АРХИТЕКТУРА, МОДЕЛИ И МЕТОДИКИ ФУНКЦИОНИРОВАНИЯ СИСТЕМЫ ПРОАКТИВНОГО МОНИТОРИНГА ВЫПОЛНЕНИЯ ПОЛИТИКИ БЕЗОПАСНОСТИ

В. С. Богданов, И. В. Котенко

Санкт-Петербургский институт информатики и автоматизации РАН

СПИИРАН, 14-я линия ВО, д. 39, Санкт-Петербург, 199178

<bogdanov@comsec.spb.ru>, <ivkote@comsec.spb.ru>

УДК 681.3

Богданов В. С., Котенко И. В. **Архитектура, модели и методики функционирования системы проактивного мониторинга выполнения политики безопасности** // Труды СПИИРАН. Вып. 3, т. 2. — СПб.: Наука, 2006.

Аннотация. *В статье рассмотрены вопросы, касающиеся проектирования и реализации системы проактивного мониторинга выполнения политики безопасности в компьютерных сетях. Предлагаемый подход к мониторингу основан на моделировании действий пользователя в исследуемой компьютерной сети. Описаны обобщенная архитектура системы проактивного мониторинга и методики ее функционирования, рассмотрены проблемы, связанные с работой в реальной компьютерной сети, и их возможные решения. Статья содержит описание реализации прототипа системы проактивного мониторинга и пример применения прототипа для тестирования политики безопасности компьютерной сети.* — Библ. 18 назв.

UDC 681.3

Bogdanov V. S., Kotenko I. V. **Architecture, models and techniques of functioning of the system of proactive monitoring of security policy** // SPIIRAS Proceedings. Issue 3, vol. 2. — SPb.: Nauka, 2006.

Abstract. *The paper considers issues applied to design and implementation of the system of proactive monitoring of security policy. The approach suggested is based on simulation of user actions in computer network under investigation. The generalized architecture of proactive monitoring system and its functioning techniques are described, problems applied to work in real network and their possible solutions are considered. The paper contains the description of prototype of proactive monitoring system and the example of the prototype using to test network security policy.* — Bibl. 18 items.

1. Введение

В настоящее время обеспечение информационной безопасности является одной из актуальных проблем, возникающих при использовании компьютерных систем различного назначения. Основой для организации процесса защиты информации является политика безопасности. Политика безопасности описывает множество правил функционирования системы, в том числе правил обеспечения конфиденциальности, целостности и доступности ресурсов системы.

Процесс разработки политики безопасности можно разделить на три этапа: проектирование, реализация и эксплуатация. На этапе проектирования разработчики или администраторы безопасности (сети) формулируют политику безопасности, исходя из потребностей пользователей проектируемой системы. На этапе реализации сформулированная политика безопасности воплощается в компьютерной системе. После этого наступает этап эксплуатации, на котором пользователи используют компьютерную систему по назначению, при этом они должны действовать в рамках реализованной политики безопасности.

На этапе эксплуатации компьютерной системы перед администраторами сети остро стоит задача проверки соответствия политики безопасности, сформулированной на этапе проектирования, ее реализации в реальной системе, а также анализа адекватности этой политики целям обеспечения защиты информационных ресурсов компьютерной системы от текущих угроз безопасности [1].

Существующий подход к контролю выполнения политики безопасности в компьютерных сетях предполагает, что правила политики на этапе реализации трансформируются в настройки программного и аппаратного обеспечения компьютерной сети (конфигурационные файлы отдельных программ и устройств, общие хранилища информации в виде системных реестров, списки правил фильтрации и др.).

Для проверки выполнения политики на этапе эксплуатации, как правило, используется так называемый пассивный подход к мониторингу. В соответствии с данным подходом производится периодическое сравнение текущей конфигурации системы (на основе исследования настроек программного и аппаратного обеспечения) с той конфигурацией, которая была получена на этапе реализации. Найденные отличия свидетельствуют о том, что политика не выполняется. Однако такой подход, хотя и является основным, не может дать гарантии строгого выполнения предписанной политики. В сложной распределенной системе для выполнения запросов пользователей требуется использование множества таких настроек. Не все из них могут находится под контролем. Злоумышленники обладают возможностью обхода заданных параметров или их динамической замены посредством встраивания и использования специальных программ и т.п. В результате настройки программного и аппаратного обеспечения компьютерной сети могут не соответствовать реально выполняемой политике безопасности.

Подход, предлагаемый в данной работе, схож с подходом, который используется при активном анализе уязвимостей [2]. При таком анализе исследователь пытается имитировать действия злоумышленника для компрометации различных ресурсов компьютерной сети, а затем устраняет обнаруженные уязвимости.

Предлагаемая система проактивного мониторинга (СПМ) выполнения политики безопасности действует подобным образом. СПМ использует все доступные знания о системе и текущей политике для генерации действий пользователя, которые способны опровергнуть или подтвердить выполнение политики безопасности. СПМ выполняет данные действия и фиксирует их результаты. Совокупность результатов этих действий позволяет представить картину выполнения политики безопасности в системе.

В данной работе предложена распределенная архитектура системы проактивного мониторинга выполнения политики безопасности. Описана обобщенная методика функционирования и использования системы. Рассмотрен пример тестовой сети, примеры политик и трасс функционирования системы. Описаны архитектура и реализация прототипа СПМ. Работа содержит описание экспериментов с прототипом и анализ результатов этих экспериментов.

Работа состоит из следующих разделов. Второй раздел содержит обзор некоторых релевантных работ, посвященных мониторингу политик. Третий раздел описывает обобщенную архитектуру распределенной системы проактивного мониторинга политики безопасности. В четвертом разделе представлена обобщенная методика функционирования СПМ. В пятом разделе описана структура тестовой сети и приведен пример функционирования системы при

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

проверке политики доступа к FTP серверу. Шестой раздел содержит описание реализации прототипа системы. Седьмой раздел содержит заключение и основные выводы.

2. Релевантные работы

Существующие работы по политикам безопасности, в частности мониторингу политик, формируют необходимый начальный базис для настоящей работы. Представленный в данном разделе перечень работ не претендует на полноту, но отражает некоторые весьма существенные положения и результаты, полученные в исследованиях по мониторингу политик безопасности.

В [3] утверждается, что одно из необходимых условий безопасности системы — мониторинг действий пользователей и проверка их соответствия политике безопасности.

Статья [4] касается вопросов мониторинга событий в системе и создания политик реагирования на эти события.

В работах [5–9] также отмечается необходимость мониторинга выполнения политики безопасности.

Работа [10] предполагает использование мониторинга для перехода на усиленную политику в случае нарушения текущей.

В [11] активные попытки нарушить политику безопасности рассматриваются, как возможность сформировать политику безопасности для уязвимого приложения.

Подход к практивному мониторингу был использован [12] для тестирования протокола IPSEC.

Статья [13] описывает способы детализации политик мониторинга событий в системе. В [14] используется проверка конфигурации сети для подтверждения соответствия политике.

Статья [15] описывает систему мониторинга политики, реагирующую на внешние события. Система проверяет корректность изменений конфигурации заданной политикой.

В [16] описывается формальный способ доказательства выполнения коллективной политики, если известно, что политика выполняется для членов коллектива.

Статья [17] описывает связь между правилами политики со сценариями поведения пользователя в системе и целями, достигаемыми пользователем.

Подход к мониторингу политики безопасности, предлагаемый в настоящей работе, базируется на активной имитации действий пользователей (как разрешенных, так и запрещенных политикой безопасности) и определении расхождений реакций системы от предписанных. В существующих работах подобный подход не был исследован в достаточной мере. Пожалуй, наиболее близким к нему являются исслендования по тестированию протокола IPSEC в работе [12].

3. Архитектура системы проактивного мониторинга

Система проактивного мониторинга выполнения политики безопасности (СПМ) может рассматриваться как самостоятельная система или входить в состав «старшей системы», например, системы мониторинга выполнения политик безопасности или системы управления политиками безопасности.

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

На рис. 1 дано высокоуровневое представление СПМ. *Целью* проактивного мониторинга выполнения политики безопасности является выявление и оценка отклонений от заданной политики безопасности и конфигурации в тестируемой системе. *Входными данными* для СПМ являются спецификация тестируемой системы, спецификация проверяемых политик безопасности и параметры тестирования. *Выходными данными* для СПМ являются отчеты о выявленных в тестируемой системе отклонениях от заданной политики безопасности.

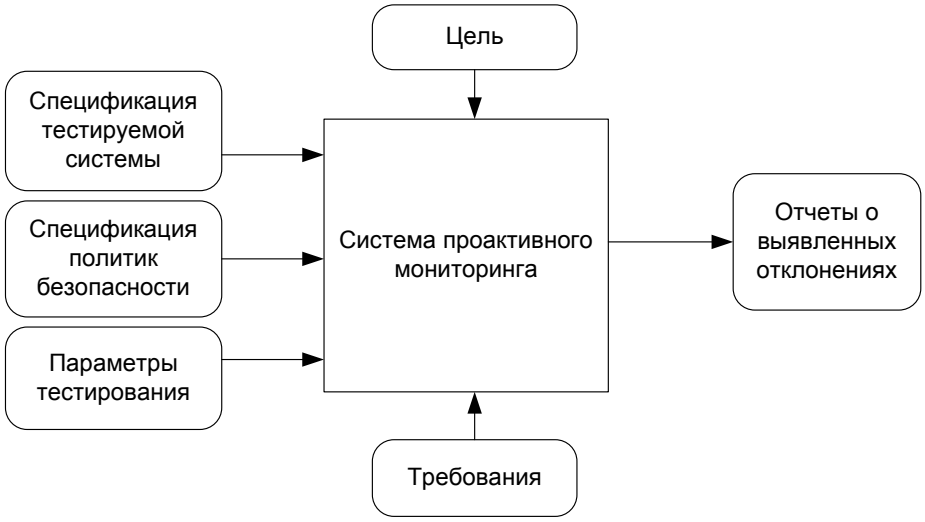

Рис. 1. Высокоуровневое представление СПМ.

СПМ должна удовлетворять следующим *требованиям*: (1) полнота охвата тестируемой системы — СПМ должна выявлять отклонения от заданной политики безопасности в любой из частей заданной системы; (2) полнота охвата политик безопасности — СПМ должна выявлять отклонения от любой части заданной политики безопасности; (3) адекватность оценки отклонений — СПМ должна адекватно оценивать отклонения от заданной политики безопасности в заданной системе; (4) производительность работы — СПМ должна работать с необходимой производительностью; (5) точность выявления отклонений — СПМ должна работать с достаточной точностью; (6) адекватность модели тестируемой системы — СПМ должна быть способна моделировать достаточно сложные системы.

Обобщенная архитектура СПМ изображена на рис. 2. СПМ состоит из четырех основных компонентов: конфигуратора, сканера, сборщика информации и пользовательского интерфейса.

Конфигуратор — это компонент, который предназначен для планирования и формирования комплекса сценариев для проведения мониторинга политик. Он получает на вход спецификацию тестируемой системы и спецификацию проверяемых политик безопасности. На основе этих данных конфигуратор формирует сценарии для сканеров, находящихся в системе. Сценарии достаточно просты для выполнения сканерами без проведения дополнительного планирования. Данный подход имеет следующие достоинства: становится возможным планировать проверки, проводимые совместно несколькими сканерами; самая трудоемкая часть работы выполняется одним компонентом централизованно, сканеры представляют собой «легкие» компоненты с простой логикой исполнения; уменьшается поток информации, передающейся по сети, так

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

как основная работа с базой данных действий пользователя совершается одним компонентом (конфигуратором), а не несколькими.

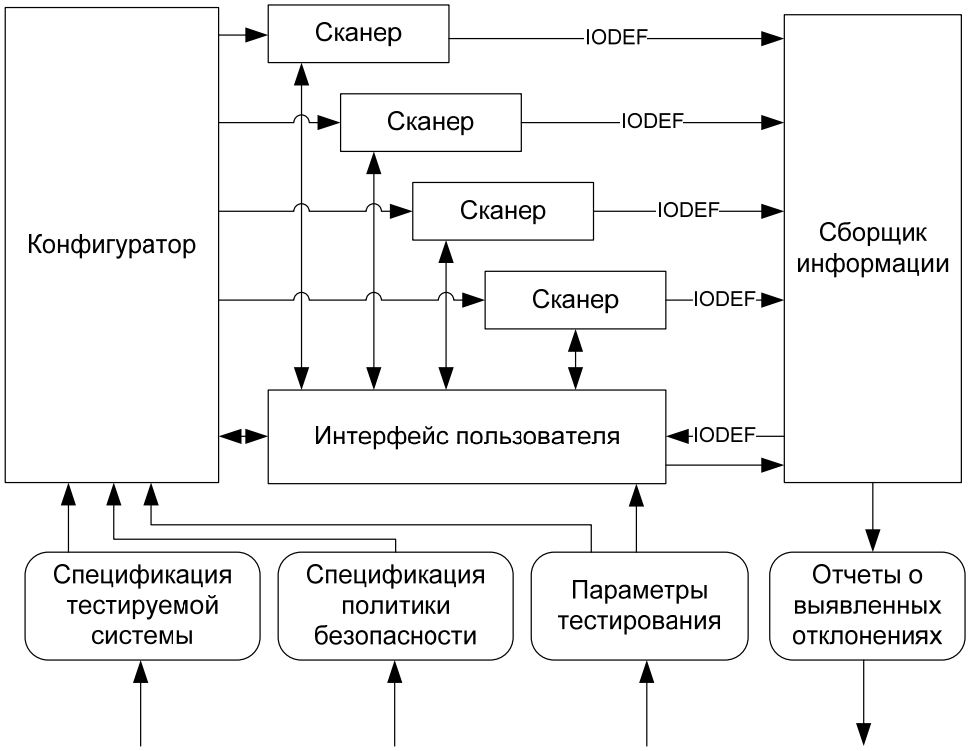

Рис. 2. Обобщенная архитектура системы.

Сканер — компонент, проверяющий определенную конфигуратором часть правил политики безопасности в заданном фрагменте тестируемой системы. Задание сканеру передается конфигуратором в виде сценария. Выполняя данный сценарий, сканер проводит проверку политики. Результаты проверки сканер отсылает сборщику информации.

Сборщик информации получает отчеты о результатах проведенных проверок от сканеров, анализирует полученную информацию и формирует сводные отчеты, которые выдаются пользователю (или «старшей системе»).

Интерфейс пользователя позволяет пользователю управлять работой всех компонентов системы, задавать входные данные конфигуратору, просматривать отчеты сборщика информации.

Для упрощения конфигурирования и передачи информации между компонентами конфигуратор, сборщик информации и интерфейс пользователя объединены в одном исполняемом модуле.

На рис. 3 показан пример распределения компонентов системы по фрагментам сети. Для достижения наилучших результатов, как минимум, один сканер должен присутствовать в каждом сегменте сети. Помимо этого необходимы сканер, установленный на рабочей станции, сканер, находящейся за пределами сети, и сканер на рабочей станции, которая подключается к анализируемой сети с помощью модема. Если используется беспроводная сеть, нужен также сканер, находящийся на хосте беспроводной сети

Информация от сканера к сборщику информации и от сборщика информации к интерфейсу пользователя передается в формате IODEF [18]. Вся осталь-

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ная информация внутри системы передается во внутреннем формате, основанном на XML.

Общая архитектура сканера нарушений политик безопасности изображена на рис. 4.

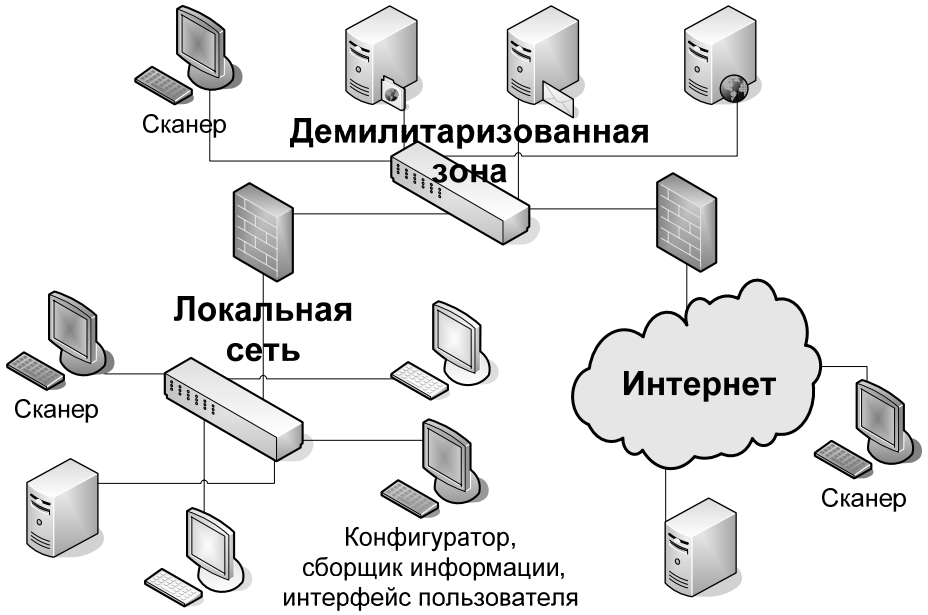

Рис. 3. Распределение компонентов системы по фрагментам сети.

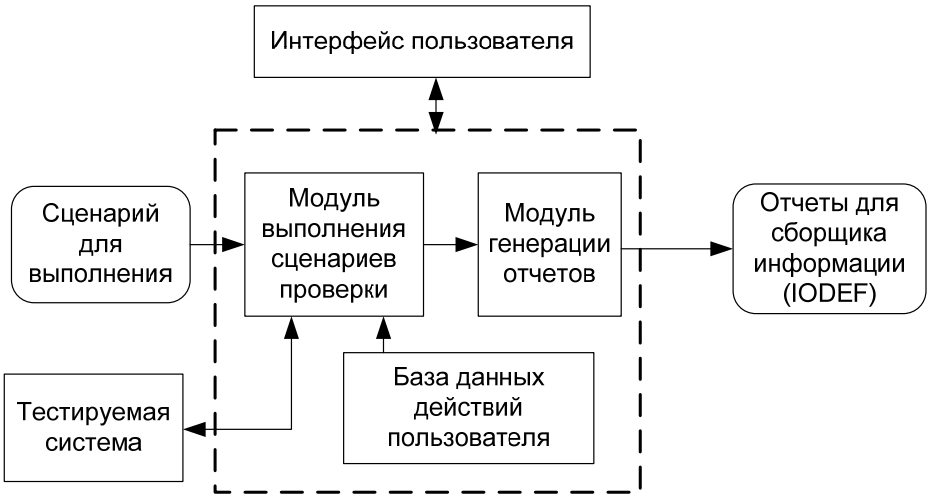

Рис. 4. Архитектура сканера.

Сценарий для выполнения представляет собой последовательность действий на формальном языке, описывающую операции производимые сканером, их параметры и последовательность. Модуль выполнения сценариев проверки, выполняет сценарии, переданные сканеру конфигуратором, производя операции над активами тестируемой системы. База данных действий пользователя содержит описание всех возможных операций пользователя над активами и описание параметров этих операций. Модуль генерации отчетов предназначен для обработки полученной в результате выполнения сценариев информации и представления ее в удобном для пользователя виде. Интерфейс пользователя позволяет управлять работой сканера, в том числе вводить необходимые па-

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

раметры, задавать команды (например, приостанавливать и возобновлять работу сканера) и просматривать информацию об его текущем состоянии. Отчет для сборщика информации включает в себя результаты проверок, произведенных сканером, предназначенные для формирования общего отчета о результатах проверки политик.

Общая архитектура конфигуратора изображена на рис. 5.

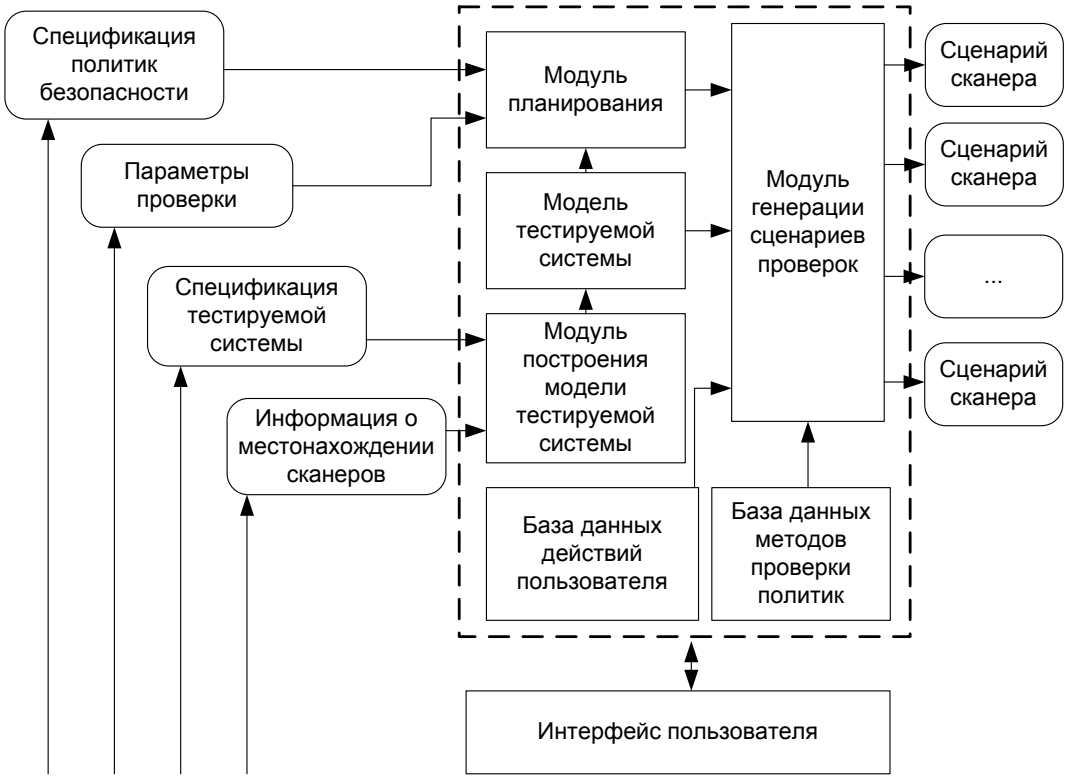

Рис. 5. Архитектура конфигуратора.

Пользователь задает спецификацию тестируемой системы, спецификацию политик безопасности и параметры проверки политик с помощью интерфейса пользователя. Интерфейс пользователя также передает конфигуратору информацию о местонахождении сканеров. Эта информация посылается конфигуратору, и обрабатывается модулем предварительной обработки данных. Сценарии для сканеров генерируются модулем генерации сценариев. Каждый сценарий сканера описывает последовательность проверок определенной части политик безопасности для фрагмента сети, в котором находится сканер. Конфигуратор передает сформированные сценарии сканерам.

Спецификация политики безопасности представляет собой описание политики в виде совокупности правил на формальном языке SPL. Спецификация тестируемой системы, содержит описание конфигурации данной системы на SDL. Параметры проверки задают подмножество правил политик для проверки, часть тестируемой системы, в которой будет выполняться проверка, и параметры пользователей, для которых будут проверяться политики.

Модуль построения модели системы на основе спецификации тестируемой системы формирует модель данной системы и дополняет данную модель информацией о местоположении сканеров в системе. Модуль планирования разделяет правила политик безопасности на множества, предназначенные для

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

проверки отдельным сканерам. Данные множества модуль планирования передает модулю генерации сценариев для формирования сценариев проверки. Модуль генерации сценариев проверок предназначен для формирования сценариев проверки правил политик безопасности. Каждый сценарий представляет собой последовательность элементарных операций пользователя над активами с фактическими параметрами этих операций. База данных действий пользователя содержит описание всех возможных операций пользователя над активами и описание параметров этих операций. База данных методов проверки политик содержит информацию, необходимую для формирования сценария проверки политик различных классов.

Общая архитектура сборщика информации представлена на рис. 6.

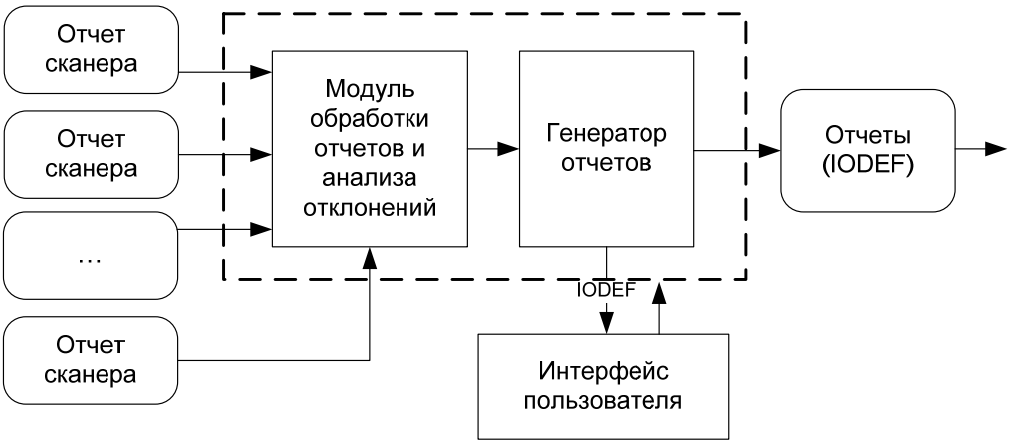

Рис. 6. Архитектура сборщика информации.

Сборщик информации предназначен для сбора отчетов со сканеров, их обработки и обобщения, формирования сводных отчетов и передачи их интерфейсу пользователя. Он получает отчеты от сканеров. Отчеты содержат выявленные нарушения политик безопасности. Модуль обработки отчетов и анализа отклонений обрабатывает полученные отчеты, обобщает информацию, содержащуюся в них, и передает обработанные данные генератору отчетов. Генератор отчетов на основе обработанных данных формирует отчеты, предназначенные для демонстрации пользователю результатов сканирования, и передает созданные отчеты интерфейсу пользователя.

4. Обобщенная методика функционирования системы

Входными данными для методики выступают спецификация тестируемой системы, спецификация проверяемых политик безопасности и параметры проверки, включающие подмножество правил политик для проверки, часть тестируемой системы, в которой будет выполняться проверка, и параметры субъектов и объектов, для которых будет проводиться проверка. Для генерации сценариев проверки используется также информация о местоположении сканеров в системе. В качестве *выходных данных* служат отчеты о выявленных нарушениях выполнения политик.

Методика задается в виде комплекса алгоритмов работы каждого компонента при проверке каждого класса политик. Охарактеризуем кратко сущность алгоритмов работы отдельных компонентов СПМ.

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Общий алгоритм работы конфигуратора:

1. Принять входные данные;
2. По спецификации тестируемой системы построить модель тестируемой системы;
3. Получить информацию о местоположении сканеров;
4. Обозначить местоположение сканеров в тестируемой системе;
5. Разделить правила политик на множества правил для каждого из находящихся в системе сканеров по сегменту сети, для которого справедливы данные правила;
6. Перебрать все множества правил полученные на предыдущем шаге;
7. Сформировать сценарий проверки данного множества правил для данного сканера:
   - Перебрать все правила из данного множества;
   - Добавить к сценарию проверку данного правила специфичным для политики алгоритмом;
8. Передать сформированные сценарии сканерам для выполнения.

Общий алгоритм выполнения сценария проверки сканером: (1) выполнить проверки правил в порядке, предусмотренном сценарием; (2) в случае нахождения отклонений от правила добавить результаты проверки и условия, при которых возникают отклонения, к отчету сканера; (3) передать результаты проверки сборщику информации.

Общий алгоритм работы сборщика информации: (1) получить результаты проверки от сканеров; (2) обработать и обобщить результаты проверки, сформировать отчет о выявленных нарушениях; (3) выдать сформированный отчет.

Детали реализации методики отличаются при проверке каждого класса политик. Для полноценной проверки выполнения правила политики безопасности, необходима полная проверка успешности/неуспешности всех допустимых операций над всеми активами, с использованием учетных записей всех пользователей системы. Такой подход очень трудоемок и не всегда позволяет проверить выполнение политики за приемлемое время. Существует ряд других подходов, которые позволяют с определенной долей вероятности утверждать, что политика выполнена, и лишены недостатков полного перебора. Рассмотрим два подхода, которые были реализованы в прототипе СПМ.

Первый подход («экспресс-анализ») состоит в извлечении необходимой информации из правила политики и проведении проверки для одного из представителей класса объектов проверки, выбранного случайным образом. Например, если правило политики формулируется следующим образом: «Администраторам компьютера разрешено создавать, модифицировать и удалять любые файлы в локальной файловой системе», то для проверки этого правила необходимо проверить возможность создания, модификации и удаления, любого, случайно выбранного, файла в системе, для случайно выбранного пользователя с ролью администратора. Такой подход подразумевает существенное сужение списка проверяемых операций там, где это возможно за счет анализа правил политики. Там же, где правило подразумевает любой объект какого-либо класса, проверка проводится для одного, случайно выбранного, объекта.

Второй подход является модификацией первого. При проверке выполнения политики для любого объекта какого-либо класса, второй подход выбирает не одного представителя данного класса, а несколько таких представителей. Количество представителей зависит от количества объектов данного класса.

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Таким образом, полный перебор является наиболее трудоемким из всех рассмотренных подходов, следующим по трудоемкости является выбор нескольких случайных объектов для проверки, и наименее трудоемким является выбор одного случайного представителя.

В силу различия в политиках, для каждого класса политик необходимо использовать свой алгоритм проверки. Рассмотрим обобщенные алгоритмы проверки для политики авторизации и аутентификации.

Алгоритм проверки политики авторизации: (1) перебрать всех пользователей; (2) перебрать все операции над активами; (3) перебрать все активы; (4) проверить успешность данной операции для данного пользователя над данным активом.

Существует четыре возможных исхода проверок:

1. Если существует правило, разрешающее операцию, и операция выполнилась успешно, значит, политика не нарушена;
2. Если существует правило, разрешающее операцию, и операция выполнилась неуспешно, значит, политика нарушена;
3. Если не существует правила, разрешающего операцию, и операция выполнилась неуспешно, значит, политика не нарушена;
4. Если не существует правила, разрешающего операцию, и операция выполнилась успешно, значит, политика нарушена.

Наиболее серьезным нарушением политики можно считать нарушение типа 4, поскольку оно свидетельствует о возможности выполнить операцию, фактически запрещенную политикой. Другой тип нарушения — тип 2, который свидетельствует о невозможности выполнить операцию, предусмотренную политикой. В табл. 1 показаны возможные результаты выполнения операций. Плюсами отмечены исходы, подтверждающие политику, минусами — исходы, свидетельствующие о нарушении политики.

Таблица 1

Результаты проверки отдельной операции

	Успех	Неуспех
Существует разрешающее правило	+	- Невозможно выполнение разрешенной операции.
Не существует разрешающего правила или существует запрещающее правило	- Возможно выполнение запрещенной операции.	+

Алгоритм проверки политики аутентификации: (1) перебрать всех пользователей; (2) перебрать все активы; (3) для данного пользователя сгенерировать сценарий аутентификации в данном активе; (4) проверить успешность выполнения сценария.

В процессе выполнения проверки политики на реальной сети может возникнуть необходимость выполнения «опасных операций». Под опасными операциями мы будем понимать операции, которые могут привести к нарушению целостности компьютерной системы и содержащейся в ней информации. Например, для файловой системы такими операциями будут удаление или изменение файла. Предотвратить нарушение целостности в результате выполнения таких операций поможет принятие следующих мер: (1) сохранение резервной копии информации, которая является объектом воздействия опасной операции; (2) проверка результатов опасной операции (с целью определения того, нару-

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

шена ли целостность информации); (3) восстановление поврежденной информации из резервной копии. Выполнение данных трех действий должно быть предусмотрено для любой из опасных операций.

Другим способом обхода опасных операций является введение нескольких режимов работы системы. Один их этих режимов позволяет проводить тестирование в полном объеме, включая опасные операции. Другой режим позволяет проводить тестирование всех операций, за исключением опасных. Опасные операции являются наиболее трудоемкими, так как требуют выполнения дополнительных действий. Поэтому использование безопасного режима также существенно увеличивает скорость работы системы.

Другим аспектом выполнения проверок на реальной сети является возможность возникновения конфликтов с работой других пользователей. Такие конфликты могут возникать в результате выполнения практически любых операций, как опасных, так и безопасных с точки зрения целостности информации. Например, сканеру может быть запрещен доступ на чтение к файлу, который заблокирован для редактирования пользователем.

Одним из решений проблемы конфликтов может быть выделение специального времени для проверки системы, в которое пользователи не работают в системе, либо работают очень редко. Проблема такого решения — невозможность полноценной проверки правил политик, которые имеют ограничения действия по времени. Другое решение проблемы конфликтов — попытка определить при их возникновении причину конфликта и учитывать эту причину при проверке. Например, можно вести учет действий пользователя в системе, и при возникновении конфликта, проверять, не занят ли ресурс пользователем. Если ресурс занят, отложить его проверку или считать проверку пройденной, поскольку пользователь успешно получил доступ к ресурсу.

5. Реализация программного прототипа

UML-диаграмма архитектуры *сканера* представлена на рис. 7.

Класс `Script` представляет собой сценарий, выполняемый сканером. Он состоит из массива шагов сценария. Шаги сценария выполняются в том же порядке, в каком они находятся в данном массиве. Каждый шаг сценария описывается классом `ScriptStep`. Интерфейс класса `Script` содержит метод, принимающий на вход XML-документ во внутреннем формате и формирующий из данного документа массив шагов сценария. Класс `ScriptStep` содержит имя клиентского приложения, выполняющего операцию на данном шаге, имя данной операции и массив параметров данной операции.

Класс `Interpreter` представляет собой интерпретатор сценария. Он получает на вход сценарий и последовательно перебирает все шаги сценария. Из каждого шага он извлекает имя клиентского приложения, а также имя и параметры операции. Интерпретатор находит в коллекции `Clients` объект, соответствующий извлеченному имени клиентского приложения. Все такие объекты реализуют интерфейс `IClient`. Затем, интерпретатор вызывает метод `performOperation`, определенный в интерфейсе `IClient`, и передает ему имя операции для выполнения вместе с параметрами данной операции. Метод `performOperation` выполняет переданную ему операцию с заданными параметрами и возвращает статус успешности выполнения операции. Возвращенный статус вместе с текущим объектом `ScriptStep` интерпретатор передает

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

генератору отчетов. Если текущий шаг сценария завершился успешно, интерпретатор переходит к следующему шагу, в противном случае выполнение сценария завершается.

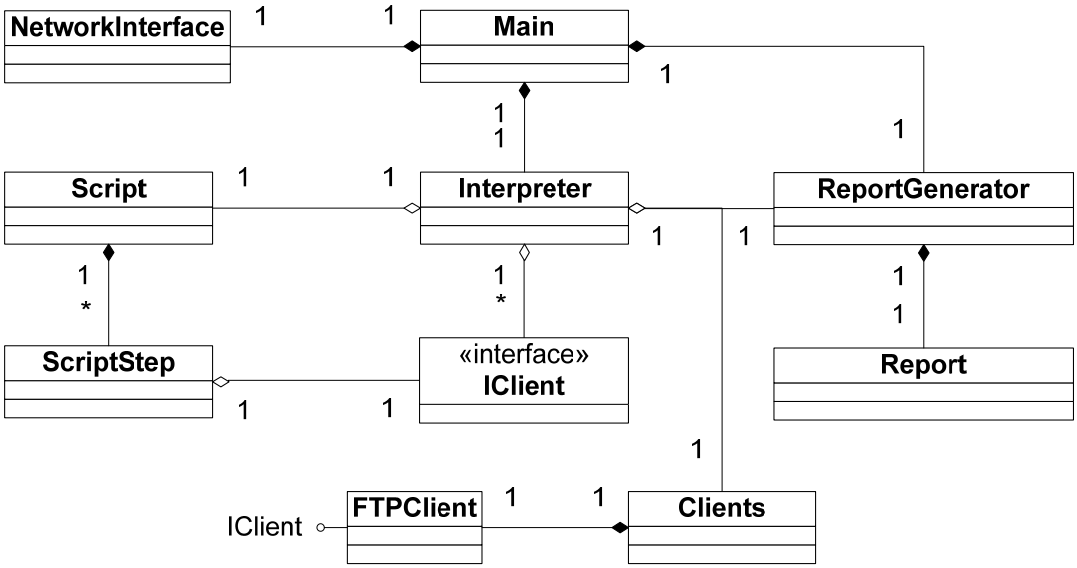

Рис. 7. UML-диаграмма архитектуры сканера.

Генератор отчетов представлен классом `ReportGenerator`. Задачей данного класса является формирование и выдача отчетов о ходе выполнения сценария. Генератор отчетов получает на вход описание выполненного шага сценария и статус завершения выполненной операции. Полученные данные он добавляет к отчету. Отчет сканера представлен классом `Report`. Данный класс содержит массив шагов сценария и статусов завершения этих шагов, переданных ему генератором отчетов. Интерфейс класса `Report` содержит метод, формирующий из содержимого класса XML-документ в формате IODEF.

Интерфейс `IClient` описывает интерфейс клиента, выполняющего отдельную операцию на заданном шаге сценария. Клиент может быть FTP-клиентом или приложением, работающим с файлами в локальной файловой системе. Тип клиента зависит от характера проверок, проводимых на данном шаге сценария. Тип клиента задается строковым идентификатором (например, `"ftp"`). Каждому типу клиента соответствует отдельный класс, реализующий интерфейс `IClient` (например, класс `FTPClient`). Все такие классы содержатся в коллекции `Clients` и могут быть получены с помощью идентификатора типа. Интерфейс `IClient` содержит метод `performOperation` со следующей сигнатурой:

```
public boolean performOperation(String operation, String[]
parameters);
```

где `operation` — имя операции, выполняемой клиентом (например, `"USER"`), `parameters` — массив параметров данной операции (например, `{"professor"}`), а возвращаемое значение: `true`, если операция была выполнена успешно, `false` — в противном случае.

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Класс `NetworkInterface` содержит методы, необходимые для установления соединения и обмена информацией между конфигуратором и сканером, сканером и сборщиком информации, сканером и интерфейсом пользователя.

Класс `Main` управляет основными потоками данных внутри сканера. Данный класс реализует следующий общий алгоритм работы сканера. После запуска сканера создаются классы `NetworkInterface`, `Interpreter`, `Clients` и `ReportGenerator`. Класс `NetworkInterface` переводится в режим ожидания внешних соединений. Класс `NetworkInterface` получает данные с инструкциями от внешних модулей СПМ и передает полученную информацию классу `Main`. Если конфигуратор передает сканеру сценарий для выполнения, то `Main` создает класс `Script`, инициализирует его с помощью XML-документа, полученного от конфигуратора, передает его классу `Interpreter`, дожидается окончания обработки, получает отчет от `ReportGenerator`, формирует XML-документ с отчетом для сборщика информации и передает данный отчет сборщику информации. Если запрос поступил от интерфейса пользователя, `Main` формирует отчет о статусе сканера и передает данный отчет в качестве ответа интерфейсу.

На рис. 8 представлена UML-диаграмма модели сети. Модель сети содержит три набора элементов сети: объекты защиты (`ISecuredObject`), серверы (`IActionPerformer`) и сканеры (`Scanner`). Под объектом защиты мы будем понимать некоторые ресурсы, доступ к которым регулируется политиками безопасности. Под сервером мы будем понимать некую систему, обеспечивающую доступ и производящую операции над объектами защиты. Сервер содержит массив объектов защиты, к которым он предоставляет доступ. Сервер также может являться объектом защиты. Сканер — это класс, содержащий всю необходимую СПМ информацию об отдельном сканере.

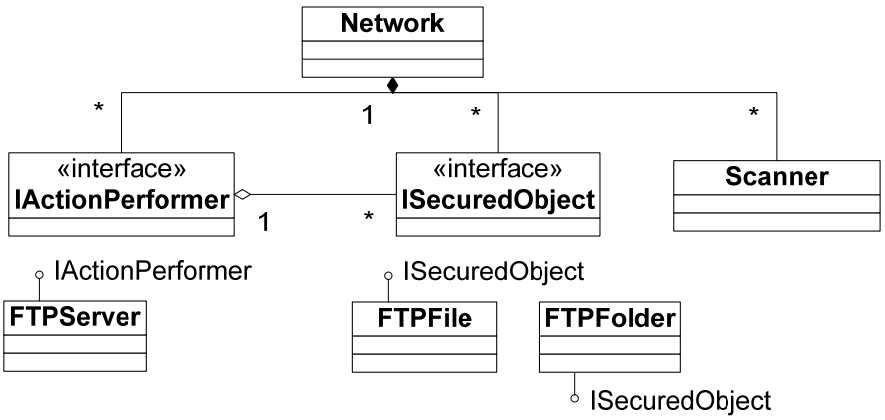

Рис. 8. UML-диаграмма модели сети.

Классы `FTPFile` и `FTPFolder` являются примерами объектов защиты. Класс `FTPServer` является примером сервера FTP, обеспечивающего доступ к файлам и папкам. Отметим, что такая модель сети не учитывает некоторые взаимосвязи элементов сети. Например, в ней отсутствует информация о брандмауэрах или об иерархии файлов и папок на FTP сервере. Такие ограничения существенно упрощают модель сети, с другой стороны модель, тем не менее, содержит всю необходимую для работы системы информацию: описание объектов защиты, серверов и связей между ними.

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

На рис. 9 представлена UML диаграмма политики безопасности. Политика безопасности сканируемой системы (`Policy`) содержит массив правил политики безопасности. Правила политики безопасности могут быть двух видов: разрешающие/запрещающие (`ActionRule`) и правила инициирующие действия (`ReactionRule`).

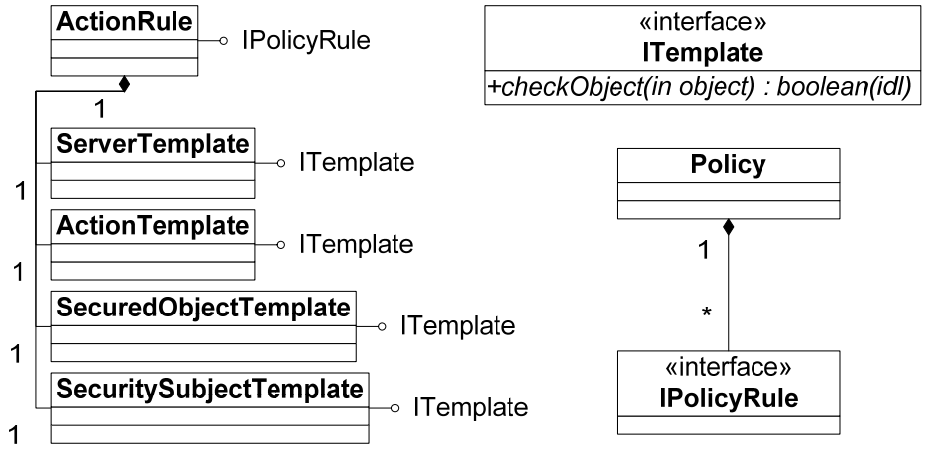

Рис. 9. UML-диаграмма политики безопасности.

Интерфейс `ITemplate` содержит методы для проверки объектов на соответствие условию и методы для перебора всех возможных вариантов объектов, соответствующих условию. Классы `SecuredObjectTemplate`, `SecuritySubjectTemplate`, `ServerTemplate` и `ActionTemplate` реализуют интерфейс `ITemplate` и предназначены для обработки объектов защиты, субъектов операции, серверов и операций соответственно.

Класс `ActionRule` представляет собой разрешающее/запрещающее правило. Он содержит условия, налагаемые на объект защиты, субъект операции, сервер и саму операцию, а также логическое значение, которое определяет, запрещена или разрешена данная операция при выполнении данных условий.

На рис. 10 представлена UML-диаграмма *конфигуратора* СПМ.

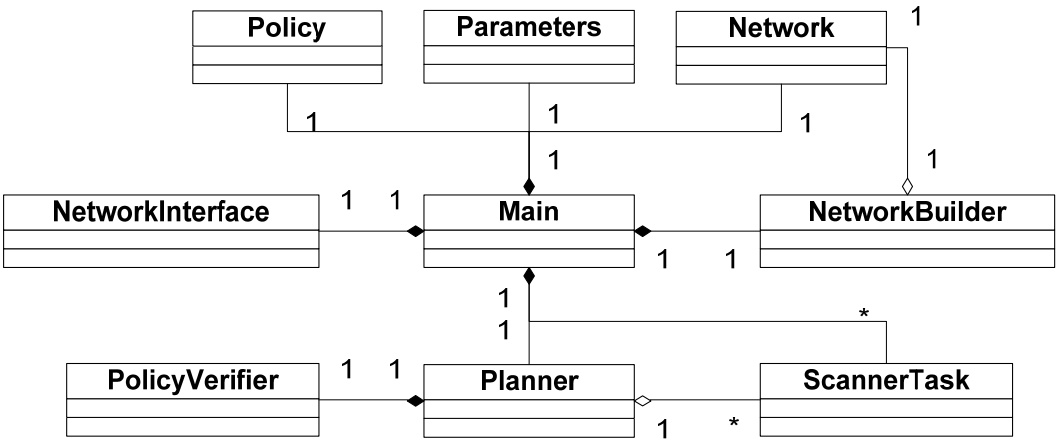

Рис. 10. UML-диаграмма конфигуратора СПМ.

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Класс `NetworkInterface` обеспечивает взаимодействие конфигуратора с компьютерной сетью и аналогичен классу `NetworkInterface` в описании сканера.

Класс `Main` реализует взаимодействие всех основных компонентов и передачу данных между ними. Класс `Main` получает по сети входную информацию двух видов: (1) от сканеров, находящихся в сети, — информацию об их местоположении (эта информация включается классом `NetworkBuilder` в модель сети); (2) от интерфейса пользователя — спецификацию тестируемой системы, спецификацию политик безопасности и параметры проверки политики безопасности (эта информация используется для построения модели сети, модели политики безопасности и при генерации сценариев проверки данной политики).

Класс `NetworkBuilder` предназначен для создания и инициализации модели сети, по полученной от интерфейса пользователя спецификации и информации от сканеров, находящихся в сети. Он получает входную информацию в виде XML-документов: спецификацию сети на языке SDL и информацию от сканеров во внутреннем формате. Результатом его работы является класс Network, представляющий собой модель сети, описанную выше.

Класс `Policy` представляет собой модель политики безопасности. Входная информация для инициализации этого класса представляет собой XML-документ в формате SPL.

Класс `Parameters` представляет собой параметры проверки политики.

Класс `Planner` представляет собой модуль планирования. Входной информацией для него являются политика безопасности (объект класса `Policy`), модель тестируемой системы (объект класса `Network`) и параметры проверки политики (объект класса `Parameters`). Модуль планирования выбирает правила политики безопасности и проверяющие сканеры, в соответствии с ограничениями, которые наложены параметрами проверки. Сформированные пары `<правило политики, сканер>` модуль планирования передает объекту класса `PolicyVerifier` для формирования группы сценариев проверяющих данное правило с помощью данного сканера.

Класс `PolicyVerifier` генерирует группу сценариев для проверки данного правила политики с помощью данного сканера. Сам по себе это класс является абстрактным, и различные его конкретизации реализуют различные алгоритмы проверки.

На рис. 11 представлена UML-диаграмма *сборщика информации*.

Класс `NetworkInterface` обеспечивает взаимодействие сборщика информации с компьютерной сетью и полностью аналогичен классам с таким же названием в описании сканера безопасности и конфигуратора.

Класс `Main` обеспечивает передачу потоков данных между основными классами системы. Класс `Main` получает по сети отчеты от сканеров (`scanner.Report`) и передает их обработчику отчетов. Обработчик отчетов передает информацию о выявленных нарушениях генератору отчетов для внесения в итоговый отчет. Итоговый отчет передается интерфейсу пользователя.

Класс `scanner.Report` практически полностью был описан выше, при описании архитектуры сканера. Также он содержит метод, инициализирующий класс с помощью XML документа в формате IODEF.

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

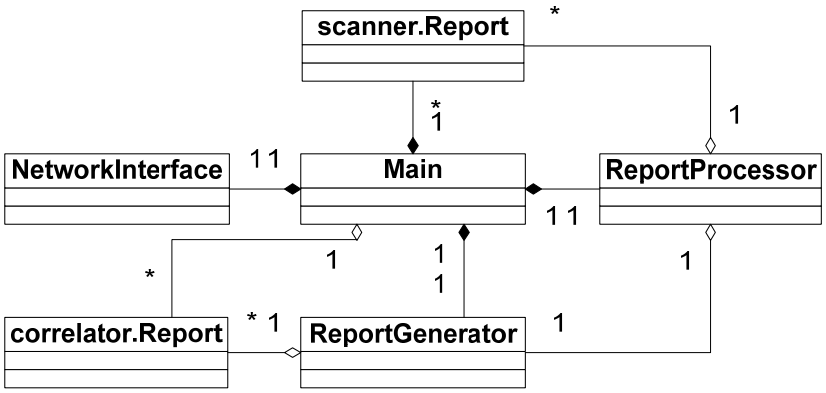

Рис. 11. UML-диаграмма сборщика информации.

Класс `ReportProcessor` представляет собой обработчик отчетов. Он анализирует полученные от сканеров отчеты, обобщает содержащуюся в них информацию и передает обобщенную информацию генератору отчетов.

Класс `ReportGenerator` представляет собой генератор отчетов. Он получает обобщенную информацию о выявленных в системе нарушениях политики от обработчика отчетов и формирует отчет сборщика информации.

Класс `correlator.Report` представляет собой отчет сборщика информации. Он содержит обобщенную информацию о выявленных нарушениях политики безопасности. Интерфейс `correlator.Report` содержит метод для сохранения данной информации в виде XML документа в формате IODEF.

6. Эксперименты по проверке политики безопасности

На рис. 12 представлена структура компьютерной сети, используемой для проведения экспериментов с прототипом СПМ.

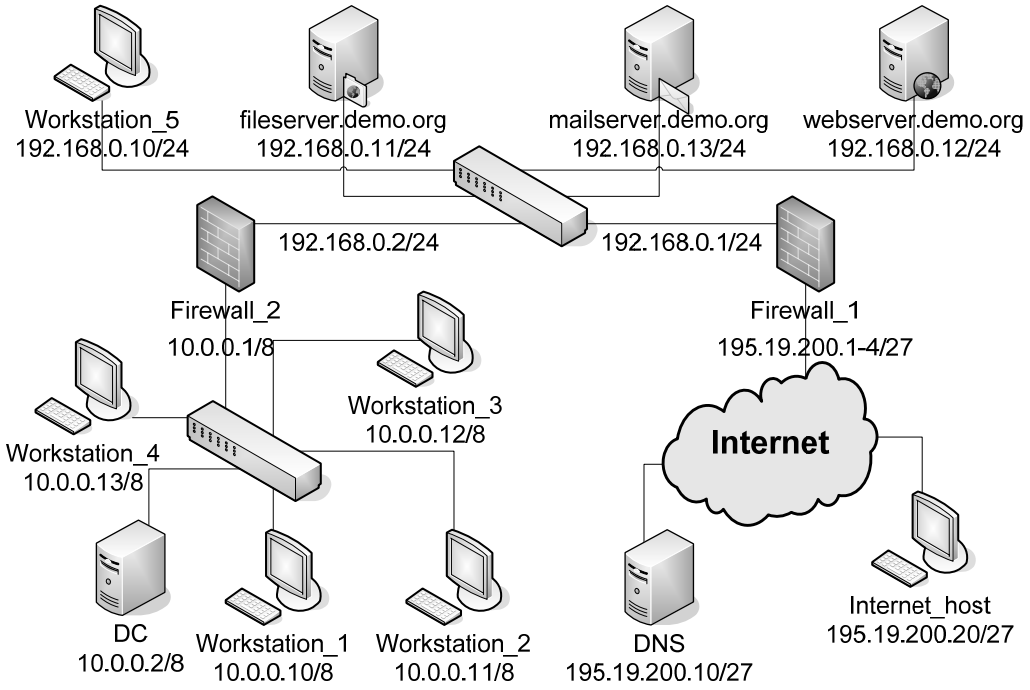

Рис. 12. Структура компьютерной сети.

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Компьютерная сеть состоит из трех подсетей: (1) части глобальной сети Internet с IP адресами 195.19.200.*; (2) демилитаризованной зоны (ДМЗ) с IP адресами 192.168.0.*; (3) локальной вычислительной сети с IP адресами 10.0.0.*.

Правила политик безопасности могут быть описаны на естественном языке следующим образом:

- Пользователь, играющий роль Administrator, имеет право на конфигурирование сервера из локальной сети;
- Пользователь, находящийся в глобальной сети Интернет и играющий роль Administrator, User, Student или Professor, имеет право на чтение документов FTP-сервера, находящихся в открытом доступе;
- Пользователь, играющий роль Administrator, Student, Professor или User, имеет право на чтение документов FTP-сервера, находящихся в закрытом доступе, из локальной сети;
- Пользователь, играющий роль Student или Professor, имеет право на чтение, создание, удаление и модификацию документов FTP-сервера, находящихся в его личном доступе, из локальной сети.

Рассмотрим примеры нескольких сценариев, сгенерированных конфигуратором для сканеров. Каждый сценарий предназначен для проверки выполнимости одной операции сканерами, находящимися в различных местах сети. В зависимости от местоположения сканера успешность/неуспешность выполнения сценария либо подтверждает выполнение политики безопасности, либо опровергает ее. Каждый сценарий необходимо связать с правилом политики, которое он проверяет.

Сценарий 1 (возможность чтения):

```
ftp.openConnection("fileserver.demo.positif.org");
ftp.USER("scannerUser");
ftp.PASS("secret");
ftp.CWD("public");
ftp.LIST(".");
ftp.RETR("public/a.txt");
ftp.QUIT();
ftp.closeConnection();
```

Сценарий 2 (невозможность записи):

```
ftp.openConnection("fileserver.demo.positif.org");
ftp.USER("scannerUser");
ftp.PASS("secret");
ftp.CWD("public");
ftp.MKD("public/unique");
ftp.STOR("unique.txt", "public/unique.txt");
ftp.QUIT();
ftp.closeConnection();
```

Сценарий 2.3 (невозможность удаления):

```
ftp.openConnection("fileserver.demo.positif.org");
ftp.USER("scannerUser");
ftp.PASS("secret");
ftp.CWD("public");
ftp.RMD("public/unique");
ftp.DELE("public/unique.txt");
ftp.QUIT();
ftp.closeConnection();
```


Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Основной экран интерфейса пользователя показан на рис. 13. Панель закладок на основном экране интерфейса пользователя позволяет просматривать информацию, касающуюся текущей загруженной конфигурации сети, текущей загруженной политики безопасности, состояния сканеров, а также отчет о выявленных отклонениях от политики безопасности.

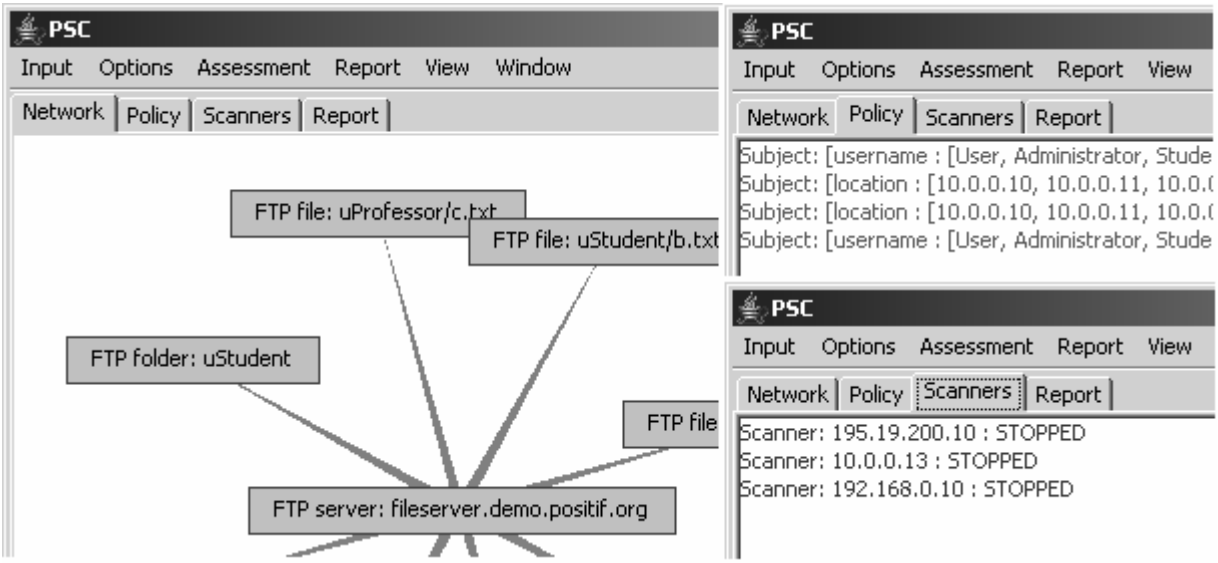

Рис. 13. Главный экран интерфейса пользователя.

Рис. 14 показывает результаты работы системы в тестовой сети. Вкладка Report отображает отчеты о выявленных отклонениях от политики безопасности. С помощью вкладки Log можно просмотреть полный лог действий системы.

Результаты проведенной проверки политики работы с FTP-сервером можно свести в табл. 2.

Таблица 2

Результаты проверки FTP сервера

	Сканер 10.0.0.13	Сканер 195.19.200.10	Сканер 192.168.0.10
Сценарий 1	+	**+**	**+**
Сценарий 2.1	+	**+**	-
Сценарий 2.2	-	-	-
Сценарий 2.3	-	-	-
Сценарий 3.1	+	**+**	-
Сценарий 3.2	-	-	-
Сценарий 3.3	-	-	-
Сценарий 4.1	+	**+**	-
Сценарий 4.2	+	**+**	-
Сценарий 4.3	+	**+**	-

Жирным шрифтом в таблице выделены выявленные отклонения от политики безопасности. В первой строчке показана возможность сконфигурировать сервер не только из локальной сети, но также из ДМЗ и Интернет. Во втором столбце показана возможность выполнения допустимых операций не только из локальной сети, но также и из ДМЗ, что противоречит политике.

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

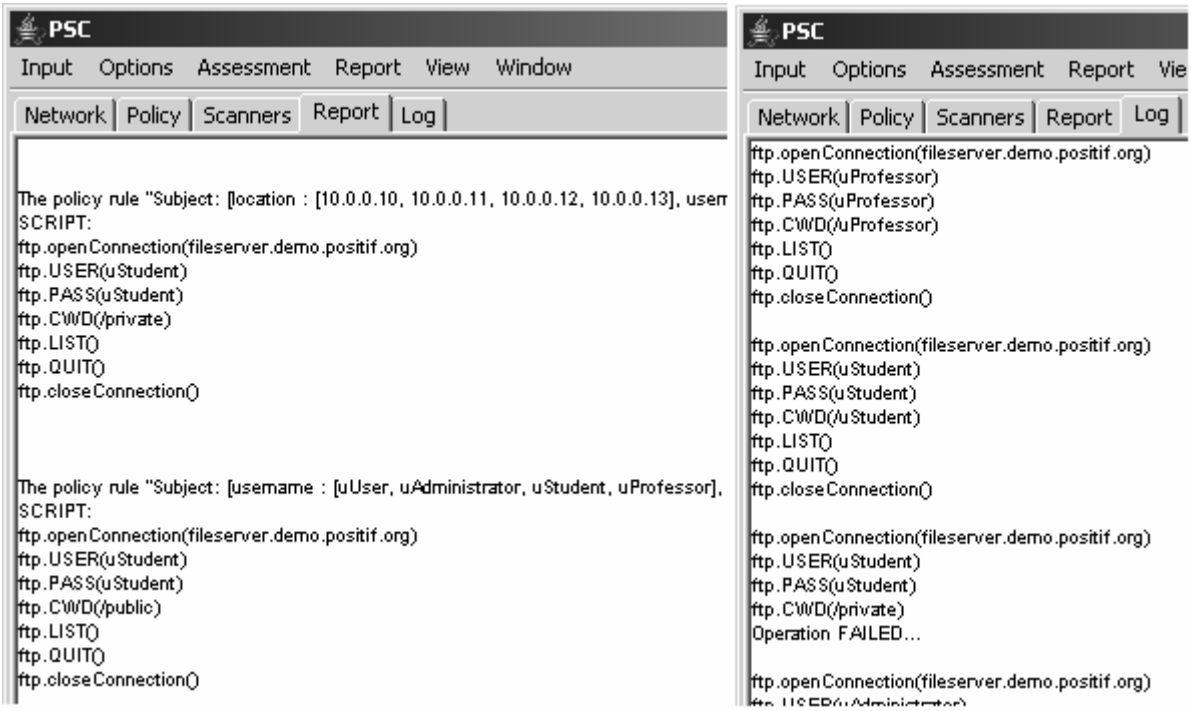

Рис. 14. Результаты работы системы.

7. Заключение

В работе предложен подход к проактивному мониторингу выполнения политики безопасности в компьютерных сетях. Предлагаемый подход основан на имитации различных действий пользователя в исследуемой компьютерной сети, которые способны опровергнуть или подтвердить выполнение политики безопасности. Подход обладает следующими особенностями. Распределенная архитектура СПМ с одним координирующим модулем позволяет моделировать взаимодействие нескольких пользователей. Таким образом, становится возможным смоделировать нарушение политики безопасности несколькими пользователями, находящимися в сговоре, а также ситуацию нарушения политики в результате одновременной работы большого числа пользователей.

СПМ имеет три режима работы, которые отличаются точностью и скоростью проверки: режим проверки с высокой точностью, средней точностью, малой точностью. Первый режим позволяет провести исчерпывающую проверку политики безопасности, но требует продолжительного времени работы. Режим малой точности служит для экспресс-проверки выполнения политики безопасности за короткое время. Режим средней точности позволяет достичь баланса между скоростью и временем работы.

СПМ позволяет производить проверку политики безопасности без выполнения «опасных» действий, то есть действий, которые могут нарушить целостность информации в системе. Если проверка производится в обычном режиме, то СПМ предпринимает дополнительные меры для предотвращения потерь информации в результате выполнения «опасных» действий. В работе рассмотрены возможные методы решения проблемы конфликтов взаимодействия между реальными пользователями и СПМ. Представлена общая методика проверки правил политики безопасности, описаны методика проверки правил политики авторизации и методика оценки результатов данной проверки.

Труды СПИИРАН. 2006. Вып. 3. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2006. Issue 3. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Направлениями дальнейших исследований является совершенствование предложенных моделей и методик проактивного мониторинга и проведение экспериментальной оценки предложенных решений.

Работа выполнена при финансовой поддержке РФФИ (проект №04-01-00167), программы фундаментальных исследований ОИТВС РАН (контракт №3.2/03) и при частичной финансовой поддержке, осуществляемой в рамках проекта Евросоюза POSITIF (контракт IST-2002-002314).

Литература

1. *Котенко И. В., Юсупов Р. М.* Перспективные направления исследований в области компьютерной безопасности // Защита информации. Инсайд. 2006. № 2. С. 46–57.
2. *Котенко И. В., Степашкин М. В., Богданов В. С.* Архитектуры и модели компонентов активного анализа защищенности на основе имитации действий злоумышленников // Проблемы информационной безопасности. Компьютерные системы. 2005. № 4.
3. *Russell D., Gangemi G. T.* Computer Security Basics. Sebastopol: O'Reilly & Associates, 1991. 448 p.
4. *Marriott D., Sloman M.* Management Policy Service for Distributed Systems // Proceedings of the Third IEEE International Workshop on Services in Distributed and Networked Environments. Macau, 1996. P. 2–9.
5. *Sademies A.* Process Approach to Information Security Metrics in Finnish Industry and State Institutions. Espoo: VTT Technical Research Centre of Finland, 2004. 89 p.
6. *Canavan S.* An Information Security Policy Development Guide for Large Companies [Электронный ресурс] / S. Canavan; SANS Institute, 2004. 21 p. // <http: // www.sans.org / rr / whitepapers / policyissues / 1331.php> (по состоянию на 17.03.2006).
7. *Kee C. K.* Security Policy Roadmap — Process for Creating Security Policies [Электронный ресурс] / C. K. Kee; SANS Institute, 2001. 8 p. // <http: // www.sans.org / rr / whitepapers / policyissues / 494.php> (по состоянию на 17.03.2006).
8. Security Assessment: Case Studies for Implementing the NSA IAM / *Rogers R., Miles G., Fuller E. et al.* Rockland: Syngress, 2004. 448 p.
9. *Wack J., Tracy M., Souppaya M.* Guideline on Network Security Testing // NIST Special Publication 800-42. Gaithersburg, 2003. 92 p.
10. *Carney M., Loe B.* A Comparison of Methods for Implementing Adaptive Security Policies // Proceedings of the 7th USENIX Security Symposium. San Antonio, 1998. P. 1–14.
11. *Ghosh A.K., O'Connor T., McGraw G.* An Automated Approach for Identifying Potential Vulnerabilities in Software // Proceedings of the 1998 IEEE Symposium on Security and Privacy. Oakland, 1998. P. 104–114.
12. IPSECvalidate — A Tool to Validate IPSEC Configurations / *Sailer R., Acharya A., Beigi M. et al* // Proceedings of the 15th Conference on Systems Administration. San Diego, 2001. P. 19–24.
13. *Gama P., Ferreira P.* Obligation Policies: An Enforcement Platform // Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks. Stockholm, 2005. P. 203–212.
14. *Beigi M. S., Calo S., Verma D.* Policy Transformation Techniques in Policy-Based Systems Management // Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks. Yorktown Heights, 2004. P. 13–22.
15. Policy-Based Validation of SAN Configuration / *Agrawal D., Giles J., Lee K.-W., et al* // Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks. Yorktown Heights, 2004. P. 77–86.
16. A Framework for Contractual Resource Sharing in Coalitions / *Firozabadi B. S., Sergot M., Squicciarini A. et al* // Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks. Yorktown Heights, 2004. P. 117–126.
17. *Strembeck M.* Embedding Policy Rules for Software-Based Systems in a Requirements Context // Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks. Stockholm, 2005. P. 235–238.
18. IODEF/IDMEF Solutions [Электронный ресурс] / PRESECURE Consulting GmbH, 2004 // <http: // www.ecsirt.net / service / products.html> (по состоянию на 17.03.2006).