Гранулярный контроль безопасности поведения приложений со стороны ядра Linux

Гамаюнов Д. Ю., Сапожников А. В., Сахаров Ф. В.
Лаборатория вычислительных комплексов
факультет ВМК МГУ имени М. В. Ломоносова
{gamajun, askold, sakharov}@lvk.cs.msu.su

В работе рассматривается задача автоматического контроля безопасного выполнения приложений под управлением операционной системы Linux с целью раннего обнаружения атак, изменяющих поток выполнения программы или поток данных. В качестве решения данной задачи предлагается комбинированный подход с использованием Security Enhanced Linux (SELinux) и механизма отслеживания контрольных точек в исполняемом коде приложений.

Контроль безопасного поведения приложений на узлах в настоящее время является одной из наиболее актуальных задач комплексной защиты информации в сетях. Программные уязвимости в сетевых приложениях – это один из распространённых путей проникновения вредоносного программного обеспечения в сети организаций. Для ядра Linux существуют проекты механизмов «ядерного» уровня, предназначенных для контроля поведения приложений на основе профилей нормального поведения. Примерами таких проектов являются SELinux [1-3] и AppArmor [4, 5]. Они реализованы как подсистемы ядра операционной системы (ОС) и используют профили (политики) нормального поведения приложений, в которых перечислены все разрешённые для данного приложения операции и группы ресурсов ОС.

SELinux и AppArmor рассматривают контролируемые приложения как «чёрный ящик», то есть не различают их внутренние состояния. В то же время существует большой класс приложений, для которых необходимый набор системных вызовов и доступных ресурсов различен на разных участках графа потока управления. Например, это все приложения, в которых есть этап авторизации пользователя – возможности неавторизованной сессии могут быть значительно уже, чем авторизованной. В результате профили нормального поведения реальных приложений оказываются либо избыточно «разрешительными», так что реальных ограничений на возможности нарушителя не накладывается, либо излишне «жёсткими», что нарушает нормальное функционирование приложений.

В работе [6] был предложен подход к обнаружению широкого класса атак на приложения на основе динамического сопоставления наблюдаемого поведения приложения с моделью его нормального поведения в виде альтернирующего автомата. В развитие данного подхода предлагается расширить механизм контроля поведения приложений SELinux за счёт введения меток состояний приложений. Основная цель предлагаемой модификации – повышение точности описания нормального поведения в SELinux для тех приложений, для которых можно явно выделить состояния – например, для сетевых сервисов. В этом случае понятие состояния приложения трактуется в соответствии со спецификацией сетевого протокола, который реализуется в контролируемом приложении.

Модификация предполагает переключение политик SELinux при изменении состояния приложения. Это позволит, например, разграничить возможности клиент-серверных приложений в состояниях «не авторизован» и «успешно авторизован». Под контрольной точкой подразумевается адрес в виртуальном адресном пространстве процесса. Если исполнение переходит на очередной контролируемый адрес, считаем, что приложение перешло в новое состояние.

Для динамической установки контрольных точек в приложениях используется система utrace [7] и uprobes [8]. Utrace — это система профилировки и отладки пользовательских приложений из пространства ядра Linux. Использование отладчика на стороне ядра позволяет

достичь высокой скорости обработки событий за счёт снижения накладных расходов на переключение контекстов и передачу данных из/в пользовательские приложения. Uprobes – это клиент utrace, который позволяет расставлять контрольные точки в пользовательском процессе. Поддерживаются два типа контрольных точек: uprobe — может быть установлена в любом месте виртуального адресного пространства процесса, uretprobe – уведомляет о выходе из определенной функции. Utrace работает примерно на два порядка быстрее gdb и strace.

Механизм принятия решений SELinux о разрешении или запрещении системных вызовов со стороны приложения использует переменные-идентификаторы (SID). Эти переменные идентифицируют политику, которая должна быть применена к данному объекту системы. Для изменения политики приложения в процессе его исполнения достаточно заменить SID на другой.

В рамках модификации системы SELinux синтаксис языка политик был расширен конструкциями, которые позволяют хранить в политике адреса контрольных точек и соответствующие изменения политики при переходе в новое состояние. Для контроля поведения приложений реализован модуль ядра, использующий систему utrace для наблюдения за ходом исполнения приложения и политики SELinux. При загрузке он запрашивает текущее состояние SELinux, информацию о состояниях приложений, привязывается к целевым процессам с целью отслеживания вызовов fork() и exec(). По параметрам вызова exec() модуль определяет, какое приложение будет запущено. В том случае, если в политике SELinux существует информация о наборе состояний данного приложения, модуль начинает наблюдение за контрольными точками.

Одна из сложностей использования предложенного подхода заключается в трудоёмкости правильной расстановки контрольных точек в ядре и построения политики переключения контекстов SELinux при проходе через них. В настоящее время ведётся разработка автоматизированного средства создания политик SELinux с расстановкой контрольных точек на основе анализа трасс выполнения приложений с примерами трасс нормального выполнения, а также трасс, соответствующих примерам атак.

Литература

1. Michael Wikberg. Secure computing: SELinux // [www] http://www.tml.tkk.fi/Publications/C/25/papers/Wikberg_final.pdf, 2007.
2. Runge, C. (2004), SELinux: A new approach to secure systems, Technical report, Red Hat, Inc.
3. Redhat SELinux Guide // [www] http://www.redhat.com/docs/manuals/ enterprise/ RHEL-4-Manual/selinux-guide
4. AppArmor Detail // [www] http://en.opensuse.org/AppArmor_Detail, 2006.
5. R. Spenneberg. Shutting out Intruders with AppArmor // [www] http://www.linux-magazine.com/w3/issue/69/Shutting_out_Intruders_with_AppArmor.pdf, 2007.
6. Гамаюнов Д.Ю. Обнаружение компьютерных атак на основе анализа поведения сетевых объектов // ВМиК МГУ, Москва, 2007.
7. Utrace. [WWW] http://people.redhat.com/roland/utrace/
8. Uprobes. [WWW] http://sourceware.org/systemtap/