Современные некоммерческие средства обнаружения атак

Д. Ю. Гамаюнов,
факультет Вычислительной Математики и Кибернетики,
МГУ им. М. В. Ломоносова,
Москва, 21 сентября 2002 г.

Аннотация

В данной статье приводится обзор и сравнительный анализ нескольких существующих некоммерческих систем обнаружения атак (СОА) на компьютерные системы и сети. Проведен сравнительный анализ 7 систем по определенному набору критериев, который разрабатывался специально для сравнения и оценки качественных характеристик СОА.

1. Введение

Последнее десятилетие ознаменовалось бурным развитием компьютерных сетей, что породило множество проблем, связанных с безопасностью информационных ресурсов. В конце 70-х – начале 80-х годов прошлого века стали появляться первые системы обнаружения атак на вычислительные системы. В 90-х годах происходил бурный рост количества исследований и действующих систем в данной области. Сегодня можно насчитать более сотни разных систем обнаружения атак, большинство из которых в настоящее время не развиваются и не поддерживаются.

Несмотря на высокую активность в исследовании компьютерных атак, на сегодня нет теории обнаружения атак. Нет даже общепринятого определения самого понятия атаки. Поэтому формальные методы обнаружения атак проработаны недостаточно для широкого использования в реальных системах. Методы обнаружения, используемые сегодня в коммерческих и некоммерческих СОА, можно назвать эвристическими. Они все используют некоторые априорные предположения о том, что есть атака, какое поведение объекта в сети можно считать нормальным, а какое подозрительным.

Коммерческие системы большей частью используют один и тот же эвристический метод обнаружения, основанный на экспертном подходе, когда система анализирует наблюдаемое поведение объектов в сети на основе существующей у нее базы описаний известных атак. Эти описания строятся на основе знаний экспертов в области сетевой безопасности. При этом для коммерческих систем характерны высокая начальная стоимость и высокая стоимость поддержки.

Для экспериментальных систем характерно использование в большей степени формализованных методов обнаружения атак, которые используют формальную модель атаки и пытаются приблизить процесс ее обнаружения к полной автоматизации. Используются такие разделы математики как нейронные сети, сети Петри, конечные автоматы.

В последние годы активно начали разрабатываться и использоваться некоммерческие системы обнаружения атак. Часть из них разрабатываются в университетах и «выросли» из исследовательских систем. Прочие являются разработками компаний, которые работают по распространенному сегодня принципу открытого программного обеспечения. В данной работе приводится обзор современных некоммерческих СОА. Обзор не претендует на полноту, его основная цель – дать читателю общее представление о возможностях некоммерческих СОА, помочь лучше сориентироваться на рынке коммерческих систем.

2. Исследованные системы обнаружения атак

Всего рассмотрено 7 систем обнаружения атак. В табл. 2.1. приведена краткая информация по каждой из них.

Название системы	Производитель	Ссылки
AAFID	Purdue University, West Lafayette, IN, USA	http://www.cs.purdue.edu/coast/projects/autonomous-agents.html
ASAX	University of Namur, Belgium	http://www.ja.net/CERT/Software/asax/
NetSTAT	University of California at Santa Barbara	http://www.cs.ucsb.edu/~kemm/netstat.html/
Prelude	Yoann Vandoorselaere yoann@mandrakesoft.com Laurent Oudot oudot.laurent@wanadoo.fr	http://www.prelude-ids.org/
SHADOW	Naval Surface Warfare Center, Dahlgren Division	http://www.nswc.navy.mil/ISSEC/CID
Snort	Martin Roesch	http://www.snort.org/
SnortNet	Fyodor Yaroshkin	http://snortnet.scorpions.net/

Таблица 2.1. Некоммерческие системы обнаружения компьютерных атак.

Часть рассмотренных систем (AAFID, ASAX, NetSTAT) разработаны в университетах и базируются на исследованиях в области обнаружения атак, проведенных в этих университетах.

3. Методика и критерии сравнения

Сравнительный анализ вышеназванных систем производился следующим образом: был составлен набор критериев сравнения качественных характеристик СОА, для которых определено множество возможных значений. Далее по каждому из критериев оценивалось соответствие ему каждой системы в отдельности и сравнение полученных характеристик систем в совокупности.

Для сравнительного анализа были выбраны следующие критерии:

Класс обнаруживаемых атак. Данный критерий определяет, какие классы атак способна обнаруживать рассматриваемая система. Это один из ключевых критериев, так как сегодня ни одна система не способна обнаруживать атаки всех классов. Поэтому для более полного покрытия всего спектра атак необходимо комбинировать различные СОА. Здесь мы используем классификацию атак, основанную на разделении ресурсов защищаемой системы по типам.

Класс атаки – это четверка <**L**,**R**,**A**,**D**>, где **L** – расположение атакующего объекта, **R** – атакуемый ресурс, **A** – целевое воздействие на ресурс, **D** – признак распределенного характера атаки.

L: расположение атакующего объекта. Оно может быть внутренним по отношению к защищаемой системе (***li***), либо внешним (***le***).

R: атакуемый ресурс. Ресурсы разделяются по расположению и по типу.

По расположению: локальные (***rl***), сетевые (***rn***).

По типу: пользовательские ресурсы (***ru***), системные ресурсы (***rs***), ресурсы СУБД (***rd***), вычислительные ресурсы (***rc***), ресурсы защиты (***rp***).

A: целевое воздействие на ресурс: сбор информации (***as***), получение прав пользователя ресурса (***au***), получение прав администратора ресурса (***ar***), нарушение целостности ресурса (***ai***), нарушение работоспособности ресурса (***ad***).

D: признак распределенного характера атаки: распределенные (***dd***), нераспределенные (***dn***).

Следующий критерий характеризует источники и способы сбора информации о поведении объектов и состоянии ресурсов:

Уровень наблюдения за системой. Определяет, на каком уровне защищаемой системы собирают данные для обнаружения атаки. Различаются системные и сетевые источники. В

пределах системных источников разделяются уровни ядра, процесса и приложения. От уровня наблюдения за системой зависит скорость сбора информации, влияние системы на собираемую информацию, вероятность получения искаженной информации.

В одном случае СОА может иметь собственные наблюдающие модули, которые в реальном времени получают информацию из первичных источников – из канала передачи данных в сети, системной шины узла сети, ядра ОС узла, от встроенных в приложения модулей.

В другом случае СОА может использовать лишь вторичные источники информации – журналы регистрации ОС узла, приложений, сетевые данные, собранные другими средствами (на других узлах). В этом случае вероятность получение искаженных данных выше и заведомо меньше скорость обнаружения возможной атаки.

Следующий критерий определяет эффективность обнаружения атаки на основе анализа полученной информации.

Используемый метод обнаружения. Метод обнаружения также является ключевым критерием сравнения. Методы подразделяются на формальные и эвристические. Формальные методы базируются на механизмах определения текущего состояния ресурсов системы и оценки опасности этих состояний. В этих методах ля каждого состояния одного ресурса должна быть однозначно определена мера «безопасности». Существующие системы используют эвристические методы обнаружения, которые обнаруживают и классифицируют атаки на основе неполной информации и с некоторой долей вероятности неправильного обнаружения и классификации.

Адаптивность к неизвестным атакам. Определяет, позволяет ли используемый метод обнаруживать ранее неизвестные атаки. Экспертные методы требуют постоянного обновления базы знаний об атаках описаниями новых атак, которые производятся экспертами. Пока не построено эффективного представления базы знаний об атаках, которое позволило бы получать описания возможных новых атак на основе описания известных атак. Методы обнаружения аномального поведения объектов позволяют обнаруживать неизвестные атаки, но они также имеют много слабых сторон – неустойчивость к малым изменениям поведения объектов, низкая точность классификации атаки.

Следующие три критерия определяют такие архитектурные особенности СОА как управление, распределенность,

Управление. Определяет организацию управления системой. Управление может быть централизованное, распределенное. Дополнительно может присутствовать возможность удаленного управления СОА. Сюда включаются задачи установки, настройки и администрирования системы. При полностью распределенном управлении необходимо управлять всеми компонентами СОА в отдельности. При полностью централизованном управлении все компоненты СОА могут управляться с одного узла.

Архитектура. Определяет распределенность архитектуры СОА. Архитектура может быть нераспределенной или распределенной. Данная характеристика определяет скорость реакции СОА на атаку. При нераспределенной архитектуре СОА может собирать и анализировать информацию о поведении объектов в некотором ограниченном сегменте защищаемой системы. Она не может, в общем случае, оценить безопасность состояния всей защищаемой системы. Таким образом, распределенная архитектура СОА позволяет расширить горизонт наблюдения за защищаемой системой, который определяет время оценки защищенности системы и обнаружения атаки.

Расширяемость. Определяет, насколько система является открытой для интеграции в нее компонентов сторонних разработчиков и для сопряжения ее с другими системами защиты информации. Это могут быть программные интерфейсы для встраивания дополнительных модулей, реализация стандартов взаимодействия сетевых компонентов.

Формирование ответной реакции на атаку. Определяет наличие в системе встроенных механизмов ответной реакции на атаку, кроме самого факта ее регистрации. Возможны такие способы реагирования как разрыв соединения с атакующим объектом, блокировка его на межсетевом экране, отслеживание пути проникновения атакующего объекта в защищаемую системы.

Защищенность. Определяет степень защищенности СОА от атак на ее компоненты, включая защиту передаваемой информации, устойчивость к частичному выходу компонентов из строя или их компрометации. Затрагиваются вопросы наличия уязвимостей в компонентах СОА, защищенности каналов передачи данных между ними, авторизации компонентов внутри СОА.

Данный обзор строился, по большей части, на основе изучения описаний систем и их документации. Поэтому не изучены такие критерии сравнения как точность и полнота систем

обнаружения атак, так как их изучение требует проведения большой экспериментальной работы.

4. Результаты

В данном разделе приводятся результаты сравнения рассмотренных семи СОА. Системы сначала сравниваются отдельно по каждому из критериев, а в конце раздела приводится сводная таблица сравнения по всем критериям.

4.1. Класс обнаруживаемых атак

Все исследованные системы могут обнаруживать атаки нескольких классов. Поэтому для улучшения читаемости и сокращения объема текста вводятся понятия объединения, пересечения и вложения классов атак. Для обозначения объединения и пересечения классов атак будем использовать символы "$\cup$" и "$\cap$" соответственно.

Системы, рассмотренные в данной работе, предназначены для обнаружения атак разных классов. Часть систем ориентированы на обнаружения локальных атак и при этом используют для анализа такие источники как журналы регистрации приложений, ОС, журналы систем аудита (AAFID, ASAX). Другие системы обнаруживают только внешние (сетевые) атаки и используют для анализа информацию, получаемую из каналов передачи данных в сети (SHADOW, Snort, SnortNet). Остальные системы являются гибридными и обнаруживают как локальные атаки, так и внешние атаки (NetSTAT, Prelude).

В пределах атак тех классов, на которые они ориентированы, все системы имеют различную полноту обнаружения, т.е. потенциальное число обнаруживаемых атак (отношение числа обнаруженных атак к числу проведенных атак). Это зависит от используемого метода обнаружения и от полноты базы описаний известных атак (если она есть).

Система AAFID является наименее развитой в этом плане, так как все еще не вышла за рамки экспериментальной системы. Набор обнаруживаемых атак для нее довольно беден. Система имеет набор агентов, обнаруживающих аномалии для конкретных групп ресурсов. Она обнаруживает следующие аномалии:

- аномалии загрузки процессора;
- аномалии входа в систему;
- аномалии прав доступа к системным файлам;
- аномалии конфигурации NFS и др.

Система обнаруживает атаки следующих классов: (L,R,A,D). Где:

- L={***li*** } (атакующие объекты внутренние для системы);
- R={***rl***} $\cap$ {***ru*** $\cup$ ***rs*** $\cup$ ***rc***} (локальные пользовательские, системные ресурсы и вычислительные ресурсы);
- A={***as*** $\cup$ ***au*** $\cup$ ***ar*** $\cup$ ***ad***} (сбор информации о системе, попытки получения прав пользователя, попытки получения прав администратора, нарушение работоспособности ресурса);
- D={***dn***} (нераспределенные).

Система ASAX единственная из рассмотренных здесь является изначально ориентированной на обнаружения атак уровня системы (локальных). Она появилась раньше всех из рассмотренных здесь систем, и последняя версия предназначена, в частности, для анализа журналов регистрации системы аудита Solaris BSM. ASAX имеет набор правил на языке RUSSEL для обнаружения определенного набора атак для некоторых платформ. Эти правила включают в себя обнаружение несанкционированного доступа к системным файлам, сигнатуры известных атак на почтовые сервисы, обнаружение нарушения прав доступа к программам suid и пр. ASAX более эффективен, чем AAFID, так как долго использовался в целевых ОС и, соответственно, имеет более полную базу описаний известных атак.

Обнаруживаются атаки следующих классов: (L,R,A,D). Где:

- L={***li*** } (атакующие объекты внутренние для системы);
- R={***rl***} $\cap$ {***ru*** $\cup$ ***rs***} (локальные пользовательские и системные ресурсы);
- A={***au*** $\cup$ ***ar*** $\cup$ ***ad***} (попытки получения прав пользователя, попытки получения прав администратора, нарушение работоспособности ресурса);
- D={***dn***} (нераспределенные).

Система NetSTAT также пока находится в стадии экспериментальной разработки. Для компонентов, анализирующих сетевые данные, есть описания некоторого набора известных атак на типовые сервисы. Локальные системные компоненты ориентированы на анализ журналов

регистрации системы аудита Solaris BSM. Базы описаний известных атак малы, поэтому эффективность системы низка.

СОА обнаруживает атаки следующих классов: (L,R,A,D). Где:

- L={***li*** ∪ ***le***} (внутренние и внешние атаки);
- R={***rl*** ∪ ***rn*** } ∩ {***ru*** ∪ ***rs***} (атаки на локальные или сетевые пользовательские ресурсы и системные ресурсы);
- A={***as*** ∪ ***au*** ∪ ***ar*** ∪ ***ad***} (сбор информации о системе, попытки получения прав пользователя, попытки получения прав администратора и нарушение работоспособности ресурса);
- D={***dn***} (нераспределенные).

Система Prelude, как и NetSTAT, является гибридной, т.е. способна обнаруживать атаки как на уровне системы, так и на уровне сети. Данная система активно разрабатывается в настоящее время. По классам обнаружения сетевых атак Prelude превосходит систему Snort, так как способна использовать базу описаний атак этой системы в дополнение к собственной. Системная часть Prelude имеет достаточно широкий набор описаний атак и использует в качестве источника информации различные журналы регистрации:

- журналы регистрации межсетевого экрана IPFW, входящего в состав ОС FreeBSD;
- журналы регистрации NetFilter ОС Linux 2.4.x;
- журналы регистрации маршрутизаторов Cisco and Zyxel;
- журналы регистрации GRSecurity;
- журналы регистрации типовых сервисов ОС UNIX.

СОА обнаруживает атаки следующих классов: (L,R,A,D). Где:

- L={***li*** ∪ ***le***} (внутренние и внешние атаки);
- R={***rl*** ∪ ***rn***} ∩ {***ru*** ∪ ***rs*** ∪ ***rp***} (атаки на локальные или сетевые пользовательские ресурсы, системные ресурсы и ресурсы защиты);
- A={***as*** ∪ ***au*** ∪ ***ar*** ∪ ***ad***} (сбор информации о системе, попытки получения прав пользователя, попытки получения прав администратора и нарушение работоспособности ресурса);
- D={***dn***} (нераспределенные).

Система SHADOW является сетевой системой обнаружения атак. Она представляет собой систему фильтров собираемых сетевых данных и обнаруживает простейшие аномалии в сети. При этом она является довольно требовательной к ресурсам и требует выделенного оборудования для каждого из своих компонентов.

Системой обнаруживаются атаки следующих классов (L,R,A,D):

- L={***li*** ∪ ***le***} (внутренние и внешние атаки);
- R={***rn***} ∩ {***ru*** ∪ ***rs***} (атаки на сетевые пользовательские ресурсы и системные ресурсы);
- A={***as*** ∪ ***au*** ∪ ***ar*** ∪ ***ad***} (сбор информации о системе, попытки получения прав пользователя, попытки получения прав администратора и нарушение работоспособности ресурса);
- D={***dn***} (нераспределенные).

Система Snort это наиболее популярная на сегодняшний день некоммерческая СОА. Она активно и динамично развивается, обновления базы известных атак происходят с частотой, сравнимой с коммерческими аналогами. Snort является чисто сетевой СОА и, кроме основной базы описаний атак, имеет набор подключаемых модулей для обнаружения специфических атак или реализующих альтернативные методы обнаружения.

Системой обнаруживаются атаки следующих классов (L,R,A,D):

- L=”внутренние” ∪ ”внешние”;
- R={***rl*** ∪ ***rn***} ∩ {***ru*** ∪ ***rs*** ∪ ***rp***} (атаки на локальные или сетевые пользовательские ресурсы, системные ресурсы и ресурсы защиты);
- A={***as*** ∪ ***au*** ∪ ***ar*** ∪ ***ad***} (сбор информации о системе, попытки получения прав пользователя, попытки получения прав администратора и нарушение работоспособности ресурса);
- D={***dn***} (нераспределенные).

Система SnortNet это распределенная СОА, основанная на системе Snort. В SnortNet добавляется управляющая станция и набор агентов пересылки сообщений для организации распределенной сети сенсоров Snort. Классы обнаруживаемых атак у систем SnortNet и Snort эквивалентны.

Таким образом, по критерию обнаруживаемых классов атак, наиболее эффективными и перспективными представляются системы Prelude и Snort (SnortNet).

4.2. Уровень наблюдения за системой

Из рассмотренных семи систем все работают с данными уровня приложений на системном уровне и с сетевыми данными. То есть анализируемая информация получается из вторичных источников, таких как журналы регистрации приложений, ОС, либо из сетевого канала передачи данных.

Системы AAFID и ASAX работают только с журналами регистрации приложений и операционной системы. Системы SHADOW, Snort, SnortNet анализируют только сетевые. Системы NetSTAT и Prelude анализируют как данные из локальных системных источников, так и сетевые данные.

4.3. Используемый метод обнаружения

Все рассмотренные системы используют в качестве основного метода обнаружения атак экспертный метод.

Система AAFID имеет в своем составе набор агентов, которые в явном виде используют характеристики атак, заданные человеком, реализовавшим агенты. При этом не используется никакого формального языка описания атак. Агенты представляют собой программные модули, написанные на алгоритмическом языке общего назначения, в которых жестко определены признаки тех атак, для обнаружения которых они предназначены. Такая организация базы описаний известных атак является трудно расширяемой.

Система ASAX использует язык описания сценариев атак RUSSEL, который по описательной мощности эквивалентен алгоритмическому языку C [1]. Используется архитектурно-независимый формат представления данных журналов регистрации NADF (Normalized Audit Trail Format). Добавление нового источника данных можно осуществить, добавив конвертер формата данных в NADF и добавив сценарии атак при необходимости.

Система NetSTAT использует язык описания сценариев атак STATL, особенностью которого является возможность описания сценария атаки в виде последовательности состояний атакуемого ресурса. Таким образом, эта система использует метод обнаружения, близкий к формальным методам.

Система Prelude использует отдельные базы сигнатур атак для сетевых данных и для журналов регистрации. Для анализа сетевых данных можно импортировать сигнатуры системы Snort. Также используется набор специализированных модулей для обнаружения специфических атак, таких как сканирование портов, некорректные ARP-пакеты. Специальные модули производят дефрагментацию IP, сборку TCP-потока, декодирование HTTP-запросов.

Система SHADOW использует набор фильтров на языке Perl - в сенсоры и анализаторы «зашиты» знания разработчиков системы обнаружения атак о том, какие сетевые пакеты могут быть признаками атаки. Подобно системе AAFID является трудно расширяемой.

Система Snort использует базу сигнатур известных атак. Также используется набор специализированных модулей для обнаружения специфических атак, таких как сканирование портов, отправка большого числа фрагментированных пакетов. Специальные модули производят дефрагментацию IP, декодирование HTTP-запросов. В 2001 году появился модуль статистического анализа потока сетевых данных, но его эффективность неизвестна.

Система SnortNet эквивалентна системе Snort по методам обнаружения.

По данному критерию наиболее перспективными представляются системы с открытой архитектурой, позволяющие встраивать альтернативные методы обнаружения атак. Это системы Prelude, Snort (SnortNet) и NetSTAT.

4.4. Адаптивность к неизвестным атакам

На данный момент эта возможность отсутствует у большинства рассмотренных СОА. Возможно использование экспериментального модуля статистического анализа системы Snort, но его эффективность не изучена. Таким образом, по данному критерию также справедлив предыдущий вывод, наиболее перспективны системы Prelude, Snort (SnortNet) и NetSTAT, которые позволяют встраивать альтернативные методы обнаружения атак.

4.5. Управление

Для некоммерческих систем характерна слабая проработка вопросов удобства пользования и администрирования. Поэтому большинство систем управляются локально с тех узлов, на которых установлены их компоненты. Некоторые системы имеют подсистемы управления удаленно через web-интерфейс (SHADOW, Prelude), но возможности этих интерфейсов ограничены.

Система AAFID управляется централизованно с основного монитора системы (пользовательского интерфейса). При этом основной монитор может управлять вторичными мониторами. Подробнее об архитектуре системы см. раздел 6.

Система ASAX управляется централизованно на том узле, где она установлена, при помощи файлов конфигурации.

Система NetSTAT управляется распределенно через файлы конфигурации на всех узлах, где расположены компоненты системы.

Система Prelude управляется централизованно при помощи управляющей консоли. Компоненты системы сами предоставляют управляющей консоли те параметры их функционирования, которые могут изменяться. Также управление может осуществляться через локальные конфигурационные файлы на тех узлах, где установлены компоненты COA.

Система SHADOW управляется распределенно через файлы конфигурации на всех узлах, где расположены компоненты системы.

Система Snort управляется централизованно через файлы конфигурации, консольные команды и сигналы UNIX.

Система SnortNet управляется распределенно через файлы конфигурации на всех узлах, где расположены сенсоры Snort. При этом используется централизованная станция мониторинга.

4.6. Архитектура

Системы ASAX и Snort являются нераспределенными, остальные системы имеют распределенную архитектуру. Детальное описание архитектуры каждой из систем приводится в разделе 6.

4.7. Расширяемость

Все рассмотренные системы, кроме SHADOW, являются расширяемыми за счет добавления новых модулей со стандартизированным интерфейсом, а также за счет использования стандартных протоколов обмена сообщениями.

AAFID имеет открытый интерфейс для добавления новых агентов и фильтров.

ASAX имеет открытый интерфейс для добавления новых источников информации.

NetSTAT имеет открытый интерфейс для добавления новых агентов и фильтров.

Prelude имеет открытый интерфейс для добавления новых модулей анализа и реагирования, ведения журналов регистрации. Обмен сообщениями происходит по стандарту IDMEF (Intrusion Detection Message Exchange Format), оптимизированному для высокоскоростной обработки.

Snort имеет открытый интерфейс для добавления новых модулей анализа, имеется модуль, реализующий протокол SNMPv2.

SnortNet аналогична системе Snort, а также использует стандарт обмена сообщениями IAP (Internet Alert Protocol). В дальнейшем планируется реализовать обмен сообщениями в формате IDMEF.

Наиболее расширяемыми являются системы Prelude и Snort.

4.8. Формирование ответной реакции на атаку

Встроенную возможность реагирования на атаку имеют системы NetSTAT, Prelude и Snort. В системе NetSTAT это реализовано лишь в тестовом варианте. Система Prelude имеет набор агентов ответной реакции, которые могут блокировать атакующего при помощи межсетевого экрана. Ведутся работы по агентам, способным изолировать атакующего, либо уменьшить пропускную способность его канала. Система Snort имеет встроенную ограниченную возможность реагирования на атаку путем отправки TCP-пакетов, разрывающих соединение (с установленным флагом RST), а также ICMP-пакетов, сообщающих атакующему узлу о недоступности узла, сети или сервиса.

4.9. Защищенность

Все системы, которые пересылают какие-либо данные, используют для этого защищенные каналы. AAFID неустойчива к возможным атакам, направленным на нее саму, в силу особенностей реализации – использование алгоритмического языка Perl, который позволяет изменение программного кода на этапе выполнения. NetSTAT, Prelude, SnortNet используют библиотеку OpenSSL для шифрования канала между компонентами. SHADOW использует протокол SSH. Snort реализует протокол SNMPv2, в котором присутствуют функции шифрования паролей при передаче данных.

COA Prelude имеет дополнительные механизмы, обеспечивающие безопасность ее компонентов. В системе используется специализированная библиотека, которая делает безопасными такие библиотечные функции алгоритмического языка C функции как printf, strcpy, которые не проверяют размер передаваемых им данных. Библиотека предотвращает классические ошибки выхода за границы массивов и переполнения буферов.

Дополнительные модули анализа сетевых данных делают систему устойчивой к некорректным сетевым пакетам на разных уровнях стека и выходу ее компонентов из строя. Такие атаки как отправка пакетов с неправильными контрольными суммами, обнуленными флагами TCP, ресинхронизация сессий, случайная отправка и «обрезание» сегментов системой игнорируются.

Из рассмотренных систем вопрос безопасности наиболее проработан в системе Prelude.

4.10. Итоговая таблица по критериям сравнения

Ниже в табл. 4.1 приведены сводные результаты сравнения рассмотренных систем по выбранным критериям.

	AAFID	**ASAX**	**NetSTAT**	**Prelude**	**SHADOW**	**Snort**	**SnortNet**
Классы атак	({***li***}, {rl} ∩ {ru ∪rs ∪ rc}, {***as*** ∪ ***au*** ∪ ***ar*** ∪ ***ad***}, {***dn***})	({***li*** }, {***rl***} ∩ {***ru*** ∪ ***rs***}, {***au*** ∪ ***ar*** ∪ ***ad***}, {***dn***})	({***li*** ∪ ***le***}, {***rl*** ∪ *rn)* ∩ (***ru*** ∪ ***rs***}, {***as*** ∪ ***au*** ∪ ***ar*** ∪ ***ad***}, {***dn***})	({***li*** ∪ ***le***}, {***rl*** ∪ *rn)* ∩ {***ru*** ∪ ***rs*** ∪***rp***}, {***as*** ∪ ***au*** ∪ ***ar*** ∪ ***ad***}, {***dn***})	({***li*** ∪ ***le***}, {***rl*** ∪ *rn)* ∩ {***ru*** ∪ ***rs***}, {***as*** ∪ ***au*** ∪ ***ar*** ∪ ***ad***}, {***dn***})	({***li*** ∪ ***le***}, {***rl*** ∪ *rn)* ∩ {***ru*** ∪***rs***∪***rp***}, {***as*** ∪ ***au*** ∪ ***ar*** ∪ ***ad***}, {***dn***})	({***li*** ∪ ***le***}, {***rl*** ∪ *rn)* ∩ {***ru*** ∪***rs***∪***rp***}, {***as*** ∪ ***au*** ∪ ***ar*** ∪ ***ad***}, {***dn***})
Уровень наблюдения за системой	системный	системный	системный, сетевой	системный и сетевой	сетевой	сетевой	Сетевой
Метод обнаружения	экспертный	экспертный	экспертный, контроль переходов состояний	экспертный	экспертный	экспертный, статисти-ческий анализ	Экспертный , статисти-ческий анализ
Адаптивность	-	-	-	-	-	+	+
Управление	централизо-ванное	централизо-ванное	распреде-ленное	централизо-ванное	распреде-ленное	централизо-ванное	распреде-ленное
Архитектура	распреде-ленная	нераспреде-ленная	распреде-ленная	распреде-ленная	распреде-ленная	нераспреде-ленная	распреде-ленная
Расширяемость	програм-мный интерфейс	програм-мный интерфейс	програм-мный интерфейс	програм-мный интерфейс, IDMEF	-	програм-мный интерфейс, SNMPv2	програм-мный интерфейс, IAP
Реакция	-	-	+	+	-	+	+
Защита	-	-	SSL	SSL, Libsafe	SSH	-	SSL

Таблица 4.1. Результаты сравнения СОА.

5. Описание исследованных систем

В данном разделе приводится более детальное описание рассмотренных в данной работе СОА. Для каждой системы описывается ее архитектура, используемая платформа и некоторые индивидуальные особенности.

5.1. AAFID (Autonomous Agents for Intrusion Detection)

Система AAFID разработана в университете Purdue, , West Lafayette, IN, USA (***http://www.cs.purdue.edu/coast/projects/autonomous-agents.html***). Первые публикации по системе датированы 1998 г., последние – 1999 г. AAFID – это одновременно название распределенной архитектуры систем обнаружения атак и собственно системы обнаружения атак [3,5,6,16]. Основой системы являются *автономные агенты* обнаружения. Наиболее интересна в системе именно ее архитектура. Данная система базируется на работах Crosbie и Spafford, которые предложили использование автономных агентов, работающих на основе генетических алгоритмов и адаптирующихся к поведению пользователей. Идея использования генетических алгоритмов не была реализована, но архитектурные идеи были воплощены в системе AAFID.

Основными компонентами системы являются: *агенты*, *фильтры*, *трансиверы*, *мониторы*.

Система AAFID является полностью распределенной. На любом узле в ЛВС может быть размещено любое число агентов, наблюдающих за интересными с их точки зрения событиями на данном узле. *Агент* – автономная программная компонента системы обнаружения атак, функционирование которой зависит лишь от ОС, под управлением которой она работает. То есть функционирование агента не зависит от других компонентов системы обнаружения атак. Агенты могут использовать *фильтры* для сбора информации о поведении объектов в архитектурно-независимом представлении. Все агенты на одном узле передают собранную информацию одному *трансиверу*. Трансивер – компонент системы обнаружения, который управляет запуском и остановкой агентов на данном узле. Каждый узел, на котором есть агенты,

имеет один трансивер. Кроме того, трансиверы могут осуществлять некоторую редукцию получаемой от агентов информации в более обобщенное представление. Трансиверы передают полученную информацию одному или нескольким *мониторам*. Каждый монитор наблюдает и взаимодействует с несколькими трансиверами. Мониторы функционируют на уровне защищаемой системы в целом, поэтому они могут анализировать получаемую информацию с учетом корреляции событий в разных областях защищаемой системы. Мониторы могут располагаться в иерархическом порядке. Монитор также является связующим компонентом между системой обнаружения и пользовательским интерфейсом – он получает управляющие команды и отдает полученную и проанализированную информацию. На рис. 5.1 представлена архитектура системы обнаружения, а на рис. 5.2 – иерархия компонентов системы обнаружения.

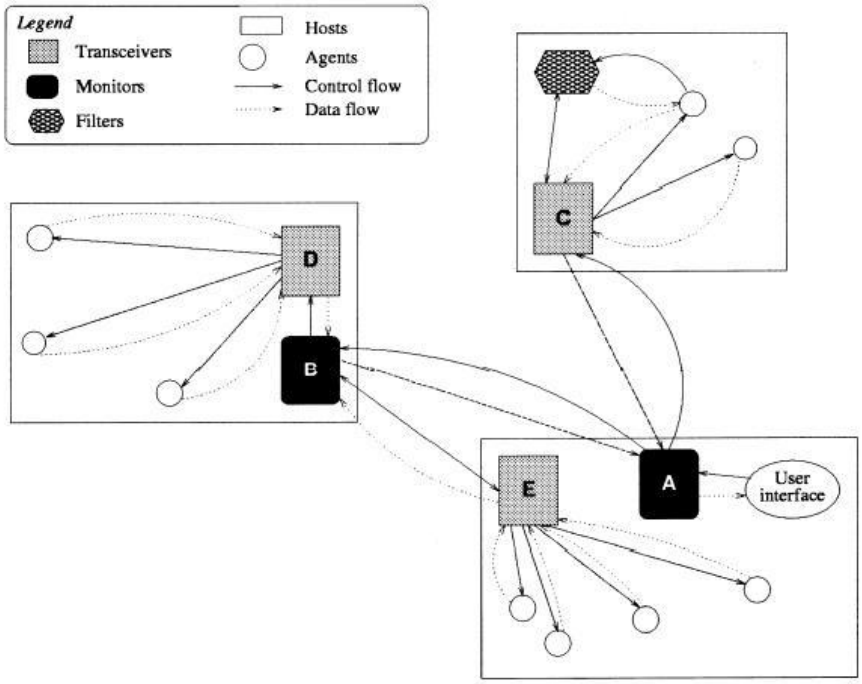

Рис. 5.1. – Архитектура системы AAFID, основные компоненты системы.

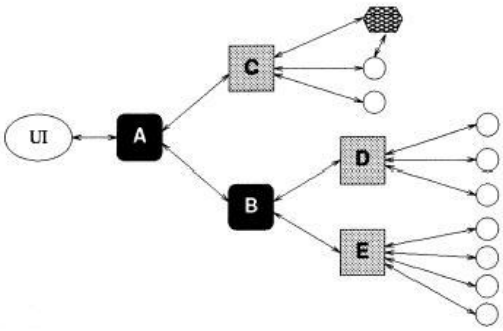

Рис. 5.2. – Иерархия компонентов системы AAFID

Остановимся более подробно на описании компонентов системы AAFID.

Агент наблюдает за конкретными аспектами функционирования узла и уведомляет соответствующий трансивер об аномальных событиях на узле. Агенты не могут непосредственно генерировать сообщения для пользователя. Таким образом, трансиверы получают полную информацию о функционировании узла, а мониторы – о сети в целом.

Функциональность агента ничем не ограничена, он может реализовывать любой метод обнаружения атак. Агенты могут быть предназначены для обнаружения аномалий и злоупотреблений любого типа. Так же способы реализации конкретного агента, включая язык программирования.

Фильтры предназначены для сбора однотипной информации из различных источников (различных системных журналов или от наблюдателей). Так же фильтры выполняют функции уровня представления модели OSI (Open System Interconnect): преобразовывают форматы данных, собираемых из аналогичных источников, но на разных архитектурах, к единому виду. Один фильтр может предоставлять данные нескольким агентам. Типичный пример применения фильтра – использование его для преобразования журналов регистрации различных версий ОС UNIX в формат, понимаемый агентами. Это снимает необходимость реализовывать различные агенты для разных платформ.

На каждый источник данных существует только один фильтр. Агенты могут *подписываться* на получение данных от фильтра.

Трансивер – интерфейс взаимодействия между узловыми компонентами системы обнаружения и сетевыми компонентами системы обнаружения. Основными функциями трансивера являются управление агентами на данном узле (запуск, останов), сбор и анализ данных от агентов, передача данных и результатов анализа мониторам или другим агентам.

Мониторы – наивысшие в иерархии компоненты системы AAFID. Функционально они похожи на трансиверы с тем отличием, что сбор и анализ информации происходит не на одном узле, а на части сети или на всей сети. Мониторы могут управлять трансиверами и другими мониторами. Мониторы имеют механизмы взаимодействия с пользовательским интерфейсом и являются точками доступа к системе обнаружения атак.

В системе AAFID практически не реализованы агенты, необходимые для эффективного обнаружения атак различных классов и основной ценностью данной системы является ее архитектура. В документации хорошо описаны и стандартизированы интерфейсы между компонентами системы обнаружения атак. Особое внимание уделяется вопросам безопасного взаимодействия компонентов распределенной системы обнаружения.

Насколько можно судить по описанию системы, в настоящее время все рабочие агенты реализованы по принципу экспертной системы, они обнаруживают заранее описанные ненормальные ситуации и события на узле и в сети.

Система все еще находится на стадии прототипа. Последнее обновление имело место в сентябре 1999 года и данная версия была реализована и протестирована под операционными системами Solaris и Linux.

Языком реализации является алгоритмический язык perl. На нем реализованы все компоненты системы.

5.2. ASAX (Advanced Security audit trail Analyzer on uniX)

Система ASAX разработана в университете Namur, Belgium. Первые публикации по системе датированы 1991 г., последние – 1998 г. Система предназначена для анализа журналов регистрации ОС UNIX [1,4,8,12].

Первоначально система ASAX была разработана под практически несовместимые ОС BS2000 и SINIX компании Siemens-Nixdorf A.G. Она представляла собой систему анализа журналов регистрации, для которых тогда не существовало единого стандарта. Поэтому задача построения системы разбивалась на две большие подзадачи: разработать унифицированный и гибкий формат журнальных записей, в которые можно было бы корректно преобразовывать исходные журналы регистрации ОС, разработать эффективный, мощный и при этом удобный язык описания и обнаружения сложных шаблонных конструкций в журнальных записях. Первую задачу решала сама компания Siemens-Nixdorf A.G., решением второй задачи занимался Университет Namur в Бельгии.

В результате исследовательской работы был разработан язык RUSSEL, в качестве модели анализа потоков данных вообще и анализа журналов безопасности в частности. В качестве единого формата журнальных записей был разработан т.н. *нормальный формат*

журнальных записей (NADF – Normalized Audit Data Format). Единственное требование к формату являлась возможность прямого преобразования записей из специфических форматов ОС в NADF.

Язык RUSSEL является императивным языком, что, как утверждают авторы, обусловлено требованиями эффективности языка. Этот язык предназначен для описания правил обработки записей в формате NADF. Каждое правило состоит из двух частей – описание *условия* срабатывания правила и *действия*. Действием могут быть вывод сообщения, вызов другого правила. Система обнаружения состоит из интерпретатора языка RUSSEL и источников информации, преобразующих записи системных журналов в записи в формате NADF. Интерпретатор получает на вход *модули* – наборы правил на языке RUSSEL и одну журнальную запись в формате NADF. Среди правил находится инициализирующее правило, которое должно применяться первым, и дальше вызываются те правила, которые стоят в блоке действия предыдущего примененного правила. На рис. 5.3 представлен модуль, состоящий из правил обнаружения попыток несанкционированного доступа (НСД) в систему на основе задания порога для числа неудачных попыток входа в систему. Для увеличения эффективности используются специальные модули, передающие записи журналов анализатору. При этом одна копия записи проходит через все используемые модули правил и возможно применение системы для обработки данных журналов регистрации в реальном времени.

```
rule Failed_login (maxtimes , duration : integer)

# This rule detects a first failed login and triggers off an accounting rule with an
# expiration time

begin

if evt='login' and res='failure' and is_unsecure (terminal)
    —> Trigger off for next Count_rule1 (maxtimes-1, timestp+duration )
fi;

Trigger off for next Failed_login ( maxtimes , duration)

end
```

```
rule Count_rule1 ( countdown , expiration : integer)

# This rule counts the subsequent failed logins,
# it remains active until its expiration time or until the countdown becomes 0

if evt='login' and res='failure'
   and is_unsecure(terminal) and timestp < expiration
   —> if countdown >1
         —> Trigger off for next Count_rule1(countdown-1, expiration) ;
            countdown =1
         —> SendMessage ("too much failed login's")
         fi ;

   timestp ≥ expiration
   —> Skip;

   true
   —> Trigger off for next Count_rule1 (countdown, expiration)

fi
```

Рис. 5.3 Обнаружение попытки НСД в систему

Правило 1 обнаруживает неудачную попытку доступа к ресурсу и вызывает правило 2. Правило 2 подсчитывает число неудачных попыток доступа и при превышении некоторой пороговой величины, сообщает о том, что обнаружен признак попытки НСД к ресурсу.

Система ASAX реализована и существует только для платформ UNIX и реализована на алгоритмическом языке C.

5.3. NetSTAT (Network-based State Transition Analysis Tool)

Система NetSTAT является развитием проекта Калифорнийского Университета в г.Санта-Барбара STAT – средство анализа состояний переходов (State Transition Analysis Tool) [7,13,17,18]. Первые публикации по системе датированы 1992 г., последние – 2001 г. Основой используемого системой метода является описание исходной защищаемой системы в виде

набора состояний ее компонентов и последующий анализ переходов из состояния в состояние в результате активных внешних воздействий.

Состояния защищаемой системы определяются при настройке и конфигурации системы обнаружения атак. Для каждого состояния определяется характеристика защищенности. Определяются *переходы* – изменения состояния защищаемой системы. В результате, атаки описываются в виде последовательностей переходов. Данный подход также является эвристическим относительно рассматриваемого в данной работе и родственен подходу, используемому в экспертных системах, базирующихся на сигнатурах атак. Описание атак в виде последовательности переходов терминах состояний призвано избежать традиционных ограничений методов, основанных на сигнатурах и описывать шаблоны для целых классов типовых атак.

Основой системы является язык STATL – расширяемый язык, предназначенный для описания шаблонов атак в терминах STAT. Базовый язык оперирует наиболее абстрактными понятиями, не зависящими от конкретной системы и ее конфигурации. Язык позволяет достраивать себя, добавляя специфичные для конкретной системы *события*. Для каждого нового события описывается его *предикат*. К примеру, для расширения языка с целью определения событий, характерных для веб-сервера Apache, необходимо определить события, описывающие появляющиеся в журналах данного приложения записи. То есть, событие будет иметь поля *host, ident, authuser, request, status* и прочие, определенные в Apache's Common Log Format. После этого требуется описать предикаты интересующих событий. Например, предикат isCGIrequest() будет возвращать *true*, если имел место вызов CGI-сценария. Описания событий и предикатов группируются в Language Extension Module (модуль расширения языка) и в последствии их можно использовать в описании сценариев атак для STAT. Поток реальных событий сравнивается со сценариями ядром STAT. Все конструкции языка STATL и его расширения транслируются в язык C++.

Основной функциональной частью системы является *ядро* STAT. Эта программная компонента оперирует абстрактными сущностями и событиями, не зависящими от конкретной системы. Она сравнивает входящий поток событий с имеющимися сценариями атак и выполняет непосредственно функцию обнаружения. Для генерации потока событий используется *источник событий* – программный компонент системы обнаружения, осуществляющий преобразование информации из системных источников, таких как журналы регистрации, в формат, пригодный для функционирования ядра STAT.

В системе также могут использоваться *модули реакции* – связанные с ядром компоненты, осуществляющие реагирование на обнаруженную атаку.

Данная система является чрезвычайно гибкой и позволяет строить масштабируемы системы обнаружения. На основе STAT строятся агенты или сенсоры системы обнаружения, все остальное – вопросы архитектуры и взаимодействия агентов в распределенной системе. Этому и посвящены все работы xSTAT, которые применяют данную технологию для конструирования систем обнаружения атак различных типов – узловые, сетевые.

На рис. 5.4 представлена архитектура системы обнаружения атак, построенной на основе STAT.

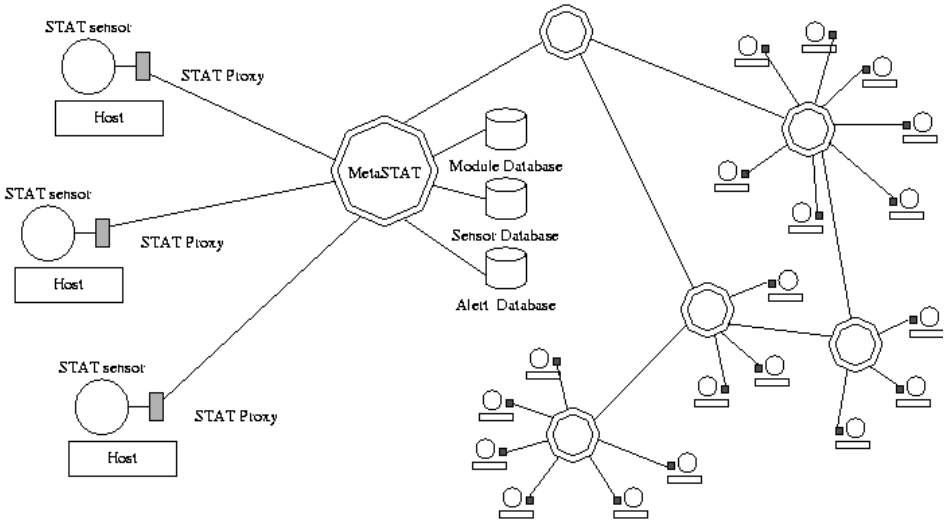

Рис. 5.4 – архитектура системы обнаружения атак NetSTAT

Основные компоненты:

STAT sensor – модуль обнаружения атак на узле на основе ядра STAT.

STAT proxy – модуль, связывающий сенсоры с центральным модулем MetaSTAT.

MetaSTAT – модуль сбора информации об атаках, уведомления администратора, хранения информации об атака.

Языком реализации системы NetSTAT является язык C++. Она работает под управлением ОС Linux и Solaris.

5.4. SHADOW (Secondary Heuristic Analysis for Defensive Online Warfare)

Система SHADOW разработана в Naval Surface Warfare Center, подразделении морской пехоты США. Система изначально разрабатывалась в открытых исходных текстах [14].

SHADOW работает под управлением ОС Linux и состоит из двух типов компонентов – сенсора сетевого трафика и анализатора. Они могут находиться на одном узле, или на разных узлах.

Сенсор основан на библиотеке libpcap и утилите tcpdump, которые широко доступны на различных платформах UNIX. Требованием для сенсора является наличие выделенной машины для его работы. Задачей сенсора – сбор и архивирование сетевого трафика по типам протоколов (задается административно при конфигурировании сенсора).

Анализатор собирает данные с сенсоров и применяет к ним специализированные фильтры, содержащие сигнатуры известных атак и сигнатуры, заданные пользователем.

Сенсоры и анализаторы должны располагаться на выделенных быстродействующих компьютерах. Наибольшие требования производительности предъявляются компьютеру, на котором работает анализатор – им должна быть машина с максимально возможным быстродействием центрального процессора (производительностью узлов определяется порог скорости передачи данных в каналах связи, до превышения которого не будет пропущен ни один сетевой пакет).

Язык реализации системы Perl.

5.5. Prelude

Система Prelude является системой с открытыми исходными текстами. Начало разработки – 1998 год. Она изначально задумывалась как гибридная СОА, которая могла бы помочь администратору сети отслеживать активность как на уровне сети, так и на уровне отдельных узлов. Система распределенная и состоит из следующих компонентов [20]:

сетевые сенсоры – различные сенсоры, анализирующие данные на уровне сети на основе экспертного анализа. Сенсоры генерируют сообщения об обнаружении аномалий и отправляют их модулям управления. Система Prelude имеет собственную базу сигнатур атак, а также может использовать сигнатуры в формате системы Snort, что позволяет эффективно обновлять базу сигнатур атак;

системные сенсоры – различные сенсоры уровня системы, анализирующие журналы регистрации ОС, приложений. Сенсоры генерируют сообщения об обнаружении аномалий и отправляют их модулям управления. Существующий набор сенсоров позволяет анализировать данные журналов регистрации таких систем и приложений как межсетевй экран IPFW, входящий в состав ОС FreeBSD, NetFilter ОС Linux 2.4.x, маршрутизаторы Cisco и Zyxel, GRSecurity; типовые сервисы ОС UNIX.;

модули управления – процессы, которые получают и обрабатывают сообщения сенсоров. Различаются следующие виды модулей управления:

- модули журнализации – отвечают за регистрацию сообщений в журналах регистрации или базах данных. В настоящее время реализованы модули для MySQL, PostgreSQL;
- модули реагирования – анализируют сообщение и генерируют возможную ответную реакцию СОА на атаку. Возможны такие виды реакции как блокирование нарушителя на межсетевом экране (NetFilter, IPFilter). В дальнейшем возможны такие типы реакции как изоляция нарушителя и сужение пропускной способности канала нарушителя.

агенты реагирования – реализуют сгенерированную менеджером реакцию на атаку;

интерфейс – основан на протоколе http. Предоставляет возможность получать статистику и управлять системой при помощи web-браузера.

На рис. 5.5. представлена логическая схема соединения компонентов СОА Prelude.

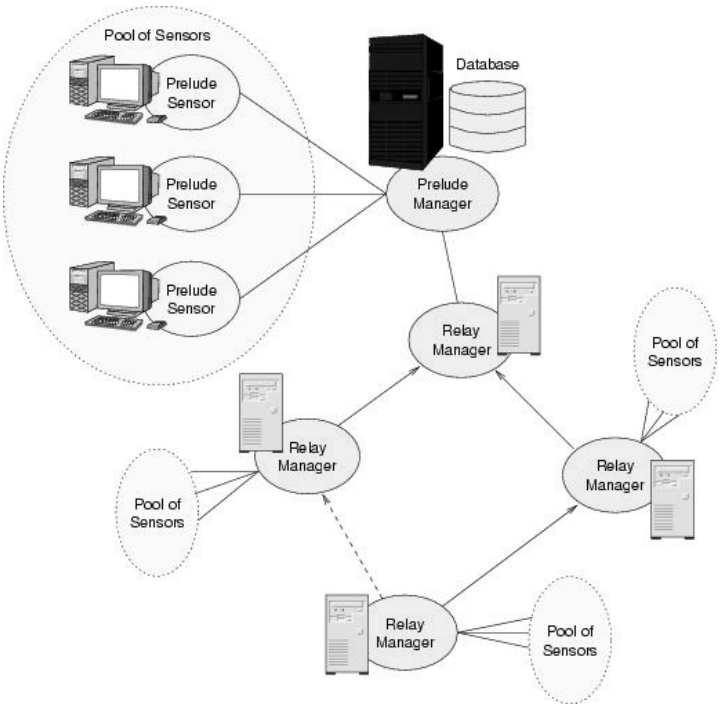

Рис. 5.5. Логическая структура СОА Prelude.

У системы Prelude есть несколько особенностей, которые отличают ее от других современных некоммерческих СОА. Система везде, где возможно, построена на использовании открытых стандартов. Так, для обмена сообщениями используется формат IDMEF (Intrusion Detection Message Exchange Format), оптимизированный для высокоскоростной обработки. Это позволяет в дальнейшем интегрировать компоненты в системы сторонних производителей и наоборот.

При разработке системы особое внимание было уделено вопросам безопасности. Каналы передачи данных шифруются по протоколу SSL, кроме того, используется специализированная библиотека, которая предотвращает классические ошибки выхода за границы массивов и переполнения буферов.

Дополнительные модули анализа сетевых данных делают систему устойчивой к некорректным сетевым пакетам на разных уровнях стека и выходу ее компонентов из строя. Такие атаки как отправка пакетов с неправильными контрольными суммами, обнуленными флагами TCP, ресинхронизация сессий, случайная отправка и «обрезание» сегментов системой игнорируются и они не приводят к отказу компонентов СОА.

Из всех рассмотренных в данной работе систем, система Prelude имеет меньше всего недостатков как в архитектуре, так и в реализации. Хотя она и не является такой популярной как COA Snort, но тем не менее имеет некоторые преимущества, такие как масштабируемость и расширяемость за счет распределенности.

5.6. Snort

Система Snort является классическим продуктом с открытыми исходными текстами [15]. Разработку начал один автор, но за счет открытой архитектуры и исходных текстов, система стала довольно быстро развиваться благодаря помощи других разработчиков, и, кроме того, интегрироваться с прочими программными продуктами, такими как базы данных для ведения журналов обнаружения, анализаторы журналов регистрации.

Snort называется автором *сетевой системой обнаружения атак*. Это прежде всего означает, что система обнаруживает атаки исключительно на основе анализа сетевого трафика. Основным методом обнаружения атак, используемым в системе, является обнаружение злоупотреблений на основе описания сигнатур атак. В системе используется простой язык описания сигнатур атак, который полностью описан в документации и позволяет администраторам системы дополнять базу сигнатур своими сигнатурами. Каждое правило на этом языке состоит из двух частей: условие применения и действие.

Пример правила системы Snort:

alert tcp any any -> 10.1.1.0/24 80 (content: "/cgi-bin/phf"; msg: "PHF probe!";)

Это правило определяет, что любой сегмент TCP, направленный на порт 80 на любой адрес в сети 10.1.1.0/24, и при этом имеющий в поле данных строку “/cgi-bin/phf”, является подозрительным и должно быть послано уведомление администратору.

Кроме того, в последних версиях системы появилась специальная конструкция языка сигнатур, позволяющая классифицировать сетевой трафик по степени потенциальной опасности. Степень опасности определяется экспертом, который формирует сигнатуру атаки.

В настоящее время система находится в стадии активной разработки, и каждые несколько месяцев появляются новые версии системы и новые функции.

Архитектура системы Snort целиком разрабатывалась из соображений эффективности и скорости работы. Поэтому она предельно проста и состоит из трех подсистем: декодер пакетов, ядро обнаружения и подсистемы оповещения и реагирования. Декодер пакетов реализует набор процедур для последовательной декомпозиции пакетов в соответствии с уровнями сетевого стека, то есть принятый кадр последовательно преобразуется в пакет, сегмент и блок данных с применением специфичных для данного уровня сигнатур атак. В настоящее время поддерживаются протоколы канального уровня Ethernet, SLIP, PPP, ожидается поддержка ATM. Ядро выстраивает имеющиеся правила в т.н. *цепи правил* – двумерные последовательности правил, где правила с общей частью условий применения объединяются в одно звено цепи, а несовпадающие компоненты правил строятся цепью во втором измерении от полученного звена. Это сделано для ускорения анализа сетевого трафика. Каждый пакет проходит по цепочке от корня, и первое подходящее правило выполняет свой блок действий и проход завершается. На рис. 5.6. представлен небольшой пример такой цепи правил.

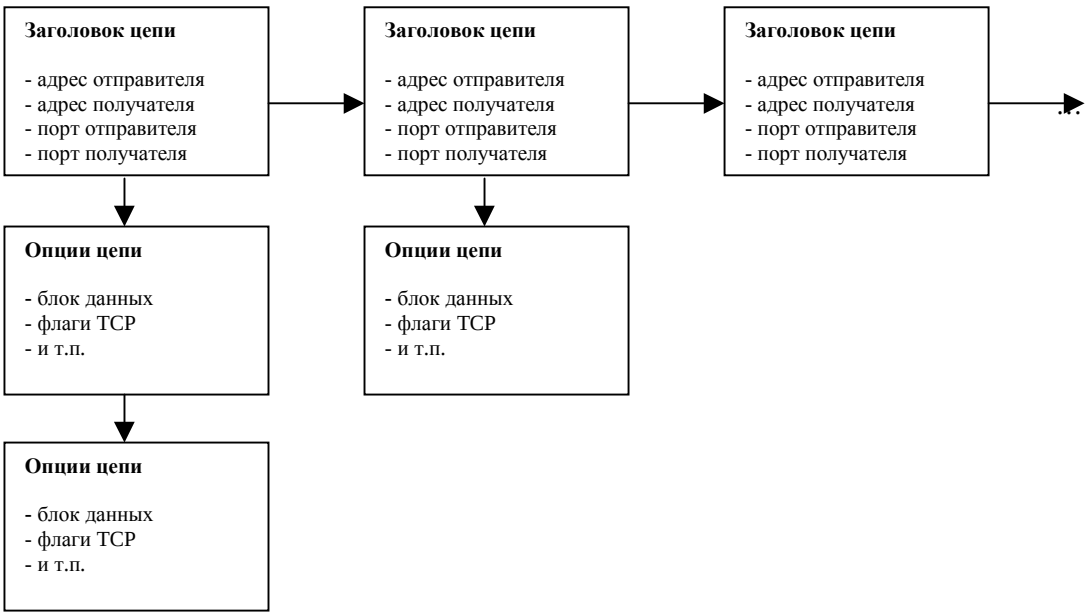

Рис. 5.6 Цепи правил системы Snort

Кроме модуля анализа трафика на основе правил, к ядру обнаружения могут подключаться модули сторонних разработчиков и производить анализ на одном из уровней декомпозиции пакета. С помощью таких модулей можно добавлять функциональности ядру обнаружения атак и реализовывать различные методы обнаружения. В поставку системы входит несколько модулей, например, модуль обнаружения сканирования портов, модуль обнаружения Unicode-атаки на веб-сервер компании Microsoft. Кроме того, в последних версиях появился модуль статистического анализа, который предназначен для обнаружения аномалий в сетевом трафике.

Подсистема оповещения и реагирования отвечает за сохранение результатов анализа трафика в журналы регистрации самой системы Snort, либо вывод этой информации через системные службы регистрации событий ОС. Например, в UNIX-подобной ОС это может быть сервис регистрации событий *syslog*.

Система Snort имеет реализации под множество UNIX платформ, а так же под ОС компании Microsoft.

5.7. SnortNet

SnortNet это распределенное расширение системы Snort, целью которого является придание ей дополнительных возможностей по масшабируемости и расширяемости [21].

Система состоит из нескольких программных модулей: сенсоров, модулей пересылки сообщений и станции мониторинга. Система позволяет осуществлять мониторинг сетевого трафика и осуществлять информирование станции мониторинга обо всех обнаруженных аномалиях в поведении сетевых объектов. Система использует Snort в качестве сетевого сенсора. В качестве протокола обмена данными система использует протокол Internet Alert Protocol (Протокол передачи сигналов тревоги – IAP). Для шифрования каналов передачи данных, аутентификации и контроля доступа система использует библиотеки SSL и TCP wrappers.

Обнаружение атак производится сенсорами Snort , после чего сообщения отправляются по протоколу IAP через модули пересылки сообщений (proxy) на станцию мониторинга. Модули пересылки сообщений и станцию мониторинга рекомендуется располагать в демилитаризованной зоне сети (см. рис 5.7).

Система SnortNet протестирована под операционными системами Linux, FreeBSD, OpenBSD.

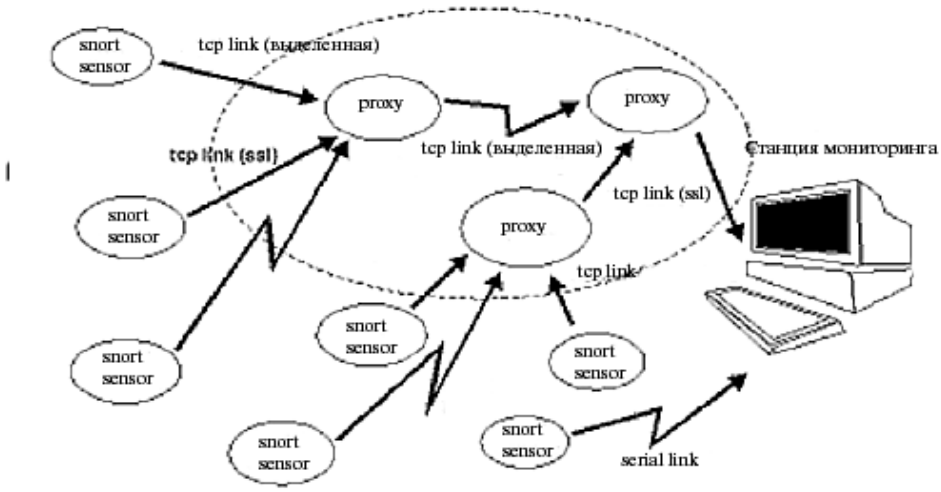

Рис. 5.7. Конфигурация системы сенсоров SnortNet.

6. Заключение

Роль систем обнаружения атак для обеспечения информационной безопасности неуклонно возрастает. Стоимость систем и затраты на их эксплуатацию для коммерческих СОА очень велика. Учитывая рост популярности открытого программного обеспечения и его существенно более низкую, по сравнению с коммерческим программным обеспечением, стоимость, представляется, что роль некоммерческих СОА будет расти еще быстрее.

Рассмотренные в данной работе СОА сильно отличаются друг от друга по качественным характеристикам. Сравнение количественных характеристик может послужить предметом отдельного исследования.

По ключевым критериям сравнения систем можно сделать следующие выводы:

Классы обнаруживаемых атак. Часть из рассмотренных систем ориентированы на обнаружение исключительно сетевых атак, часть – на обнаружение системных атак. Две системы являются гибридными. Поэтому при выборе СОА для внедрения в систему можно либо выбирать несколько наиболее эффективных специализированных сетевых и системных СОА и устанавливать их в дополнение друг к другу, либо брать одну гибридную систему. Первый вариант может обеспечить высокую начальную эффективность обнаружения атак, но при этом дороже в поддержке, так как придется одновременно поддерживать и внедрять две или более несовместимых или слабо совместимых системы. Второй вариант может иметь меньшую начальную эффективность, но будет дешевле в эксплуатации. Таким образом, использование гибридных СОА представляется более предпочтительным. Наиболее эффективны в этом смысле системы Prelude, Snort, SnortNet.

Методы обнаружения. Наиболее распространенные сегодня методы обнаружения атак не позволяют системе адаптироваться к новым атакам и обнаружение атак начинается лишь после явного внесения их описания в базу знаний системы. Такие системы требуют постоянной поддержки. Более сложные методы пока используются не очень широко, но их роль неизбежно будет возрастать.

Архитектура. Распределенные СОА сложнее в установке и администрировании, но они сильно выигрывают у нераспределенных СОА в масштабируемости и расширяемости. Для крупных сетей распределенные СОА более предпочтительны.

Открытая архитектура необходима также для сопряжения СОА с другими средствами защиты информации в сетях. Одним из слабых мест коммерческих СОА является поддержка стандартов в области обнаружения атак. Открытая архитектура и поддержка открытых стандартов позволит некоммерческим системам занять достойное место на рынке. Наиболее эффективнор проработана архитектура в системах AAFID, NetSTAT, Prelude.

Защита. Собственная защита СОА часто является довольно слабой. Некоторые коммерческие СОА требуют выделения отдельной технологической сети для связи компонентов. Устойчивость СОА к атакам является одним из ключевых вопросов обеспечения безопасности в сети, частью системы защиты которой является эта СОА.

Из рассмотренных 7 систем наиболее полно удовлетворяют всем критериям следующие 3 системы: Prelude, Snort, SnortNet. При этом система Prelude менее популярна, но имеет несколько важных преимуществ перед системой Snort – распределенность, масштабируемость и расширяемость.

7. Литература

1. A. Mounji, Languages and Tools for Rule-Based Distributed Intrusion Detection, PhD Thesis, Computer Science Institute, University of Namur, Belgium, Sept 1997.
2. Amoroso, Edward, G., Intrusion Detection, 1st ed., Intrusion.Net Books, Sparta, New Jersey, USA, 1999.
3. Jai Balasubramaniyan, Jose Omar Garcia-Fernandez, E. H. Spafford, Diego Zamboni, An Architecture for Intrusion Detection using Autonomous Agents, Department of Computer Sciences, Purdue University; Coast TR 98-05; 1998.
4. A. Baur & W. Weiss, “Audit Analysis Tool for Systems with High Demands Regarding Security and Access Control”, Research Report, ZFE F2 SOF 42, Siemens Nixdorf Software, München, November 1988.
5. M. Crosbie, E. Spafford, Defending a computer system using autonomous agents, Technical Report 95-022, COAST Laboratory, Department of Computer Sciences, Purdue University, West Lafayette, IN 47907-1398, March 1994.
6. M. Crosbie, E. Spafford, Defending a computer system using autonomous agents, in: Proceedings of the 18th National Information Systems Security Conference, October 1995.
7. S.T. Eckmann, G. Vigna, and R.A. Kemmerer, "STATL: An Attack Language for State-based Intrusion Detection," in Proceedings of the ACM Workshop on Intrusion Detection, Athens, Greece, 2000.
8. N. Habra, B. Le Charlier, A. Mounji & I. Mathieu, “Preliminary Report on Advanced Security Audit Trail Analysis on UniX”, Research Report 1/92, Institut d’Informatique, University of Namur, January 1992.
9. N. Habra, B. Le Charlier, A. Mounji, Advanced Security Audit Trail Analysis on uniX. Implementation Design of the NADF Evaluator. Research Report, Computer Science Institute, University of Namur, Belgium, March 1993.
10. S. Kumar, Classification and detection of computer intrusions, Ph.D. Thesis, Purdue University, West Lafayette, IN 47907, 1995.
11. Kumar, S. & Spafford, E. (1995) A Software Architecture to Support Misuse Intrusion Detection. Department of Computer Sciences, Purdue University; CSD-TR-95-009.
12. A. Mounji, B. Le Charlier, D. Zampuniéris, N. Habra, Preliminary Report on Distributed ASAX. Research Report, Computer Science Institute, University of Namur, Belgium, May 1994
13. Porras, P. A., Ilgun, K., and Kemmerer, R. A. (1995). State transition analysis: A rule-based intrusion detection approach. IEEE Transactions on Software Engineering,SE-21: 181–199.
14. Ralph, William D., SHADOW version 1.7 Installation manual, 2001, http://www.nswc.navy.mil/ISSEC/CID/Install.pdf
15. Roesch, Martin, Snort Users Manual, Snort Release: 1.8.1, 2001, http://www.snort.org/
16. *Eugene H. Spafford*, *Diego Zamboni*, Intrusion detection using autonomous agents, Computer Networks, 34(4):547-570, October 2000.
17. G. Vigna, R. Kemmerer, "NetSTAT: A Network-based Intrusion Detection Approach," in Proceedings of the 14th Annual Computer Security Application Conference, Scottsdale, Arizona, December 1998.
18. G. Vigna, R.A. Kemmerer, "NetSTAT: A Network-based Intrusion Detection System," Journal of Computer Security, 7(1), IOS Press, 1999.
19. Diego Zamboni , E. H. Spafford, *AAFID2 Users Guide*. Department of Computer Sciences; 1998.
20. Yoann Vandoorselaere, Laurent Oudot, “Prelude, an Hybrid Open Source Intrusion Detection System”, http://www.prelude-ids.org/, 2002.

21. Yaroshkin Fyodor, “SnortNet – A Ditributed Intrusion Detection System”, IVT-1/95, Kyrgyz Russian Slavic University, Bishkek, Kyrgystan, June 2000.