного функционирования логическую функцию нарушения C_i, $i \in [1, 7]$. Функция C_i принимает значение «истина», если соответствующее условие безопасного функционирования не выполняется в системе. Эти функции составляют множество функций нарушений *C***.** Общая функция нарушения, выявляющая нарушения условий безопасного функционирования, для данного набора *C* в системном состоянии *S′*, представим в виде:

$$F(C) = C_1 \,||\, C_2 \,||\, C_3 \,||\, C_4 \,||\, C_5 \,||\, (C_6 \,\&\&\, C_7)\textbf{.}$$

Функция состоит из нескольких логических слагаемых, объединенных операцией логического сложения, и логических множителей, объединенных операцией логического умножения. Ее вид согласуется с введенной степенью критичности нарушения: условия с малой критичностью C_6 и C_7 объединены логическим умножением, что означает срабатывание при одновременном нарушении этих условий. Таким образом, для сообщения о нарушении безопасности достаточно нарушения любого из высококритичных условий, либо одновременного нарушения некритичных условий.

Если обнаружено, что общая функция нарушения истинна, то происходит анализ и поиск нарушенного условия или группы условий. Данный процесс допускает автоматизацию, поскольку основан на выполнении операций над множествами. После нахождения нарушений производятся управляющие воздействия на систему для перевода системы в безопасное состояние, удовлетворяющее условиям безопасного функционирования (табл. 2).

Таблица 2

Управляющие воздействия

Именование управляющего воздействия	**Накладываемая маска битов доступа**
RfO (Read for Owner)	{100 000 000}
NoW (No Write)	{101 101 101}
NoSaS (No Suid and Sgid)	{110 110 111}
WfO (Write for Owner)	{111 101 101}

В табл. 2 значения битов доступа соответствующего воздействия условно обозначены единицами и нулями. При пересечении множеств прав объекта с определенной маской, права соответствующие нулям в уровне будут убраны. Например, на файл установлены права *rw-r-xr--*, что представимо в виде множества {110 101 100}. При пересечении этих прав с уровнем RfO производится обнуление лишних прав ({110 101 100}∩{100 000 000} = {100 000 000}), т. е. права файла в результате управляющего воздействия должны принять вид *r-- --- ---*.

Для каждого нарушенного условия безопасного функционирования существует свое управляющее воздействие по ликвидации последствий нарушения (табл. 3, где O — множество объектов системы; *rights*(o) — маска прав доступа объекта o; o_{root} — объект, владельцем которого является суперпользователь, т. е., фактически, любой системный файл; $o_{sysscript}$ — объект, соответствующий системному скрипту, т. е., фактически, любой исполняемый системный файл, содержащий программный код; o_{suid} — объект с установленным битом подмены идентификатора; $o_{dev/proc}$ — объекты каталогов */dev* и */proc*, o_{dir} — обычный каталог, o_{home} — объекты домашнего каталога пользователя; o_{log} — системный журнал аудита).

Если нарушено условие запрета записи для системных файлов (файлов суперпользователя), то необходимо снять все биты записи. Если нарушено условие запрета установки бита подмены идентификатора для системных скриптов, то необходимо снять биты подмены идентификатора и группового идентификатора. Если нарушено условие

Таблица 3

Соответствие управляющих воздействий условиям безопасного функционирования

Условие безопасного функционирования	**Управляющее воздействие**
C1	$\forall$oroot $\in$ O {rights(oroot)} $\cap$ NoW
C2	$\forall$osysscript $\in$ O {rights(osysscript)} $\cap$ NoSaS
C3	$\forall$osuid $\in$ O {rights(osuid)} $\cap$ NoW
C4	$\forall$odev/proc $\in$ O {rights(odev/proc)} $\cap$ RfO
C5	$\forall$odir $\in$ O {rights(odir)} $\cap$ NoW
C6	$\forall$ohome $\in$ O {rights(ohome)} $\cap$ WfO
C7	$\forall$olog $\in$ O {rights(olog)} $\cap$ RfO

запрета записи для файлов с suid-битом, то необходимо снять все биты записи. Если на каталоги /*proc* и /*dev* и файлы, содержащиеся в них, установлено не только право чтения, то необходимо оставить только бит чтения для владельца. Если нарушено условие соответствия прав для каталога и файлов, то для каталога необходимо убрать все права записи. Если нарушено условие защиты домашних каталогов, то необходимо снять биты записи для групп и остальных пользователей. Если на системных журналах установлено не только право чтения, то необходимо установить право чтения только владельцем, что позволит запретить записи системных журналов и поможет проследить следы нарушителя

Рассмотренное множество условий безопасного функционирования является базовым. Оно расширяемо путем добавления специфических ограничений. Например, если потребовалось, чтобы к конфиденциальным файлам в каталоге */home/user*1 доступ имел только владелец и только на чтение, то добавляется новое условие безопасного функционирования. Новые условия должны вводиться только после окончания цикла управления, что позволит гарантировать неизменность набора условий в течение одного рабочего цикла системы управления.

Таким образом, схема системы управления кибербезопасностью для ОС UNIX, реализующая представленные принципы управления, приведена на рис. 3.

Рассмотрим пример функционирования системы управления кибербезопасностью ОС UNIX. Допустим, что в каталоге */home/user/dir* находится четыре файла: *1.txt* — пользовательский текстовый файл; *2.sh* — скрипт суперпользователя; *log* — журнал аудита; *rootfile* — системный файл. Права файлов изначально представлены в виде матрицы доступа (рис. 4). Контроль параметров безопасности ОС UNIX осуществляется согласно набору функций нарушений C_1, C_2, C_3 и C_7. Матрица доступа вместе с множеством условий безопасного функционирования описывает безопасность состояния смоделированной системы. На вход алгоритму управления кибербезопасностью подается текущее значение матрицы доступа и набор условий безопасного функционирования. Производится расчет функций нарушений: $C_1 = 1$, так как у *rootfile* присутствует право записи; $C_2 = 1$, так как присутствует бит подмены идентификатора; $C_3 = 0$, так как у файла *2.sh* отсутствует право записи; $C_7 = 1$, так как у файла *log*, присутствует право записи. Расчет значения общей функции нарушений производится следующим образом:

$$F(C) = C_1 \,||\, C_2 \,||\, C_3 \,||\, C_{7=} 1 \,||\, 1 \,||\, 0 \,||\, 1 = 1,$$

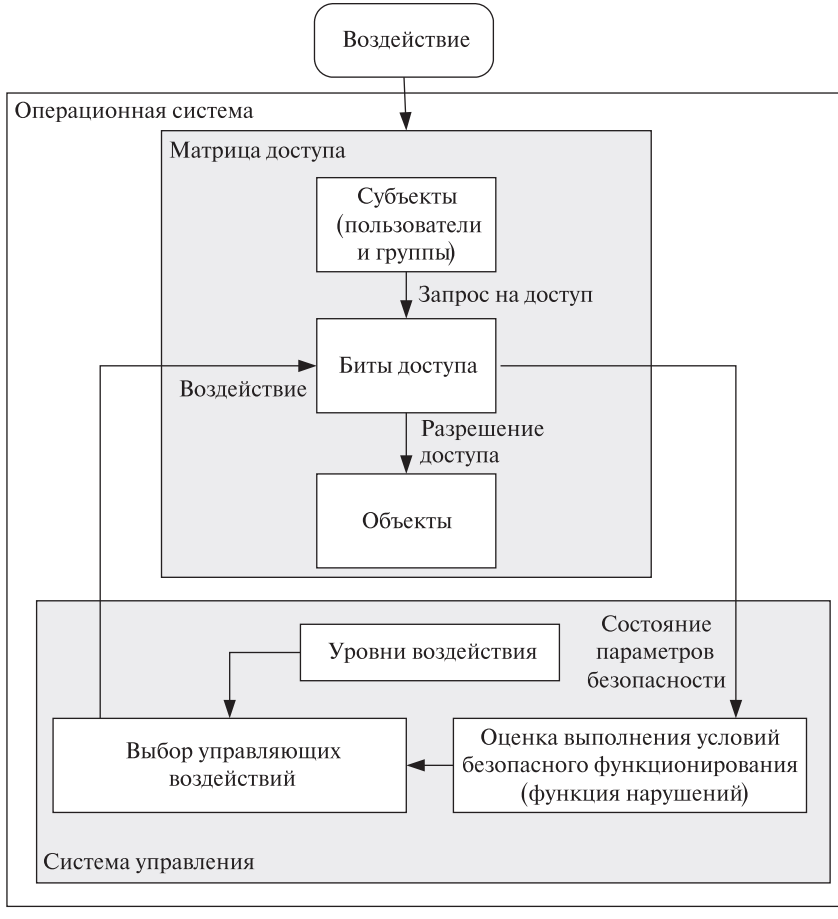

Рис. 3. Схема системы управления кибербезопасностью ОС UNIX

т. е. необходимо управляющее воздействие на параметры безопасности. Поскольку сработали функции C_1, C_2 и C_7, то в зависимости от нарушенного условия осуществляются следующие действия: для C_1 — для *rootfile* выполняется команда *chmod a-w rootfile*; C_2 — для *2.sh* команда *chmod ug-s 2.sh*; C_7 — для *log* команда *chmod a-w log*. Аргументы *a-w* и *us-g* команды *chmod* представляют собой уровни воздействия NoW и NoSaS, соответственно.

В итоге на выходе имеем состояние системы, представленное на рис. 5.

	$1.txt$	$2.sh$	log	$rootfile$
Владелец	**rw**	**rs**	**rw**	**rw**
Группа	**r**	**rx**	**r**	**r**
Остальные	**r**	**rx**	**r**	**r**

Рис. 4. Исходная матрица доступа

	$1.txt$	$2.sh$	log	$rootfile$
Владелец	rw	**rx**	**r**	**r**
Группа	**r**	**rx**	**r**	**r**
Остальные	**r**	**rx**	**r**	r

Рис. 5. Преобразованная матрица доступа

Предположим, что в течение рабочего цикла управления были изменены права у файла *1.txt* (например, добавлено право записи для остальных пользователей) и было расширено множество критериев путем добавления функции нарушения C_8, соответствующее специальному условию безопасного функционирования: «Для пользовательских файлов в каталоге */home/user1/dir* разрешена запись только владельцу». Определим для введенной функции приоритет ниже приоритет C_7, т. е. сигнал о нарушении отдельного условия поступит, но позволим его не обрабатывать, поскольку нарушение классифицировано как незначительное. В общей функции нарушения функции C_7 и C_8 объединяются с помощью логического умножения. Введение новой функции нарушения определяет расширение множества управляющих воздействий за счет добавления воздействия WfO к уже использующимся воздействиям NoW и NoSaS. После этого выполняется новый цикл управления: функции нарушений C_1, C_2, C_3 и C_7 не выполнены, а функция C_8 выполнена, т. е. общая функция нарушений $F(C) = C_1 \,||\, C_2 \,||\, C_3 \,||\, (C_7 \,\&\&\, C_8) = 0$, т. е. управляющее воздействие не требуется, так как для функции нарушения C_8 определен самый низкий приоритет и она не повлияла на результирующее значение функции F. Следовательно, состояние рассматриваемой системы не должно изменяться.

Таким образом, предложенный подход к параметрическому управлению кибербезопасностью ОС UNIX позволяет в значительной мере упростить решение задачи автоматизации администрирования систе-

мы, контроля системных параметров безопасности и реакции на нарушения безопасности путем построения системы управления, берущей на себя выполнение операций по оценке защищенности системного состояния и оказания своевременных воздействий на множество параметров безопасности с целью предотвращения эксплуатации выявленных нарушений.

Литература

1. *Ким Д. П.* Теория автоматического управления. Т. 1: Линейные системы. М.: Физматлит, 2007. 312 с.
2. *Ким Д. П.* Теория автоматического управления. Т. 2: Многомерные, нелинейные, оптимальные и адаптивные системы. М.: Физматлит, 2004. 464 с.
3. *Александров А. Г.* Оптимальные и адаптивные системы. М.: Высшая школа, 1989. 263 с.
4. *Harrison M. H., Ruzzo W. L., Ullman J. D.* Protection in Operating Systems, Communications of the ACM, 19(8):461–471. 1976.
5. Защита от несанкционированного доступа к информации. Термины и определения. М.: Гостехкомиссия России, 1998.
6. *Лепаж И., Яррера П.* UNIX. Библия пользователя. М.: Вильямс, 2001. 640 с.