Секция 3

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

УДК 519.17

АНАЛИЗ НАДЕЖНОСТИ ГРАФИЧЕСКИХ CAPTCHA-СИСТЕМ НА ПРИМЕРЕ ПРОЕКТА KCAPTCHA

М. Б. Абросимов, А. А. Маторин

Тест Тьюринга — тест, предложенный Аланом Тьюрингом в 1950 г. Тьюринг предложил исследовать возможность машин делать то, что могут делать люди как мыслящие создания. Тест проходит следующим образом. Человек задает вопросы в письменном виде человеку и компьютеру. Его задача — определить, с кем он общается. В 1952 г. ученый предложил другую версию теста, получившую название «Стандартная интерпретация». В этой версии жюри задает вопросы компьютеру, а роль компьютера состоит в том, чтобы заставить значительную часть членов жюри поверить, что он на самом деле человек.

Существуют модификации теста Тьюринга, в которой роли машины и человека поменяли местами. Такие тесты называются обратными тестами Тьюринга. CAPTCHA — это разновидность обратного теста. CAPTCHA от англ. «Completely Automated Public Turing test to tell Computers and Humans Apart» — полностью автоматизированный публичный тест Тьюринга для различения компьютеров и людей. Основная идея теста — предложить пользователю такую задачу, которую может решить человек, но которую несоизмеримо сложно предоставить для решения компьютеру.

CAPTCHA чаще всего используется при необходимости предотвратить использование интернет-сервисов ботами, в частности для предотвращения автоматических регистраций почтовых ящиков, отправок сообщений, скачивания файлов, массовых рассылок.

Актуальность применения CAPTCHA можно увидеть, например, из статистики объемов рассылаемого спама. Более 97 % электронных сообщений, отправляемых через Интернет, являются спамом. Применение CAPTCHA-защиты позволяет усложнить задачу регистрации почтовых ящиков ботами.

В наиболее распространённом варианте CAPTCHA от пользователя требуется ввести символы, как правило, изображённые на предлагаемом ему рисунке в искажённом виде. Альтернативами являются аудио-CAPTCHA, математические примеры, текстовые задачи, задачи на распознавание предметов.

Для анализа надежности графических CAPTCHA была разработана компьютерная программа на языке Java и проанализирована одна из распространенных систем — проект KCAPTCHA.

KCAPTCHA [1] — это готовое решение с открытым исходным кодом для генерации графических капч (картинок с проверочным текстом), изначально написанное на языке PHP. На данный момент имеется порт на платформу Java. При генерации используется около 20 шрифтов и применяются волновые алгоритмы искажения, которые являются наиболее сложными для автоматического распознавания, но в то же

время искаженные символы остаются довольно легко читаемыми человеком. Подобные алгоритмы искажения применяются при генерации многих других защищенных капч. KCAPTCHA была выбрана для анализа устойчивости графических капч к автоматическому распознаванию.

Распознавание состоит из нескольких этапов. На начальном этапе происходит очистка изображения. Цвет фона определяется после просмотра крайней области изображения, в которой не содержится символов. Всем пикселям, цвета которых отличаются от фонового не более чем на заданную константу, присваивается цвет фона.

Распознавание отдельных символов происходит двумя основными способами. Первый способ — распознавание на основе количества «ног» символа и распознавание на основе количества «ног» повернутого символа. Например, символ «m» стоит на трех «ногах». Символ «c», повернутый на 0, 180 и 270°, стоит на одной «ноге»; символ «c», повернутый на 90°, — на двух «ногах».

Второй способ — распознавание на основе анализа контрольных точек скелета изображения и связей между ними. Для выделения скелета используется алгоритм утончения изображения Зонга — Суня [2]. *Контрольными точками* или *вершинами* называются точки, лежащие на пересечении двух или более отрезков, и точки на концах этих отрезков. Для каждой контрольной точки определяются и анализируются их соседи и пути до них.

Такие тесты были написаны для цифр и букв английского алфавита. Строчные и прописные буквы не различаются. При стандартных настройках системы KCAPTCHA точность распознавания отдельных символов лежит в диапазоне от 87 до 99 %. Время распознавания одного символа при работе одного ядра процессора мощностью 2,4 ГГц составляет примерно 30 мс.

Общий алгоритм распознавания выглядит следующим образом.

1. Находится скелет изображения.

2. Находятся контрольные точки скелета изображения и сортируются по X-координате.

3. Для первой вершины находятся все контрольные точки, до которых существует маршрут от данной вершины. Если разница X-координат текущей вершины и любого ее соседа меньше заданной константы, то список этих вершин передается на распознавание. Если символ распознан, выбирается новая текущая вершина и алгоритм продолжается. Иначе распознаваемый символ имеет общие точки со своим соседом. Необходимо их разделить. Для этого просматриваются вершины, имеющие трех или более соседей. Если вершина удовлетворяет определенным критериям, то из списка ее соседей удаляются некоторые вершины. Новый список соединенных вершин передается на распознавание. Если символ распознан, его вершины помечаются как просмотренные и происходит переход к началу алгоритма.

Эффективность алгоритма зависит от количества символов, имеющих общие точки со своими соседями, и от количества мест соединений различных символов. Наибольшую сложность при автоматическом распознавании вызывает сегментирование символов, имеющих общие точки.

ЛИТЕРАТУРА

1. `http://www.captcha.ru/kcaptcha/` — Проект KCAPTCHA. 2011.
2. *Zhang T. Y. and Suen C. Y.* A fast parallel algorithm for thinning digital patterns // Comm. ACM. 1984. V. 27. No. 3. P. 236–239.

УДК 004.94

О РЕЗУЛЬТАТАХ РАЗРАБОТКИ РОЛЕВОЙ ДП-МОДЕЛИ ДЛЯ ОПЕРАЦИОННЫХ СИСТЕМ СЕМЕЙСТВА LINUX[1]

П. Н. Девянин

Рассматривается развивающая ролевые ДП-модели [1, 2] ролевая ДП-модель управления доступом и информационными потоками в операционных системах (ОС) семейства *Linux* (или, сокращенно, РОСЛ ДП-модель), в которую по сравнению с БРОС ДП-моделью внесены следующие основные изменения.

Виды прав доступа ($read_r$, $write_r$, $execute_r$, own_r) и доступов ($read_a$, $write_a$, own_a) заданы в соответствии с реализуемыми в ОС семейства *Linux* правилами дискреционного управления доступом. При задании иерархии сущностей (отношения частичного порядка $\leqslant$ на множестве сущностей E) и функции иерархии сущностей (H_E) учтено наличие механизма создания «жестких» ссылок (*hard link*) в файловой системе ОС рассматриваемого семейства, обеспечивающего возможность размещения сущностей-объектов одновременно в нескольких сущностях-контейнерах.

В РОСЛ ДП-модели впервые по существу анализируется механизм ограничений (описанный в БР ДП-модели) на значения множеств авторизованных ролей учетных записей пользователей ($Constraint_U$), прав доступа ролей ($Constraint_P$) и текущих ролей субъект-сессий ($Constraint_S$). Этот механизм включен в модель для реализации в перспективе на основе ролевого управления доступом востребованного во многих защищенных ОС мандатного управления доступом.

Наибольшие изменения по сравнению с БРОС ДП-моделью внесены в описания условий и результатов применения правил преобразования состояний, в которых учтено наличие механизма ограничений и «жестких» ссылок. Правило создания сущности заменено двумя правилами — создания объекта и создания контейнера; добавлены правила создания и удаления «жесткой» ссылки, удаления доступа и получения субъект-сессией при наличии доступа владения к другой субъект-сессии всех ее информационных потоков.

В рамках РОСЛ ДП-модели по аналогии с существующими ДП-моделями рассматриваются монотонные правила преобразования состояний, которые по определению не приводят к удалению из состояний: ролей из множества текущих ролей субъект-сессий; прав доступа ролей к сущностям; субъект-сессий, сущностей или «жестких» ссылок на сущности-объекты; доступов субъект-сессий к сущностям; информационных потоков. Наличие в модели механизма ограничений в общем случае требует использования монотонных и немонотонных правил при передаче прав доступа ролей или возникновения информационных потоков. С учетом этого формулирование и обоснование алгоритмически проверяемых условий передачи прав доступа ролей или возникновения информационных потоков между сущностями целесообразно осуществлять для некоторых заданных в конкретных системах множеств ограничений. Для этого предлагается рассматривать ограничения, инвариантные относительно немонотонных правил преобразования состояний, каждое из которых по определению обладает следующим свойством: если в системе задано только ограничение данного вида, то для любой траектории системы применение или неприменение на ней любого немонотонного правила не влияет на выполнение ограничений у последующих за ним правил преобразования

[1]Работа выполнена при поддержке гранта МД-2.2010.10.

состояний. С использованием обозначений БРОС ДП-модели [2] дадим определение и сформулируем утверждение.

Определение 1. Ограничение инвариантно относительно немонотонных правил преобразования состояний в системе $\Sigma(G^*, OP)$, если при условии, что в ней задано только данное ограничение, для любых состояния системы G_0, немонотонного правила преобразования состояний op_1, правил преобразования состояний op_2, …, op_N, где $N > 1$, справедливо следующее: если $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_{N-1}} G_{N-1}$, $G_0 \vdash_{op_2} G'_2 \vdash_{op_3} \ldots \vdash_{op_{N-1}} G'_{N-1}$ и в состоянии G_{N-1} выполнены ограничения, заданные в условиях применения правила op_N, то эти ограничения выполнены в состоянии G'_{N-1}.

Утверждение 1. Пусть G_0 — начальное состояние системы $\Sigma(G^*, OP, G_0)$, в котором все ограничения инвариантны относительно немонотонных правил преобразования состояний, и функции $(i_u, i_e, i_r, i_s)_0$ удовлетворяют условиям предположения 7 БРОС ДП-модели [2]. Пусть также существуют состояния системы G_1, …, G_N и правила преобразования состояний $op_1, \ldots, op_N$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$, где $N \geqslant 0$. Тогда существуют состояния G'_1, …, G'_M, где $M \geqslant 0$, и монотонные правила преобразования состояний op'_1, …, op'_M, такие, что $G_0 \vdash_{op'_1} G'_1 \vdash_{op'_2} \ldots \vdash_{op'_M} G'_M$ и выполняются следующие условия:

1. Верно включение $S_N \subset S'_M$ и для каждой субъект-сессии $s \in S_N$ выполняются условия $user_N(s) = user'_M(s)$, $roles_N(s) \subset roles'_M(s)$.

2. Верно включение $E_N \subset E'_M$, для каждой сущности $e \in E_N \setminus S_N$, не являющейся субъектом, выполняется условие $H_{E_N}(e) \subset H'_{E_M}(e)$, и для любых сущностей e, $e' \in E_N$ если в состоянии G_N выполняется условие $e < e'$, то данное условие выполняется в состоянии G'_M.

3. Для каждой роли $r \in R$ выполняется условие $PA_N(r) \subset PA'_M(r)$.

4. Верно включение $A_N \subset A'_M$.

5. Верно включение $F_N \subset F'_M$.

6. Функции $(i_u, i_e, i_r, i_s)'_M$ удовлетворяют условиям предположения 7 БРОС ДП-модели [2].

Из утверждения 1 следует, что при наличии в системе только ограничений, инвариантных относительно немонотонных правил преобразования состояний, при анализе условий передачи прав доступа ролей, реализации информационных потоков достаточно использовать только монотонные правила преобразования состояний.

Таким образом, РОСЛ ДП-модель позволяет анализировать перспективные для ОС семейства *Linux* механизмы защиты: ролевое управление доступом, включая ограничения, и мандатный контроль целостности. В дальнейшем планируется развитие модели с целью включения в нее новых элементов, более точно учитывающих особенности функционирования ОС рассматриваемого семейства, и выработки научно-обоснованных технических решений, направленных на совершенствование механизмов защиты информации в ОС.

ЛИТЕРАТУРА

1. *Девянин П. Н.* Модели безопасности компьютерных систем. Управление доступом и информационными потоками. Учеб. пособие для вузов. М.: Горячая линия-Телеком, 2011. 320 с.

2. *Девянин П. Н.* Правила преобразования состояний базовой ролевой ДП-модели управления доступом и информационными потоками в операционных системах // Прикладная дискретная математика. 2011. № 1(11). С. 78–95.

УДК 004.94

РАЗРАБОТКА РОЛЕВОЙ МОДЕЛИ БЕЗОПАСНОСТИ УПРАВЛЕНИЯ ДОСТУПОМ И ИНФОРМАЦИОННЫМИ ПОТОКАМИ КОМПЬЮТЕРНОЙ СИСТЕМЫ SELinux[1]

М. А. Качанов

В настоящее время наиболее рапространенным методом реализации безопасного управления доступом и информационными потоками в операционных системах (ОС) семейства $GNU/Linux$ является применение программного средства $SELinux$, в связи с чем возникает задача разработки формальной модели его безопасности. В данной работе решается задача анализа безопасности управления доступом и информационными потоками в компьютерной системе (КС) $SELinux$, предлагаются ролевая модель безопасности данной КС, алгоритм проверки возможности получения права доступа и реализации информационного потока, а также метод применения данной модели на практике для анализа безопасности рассматриваемой КС.

Будем использовать основные понятия и обозначения теории ДП-моделей, считая, что моделируемая КС представляется системой, каждое состояние в которой задаётся набором объектов, а каждый переход из состояния в состояние осуществляется в результате применения одного из правил преобразования состояний. Определим новые элементы ДП-модели, необходимые для адекватного анализа безопасности КС $SELinux$:

T — множество типов сущностей;

M — множество известных классов сущностей;

R_M — множество прав доступа, допустимых для известных классов сущностей;

$label : E \to U \times R \times T$ — функция, сопоставляющая сущности контекст безопасности (метку);

$user : E \to U$, $role : E \to R$, $type : E \to T$ — такие функции, что если $e \in E$ и $label(e) = (u, r, t)$, то $user(e) = u$, $role(e) = r$, $type(e) = t$;

$class : E \to M$ — функция, сопоставляющая каждой сущности известный класс;

$allow\_role : R \to 2^R$ — функция, сопоставляющая каждой роли множество ролей, которые она может занять;

$role\_types : R \to 2^T$ — функция, сопоставляющая каждой роли множество типов субъектов, к которым ей разрешено получать доступ;

$user\_roles : U \to 2^R$ — функция, сопоставляющая каждому пользователю множество ролей, на которые он может быть авторизован;

$class\_perms : M \to 2^{R_M}$ — функция, сопоставляющая каждому известному классу сущностей набор прав доступа, к нему применимый;

$R_r = \{(c, p) : c \in M, p \in class\_perms(c)\}$ — множество видов прав доступа;

$P \subseteq S \times E \times (R_r \cup \{own_r\})$ — множество текущих прав доступа субъектов к сущностям;

функции $fa : U \times E \to 2^E$, $fp : U \times E \to 2^E$, $ft : S \times E \times R_r \to 2^E$, $type\_rights : T \times T \to 2^{R_r}$, $role\_transition : R \times T \to R$, $type\_transition : T \times T \times M \to 2^T$, $login : R \times T \to 2^{R \times T}$, $constrain : R_r \times (U \times R \times T)^2 \to \{0, 1\}$, $validatetrans : M \times (U \times R \times T)^3 \to \{0, 1\}$.

[1]Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг. (гос. контракт № П1010).

Определение 1. Иерархия известных классов сущностей — это заданное на множестве M отношение частичного порядка $\leqslant_M$, такое, что

$$\forall m_1, m_2 \in M \ (m_1 \leqslant_M m_2 \Leftrightarrow class\_perms(m_1) \supseteq class\_perms(m_2)).$$

Определение 2. Совокупность объектов $G = (U, U_L, R, E, S, L_S, T, F, M, R_M, P, label, allow\_role, role\_types, user\_roles, class\_perms, type\_rights, role\_transition, type\_transition, H_E, login, constrain, validatetrans, \leqslant_M)$ будем называть состоянием системы.

В модели определены 19 правил преобразования состояний, образующих множество OP.

Используем следующие обозначения:

$\Sigma(G^*, OP)$ — система, где G^* — множество всех возможных состояний; OP — множество правил преобразования состояний;

$G \vdash_{op} G'$ — переход системы из состояния G в состояние G' с использованием правила преобразования состояний $op \in OP$.

Предположение 1. В рамках модели КС *SELinux* на траекториях функционирования системы доверенные пользователи не создают новых субъектов, доверенные субъекты не участвуют в передаче прав доступа, не реализуют информационных потоков по времени, не создают субъектов, не используют права доступа владения к другим субъектам.

Предположение 2. В рамках модели КС *SELinux* на траекториях функционирования системы не изменяются множества U, U_L, L_S, R, T, M, R_M, отношение $\leqslant_M$ на множестве M, функции $allow\_role$, $role\_types$, $user\_roles$, $class\_perms$, $type\_rights$, $role\_transition$, $type\_transition$, $login$, предикаты $constrain$, $validatetrans$.

Ввиду предположения 2 состояние системы будем записывать как $G = (S, E, P, F, H_E, class)$.

Определение 3. Система $\Sigma(G^*, OP)$ с множеством правил преобразования состояний OP называется ДП-моделью КС *SELinux*, если её состояния подчиняются определению 2 и удовлетворяют предположениям 1 и 2.

Для формализации возможности получения права доступа и реализации информационного потока сформулируем определения предикатов безопасности ДП-модели КС *SELinux*.

Определение 4. Пусть $G_0 = (S_0, E_0, P_0, F_0, H_{E0}, class_0)$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют пользователь $u \in U$, право доступа $\alpha_r \in R_r$ и сущность $e \in E_0$. Определим предикат $can\_share(u, e, \alpha_r, G_0)$, который будет истинным тогда и только тогда, когда существуют состояния $G_1, \ldots, G_N = (S_N, E_N, P_N, F_N, H_{EN}, class_N)$ и правила преобразования состояний $op_1, \ldots, op_N$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$ и существует субъект $x \in S_N$, такой, что $user(x) = u$, и $(x, e, \alpha_r) \in P_N$, где $N \geqslant 0$.

Определение 5. Пусть $G_0 = (S_0, E_0, P_0, F_0, H_{E0}, class_0)$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют сущности $x, y \in E_0$. Определим предикат $can\_write\_memory(x, y, G_0)$, который будет истинным тогда и только тогда, когда существуют состояния $G_1, \ldots, G_N = (S_N, E_N, P_N, F_N, H_{EN}, class_N)$ и правила преобразования состояний $op_1, \ldots, op_N$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$ и $(x, y, write_m) \in F_N$, где $N \geqslant 0$.

Определение 6. Пусть $G_0 = (S_0, E_0, P_0, F_0, H_{E0}, class_0)$ — состояние системы $\Sigma(G^*, OP)$, в котором существуют сущности $x, y \in E_0$. Определим предикат $can\_write\_time(x, y, G_0)$, который будет истинным тогда и только тогда, когда существуют состояния $G_1, \ldots, G_N = (S_N, E_N, P_N, F_N, H_{EN}, class_N)$ и правила преобразования состояний $op_1, \ldots, op_N$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_N} G_N$ и $(x, y, write_t) \in F_N$, где $N \geqslant 0$.

Замечание 1. Заметим, что для проверки истинности предикатов $can\_share(u, e, \alpha_r, G_0)$, $can\_write\_memory(x, y, G_0)$ и $can\_write\_time(x, y, G_0)$ в соответствии с определением требуется учесть все траектории функционирования КС, что не осуществимо на практике.

В связи с замечанием 1 представляется целесообразным сформулировать и обосновать алгоритмы проверки истинности предикатов can_share, can_write_memory и can_write_time. Такие алгоритмы реализуют преобразование начального состояния КС в его замыкание и позволяют проверить истинность предикатов для всех пользователей, сущностей и прав доступа одновременно. В работе вводятся определения замыканий ДП-модели КС $SELinux$ ($time$-замыкания и $memory$-замыкания), предлагаются и обосновываются алгоритмы их построения.

Предлагается также метод применения построенной модели на практике для проверки возможности получения права доступа и реализации информационного потока в КС $SELinux$. Метод состоит из двух основных этапов. Первый этап — это построение начального состояния предложенной ДП-модели КС $SELinux$ по набору конфигурационных файлов КС. Второй этап — это построение $time$-замыкания полученного состояния и интерпретация полученных результатов с точки зрения КС $SELinux$. На входе метод имеет весь набор необходимых конфигурационных файлов КС $SELinux$, а на выходе — ответ на вопрос, возможны ли получение заданного права доступа или реализация заданного информационного потока.

УДК 004.94

ОСОБЕННОСТИ РАЗРАБОТКИ ДП-МОДЕЛЕЙ СЕТЕВОГО УПРАВЛЕНИЯ ДОСТУПОМ[1]

Д. Н. Колегов

Работа посвящена особенностям построения ДП-моделей компьютерных систем (КС), реализующих сетевое управление доступом. Такие модели будем называть сокращенно СУД ДП-моделями, а при их описании используем основные определения и обозначения из [1].

Механизмы сетевого управления доступом в современных КС, как правило, обладают следующими свойствами, затрудняющими применение элементов и средств существующих ДП-моделей для их описания и исследования:

— распределенностью компонентов управления доступом и их сетевым взаимодействием;
— динамическим управлением доступом субъектов к сущностям на основе правил доступа и, как следствие, предоставлением субъектам различных прав доступа в зависимости от истинности условий того или иного правила доступа;

[1]Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг. (гос. контракт № П1010).

— наличием различных правил управления доступом для одних и тех же субъектов доступа;
— принадлежностью сущностей одновременно нескольким иерархиям и использованием последних при определении и проверке правил управления доступом.

Для адекватного отражения этих особенностей КС с сетевым управлением доступом в ДП-моделях предлагается язык описания последних расширить следующим образом.

1. Вместо одной функции иерархии сущностей вводится множество $H = \{H_1, \ldots, H_n\}$, где $H_i : E \to 2^E$ для $i = 1, \ldots, n$ есть функция иерархии сущностей. Например, такой набор функций соответствует иерархиям типов, назначения и расположения узлов сетевой КС управления доступом Cisco Access Control Server.

2. В дополнение к сущностям, параметрически ассоциированным с субъектом, вводятся множества сущностей, параметрически ассоциированных с субъектом, или ПАС-множества. Сущности из ПАС-множества $Z \subseteq E$ называются параметрически ассоциированными с субъектом $s \in S$ в состоянии G, если данные в этих сущностях позволяют идентифицировать вид преобразования данных, реализуемого субъектом s в этом или последующих состояниях КС. Наличие у субъекта нескольких ПАС-множеств является типичной ситуацией в современных КС. Например, одно из ПАС-множеств может содержать сущность-пароль и IP-адрес в базе данных сетевой системы управления доступом КС, а второе — сущность-пароль в базе данных программного обеспечения, реализующего удаленный доступ. Другим примером наличия нескольких ПАС-множеств является реализация в операционной системе механизмов двухфакторной аутентификации. В этом случае первое ПАС-множество содержит сущности, являющиеся частями разделенного пароля, одна часть которого может храниться на аппаратном носителе, другая — в базе данных; второе содержит сущности, являющиеся сегментами виртуальной памяти ОС.

Предполагается, что если субъект реализовал информационные потоки по памяти от всех сущностей из ПАС-множества другого субъекта к себе, то первый субъект получает право доступа владения ко второму субъекту. В соответствии с этим предположением вместо правила $know(x, y, z)$ вводится правило $know(x, y, Z)$.

3. Дополнительно к правам доступа, традиционно рассматриваемым в ДП-моделях, вводятся права доступа, специфичные для конкретной КС сетевого управления доступом. Примерами таких прав доступа являются $access_r$ — право доступа к сущности сетевой КС и $control_r$ — право конфигурирования сущностей сетевой КС. При добавлении значительного числа новых прав доступа выполняется агрегирование прав доступа — замена нескольких прав доступа одним с более высоким уровнем абстракции.

В некоторых сетевых КС, например в Cisco Secure Access Control Server, можно явно разрешить или запретить субъектам выполнение определенных команд в рамках сессии конфигурирования сущности-узла КС. Формальное описание данного механизма возможно с использованием элементов ролевого управления доступом совместно с агрегированием прав доступа.

4. Для формального описания элементов управления доступом на основе правил доступа и функций иерархии сущностей вводится множество учетных записей U, от имени которых субъекты реализуют доступ к сущностям КС, и множество векторов доступа V, описывающих права доступа некоторой учетной записи $u \in U$ с сущности-контейнера (узла) $c \in C$ к сущности $e \in E$ и определяемых следующим образом.

Пусть имеются: множество прав доступа R, учетная запись $u \in U$, сущность $e \in E$ и контейнеры $c_1, \ldots, c_n \in C$, такие, что $u < c_i$ для $i = 1, \ldots, n$. Пусть также есть функция

$f : U \times E \times C \to 2^R$, описывающая права доступа учетной записи $u \in U$ к сущности $e \in E$ при инициализации сессии с контейнера $c \in C$. Тогда вектором доступа учетной записи u к сущности e будем называть набор пар $(c_1, f(u, e, c_1)), \ldots, (c_n, f(u, e, c_n))$.

5. В соответствии с этим определением к правилам преобразования состояний добавляется правило $create\_session\_remote(s, u, c, e)$, описывающее создание сессии удаленного доступа субъектом $s \in S$ с правами доступа учетной записи $u \in U$ к сущности $e \in E$ и назначение субъекту s в рамках этой сессии прав доступа учетной записи u в зависимости от ребра (u, e, v) в графе доступа и контейнера c, с которого порождена сессия субъектом s.

С использованим этих расширений языка ДП-моделей базовая СУД ДП-модель может быть разработана по стандартной схеме построения ДП-моделей в [1].

ЛИТЕРАТУРА

1. *Девянин П. Н.* Модели безопасности компьютерных систем. Управление доступом и информационными потоками. Учеб. пособие для вузов. М.: Горячая линия-Телеком, 2011.

УДК 004.75

МОДЕЛЬ БЕЗОПАСНОСТИ КРОСС-ПЛАТФОРМЕННЫХ ВЕБ-СЕРВИСОВ ПОДДЕРЖКИ МУНИЦИПАЛЬНЫХ ЗАКУПОК

Д. Д. Кононов, С. В. Исаев

Целью работы является создание на основе RBAC [1] модели безопасности, учитывающей специфику веб-приложений. На базе этой модели разработаны программные средства для решения задач муниципального заказа администрации г. Красноярска, в том числе для проведения открытых электронных аукционов. Юридическая значимость действий пользователей обеспечивается использованием электронной цифровой подписи. Работа выполняется в рамках развития Автоматизированной системы проведения муниципальных заказов (АСП МЗ). Определяющими документами являются федеральные законы №1, 94 и 149.

В результате анализа предметной области была выбрана модель RBAC. В модель были внесены изменения, учитывающие особенности веб-приложений. К существующим понятиям «субъект» (subject), «роль» (role), «разрешение» (permission), «сессия» (session) были добавлены «токен» (token) и «запрос» (request).

Определение 1. *Токен* — набор атрибутов субъекта, позволяющих осуществить его аутентификацию в системе. Токеном является пара (имя, пароль) либо пара (сертификат ЭЦП, закрытый ключ ЭЦП).

Определение 2. *Запрос* — набор информации, пересылаемой клиентом серверу по протоколу HTTP. Запрос содержит набор заголовков, уникальный идентификатор ресурса, набор параметров имя/значение и тело запроса.

Запрос принадлежит сессии, в рамках одной сессии может выполняться несколько запросов. Понятия «запрос» и «разрешение» связаны отношением «многие-ко-многим». На множестве запросов вводится отношение включения.

Определение 3. Запрос A *включает* запрос B, если путь уникального идентификатора ресурса запроса A содержит путь уникального идентификатора ресурса запроса B с начальной позиции строки.

Например, запрос «/library/book» включает запрос «/library». Полученная модель отражает предметную область и позволяет эффективно разграничивать доступ в веб-системах.

В работе D. F. Ferraiolo et al. [2] рассматриваются вопросы применения модели RBAC для веб-приложений. В первом случае предлагается хранить идентифицирующую и служебную информацию (в том числе пароль), подписанную сервером, на стороне клиента в виде `cookies`. Данный метод не учитывает истечения срока действия и возможности компрометации сертификата сервера. Во втором случае используются атрибуты сертификата X.509 для хранения политик безопасности. Описанный подход имеет следующие недостатки. Во-первых, жестко указаны роли и разрешения, что не позволяет добавлять новые полномочия для субъекта без замены сертификата. Во-вторых, не учитывается ситуация, когда система работает с сертификатами разных удостоверяющих центров, имеющих собственные политики безопасности.

В настоящей работе предлагается комбинированный подход. Предварительная аутентификация производится с помощью имени пользователя и пароля, что дает субъекту доступ на чтение к закрытой части системы. Дальнейшие действия субъекта по добавлению, модификации и удалению информации подтверждаются электронной цифровой подписью, что обеспечивает юридическую значимость и аутентичность информации. На стороне клиента в `cookies` хранится только идентификатор сессии, который становится недействительным после ее завершения. Сертификат ЭЦП содержит только стандартные атрибуты X.509. Информация о политиках безопасности хранится на сервере системы и доступна для редактирования администратором. Данный подход позволяет использовать произвольный удостоверяющий центр для выдачи сертификатов пользователям и гибко настраивать права доступа.

Особое внимание уделялось возможности кросс-платформенного функционирования системы. В настоящий момент система может функционировать на платформах Windows, Linux, FreeBSD. В работе используется криптопровайдер КриптоПро CSP, который обеспечивает работу стандартов ГОСТ Р 34.10-94, ГОСТ Р 34.11-94, ГОСТ Р 34.10-2001, ГОСТ 28147-89.

Реализованные веб-сервисы обеспечивают защиту публикуемых данных, простоту и удобство работы специалистов муниципального заказа, оперативность размещения информации на сайте, а также защиту от несанкционированного использования. Система успешно эксплуатируется более четырех лет. За период 2009–2010 гг. с использованием системы было проведено 354 электронных аукциона на общую сумму 238 660 934 руб., экономия бюджета составила 42 443 682 руб. Имеются свидетельства о регистрации программного обеспечения [3, 4].

ЛИТЕРАТУРА

1. *Sandhu R., Coyne E. J., Feinstein H. L., and Youman C. E.* Role-Based Access Control Models // IEEE Computer (IEEE Press). 1996. V. 29. No. 2. P. 38–47.
2. *Ferraiolo D. F., Kuhn D. R., and Chandramouli R.* Role-Based Access Control. Norwood, USA: Artech House, 2003.
3. *Ноженкова Л. Ф., Исаев С. В., Кононов Д. Д. и др.* Система проведения электронных интернет-аукционов // Свидетельство об официальной регистрации в Реестре программ для ЭВМ № 2009612095 от 24.04.2009. (Федеральная служба по интеллектуальной собственности, патентам и товарным знакам). 2009.
4. *Ноженкова Л. Ф., Исаев С. В., Кононов Д. Д., Исаева О. С.* Интернет-система поддержки муниципального заказа // Свидетельство об официальной регистрации в Реестре про-

грамм для ЭВМ № 2009612093 от 24.04.2009. (Федеральная служба по интеллектуальной собственности, патентам и товарным знакам). 2009.

УДК 004.056

РАЗРАБОТКА КОМПЛЕКСНОЙ ОБУЧАЕМОЙ СИСТЕМЫ ЗАЩИТЫ ИНФОРМАЦИОННЫХ РЕСУРСОВ ОТ ФИШИНГОВЫХ АТАК[1]

А. В. Милошенко, Т. М. Соловьёв, Р. И. Черняк, М. В. Шумская

В настоящее время многие услуги стали доступны через Интернет. Финансовый сектор также не стал исключением. Появились различные платежные системы, интернет-кошельки и терминалы оплаты. Безопасность платежей внутри данных систем обеспечивается множеством высокотехнологичных решений, таких, как сертификаты безопасности, криптографические протоколы и др. Однако все эти решения оказываются неэффективными при применении злоумышленниками методов социальной инженерии, использующих слабости человеческого фактора.

Одним из наиболее распространенных видов такого рода атак сегодня является фишинг [1]. Фишинг (от англ. *phishing*, *password fishing* — выуживание паролей) — это вид сетевого мошенничества, целью которого является получение доступа к конфиденциальным данным пользователей обманным путём. Популярной методикой фишинга является создание поддельных веб-сайтов, внешне неотличимых от подлинных. Ущерб от преступлений, связанных с фишингом, только за 2010 г. исчисляется миллиардами долларов США. При этом, согласно статистике, количество фишинговых атак с каждым годом увеличивается примерно в полтора раза.

Остановить бурное развитие такого рода преступлений можно посредством создания комплексных систем защиты от фишинговых атак. Поскольку арсенал фишеров растёт стремительными темпами, необходимо обеспечивать обучаемость таких систем. В настоящее время не существует подобного рода решений, о чем красноречиво повествует статистика, поэтому создание комплексной обучаемой высокоэффективной системы защиты от фишинговых атак является интересным и актуальным направлением. Создание такой системы предполагает предварительное изучение характерных признаков фишинговых ресурсов и разработку на их основе методов оценивания степени опасности информационного ресурса и определения потенциально опасных ресурсов. Этому, в основном, и посвящена данная работа. На основе полученных результатов предложена схема функционирования системы защиты от фишинговых атак.

Характерные признаки фишинговых ресурсов

Анализ существующих фишинговых ресурсов позволил составить перечень их характерных признаков.

1. *Сходство графического контента*

Основная задача злоумышленников при проведении фишинговой атаки — заставить пользователя поверить в аутентичность фишингового ресурса. Наиболее простым способом сделать это является заимствование графического оформления у атакуемого сайта.

Несмотря на большое количество исследований, задача определения степени сходства изображений в настоящее время не имеет универсального и эффективного решения. Объясняется это в первую очередь тем, что понятие похожести двух изображе-

[1]Работа выполнена в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009–2013 гг. (гос. контракт № П1010).

ний неразрывно связано с особенностями человеческого восприятия и вследствие этого трудно формализуемо. Уверенно можно говорить лишь о факте полной идентичности изображений, которая определяется попиксельным сравнением или сравнением на основе значений некоторой хэш-функции.

Для определения похожести изображений, подвергаемых незначительной модификации, возможно использование методов, основанных на сравнении усредненных характеристик изображения, например метода поблочного анализа цвета или метода сравнения гистограмм.

Более сложный метод сравнения изображений — метод SURF [2]. Он базируется на сравнении ключевых точек, инвариантных как к геометрическим и фотометрическим преобразованиям, так и к изменению масштаба. Нахождение ключевых точек осуществляется с помощью прохода по пикселям изображения и поиска максимума гессиана — определителя матрицы, составленной из вторых частных производных (матрицы Гессе) функции яркости изображения. Реализация данного метода содержится во многих открытых библиотеках алгоритмов, например в библиотеке openCV.

2. *Сходство текстового контента*

Понятие похожести текстов также является нечетким, и, как и в предыдущем случае, с уверенностью можно заявлять лишь об их полной идентичности. Однако на практике при изменении текстового контента оригинального ресурса злоумышленники, как правило, используют стандартный набор преобразований, таких, как:

— вставка наборов случайных символов;
— произвольная вставка и удаление пробелов;
— замена символов одной кодировки на похожие по написанию символы другой кодировки;
— вставка ключевых слов на случайные позиции.

В большинстве случаев производится сравнение не самих исходных текстов, а построенных на их основе подмножеств или отпечатков этих подмножеств — значений некоторых хэш-функций. В зависимости от способов построения отпечатков методы определения похожести текстов можно разделить на два класса — синтаксические и лексические. В первом случае анализируются построенные по определенным правилам последовательности слов из текста, во втором — строятся словари ключевых слов. С точки зрения эффективности и простоты реализации оптимальными являются два метода определения близости текстового контента: метод определения индекса повторяемости и метод поиска длинных предложений [3].

Сложность выявления сходства контента растёт пропорционально числу защищаемых ресурсов, что серьёзно сказывается на производительности антифишинговой системы. Проверка признаков, перечисляемых далее, не требует сравнения с оригиналом.

3. *Наличие ресурса в фишинговых базах*

Интернет-сообществом поддерживается большое количество баз опасных ресурсов. Одни пользователи добавляют ссылки на такие ресурсы, другие — подтверждают или опровергают их опасность. Окончательное решение принимается администрацией конкретной базы. Как правило, фишинговые ресурсы попадают в такие списки в течение нескольких суток. Данный признак является очень сильным, однако сама концепция ведения списка подразумевает постоянную его актуализацию, которая всегда отстаёт от деятельности злоумышленников.

4. *Использование особенностей формата URL*

Формат URL имеет много параметров, часть из которых крайне редко применяется на практике. Зачастую злоумышленники используют более полную форму, добавляя в неё редко используемые параметры, чтобы ввести пользователя в заблуждение и убедить его в подлинности фишингового ресурса.

5. *Подозрительные регистрационные данные ресурса*

К регистрационным данным можно отнести географическое положение, дату регистрации домена, имя собственника сайта или название организации-владельца. Как правило, фишинговые сайты активны в первые пять дней после их создания. В связи с этим большое значение имеет дата регистрации домена. Часто фишинговые ресурсы регистрируются в стране, отличной от той, в которой расположен оригинальный сайт, поэтому необходимо также отслеживать соответствие домена верхнего уровня реальному местонахождению сайта.

6. *Наличие ресурса на одном IP-адресе с выявленными ранее фишинговыми ресурсами*

Расположение нескольких ресурсов на одном IP-адресе является достаточно распространенной ситуацией, поэтому целесообразно использование списков IP-адресов, на которых были замечены фишинговые ресурсы. Таким образом можно идентифицировать ресурс как потенциально опасный, если он расположен на одном IP с множеством фишинговых ресурсов.

7. *Наличие изображений, содержащих в себе текст в графическом представлении*

Данный способ применяется злоумышленниками для усложнения идентификации опасного ресурса. Пользователь будет воспринимать такой объект как обычный текст, а автоматизированные антифишинговые системы — как изображения. Если контент изображений дополнительно не анализируется, система может принять неправильное решение о степени опасности ресурса.

8. *Использование «неоправданно большого» количества скриптов*

Как правило, объём исполняемого кода на странице растёт пропорционально её информативности и предоставляемой функциональности. «Неоправданно большое» количество скриптов может являться признаком недокументированных возможностей. Для определения максимально допустимого количества скриптов предлагается использовать статистистические данные из выборки в конкретной предметной области.

Все выделенные признаки имеют разную значимость, которая может варьироваться в зависимости от их комбинаций. Это говорит о необходимости разработки механизмов совокупного анализа признаков — механизмов принятия решения о степени опасности ресурса.

Методы оценивания степени опасности ресурса

Далее предлагаются алгоритмы, определяющие степень опасности информационных ресурсов с учетом описанных признаков. На вход каждого алгоритма поступает информационный ресурс, выходом является рациональное число из отрезка $[0, 1]$, которое характеризует степень опасности ресурса; наибольшей опасности соответствует единица, наименьшей — ноль.

1. *Последовательное использование алгоритмов оценки признаков*

Выполняется непосредственная проверка всех признаков. Поскольку различные алгоритмы анализа данных используют признаки разной значимости, на вход объемлющего алгоритма необходимо подать также *таблицу значимости признаков*.

Метод работает следующим образом. Если ресурс удовлетворяет некоторому признаку, то результат увеличивается на значимость этого признака. По завершении работы алгоритма результат нормируется, с тем чтобы он содержался в отрезке $[0, 1]$.

Основное достоинство данного подхода — простота реализации; недостатки — необходимость проверки всех признаков и негибкий учет значимости каждого из них.

2. *Использование булевой функции*

При определении степени опасности информационного ресурса некоторый набор признаков может оказаться достаточным для отнесения ресурса к разряду опасных. Например, если доменное имя сайта содержится в фишинговых базах, нет необходимости проверять остальные признаки — такой сайт априори представляет опасность.

Для работы алгоритма необходимо задать булеву функцию $f : \{0, 1\}^n \to \{0, 1\}$, принимающую значение 1 на тех и только тех наборах значений признаков информационного ресурса, которые гарантируют опасность последнего. Выделение достаточных подмножеств признаков (определение функции f) относится к предвычислениям, а сам алгоритм состоит в ее однократном вычислении на наборе аргументов, соответствующем признакам проверяемого ресурса.

Недостатком алгоритма является высокая зависимость от таблицы значимости признаков. Кроме того, процесс определения достаточных условий не автоматизирован, поэтому на данном этапе возможны ошибки разного рода.

3. *Использование нейронных сетей*

Одним из способов отказа от таблицы значимости признаков является использование статистических методов. Пусть значение функции $f : [0, 1]^n \to \{0, 1\}$ характеризует опасность (1) или безопасность (0) ресурса. Аргументы функции — вещественные переменные со значениями в отрезке $[0, 1]$, определенными алгоритмами оценивания признаков.

Функция f неизвестна, и вычислить ее непосредственно невозможно. Поведение функции может быть промоделировано с помощью нейронной сети. Вычисление функции f происходит в два этапа:

1) обучение нейронной сети;
2) вычисление значения функции.

При этом обучение может производиться параллельно, не оказывая существенного влияния на производительность самого алгоритма. В качестве обучающей выборки могут быть использованы базы фишинговых ресурсов.

Основными преимуществами данного способа являются гибкость при оценивании значимости признаков и высокая адекватность действительности. Последнее достигается путем использования в алгоритме статистических методов.

4. *Использование методов линейной регрессии*

Обобщим предыдущую модель на непрерывный случай, полагая, что степень опасности ресурса оценивается как вещественное число в отрезке $[0, 1]$ и определяется функцией $f : [0, 1]^n \to [0, 1]$.

Одним из статистических способов вычисления неизвестной вещественнозначной функции f является метод линейной регрессии. Он позволяет построить функцию g — линейную аппроксимацию f в виде $g = \sum\limits_{i=1}^{n} w_i x_i$. Для того чтобы оценить степень опасности того или иного ресурса, достаточно вычислить значение функции g от соответствующих ему аргументов.

Для построения приближенной функции используются заданные значения опасностей некоторых информационных ресурсов. При неизвестной функции определения

степени опасности не вполне понятно, как получить достаточно большое количество значений этой функции на заданных значениях аргументов. Эта проблема решается с помощью экспертных оценок.

Механизмы выбора потенциально опасных ресурсов

Поскольку наибольший ущерб фишинговые ресурсы наносят в первые сутки своего существования, для обеспечения эффективной защиты от них необходимо проводить непрерывный мониторинг информационных ресурсов. Однако анализировать все ресурсы сети Интернет не представляется возможным, поэтому следует сконцентрироваться на тех, которые с наибольшей вероятностью могут оказаться опасными.

При разработке фишинговых сайтов злоумышленники ориентируются на то, что пользователь либо допустит ошибку при наборе доменного имени, либо перейдёт по ссылке, визуально похожей на доверенный ресурс. Согласно этому, были выделены несколько принципов построения доменных имён, представляющих потенциальную опасность по отношению к заданному.

1. *Доменные имена с незначительными изменениями*

Данный принцип включает в себя следующие методы генерации: опечатки (замена символа другим символом, расположенным на соседней клавише); выпадение гласной между согласными; приписывание к началу доменного имени комбинации «www»; вставка дефиса; вставка символа вследствие нажатия нужной клавиши и соседней; удвоение символов доменного имени; замена символов таким образом, что полученное имя схоже по произношению с исходным; добавление в конце доменного имени, написанного латиницей и оканчивающегося на гласную, символов «s», «r» или «rs»; перестановка букв.

2. *Транслитерация кириллического доменного имени*

Под транслитерацией домена понимается замена кириллических символов на один или несколько латинских символов. Можно выделить три принципа транслитерации: транскрипция (преобразование на основе схожести соответствующих звуков), преобразование типа «один символ — одна буква» и преобразование на основе графической схожести символов. При генерации доменных имён могут использоваться любые из принципов транслитерации, а также их комбинации.

3. *Перевод на английский язык лексем заданного доменного имени*

В данном случае сгенерированные доменные имена содержат лексемы, являющиеся либо переводом, либо транслитерацией (если невозможно осуществить перевод) лексем заданного домена.

4. *Доменные имена, полученные на основе использования сервисов поисковых систем по поиску URL*

В первые дни существования опасные ресурсы имеют высокую популярность, поэтому при использовании запросов, осуществляющих поиск заданных лексем в URL, такие ресурсы могут оказаться в числе первых, выданных поисковой системой.

5. *Доменные имена, содержащие комбинации лексем*

Принцип генерации доменов, содержащих комбинации лексем, включает в себя следующие варианты написания: слитное написание, использование дефиса, а также их комбинации.

Для последующего мониторинга с целью определения наличия ресурса и оценки степени его опасности каждое сгенерированное имя должно быть расширено каждым возможным для него постфиксом, идентифицирующим доменную зону. Кроме этого, предлагается сгенерировать группу доменных имён, используя уже расширенные домены. Из каждого доменного имени вида *доменное_имя.доменная_зона1* генериру-

ются всевозможные имена вида *доменное_имя|доменная_зона2.доменная_зона1*, где операция | — операция конкатенации, *доменная зона2* принимает все значения из списка возможных доменных зон для доменного имени *доменное_имя*. Такая мера целесообразна, поскольку при разработке фишинговых ресурсов злоумышленники часто используют «двойные» доменные зоны.

Система защиты от фишинговых атак

На основе полученных результатов может быть создана полноценная система защиты от фишинговых атак. Предлагается следующая схема функционирования такой системы защиты для каждого информационного ресурса:

1) генерация списка потенциально опасных доменных имен;
2) получение списка зарегистрированных и доступных потенциально опасных информационных ресурсов;
3) определение степени опасности каждого из потенциально опасных информационных ресурсов;
4) пополнение антифишинговых баз вновь обнаруженными опасными ресурсами.

Для обеспечения высокой степени защиты ресурса необходимо регулярно повторять последовательность действий 2 — 4 и проводить непрерывное обучение системы.

ЛИТЕРАТУРА

1. *Lininger R. and Vines D.* Phishing: Cutting the Identity Theft Line. Wiley, 2005. 334 p.
2. *Bay H., Ess A., and Tuytelaars T.* SURF: Speeded Up Robust Features // Computer Vision and Image Understanding (CVIU). 2008. V. 110. No. 3. P. 346–359.
3. *Зеленков Ю. Г., Сегалович И. В.* Сравнительный анализ методов определения нечетких дубликатов для Web-документов // Труды 9 Всерос. науч. конф. «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» — RCDL'2007. Переславль-Залесский, Россия, 2007. С. 166–174.

УДК 004.94

ПОДХОДЫ К РАЗРАБОТКЕ ДИСКРЕЦИОННОЙ ДП-МОДЕЛИ СОВРЕМЕННЫХ ЗАЩИЩЕННЫХ ОПЕРАЦИОННЫХ СИСТЕМ[1]

В. Г. Проскурин

Рассматривается дискреционная ДП-модель, развивающая и конкретизирующая существующие ДП-модели для случая, когда моделируемая компьютерная система является защищенной операционной системой (ОС). Далее будем называть предлагаемую модель ЗОС ДП-моделью. Она основывается на базовой ДП-модели [1], ФПАС ДП-модели [2] и ФС ДП-модели [3], а также отчасти на модели мандатной политики целостности информации Биба [1]. Будем использовать систему обозначений [1].

Предлагаемая модель характеризуется следующими основными отличиями от ранее разработанных ДП-моделей.

1. В модель вводится новый элемент — множество учетных записей пользователей U. Права доступа к сущностям назначаются не напрямую субъектам, как в других ДП-моделях, а учетным записям пользователей, которые делегируют права доступа субъектам, выполняющимся от имени соответствующей учетной записи. Другими словами, множество разрешенных прав доступа R определяется как подмножество множества $U \times (E \cup U) \times R_r$. Задается отображение $user : S \to U$, определяющее для каждого

[1]Работа выполнена при поддержке Министерства образования и науки Российской Федерации.

субъекта, от имени какой учетной записи он выполняется. Для каждой учетной записи $u \in U$ задается множество параметрически ассоциированных с ней сущностей $]u[$, реализация от каждой из которых информационного потока по памяти к субъекту позволяет породить нового субъекта от имени учетной записи u. При этом сущности, параметрически ассоциированные с учетной записью, не обязательно являются параметрически ассоциированными с субъектом, порожденным от имени данной учетной записи. По аналогии с множеством доверенных субъектов L_S среди учетных записей выделяется подмножество доверенных учетных записей L_U.

2. В дополнение к правам доступа $read_r$, $write_r$, $execute_r$, own_r, традиционно рассматриваемым в дискреционных ДП-моделях, вводится право доступа $grant_r$, позволяющее формализовать ситуацию, когда субъект-владелец предоставляет к сущности ограниченный доступ субъектам, выполняющимся от имени других учетных записей.

3. В модель вводятся новые элементы: множество видов совместного доступа $Share = \{read_a, write_a\} \subset R_a$ и две функции — $sa : E \setminus S \to 2^{Share}$, задающая текущие доступы к сущностям, не являющимся субъектами, и $sm : E \setminus S \to 2^{Share}$, задающая разрешенные совместные доступы к сущностям, не являющимся субъектами. При формальном описании доступа субъекта к объекту учитывается совместный доступ, т. е. множество доступов A определяется как подмножество множества $S \times E \times R_a \times 2^{Share}$. Перечисленные новые элементы модели позволяют формализовать ситуацию, когда субъект s_1, имеющий право $read_r$ или $write_r$ на доступ к некоторой сущности e, не может реально осуществить доступ $read_a$ или соответственно $write_a$, поскольку другой субъект s_2 уже осуществляет к сущности e доступ, не допускающий совместного использования данной сущности более чем одним субъектом. Рассматриваются совместные доступы $read_a$ и $write_a$, по определению устанавливается, что доступы $read_a$ совместимы между собой, но несовместимы с доступами $write_a$, а доступ $write_a$ к сущности может осуществляться субъектом только в монопольном режиме. Данное допущение описывает наиболее типичный сценарий организации совместного доступа к объектам; другие сценарии (например, комбинация параметров $GENERIC\_READ$ и $FILE\_SHARE\_WRITE$ в терминах $Windows$) реализуются крайне редко, в основном при обращениях к файлам баз данных, формализация доступа к которым далеко выходит за рамки предлагаемой модели.

4. Для каждого права доступа $r \in R$ к сущности $e \in E$ вводится множество (возможно, пустое) сущностей-параметров $](e, r)[$, не являющихся субъектами. Если для некоторой конкретной пары (e, r) множество $](e, r)[$ непусто, то для реализации доступа к сущности субъекту необходимо реализовать к себе информационные потоки по памяти от всех сущностей из $](e, r)[$. Введение в ДП-модель сущностей-параметров позволяет формализовать непрямые доступы субъектов к сущностям, требующие временной смены эффективного идентификатора учетной записи, от имени которой выполняется субъект (*su*, *sudo*, $SUID/SGID$ в $UNIX$, олицетворение учетной записи в $Windows$ и т. п.), либо других, более сложных взаимодействий субъекта со специальными сущностями ОС (UAC в Windows и т. п.).

5. В модель вводятся новые элементы, позволяющие формально описать систему мандатного контроля целостности (MIC), входящую в ОС семейства $Windows$ начиная с версии $WindowsVista$. К этим элементам относятся линейная шкала $(LI, \leqslant)$ поддерживаемых уровней целостности, функция $i_u : U \to LI$, устанавливающая максимально допустимый уровень целостности для субъектов, выполняющихся от имени заданной учетной записи, функция $i_s : S \to LI$, задающая текущий уровень целостности субъекта, и функция $i_e : E \setminus S \to LI$, задающая уровень целостности для сущ-

ностей, не являющихся субъектами. Без ограничения общности можно рассматривать случай, когда в моделируемой системе заданы только два уровня мандатной целостности — низкий и высокий.

6. В правила преобразования состояний ЗОС ДП-модели добавлены два новых правила: $create\_first\_subject$, формализующее порождение первого субъекта в сеансе работы пользователя с моделируемой системой, и $delete\_access$, формализующее прекращение доступа субъекта к сущности и обусловленное этим снятие ограничений по совместному доступу к данной сущности.

Предлагаемая ЗОС ДП-модель позволяет применять научный аппарат ДП-моделей к задачам формального описания и научного обоснования различных практических решений, реализуемых в современных защищенных ОС. В частности, предполагается использовать ЗОС ДП-модель в рамках работ по повышению защищенности отечественных ОС.

ЛИТЕРАТУРА

1. *Девянин П. Н.* Модели безопасности компьютерных систем. Управление доступом и информационными потоками. Учеб. пособие для вузов. М.: Горячая линия-Телеком, 2011. 320 с.
2. *Колегов П. Н.* ДП-модель компьютерной системы с функционально и параметрически ассоциированными с субъектами сущностями // Вестник Сибирского государственного аэрокосмического университета им. акад. М. Ф. Решетнева. 2009. Вып. 1(22). Ч. 1. С. 49–54.
3. *Буренин П. Н.* Подходы к построению ДП-модели файловых систем // Прикладная дискретная математика. 2009. № 1(3). С. 93–112.