Многоагентные технологии анализа уязвимостей

и обнаружения вторжений в компьютерных сетях*

И. В. Котенко, *д. т. н.,*
профессор СПИИРАН
ivkote@spiiras.nw.ru

Введение

В связи с бурным развитием задач распределенной обработки информации и использованием открытых сетей (Интернета), не приспособленных для защищенного обмена информации, вопросы защиты информационных ресурсов в компьютерных системах приобрели исключительную актуальность. Однако сложившееся состояние в области обеспечения безопасности этих систем, в том числе в области построения средств защиты компьютерных сетей, оставляет желать лучшего. Существующие системы защиты информационных ресурсов в компьютерных сетях, как правило, имеют централизованную структуру, характеризуются неразвитыми адаптационными возможностями, пассивными механизмами обнаружения атак, большим процентом ложных срабатываний при обнаружении вторжений, значительной деградацией трафика целевых информационных потоков из-за большого объема ресурсов, выделяемых на защиту и т. п. [1, 2].

Перспективным подходом к построению комплексных систем защиты информации в компьютерных сетях, позволяющим преодолеть некоторые из перечисленных недостатков, является использование интеллектуальных систем защиты информации, базирующихся на технологии многоагентных систем [3]. По сравнению с традиционными методами, этот подход позволяет существенно повысить эффективность механизмов защиты информации, в том числе их оперативность, адекватность, отказоустойчивость, устойчивость к деструктивным действиям, гибкость и т. д.

Настоящая работа посвящена разработке формальных моделей, архитектур и программных реализаций многоагентных систем защиты информации, служащих для анализа уязвимостей и обнаружения вторжений в компьютерные сети. Рассмотрены решения по созданию системы моделирования атак, предназначенной для активного анализа уязвимостей компьютерных сетей, и системы обнаружения вторжений.

Общее описание технологии проектирования, архитектуры типового агента и схемы кооперации многоагентных систем защиты информации

Интеллектуальный агент – это программно или аппаратно реализованная система, обладающая автономностью, совокупностью «ментальных свойств» и способная функционировать в сообществе с другими агентами [3].

Выделяют следующие «ментальные свойства» агента:

знания – постоянная часть знаний агента о себе, среде и других агентах, не изменяемая в процессе его функционирования;

убеждения – знания агента о среде (в частности, о других агентах), которые могут изменяться во времени и становиться неверными;

желания – состояния, достижение которых по разным причинам

* *Работа выполнена при поддержке РФФИ (проект № 01-01-00108).*

является для агента желательным, однако они могут быть противоречивыми, и потому агент не ожидает, что все они будут достигнуты;

намерения – то, что агент или обязан сделать в силу своих обязательств по отношению к другим агентам, или то, что вытекает из его желаний (то есть непротиворечивое подмножество желаний, выбранное по тем или иным причинам и которое совместимо с принятыми на себя обязательствами);

цели – конкретное множество конечных и промежуточных состояний, достижение которых агент принял в качестве текущей стратегии поведения;

обязательства по отношению к другим агентам – задачи, которые агент берет на себя по просьбе (поручению) других агентов в рамках кооперативных целей или целей отдельных агентов в рамках сотрудничества.

Согласно разработанной в лаборатории интеллектуальных систем СПИИРАН технологии процесс разработки многоагентных систем для любой предметной области, в том числе защиты информации в компьютерных сетях, предполагает решение двух высокоуровневых задач [4, 5]: (1) создание «Системного ядра» многоагентной системы; (2) клонирование программных агентов и отделение сгенерированной многоагентной системы от «Системного ядра». Для спецификации «Системного ядра» используются два компонента программного инструментария MASDK. Первый из них – это так называемый «Типовой агент» («Generic Agent»), который предназначен для создания высокоуровневой спецификации класса агента. Второй компонент служит для формирования проблемно-ориентированной архитектуры приложения, заполнения данных, знаний, а также определения коммуникационного компонента [4, 5].

Агенты, сгенерированные с использованием MASDK, имеют аналогичную архитектуру (рис. 1). Различия отражаются в содержании данных и баз знаний агентов. Каждый агент взаимодействует с другими агентами, средой, которая воспринимается и, возможно, изменяется агентами, а также пользователем, общающимся с агентами через пользовательский интерфейс.

Получатель входных сообщений и **отправитель исходящих сообщений** служат для выполнения соответствующих функций. Полученные сообщения фиксируются в **буфере входных сообщений**. **Процессор входного трафика** осуществляет обработку этих сообщений. Кроме того, данный компонент выполняет синтаксический анализ, интерпретацию KQML-сообщений и извлечение содержания сообщений.

Компонент **База данных диалогов агента** хранит для каждого входного сообщения его атрибуты (идентификаторы, тип сообщения, его источник и др.). Если сообщение предполагает ответное сообщение, ему ставится в соответствие определенное выходное сообщение.

Мета-автомат управляет семантической обработкой входных сообщений, направляя их для обработки соответствующими **автоматами**. Другое назначение этого компонента состоит в управлении параллельным выполнением различных процессов в рамках агента. Базовые вычисления агента, предписанные его ролью в многоагентной системе, выполняются набором **автоматов**, осуществляющих различные сценарии обработки входных сообщений.

Каждый класс агентов в соответствии с его функциональными возможностями располагает определенным набором шаблонов сообщений. Разработчик выполняет процедуру специализации с использованием компонента MASDK, называемого **Редактором шаблонов сообщений**. Шаблоны сообщений определены на языке KQML, и специализация заключается в назначении каждому шаблону соответствующего KQML-перформатива. Коммуникационный компонент каждого агента включает также данные о потенциальных адресатах сообщений данного шаблона. Эти данные назначаются на стадии клонирования экземпляров класса агента. Содержание сообщений определяется на XML.

Программный код описываемых в настоящей статье многоагентных систем защиты информации – агентно-ориентированной системы моделирования атак (АСМА) и многоагентной системы обнаружения вторжений (МСОВ) – написан с использованием инструментария MASDK, Visual C++, JAVA 2 и средств разработки XML.

АСМА имитирует входной трафик, то есть комбинацию потоков как нормальных, так и аномальных событий. Поток аномальных событий задает атаки на компьютерную сеть. Входной трафик аномальных событий соответствует последовательностям «однофазных» (про-

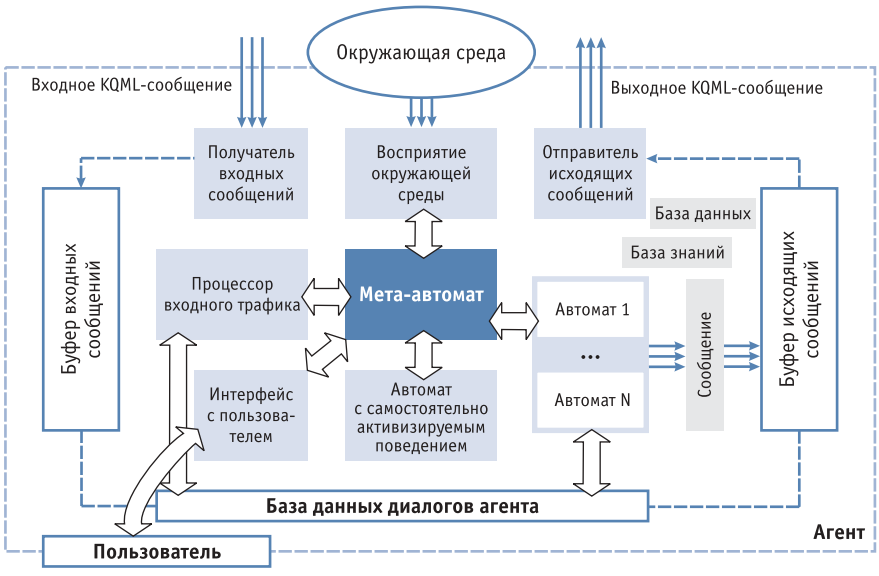

Рис. 1. ***Архитектура типового агента***

стых) атак, использующих различные хосты, рассматриваемые в качестве «точек входа инициирования» атак.

МСОВ ответственна за обнаружение атак на компьютерную сеть, формируемых АСМА.

В экспериментах с указанными системами используются различные конфигурации компьютерной сети. Эта сеть может включать несколько различных сегментов, и входной трафик может формироваться как внутри, так и снаружи защищаемой сети. Предполагается, что злоумышленники могут находиться в различных точках сети, как вне, так и внутри защищаемого фрагмента сети.

Агентно-ориентированная система моделирования атак на компьютерные сети

В предложенной формальной модели и прототипе агентно-ориентированной системы моделирования атак (АСМА) распределенные скоординированные атаки на компьютерную сеть рассматриваются в виде последовательности совместных действий агентов-хакеров, которые выполняются с различных хостов (рис. 2) [6–9]. Предполагается, что хакеры координируют свои действия согласно некоторому общему сценарию. На каждом шаге сценария атаки они пытаются реализовать некоторую частную подцель.

Таким образом, общий сценарий распределенной атаки задается как упорядоченная во времени последовательность шагов, направленных на последовательное достижение цели команды хакеров. При реализации сложных координированных атак в зависимости от установленной общей цели атаки специально назначенный мета-агент (агент-лидер) выбирает общий сценарий атаки и назначает сферы ответственности другим агентам. Агенты, ответственные за отдельные фрагменты (шаги) общего сценария, могут, в свою очередь, «нанять» других агентов или осуществить реализацию отдельных действий самостоятельно. Для этого предназначены специальные сценарии (планы) действий и протоколы обмена сообщениями.

Сценарий задается в виде последовательности намерений (подцелей) и способов их достижения (действий), которые могут описываться на различных уровнях детальности. Намерения самого нижнего уровня реализуются конкретными действиями, которые выражаются в виде сетевых пакетов, команд операционной системы или операций по работе с конкретным приложением.

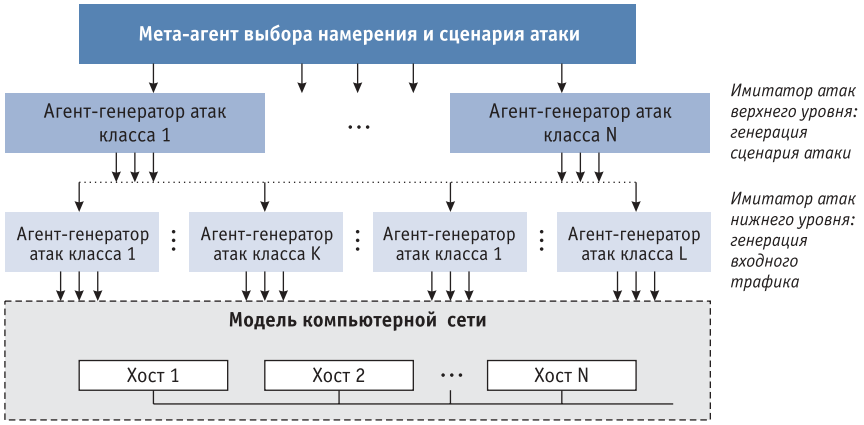

Рис. 2. ***Иерархическая структура АСМА при реализации сложных скоординированных атак***

Выделены следующие базовые классы высокоуровневых намерений хакеров:

- «разведка» с подклассами намерений «определение функционирующих хостов», «идентификация служб хоста», «идентификация типа операционной системы хоста», «определение разделяемых ресурсов», «определение учетных записей пользователей и групп», «определение приложений и заголовков»;
- «внедрение и реализация угрозы» с подклассами намерений «получение доступа к ресурсам хоста», «получение расширенного доступа к ресурсам хоста», «нарушение конфиденциальности», «нарушение целостности», «нарушение доступности», «создание потайных ходов».

Задача злоумышленника (группы злоумышленников), определяющая подмножество сценариев, которые ведут к достижению цели атаки, специфицируется следующей четверкой (такая спецификация используется в разработанном прототипе системы моделирования атак):

<Адрес сети (хоста) – цели атаки; Намерение; Параметры сети (хоста), известные злоумышленникам; Объект атаки>.

Спецификации атак задаются в распределенной базе знаний агентов, которая совместно используется агентами и структурируется в соответствии с разработанной онтологией предметной области «Атаки на компьютерные сети». Эта онтология представляет собой иерархию взаимосвязанных понятий, описывающих действия хакеров по реализации сетевых атак различных классов на разных уровнях детализации, и отношений на множестве таких понятий. Иерархия узлов (понятий) онтологии разбита на два подмножества, в соответствии с макро- и микроуровневым описанием атак. На макроуровне общий сценарий атаки моделируется множеством допустимых последовательностей скоординированных действий одного или нескольких хакеров. На микроуровне задается детальное описание атаки, при этом каждый шаг сценария макроуровня на микроуровне представляется последовательностью событий.

Содержательно каждому из узлов онтологии отвечает или подцель, которую пытается достичь хакер (его намерение), или набор действий, с помощью которых определенная подцель может быть достигнута.

Рассмотрим фрагмент разработанной онтологии, описывающий понятия макроуровня (рис. 3). На макроуровне понятия онтологии вышележащего уровня связываются с соответствующими понятиями смежного нижележащего уровня при помощи **отношений** трех ви-

дов: Part of – отношение декомпозиции («Целое – часть»), Kind of – отношение специализации («Понятие – класс понятия»), Seq of – отношение порядка выполнения («Операция – этап реализации»).

«Оконечные» понятия макроуровня раскрываются на микроуровне описания атак с использованием трех ранее представленных отношений, а также отношения Example of («Понятие – экземпляр реализации»). Так, например, понятие «Диапазонная проверка по ping» (DC) может связываться отношением Kind of с понятиями ping, fping вместе с gping, Pinger, Ping Sweep, WS_PingProPack, Netscan, которые соответствуют названиям утилит выполнения диапазонной проверки по ping. В свою очередь, каждое из этих понятий, например ping, может связываться отношением Seq of с сетевыми пакетами ICMP ECHO REQUEST, направленными на хост (или сеть) – цель атаки. Каждый пакет ICMP ECHO REQUEST связан отношением Example of со своей конкретной реализацией.

Спецификация планов реализации распределенных атак задается с использованием семейства **контекстно-свободных атрибутных стохастических грамматик**, связанных между собой операцией подстановки. Последовательности символов, выводимые в каждой их таких грамматик, соответствуют либо описанию множества сценариев атаки, ведущих к достижению заданной конечной цели хакера, либо описанию определенного типа атаки. То есть каждой высокоуровневой цели атаки (намерению хакера), а также каждому типу атаки ставится в соответствие своя грамматика. Содержательно **операция подстановки** приводит к замене символа *a* атаки определенного типа на последовательность символов $G(a)$ более низкого уровня ее описания. Таким образом, операция подстановки грамматик реализует механизм многоуровневого описания атаки.

Каждая из **грамматик** задается в виде пятерки: $G(C) = < V_N, V_T, S, P, A >$, где C – имя грамматики, V_N – нетерминальные символы, V_T – терминальные символы, S – начальный нетерминальный символ (аксиома), P – продукции грамматики, A – атрибуты и правила их вычисления. Каждая продукция из P задается в виде:

$$[(U)]\ X \rightarrow \alpha\ (Prob),$$

где U – условие применимости продукции в зависимости от предыстории атаки к текущему шагу; [] – обозначают необязательность элемента U; X – нетерминальный символ; α – цепочка терминальных и нетерминальных символов; *Prob* – начальная вероятность выбора правила.

Атрибутный компонент A грамматики служит для задания вероятностей выбора очередной подстановки и задания условий применимости продукций в зависимости от предыстории атаки. Вероятности, поставленные в соответствие продукциям, служат для обеспечения необходимого разнообразия генерируемых атак, а также для управления выбором очередной подстановки в случае, когда такой выбор неоднозначен.

В алгоритмической интерпретации процедур генерации атак каждой из грамматик ставится в соответствие **конечный автомат**. Основными элементами автомата являются: состояния; дуги переходов; параметры и условия переходов; скрипты реализуемых действий. Состояния каждого автомата подразделяются на три типа: начальное; промежуточные; конечное (метка этого состояния – End). Начальное и промежуточные состояния делятся на следующие классы: нетерминальные, инициирующие работу соответствующих вложенных автоматов; терминальные, взаимодействующие с моделью хоста; абстрактные (вспомога-

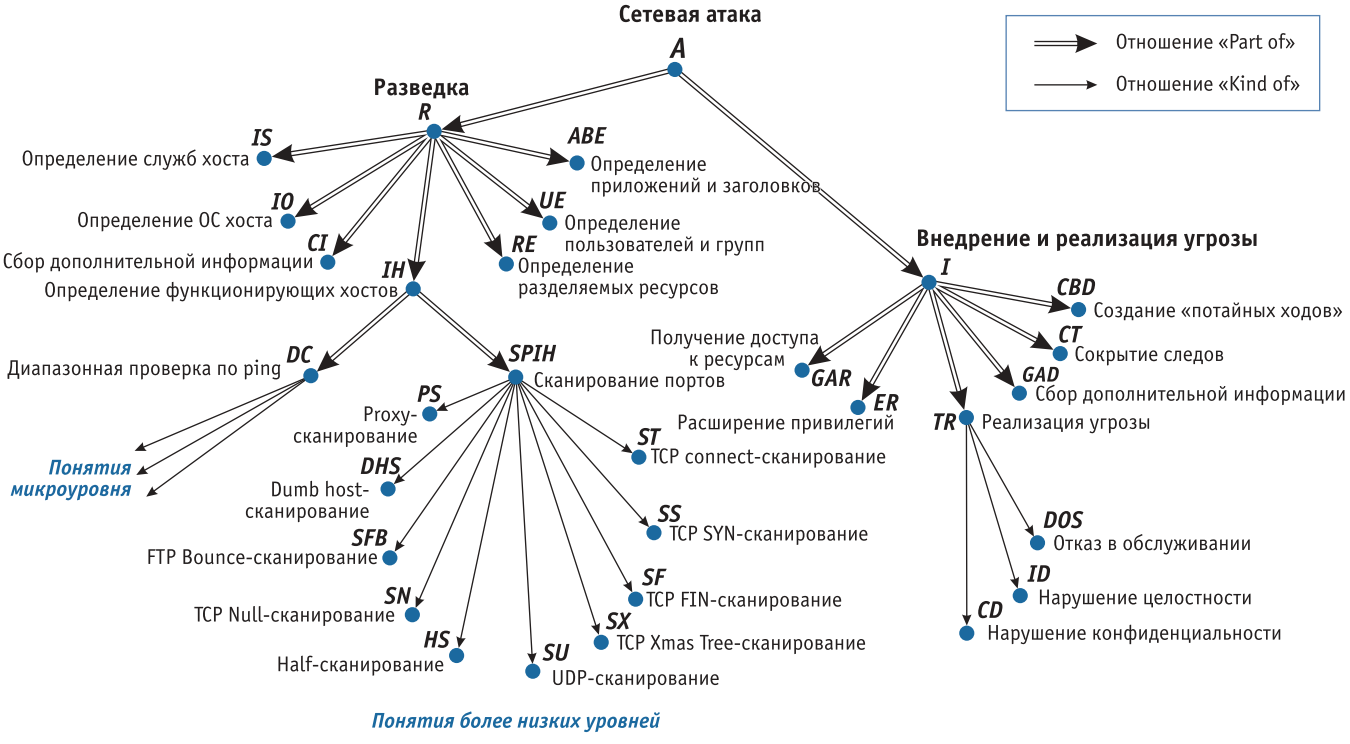

Рис. 3. ***Фрагмент онтологии «Сетевые атаки на компьютерные сети» макроуровня***

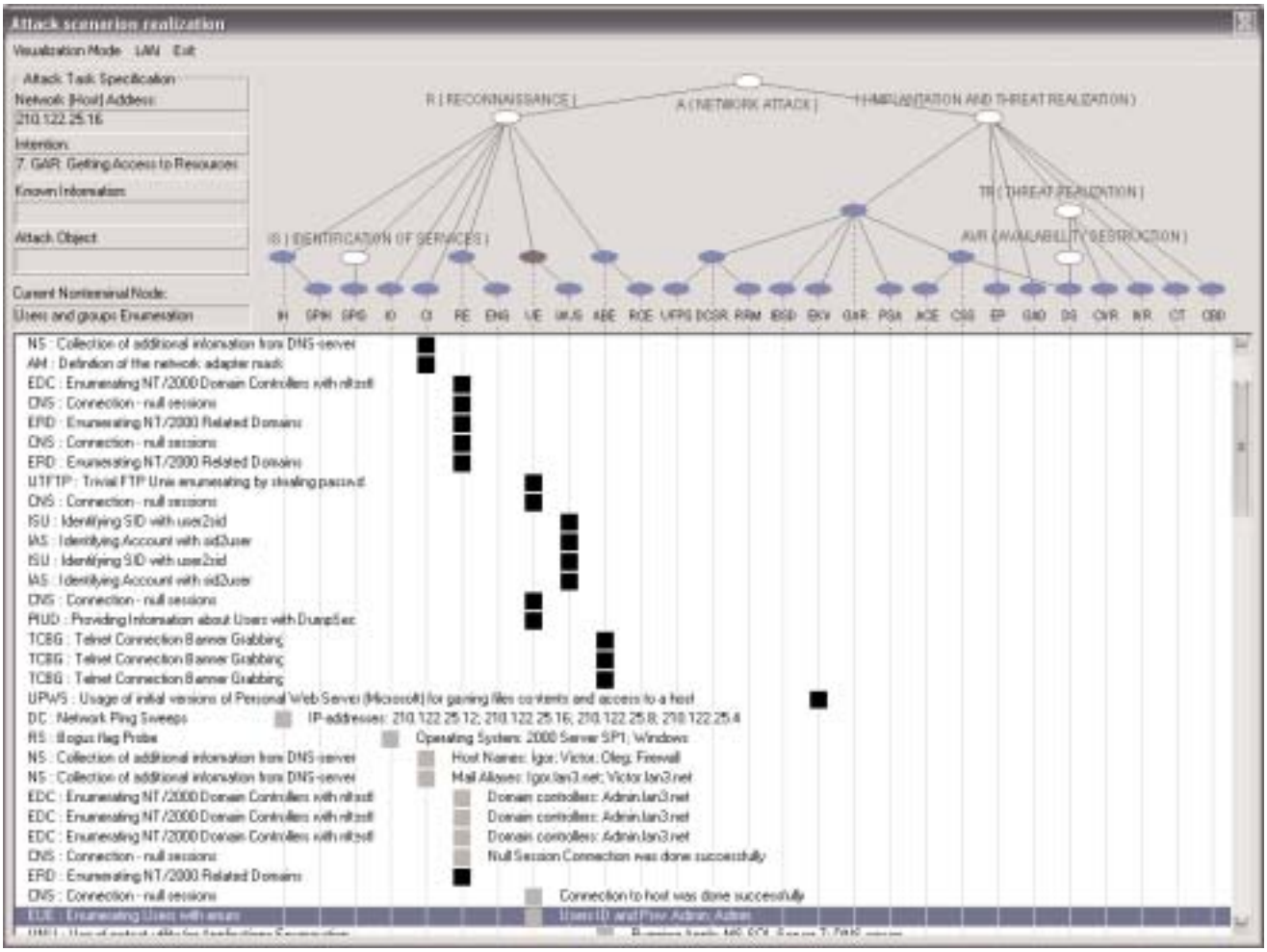

Рис. 4. ***Окно визуального представления развития сценария атаки GAR***

тельные), которые не инициируют работу других автоматов и не взаимодействуют с моделью хоста. Дуги переходов ставятся в соответствие правилам подстановки грамматик.

Команда агентов-хакеров функционирует в антагонистической среде, то есть в условиях, когда различные подсистемы защиты атакуемой сети пытаются препятствовать реализации атаки, обнаружить ее и «подавить» деятельность агентов-хакеров. Развитие атаки (выбор сценария и способа его реализации, изменение цели и т. п.) зависит от результата, достигнутого к текущему шагу атаки, то есть от того, какая информация собрана об атакуемом объекте, какие «препятствия» уже удалось преодолеть в процессе атаки, какие операции по реализации угрозы выполнены. Текущее «состояние» атаки определяется начальными сведениями об атакуемой сети (хосте), которыми обладал хакер, и информацией, собранной на уже выполненных шагах атаки, а также реализованной последовательностью шагов и их результативностью: были ли они успешными или безуспешными. Выбор продолжения атаки «почти всегда» недетерминирован, и поэтому **сценарий атаки не может быть определен заранее**.

Разработанный к настоящему времени программный прототип АСМА состоит из следующих компонентов (агентов): (1) множества агентов-хакеров, каждый из которых реализует модель атакующего, (2) агента – модели атакуемой компьютерной сети и (3) генератора фонового «нормального» трафика. Общий трафик, полученный путем интеграции потоков данных от этих компонентов, может быть входом для систем обнаружения вторжений. В процессе атаки агенты обмениваются сообщениями с целью координации своих действий. В качестве языка обмена сообщениями используется KQML, а содержание сообщения задается в терминах XML.

На рис. 4 зафиксирован процесс генерации атаки «Получение доступа к ресурсам хоста» (GAR) на этапе разведки после выполнения действия EUE («Определение записей пользователей и групп утилитой enum»). На рисунке данные о реализуемой атаке разбиты на четыре группы: (1) в левой верхней части экрана отображается элементы спецификации задачи атакующего; (2) справа от них визуализируется дерево генерации атаки; (3) в левой части экрана ниже данных о спецификации задачи размещаются строки генерируемых действий злоумышленника; (4) для каждого действия злоумышленника справа отображаются признак успеха (неуспеха) в виде зеленого (черного) квадрата и данные, полученные от атакуемого хоста (реакция хоста).

Окончание следует