Российская Академия Наук
Институт системного программирования

Анализ подходов к верификации функций безопасности и мобильности

Москва 2004 г.

Список исполнителей

Исполнитель	Контактная информация	Примечания
Косачёв Александр Сергеевич, к.ф.-м.н., ведущий научный сотрудник ИСП РАН	kos@ispras.ru	Автор.
Пономаренко Вера Николаевна, к.ф.-м.н., научный сотрудник ИСП РАН	vera@ispras.ru	Автор.

История документа

№ версии	Дата	Примечания
1.0	27.12.2004	Обзор опубликован.

Аннотация.

В данном документе представлен обзор методов верификации функций безопасности и мобильности. Обзор основан на изучении более 280 работ, описывает состояние дел в области верификации функций безопасности и мобильности.

Исследование выполнено в рамках проекта по гранту Российского фонда фундаментальных исследований № 04-07-90308 «Верификация функций безопасности и мобильности протоколов IP».

Предисловие

Компьютерная безопасность и мобильность являются одними из самых актуальных областей исследования и разработок. В первую очередь это связано с тем, что всё большее расспространение получают компьютеры, и с тем, что всё большее их число получают возможность связываться для распределенного решения задач. Настоящий обзор, основанный на изучении более 280 работ, описывает состояние дел в области верификации функций безопасности и мобильности. Обзор состоит из двух частей.

Первая часть посвящена вопросам, связанным с верификацией безопасности. В начале её рассматриваются стандарты безопасности, как наборы требований к «безопасным» системам. Перечисляются свойства безопасности. Для каждого свойства безопасности рассматриваются различные его модели – как набор относящихся к проблеме аспектов явления. Как отдельный круг проблем, имеющих наиболее давнии традиции в изучении и решении, рассматриваются криптографические протоколы.

Затем рассматриваются способы формального описания моделей: спецификационные языки (универсальные и специального назначения), алгебры процессов, конечные автоматы , модальные логики, сети Петри, графическое моделирование.

Этими способами формального описания определяются методы формальной верификации свойств безопасности, описываемые в следующей главе. К ним относятся: верификация на основе конечных автоматов, проверка модели (model-checking), доказательство теорем (theorem proving), метод проверки типа (type checking). Рассматриваются и другие подходы.

Как необходимое дополнение формальной верификации рассматривается и тестирование свойств безопасности. В этой же главе описаны и наиболее характерные инструменты, поддерживающие верификацию свойств безопасности.

Вторая часть обзора рассматривает проблемы, характерные для класса распределенных мобильных систем. Структура второй части повторяет структуру первой: рассматриваются модели мобильности, способы их формального описания, верификации и тестирования.

Общий вывод обзора: работы в этой области ведутся с нарастающим темпом, но пока нельзя сказать, что существует «серебрянная пуля» – общий способ решения всего круга проблем.

Оглавление

Введение

Понятие безопасности (security) ПО появилось в мире ИТ в 70-ые годы XX века. С тех пор обеспечение безопасности стало одним из основных требований, которым должен удовлетворять программный продукт. Стремительное развитие в эти годы сетей, распределенных и мобильных систем привело к тому, что роль безопасности как средства защиты частной информации становится все более существенной, в особенности в связи с бурным развитием электронного бизнеса.

Компьютерную безопасность как область специализации довольно трудно точно определить. Даже профессионалы, работающие в области компьютерной безопасности, испытывают трудности при определении этого понятия. Отчасти причина заключается в том, что безопасность сложно описать, поскольку это понятие затрагивает большое количество сфер деятельности в компьютерных технологиях. Ближе всего, по всей вероятности, безопасность касается разработки программного обеспечения – компьютерная безопасность призвана гарантировать, что программное и аппаратное обеспечение удовлетворяет своим спецификациям и требованиям при использовании его в потенциально враждебной среде. Компьютерная безопасность, таким образом, включает вопросы спецификации, верификации, тестирования, валидации, надежности и живучести компьютерной системы. Однако безопасность охватывает гораздо больший круг проблем, касающихся проектирования операционных систем, архитектурного проектирования, информационной безопасности, анализа и предсказания рисков, организации баз данных, шифрования и кодирования, формальной модели вычислений, отказоустойчивости, проектирования интерфейсов, постановлений и политики государства, административных решений, компетентность в вопросах безопасности, а также соответствующее образование.

Поскольку термин безопасность несет на себе большую смысловую нагрузку, следует подчеркнуть, что в данной работе под этим термином понимается безопасность информации, которую должны обеспечивать программные продукты, функционирующие в распределенных сетях.

При изучении проблем обеспечения безопасности современных ИТ-продуктов необходимо принимать во внимание такие аспекты, как проектирование системы, используемые протоколы, реализацию, среду, а также поведение пользователей и администрации. Таким образом, условия безопасности невозможно обеспечить полностью при проектировании системы. Следовательно, необходима верификация реализации и периодические контрольные проверки состояния безопасности системы.

В книге “Основы информационной безопасности” В.А. Галатенко [Гал03] ратует за комплексный подход к обеспечению безопасности, сочетающий законодательные, административные, процедурные и программно-технические меры. Поскольку данная работа имеет научно-исследовательское направление, в ней не рассматриваются законодательные аспекты, связанные с правовыми актами, административные и процедурные аспекты, а также аппаратные средства обеспечения безопасности. Дается краткое описание основных стандартов, касающихся обеспечения безопасности распределенных систем, которые по градации Галатенко относятся к законодательному уровню безопасности, а также рассматриваются научные подходы обеспечения безопасности в программных продуктах.

В последние годы были приложены значительные научно-исследовательские усилия в области обеспечения безопасности систем. Главным образом, эти исследования касаются формальной верификации свойств безопасности. Основной целью при этом является разработка формальной математической модели свойств безопасности в системе, а также верификация этой модели с помощью математических доказательств.

Для обеспечения безопасности системы в течение многих лет акцент делался на обеспечение контроля доступа субъектов к объектам. Невозможность устранить утечку

информации при этом подходе постепенно привела специалистов в области обеспечения безопасности к пониманию важности исследований информационного потока в системе. Было предложено большое количество формальных моделей поведения системы и определения информационного потока на таких моделях. Наложение правил на информационные потоки дает возможность контролировать как прямые, так и косвенные передачи информации.

В настоящее время большое количество исследований фокусируется также на формальной верификации криптографических протоколов, которые являются службами поддержки свойств безопасности в распределенных системах.

Ускоряющееся внедрение мобильных устройств в бизнесс-процессы и повседневную жизнь требует внимательного изучения концепции мобильности. Очевидное преимущество переносимых компьютеров общепризнанно, поскольку они резко улучшают эффективность служащего, который, вооружившись таким компьютером с простым подключением к беспроводной сети, теперь способен оставаться на связи постоянно, независимо от его передвижения по земному шару. Появление небольших карманных компьютеров и смартфонов еще более увеличило интерес к мобильности. Достаточно мощные для выполнения основных корпоративных приложений, эти доступные переносные устройства позволяют профессионалам увеличивать продолжительность активного состояния, улучшать продуктивность и расширять связи с заказчиками. Но увеличение мобильности влечет за собой повышение рисков.

Сегодня большинство мобильных устройств поставляются со встроенной поддержкой подключения к беспроводной сети (“wireless-ready”), и организациям больше не нужно беспокоиться о сложных соединениях с корпоративными сетями. В результате, ценные данные предприятия, которые были когда-то защищены в пределах сетевого периметра, теперь становятся незащищенными, попадая на мобильные пункты назначения, на смартфоны, портативные компьютеры, tablet PCs, и карманные компьютеры – PDAs (personal digital assistants). Оставаясь незащищенными, мобильные устройства являются открытой дверью для доступа к одному из самых ценных корпоративных активов – информации.

Угрозы безопасности, введенные мобильными устройствами, вынуждают организации существенно изменить их философию о рамках обеспечения безопасности. Когда организации нуждались только в защите физически связанных компьютеров от неавторизованного доступа, они быстро добавили брандмауэры, антивирусное программное обеспечение, виртуальные частные сети (VPNs – Virtual Private Network), строгую аутентификацию (установление подлинности), чтобы защитить ценную информацию. Однако когда добавляются вычислительные устройства, которые меньше портативных компьютеров, которые работают совершенно независимо от сети, проблема обеспечения безопасности становится более сложной. Поскольку эти устройства используются за пределами управления обычной сетью, они требуют собственной стратегии безопасности.

Доля мобильных устройств в общем распределении компьютерного оснащения растет лавинообразно. Организации внедряют мобильные и беспроводные устройства быстрее, чем любую другую платформу. Гартнер (Gartner), аналитик в области индустрии, предсказывает, что к 2007 году во всем мире будет около 120 000 выходов в беспроводную сеть, что будет обеспечивать доступ к частным и общественным сетям более 200 миллионам мобильных устройств, используемых в бизнесе. Гартнер также предсказывает, что более 60% сотрудников в 2000 мировых компаниях будут иметь мобильный доступ к корпоративным приложениям, и что 40% корпоративных данных будут располагаться на переносных устройствах к 2005 году.

1 Безопасность

Разнообразие угроз, подстерегающих пользователя, работающего в сети, огромно. Часть из них является платой за использование сложных информационных технологий, уязвимых к внешним воздействиям, другая часть сопряжена с деятельностью людей. Мы будем рассматривать только угрозы данным, которые не зависят от физических условий окружающей среды. К угрозам для данных можно отнести нарушение конфиденциальности, атаки, ложную идентификацию. Наиболее распространенными атаками в сети являются: несанкционированный доступ (перехват и взлом паролей, взлом ОС, взлом приложений, взлом протокола), отказ в обслуживании (Denial of Service – DoS), атаки в стиле эпидемий (вирусы, черви), троянские кони. Список известных атак можно посмотреть, например, в [Intrus].

Программными средствами обеспечения сетевой безопасности являются криптографические протоколы, брандмауэры, частные виртуальные сети, системы обнаружения вторжений (IDSs – Intrusion Detection Systems), сканеры уязвимостей, антивирусное программное обеспечение.

Брандмауэры являются важной составляющей системы безопасности, но они неспособны полностью защитить компьютеры от нападений. Брандмауэры используют списки контроля доступа (ACLs – Access Control Lists) для инспекции сетевых пакетов, выполняя фильтрацию пакетов на основе номеров портов или IP, содержащихся в ACL. Более умные брандмауэры поддерживают информацию о каждом подключении и о состояниях его сеансов, выполняя также строгую проверку на этапах установления подключения.

Сканеры уязвимостей предназначены для автоматического сканирования сети с целью обнаружения слабых мест в сети.

Системы обнаружения вторжений (IDSs) являются сетевыми контроллерами, которые следят за сетью и сообщают о любой подозрительной деятельности в сети. Эти системы не могут обеспечить защиту сети, они всего лишь являются информаторами о возможном или реальном нападении. Они сообщают, на что направлена атака, откуда она исходит, и предлагают возможные варианты защиты. Качество IDS определяется большой вероятностью обнаружения истинных негативов (реальных атак) и низкой вероятностью ложных позитивов (ложных тревог). Самыми распространенными принципами IDS является обнаружение “неправильного” кода (например, детектор вирусов для сетевых пакетов) и обнаружение аномалий в сети. Во втором случае IDS осуществляет некоторое статистическое моделирование сети и запоминает нормальное состояние сети. Всякое отклонение от нормы рассматривается системой на предмет возможных нападений. Существуют еще сетевые сигнализации и приманки, которые являются разновидностью IDS и специально разрабатываются для сетей, подвергающихся постоянным нападениям. Часто IDSs называют системами предотвращения вторжения (IPSs – Intrusion Prevention Systems).

Частная виртуальная сеть (VPN) выполняет функцию безопасного соединения в открытой сети. Такие сети имеют два применения – они соединяют удаленные части одной корпоративной сети и осуществляют подключение мобильных пользователей. VPN обеспечивает аутентификацию и шифрование для каждого соединения между узлами или пользователями. Для шифрования применяются распространенные национальные стандарты шифрования, для аутентификации – HMAC (Hash-Message Authentication Codes) – коды аутентификации сообщений с использованием хэш-функций. Обычно безопасность частных виртуальных сетей обеспечивается криптографическими протоколами, особенно распространен протокол IPsec. На практике брандмауэры и IPSs часто используются

совместно с частными виртуальными сетями для осуществления удаленного доступа в корпоративной сети.

Важными вопросами при рассмотрении проблем безопасности являются стандарты и спецификации безопасности, выработанные международным сообществом, а также изучение свойств безопасности.

Необходимость следования некоторым стандартам в области безопасности во многих странах закреплена законодательно. Стандарты и спецификации, как правило, вырабатываются на основе накопленных знаний в предметной области. В них собраны апробированные, высококачественные решения и методологии. Кроме того, стандарты и спецификации обеспечивают совместимость аппаратного и программного обеспечения информационной системы.

В книге “Стандарты информационной безопасности” В.А. Галатенко [Гал04] выделяет две основные категории стандартов и спецификаций – оценочные стандарты безопасности и спецификации, регламентирующие различные аспекты реализации и использования средств и методов защиты. К числу оценочных автор относит стандарт Министерства обороны США “Критерии оценки удостоверяющих компьютерных систем” (TCSEC – Trusted Computer System Evaluation Criteria) [DoD85], “Гармонизированные критерии Европейских стран”, международный стандарт “Критерии оценки безопасности информационных технологий” (Common Criteria for IT Security Evaluation) [CC99] и Руководящие документы Гостехкомиссии России. Спецификации для распределенных систем вырабатываются, главным образом, “Тематической группой по технологии Internet” (Internet Engineering Task Force – IETF) и ее подразделением – рабочей группой по безопасности. Основными документами этих спецификаций являются документы по безопасности на IP-уровне (IP-sec) [Ken98]. Кроме того, специфицируется защита на транспортном уровне (Transport Layer Security – TLS [Dier99]), а также на уровне приложений (GSS-API [Lin00], Kerberos [Koh93]). В области сетевой безопасности основными документами являются спецификации X.800 “Архитектура безопасности для взаимодействия открытых систем” [X.800], X.500 “Служба директорий: обзор концепций, моделей и сервисов” и X.509 “Служба директорий: каркасы сертификаторов открытых ключей и атрибутов”.

Еще немного о терминологии. В литературе по безопасности термин “спецификация” часто заменяется термином “политика” (что система должна делать), вместо “реализации” употребляется термин “механизм” (как система должна работать), а вместо понятия “корректность” (работает ли система) часто используется слово “уверенность” (assurance). Описание пользовательских требований к безопасности называется политикой безопасности.

1.1 Стандарты безопасности

Начиная с 70-гг. службами национальной безопасности таких стран, как США и Канада, делаются значительные вложения в развитие формальных методов. Инвестиции этих служб помогают ускорить исследования и развитие некоторых аспектов формальных методов, однако специфика этих организаций послужила существенным тормозом для широкого распространения достигнутых результатов в этой области. К концу 90-х эти ограничения были значительно ослаблены, что послужило более широкому развитию исследований формальных методов для систем, требующих обеспечения безопасности.

Необходимость в получении некоторой уверенности в защите продуктов, критических по безопасности, привела к инициативе по созданию эффективного критерия безопасности в информационных технологиях и оценке относительно этого критерия продуктов, обеспечивающих коммерческую безопасность.

В США эта деятельность привела к разработке критерия TCSEC (Trusted Computer System Evaluation Criteria) [DoD85], более известного под названием “Оранжевая книга”

благодаря цвету обложки, и федерального критерия для безопасности информационных технологий (Federal Criteria for Information Technology Security). Аналогичные критерии (ITSEC) были разработаны в ряде других стран, включая Канаду, Германию, Нидерланды и Великобританию. Понимание того факта, что национальные критерии в области безопасности будут препятствовать распространению в других странах продуктов, обеспечивающих безопасность, заставило ряд стран (в частности, Канаду, Францию, Германию, Нидерланды, Великобританию и США) согласовать свои критерии и выработать международный стандарт "Общие критерии оценки безопасности информационных технологий" (Common Criteria for IT Security Evaluation, ISO/IEC 15408) [CC99].

Четырнадцать государств, ратифицировавших "Общие критерии" (Common Criteria – CC), рассчитывали, что принятие общего набора стандартов в области безопасности информационных технологий, поможет "созданию тщательно протестированных ИТ-продуктов, обладающих функциями защиты информации". Эти государства также осознавали, что "Общие критерии", далее ОК, внесут свой вклад в повышение уверенности потребителей в безопасности ИТ-продуктов, поможет повысить эффективность процесса оценки и сертификации и сделать его менее дорогостоящим.

Стандарт ОК помогает потребителям объективно оценивать защищенность ИТ-продуктов:

- Потребители могут использовать четкие и универсальные критерии ОК для оценки собственных потребностей и выбора необходимого уровня защиты.
- Потребителям становится проще определить, соответствует ли конкретный продукт их требованиям в области безопасности. Поскольку ОК предусматривает составление сертифицирующими организациями подробных отчетов о характеристиках защиты продуктов, успешно прошедших сертификацию, потребители могут использовать данные отчеты для сравнения конкурирующих ИТ-продуктов.
- Пользователи могут доверять оценкам, сделанным в ходе аттестации на соответствие стандарту ОК, поскольку оценивание производится не самим производителем, а независимой тестирующей лабораторией. На ОК все чаще ориентируются при принятии решений о покупке. Например, Министерство обороны США недавно объявило о том, что планирует в дальнейшем использовать только системы, сертифицированные по стандарту ОК.
- Поскольку ОК является международным стандартом, компании, имеющие отделения в различных странах, могут ориентироваться на его критерии при выборе продуктов, удовлетворяющих требованиям к безопасности, предъявляемым конкретным филиалом в какой-либо стране.

Наличие детализированного набора критериев позволяет ОК служить своеобразным "международным языком", понятным как производителям, так и потребителям ИТ-продуктов. Производители могут указывать, какие тесты ОК были пройдены их продуктами, при описании характеристик их защиты. Потребители также могут пользоваться данной терминологией, формулируя свои потребности в области защиты данных, что позволит производителям создавать продукты, удовлетворяющие эти потребности.

Кроме того, производители могут учитывать ОК при разработке своих продуктов с тем, чтобы было более очевидно, что тот или иной продукт соответствует определенному набору требований в области безопасности, а в процессе прохождения сертификации система его защиты может быть проверена беспристрастной сторонней организацией.

С точки зрения программиста ОК можно рассматривать как набор библиотек, с помощью которых пишутся задания по безопасности, типовые профили защиты и т.п. Следует отметить, что требования могут быть параметризованы.

ОК содержат два основных вида требований:

- функциональные, соответствующие активному аспекту защиты, предъявляемые к функциям безопасности и реализующим их механизмам (так называемый “профиль защиты” – Protection Profiles),
- требования обеспечения безопасности, соответствующие пассивному аспекту, предъявляемые к технологии и процессу разработки и эксплуатации.

Для структуризации требований, представленных в виде набора библиотек, в ОК введена иерархия класс-семейство-компонент-элемент. В ОК нет готовых классов защиты, и формирование такого класса требует определения нескольких иерархически упорядоченных профилей защиты, использующих стандартные функциональные требования и требования обеспечения безопасности.

Всего в ОК представлено 11 функциональных классов, 66 семейств, 135 компонентов. Приведем классы функциональных требований в порядке их описания в стандарте ОК:

- аудит безопасности – Class FAU: Security audit – выявление, регистрация, хранение, анализ информации, касающейся безопасности,
- связь – Class FCO: Communication – идентификация участников и гарантии их неотказуемости,
- криптографическая поддержка – Class FCS: Cryptographic support – управление ключами и криптографическими операциями,
- защита данных пользователя – Class FDP: User data protection – функции безопасности и политики функций безопасности объекта оценки, относящиеся к защите пользовательских данных,
- идентификация и аутентификация – Class FIA: Identification and authentication – выполнение гарантий, что пользователям приданы надлежащие атрибуты безопасности (подлинность, группы, роли, уровни безопасности или целостности),
- управление безопасностью – Class FMT: Security management – управление атрибутами, данными и функциями безопасности,
- приватность – Class FPR: Privacy – анонимность, псевдонимность, возможность разъединения, ненаблюдаемость,
- защита функций безопасности – Class FPT: Protection of the TSF – целостность и управление механизмами функций безопасности, целостность данных функций безопасности, где TSF – это TOE (Target of Evaluation) Security Functions,
- использование ресурсов – Class FRU: Resource utilization – отказоустойчивость, приоритет сервиса, распределение ресурсов,
- доступ к объекту оценки – Class FTA: TOE access – ограничения на выбираемые атрибуты и на количество параллельных сеансов, блокирование сеанса, история доступа, установление сеанса пользователя,
- удостоверяющий маршрут/канал – Class FTP: Trusted path TSF/channels – удостоверяющий коммуникационный маршрут между пользователями и TSF, удостоверяющий коммуникационный канал между TSF и другими IT-продуктами.

Что касается требований обеспечения безопасности, то в ОК существует 10 классов, 44 семейства, 92 компонента требований обеспечения безопасности. Вот классы требований обеспечения безопасности:

- управление конфигурацией – Class ACM: Configuration management,
- поставка и функционирование – Class ADO: Delivery and operation,
- разработка – Class ADV: Development – требования для поэтапного уточнения функций безопасности от функциональной спецификации до реализации,

- руководства по администрированию (документация) – Class AGD: Guidance documents,
- поддержка жизненного цикла – Class ALC: Life cycle support – требования к модели жизненного цикла, включая процедуры и политики устранения недостатков, корректное применение инструментов и методов, а также показателей безопасности, используемых для защиты среды разработки,
- тесты – Class ATE: Tests – требования к тестированию, которое должно демонстрировать, что функции безопасности объекта оценки удовлетворяют функциональным требованиям (покрытие, глубина, функциональные тесты, независимое тестирование),
- оценка уязвимостей – Class AVA: Vulnerability assessment – уязвимости конструкции, операций, неправильного использования, некорректной конфигурации объекта оценки (включает анализ скрытых каналов, оценку прочности функций безопасности),
- поддержка обеспечения – Class AMA: Maintenance of assurance (в основном, после оценки),
- оценка профиля защиты – Class APE: Protection Profile evaluation,
- оценка цели безопасности – Class ASE: Security Target evaluation.

По отношения к требованиям обеспечения безопасности введены оценочные уровни обеспечения (Evaluation Assurance Levels – EAL), хотя это не сделано для функциональных требований. Всего существует семь таких уровней. Оценочный уровень EAL1 определяется, как функционально протестированный. Он обеспечивает анализ функций безопасности с использованием функциональной спецификации и спецификации интерфейсов, руководящей документации, а также независимое тестирование. На этом уровне угрозы не рассматриваются как серьезные. Уровень EAL2 определяется, как структурно протестированный. Он предусматривает создание описательного проекта верхнего уровня объекта оценки, описания процедур инсталляции и поставки, руководств администратора и пользователя, функциональное и независимое тестирование, оценка прочности функций безопасности, анализ уязвимостей разработчиками. Уровень EAL3 определяется, как методически протестированный и проверенный. На уровне EAL3 осуществляется более полное, по отношению к уровню EAL2, тестирование покрытия функций безопасности, а также контроль среды разработки и управление конфигурацией объекта оценки. Уровень EAL4 определяется, как методически спроектированный, протестированный и пересмотренный. Он предполагает наличие автоматизации управления конфигурацией, полной спецификации интерфейсов, описательного проекта нижнего уровня, подмножества реализаций функций безопасности, неформальной модели политики безопасности, модели жизненного цикла, анализ валидации, независимый анализ уязвимостей. По всей вероятности, это самый высокий уровень, которого можно достичь на данном этапе развития технологии программирования с приемлемыми затратами. Уровень EAL5 определяется, как полуформально спроектированный и протестированный. Он предусматривает создание полуформальных функциональной спецификации и проекта высокого уровня с демонстрацией соответствия между ними, формальной модели политики безопасности, стандартизованной модели жизненного цикла, а также проведение анализа скрытых каналов. Уровень EAL6 определяется, как полуформально верифицированный и протестированный. На уровне EAL6 реализация должна быть представлена в структурированном виде, анализ соответствия распространяется на проект нижнего уровня, проводится строгий анализ покрытия, анализ и тестирование небезопасных состояний. И только самый высокий уровень, уровень EAL7, предполагает формальную верификацию проекта объекта оценки. Он применим к системам очень высокого риска.

Несмотря на полноту и продуманность документа ОК, нельзя пройти мимо его недостатков. Галатенко [Гал04] отмечает отсутствие в ОК объектно-ориентированного подхода и архитектурных требований. Следование библиотечному подходу сужает пространство фиксируемых знаний, усложняет их корректное использование. Функциональные требования не сгруппированы в объектные интерфейсы, к которым можно было бы применить отношение наследования, что чревато появлением слишком большого числа комбинаций несопоставимых функциональных компонентов. Отсутствие архитектурных требований является следствием избранного старомодного подхода программирования “снизу вверх”.

В соответствии с требованиями ОК, продукты определенного класса (например, операционные системы) оцениваются на соответствие ряду функциональных критериев и критериев доверия – профилей защиты. Существуют различные определения профилей защиты в отношении операционных систем, брандмауэров, смарт-карт и прочих продуктов, которые должны соответствовать определенным требованиям в области безопасности. Например, профиль защиты систем с разграничением доступа (Controlled Access Protection Profile) действует в отношении операционных систем и призван заменить старый уровень защиты C2, определявшийся в соответствии с американским стандартом TCSEC. В соответствии с оценочными уровнями доверия сертификация на более высокий уровень означает более высокую степень уверенности в том, что система защиты продукта работает правильно и эффективно, и, согласно условиям ОК, уровни 5-7 рассчитаны на тестирование продуктов, созданных с применением специализированных технологий безопасности.

Следует отметить, что большинство усилий по оценке продуктов безопасности сосредоточены на уровне 4 стандарта ОК и ниже, что говорит об ограниченном применении формальных методов в этой области. Однако внутри сообщества по проблемам безопасности растет понимание о том, как сделать применение технологии формальных методов более эффективным, чем в 70-е/80-е гг., когда инструменты с использованием формальных методов преждевременно применялись к слишком сложным задачам. Использование формальных методов внутри сообщества по проблемам безопасности в настоящее время вполне согласуется с использованием этой технологии другими сообществами, а именно, формальное моделирование применяется к продвинутому пониманию разработанных артефактов, в то время как формальный анализ применяется как к отладке формальных моделей, так и для обеспечения уверенности в согласованности (например, между политикой безопасности и спецификацией высокого уровня).

Основным органом по защите информационной безопасности в Российской Федерации является Государственная техническая комиссия при Президенте Российской Федерации (РФ) (Гостехкомиссия России), которая была создана Указом Президента РФ N 9 в январе 1992 года на базе Гостехкомиссии СССР. Гостехкомиссия России является органом государственного управления, и была создана для проведения единой технической политики и осуществления координации работ в области защиты информации. Гостехкомиссия России выпускает Руководящие документы, играющие роль российских стандартов в области. В качестве стратегического направления Гостехкомиссия России выбрала ориентацию на ОК. В 2002 году Гостехкомиссия России приняла в качестве Руководящего документа [ГТКР02] русский перевод международного стандарта ISO/IEC 15408:1999 ОК.

Спецификации IPsec [Ken98] – это стандарт IETF для обеспечения безопасности взаимодействий реального времени. Эти спецификации имеют полный набор средств обеспечения конфиденциальности и целостности на сетевом уровне. Механизмами IP-безопасности пользуются протоколы более высоких уровней, в частности управляющие протоколы, протоколы конфигурирования и маршрутизации. Протоколы IPsec обеспечивают управление доступом, целостность вне соединения, аутентификацию источника данных, защиту от воспроизведения, конфиденциальность и частичную защиту от анализа трафика. Протоколы аутентичности и конфиденциальности в IPsec не зависят от конкретных

криптографических алгоритмов. В каждой стране могут применяться свои национальные стандарты шифрования, что, однако, влечет за собой предварительное согласование набора применяемых алгоритмов для взаимодействующих сторон. Существует два режима применения протоколов аутентичности и конфиденциальности – транспортный и туннельный. В транспортном режиме защищается только содержимое пакетов и, возможно, некоторые поля заголовков. Этот режим, как правило, используется хостами. В туннельном режиме защищается весь пакет – он инкапсулируется в другой IP-пакет. Туннельный режим реализуется на специально выделенных защитных шлюзах (роль которых выполняют маршрутизаторы или межсетевые экраны).

Протокол TLS [Dier99] защищает приложения архитектуры клиент/сервер от пассивного и активного прослушивания сети и подделки сообщений. Протокол TLS имеет двухуровневую структуру. На нижнем уровне располагается протокол передачи записей (TLS Record Protocol), на верхнем уровне – протокол установления соединений (TLS Handshake Protocol). Протокол передачи записей отвечает за обеспечение конфиденциальности и целостности передаваемых данных. Протокол установления соединений позволяет серверу и клиенту провести взаимную аутентификацию, выбрать алгоритм шифрования и криптографические ключи до того, как прикладной протокол начнет передачу данных.

В документе X.800 “Архитектура безопасности для взаимодействия открытых систем” [X.800] рассматриваются важнейшие сетевые сервисы безопасности: аутентификация, управление доступом, обеспечение конфиденциальности и целостности данных, невозможность отказа от совершенных действий. Для реализации этих сервисов предусматриваются следующие сетевые механизмы безопасности: шифрование, электронная цифровая подпись, управление доступом, контроль целостности данных, аутентификация, дополнение трафика, управление маршрутизацией, нотаризация.

“Обобщенный прикладной программный интерфейс службы безопасности” (Generic Security Service Application Program Interface – GSS-API) [Lin00] предназначен для защиты взаимодействий между компонентами программных комплексов, имеющих архитектуру клиент/сервер. Он обеспечивает возможность взаимной аутентификации принципалов, контролирует целостность пересылаемых сообщений и гарантирует их конфиденциальность. Пользователями GSS-API являются коммуникационные протоколы или иные программные системы, выполняющие пересылку сообщений.

Спецификация Internet-сообщества RFC 1510 “Сетевой сервис аутентификации Kerberos” [Koh93] обеспечивает аутентификацию в разнородной распределенной среде с поддержкой единого входа в сеть. Сервер аутентификации Kerberos является удостоверяющим сервером, обеспечивает секретными ключами обращающиеся к нему субъекты и помогает им в попарной проверке подлинности. Многие операционные системы имеют клиентские компоненты Kerberos.

1.2 Свойства безопасности

При рассмотрении свойств безопасности распределенной системы удобно использовать рамочную концепцию, приведенную в [McC87] и в [JT88] и основанную на трассах. Событийная система взаимодействует со своим окружением через события. Эти события соответствуют примитивным действиям, которые совершает сама система или ее окружение. Последовательность событий, соответствующая возможной последовательности выполнения, называется трассой. События системы определяются в терминах возможных трасс.

Пусть выполнение будет последовательностью состояний некоторой системы, например, трассой прогона программы. Следуя Альперну (Alpern) и Шнейдеру (Schneider) [AS85], [AS86], [Sch99], политика безопасности (security policy) есть предикат на множестве

выполнений. Каждая система имеет множество возможных выполнений. Система удовлетворяет политике безопасности, если политика безопасности поддерживается для множества ее выполнений, система нарушает политику безопасности, если она не удовлетворяет политике безопасности. Политика безопасности позволяет характеризовать предписанное и запрещенное поведение для ненадежной системы.

Свойство безопасности (security property) также есть некий предикат на множестве выполнений, все свойства безопасности могут рассматриваться как политики безопасности. Система удовлетворяет свойству безопасности, если свойство поддерживается для множества ее выполнений, и, эквивалентно, система удовлетворяет свойству безопасности, если множество ее выполнений есть подмножество выполнений, для которых это свойство поддерживается.

В [AS85] и [Sch87] рассматривается концепция safety-liveness (надежность-живучесть) для описания свойств безопасности. Некоторые свойства безопасности являются свойствами надежности (safety), другие свойствами живучести, а некоторые нельзя отнести ни к той, ни к другой категории. Неформально, свойство надежности утверждает, что указанная "плохая вещь" не должна появиться во время выполнения: если свойство надежности поддерживается для выполнения, тогда оно поддерживается для каждого предписанного выполнения. Свойство живучести утверждает, что указанная "хорошая вещь" будет присутствовать: если свойство живучести не поддерживается для выполнения, тогда это выполнение есть префиксная операция для каждого поддерживаемого свойства. Например, "каждый захват может быть запрещен только один раз" – это свойство надежности. В то время как "все запрошенные захваты должны быть сняты" – свойство живучести. В [AS85] и [Sch87] показано, что все свойства безопасности являются конъюнкцией свойств надежности и живучести.

Маклин (McLean) в [McL94] пришел к выводу, что свойства безопасности невозможно описать в концепции safety-liveness, и предложил теорию, основанную на функциях селективного интерливинга (SIF). SIF можно использовать для описания свойств, которые являются чередованием двух трасс системы.

В работе [ZL97] описывается рамочная концепция, также названная авторами теорией, для описания свойств безопасности, которая позволяет описывать более широкие классы свойств, чем концепция Маклина.

Чтобы понять, что такое свойство безопасности, сначала рассмотрим, для чего оно служит. Цель свойства безопасности состоит в том, чтобы предотвратить возможность для пользователей низкого уровня делать выводы о событиях, которые имеют отношение к пользователям высокого уровня. Что именно низкоуровневому пользователю не должно быть позволено делать, это зависит от того, какая политика информационного потока будет поддерживаться для обеспечения данного свойства безопасности.

К свойствам безопасности ИТ-продуктов чаще всего относят следующие функции:

- **Конфиденциальность** (confidentiality, иногда secrecy) – обеспечение гарантий в том, что информация разделяется только авторизованными персонами или организациями. Конфиденциальная информация не должна быть доступной при неавторизованном доступе. Нарушения конфиденциальности могут произойти, когда данные обрабатываются неадекватным способом.
- **Целостность** (integrity) – обеспечение гарантий в том, что информация является подлинной и полной, что на информацию можно положиться. Термин “целостность” часто используется при рассмотрении информационной безопасности, поскольку она представляет один из первичных показателей безопасности. Целостность данных должна обеспечивать не только корректность данных, она должна отвечать на вопрос, можно ли доверять этим данным и

полагаться на них. Например, создание копий (скажем, рассылая файл по электронной почте) конфиденциального документа, угрожает и конфиденциальности, и целостности информации, потому что при этом появляется опасность изменения или модификации данных. Защищенная информация не должна модифицироваться через неавторизованный доступ.

- **Доступность** (availability) – обеспечение быстрого доступа к информации и ресурсам. Информация не должна удерживаться неавторизованно. Это свойство можно понимать, как возможность получить необходимую информацию за приемлемое время.
- **Аутентификация** (authentication, иногда accountability) – установление подлинности агента при входе в систему.
- **Неотказуемость** (non-repudiation) – строгое выполнение обязательств.
- **Авторизация** (authorization) – проверка прав доступа к различным ресурсам**.**

Терминология свойств безопасности, касающихся конфиденциальности, строго говоря, является несколько несогласованной. В исследованиях, посвященных политикам конфиденциальности, основанным на информационном потоке, предполагается, что вычисления с использованием конфиденциальной информации возможны, но при этом нельзя допускать утечки информации о конфиденциальных входных воздействиях через результаты вычислений. Этот вид безопасности был описан в конструктивной статье Лампсона (Lampson) [Lam73] под названием “информационное ограничение” (*confinement*), и был также известен как “секретность” (*secrecy*). Однако, и "ограничение", и "секретность" использовались для описания более слабых свойств безопасности. В контексте возможностей системы "ограничение" ссылается к способности предотвращать несанкционированную передачу возможностей (и, следовательно, полномочий). Точно так же работы в области криптографических протоколов часто опираются на модель Dolev-Yao [DY83], где секретная информация предполагается неделимой, и ее утечка возможна, только если она будет целиком вставлена в сообщение. Наконец, термин "приватность" (*privacy*) иногда используется при рассмотрении защиты конфиденциальности принципала, или как синоним анонимности. В данной работе во всех вышеперечисленных случаях будет употребляться термин конфиденциальность, как наиболее часто употребляемый в литературе.

Политики безопасности для приложений сильно зависит от сферы их применения. Так обработка данных в оборонной индустрии сильно связана со свойством конфиденциальности, обычный бизнес с целостностью и аутентификацией, телефония с доступностью. Кроме того, целостность играет большую роль в национальной безопасности, а конфиденциальность важна в коммерческих приложениях.

При создании сложных систем, состоящих из разнородных компонентов, таких как сети, встает вопрос о способности такой системы сохранять свойства, которые обеспечиваются ее компонентами. Способность сложной системы сохранять свойства своих отдельных компонент называется композиционностью (compositionality, иногда composability).

Для описания свойств безопасности был предложен ряд моделей, получивших широкое распространение.

1.3 Модели свойств безопасности

Термин модель безопасности используется для описания таких требований к системе, как конфиденциальность, доступность или целостность. В литературе по подходам к моделированию свойств безопасности чаще всего рассматриваются модели, которые моделируют только одно из вышеуказанных требований.

При рассмотрении моделей безопасности необходимо ввести понятия угрозы и атаки. Под угрозой понимается потенциальная возможность нарушения информационной безопасности. Попытка воспроизведения угрозы называется атакой, а тот, кто осуществляет такую попытку, – злоумышленником [Гал03].

Даже при беглом взгляде на литературу, касающуюся безопасности, становится ясно, что безопасность моделируется разнообразными способами в зависимости от назначения рассматриваемого приложения. На выбор свойств безопасности сильно влияют модели угроз, существующих для конкретных приложений. Прослеживаются два основных направления в моделировании свойств безопасности – это формализация безопасности, которая направлена на устранение небезопасных информационных потоков (или скрытых каналов) в программной системе, и формализация безопасности для протоколов, основанная на модели угроз Dolev-Yao [DY83]. Авторитетами первого направления являются Рикардо Фокарди (Ricardo Focardi) и Мартин Абади (Martin Abadi), второго – Катерин Мидоуз (Catherine Meadows) и Джон Миллен (Jon Millen). В настоящее время делаются попытки для поиска точек соприкосновения этих двух направлений с целью возможности использования общей рамочной концепции безопасности и для информационных потоков, и для протоколов.

Следует отметить, что семантика многих моделей, приведенных в этой секции, описывается с помощью трасс выполнения, и таким образом, является семантикой, основанной на трассах. Все ранние модели безопасности были основаны на семантике трасс, только в более позднее время стали появляться модели с другой основой для семантики.

Вообще говоря, модели безопасности дают хорошие идеи относительно того, какие политики безопасности должны применяться для обеспечения безопасности системы, но реальные реализации обычно не могут следовать модели в точности, поэтому на практике обычно применяется компромиссное решение. Тем не менее, большинство существующих систем обеспечения безопасности основано на понятиях этих моделей.

1.3.1 Конфиденциальность

Согласно Маклину (McLean) [McL90] для представления конфиденциальности используются две модели безопасности – модель контроля доступа (access control), которая определяет механизм для поддержки конфиденциальности, и интерфейсная модель, описывающая ограничения на интерфейс системы (обычно ввод/вывод), которые являются достаточными, чтобы гарантировать, что любая реализация, отвечающая этим ограничениям, будет отвечать требованиям конфиденциальности.

1.3.1.1 Модели контроля доступа

Модель контроля доступа для конфиденциальности впервые была сформулирована Лампсоном (Lampson) [Lam71], а затем уточнена в [GD72]. Структура модели описывается в терминах конечных автоматов, где каждое состояние есть триплет (S, O, M), S – множество субъектов, O – множество объектов, M – матрица доступа, в которой для каждого субъекта есть строка, для каждого объекта – столбец, и ячейка M[s, o] содержит права доступа субъекта s на объект o. Права доступа берутся из конечного множества A. Несмотря на свою простоту, эта модель имеет долгую жизнь, в основном, благодаря работе Харрисона (Harrison), Руззо (Ruzzo) и Ульмана (Ullman) [HRU76] (модель HRU), а также работе Белла (Bell) и Лападулы (LaPadula) [BL75] (модель Bell-LaPadula или BLP).

В работе [HRU76] авторы доказали две фундаментальные теоремы о сложности проблемы безопасности. Первая состоит в том, что проблема безопасности разрешима в моно-операционных системах, т.е. в системах, в которых по каждому запросу совершается только одна операция. Вторая заключается в том, что проблема безопасности неразрешима в общем случае. Эта модель имеет много усовершенствований, однако наиболее продвинутой можно считать модель TAM (Typed Access Matrix), предложенную в [San92], которая вводит

строгое типизирование в модель HRU. Интерес представляет работа [CDM01], в которой модель контроля доступа основана на состояниях системы, переходах и логическом выводе об установлении прав доступа.

Модель BLP [BL75] – знаменитая модель защиты системы с управлением информационным потоком, названная так в соответствии с именами ее авторов (Bell и LaPadula). Проблема троянских коней и вирусов привела к выделению двух типов в политике прав доступа – разграничительному контролю доступа (Discretionary Access Control – DAC), при котором пользователям разрешается передавать права доступа, которыми они обладают, другим пользователям без ограничений, и мандатному контролю доступа (Mandatory Access Control – MAC), при котором на передачу прав накладываются ограничения. Модель BLP является моделью для MAC. Так же, как и HRU, она оперирует множествами объектов и субъектов и матрицей прав доступа. Однако в ней вводится несколько упорядоченных уровней безопасности, что исключает появления в ней троянских коней, когда права пользователя без его ведома передаются другим.

Следующие два правила определяют мандатный доступ:

- **Simple-security property** (ss-property) – субъект может читать объект, только если его уровень безопасности выше или равен уровню безопасности объекта (*read-down,* чаще No Read Up).
- ***. Property** – субъект может писать в объект, только если уровень безопасности объекта выше или равен уровню безопасности этого субъекта (*write-up*, чаще No Write Down).

Для разграничительного контроля доступа DAC определяется третье правило:

- **Discretionary security property** (*ds-property*) – каждый элемент из множества текущих доступов (read, write или append) включается в матрицу доступа для соответствующей пары субъект-объект.

Состояние определяется как безопасное, если все вышеперечисленные свойства удовлетворяются.

В работе [BL75] авторы доказали теорему о безопасных системах, известную как Basic Security Theorem или BST. Теорема состоит в утверждении, что система является безопасной тогда и только тогда, когда при выполнении условий (1) начальное состояние безопасно, и (2) все переходы из одного состояние в другое безопасны, каждое последующее состояние будет также безопасно, независимо от входных воздействий. BST подверглась ожесточенной критике, в особенности со стороны Маклина (McLean), специалиста в области компьютерной безопасности. В [McL90] он описал в качестве опровержения к BST систему "System-Z", которая была безопасной в соответствии с BST, но в действительности не выполняла требований безопасности. В основном, эта система понижала все уровни безопасности каждого субъекта и объекта до самого низкого уровня. После этого модель подверглась некоторой модификации со стороны авторов. При этом авторы постарались поддержать свойство «спокойствия» (tranquility) своей модели, состоящее в том, что уровни безопасности и права доступа никогда не изменяются, но в последующей модификации модели разрешено изменять права доступа и безопасности, чтобы обеспечить большую практичность модели.

Существует и другая проблема при применении этой модели на практике, так называемые скрытые каналы (covert channels), – это передача информации, которая осуществляется не моделью. Вообще говоря, ясно, что для модели BLP очень трудно поддержать 100% безопасность. Необходимы постоянные проверки, мониторинг и модификации для того, чтобы система оставалась безопасной.

Модель RBAC (Role-based Access Control). Модель RBAC упрощает администрирование авторизацией с помощью раздачи привилегий пользователям через роли, что добавляет некий абстрактный слой между пользователями и их привилегиями. Эта модель была предложена в работе [FK92]. Ролевая модель имеет пять первичных элементов: пользователи, роли, привилегии, операции и объекты. Широкое распространение этой модели подтверждается тем, что модель RBAC утверждена как американский национальный стандарт ANSI INCITS 359-2004 (19.02.2004). Важным преимуществом RBAC систем является возможность устанавливать темпоральные ограничения на роли, такие как время и продолжительность активации роли, а также инициирование некоей роли при активации другой роли. Ролевая модель особенно подходит для Web-приложений, т.к. с ее помощью легко определять различные наборы политик контроля доступа [San96], [OSM00], [JRV01].

1.3.1.2 Noninterference – невлияние для детерминированных систем

Необходимость контролирования информационного потока в целом (как прямого, так и косвенного) привела к тому, что при рассмотрении свойств безопасности было введено понятие «невлияния» (Noninterference – NI) [GM82]. Свойство NI накладывает ограничение, при котором входные воздействия пользователя с высоким уровнем не могут перемешиваться с выходными воздействиями пользователя низкого уровня.

Основная идея NI согласно [GM82] состоит в следующем. Пусть G и G′ – две группы пользователей, и для любой последовательности γ, γ' – ее подпоследовательность, полученная из γ удалением всех действий пользователей из G. G не интерферирует с G′ тогда и только тогда, когда для каждой входной последовательности γ пользователи из G′ получают одинаковые выходы после выполнения γ и γ'. Предполагается, что модель детерминированная, т.е. каждому входному воздействию соответствует уникальное выходное воздействие.

Поскольку примитивы BLP не обладают точной семантикой, невозможно сравнивать эти две модели, как утверждает Маклин [McLean90]. Однако все-таки можно сказать, что BLP слабее, чем NI, в том, что NI запрещает многие из скрытых каналов, которые допускаются в BLP при стандартной интерпретации ее примитивов, а NI слабее BLP в том, что в NI разрешается пользователям низкого уровня копировать некий файл высокого уровня в другой файл высокого уровня, что обычно не допускается в BLP. И все-таки модель NI ближе к интуитивному понятию безопасности, чем BLP.

1.3.1.3 Модели конфиденциальности для недетерминированных систем

Модели конфиденциальности для недетерминированных систем можно рассматривать как расширения свойства NI для недетерминированных систем. Большинство этих свойств основано на семантике трасс.

Примерами являются – невыводимость (Nondeducubility) [Sut86], NI (Noninference) [OH90], обощенное NI (Generalized Noninterference) [McC87], ограничительность (Restrictiveness) [McC87], Nondeductibility on Strategies [WJ90] и Perfect Security Property [ZL97].

1.3.2 Целостность

В исследованиях, посвященных модели целостности, которая запрещает неавторизованную модификацию информации, достигнуто меньше успехов. Самая известная модель целостности была сформулирована Кларком (Clark) и Уильсоном (Wilson) [CW87], но более ранней является модель Biba [Bib77].

1.3.2.1 Модель целостности Biba

Биба (Biba) [Bib77] первым отметил, что целостность является двойственной по отношению к конфиденциальности, и она поддерживается с помощью управления информационными потоками. Конфиденциальность требует предотвращения перетекания информации в несоответствующие пункты назначения, а целостность требует предотвращения перетекания информации из несоответствующих источников. Целостность имеет важное отличие от конфиденциальности – целостность вычислительной системы может быть нарушена и без внешнего вмешательства, просто путем некорректной обработки данных. Таким образом, строгое проведение в жизнь целостности требует доказательства корректности программ.

Модель целостности Biba является модификацией модели BLP для целостности. В этой модели рассматриваются два свойства:

- **SI-property** (простое свойство целостности) – субъект может иметь доступ к объекту с правами на запись, если уровень безопасности, которым он обладает, выше или равен уровню объекта.
- **Integrity *.property**: субъект имеет доступ только на чтение к объекту *o*, тогда он также может иметь доступ на запись к другому объекту *p*, только если уровень безопасности *p* либо ниже, либо равен уровню *o*.

Сущность модели может быть определена следующим образом:

- политика доступа для чтения – то же самое, как в BLP.
- никакая информация от субъекта не может быть передана объекту, имеющему более высокий уровень безопасности. Это предотвращает загрязнение данных более высокой целостности от данных с более низкой целостностью.

Главная проблема, связанная с этой моделью, состоит в том, что нет практической модели, которая может осуществить конфиденциальность модели BLP и целостности модели Biba.

1.3.2.2 Модель целостности Clark-Wilson

Базис модели Clark-Wilson зиждется на том, что управляемые элементы данных могут быть изменены только определенными транзакциями, и что такие транзакции могут требовать сотрудничества других людей. Недостаток модели состоит в том, что она далека от формальной, и неясно, как формализовать ее с помощью обобщенных установок. Частью проблемы является то, что, при наличии ясной формальной концепции конфиденциальности, нужно все же разрабатывать формальную концепцию для целостности.

Модель Clark-Wilson вводит систему триплетов субъект/программа/объект, в которых субъект больше не имеет доступа к объекту, вместо этого эта задача теперь выполняется набором программ, которые имеют доступ к объектам. В модели Clark-Wilson рассматриваются следующие аспекты:

- аутентификация и идентификация субъекта,
- только набор программ имеет доступ к объектам,
- субъекты могут выполнить только ограниченный набор системы программ,
- правильная работа системы должна быть гарантирована.

1.3.2.3 Другие подходы

Совершенно другой подход состоит в том, чтобы двигаться от общих моделей к моделям, специфическим для приложений. Примером такой модели может служить Secure Military Message System Model [LHM84]. Ограничения, фигурирующие в этой модели – не

общие ограничения безопасности, налагаемые на субъекты и объекты, но специфические ограничения безопасности, с которыми система обмена сообщениями сталкивается при обработке сообщений. Подход, специфический для приложения, широко применяется в ряде областей, особенно в области обеспечения безопасности баз данных.

1.3.3 Доступность

Тогда как конфиденциальность запрещает неавторизованное чтение информации, доступность запрещает неавторизованное удерживание информации. Задача работоспособности состоит не в том, чтобы заботиться о невозможности для низкоуровневых пользователей чтения высокоуровневых файлов, а в том, чтобы низкоуровневые пользователи не могли мешать доступу к этим файлам высокоуровневых пользователей. В этой области было сделано значительное количество формальных работ, примерами являются [Gli83], [YG90] и [Mil92]. Последние две модели представляют распределители ресурсов. Модель [YG90] использует темпоральную логику для специфицирования ограничений на такой распределитель, а в [Mil92] используется рамочная концепция конечных автоматов.

1.3.4 Композиционность

При создании сложных систем, состоящих из разнородных компонент, таких как компьютерные сети, встает вопрос о способности системы сохранять свойства, которыми обладают ее компоненты. Свойство системы сохранять специфические свойства своих компонент называется композиционностью (compositionality), которое также часто относят к свойствам безопасности. Это свойство полезно и для верификации, и для синтеза: если свойство сохраняется, когда система компонуется, анализ можно проводить на подсистемах, и в случае успеха система в целом будет удовлетворять нужному свойству. Хорошо известно, что в целом, свойства безопасности не сохраняются при композиции [McC88]. Однако результаты композиционности являются ключевыми при разработке больших сложных систем [McL94], [McL96], [Man02]. Композиционность считают свойством второго порядка, которое сохраняет свойство первого порядка тогда и только тогда, когда комплексная система, сформированная из двух подсистем, которые сохраняют свойство первого порядка, также удовлетворяет данному свойству первого порядка. Исследования композиционности возможностных свойств информационных потоков широко представлены в литературе. Существует несколько свойств информационных потоков, основанных на семантике трасс, для которых было доказано, что они полностью композиционны. Такими свойствами являются ограничительность (restrictiveness) [McC87], опережающая исправимость (forward correctability) [JT88], сепарабельность (separability) [McL94].

1.4 Криптографические протоколы

Для безопасных распределенных систем важно, чтобы принципалы (люди, главные компьютеры, компьютеры и пр.) могли доказать свою подлинность друг другу. Механизм доказательства подлинности в сети называется аутентификацией. Как правило, чтобы гарантировать аутентификационные и связанные с ними цели, используются криптографические протоколы.

Протокол есть точно определенная последовательность шагов взаимодействия и вычисления, как определено в [WL92]. Шаг взаимодействия передает сообщения от одного принципала (отправителя) другому (получателю), в то время как шаг вычисления обновляет внутреннее состояние принципала.

Протоколы защиты (security protocols) являются одним из наиболее критических мер повышения безопасности взаимодействия и обработки информации, гарантируя ее конфиденциальность (confidentiality), целостность (integrity), подлинность (authenticity)

доступность (availability). Эти протоколы уязвимы для искусных атак, что говорит о том, что проектирование протоколов оставляет желать лучшего.

В последнее время протоколы защиты все чаще стали называть **криптографическими протоколами**. В таком протоколе для поддержки безопасности используются криптографические функции. **Протокол аутентификации** является частным видом криптографических протоколов, его первостепенная задача – дать возможность принципалам установить подлинность друг друга.

Криптография предназначена для защиты секретности сообщений, которая осуществляется с применением функций хеширования. Криптография помогает поддерживать такие функции безопасности, как аутентификация, целостность, конфиденциальность и строгое выполнение обязательств.

Криптографическими протоколами являются протоколы, которые используют криптографию для распределения ключей и установления подлинности (аутентификации) принципалов и данных в сети. Традиционными криптографическими механизмами являются шифрация и дешифрация.

Обычно предполагается, что в сети присутствуют злоумышленники, которые способны читать, модифицировать и уничтожать информацию, по всей вероятности, управляющие одним или несколькими принципалами в сети (Рис.1). Криптографический протокол обязан работать, несмотря на атаки враждебных злоумышленников. В литературе встречается несколько синонимов к слову злоумышленник (intruder). Такими синонимами являются: атакер (attacker), шпион (spy), вредитель (saboteur), пенетрейтор (penetrator), неприятель (enemy), хакер, “man-in-the-middle” (посредник).

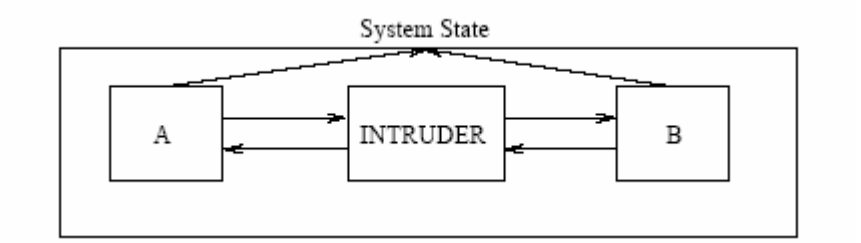

Рис. 1. Принципалы и злоумышленник в оперативной среде [GL00].

В криптографическом протоколе каждый принципал разделяет секретную информацию с некоторым удостоверяющим компьютером, называемым аутентификационным сервером. Обладая секретной информацией, принципал способен установить достоверность своей подлинности. Примером является использование паролей в многопользовательской среде. В качестве разделяемой секретной информацией в такой системе обычно служит зашифрованный ключ. Схема шифрования не позволяет пользователю, который не обладает ключом, создать или расшифровать зашифрованные данные. Аутентификация в больших распределенных системах имеет огромное значение, т.к. принципалы взаимодействуют в сети, в которой существует большая вероятность всевозможных атак. Пассивный злоумышленник может прослушивать линию. Присутствие активного злоумышленника влечет за собой трагические последствия, т.к. он способен модифицировать сообщения в сети путем блокирования передаваемых пакетов и вставки своих собственных пакетов. Такой злоумышленник способен выдавать себя за любого принципала и, возможно, перехватывать его права и привилегии. Шифрование может помешать атакам активного злоумышленника и обеспечить поддержку целостности, если любая попытка модифицировать некоторую часть информации вызывает ошибку дешифровки. Таким образом, не зная ключа, возможности злоумышленника ограничиваются блокированием данных. При обнаружении ложной информации эта информация не передается по сети.

Большинство систем с аутентификацией являются симметричными, один и тот же ключ используется как для шифрации, так и дешифрации. Предполагается, что каждый принципал разделяет секретный ключ с аутентификационным сервером, и ключ этот устанавливается с помощью некоторого безопасного автономного метода. Два принципала могут взаимодействовать безопасно, посылая зашифрованные сообщения серверу, который может перешифровать и переслать их соответствующему получателю. Однако такая схема выглядит непрактичной.

Более привлекательной видится система, в которой два принципала, желающие взаимодействовать, применяют защищенный ключ, известный только им двоим. Этот ключ обеспечивает безопасный коммуникационный канал между двумя принципалами, однако создание такого ключа, называемого сеансовым, является нетривиальной задачей.

В последнее время все большее распространение получают **асимметричные ключи**, - это пара ключей, которые являются инверсиями друг другу. Один ключ является приватным и он известен только принципалу, который им обладает. Другой ключ является открытым и он доступен всем авторизованным участникам сети. Данные, зашифрованные с помощью некоего приватного ключа, могут быть дешифрованы открытым ключом, и данные, зашифрованные с помощью открытого ключа, могут быть дешифрованы приватным ключом.

Хорошо известно, что криптографические протоколы содержат уязвимости, которые провоцируют злоумышленников производить атаки и нарушать нормальное функционирование сети. Анализ криптографических протоколов сталкивается с большими трудностями, в особенности, если эти протоколы обладают повышенными свойствами безопасности по сравнению с обычными коммуникационными протоколами. Это привело к бурному росту исследований и разработок формальных методов и инструментов для спецификации, проектирования и анализа криптографических протоколов.

Формальные методы стали использоваться при анализе криптографических протоколов в конце 70-х и с тех пор в этой области достигнуты значительные результаты, а именно, найдены существенные дефекты в нескольких широко известных криптографических протоколах. Однако, несмотря на успешные достижения в этой области, можно утверждать, что на сегодняшний день не существует известного метода, который бы обеспечивал строгое доказательство безопасности протокола.

Как правило, все методы не зависят от криптографических механизмов, и обычно рассматриваемая сеть состоит из набора принципалов и удостоверяющего аутентификационного сервера. Принципалы не являются надежными, и среди них могут присутствовать злоумышленники, способные читать, добавлять, уничтожать и модифицировать сообщения в сети.

В качестве классификации методов анализа криптографических протоколов часто используется классификация, предложенная Мидоуз (Meadows) [Mea92]:

1. Моделирование и верификация протоколов с использованием спецификационных языков и инструментов для верификации, которые не были специально разработаны для анализа безопасности. Основная идея заключается в том, чтобы истолковать криптографический протокол как любую другую программную систему и попытаться доказать ее корректность. Критика этого подхода заключается, в основном, в том, что из доказательства корректности не обязательно следует обеспечение безопасности [Sid86]. Применение этих методов и методик для анализа безопасности протоколов сталкивается с тем, что не рассматриваются некоторые тонкости, специфичные в области обеспечения безопасности, например, эффект проигрывания сообщения.
2. Разработка экспертных систем для создания и изучения различных сценариев, в которых исследуются свойства безопасности протоколов. Верификация в таких

системах стартует с некоторого нежелательного состояния, и делается попытка обнаружить, достижимо ли это состояние из некоего начального состояния. Хотя при таком подходе, возможно, имеется больше оснований найти уязвимости, чем при использовании универсальных формальных методов, все-таки он также не гарантирует безопасность протоколов и не может обеспечить автоматизированные методики для анализа атак на протокол. Экспертные системы позволяют обнаружить, действительно ли данный протокол содержит некий дефект, но с очень маленькой вероятностью смогут обнаружить неизвестные заранее уязвимости протокола.

3. Моделирование требований к протоколам с использованием логик, разработанных специально для анализа знаний и доверия.
4. Разработка формальной модели, основанной на алгебраических свойствах криптографических систем

Такой классификации придерживаются обзоры [Mea98], [But99], [RH93]. Первый подход наименее популярен, в то время как третий является наиболее общим.

Гритзалисом (Gritzalis) и Спинелисом (Spinellis) в своей работе [GSG99] предлагают иную классификацию. Согласно их классификации подходы к анализу протоколов разделяются на две большие группы:

1. методы построения логических выводов (*Inference-construction methods*), используя специализированные логики, основанные на понятиях знания и доверия о том, что участники протокола могут достичь желаемых результатов;
2. методы построения возможных атак проникновения в защищенную систему (*Attack-construction methods*) с использованием алгебраических свойств алгоритмов, лежащих в основе протоколов.

Ни одна из этих классификаций не соответствует целям настоящей работы, поэтому при изложении последующего материала иногда будут делаться только упоминания о принадлежности того или иного метода или методики к соответствующим группам вышеописанных классификаций.

Протокол Нидхама-Шредера. Первым упоминанием о формальных методах как средстве для анализа криптографических протоколов считается работа Нидхама (Needham) и Шредера (Schroeder) [NS78]. Протокол аутентификации Нидхама-Шредера [NS78] произвел коренную ломку в обеспечении безопасности в распределенных системах. Адаптации этого протокола, такие как Kerberos [SNS88] и Andrew File System [HKMNSSW88] стали универсальными.

Протокол Нидхама-Шредера распределяет ключи безопасной сессии между двумя принципалами в сети. Модель угрозы в этом протоколе предполагает, что каждый принципал разделяет безопасный ключ с аутентификационным сервером, и что злоумышленник способен читать и модифицировать все, что передается по сети. Кроме того, модель предполагает, что злоумышленники способны перехватывать сообщения и вставлять свои сообщения в пакеты.

Деннинг (Denning) и Сакко (Sacco) [DS81] обнаружили слабое место в протоколе Нидхама-Шредера, в котором предполагается, что ключи используются только один раз. Если злоумышленник записал некую предыдущую сессию протокола, то он может раскрыть защищенную информацию и использовать ключ старой сессии для атаки, выдавая себя за принципала, чьим ключом он воспользовался. Деннинг и Сакко предложили ввести временные метки для решения этой проблемы. В дополнительном сообщении Нидхам и Шредер устраняют эту уязвимость, используя уникальные идентификаторы (nonces) для каждой сессии [NS87]. Оба решения требуют начинать новую сессию каждый раз при возобновлении диалога между принципалами.

Протокол Нидхама-Шредера:

A ∅ S: A, B, Na
S ∅ A: {Na, B, Kc, {Kc, A}K(B) }K(A)
A ∅ B: {Kc, A}K(B)
B ∅ A: {Nb}Kc
A ∅ B: {Nb-1}Kc

где A, B – принципалы, S – удостоверяющий ключевой сервер; K(A) – безопасный ключ, разделяемый A и S; K(B) – безопасный ключ, разделяемый B и S;{X, Y}K означает: X конкатенируется с Y, и все это шифруется ключом K; Na, Nb – свежие нонцесы (nonces), которые прежде не использовались; Kc – свежий ключ соединения.

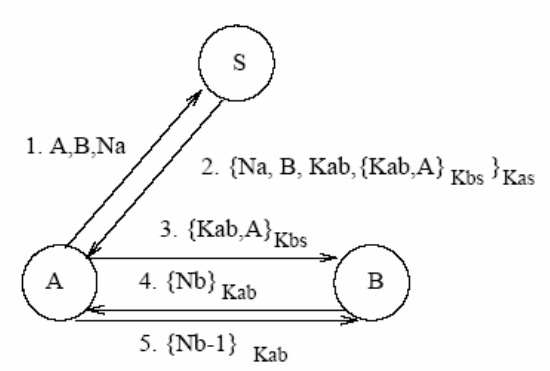

Рис.2. Протокол Нидхама-Шредера

Модель Dolev-Yao. Работа Долева (Dolev) и Яо (Yao) [DY83] является первой значительной работой в области анализа криптографических протоколов. За ней последовала работа Долева, Ивена (Even) и Карпа (Karp) [DEK82], которые в начале 80-х разработали семейство алгоритмов для анализа свойств безопасности ограниченного класса протоколов

Применение формальных методов означает введение некоторых абстракций для описания рассматриваемых свойств системы. Модель Dolev-Yao является абстракцией криптографических операций протокола защиты. Фактически эта модель явилась первой формализацией модели злоумышленника.

Однако вскоре было обнаружено, что даже небольшое ослабление ограничений на протоколы в этой модели приводит к неразрешимости проблем безопасности [EG83], и таким образом, в чистом виде эта модель не может использоваться. Работа Долева (Dolev) и Яо (Yao) является значительной в том смысле, что она была первой формальной моделью, в которой допускалось параллельное выполнение анализируемого протокола, криптографические алгоритмы рассматривались как черные ящики, подчиняющиеся ограниченному набору алгебраических свойств (например, операции кодирования и декодирования сводят друг друга на нет), и которая включала злоумышленника, способного читать, модифицировать и уничтожать информацию и, возможно, также управлять некоторыми легальными участниками системы. Надо отметить, что большинство последовавших работ по формальному анализу криптографических протоколов основано на этой модели или на ее вариантах.

На основе модели Dolev-Yao или ее вариантах были разработаны такие инструменты, как Interrogator [MCF87], NRL Protocol Analyzer [Mea92] и Longley-Rigby tool [LR92].

1.5 Методы формальной спецификации

1.5.1 Спецификационные языки

1.5.1.1 Универсальные спецификационные языки

Ina Jo. Ina Jo является языком формальной спецификации [SH88], основанный на расширении исчисления предикатов первого порядка. Этот непроцедурный язык утверждений был разработан как универсальное средство для поддержки разработки ПО и доказательства его корректности.

Кемерер (Kemmerer) в работе [Kem89] специфицирует рассматриваемую систему с помощью Ina Jo. Предполагаемые свойства системы выражаются через аксиомы Ina Jo (например, что шифрование и дешифрование коммутативны). С помощью критериев Ina Jo описываются критические требования, которым система должна удовлетворять во всех состояниях (например, что никакой ключ не должен быть доступен злоумышленнику, с помощью которого он смог бы осуществить дешифрование). На основе созданных спецификаций Ina Jo вырабатывает теоремы, которые затем используются для верификации критических требований. Кемерер находит, что преимущества описания системы в формальной нотации и последующего доказательства свойств относительно этой спецификации состоят в том, что если сгенерированные теоремы не могут быть доказаны, это часто указывает на слабые места системы или на неполноту спецификаций. Однако, значение этого метода ограничивается тем обстоятельством, что доказательство критериев о спецификациях Ina Jo не может гарантировать безопасность протокола. К тому же, чтобы специфицировать требования к безопасности системы, необходимо уметь описывать потенциальные атаки.

Подход Чена (Chen) и Гликора (Glicor) [CG90] также основан на применении Ina Jo. Авторы утверждают, что многоуровневая сетевая безопасность может быть достигнута и формально верифицирована независимо от специфических протоколов транспортного уровня, даже в присутствии злоумышленника в сети. Это достигается с помощью применения многоуровневого протокола безопасной сессии и протокола распределения ключей, которые помогают поддерживать конфиденциальность и целостность сообщений в небезопасной сети. И формальная модель политики безопасности, и формальная спецификация высокого уровня протокола групповой сессии написаны Ina Jo [SH89]. Правила вывода логики BAN [BAN89] (см. Ниже), вставленные в преобразования Ina Jo, демонстрируют корректность протокола распределения ключей.

LOTOS. LOTOS был разработан для систем, относящихся к Open Systems Interconnection (OSI) и в его основе лежат алгебры процессов. Система в LOTOS моделируется как набор процессов, в котором указывается порядок событий. LOTOS может быть использован для моделирования процесса обмена сообщениями в криптографических протоколах.

Варадхарайян (Varadharajan) [Var90] предложил использовать LOTOS для анализа криптографических протоколов. Он приводит примеры спецификации двух протоколов, принятых в качестве стандартов ISO/DP 9798 и CCITT X.509. Однако результатов никаких не дается, а делается вывод, что инструменты для LOTOS не адекватны этим задачам и оставляют желать лучшего.

В работе [Boy92] также используется подход на основе LOTOS. Более поздняя работа с использованием LOTOS сделана Ледуком (Leduc) и Джерми (Germeau) [GL00]. Они применили метод, основанный на модели, в котором протокол специфицируется с помощью алгебры процессов, основанной на CSP/CCS, и верифицируется с помощью FDR. Криптографические операции моделируются на абстрактном уровне как продукт поведения

и правил и понимаются как свойства безопасности, и их можно легко перевести в спецификацию. Каждый принципал специфицируется как сущность, которая имеет поведение и абстрактный тип данных для представления действий, сообщений и знаний. Процесс взаимодействия осуществляется через синхронизирующие шлюзы между принципалами, и злоумышленник явно помещается между двумя принципалами. Злоумышленник моделируется во время каждой синхронизации вместе со своими характеристиками (поведение, знание, старые прослеженные сообщения). Все спецификации пишутся на LOTOS, затем они транслируются в систему размеченных переходов (LTS) для верификации. Эта основанная на модели верификация является достаточно мощной для нахождения дефектов в протоколе, но с ее помощью невозможно доказать полную корректность из-за проблемы взрыва состояний.

В работе [LG00] рассматривается применение LOTOS для спецификации протоколов защиты и криптографических операций. Описывается, как свойства безопасности могут быть моделированы через свойства надежности и как метод верификации на основе модели используется для верификации устойчивости (robustness) протокола относительно атак злоумышленника.

ASTRAL [GK91] – спецификационный язык универсального назначения, который используется для спецификации систем реального времени. Созданный на его основе анализатор моделей (model-checker) ASTRAL, был применен к анализу криптографических протоколов [DK97].

О применении других спецификационных языков для анализа криптографических протоколов сообщается в следующих работах: VDM применяется в [BCLW93], [CEC93], [BC94], HOL применяется в [Sne95], Z применяется в [Boy90] и [Sne90], Coq применяется в [Bol96].

1.5.1.2 Языки спецификации специального назначения

CAPSL (Common Authentication Protocol Language). Язык CAPSL был разработан Милном (Millen) [DMR00] и позиционирован как спецификационный язык для криптографических протоколов. Этот язык имеет близкое сходство со стандартной нотацией протокола, поэтому он легко читается, и на нем легко писать. CAPSL имеет промежуточную форму CAPSL Intermediate Language (CIL), которая основана на логике переписывания (rewriting logic) и обеспечивает формальную семантику CAPSL. CAPSL позиционируется как единый язык спецификации протоколов, который может использоваться как входной формат для любой методики формального анализа, такой как Prolog'овые инструменты анализа, основанные на поиске в пространстве состояний [Mil95], NRL Protocol Analyzer [Mea96], проверка модели с помощью FDR [Low96], и HOL [Bra96].

CPAL (Cryptographic Protocol Analysis Language) [Yas96]. В то время как язык CAPSL используется довольно широко, CPAL не пользуется популярностью, несмотря на то, что он обладает теми же преимуществами, что и CAPSL, проще, лаконичнее и не требует промежуточного представления. Его формальная семантика определяется средствами пред- и постусловий.

NPATRL. Сиверсон (Syverson) и Мидоуз (Meadows) разработали язык требований для анализатора NRL Protocol Analyzer [SM93], который, в конечном счете, стал известным как NPATRL (NRL Protocol Analyzer Temporal Requirements Language). Необходимо было разработать простой темпоральный язык, который может применяться для спецификации требований, которые обычно используются в протоколах аутентификации и протоколах распределения ключей. Атомарные компоненты языка соответствуют событиям в протоколе (например, посылка и получение сообщений, или изучение злоумышленником некоего терма). Помимо обычных логических связок, язык содержит только один темпоральный

оператор, ⧫ – “произошло ранее” (happened previously). Использование этого единственного логического оператора отражает тот факт, что большинство требований пересылки сообщений может быть выражено в терминах событий, которые должны или не должны произойти перед некоторыми другими событиями.

ISL. В работе [Bra97] дается спецификация простого языка Interface Specification Language (ISL), который служит интерфейсным языком для инструмента Automatic Authentication Protocol Analyzer (AAPA) [Bra96a].

BMSL. Язык спецификации поведенческого мониторинга BMSL (behavioral monitoring specification language) [SU99] дает возможность создавать краткие спецификации свойств безопасности, основанные на событиях. Эти свойства могут отличать любое нормальное поведение программ и систем от неправильного поведения, связанного с возможной атакой.

Основанный на XML язык спецификации безопасности. Статья [BJBG04] посвящена описанию основанного на XML языка спецификации безопасности для документов Web-сервисов. Подход основан на применении ролевой модели контроля доступа (RBAC). Модель RBAC расширяется возможностью спецификации политики контроля доступа, а именно добавляются понятия иерархии ролей и разделения ограничений доступа. Предполагается работа только с XML-документами. Дается возможность специфицировать доступ к контенту на концептуальном уровне, а также на уровнях схемы, экземпляра и элемента документа. Кроме того, предлагаемое расширение модели обеспечивает возможность анализа контекста, в котором делается попытка доступа. Основанный на XML язык спецификации контроля доступа описывается в терминах мандатов пользователей, ролей и полномочий.

1.5.2 Алгебры процессов

Алгебры процессов (PA – Process algebras) – хорошо известная рамочная концепция, которая представляет семейство исчислений для описания распределенных и параллельных систем. Для описания свойств безопасности предложено большое количество расширений известных алгебр процессов CCS (Calculus of Communicating Systems) [Mil89] и CSP (Communicating sequential processes) [Hoa80]. В большинстве случаев применение таких исчислений к протоколам защиты опирается на множества примитивов, которые выражаются в терминах языка алгебры процессов.

Язык PA определяется следующей грамматикой:

Parallel processes $Q ::= 0 \mid Q \parallel P \mid Q \parallel !P$
Sequential processes $P ::= 0 \mid a(t).P \mid \overline{a}(t).P \mid [t = t'] P \mid \nu n.P$

Параллельные процессы определяются как параллельная композиция, возможно дублированных, последовательных процессов. Последовательные процессы передают сообщения t через каналы: выводящий процесс $\overline{a}(t).P$ готов послать кортеж термов t через канал a; принимающий процесс $a(t).P$ готов принять кортеж сообщений, соответствующих форматам t. Запись $[t = t']$ P означает, что для того чтобы процесс P продолжался, необходимо, чтобы термы t и t' совпадали. Создание нового процесса записывается как $\nu n.P$. Множество $c(t)$ – это множество констант, появляющихся в терме t, $c(Q)$ – это множество констант процесса Q . Операционная семантика дается следующими установками:

Single interaction $Q \Rightarrow Q'$; *Iterated interaction* $Q \Rightarrow^* Q'$

Они определяются следующим образом:

$$\frac{t' = t[\theta]}{(Q \parallel \overline{a}(t).P \parallel a(t').P') \Rightarrow (Q \parallel P \parallel P'[\theta])}$$

$$\frac{}{(Q \parallel !P) \Rightarrow (Q \parallel !P \parallel P)} \qquad \frac{Q \equiv Q'' \quad Q \Rightarrow Q'}{Q \Rightarrow Q'} \qquad \frac{}{Q \Rightarrow^* Q}$$

$$\frac{c(k) \cap (c(Q) \cup c(P)) = \emptyset}{(Q \parallel \nu n.P) \Rightarrow (Q \parallel P[k/n])} \qquad \frac{t = t'[\theta]}{(Q \parallel [t = t']\, P) \Rightarrow (Q \parallel P[\theta])} \qquad \frac{Q \Rightarrow Q' \quad Q' \Rightarrow^* Q''}{Q \Rightarrow^* Q''}$$

Первый вывод (реакция) показывает, как два последовательных процесса, соответственно, один готов совершить вывод термов t, другой готов совершить ввод термов t', реагируют, если унификация возможна между t и t'. Второй вывод (репликация) говорит, что $!P$ есть по существу фабрика процессов P. Следующие выводы описывают соответственно, рефлексивность перехода, генерацию нового имени, семантику совпадения и транзитивность отношения переходов.

Протоколы как процессы. Протоколы защиты могут быть описаны с помощью некоторого подмножества языка PA, где каждое взаимодействие происходит через сеть (P_{net} – процесс, управляющий сетью как открытым каналом, где каждый P_ρ посылает и принимает сообщения). В системе присутствует злоумышленник с некоторыми начальными знаниями ($Q_{!I}$ с начальными знаниями Q_{I0}), способный перехватывать и фальсифицировать сообщения, передаваемые по сети. Каждый принципал стартует протокол, играя некоторую роль ρ. Протокол защиты, включающий набор ролей ρ, выражается в подмножестве языка PA как процесс Q_0, определяемый параллельной композицией, состоящей из пяти компонентов:

$$P_{net} \parallel \prod_P !P_\rho \parallel Q_{!I} \parallel Q_{I0} \parallel Q_{!\pi}$$

где $\prod_P$ означает параллельную композицию всех процессов в P, а именно:

- $P_{net} = !(N_i(x).\overline{N_o}(x).0)$ описывает поведение сети. Она просто копирует сообщения из канала N_i (вход сети) в N_o (выход из сети), реализуя асинхронную форму передачи сообщений.
- P_ρ – каждый из этих процессов представляет последовательность воздействий, которые определяют роль. Эти процессы имеют следующую форму: $P_\rho = \tilde{\pi}(x).\nu n.P'_\rho$, где P'_ρ – последовательный процесс, который производит ввод/вывод только на сетевых каналах или осуществляет сравнение термов. Формально $P'_\rho ::= 0 \mid N_o(t).\, P'_\rho \mid \overline{N_i}(t).P'_\rho \mid [t = t']P'_\rho$.
- $Q_{!I} = !P_{I1} \parallel \ldots \parallel !P_{I10}$ – это спецификация модели злоумышленника в стиле Dolev-Yao. Каждый P_{Ii} описывает одну способность злоумышленника. Специализированный канал I хранит информацию, которой оперирует злоумышленник (она может быть начальной, перехваченной, или фальсифицированной).
- $Q_{I0} = \prod \overline{I}(t).0$ для некоторых термов t. Q_{I0} представляет начальные знания злоумышленника.
- $Q_{!\pi} = \prod !\overline{\pi}(t).0$ – можно предполагать наличие одинаковых имен предикатов (здесь каналов) с одинаковым смыслом. Эта информация делается доступной клиентским процессам на каждом канале π. Предполагается, что не существует других каналов, производящих вывод на канал π.

Первой публикацией о применении алгебры взаимодействующих процессов CSP для анализа протоколов была работа Роско (Roscoe) [Ros95]. Все агенты, участвующие в протоколе, включая взаимодействующих принципалов и злоумышленника, описываются в терминах CSP. Этот метод подходит для формализации сообщений, трасс, злоумышленников и т.п. Для верификации используется инструмент FDR (Failures Divergence's Refinement checker – анализатор моделей для CSP), который является универсальным инструментом для установления того, что реализация уточняет спецификацию.

Лоу (Lowe) [Low96] продемонстрировал применение алгебры процессов CSP и универсального анализатора моделей FDR к криптографическому протоколу.

Этот подход получил развитие в работах [GJR96], [Sch96] и [RS99].

Spi-исчисление. Абади (Abadi) и Гордон (Gordon) в [AG97] предложили Spi, исчисление для описания криптографических протоколов на абстрактном уровне. Авторы расширили π-исчисление [MPW92], алгебру мобильных процессов, с тем, чтобы промоделировать некоторые свойства криптографических протоколов. π-исчисление разработано для описания примитивных коммуникационных каналов. Spi-исчисление дает возможность явно представлять применение криптографии в протоколах. Доказательство безопасности протокола устанавливается через проверку эквивалентности между двумя процессами двух различных принципалов, взаимодействующих друг с другом. Злоумышленник явно не моделируется, и в этом состоит главное преимущество этого подхода. Вместо этого злоумышленник представляется как процесс Spi-исчисления.

Авторы работы [BDNN01a] описывают анализ потока управления, который гарантирует секретность модели Dolev-Yao для spi-исчисления. Предлагается некоторое консервативное расширение анализа, которое позволяет поддерживать конфиденциальность на основе свойства невлияния.

Security Process Algebra. SPA (Security Process Algebra) [FG01] выражает концепции невлияния (Noninterference) и является вариантом алгебры процессов CCS (Milner's Calculus of Communicating Systems) [Mil89]. SPA позволяет изучать свойства безопасности информационного потока на основе бисимуляции. В работе [FG94] вводится концепция свойства невлияния на языке SPA в терминах бисимуляционной семантики. Свойство называется Bisimulation-based non Deducibility on Compositions (BNDC): система обладает свойством BNDC, если пользователь низкого уровня не наблюдает модифицирование системы (в смысле бисимуляционной семантики) при выполнении любого процесса высокого уровня.

В SPA множество наблюдаемых воздействий разделяется на два множества – воздействия высокого уровня и воздействия низкого уровня для того, чтобы специфицировать многоуровневые системы. Синтаксис SPA основан на элементах CCS, а именно: множество $\mathcal{I} = I \cup O$, где $I = \{a,b,\ldots\}$ – это множество входных воздействий и $O = \{\bar{a}, \bar{b}, \bar{c}, \ldots\}$ – это множество выходных воздействий, специальное воздействие τ моделирует внутренние вычисления, т.е. ненаблюдаемые извне, а также дополнительная функция $(\bar{\ }) : \mathcal{I} \to \mathcal{I}$, такая что $\forall l \in \mathcal{L} : \bar{\bar{l}} = l$. $Act = \mathbf{L} \cup \{\tau\}$ – множество всех воздействий. Множество наблюдаемых воздействий разделяется на два множества – H и L, высокого и низкого уровня соответственно. Выражения SPA:

$E ::= 0 \mid a.E \mid E + E \mid E||E \mid E\backslash v \mid E[f] \mid Z,$

где $a \in Act$, $v \subseteq \mathcal{I}$, $E + E$ – недетерминированный выбор, $E||E$ – параллельная композиция, $E\backslash v$ – ограничение, $E[f]$ – переразметка (relabeling), Z – константа. Вводится $\boldsymbol{E}$ – набор всех SPA процессов, $\boldsymbol{E}_{\mathrm{H}}$ – процессы высокого уровня, т.е. процессы, использующие воздействия из $\mathrm{H} \cup \{\tau\}$.

Семантика SPA дается в виде набора правил, которые здесь не приводятся.

Операционная семантика SPA дается в терминах системы размеченных переходов (LTS – Labeled Transition Systems). LTS описывается триплетом $(S, A, \to)$, где S – множество состояний, A – множество разметок (воздействий), $\to \subseteq S \times A \times S$ – множество размеченных переходов. Нотация или $S_1 \xrightarrow{a} S_2$ означает, что система может перейти из состояния S_1 в состояние S_2 вследствие воздействия a.

Далее применяется концепция наблюдаемой эквивалентности (observation equivalence) для того, чтобы установить соотношения равенства между процессами. Две системы имеют одинаковую семантику тогда и только тогда, когда внешний наблюдатель не может их различить. Согласно отношению слабой бисимуляции (weak bisimulation) [Mil89], два процесса считаются равными, если их наблюдаемое поведение совпадает на каждом шаге выполнения.

Авторы утверждают, что главное преимущество BNDC по отношению к свойствам безопасности, основанных на трассах, состоит в том, что оно более мощное и позволяет обнаруживать информационные потоки, которые позволяют злоумышленным процессам высокого уровня блокировать или разблокировать систему, что показано в [FG94] и [FG01]. При этом такая система, построенная на основе трасс, будет считаться безопасной, и в ней нельзя обнаружить скрытые каналы. Свойства Noninterference, такие как BNDC, обеспечивают формальное определение безопасности информационного потока, поэтому это свойство полезно для понимания безопасности систем и сетей.

Security Process Language. В работе [CW01] описывается язык процессов для протоколов защиты с семантикой в терминах множеств событий. Процесс представляется в виде множества событий, и т.к. каждое событие сопровождается наборами пред- и постусловий, это представление может быть описано как сеть Петри. На примере демонстрируется, как семантику сетей Петри можно использовать для доказательства свойств безопасности. Модель сетей Петри и индуктивные правила Паульсона (Paulson) очень близки. В терминах сетей Петри правила Паульсона соответствуют определенному виду событий, где и сетевые, и управляющие условия являются "постоянными" условиями. Эта работа близка также к подходу strand space [THG98]. В действительности, одиночные процессы, компоненты системы, можно рассматривать как стренды, и приведенные доказательства используют их, в основном, в этом смысле. Стренды, в свою очередь, близки к структурам событий. Для сетей с "постоянными" условиями, может быть доказано, что они соответствуют безопасным сетям. Приведенная в работе семантика сетей Петри описывается в соответствующем разделе "Сети Петри".

Расширение π-исчисления. В [BDNN01] работе продемонстрировано использование анализа потока управления для исследования введения свойств безопасности модели Bell-LaPadula в π-исчисление.

1.5.3 Мультимножественная подстановка

Мультимножественная подстановка (MSR – Multiset Rewriting) уходит своими корнями в теорию параллельных процессов и логику подстановок. Этот формальных подход широко применяется для изучения фундаментальных вопросов протоколов защиты. Он также играет практическую роль в качестве языка промежуточного уровня CIL в системе для анализа протоколов CASPL [DMR00].

Язык мультимножественной подстановки первого порядка определяется следующей грамматикой:

Elements	$\tilde{a} ::= \cdot \mid a(t), \tilde{a}$
Rewriting Rules	$r ::= \tilde{a}(x) \rightarrow \exists n.\tilde{b}(x,n)$
Rule sets	$\tilde{r} ::= \cdot \mid r, \tilde{r}$

Элементы мультимножеств выбираются как атомарные формулы $a(\boldsymbol{t})$ для термов $\boldsymbol{t} = (t_1,\ldots,t_n)$ над некоей сигнатурой $\sum$ первого порядка. Записывается $\tilde{a}(\boldsymbol{x})$ и, соответственно, $t(\boldsymbol{x})$, чтобы подчеркнуть, что переменные из $\boldsymbol{x}$ появляются в мультимножестве $\tilde{a}$ (и соответственно в терме t). Символ "," означает объединение мультимножеств, которое неявно рассматривается как коммутативное и ассоциативное, в то время как "·" означает пустое мультимножество,

которое действует как нейтральный элемент, разрешающий покидать его, когда это удобно). Операционная семантика MSR выражается следующими установками:

Single rule application $\tilde{r} : \tilde{a} \longrightarrow \tilde{b}$

Iterated rule application $\tilde{r} : \tilde{a} \longrightarrow^* \tilde{b}$

Мультимножества $\tilde{a}$ и $\tilde{b}$ называются состояниями и всегда являются основными формулами. Стрелка означает переход. Эти установки определяются следующим образом:

$$\frac{}{(\tilde{r}, \tilde{a}(x) \to \exists n.\tilde{b}(x,n)) : (\tilde{c}, \tilde{a}[t/x]) \longrightarrow (\tilde{c}, \tilde{b}[t/x, k/n])}$$

$$\frac{}{\tilde{r} : \tilde{a} \longrightarrow^* \tilde{a}} \qquad \frac{\tilde{r} : \tilde{a} \longrightarrow \tilde{b} \qquad \tilde{r} : \tilde{b} \longrightarrow^* \tilde{c}}{\tilde{r} : \tilde{a} \longrightarrow^* \tilde{c}}$$

Первый вывод показывает, как используется правило подстановки $r = \tilde{a}(\boldsymbol{x}) \to \exists \boldsymbol{n}.\ \tilde{b}\ (\boldsymbol{x},\boldsymbol{n})$ для преобразования состояния в последующее состояние: находится основной экземпляр $\tilde{a}(\boldsymbol{t})$ предшественника, и он замещается экземпляром $\tilde{b}(\boldsymbol{t}, \boldsymbol{k})$, где $\boldsymbol{k}$ – это новые константы. Здесь $[\boldsymbol{t}/\boldsymbol{x}]$ означает подстановку, замещающую каждое появление переменной x в $\boldsymbol{x}$ с соответствующим термом t в $\boldsymbol{t}$. Эти правила реализуют недетерминированную, но последовательную вычислительную модель (одно правило за раз). Параллельность улавливается через перестановочность (permutability) некоторых применяемых правил. В остальных правилах $\to^*$ определяется как рефлексивное и транзитивное замыкание $\to$.

Протоколы как мультимножественная подстановка. Язык MSR опирается на следующие предикатные символы [CDL00]:

- Сетевые сообщения ($\tilde{N}$) – это предикаты для моделирования сети, где $N(t)$ означает, что t располагается в сети.
- Состояния ролей ($\tilde{A}$) – это предикаты для моделирования ролей. Пусть множество идентификаторов ролей есть $R = \{\rho_1,\ldots, \rho_n\}$, тогда семейство предикатов состояний ролей $\{A_{\rho i}\ (t) : i = 0\ldots l_\rho\}$, предназначено для хранения внутреннего состояния, t, принципала в роли $\rho \in R$ во время выполнения шагов протокола. Поведение каждой роли ρ описывается через конечное число правил, индексированных от 0 до l_ρ.
- Злоумышленник ($\tilde{I}$) – это предикаты для моделирования злоумышленника I, где $I(t)$ означает, что злоумышленник знает сообщение t.
- Постоянные предикаты ($\tilde{\pi}$) – это базовые предикаты для хранения данных, которые не изменяются во время разворачивания протокола. Правила используют эти предикаты для доступа к значениям постоянных данных.

Протокол защиты выражается в языке MSR как множество правил подстановок $\tilde{r}$ специального формата, называемого теорией протоколов защиты. Если R – это множество ролей, тогда можно записать, что $\tilde{r} = \cup_{\rho\in R}\ (\tilde{r}_\rho), \tilde{r}_I$, где $\tilde{r}_\rho$ и $\tilde{r}_I$ описывает поведение роли $\rho \in R$ и злоумышленника I. Для каждой роли ρ, правила в $\tilde{r}_\rho$ состоят из:

– правила с одной конкретизацией $r_{\rho 0}$: $\tilde{\pi}\ (x) \exists n.A_{\rho 0}(n; x), \tilde{\pi}\ (x)$

– нуль или больше ($i = 1\ldots l_\rho$) правил обмена сообщениями:

send $r_{\rho i}$: $A_{\rho i-1}\ (x) \to A_{\rho i}\ (x), N(t(x))$

receive $r_{\rho i}$: $A_{\rho i-1}\ (x), N(t(x,y)) \to A_{\rho i}\ (x, y)$

Правила в $\tilde{r}_I$ являются стандартными правилами, описывающими злоумышленника в стиле Dolev-Yao [DY83], чьи способности состоят в перехвате, анализировании, синтезировании и

создании сообщений с возможностью доступа к долговременным данным. В MSR, состояние есть мультимножество вида

$$\tilde{s} = (\tilde{A}, \tilde{N}, \tilde{I}, \tilde{\pi}),$$

где компоненты вобрали в себя базовые факты вида $N(t)$, $A_{\rho i}(t)$, $I(t)$ и $\pi(t)$, соответственно. Начальное состояние $\tilde{s}_0 = (\tilde{\pi}, \tilde{I}_0)$ содержит только постоянные предикаты ($\tilde{\pi}$), а начальные знания злоумышленника ($\tilde{I}_0$). Пара ($\tilde{r} : \tilde{s}$), состоящая из теории протокола $\tilde{r}$ и состояния $\tilde{s}$, называется конфигурацией. Начальной конфигурацией является ($\tilde{r} : \tilde{s}_0$).

В работе Лонгли (Longley) и Ригби (Rigby) [LR92] применяется система, основанная на правилах, которая трансформирует цели в подцели и способна бесконечно продолжать этот процесс. Схема, основанная на правилах, используется, чтобы построить дерево, в котором каждый узел представляет элемент данных, а дети узла представляют те элементы данных, которые требуются для знания о данных, представленных в родительском узле. Таким образом, можно построить дерево, в котором корневой узел представляет данные, необходимые злоумышленнику для атаки (например, криптографический ключ), а листья представляют элементы данных, которые требуются для корневого элемента. Соответствующий инструмент позволяет пользователю взаимодействовать с системой. Пользователь способен определить, может или не может элемент данных быть найден злоумышленником. Если делается вывод, что такой элемент данных может быть доступен, информация вставляется в систему, и процесс продолжается. С помощью такого подхода были найдены тонкие и ранее неизвестные дефекты в иерархической схеме управления ключами.

В работе [CDL00] применяется формализм MSR для спецификации правил протокола и действий злоумышленника. Этот формализм также использовался в инструменте для верификации протоколов защиты [CV01], в реализации языка OCAML в системе CASRUL [CJRTV] и для обработки правил переписывания в инструменте для автоматического доказательства теорем daTac [JRV00].

1.5.4 Конечные автоматы

Конечные автоматы (или системы размеченных переходов) могут быть определены различными способами. Здесь приводится один из них. Обычный конечный автомат есть 4-местный кортеж $A = (Q, q_0, L, R)$, где:

– Q – конечное множество состояний;

– q_0 – начальное состояние;

– L – конечное множество меток воздействий;

– $R \subseteq Q \times Act \times Q$ – отношение переходов, $(q, \lambda, q') \in R$ означает $q \xrightarrow{\lambda} q'$.

Несколько нотаций поведенческих прямых порядков (behavioral preorders) и эквивалентностей было определено на автоматах. Важным понятием является нотация бисимуляции [Mil89]. Бисимуляция автоматов определяется следующим образом. Пусть A_1 и A_2 – два автомата на одном и том же множестве меток L. Бинарное отношение $R \subseteq Q_1 \times Q_2$ есть симуляция для A_1 и A_2, если для всякого $q_1 \, R \, q_2$ выполняется следующее:

для всех $t_1 : q_1 \xrightarrow{\lambda} q'_1$ из A_1 существует $t_2 : q_2 \xrightarrow{\lambda} q'_2$ из A_2, такое что $q'_1 \, R \, q'_2$.

Отношение R является бисимуляцией, если и R и R^{-1} являются симуляциями. Два автомата A_1 и A_2 бисимулянтны, пишется $A_1 \sim A_2$, если их начальные состояния q^0_1 и q^0_2 являются бисимулянтными, а именно $q^0_1 \, R \, q^0_2$, для некоторой бисимуляции R.

Сидху (Sidhu) [Sid86] и Варадхарайян (Varadharajan) [Var89] специфицируют протокол с помощью теории конечных автоматов, используя диаграммы состояний. Для представления каждого принципала используется ориентированный граф. Специфицируется начальное состояние, затем проводится дуга к другому состоянию для каждого сообщения, которое может быть послано или получено из предыдущего состояния. Варадхарайян (Varadharajan) [Var89] приводит диаграмму состояний для каждой сущности в протоколе Нидхама-Шредера, таким образом, делается попытка описать поведение принципалов в протоколе. Представление состояний в [Sid86] слегка отличается от представления в [Var89].

В системе Interrogator [MCF87], предложенной Милленом (Millen) и др., участники протокола моделируются как взаимодействующие конечные автоматы, чьи сообщения перехватываются злоумышленником, который способен читать, уничтожать, или модифицировать сообщения. Состояние, в котором злоумышленник знает некое слово, которое должно быть секретным, принимается как конечное состояние, и из него Interrogator пытается воспроизвести маршрут в пространстве состояний, по которому можно было прийти в это конечное состояние. Если такой путь находится, это означает, что существует дефект в безопасности протокола.

NRL Protocol Analyzer [Mea92], предложенный Мидоуз (Meadows), похож на Interrogator. Специфицируется небезопасное состояние и Analyzer пытается воссоздать путь к этому состоянию из начального состояния. В отличие от Interrogator'а разрешается соединять неограниченное количество выполнений протокола в одном маршруте. Это позволяет обнаруживать атаки, которые основаны на способности злоумышленника соединять несколько различных прогонов протокола вместе. Кроме того, Analyzer не только находит пути к небезопасным состояниям, но с его помощью можно доказывать, что эти состояния недостижимы. Становится возможным доказывать, что некоторые пути, ведущие назад из небезопасных состояний, приводят к бесконечным петлям, из которых невозможно попасть в начальное состояние. При удалении этих путей результирующее пространство становится достаточно малым для проведения эффективного поиска. Однако доказательство того, что путь ведет к бесконечным петлям, в основном ложится на пользователя, и таким образом, поиск в Analyzer'е менее автоматизирован, чем в Interrogator'е.

Strand spaces. В своей работе Тайер (Thayer), Херцог (Herzog) и Гутман (Guttman) [THG98] предложили некую интерпретацию модели Dolev-Yao, основанную на теории графов и получившую название "strand space model". В этой модели действия принципалов моделируются в терминах графов. Стренд (нить, прядь) представляет принципала, выполняющего некую роль в протоколе. Посылка и получение сообщений представлены положительными и отрицательными узлами. Узлы, которые представляют два события, одно из которых непосредственно предшествует другому, на стренде соединены двойными стрелками. Связка (bundle) – это коллекция стрендов, в которых положительные посылающие узлы могут быть связаны с негативными принимающими узлами через единственную стрелку, если посланное сообщение соответствует полученному сообщению. Эта модель облегчает графическое представление протоколов, и в работе [THG98a] описывается ряд способов, в которых используются графические особенности пространства стрендов (Strand space). Модель Strand space сводит воедино многие идеи и методы, которые использовались в формальном анализе криптографических протоколов. Благодаря ее простоте и элегантности она стала применяться как в качестве основы для новых инструментов специализированного назначения [SBP01], так и в качестве рамочной концепции для обобщения теоретических результатов [Sto99].

В работе [LVH03] рассматривается применение подхода Strand spaces для анализа криптографических протоколов.

APA (asynchronous product automata). В работе [GOR02] криптографические протоколы формально определяются как система агентов протокола, использующих асинхронные

произведения автоматов (asynchronous product automata – APA). APA является универсальным операционным описанием концепции взаимодействующих автоматов. Отдельное состояние каждого агента конструируется из нескольких компонентов, описывающих его знания о ключах, его представление протокола и цели, которые должны быть достигнуты в пределах протокола. Взаимодействие моделируется с помощью добавления и удаления сообщений из разделяемого состояния компонента сети. Криптография моделируется символическими функциями с определенными свойствами. В дополнение к регулярным агентам протокола формально определяется злоумышленник, который имеет доступ в сеть, но не имеет никакого доступа к локальным состояниям агентов. Злоумышленник может перехватить сообщения и создать новые, основанные на его начальном знании и на том, что он может извлечь из перехваченных сообщений.

APA можно рассматривать как семейство элементарных автоматов. Набор всех возможных состояний АПА структурируется как множество-произведение; каждое состояние подразделяется на компоненты состояний. Далее множество всех возможных состояний называется множеством состояний. Множества состояний элементарных автоматов состоят из компонентов множества состояний APA. Различные элементарные автоматы склеиваются с помощью разделяемых компонент множеств их состояний. Элементарные автоматы могут взаимодействовать, изменяя содержимое разделяемых компонент состояний.

Протоколы рассматриваются как взаимодействующие системы, таким образом, APA – адекватное средство для формализации протокола. Рисунок 3 показывает структуру асинхронного автомата произведения, моделирующего систему из трех агентов протокола A, B и S. Круги представляют компоненты состояний, прямоугольники – элементарные автоматы. Каждый агент, принимающий участие в протоколе моделируется одним элементарным автоматом, который выполняет действия агента, и четырьмя компонентами состояний Symkeys, Asymkeys, State, и Goals, для запоминания симметричных и асимметричных ключей, локального состояния и целей безопасности, которые должны быть достигнуты в пределах протокола, соответственно. Единственный компонент состояний, разделяемый между всеми агентами (всеми элементарными автоматами) – компонент Network, который используется для коммуникации. Сообщение посылается с помощью добавления его к содержимому Network и получается с помощью удаления его из Network. Отношение соседства (графически представленное дугой) указывает, какие компонент состояний включены в это состояние элементарного автомата и могут быть изменены переходом элементарного автомата. Полная спецификация автомата включает множества состояний (типы данных), отношения переходов элементарных автоматов и начального состояния.

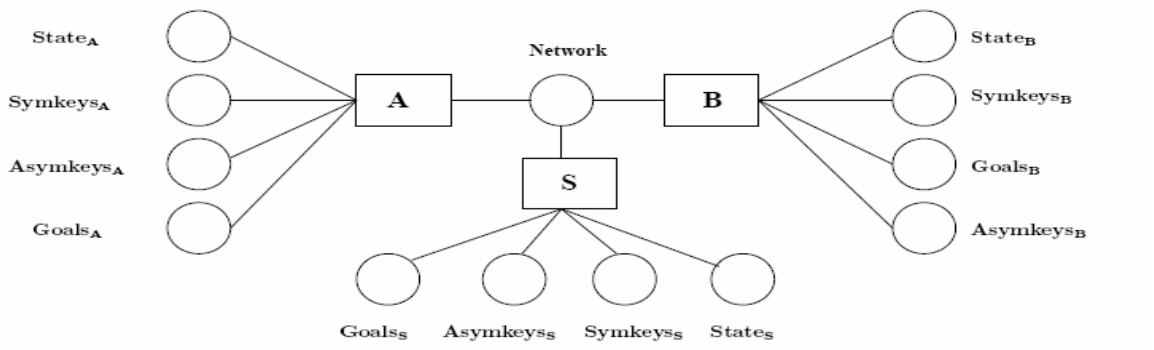

Рис. 3. Структура модели APA для агентов A, B и S.

Нарушения целей безопасности могут быть найдены с помощью SH-верификации [ORRN99]. Метод демонстрируется на симметрическом протоколе Нидхама-Шрёдера. Спецификации протокола не используют неявные предположения, таким образом, безопасность протокола не зависит от того, являются ли некоторые неявные предположения

приемлемыми для конкретного окружения. Авторы утверждают, что их метод обеспечивает необходимый базис для безопасных реализаций.

Интерактивные конечные автоматы (Interacting State Machines – ISMs). В литературе встречается, по крайней мере, три формализма, имеющие очень похожие (если не одинаковые) названия – все они являются последователями подхода, который расширяет неинтерактивные конечные автоматы явным вводом-выводом.

Взаимодействующие конечные автоматы или Interactive State Machines (ISMs) были введены Девидом фон Охеймбом (David von Oheimb) в работе [Ohe02] как общий формализм для абстрактного моделирования и верификации реактивных систем. Графическое представление автоматов описывается с помощью инструмента AutoFocus [HSSS96]. Семантика ISMs определяется через Higher-Order Logic (HOL) [GM93] внутри теорем-прувера Isabelle [Pau94]. ISMs можно рассматривать как высокоуровневые варианты автомата ввода/вывода (IOAs) [LT89]. Автор приводит семантическую трансляцию ISMs в IOAs, а также демонстрирует сильные стороны этого подхода, показывая как с его помощью можно выразить и доказать свойства безопасности, обнаруживая при этом уязвимости протокола Нидхама-Шредера.

В качестве основы для ISMs были выбраны автоматы ввода/вывода I/O Automata (IOAs) [LT89], которые разрабатывались для моделирования состояние-ориентированных асинхронных распределенных вычислений. Для того, чтобы выразить асинхронное взаимодействие таких систем через сообщения, к разрабатываемому варианту IOAs было добавлено буферизованное взаимодействие, концепция которого была заимствована из автоматов AutoFocus [HSSS96].

ISMs являются автоматами, в которых переходы между состояниями могут допускать многократные входные и выходные воздействия одновременно на любом числе портов. Ключевыми концепциями ISMs являются состояния (и в особенности переходы между ними) и взаимодействие (interaction). Под взаимодействием подразумевается явная буферизированная коммуникация через именованные порты (которые также называют подключениями), где на каждом порту один получатель слушает, возможно, многих отправителей. Система ISM – интерливинговая (interleaved) параллельная композиция из произвольного числа компонентов ISM, где состояние целой системы – по существу Декартово произведение состояний его компонентов. Состояние ISM состоит из его входных буферов и локального состояния. Локальное состояние может иметь произвольную структуру, но обычно – это Декартово произведение состояния управления, которое имеет конечный тип и состояние данных, которое является записью в названных полях, представляющих локальные переменные. Каждый ISM имеет единственное локальное начальное состояние.

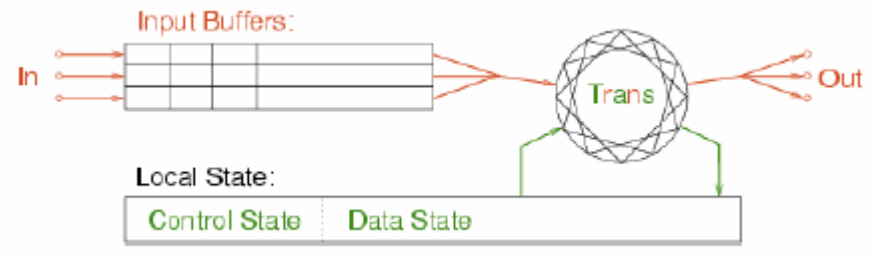

Рис. 4. Структура ISM [Ohe02].

Каждый ISM объявляет два набора имен портов, для ввода и для вывода. Входные буфера – семейство очередей (неограниченных) сообщений типа FIFO, которое индексируется именами портов. Обмен сообщениями вызывается любым ISM, посылающим сообщение в пределах системы или в окружающую среду. Входные воздействия не могут блокироваться, т. е. они могут произойти в любое время, при этом полученное значение

добавляется к соответствующей FIFO. Значения, сохраненные во входных буферах ISM, могут быть обработаны ISM, когда ISM будет готов сделать это. Это делается с помощью переходов, определенными пользователем, которые могут быть недетерминированными и могут быть специфицированы в любом реляционном стиле. Таким образом, пользователь имеет выбор для определения их в любой манере, операционной (т. е., исполнимой) или аксиоматической (т. е., ориентированной на свойство), или их композицией.. Правила перехода определяют, что (согласно некоторому предусловию, которое может включать сравнение на соответствие сообщений во входных буферах) ISM перерабатывает все входные воздействия из своих буферов, делает локальный переход и производит некоторый вывод. Вывод отправляется во входные буфера всех ISMs, слушающих на соответствующих портах, которые могут привести к прямой или косвенной обратной связи. Выполнение ISM или системы ISM – любая конечная последовательность конфигураций, достижимых из начальной конфигурации. Переходы различных ISMs, которые выполняются параллельно, связаны только причинной связью через обмен сообщениями. Выполнение застревает, когда нет никакого компонента, который может выполнить какой-нибудь шаг. Как принято для реактивных систем, нет никакого встроенного понятия конечных или "принимающих" состояний.

Interactive Algebraic state machines (IAlgSMs) [Jur03]. Алгебраические конечные автоматы (AlgSMs) были описаны в [BW00]. AlgSMs основаны на абстрактных конечных автоматах (ASMs) Гуревича (Gurevich) [Gur97]. AlgSMs замечательны тем, что в них каждое состояние есть алгебра, а асинхронное взаимодействие осуществляется через каналы. В работе [Jur03] автор предлагает интерпретировать алгебраический конечный автомат как функцию обработки потока (tream-processing function). Применяя новую концепцию к проблеме безопасности, автор расширяет этот формализм абстрактными криптографическими операциями и рассматривает применении этой концепции к протоколу TLS.

Древововидные автоматы (tree automata). В работе [Mon99] для абстрактного описания криптографических протоколов рассматривается модель, основанная на древововидных автоматах. Описывается реализация алгоритмов, позволяющих успешно анализировать некоторые небольшие экземпляры (2 принципала и 1 сервер) известных протоколов и тестовых примеров. Эта абстракция является хорошо гранулированной, что дало возможность получить успешные результаты на реальных протоколах. Основным недостатком данного метода является необходимость рассматривать большое число интерливингов, что на практике ограничивает количество одновременно анализируемых сеансов. Идея относительно аппроксимированных сверху множеств термов была далее усовершенствована в [GK00] с помощью устойчивых множеств, которые дают возможность аппроксимировать неограниченное число сеансов.

Групповые автоматы (Team Automata). В работе [BLP04] говорится, что групповые автоматы (Team Automata – TA) хорошо подходят для анализа безопасности при условии переформулирования обобщенного свойства невыводимости на композициях (Generalized Non-Deducibility on Compositions – GNDC) в терминах TA. Показывается, что при этом подходе гарантируется целостность для некоторых отдельных протоколов.

1.5.5 Модальные логики

Модальные логики получили широкое распространение в области специфицирования криптографических протоколов. Применяются эпистемические, темпоральные и дохастические логики. Дохастическая логика основана на доверии, и является системой выводов, которая применяет правила о том, как вывести новое доверие на основании

имеющегося доверия. Эпистемическая логика напоминает дохастическую, с тем отличием, что она основана на знании.

Такие логики состоят из языка, который описывает различные утверждения о доверии и знании принципалов относительно сообщений, и некоторых правил вывода, которые используются для вывода новых утверждений из предыдущих. Целью анализа является вывести утверждение, которое будет представлять корректное условие протокола. Невозможность вывода такого утверждения означает, что протокол, возможно, некорректен.

Особенно широкое распространение получили логики доверия после появления логики BAN [BAN90].

Логика BAN. Логика BAN, предложенная Барроузом (Burrows), Абади (Abadi) и Нидхамом (Needham) [BAN90] для анализа протоколов, является логикой доверия. Логика BAN позволяет описывать криптографические протоколы с помощью формальных терминов и рассуждать о состоянии доверия между принципалами в системе.

Эта логика состоит из набора модальных операторов, описывающих взаимосвязи между принципалами и данными, набора возможных взаимных доверий принципалов (например, убеждение, что сообщение было послано неким другим принципалом) и некоторого набора правил вывода для получения новых доверий из старых. Вот неформальный пример логики BAN: “Если A убежден, что A получил сообщение, зашифрованное с помощью ключа K, и A убежден, что только B и A знают K, тогда A убежден, что сообщение было создано либо B, либо A”. На Рис. 5 показаны предикаты логики BAN, которые приводятся для иллюстрации простоты этой логики, правила вывода не приводятся.

$P \mid\equiv X$	P believes X
$P \lhd X$	P sees X
$P \mid\sim X$	P said X (P once said X)
$P \mid\Rightarrow X$	P controls X (P has jurisdiction over X)
$\#(X)$	fresh(X) (X is fresh)
$P \overset{K}{\leftrightarrow} Q$	P and Q may share good key K
$\overset{K}{\mapsto} P$	K is a public key of P
$P \overset{X}{\Box} Q$	X is a secret shared by P and Q
$\{X\}_K$	X encrypted with K (or {X}K)
$\langle X \rangle_Y$	X combined with Y (or ⟨X⟩Y)
K^{-1}	inverse of (public) key K

Рис. 5. Предикаты логики BAN.

Логика BAN не пытается моделировать многие аспекты протокола, как делают другие логики. Она не пытается моделировать ни различие между простым просмотром сообщения и пониманием его, ни пересмотр доверий, ни знание. Это делается преднамеренно, только благодаря этому она является простой и прямолинейной. Однако это ведет к тому, что все эти аспекты адресуются к неформальному отображению спецификации протокола в спецификацию логики BAN. Такое отображение авторы называют идеализацией. Авторы логики BAN рассматривают идеализированные протоколы как более ясные и более полные спецификации, чем традиционные описания, которые они считают реализационно-зависимыми. Хотя с этим и можно согласиться, однако, авторы не приводят ясного метода идеализации, что зачастую приводит к неправильному пониманию и неправильному использованию этой логики [BM93].

Логика BAN состоит из простого интуитивного набора правил, что способствует ее широкому распространению, и привело к совокупности логик, или расширяющих логику

BAN, или применяющих ту же самую концепцию к различным типам проблем в криптографических протоколах.

Авторы отмечают, что поскольку протокол рассматривается на абстрактном уровне, ошибки конкретной реализации, такие как тупики или неправильное использование криптосистемы, не могут анализироваться с помощью их формализма.

Логика GNY. Гонг (Gong), Нидхам (Needham) и Яхалом (Yahalom) [GNY90] предложили расширение логики BAN, которое часто называют логикой GNY. Авторы описывают новые конструкции, которые сводят на нет некоторые предположения, сделанные в оригинальной логике BAN. В частности, логика GNY не предполагает, что принципалы являются достоверными, и что в зашифрованных сообщениях присутствует избыточность. Вместо этого вводится понятие распознаваемости (recognizability), чтобы представить тот факт, что принципал ожидает определенных форматов в сообщениях, которые он получает. Кроме того, добавляется возможность явно описать, действительно ли принципал создал само сообщение. Логика GNY делает различие между тем, чем принципал может обладать, и во что он может верить. Эта логика дает возможность описывать различные уровни доверия. Все это делает логику GNY более реалистичной моделью криптографического протокола, чем логика BAN. Однако логика GNY имеет более 40 правил вывода, что привело к тому, что многие считают ее непрактичной.

BGNY. Логика BGNY была введена Браскином (Brackin) [Bra96]. На основе теории Higher Order Logic (HOL) [GM93] она формализует и расширяет логику GNY. Эта логика доверия используется программным обеспечением, которое автоматически доказывает свойства аутентификации криптографических протоколов. Подобно логике GNY, BGNY обращается только к аутентификации. Однако BGNY расширяет логику GNY, включая возможность специфицировать свойства протокола на промежуточных стадиях, причем протоколы могут использовать многочисленные операции кодирования и хеширования, коды аутентификации сообщений, коды хеширования в качестве ключей, и алгоритмы обмена ключами.

Другими расширениями логики BAN являются [MB93], которая предлагает некоторые усовершенствования логики BAN; [GS91] предлагает расширить логику BAN конструкциями, учитывающими понятие времени; [CSP92] расширяет логику BAN, применяя вероятностные рассуждения для вычисления меры доверия.

Большинство работ, связанных с применением логического подхода для анализа безопасности, посвящается аксиоматизации доверия. Шухам (Shoham) и Мозес (Moses) [SM89] описывают взаимосвязь между знанием и доверием и отмечают близость доверия к немонотонным умозаключениям. Сиверсон (Syverson) [Syv92] показывает, что понятия доверия и знания равно адекватны для анализа протоколов на логическом уровне.

Применение формального анализа протоколов ограничивается качеством средств, которые при этом используются. Сиверсон (Syverson) в [Syv91] отмечает, что одной из главных ролей семантики является то, что она дает возможность оценивать логики. При оценке логики, в первую очередь, интересны два вопроса: ее полнота (completeness) – можно ли с ее помощью вывести все, что нужно, и ее непротиворечивость (soundness) – можно ли избежать вывода ненужных вещей. Формальная семантика обеспечивает точную структуру, по отношению к которой полнота и непротиворечивость логики может быть доказана. Однако Сиверсон объясняет, что семантика логики не должна выводиться непосредственно из нее самой. Только независимая семантика служит надежным средством для оценки этой логики.

Попытка создать более совершенную логику была сделана Сиверсоном в работе [SvO94]. Логика, впоследствии названная логикой SvO, объединила четырех своих предшественниц из семейства BAN, а именно логику GNY [GNY90], логику Abadi-Tuttle

[AT91], логику, предложенную в [vOor93], и собственно логику BAN [BAN90]. Сложность этой логики не превышает сложности логик, на основе которых она создана, но не приводится примеров ее использования.

Существует ряд логик, которые не принадлежат к семейству логики BAN. Они включают CKT5 [Bie90], KPL [Syv90], логику Мозера (Moser) [Mos89], и систему Яхалома (Yahalom) и др. [YKB93]. Логики CKT5 и KPL дают возможность делать выводы об эволюции знаний о словах, используемых в криптографическом протоколе, кроме того, в них различается простой просмотр сообщения от понимания его содержания. Логика Мозера является немонотонной логикой доверия, т.е. в отличие от монотонных логик, в ней можно рассуждать об изменении доверий между принципалами, например, в случае дискредитации ключа во время взаимодействия. Система Яхалома и др. используется для получения информации о природе доверительностей, которые должны иметь стороны по отношению друг к другу для того, чтобы протокол функционировал корректно.

Использование модальных логик знания и доверия является самым популярным подходом в области применения формальных методов к анализу криптографических протоколов. По всей вероятности, это происходит из-за простоты и понятности их применения. Однако этот путь зачастую влечет за собой отягчающие последствия, т.к. неполное понимание тонкостей предположений, лежащих в основе этих логик, приводит к частым ошибкам в анализе.

Вообще говоря, логики слабее методов исследования пространства состояний, т.к. они оперируют на более высоком уровне абстракции. Интерес к ним уменьшается по мере развития систем, основанных на исследовании пространства состояний (model-checking). Тем не менее, эти логики все-таки имеют преимущество в том, что они обычно разрешимы и зачастую эффективно вычислимы, и, следовательно, могут быть полностью автоматизированы, что и было сделано анализатором Automated Authentication Protocol Analyzer [Bra98].

Критику этого подхода можно найти в [Nes90], [Sne91], [Syv91], [MB93].

1.5.6 Сети Петри

Них и Таварес [NT92] предложили использовать сети Петри для формального моделирования и анализа криптографических протоколов. В частности, они применили раскрашенные сети Петри для моделирования протоколов. Их модель включает модель злоумышленника, которая может быть использована для описания атак злоумышленника и для генерации тестовых вариантов. Анализ свойств безопасности криптографических протоколов основан на исчерпывающем тестировании на проникновение (penetration test), при котором ищутся сценарии, нарушающие некоторые указанные критерии. Такие критерии определяются в терминах состояний сетей Петри для данного протокола. Хотя раскрашенные сети Петри позволяют создавать компактные и выполнимые описания протоколов, инструменты для поддержки эффективного выполнения исчерпывающего поиска все еще отсутствуют. Можно преобразовать раскрашенные сети Петри в обычные, и затем применить инструменты, существующие для обычных сетей Петри, но тогда придется иметь дело с проблемой взрыва состояний.

В работе [CW01] описывается язык процессов для протоколов защиты, а его операционная семантика дается в виде сетей Петри. Строительными блоками семантики являются события. События состоят из действия вместе с предусловиями, необходимыми для того, чтобы это событие произошло, и постусловиями, которые должны быть удовлетворены по окончании этого события.

1.5.7 Графическое моделирование

В [Jur01a], [Jur01b] рассматривается моделирование различных аспектов безопасности систем (включая многоуровневую безопасность, безопасность информационных потоков и протоколов безопасности) с помощью UML. В [Jur01c] ядро UML используется для спецификации контроля доступа в распределенных Java-системах. Показано, как определить требования безопасности и доказать, что моделируемые механизмы контроля доступа типа (guarded objects) удовлетворяют требованиям безопасности, и что эти механизмы являются совместимыми с полной функциональностью, требуемой от системы. Автор считает, что механизмы контроля доступа, которые предоставляет JDK 1.2, применять практически достаточно сложно (особенно, когда допускаются косвенные полномочия на доступ), поэтому весьма полезным кажется использование UML, который обеспечивает корректную спецификацию этих механизмов. В [Jur02] предлагается расширение UML, так называемое UMLsec, для описания свойств безопасности.

Strand space pictures. Модель Strand space [THG98] облегчает графическое представление протоколов, и в работе [THG98a] описывается ряд способов, в которых используются графические особенности пространства стрендов (Strand space). Используя представление протоколов с помощью пространства стрендов, можно представить требования пересылки сообщений в терминах относительного расположения стрендов. Таким образом, при спецификации требования, которое состоит в том, что, если определенные сообщения были приняты, то другие сообщения были посланы предварительно, можно представить посылку и принятие сообщений как порцию стрендов, используя при этом расположение стрендов (так, чтобы более ранние узлы появлялись над более поздними узлами) для того, чтобы указать, какие события произошли ранее других. Стренд может быть параметризован именем принципала, который выполняет этот стренд, и данными, которые он посылает и принимает.

В работе [CM03] описывается графическое представление языка NPATRL. Это представление основано на том, что запросы в NRL Protocol Analyzer, для которого NPATRL был разработан, описываются в терминах событий, которые должны или не должны предшествовать некоторому указанному событию. Такой способ форматирования запросов имеет очевидную связь с деревьями неисправностей (fault trees). Дерево неисправностей – графическое средство представления режимов отказа в критических с точки зрения безопасности системах. Корень дерева представляет неисправность, которая интересует системного проектировщика, а ветви представляют условия, при которых эта неисправность может произойти. Основное различие между запросами NPA и деревьями неисправностей состоит в том, что в запросах NPA связь основана на предшествовании, в то время как в деревьях неисправностей – на причинности. Во всем другом структура очень похожа. Кроме того, графическое представление облегчает понимание связей между различными событиями.

1.6 Методы формальной верификации

1.6.1 Верификация на основе конечных автоматов

Конечные автоматы могут быть использованы не только для спецификации, но также и для проведения анализа криптографических протоколов. В этом случае применяется методика, известная под названием методика анализа достижимости [Wes78].

Эта методика предполагает описание системы в следующем виде. Для каждого перехода строится глобальное состояние системы, которое выражается через состояния сущностей системы и состояния коммуникационных каналов между ними. Каждое глобальное состояние затем анализируется, и определяются его свойства, такие как тупик и корректность. Если сущность не способна получить сообщение, а предполагалось, что оно

должно было быть получено в этом состоянии, тогда существует проблема в протоколе. Пример такого анализа приводится в [Var89].

Методика анализа достижимости эффективна для определения корректности протокола по отношению к его спецификациям, однако она не гарантирует безопасности от активного злоумышленника.

В работе [GM84] абстракция конечных автоматов используется для верификации свойства невлияния. Авторы разработали совокупность раскручивающихся (unwinding) условий, которая является достаточной для установления свойства невлияния (NI) в конечных автоматах. Несмотря на то, что эти условия относительно просты, их применение зависит от модели конечного автомата для рассматриваемой системы. Позже Маклин показал, как разрабатывать такой автомат и верифицировать NI [McL92]. Эта верификационная методика помогает сделать свойство NI таким же полезным на практике, как и BLP. Хотя верификация NI, вообще говоря, может быть труднее, чем верификация BLP, в первой отсутствует анализ скрытых каналов, который необходим после верификации BLP.

В работе [Gou00] описывается автоматический завершенный метод для верифицирования свойств конфиденциальности криптографических протоколов. Метод основан на надежной абстрактной интерпретации криптографических протоколов, в котором используется расширение древовидных автоматов, а именно V-параметризованных древовидных автоматов, которые смешивают автомато-теоретические методики с особенностями дедуктивных методик. Вопреки большинству подходов, основанных на проверке модели, этот метод предлагает фактические гарантии защиты. Описывается возможность анализа протоколов в присутствии параллельных многосеансных принципалов.

1.6.2 Проверка модели (model-checking)

Метод проверки модели впервые был предложен в начале 80-ых. В последнее время он становится индустриальной практикой и широко используется в практических приложениях, в особенности для проверки аппаратуры и коммуникационных протоколов, для анализа гибридных систем.

Метод заключается в специфицировании свойств системы на языке алгебры процессов или в виде формул темпоральной логики, построении модели в виде конечного автомата, автоматической проверке того, что эта модель удовлетворяет спецификации, причем при отрицательном выводе вырабатывается контрпример. Метод проверки модели состоит в переборе всех возможных переходов автомата из одного состояния в другое из некоего начального состояния системы. Все возможные трассы из начального набора состояний перебираются, чтобы убедиться в том, что все они являются безопасными, или доказать, что живучесть системы не может быть нарушена.

Метод проверки модели страдает недостатком, широко известным под названием “взрыв количества состояний“. При моделировании времени как непрерывной сущности даже самая простая модель имеет бесконечное число состояний. Наиболее известным подходом к решению этой проблемы является метод символической проверки модели, в котором проблема упрощается с помощью формирования классов эквивалентности. При этом используются компактные булевские формы для представления наборов состояний и переходов, как, например, BDDs (Binary Decision Diagrams), и накладываются некоторые ограничения на структуру пространства состояний.

В области коммуникационных протоколов проводились многочисленные исследования для анализа условий, при которых конечное число состояний при проверке модели становится достаточным. По всей видимости, наилучших результатов достигли исследования

Лоу (Lowe), в которых приводится набор условий, при которых проверка небольшого количества сессий становится достаточным для доказательства секретности ключа [Low99].

Успешное применение методики проверки модели к анализу протоколов защиты показали работы [MMS97], [DK97], в которых использовались анализаторы моделей универсального применения. Широкие возможности методики проверки модели привели к созданию некоторых специализированных для протоколов инструментов, основанных на этом подходе [KS99], [SBP01], [Mar97], и использованию специализированных инструментов, изначально предназначавшихся для других приложений [DFG99].

Многие исследователи в этой области сосредоточили свое внимание на разработке метода проверки модели, в котором предполагается ограниченное количество сессий, но при этом присутствует доказательство полноты, причем не накладываются ограничения на сложность сообщения, например, на глубину шифрования. Работа Хуимы (Huima) [Hui99] была первой в ряду таких исследований, затем последовали ее приложения и расширения [SBP01], [FA01], [AL00], [MS01]. Рузинович (Rusinowitch) и Туруани (Turuani) показали NP-полноту проблемы секретности при таких предположениях [RT01].

В работе [Mar98] показано, что свойство BNDC разрешимо для процессов с конечным числом состояний, и предлагается метод верификации, основанный на частичной проверке модели. Однако проблема эффективного способа верификации BNDC пока еще остается открытой, также как и проблема разрешимости BNDC. Решение этих проблем можно найти на пути адаптации достаточных условий для BNDC.

В работе [BFPR04] исследуются свойства безопасности информационного потока на основе бисимуляции. Эти свойства являются постоянными в том смысле, что если система безопасна, тогда все ее достижимые состояния являются также безопасными. Показывается, что такие свойства могут быть выражены в терминах бисимуляционно-подобных отношений эквивалентности между полной системой и системой, не допускающей выполнения конфиденциальных операций. Дается также характеристика таких свойств в терминах раскручивающихся условий. Эти два различных подхода к описанию свойств безопасности ведут к эффективным методам для верификации и конструирования систем безопасности. Приводятся также некоторые результаты, полученные для композиционности, которые позволяют проверять безопасность системы, верифицируя только ее компоненты. Предлагаются два метода для определения, обладает ли система свойством P_BNDC, SBNDC или PPBNDC. Свойство P_BNDC (Persistent BNDC) требует отсутствия влияния между взаимодействиями высокого и низкого уровней в каждом возможном состоянии, это и есть постоянство свойства BNDC. Свойство SBNDC (Strong BNDC) требует, чтобы и до, и после каждого шага высокого уровня, с точки зрения низкого уровня система оставалась бы той же самой. Это свойство является достаточным для верификации BNDC. Свойство PPBNDC (Progressing PBNDC) является композиционным для недетерминированных систем.

Первый метод основан на получении характеристических формул [SI94] на языке модального μ-исчисления. Характеристические формулы могут быть автоматически верифицированы с помощью анализаторов моделей для μ-исчисления, таких как NCSU Concurrency Workbench [CS96]. Даже если в худшем случае этот метод имеет экспоненциальную сложность по времени из-за количества состояний процесса, это все еще приемлемо во многих случаях, и имеет преимущество в сведении проверки свойств безопасности к стандартной проблеме верификации формул μ-исчисления.

Второй метод основан на вычислении своего рода транзитивного замыкания (Closure up to high level actions) системы и на верификации строгой бисимуляции. Это позволяет использовать существующие инструменты для верификации, т.к. к настоящему времени многие различные алгоритмы для вычисления строгой бисимуляции между двумя процессами интегрированы в анализаторы моделей, такие как NCSU Concurrency Workbench,

XEVE [Bou98], FDR2 [Ros98]. В частности, этот второй подход изменяет к лучшему полиномиальную сложность по времени инструмента CoSeC (Compositional Security Checker) [FG97], т.к. в этом случае потребуется только одно испытание бисимуляции.

В работе [GLL02] представляется рамочная концепция методики проверки модели для диалекта spi-исчисления, который использует линейную временную темпоральную логику для того, чтобы выразить свойства безопасности. Рассматривается диалект spi-исчисления, названный SPID, с семантикой, основанной на размеченных системах перехода (LTS), где злоумышленник моделируется в стиле Dolev-Yao как активная среда, управляющая взаимодействием, и способная составлять новые сообщения, соединяя и разбивая шифрование или расшифровку. Системы LTS, пришедшие из спецификации протоколов, обычно имеющие бесконечное количество состояний из-за возможности злоумышленника генерировать бесконечное количество сообщений, являются моделями, на которых была определена выполнимость логических формул. В качестве логики используется логика, определенная для анализатора моделей BRUTUS [CJM00], которая позволяет выразить широкий класс свойств безопасности, таких как конфиденциальность, целостность, подлинность, некоторая слабая форма анонимности и общих свойств надежности (safety). Обеспечение процедуры выполнимости на моделях, полученным из SPID-спецификаций протоколов, является основным достижением этой работы. Планируется разработка инструмента для автоматической проверки модели на основе этой концепции. Для этого необходим переход к конечным моделям. В качестве первого решения на этом пути дается ограничение на длину сообщений, которые злоумышленник может генерировать.

В работе [ACG03] исследуется автоматическая проверка модели, основанная на сведении проблем ненадежности протоколов к проблемам выполнимости в пропозициональной логике (SAT), которую можно использовать для выявления атак на протоколы защиты. Подход возник из комбинации сведения проблем ненадежности протоколов к проблемам планирования и известных методик SAT-редукции, называемых линейным шифрованием, разработанных для планирования. Экспериментальные результаты подтвердили эффективность подхода, но также показали, что время, потраченное на генерацию SAT-формулы, значительно превышает время, которое необходимо SAT-решающему устройству для проверки ее выполнимости. Кроме того, сгенерированные SAT-шаблоны имеют неуправляемый размер на самых сложных протоколах. Исследуется приложение методики шифрования, основанной на Graphplan, к анализу протоколов защиты и представляются экспериментальные данные, показывающие, что шифрование, основанное на Graphplan, значительно проще линейного кодирования. Эти результаты подтверждают эффективность SAT-подхода к анализу протоколов защиты и прокладывают путь к его приложениям для анализа больших протоколов, существующим в практических приложениях.

1.6.3 Доказательство теорем (theorem proving)

В работе [DD77] было впервые отмечено, что статический анализ программ можно использовать для управления информационным потоком с большой точностью и малыми накладными расходами. Статическое определение характеристик информационного потока было реализовано с помощью программ для доказательства теорем (theorem prover) [Fei80], [MG85].

Успешное применение методики доказательства теорем к анализу протоколов защиты показали исследования, о которых сообщается в работах [Pau98], [DS97].

Паульсон (Paulson) применил инструмент Isabelle (на основе метода доказательства теорем) для анализа достаточно сложных протоколов [Pau97]. Хотя по своей природе применение метода доказательства теорем требует большей интерактивности, чем метод

проверки модели, Паульсон выработал совокупность теорем и методик, которая может переиспользоваться в других анализах с помощью Isabelle.

В работе [GA98] вводится определение бисимуляции для криптографических протоколов. Это определение включает простую и точную модель знаний о среде, в которой действует протокол. Бисимуляция является основой эффективной методики доказательств, которая осуществляет доказательство классических свойств безопасности, а также устанавливает некоторую оптимизацию протокола. Доказывается семантическая непротиворечивость (soundness) бисимуляционной методики доказательств внутри spi-исчисления.

Разработаны также специализированные алгоритмы и инструменты, основанные на методике доказательства теорем, которые настроены на анализ криптографических протоколов – это система TAPS [Coh00] и система Хуимы [Hui99]. Специализированные системы требуют значительно меньшего взаимодействия с пользователем, чем универсальные, а также обеспечивают большее покрытие, чем анализаторы моделей. Однако, в отличие от анализаторов моделей, они не вырабатывают контрпримеров, если доказательство свойств безопасности приводит к ложным выводам.

В работе [BFPR02] используются раскручивающиеся условия для определения системы доказательств для процессов, обладающих свойством постоянной BNDC (PBNDC). Эта система доказательств обеспечивает мощную методику для верификации и разработки таких процессов. В самом деле, эта система доказательств позволяет верифицировать, является ли процесс безопасным, простым инспектированием его синтаксиса, что дает возможность избежать проблемы взрыва состояний. В частности, она допускает рекурсивные процессы, которые могут выполнять неограниченные последовательности действий, при этом, возможно, достигая бесконечного числа состояний.

1.6.4 Метод проверки типа (type checking)

Самым новым подходом к формальному анализу протоколов является использование метода проверки типа (type checking), введенным Абади (Abadi) [Aba99]. Абади ввел тип *untrusted* (Un) для открытых сообщений, которые исходят от оппонента (в качестве оппонентов выступают все, кроме удостоверяющих принципалов). Этот подход применяется также в работе [GJ01].

В методе проверки типа сообщениям и каналам присваиваются типы (например, единица данных приватного типа появляется на открытом канале). Метод проверки типа имеет существенное достоинство, состоящее в том, что он, как и метод проверки модели, является полностью автоматическим, но в отличие от последнего способен оперировать с несколькими классами бесконечных систем. Однако он имеет потенциальный недостаток, состоящий в том, что, так как нарушения безопасности определены в терминах несогласованностей типа, требования безопасности, которые должны быть доказаны, должны быть сформулированы в спецификации в процессе её написания. Это отличает метод проверки типов от метода проверки модели, для которого любое свойство безопасности, которое может быть выражено в терминах темпоральной логики, может специфицироваться независимо, уже после того, как сам протокол специфицирован.

В работе [GJ01] предлагается метод для проверки свойств аутентификации криптографических протоколов, который основан на двух идеях – идеи Ву (Woo) и Лэма (Lam) [WL93] об утверждениях соответствия для спецификации свойств аутентификации и идеи Абади о верификации свойств протокола с помощью проверки типов. Работа [WL93] описывает формальную семантику утверждений соответствия, но не предлагает метода верификации протокола. В работе [GJ01] протокол специфицируется с помощью типизированного расширения spi-исчисления. Свойства аутентификации выражаются в виде утверждений в стиле Ву и Лэма [WL93] и записываются как комментарии. Вычисляются

типы для ключей, nonces'ов (уникальные идентификаторы соединения) и сообщений протокола, и затем производится проверка типов по методу Абади [Aba99]. Формальная операционная семантика для spi-исчисления описывается с помощью семантики трасс, основанной на Chemical Abstract Machine [BB92].

Метод проверки типа для статического управления информационным потоком был реализован в компиляторе Jif [Mye99], [MNZZ01]. Каждое выражение в программе имеет тип безопасности, который состоит из двух частей – обычного типа (например, int) и метки, которая описывает, как можно использовать значение. Безопасность гарантируется проверкой типов – компилятор читает программу, содержащую размеченные типы, и проверка типов гарантирует, что программа не будет содержать несоответствующие информационные потоки во время выполнения. Система типов в таком языке является системой типов безопасности, которая осуществляет политику, основанную на информационных потоках.

В работе [Smi01] рассматривается система типов, в которой все переменные программы классифицируются как *L* (открытый) или *H* (частный). Такая типизация препятствует утечке информации о переменных *H* в *L* переменные. В многопотоковом императивном языке с вероятностным планированием это формализует свойство вероятностного невлияния. Кроме того, каждой команде присваивается тип в форме *\tau_1 cmd \tau_2*; это означает, что команда оперирует только переменными уровня *\tau_1* (или выше) и имеет продолжительность, которая зависит только от переменных уровня *\tau_2* (или ниже). Также используются типы в форме *\tau cmd n* для команд, которые заканчиваются точно за *n* шагов. Утверждается, что с такой типизацией можно предотвратить утечки информации, связанные со временем, если потребовать, чтобы никакое присваивание *L*-переменной не могло следовать за командой, продолжительность которой зависит от *H*-переменных. В такой системе появляется возможность использовать *H*-переменные более гибко; например, поток, в котором обрабатываются только *H*-переменные, всегда хорошо типизирован. Семантическая непротиворечивость (soundness) такой системы типов доказана с помощью понятия вероятностной бисимуляции.

1.6.5 Другие подходы

В работе [Bla03] представляется методика автоматической верификации криптографических протоколов, основанная на промежуточном представлении протокола с помощью набора фраз Хорна (логическая программа). Эта методика дает возможность верифицировать свойства безопасности протоколов, таких как конфиденциальность и аутентификация, полностью автоматическим способом. Кроме того, получаемые с ее помощью доказательства являются правильными для неограниченного числа сеансов протокола.

Метод обоснования на основе слабейшего предусловия (Weakest Precondition reasoning) был изложен в [DIJK76] и предназначался для верификации программ. Эта методика рассматривает три компонента: состояние до выполнения инструкции программы, инструкция программы непосредственно, и цель, которая должна быть истинна после того, как инструкция выполняется. Недостаток этой методики заключается в трудности доказательства для сложных предикатов. Для длинных программ, с большим количеством целей, доказательства могут быть невозможны.

В работе, посвященной интегрированной среде CPAL-ES (Cryptographic Protocol Analysis Language Evaluation System), для верификации применяется метод «слабейшего предусловия». Поскольку криптографические протоколы имеют тенденцию быть короткими, этот метод успешно применяется для этих протоколов.

1.7 Тестирование свойств безопасности.

Существуют тонкие различия между традиционным тестированием программного обеспечения на соответствие спецификациям и тестированием безопасности в терминах целей, сферы действия, акцентов, последствий ошибок и стратегии. Более того, признанный эксперт в области безопасности Брюс Шнайер (Bruce Schneier) [Schn00] утверждает, что безопасность не имеет ничего общего с функциональностью, поэтому никакое бета-тестирование не сможет выявить недостатки безопасности. По его мнению, единственным способом получения некоторой уверенности в устойчивости системы к нападениям является тщательное ее тестирование специалистами в области безопасности.

Основная цель тестирования на соответствие программного обеспечения – это верификация корректности реализаций по отношению к спецификациям. Эффективность реализаций в значительной степени определяет рынок. Однако тестирование безопасности касается и корректности, и эффективности, т.к. показатели эффективности, такие как прочность и устойчивость, являются неотъемлемой частью любых спецификаций безопасности. В традиционном тестировании на соответствие акцент делается на тестирование реализации на соответствие функциональным спецификациям, в то время как в тестировании безопасности продукт должен быть протестирован не только на соответствие спецификациям функций безопасности, но также и на согласованность с принудительными особенностями основополагающей модели безопасности. В традиционном тестировании на соответствие верификация с использованием тестовых вариантов, которые удовлетворяют некоторым статистическим критериям покрытия, может гарантировать, что определенные дефекты будут проявляться довольно редко. Однако в тестировании безопасности желательно полное тестовое покрытие, поскольку скрытые дефекты в системе, которые были задействованы злонамеренно или без злого умысла, способны полностью разрушить поведение корректно реализованных функций. Необходимость полного тестового покрытия для тестирования безопасности может приводить к громадному количеству тестовых вариантов.

Тестирование безопасности можно, вообще говоря, классифицировать как тестирование функций безопасности и тестирование уязвимостей в безопасности. Тестирование функций безопасности включает тестирование продукта или реализации на соответствие со спецификациями функций безопасности, а также для тестирования основополагающей модели безопасности. Критерии соответствия ставят условия, необходимые для того, чтобы продукт демонстрировал желательное безопасное поведение или удовлетворял свойству безопасности. Другими словами тестирование функций безопасности должно показывать, что продукт *должен делать*. Тестирование уязвимостей в безопасности касается выявления дефектов в проекте или реализации, эксплуатация которых может нарушить безопасное поведение, хотя тестирование функций безопасности показало их корректную реализацию. Другими словами тестирование уязвимостей в безопасности должно показывать, что продукт *не должен делать*.

В настоящее время существует большое количество стандартов и рекомендаций, выработанных международным сообществом в области тестирования безопасности. Наиболее известными являются:

- документы Национального Института Стандартов и Технологии США (NIST), в частности документ [WT01], посвященный рекомендациям для организаций по тестированию безопасности сети,
- OSSTMM [OSSTMM] (Open Source Security Testing Methodology Manual) – методология тестирования безопасности,

- ISO 17799-2000 (BS 7799) – руководство, учитывающее все требования BS 7799 (и его интернационального эквивалента ISO 17799) для тестирования информационной безопасности,
- GAO FISCAM – документ FISCAM (Federal Information System Control Audit Manual) организации GAO (US General Accounting Office), который разработан для сетевой безопасности [1],
- CASPR [CASPR] (Commonly Accepted Security Practices and Recommendations) – этот документ касается тестирования безопасности в Интернете,
- OWASP [OWASP] (Open Web Application Security Project) – руководство по тестированию удаленной безопасности и аудиту Web-приложений,

Рассмотрим более подробно первые два из вышеперечисленных документов.

Руководство по тестированию сетей.

В документе Национального Института Стандартов и Технологии США (NIST) «DRAFT Guideline on Network Security Testing: Recommendations of the National Institute of Standards and Technology»[WT01], описывается методология тестирования сетевой безопасности. Этот документ не рассматривает подходы к тестированию сетевых протоколов, но дает представление о методах следующих видов тестирования безопасности распределенной сети:

- Отображение сети (Network Mapping),
- Сканирование уязвимости (Vulnerability Scanning),
- Тестирование на проникновение (Penetration Testing),
- Тест и оценка безопасности (Security Test & Evaluation – ST&E),
- Взламывание пароля (Password Cracking),
- Анализ системных журналов (Log Review),
- Проверка целостности файлов,
- Обнаружение вируса,
- Агрессивный дозвон (War Dialing).

Зачастую несколько из этих методов тестирования используются совместно, чтобы получить наиболее всестороннюю оценку состояния защиты сети. Например, тестирование на проникновение почти всегда включает отображение сети и сканирование уязвимости для идентификации уязвимых хостов и служб, которые потом будут тестироваться на проникновение. Ни одна из этих методик тестирования сама по себе не обеспечит законченную картину сети или состояния ее безопасности.

Отображение сети включает использование сканера порта для идентификации всех активных хостов, подключенных к сети, сетевых служб, работающих на этих хостах (например, протокол передачи файлов FTP и протокол передачи гипертекста HTTP), и специального приложения, исполняющего роль идентификационной службы (например, Internet Information Server и Apache для HTTP). Сканеры портов распознают активные хосты и открытые порты. Результат сканирования - исчерпывающий список всех активных хостов и служб, работающих в адресном пространстве, которое сканировал инструмент сканирования портов. Название "сетевая карта" является неправильным употреблением, поскольку сканер порта видит сеть как область одноуровневой адресации и обычно не обеспечивает никакого значимого графического представления отсканированной сети. Этот вид тестирования помогает обнаружить неавторизованные хосты, подключенные к данной сети, распознавать уязвимые службы и отклонения от разрешенных служб в соответствии с выбранной

[1] К сожалению, документ достать не удалось.

политикой безопасности, а также осуществляет подготовку для тестирования на проникновение. Хотя отображение сети, главным образом, автоматизировано, все же эта деятельность требует достаточно высокого уровня человеческой экспертизы для интерпретации результатов. Кроме того, она вносит изменения в характеристики сетевых операций, используя ту же полосу пропускания и замедляя времена ответов в сети. Однако, отображение сети обеспечивает средство для организации управления пространством IP-адресов сети и гарантирует, что данная конфигурация хостов выполняет только санкционированное сетевое обслуживание. Отображение сети следует проводить ежеквартально. С целью уменьшения влияния этой деятельности на сетевые операции необходимо тщательно отбирать ПО, которое подвергается сканированию. Отображение сети можно проводить в часы наименьшей загрузки сети (например, после закрытия банка).

Сканирование уязвимостей. Сканеры уязвимостей по существу являются сканерами портов следующего уровня. Сканер уязвимостей идентифицирует не только хосты и открытые порты, но также автоматически обнаруживает любые связанные с ними уязвимости вместо того, чтобы полагаться на человеческую интерпретацию результатов сканирования. Большинство сканеров уязвимостей также пытается обеспечить информацию относительно смягчения обнаруженной уязвимости. Сканеры уязвимостей могут помочь идентифицировать устаревшие версии ПО, уязвимости, возможные патчи или системные обновления, и проверить соответствие с, или отклонения от, политики безопасности сети. Для идентификации уязвимостей сканеры уязвимостей используют большие базы данных уязвимостей. Кроме того, сканеры уязвимостей могут автоматически вносить исправления и, таким образом, устранять обнаруженные уязвимости. Для осуществления этой деятельности сканер уязвимостей должен иметь права администратора на уязвимом хосте. Однако сканеры уязвимости имеют некоторые существенные слабости. Вообще, они находят только поверхностную уязвимость и неспособны выходить на всеобъемлющий уровень рисков сканируемой сети. Хотя процесс самого просмотра высоко автоматизирован, сканеры уязвимостей могут иметь высокую ложную положительную ошибочную норму (сообщая об уязвимостях даже при их отсутствии). Это означает, что для интерпретации результатов необходимо вмешательство человека. Так как сканеры уязвимостей требуют больше информации для надежной идентификации уязвимостей на хосте, чем сканеры порта, сканеры уязвимостей имеют тенденцию генерировать значительно больший сетевой трафик, чем сканеры порта. Это может иметь отрицательное воздействие на хосты или сеть, подвергающиеся сканированию, или сетевые сегменты, через которые проходит трафик сканирования. Многие сканеры уязвимостей также включают тесты на атаки типа отказа в обслуживании (DoS), которые в руках неопытного пользователя могут иметь значительные отрицательные воздействия на сканируемые хосты. Другим существенным ограничением сканеров уязвимостей является то, что они полагаются на постоянное обновление базы данных уязвимостей, поэтому перед выполнением такого сканера необходимо убедиться, что в базу данных уязвимостей внесены последние обновления. Проще говоря, сканеры уязвимостей хорошо обнаруживают известные уязвимости, однако редко помогают при наличии неизвестных дыр. Это происходит также из-за желания изготовителей сохранить высокую скорость их сканеров (обнаружение большего количества уязвимостей требует большего количества тестов, что в целом замедляет процесс сканирования). Итак, сканеры уязвимостей обеспечивают идентификацию активных хостов в сети, активных и уязвимых служб (портов) на хостах, операционных систем, уязвимостей, связанных с обнаруженными операционными системами и приложениями; выполняют тестирование на соответствие с политиками использования/безопасности для приложений на хостах; подготавливают основу для тестирования на проникновение. Сканеры уязвимостей бывают двух типов: сетевые сканеры и сканеры хостов. Сетевые сканеры используются для отображения сети и идентификации открытых портов. В большинстве случаев, эти сканеры не ограничиваются операционной системой целевых систем. Сканеры могут быть установлены на одной системе в сети и могут быстро определять местонахождение и тестировать многочисленные хосты.

Сканеры хостов должны быть установлены на каждом хосте, который будет тестироваться, и используются, прежде всего, для идентификации некорректно сконфигурированных операционной системы и приложений, а также для выявления уязвимостей. Сканеры хостов имеют высокую степень детализации и обычно требуют root-вых или административных полномочий для доступа к хосту. Некоторые сканеры хостов предлагают возможность исправлять любую неправильную конфигурацию.

Тестирование на проникновение – это тестирование безопасности, при котором делается попытка обойти особенности безопасности системы, основанные на понимании проектирования системы и ее реализации. Цель тестирования на проникновение состоит в том, чтобы выявить методы получения доступа к системе с помощью использования распространенных инструментов и методик, разработанных хакерами. Это испытание настоятельно рекомендуется для сложных или критических систем. Однако, это - очень трудоемкая деятельность и требует тщательной экспертной оценки для минимизации рисков целевой системы. Как минимум, эта деятельность может замедлить время ответа в сети из-за отображения сети и сканирования уязвимостей. Кроме того, существует вероятность повреждения системы в ходе тестирования на проникновение, и она может выйти из строя. Хотя этот риск смягчается, если такое тестирование проводят опытные тестировщики, полностью устранить его невозможно. Поскольку тестирование на проникновение нацелено на имитацию атак и использование инструментов и методик, применение которых может быть ограничено законом, федеральными инструкциями и организационной политикой, необходимо получить письменное разрешение для проведения тестирование на проникновение до его начала. Тестирование на проникновение может быть открытым или скрытым, и называется, соответственно, Синим Объединением в команду (Blue Teaming) и Красным Объединением в команду (Red Teaming). Синее Объединение в команду проводится с согласия организации, которая подвергается тестированию. Красное Объединение в команду проводится с разрешения вышестоящей организации в условиях, когда сотрудники организации не знают о проводящемся тестировании. Иногда к такому тестированию подключается независимая удостоверяющая третья сторона, чтобы осуществлять контроль над проведением тестирования на проникновение и помогать отличать реальные атаки от имитаций, проводимых группой тестирования. Blue Teaming менее дорогая деятельность, и используется чаще, чем Red Teaming. Тест на проникновение может быть создан для имитации как внутренней, так и внешней атаки. Если необходимо и внутреннее, и внешнее тестирование, сначала обычно проводится внешнее. При внешнем тестировании на проникновение, брандмауэры обычно ограничивают количество и типы трафика, которые допускаются во внутреннюю сеть из внешних источников. В зависимости от того, какие протоколы пропускаются через брандмауэры, внутренние атаки, в основном, сосредотачиваются на общеизвестных протоколах приложений, таких как FTP или HTTP. При имитации реальной внешней атаки тестировщикам не предоставляется никакой информации о целевой среде, кроме целевого адреса IP, и они вынуждены скрытно собирать необходимую информацию перед атакой. Они собирают информацию о цели нападения из публичных web-страниц, телеконференций и т.п. При этом используются сканеры порта и сканеры уязвимостей для идентификации целевых хостов. Так как в данном случае проникновение в систему осуществляется через брандмауэры, количество информации, которую при этом удается собрать, гораздо меньше, чем если бы это делалось изнутри системы. После идентификации хостов сети производится попытка дискредитации одного из них. Если она успешна, тогда ставятся под угрозу другие хосты, которые вообще недоступны снаружи. В этом смысле тестирование на проникновение является итерационным процессом. Внутреннее тестирование на проникновение подобно внешнему, за исключением того, что тестировщики находятся теперь во внутренней сети (т.е. позади брандмауэра) и им предоставлен некоторый уровень доступа к сети. Тестировщики на проникновение пробуют получить более высокий уровень доступа к сети. Тестирование на проникновение имеет четыре фазы – планирование, обнаружение, атака и составление отчета. На фазе

планирования устанавливаются цели тестирования, никакого реального тестирования не делается. На фазе обнаружения начинается реальное тестирование. Сначала производится отображение сети (сканирование портов) для идентификации потенциальных целей. Кроме того, для сбора информации о сети применяются другие методы, такие как: опрос DNS (Domain Name System), запросы InterNIC (кто есть кто), поиск Web-сервера целевой организации, поиск серверов LDAP (Lightweight Directory Access Protocol), перехват пакетов (обычно только для внутренних тестов), перечисление NetBIOS (обычно только для внутренних тестов), NIS (Network Information System) (обычно только для внутренних тестов), перехват заголовков. Во второй части фазы обнаружения проводится анализ уязвимостей, т.е. сервисы, приложения и операционные системы сканируемых хостов проверяются на наличие для них уязвимостей в базах данных (для сканеров уязвимостей, этот процесс является автоматическим). Зачастую тестировщики используют свою собственную базу данных, или публичные базы данных и проводят идентификацию уязвимостей вручную. Считается, что ручной процесс лучше подходит для выявления новых или неясных уязвимостей, хотя он намного медленнее. Выполнение атаки – это основа любого тестирования на проникновение. Делается попытка применить обнаруженные ранее потенциальные уязвимости. Если атака успешна, уязвимость верифицирована, и соответствующая защита выявлена для смягчения данного дефекта безопасности. Фаза составления отчета проводится одновременно с тремя другими фазами тестирования на проникновение. На фазе планирования разрабатываются цели тестирования, тестовые планы. На фазе обнаружения и на фазе атаки, составляются журналы проведения тестирования. По завершению каждого теста составляется всеобъемлющий отчет о тестировании, который описывает выявленные уязвимости, обеспечивает оценку риска, и дает руководство по смягчению обнаруженных слабостей. Из-за высокой стоимости и потенциального воздействия тестирование на проникновение обычно проводится ежегодно.

Тест и оценка безопасности (ST&E) **–** это экспертиза или анализ защитных мер, которые накладываются на информационную систему во время ее эксплуатации. Цели ST&E:

- вскрыть дефекты проектирования, реализации и операций, которые могут повлиять на нарушение политики безопасности,
- определить адекватность механизмов безопасности, гарантий, и других свойств, обеспечивающих выполнение политики безопасности,
- оценить степень согласованности между системной документацией и ее реализацией.

Сфера действия ST&E обычно обращается к компьютерной безопасности, безопасности соединения, безопасности излучения, физической безопасности, безопасности персонала, административной безопасности и безопасности операций. Компьютерная безопасность состоит из мер по защите системы против DoS и неавторизованного раскрытия, модификации, или разрушения системных данных. Компьютерная безопасность может быть протестирована через конфигурационное и оперативное тестирование, чтобы удостовериться, что механизмы безопасности системы реализованы и работают должным образом. Конфигурационное тестирование выполняется путем сравнения инсталлированной конфигурации с конфигурацией, описанной в требованиях безопасности, Оперативное тестирование обеспечивает оценку механизмов безопасности в операционной среде, чтобы определить, обеспечивают ли данные механизмы политику безопасности. Оперативное тестирование осуществляется путем выполнение предопределенных тестов. Эти тесты устанавливают основу для конфигурационного управления и тестирования системы. Безопасность коммуникации включает меры для предотвращения несанкционированного доступа через передачу данных. Тестирование гарантирует, что линии связи защищены на уровне, соразмерном с уровнем чувствительности передаваемых данных. Кроме того, тестирование коммуникации должно определить, не вводит ли подключение системы новые уязвимости в сеть. Безопасность излучения, физическая безопасность, безопасность

персонала и административная безопасность не рассматриваются в данной работе. Безопасность операций – аналитический процесс, с помощью которого потенциальным противникам отказывают в информации о возможностях и намерениях системы, по возможности скрывая планирование и выполнение чувствительных деятельностей и операций. Безопасность операций в ST&E проверяется путем анализа способности систем ограничить доступ к этой информации. ST&E является очень трудоемкой деятельностью, которая обычно выполняется каждые 3 года или в случае, когда система подвергается большим изменениям. ST&E обычно выполняется для новых или существенно обновленных систем и может включать любые другие виды тестирования.

Взламывание пароля. Программы взламывания пароля используются для получения слабых паролей. Обычно пароли хранятся и передаются в зашифрованной форме, называемой хэш-формой (hash**)**. Когда пользователь входит в компьютер/систему и вводит пароль, создается его хэш-форма и она сравнивается с хранимой хэш-формой. Если они совпадают, пользователь авторизуется. Во время выполнения теста на проникновение или реальной атаки взламывание пароля использует перехваченные хэш-формы пароля, которые могут быть перехвачены во время передачи их по сети (с помощью сетевого анализатора пакетов – сниффера) или получены от целевой системы (последнее, вообще говоря, требует административного или root-ого доступа к целевой системе). Как только хэш-формы получены, автоматизированный взломщик паролей быстро генерирует хэш-формы, пока соответствие не будет найдено. Самый быстрый метод для генерации хэш-форм – это словарная атака (dictionary attack), которая использует все слова в словаре или текстовом файле. В Интернете существует много словарей, которые покрывают большинство основных и второстепенных языков, имена, популярные телешоу, и т.д. Таким образом, любое “словарное” слово, неважно как зашифрованное, становится слабым в качестве пароля. Другой метод взламывания называется гибридной атакой, который основан на словарном методе с добавлением числовых и символических знаков к словам словаря. В зависимости от используемого взломщика пароля, этот тип атаки имеет ряд вариаций – применение подстановок, добавление символов к началу и к концу слов словаря. Самый мощный метод взламывания паролей называется методом взламывания "в лоб". Такой взломщик беспорядочно генерирует пароли и их хэш-формы. Несмотря на то, что могут потребоваться месяцы, чтобы взломать пароль, теоретически все пароли являются “взламываемыми” с помощью атаки "в лоб", если имеется достаточно времени и мощности обработки. Тестировщики на проникновение и хакеры часто имеют несколько машин, на которых они могут распределить задачу взламывания пароля. Это может резко сократить отрезок времени, требуемый для взламывания сильных паролей. Сильный пароль – тот, который является длинным (больше 10 символов) и сложным (содержит символы верхнего и нижнего регистра, символы и числа). Взломщики пароля обычно запускаются раз в месяц, чтобы гарантировать правильные композиции паролей.

Анализ системных журналов. Для выявления отклонений от политики безопасности могут использоваться различные системные журналы, включая журналы системы сетевой защиты, журналы IDS (Intrusion Detection System – система обнаружения злоумышленного вторжения**)**, файлы регистрации сервера, и любые другие журналы, которые собирают контрольные данные относительно системы и сети. По существу анализ системных журналов может использоваться для проверки того, что система работает согласно политикам безопасности. Например, если датчик IDS помещен позади брандмауэра, и в его журналах отражена неавторизованная деятельность, то это означает, что брандмауэр не отвечает требованиям безопасности.

Аудит системных журналов вручную является чрезвычайно тяжелым и трудоёмким. Автоматизированные инструменты для аудита обеспечивают возможность значительно сократить время обзора и составления отчетов, в которых отражается специфическая деятельность. Обзоры системных журналов обычно проводятся еженедельно.

Проверка целостности файлов. Анализатор целостности файла вычисляет и сохраняет контрольную сумму для каждого файла, требующего защиты, и создает базу данных контрольных сумм файлов. Это дает возможность выявлять неавторизованные модификации файлов. Возможность проверки целостности файлов обычно включается в коммерческие системы обнаружения вторжения. Анализатор целостности – полезный инструмент, который не требует высокой степени участия человека, однако, чтобы его использование стало эффективным, необходимо соблюдать определенные меры предосторожности. Во время первоначального заполнения базы данных требуется, чтобы система находилась в безопасном состоянии. База данных должна храниться автономно так, чтобы злоумышленник не смог поставить под угрозу систему и скрыть свои следы, модифицируя базу данных. Анализатор целостности файла может также генерировать ложные аварийные сигналы. Каждое обновление файла или патч системы изменяет определенный файл (или файлы), и будет, следовательно, требовать обновления базы данных контрольных сумм. Поэтому, поддержание такой базы данных на современном уровне требует значительных усилий. Однако даже если анализатор целостности выполнялся только однажды (при первоначальной установке системы), это может оказать пользу при определении, какие файлы подверглись изменениям в случае подозрении на вторжение. Анализаторы целостности обычно запускаются ежедневно на системных файлах, которые могут подвергаться дискредитации. Кроме того, следует их использовать в случае подозрения на атаку для того, чтобы определить степень возможного повреждения.

Обнаружение вируса. Все организации подвергаются опасности столкнуться с компьютерными вирусами, троянскими конями и червями, если они подключены к Интернету, используют сменные носители, допускают безнадзорный доступ пользователей или используют разделяемое программное обеспечение. Компьютерный вирус – это код, который прикрепляется к некоторой компьютерной программе или документу. Как только это произошло, вирус использует некоторые ресурсы поглощенной программы или документа, чтобы копировать и прикрепить себя к другим программам и документам хоста. Злонамеренный код не ограничивается вирусами; существует несколько типов злонамеренного кода, которые обнаруживаются антивирусным программным обеспечением, даже если код, строго говоря, и не вирус. Другие категории злонамеренного кода включают червей, троянских коней и злонамеренный мобильный код. Воздействие вируса, троянских коней, червей или злонамеренного мобильного кода может быть столь же безвредно, как всплывающее сообщение на экране компьютера, или столь же разрушительно, как удаление всех файлов на жестком диске. Однако злонамеренный код создает риск выставления на показ или разрушения чувствительной или конфиденциальной информации. Существует два простых типа доступных антивирусных программ: те, которые установлены в инфраструктуре сети, и те, которые установлены на компьютерах конечных пользователей. Каждый из них имеет преимущества и недостатки, но, вообще говоря, для самого высокого уровня безопасности требуется их совместного использования. Независимо от того, какой тип программы обнаружения вируса используется, он не может предложить полноценную защиту сети, если он не имеет свежей базы данных идентификации вирусов (иногда называемую вирусными сигнатурами), которая дает возможность распознавать все вирусы. Если программа обнаружения вируса будет не самой свежей, то она не будет обнаруживать новый вирус. Чтобы обнаруживать вирусы, антивирусное программное обеспечение сравнивает содержание файлов с известными сигнатурами компьютерных вирусов, выявляет зараженные файлы, и восстанавливает их, если возможно, или удаляет их, если это невозможно. Более сложные программы кроме того еще осуществляют поиск деятельности, подобной вирусу, в попытке идентифицировать новые или мутирующие вирусы, которые невозможно распознать с помощью текущей базой данных идентификации вирусов. Пока еще не совершенная, такая система способна предоставить дополнительный уровень защиты. Файлы определения вирусов должны обновляться, по крайней мере, дважды в месяц и всякий раз, когда происходит вспышка нового вируса.

Вирусный детектор, присутствующий в инфраструктуре сети, обычно устанавливается на почтовых серверах и на (или вместе с) брандмауэрах на границе сети организации. Преимущество программ обнаружения вируса с размещением на сервере в том, что они могут обнаружить вирусы до того, как они попадут в сеть, или прежде, чем пользователь загрузит свою электронную почту. Другое преимущество обнаружения вируса с размещением на сервере в том, что все вирусные детекторы требуют частого обновления для того, чтобы их использование оставалось эффективным. К сожалению программы, размещенные на сервере, оказывают отрицательный эффект на производительность сети.

Другой тип программного обеспечения обнаружения вируса устанавливается на компьютерах конечных пользователей. Это программное обеспечение обнаруживает злонамеренный код в сообщениях электронной почты, на дискетах, жестких дисках, в документах и т.п.. Такие детекторы иногда также обнаруживают злонамеренный код, приходящий с Web-сайтов. Этот тип программ обнаружения вируса оказывает меньшее воздействие на производительность сети, но в основном полагается на конечных пользователей для обновления их сигнатур, которые не всегда надежны. Кроме того, антивирусная защита конечного пользователя не может защитить сеть от всех вирусных угроз.

Агрессивный дозвон. В хорошо сконфигурированной сети слабым местом является присутствие неавторизованных модемов. Эти неавторизованные модемы обеспечивают возможность обойти большинство или все меры безопасности, которые принимаются для того, чтобы воспрепятствовать доступу в сеть неавторизованных пользователей. Существует несколько доступных пакетов программ, которые позволяют хакерам и сетевым администраторам набирать большие блоки номеров телефонов в поисках доступных модемов. Этот процесс называют агрессивным дозвоном. Компьютер с четырьмя модемами может набрать 10 000 номеров в течение дня. Некоторые программы агрессивного дозвона даже делают попытку некоторого ограниченного автоматического взлома, когда модем обнаружен, и все обеспечивают сообщение об обнаруженных номерах с модемами. Хотя атаки через Интернет и преобладают, все-таки много атак осуществляется через неавторизованные модемы. Увеличение количества ноутбуков обостряет эту проблему, т.к. большинство из них работают через модем. Прямой доступ в сеть через авторизованный модем позволит злоумышленнику обойти все меры безопасности, и, скорее всего такая атака останется необнаруженной.

Методология тестирования безопасности.

Методология тестирования безопасности – Open Source Security Testing Methodology Manual (OSSTMM), разработанная Питом Герцогом (Pete Herzog), – стала интернациональным открытым стандартом с тех пор, как она стартовала в январе 2001. В настоящее время Пит Герцог возглавляет институт ISECOM (Institute for Security and Open Methodologies). Несмотря на открытость этой методологии, она широко используется известными организациями, такими как U.S. Treasury Department, Home Depot, Verisign, и IBM. Краеугольными камнями методологии OSSTMM являются: фокусировка на технических деталях именно тех элементов, которые должны быть протестированы, и рекомендации о том, как и когда выполнять различные типы тестов по безопасности. OSSTMM обеспечивает методологию тестирования для следующих шести категорий безопасности: информационная безопасность, безопасность процесса, безопасность интернетовских технологий, безопасность коммуникации, беспроводная безопасность, и физическая безопасность.

Методология разбита на разделы, соответствующие этим категориям, а разделы состоят из модулей и задач. Разделы – это определенные области в схеме безопасности, которые частично перекрывают друг друга. Модули являются потоком (технологическим процессом) методологии от одного раздела безопасности к другому. Каждый модуль имеет ввод и вывод. Ввод – информация, используемая для выполнения отдельной задачи. Вывод – результат

завершенных задач. Вывод может или, возможно, не может быть анализируемыми данными (также известными как сведения), которые служат вводом для другого модуля (модулей) или раздела. Технологический процесс (поток) методологии движется от начального модуля до завершения конечного модуля. Методология делает различие между сбором данных и верификационным тестированием этих собранных данных. Поток может также определить конкретные точки во времени извлечения и вставки этих данных. Автор считает, что при определении методологии тестирования, очень важно не ограничивать творческий потенциал тестировщика, вводя стандарты, такие формальные и неумолимые, что от этого страдает качество тестирования. Важно оставить отдельные задачи открытыми для некоторой интерпретации, где точное определение заставит методологию страдать при введении новой технологии. Каждый модуль имеет взаимосвязи с предыдущим и последующим. Каждый раздел имеет взаимоотношения со всеми другими разделами. В целом, тестирование безопасности начинается с ввода, который содержит, в конечном счете, адреса систем, которые должны быть протестированы. Тестирование безопасности заканчивается фазой анализа и составлением завершающего отчета. Эта методология не затрагивает форму, размер, стиль или содержание завершающего отчета, и не определяет, как данные должны анализироваться. Это возлагается на тестировщика или организацию.

Разделы составляют целостную модель безопасности, разделенную на управляемые, тестируемые слои. Модули – это переменные тестов в разделах. Модуль требует ввода для выполнения своих задач и задач модулей других разделов. Задачи являются тестами безопасности, которые зависят от ввода модуля. Результаты задач могут быть немедленно проанализированы, и далее они будут действовать как обработанный результат, или оставлены необработанными. В любом случае, они считаются выводом модуля. Этот вывод часто является вводом для следующего модуля или в определенных случаях, когда появляются только что обнаруженные хосты, может быть вводом для предыдущего модуля.

В методологии OSSTMM модулями безопасности Интернета являются:

- обследование Интернета,
- сканирование портов,
- идентификация системы,
- идентификация сервисов,
- исследование и верификация уязвимостей,
- тестирование интернетовских приложений,
- тестирование маршрутизации,
- тестирование брандмауэров,
- тестирование систем обнаружения вторжения (IDSs),
- взламывание пароля,
- тестирование отказа в обслуживании (DoS),
- тестирование мер сдерживания (Containment Measures), т.е. выявление атак.

Модулями беспроводной безопасности являются:

- тестирование беспроводных сетей,
- тестирование беспроводных коммуникаций,
- экспертиза конфиденциальности
- тестирование инфракрасных систем.

Модулями физической безопасности являются:

- тестирование контроля доступа,
- экспертиза периметра,

- экспертиза мониторинга,
- экспертиза местоположения,
- экспертиза окружения.

С таким разбиением на модули можно не согласиться, однако заслуживает внимания явное стремление автора учесть все проблемы, которые встречаются при тестировании распределенных систем.

Тестирование и мониторинг безопасности обычно осуществляется либо центральными мониторами безопасности, либо анализаторами целостности, расположенными на хостах.

Центральный монитор безопасности отвечает за тестирование безопасности и мониторинг сети. При отказе этого центрального монитора безопасности система становится неспособной к выполнению тестирования безопасности и защите сети от атак. Очевидно, что такой подход ("командный центр") к тестированию безопасности и мониторингу не очень отказоустойчив, т.к. отказ аппаратуры центрального компьютера выводит из строя управление безопасностью всей сети. К тому же реализации централизованного монитора безопасности плохо масштабируются. При возрастании загрузки в сети производительность такого централизованного монитора падает, интервалы тестирования безопасности увеличиваются, в результате чего расширяется окно возможностей, которым может воспользоваться злоумышленник для проникновения в систему.

Анализаторы целостности, расположенные на хостах, постоянно находятся на проверяемом узле и сообщают об изменениях в критических системных файлах или о подозрительной деятельности центральному монитору или передают результаты тестов в остальную часть сети. Мониторы на хостах могут работать как фоновые демоны или как плановые процессы. Неавторизованный пользователь, который получает управление над таким компьютером, может действовать злонамеренно, при этом другие части системы будут убеждены в надежности этого хоста, когда безопасность системы фактически поставлена под угрозу. Злоумышленник может блокировать демоны безопасности, утилиты сообщений и уничтожить аудиторские следы. Таким образом, невозможно полностью полагаться на компьютер в вопросе диагностирования его с помощью анализатора целостности. В отличие от централизованного мониторинга безопасности тестирование безопасности на хостах является отказоустойчивым для аппаратных отказов, но не для нарушений безопасности, поскольку злоумышленник может блокировать мониторы безопасности на хостах.

С целью преодоления многих недостатков централизованного управления безопасностью и мониторов безопасности на хостах был предложен подход, основанный на использовании мобильных агентов для тестирования безопасности [Kar98].

Формальные методы широко применяются для доказательства свойств безопасности: криптографические транзакции, сообщения, ключи и случайные числа в моделях обычно представляются абстрактными сущностями, которые могут быть параметрами абстрактных операций, таких как шифрование или хеширование, и часть реальных сообщений, возможно, выпадает из рассмотрения. Кроме того, т.к. модель безопасности обычно разрабатывается независимо от реализации (главным образом, после реализации, хотя это и нежелательно), из корректности модели безопасности еще не следует, что реализация безопасна. Уверенность в правильности реализации может быть достигнута только исчерпывающим тестированием. Тестирование на дыры в безопасности обычно ограничивается тестированием на проникновение. Так называемая "tiger-team" [LTR99], группа экспертов вручную пытается взломать систему. По существу эта группа имитирует поведение хакеров. Или для поиска известных уязвимостей используются инструменты, такие как SATAN (Security Administrator Tool for Analyzing Networks). Такой подход не удовлетворителен, поскольку его успех в значительной степени зависит от мастерства группы tiger-team или от знаний, заложенных в инструмент, который не рассматривает специфические для приложения требования

безопасности. Отсутствие объективного критерия для оценки адекватности тестирования на проникновение приводит к неопределенности в надежности тестируемой программной системы, для которой тестирование на проникновение не вскрыло никаких дефектов в безопасности. И все-таки на данный момент тестирование на проникновение остается наиболее часто применяемым методом для тестирования безопасности.

Среди других подходов к тестированию безопасности можно отметить формальные методы на основе спецификаций; метод, основанный на синтаксическом тестировании [KLT00]; метод тестирования, основанный на свойствах, который сочетает формальное тестирование с неформальной верификацией [FB97]; внесение неисправностей (fault injection) [DM98]; использование мобильных агентов [Kar98]. Все эти подходы ограничиваются выявлением специфических дефектов в безопасности, а широко известные методы тестирования не адаптированы к решениям проблем безопасности. К тому же формальные методы страдают присущими им трудностями в специфицировании требований к системе и в применении проверки соответствия модели системы и ее реализации.

1.7.1 Тестирование на основе спецификаций

При тестировании на основе спецификаций, тестовые последовательности создаются из абстрактной спецификации системы, и затем используются для верификации реализации. Для критических с точки зрения безопасности систем создание тестов, которые с большой вероятностью обнаруживают возможные уязвимости, особенно трудно, поскольку это обычно требует создания тонких и сложных сценариев выполнения и рассмотрение специфичных для этой области понятий, таких как криптография и случайные числа.

В работе [FB97] описывается применение метода тестирования, основанного на свойствах, для тестирования безопасности. Как правило, код, относящийся к безопасности, составляет малую часть функциональности программы. Для тестирования безопасности предлагается тестировать только код, который отвечает за функционирование безопасности, и не тестировать программу целиком. Этот подход обеспечивает методологию для тестирования тонких свойств исходного кода, дает специфическую и абсолютную метрику для успешного тестирования этих свойств. Успешный тест подтверждает, что свойство не нарушено; и если эти свойства формируют политику безопасности системы, тогда система безопасна. В процессе тестирования используется модель безопасности системы, а также библиотеки основных дефектов. После тестирования считается что, в целевой программе отсутствуют дефекты, содержащиеся в библиотеках.

Работа [US01] описывает подход обнаружения вторжения на основе спецификаций. Спецификации используются как базис для обнаружения атак. Авторы считают, что этот подход является перспективной альтернативой для тестирования безопасности, которая комбинируют сильные стороны обнаружения неправильного употребления (точное обнаружение известных атак) и обнаружения аномалии (способность обнаруживать новые атаки). Однако вопрос о том, будет ли этот подход быть применяться на практике, остается открытым. Приводится описание экспериментального применения данного метода. Эксперимент показал, что эффективная система обнаружения вторжения на основе спецификаций может быть разработана с умеренными затратами. Они также показывают, что этот подход хорошо справляется как с обнаружением известных, так и неизвестных атак, показывая при этом очень низкий процент ложных срабатываний. Центральным понятием метода является наблюдение событий, которые отклоняются от нормы. В качестве спецификационного языка используется разработанный авторами язык спецификации поведенческого мониторинга BMSL [SU99].

В работе [WJ02] описывается исследование, направленное на генерацию тестовых последовательностей для систем транзакций из формальной модели безопасности. Исследование проводится с поддержкой CASE-инструмента AUTOFOCUS. Тестовые

последовательности определяются по отношению к требуемым свойствам безопасности системы, с применением мутации системной спецификации и сценариев атак. Далее абстрактные тестовые последовательности конкретизируются, для того чтобы их можно было применять к существующим реализациям. Тестирование реализации на уязвимости требует построения таких тестовых последовательностей из модели безопасности, которые покрывают возможные нарушения требований безопасности. Можно использовать структурные критерии покрытия, такие как покрытие состояний или покрытие переходов модели [OXL99], и ограничиться только теми, которые помечены как "критические", но при этом не принимаются во внимание требования безопасности. Трудность с определением критериев покрытия, связанных с требованиями безопасности, состоит в том, что эти требования являются, главным образом, универсальными свойствами. Требование безопасности Φ_i можно использовать только для проверки модели, но не реализации. Если находится трасса, удовлетворяющая $\neg\Phi_i$, тогда модель нарушает требование, и должна быть откорректирована. Иначе, Φ_i не может использоваться для выборки подходящих трасс, поскольку все трассы удовлетворяют Φ_i. В этом случае авторы предлагают использовать мутационное тестирование [Off95] с применением методик внесения неисправностей (fault injection) [VM98]. В мутационном тестировании, ошибки вводятся в программу (приводя к ряду мутантов), и качество тестового набора измеряется его способностью отличить мутантов от оригинальной программы (чтобы "уничтожить" мутанты). Внесение неисправностей работает подобным образом, но часто также используется для оценки надежности. Введенные ошибки могут соответствовать ошибкам в реализации или атакам, приводящим к таким ошибкам. Для некоторого перехода t функция мутации применяется к его предусловию, или к одному из его результирующих выражений, или к его постусловию, что приводит к множеству мутированных переходов t'_j. Для тестирования безопасности функция мутации должна быть основана на общих программных ошибках, с большой вероятностью приводящих к уязвимостям, таких как опущение правдоподобных проверок или неправильное использование идентичности. Кроме того, в модели должна приниматься во внимание криптография, приводящая к мутациям, соответствующим путанице ключей или секретов, или опущение, или неправильно реализованной верификации аутентификационных кодов.

Эта работа является первой, применившей формально сгенерированные тестовые последовательности для тестирования безопасности, если не считать работы [JW01], касающейся тестирования брандмауэров.

Комбинированный подход Model&Interface-driven. В работе [CB04] представлен подход (и инструментальные средства) для автоматизированного тестирования функций безопасности, который использует формальную модель поведения, расширенную спецификациями интерфейса. Разработка инструмента TAF-SFT и его применение к тестированию функций безопасности сложной коммерческой СУБД (системы управления базами данных) показало, что и модель, и процесс генерации тестов являются масштабируемыми. К основным недостаткам этого подхода можно отнести то, что от разработчика модели требуется детальное знание семантики функций безопасности, и сложность отображения информации, которая может возникнуть, если тестируемый продукт имеет сложные интерфейсы. Основополагающей рамочной концепцией служит TAF (Test Automation Framework) [Saf00], которая автоматизирует процесс тестирования системы. Процесс тестирования включает разработку функциональной модели, анализ модели, автоматизированная генерация тестового кода, автоматизированное выполнение тестов и анализ результатов. Для написания спецификаций используется формальный язык SCR (Software Cost Reduction) [HKLB98]. Инструмент TAF-SFT применяет TAF к тестированию функций безопасности. На основе результирующей поведенческой спецификационной модели TAF-SFT генерирует тестовые векторы. Затем поведенческая модель и тестовые векторы комбинируются со спецификациями интерфейсов продукта для того, чтобы

автоматически сгенерировать тестовые драйверы (код выполнения тестов). Тестовый код выполняется над тестируемым продуктом, результаты прогона тестов сравниваются с ожидаемыми результатами, после чего генерируется тестовый отчет. Качество системы генерации тестов основано на том, насколько хорошо сгенерированные тесты удовлетворяют выбранному критерию покрытия. Для инструмента TAF-SFT в качестве критерия покрытия был выбран критерий покрытия модели, т.е. генератор тестовых векторов будет генерировать, по крайней мере, один тестовый вектор для каждого маршрута модели. Хотя в статье описывается применение этого инструмента к тестированию безопасности коммерческой СУБД, а именно к тестированию функции контроля доступа, авторы утверждают, что TAF-SFT может применяться к тестированию любых функций безопасности, которые могут быть смоделированы на основе опубликованных интерфейсов продукта. Инструмент TAF-SFT применялся для моделирования и генерации тестов также и для таких функций безопасности, как генерация аудита, управление безопасностью, идентификация и аутентификация, управление сеансами.

1.7.2 Другие подходы

Использование мобильных агентов. В работе [Kar98] описывается методология (и инструмент на ее основе), которая использует автономные мобильные агенты для тестирования безопасности распределенных сетей.

Способность мобильных агентов автономно перемещаться по сети дает новый уровень отказоустойчивости для тестирования безопасности. Перемещение мобильных агентов по сети и перенос инкапсулированных данных дает возможность скрыть данные, код, и информацию, относящуюся к безопасности, от потенциальных злоумышленников. Кроме того, тестирование безопасности сети становится неуязвимым относительно единственной точки отказа, т.е. оно может продолжаться, даже если происходит сбой отдельных узлов или они становятся недоступными. Злоумышленнику труднее блокировать тесты безопасности, если они распределены в сети.

С помощью мобильных агентов легко масштабировать тестирование безопасности, т.к. при добавлении новых компонентов к сети достаточно диспетчеризации или клонирования новых агентов для разделения рабочей нагрузки. Поддержка и обновление набора тестов по безопасности может также производиться с помощью диспетчеризации или клонирования новых агентов и отказываясь или избавляясь от старых агентов.

Мобильные агенты могут использовать параллелизм, присущий большим сетям, для того, чтобы улучшить производительность централизованного мониторинга безопасности с помощью распределения рабочей нагрузки тестирования безопасности в сети.

С помощью мобильных агентов можно осуществлять адаптивное тестирование. Появление агента в узле может быть связано со временем его последнего визита в данный узел, или с получением сообщений от другого агента, или управляться неким предопределенным расписанием, основанным на загрузке сети. Агенты могут также самостоятельно решать, какой маршрут выбрать и какие действия совершать на основании собранных данных из различных узлов, которые они посетили. Если злоумышленник заблокировал некий узловой хост, агент может перейти на другой хост и сообщить об этой проблеме. Агенты могут быть адаптивно назначены на различные задачи, охватывающие диапазон обязанностей от обнаружения вторжения и диагностики до реконфигурации и восстановления. Например, если агент обнаруживает подозрительную деятельность одного компьютера и уведомляет об этом остальную часть сети, другие компьютеры могут решать, лишить ли его прав, которые он имел ранее, отказать ли ему в определенных сетевых сервисах, или реконфигурировать сеть, до тех пор, пока подозрительный узел не вернется в безопасное состояние.

Мета-агенты могут контролировать, координировать, отказываться от агентов и рассылать агентов. Агент может быть прикреплен к определенному хосту в сети или к некоторому набору хостов для их тестирования. Как только агент завершает свои тесты, это может послать сообщения, чтобы вызвать другие агенты для продолжения тестирования. Агенты могут передавать информацию мета-агентам о достижении ими своих адресатов и завершении своих задач. Агенты могут передавать "сигналы тревоги", чтобы уведомить других агентов о потенциальных проблемах безопасности. Все взаимодействия обрабатываются прозрачно прокси-серверами. Агенты могут иметь несколько прокси-серверов, т.к. отказ единственного хоста с прокси-сервером отключил бы связь с агентом, который движется по сети.

Агенты конструируются в виде легковесных процессов так, что каждый процесс тестирует единственную уязвимость. Когда обнаруживается новая уязвимость, и тесты для этой уязвимости разработаны, новые агенты добавляются к набору тестов. Поскольку все время происходят изменения в конфигурации системы, от некоторых агентов приходится отказываться, если в них больше нет необходимости. Наборы тестов настраиваются для каждого отдельного узла в зависимости от его конфигурации. Легковесная архитектура агента делает набор тестов хорошо реконфигурируемым, особенно для гетерогенных сред.

Использование мобильных агентов для тестирования безопасности и мониторинга сети вводит новые проблемы безопасности и уязвимости. Многие качества агентов делают их идеальным механизмом атаки. Хосты должны быть защищены от злонамеренных агентов, агенты должны быть защищены от злонамеренных хостов, и агенты должны также быть защищены от других злонамеренных агентов. Атаки агентов могут включать рассылку не по адресу (spamming), получение доступа путем обмана (spoofing) и атаки отказа в обслуживании. Недостатком тестирования с помощью мобильных агентов является то, что наборы тестов получаются довольно больших размеров, а манипулирование многочисленными легковесными процессами требует дополнительных взаимодействий, повышая накладные расходы. К тому же среда управления агентами должна быть установлена на каждом хосте.

Адаптивный анализ уязвимостей (AVA – Adaptive Vulnerability Analysis), разработанный Восом (Voas) и др. [VM98], для количественной оценки информационной безопасности и живучести системы. Этот метод можно отнести к методике внесения неисправностей. Подход предполагает выполнение программного обеспечения в условиях имитации злонамеренных и неумышленных атак, которые принадлежат различным классам угроз. Неисправности вносятся во внутреннее состояние выполняющегося приложения с помощью искажения потока данных и внутренних состояний переменных приложения. С помощью метода AVA невозможно имитировать атаки, которые не затрагивают внутренние состояния приложения. Недостатком метода является семантический разрыв между атаками во время использования приложения и отклонениями, которые вызываются во время тестирования. Другими словами, из знания того, что приложение упадет при определенных условиях, трудно установить, какие атаки соответствуют конкретному отказу. Это затрудняет проводить оценку обоснованности отклонения.

Внесение неисправностей. Существует большое количество работ, посвященных исследованиям на эту тему. Здесь рассматривается одна их них. Работа [DM98] рассматривает проблему тестирования безопасности как проблему тестирования свойств отказоустойчивости программной системы. Каждое отклонение от нормальной работы рассматривается как неисправность, а терпимость к таким неисправностям со стороны системы безопасности – как ее отказ в устойчивости к этим неисправностям. В отличие от подхода, предлагаемого в [VM98], неисправности вносятся не в приложение, а среду окружения. Изменениям подвергаются атрибуты сущностей окружения и входных воздействий, которые получает приложение из окружения. После внесения неисправностей в тестируемую систему проводится наблюдение за поведением системы. Отказ,

выражающийся в не распознавании неисправностей, – это индикатор потенциального дефекта в безопасности системы. Метод можно рассматривать как дополнение к подходу [VM98].

Анализ уязвимостей с помощью синтаксического тестирования. Работа [KLT00] представляет метод анализа уязвимости **с** помощью синтаксического тестирования [Beiz90] и фундаментальные принципы для инструмента такого анализа. Предлагаемый метод касается только уязвимостей, а не корректного поведения программного обеспечения. Тестирование проводится путем подачи исключительных входных воздействий на программное обеспечение и наблюдения за аспектами безопасности в результирующем поведении. Метод эффективен для выявления дефектов, которые появились при реализации программного обеспечения, и имеет много приложений. С помощью этого подхода могут быть найдены некоторые уязвимости программных компонентов без их тестирования на корректность. Авторы полагают, что такой подход требует меньше затрат по сравнению с исчерпывающим тестированием.

1.7.3 Инструменты для тестирования свойств безопасности

В настоящее время организации, как правило, тестируют безопасность своих систем, брандмауэров и сетей с помощью коммерчески доступных инструментов для анализа уязвимостей таких, как STAT Scanner, ISS Scanner, Cybercop, формируя так называемую “tiger team”, которая имитирует действия хакеров для нападения на сеть предприятия (например, тестирование на проникновение). Другой альтернативой для тестирования безопасности распределенных систем является применение инструментов, в основе которых лежат формальные методы. Для тестирования свойств безопасности и мобильности применяются как специально разработанные для этих целей инструменты, так и существующие инструменты тестирования общего назначения.

Первые инструменты для анализа протоколов защиты были основаны на модели Долева-Яо или на некоторых ее вариантах. К таким инструментам относятся Interrogator [MCF87], NRL Protocol Analyzer [Mea96] и инструмент Longley-Rigby [LR92]. В большинстве таких инструментов применялся метод исследования пространства состояний, в котором пространство состояний сначала определяется, а затем с помощью инструмента устанавливается, существуют ли в этом пространстве маршруты, по которым злоумышленник может провести атаку. В некоторые инструменты были включены техники индуктивного доказательства теорем, как, например, в NRL Protocol Analyzer, для того, чтобы показать, что размер рассматриваемого пространства достаточен для гарантирования безопасности. Применение этих инструментов в конце 80-х – начале 90-х годов к анализу протоколов защиты дало хорошие результаты, т.к. с их помощью были выявлены многие дефекты, которые были пропущены специалистами, создававшими протоколы защиты.

Дальнейшие работы в области создания новых и использования уже существующих инструментов для исследования пространства состояний и доказательства теорем, основанные на модели Долева-Яо, получили новый толчок после того, как Лоу (Lowe) продемонстрировал применение FDR – анализатора моделей универсального назначения – для обнаружения атаки посредника (man-in-the-middle) на протокол Нидхама-Шредера [Low96]. Были проведены работы, посвященные применению в рассматриваемой области известных анализаторов моделей Murϕ [MMS97] и ASTRAL [DK97], а также инструментов для доказательства теорем – PVS [DS97] и Isabelle [Pau98]. Усилия, которые были сделаны по применению анализаторов моделей универсального назначения для верификации протоколов защиты, таких как Murϕ [MMS97] и FDR [Low96], привели к пониманию того, что анализаторы моделей универсального назначения слабо приспособлены к моделированию злоумышленника. Модель нуждалась в том, чтобы хранить информацию о том, какие сообщения известны злоумышленнику, а какие нет. В обычной структуре модели

Крипке каждое такое сообщение соответствует атомарному высказыванию, которое является истинным тогда и только тогда, когда злоумышленнику известно данное сообщение. Поскольку число сообщений бесконечно, модель становится также бесконечной. Ясно, что нужно как-то ограничить модель, чтобы сохранять только конечное подмножество всех возможных сообщений. Встает вопрос о том, какие сообщения должны быть включены в модель, а какие нет. По существу ответственность за конечно-автоматное описание модели злоумышленника возлагается на пользователя. Этот процесс еще более усложняется, если рассматриваются возможные взаимодействия между различными протоколами, т.к. становится совсем непонятным, какие сообщения являются более важными.

Позднее были разработаны специализированные для протоколов анализаторы моделей Athena [SBP01], BRUTUS [Mar01], [Hui99]. Широкие возможности методики проверки модели привели также к использованию для анализа протоколов специализированных инструментов, изначально предназначавшихся для других приложений [DFG99].

NRL Protocol Analyzer

NRL Protocol Analyzer [Mea96] – это инструмент для анализа свойств безопасности криптографических протоколов, который основан на формальных методах. Этот инструмент осуществляет автоматическую генерацию инвариантов с целью ограничить потенциально бесконечное пространство состояний. Полученное пространство исследуется на возможность проведения атак на протоколы и для доказательства безопасности протоколов, даже при потенциально неограниченном числе выполнений протокола или неограниченном числе атак злоумышленника. Протоколы в Анализаторе специфицируются как конечные автоматы, один из них моделирует злоумышленника, который согласно модели Долева-Яо способен читать, изменять или уничтожать записи в потоке информации, а также совершать криптографические операции и взаимодействовать с некоторыми законными пользователями системы. Пользователь Анализатора пытается определить, является ли протокол безопасным с помощью специфицирования небезопасного состояния. Такое состояние специфицируется не только в терминах значений переменных локального состояния и терминах, известных злоумышленнику, но и в терминах последовательности событий, которые должны или не должны произойти. Например, можно запросить осуществить поиск состояния, в котором один и тот же ключ принят дважды принципалом (два события происходят), или состояние, в котором отвечающая сторона B принимает некий ключ для взаимодействия с инициатором A, в то время как A не инициировал этот протокол. Анализатор начинает движение из этого состояния и проходит в обратном направлении все маршруты в пространстве состояний, которые заканчиваются либо начальным состоянием, либо недостижимым состоянием. Таким образом, Анализатор можно использовать для доказательства того, что никакая сторона не будет аутентифицирована некорректно, но не для того, чтобы доказать, что аутентификация завершается всегда.

Анализатор не делает никаких предположений об ограничениях на число выполнений протокола, на количество принципалов, на число интерливинговых выполнений, или на количество применяемых криптографических функций. Все это приводит к бесконечности пространства состояний. Однако Анализатор обеспечивает средства для специфицирования и доказательства индуктивных лемм о недостижимости бесконечных классов состояний, что позволяет во многих случаях проводить исчерпывающее исследование пространства состояний.

Анализатор снабжен языком темпоральной логики NPATRL (NRL Protocol Analyzer Temporal Requirements Language) [SM93], позволяющим специфицировать требования к протоколу в терминах желательной или нежелательной последовательности событий.

NRL Protocol Analyzer [Mea96] и во многом похожий на него Interrogator [MCF87] были первыми инструментами для анализа протоколов защиты, которые можно отнести к инструментам для исследования пространства состояний в обратном направлении. Движение в пространстве состояний начиналось из небезопасного состояния, и с помощью

инструмента делалась попытка обнаружить состояние, из которого существовал маршрут в это небезопасное состояние, и которое бы удовлетворяло начальной конфигурации. Такие инструменты имели дело с бесконечными моделями и должны были иметь окраску теорем-пруверов для доказательства корректности редукции пространства состояний. Во время верификационного процесса требовалось вмешательство пользователя, кроме того, этот процесс не всегда можно было завершить.

FDR

Лоу (Lowe) [Low96] исследовал применение FDR к анализу криптографических протоколов, специфицированных в CSP (Communicating sequential processes) [Hoa80]. Каждый агент, выполняющий протокол, моделируется с помощью процесса CSP, который находится либо в состоянии ожидания сообщения, либо посылает сообщение. Каналы используются для взаимодействия между процессами (т.е. между участниками протокола). Кроме того, каналы используются для моделирования злоумышленника и для поддержки трека важных событий в протоколе, таких как точки фиксации. После появления компилятора Casper [Low97] конструирование злоумышленника стало автоматизированным. Злоумышленник истолковывается как параллельная композиция нескольких процессов, по одному на каждый факт или сообщение, из которых злоумышленник может получить какие-либо сведения о выполнении протокола. Каждый процесс имеет два состояния, в одном он знает сообщение, в другом – нет. Каждый из процессов может сгенерировать ряд событий. Casper создает спецификационный процесс для верификации, который затем используется инструментом FDR для проверки того, что протокол, выполняющийся параллельно со злоумышленником, является уточнением рассматриваемого спецификационного процесса. Под безопасным спецификационным процессом понимается такой процесс, в котором отсутствуют последовательности событий, соответствующие получению злоумышленником секретных сведений.

Murϕ

В работе [MMS97] описывается применение универсального анализатора моделей Murϕ к анализу криптографических протоколов. В Murϕ состояние системы определяется с помощью набора глобальных переменных состояния, включая разделяемые переменные, которые используются для моделирования взаимодействия. Правила, связанные с переходами, описывают, как подлинные агенты осуществляют переход между состояниями, и как новые сообщения поступают в сеть. Поведение злоумышленника также описывается подобными правилами. Эти правила авторами не описываются, только упоминается, что описание злоумышленника выглядит довольно сложным. Спецификация протокола дается как инвариант на достижимых глобальных состояниях системы. Спецификация безопасности авторами не приводится. Поскольку поддержка трека знаний каждого агента вещь достаточно трудная при таком подходе, скорее всего она потребует нетривиального расширения модели.

Athena

Инструмент Athena [SBP01] является анализатором моделей, специально разработанным для протоколов защиты. Как и NRL Protocol Analyzer [Mea96], Athena работает в обратном направлении, стартуя из неисправного состояния, и пытается выявить начальные условия, необходимые для достижения этого состояния. Как и в NRL Protocol Analyzer, состояния поддерживаются как можно более абстрактными. Описывается состояние “наиболее общая ошибка”. Инструмент старается соединить это состояние с правой стороной правила. В случае NRL Protocol Analyzer – это правило переписывания, в случае Athena – это правило вывода. По мере продвижения поиска состояния конкретизируются в том смысле, что все больше переменных становятся связанными. Athena использует расширение модели пространства стрендов (strand space) [THG98] в качестве модели вычислений. Благодаря простоте и интуитивности этой модели инструмент Athena обладает неплохой эффективностью.

Isabelle

Паульсон (Paulson) исследовал применение инструмента Isabelle для доказательства корректности протокола [Pau98]. Isabelle является инструментом для доказательства теорем универсального назначения. Подобно моделям, используемым в Murϕ и NRL Protocol Analyzer, протокол в Isabelle специфицируется с помощью набора правил, которые описывают поведение подлинных участников протокола. Эти правила описывают, при каких обстоятельствах агент создает и посылает сообщение. Murϕ и NRL Protocol Analyzer используют эти правила для описания состояния, в которое переходит система, когда совершается некое действие или посылается сообщение. В противоположность этому, Паульсон использует эти правила для индуктивного определения множества возможных трасс. Каждое правило Паульсона имеет форму: “если некая трасса содержит определенные события, тогда она может быть продолжена с помощью добавления определенного нового события в ее конец”. Благодаря тому, что дается индуктивное определение для множества трасс в протоколе, доказательство корректности справедливо для произвольного числа протокольных сессий, а не только для модели с конечным числом состояний. Однако, как и в случае NRL Protocol Analyzer, этот инструмент не гарантирует завершения. К тому же не ясно, как осуществлять обратную связь на предмет получения информации о возможных ошибках в протоколе, для которого доказательство корректности не проходит. Несмотря на то, что с помощью Isabelle можно верифицировать более сильные утверждения о протоколе, все-таки метод проверки модели больше подходит для проектировщиков протоколов в целях их отладки.

Все инструменты, основанные на методе доказательства теорем, имеют общий недостаток, потому что требуют большого взаимодействия с пользователем и глубокого проникновения в суть проблемы при верификации протоколов.

Revere

Revere [Kin99] является инструментом для автоматизированного анализа протоколов, который в качестве формализма применяет теорию логику BAN [BAN90]. С помощью подхода, названного авторами “генерация теории” (theory generation), инструмент позволяет автоматически закрыть разрывы, присущие логике BAN, между “идеализированным” и конкретным протоколами. Поскольку этот подход является разновидностью метода доказательства теорем, данный инструмент дает возможность доказывать положительные утверждения о протоколах, но не обеспечивает контр примеры. Кроме того, благодаря тому, что инструмент имеет дело с простыми логиками, обеспечивается такой же уровень автоматизации, которым обладают анализаторы моделей.

BRUTUS

Инструмент BRUTUS [Mar01] был разработан для верификации протоколов защиты, и по существу является анализатором моделей. BRUTUS позволяет моделировать и анализировать совместно несколько различных протоколов. При его разработке была сделана попытка выделить злоумышленника из модели и закодировать его как набор правил переписывания, которые можно применить к сообщениям во время выполнения протокола (или возможно во время выполнения иного протокола). От ограничения на количество применяемых правил переписывания разработчику не удалось избавиться, однако этот инструмент дает возможность не указывать заранее, о каких сообщениях модель должна сохранять информацию. BRUTUS имеет две компоненты. Одна компонента осуществляет исследование пространства состояний, другая является механизмом, вырабатывающим сообщения, что позволяет моделировать действия злоумышленника. Кроме того, BRUTUS снабжен темпоральной логикой знаний, с помощью которой можно описывать свойства безопасности. Эта логика позволяет специфицировать сведения о том, что не должен знать злоумышленник, и что другие участники должны или не должны знать. Для редукции пространства состояний устанавливается ряд симметрий, которые являются общими для всех

моделей протоколов защиты. Кроме того, для сокращения исследуемого пространства состояний применяется метод редукции частичного порядка, суть которого в том. Что относительный порядок некоторых пар действий не существенен для корректного описания модели.

Автор отмечает, что поскольку модели, с которыми работает инструмент, всегда являются абстрактными, корректность протокола доказать невозможно. Несмотря на это, инструмент очень полезен в качестве отладчика, и способен выявлять дефекты в протоколах.

AutoFocus

В работе [WJ02] описывается моделирование и тестирование систем, критических с точки зрения безопасности, с использованием инструмента AutoFocus. Этот инструмент представляет собой CASE-инструмент для графического специфицирования распределенных систем, который основан на формальном методе Focus [BS01]. AutoFocus обеспечивает симуляцию, генерацию кода, генерацию тестовых последовательностей и формальную верификацию моделируемых систем.

Системы описываются в AutoFocus с использованием статического и динамического представлений, которые концептуально подобны аналогичным представлениям в UML-RT. Для моделирования систем, критических с точки зрения безопасности, в AutoFocus были добавлены некоторые аспекты описания свойств безопасности.

Сценарии атак злоумышленника встраиваются в спецификацию системы, что стало возможным после добавления атрибутов безопасности в диаграммы SSD (System Structure Diagrams), которые подобны компонентным диаграммам UML и описывают структуру и интерфейс систем. Строится отображение, присваивающее атрибуты безопасности каждому компоненту системы и каналу, на основе которого AutoFocus автоматически производит соответствующие сценарии угроз. Поведение компоненты *Intruder* должно быть специфицировано с помощью диаграмм State Transition Diagrams или смоделировано в виде программы.

Для моделирования транзакций, критических с точки зрения безопасности, вводятся типы данных для ключей, сообщений, и добавляются криптографические операции.

Спецификация поведения моделируется с помощью конечных автоматов, которые в AutoFocus представляются в виде STD (State Transition Diagrams), т.е. STD – это пара (`Состояния, Переходы`). С переходом связывается набор входных локальных переменных и набор выражений на формальном языке Quest, которые могут содержать входные и компонентные переменные (пред- и постусловия). Кроме того, как и для SSD, каждому состоянию и переходу присваивается атрибут безопасности. Для прогонов тестовых сценариев AutoFocus использует трассы событий EETs (Extended Event Traces).

В дополнение к сценарию угроз устанавливаются требования безопасности, которые формулируются в терминах предикатов первого порядка на последовательности выполнения. В качестве альтернативы могут быть использованы формулы темпоральной логики, т.к. их можно преобразовать в такие предикаты. Для облегчения описания требований безопасности AutoFocus был дополнен библиотекой шаблонов требований безопасности различного уровня абстракции.

С помощью введенных расширений для систем, критически с точки зрения безопасности, инструмент AutoFocus позволяет автоматически генерировать тестовые последовательности из формальной модели безопасности, использовать мутации системной спецификации и сценарий атак. При тестировании конкретной реализации абстрактные тестовые последовательности подвергаются процессу конкретизации.

On-the-Fly Model-Checker

Инструмент OFMC (On-the-Fly Model-Checker) [BMV03], являющийся анализатором моделей для анализа протоколов защиты, комбинирует два метода. Первым является использование инертных (lazy) типов данных [Bas99] для простого способа построения эффективного оперативного анализатора моделей, работающего на бесконечном пространстве состояний. Второй состоит в применении символического метода для моделирования злоумышленника в модели Долева-Яо, чьи действия воспроизводятся оперативно в стиле управления по запросу (demand-driven way). Авторы формализовали применение символического метода для того, чтобы значительно сократить пространство состояний без исключения из рассмотрения каких-либо атак. Подход, названный авторами методом инертного злоумышленника, использует символическое представление с целью избежать явного перечисления возможных сообщений, которые злоумышленник может создавать. Сообщения злоумышленника представляются выражениями с переменными. Поддерживаются ограничения, описывающие, что должно быть сгенерировано и из каких знаний.

Spider

Spider [LGL03] представляет собой интегрированную среду для проверки модели протоколов защиты. В Spider протоколы описываются в терминах алгебры процессов (а именно, spy-исчисления). Взаимодействующие процессы с конечным поведением представляются некоей параллельной композицией. Каждый процесс представляет экземпляр отдельной роли в протоколе. Злоумышленник в стиле Dolev-Yao неявно определяется семантикой исчисления, поскольку среда контролирует все события, касающиеся взаимодействия. Свойства безопасности записываются в виде формул темпоральной логики линейного времени. Более конкретно, spy-исчисление имеет семантику, основанную на системах размеченных переходов (LTSs), где трассы являются темпоральной моделью, на которой определяются отношения выполнимости формул темпоральной логики. Алгоритм проверки модели осуществляет поиск в глубину, который проверяет выполнимость формулы на всех трассах, сгенерированных оперативно из типизированной версии исчисления. На этом этапе использованию типов придается окончательный вид для того, чтобы получить конечно-автоматные модели, где число сообщений, исходящих от злоумышленника, конечно. Во время выполнения получаются типизированные версии описания протокола с помощью придания каждой переменной типа из набора базовых типов. Инструмент обладает ощутимой гибкостью, поскольку среда моделирования является абстрактной, и протокол специфицируется один раз для всех случаев, т.к. не типизированное выражение в spy-исчислении допускает семантику бесконечного пространства состояний. После создания такой спецификации из нее полу-автоматизированным способом могут быть получены различные типизированные версии с семантикой конечного пространства состояний с помощью наложения ограничений на типы. Кроме того, использование логики дает возможность не фиксировать заранее класс исследуемых свойств.

CoSeC

CoSeC (Compositional Security Checker) [FG97] является инструментом для автоматической верификации некоторых композиционных свойств информационных потоков, который основан на формальной семантике. Спецификация, поступающая на вход CoSeC, пишется на языке Security Process Algebra, который был разработан на основе алгебры процессов и служит для спецификации параллельных систем, где действия принадлежат двум различным уровням конфиденциальности. Свойства, которые можно верифицировать с помощью CoSeC, выводятся из свойства невлияния (Non Interference) [GM82].

TAF-SFT

Инструмент TAF-SFT [CB04] предназначен для автоматизированного тестирования функций безопасности. Основополагающей рамочной концепцией служит TAF (Test Automation Framework) [Saf00], которая автоматизирует процесс тестирования системы. Процесс тестирования включает разработку функциональной модели, анализ модели, автоматизированную генерацию тестового кода, автоматизированное выполнение тестов и анализ результатов. Для написания спецификаций используется формальный язык SCR (Software Cost Reduction) [HKLB98]. На основе результирующей поведенческой спецификационной модели TAF-SFT генерирует тестовые векторы. Затем поведенческая модель и тестовые векторы комбинируются со спецификациями интерфейсов продукта для того, чтобы автоматически сгенерировать тестовые драйверы (код выполнения тестов). Тестовый код выполняется над тестируемым продуктом, результаты прогона тестов сравниваются с ожидаемыми результатами, после чего создается тестовый отчет. Качество системы генерации тестов основано на том, насколько хорошо сгенерированные тесты удовлетворяют выбранному критерию покрытия. Для инструмента TAF-SFT в качестве критерия покрытия был выбран критерий покрытия модели, т.е. генератор тестовых векторов будет генерировать, по крайней мере, один тестовый вектор для каждого маршрута модели. Хотя в статье описывается применение этого инструмента к тестированию безопасности коммерческой СУБД, а именно к тестированию функций контроля доступа, авторы утверждают, что TAF-SFT может применяться к тестированию любых функций безопасности, которые могут быть смоделированы на основе опубликованных интерфейсов продукта. Инструмент TAF-SFT применялся для моделирования и генерации тестов также и для таких функций безопасности, как генерация аудита, управление безопасностью, идентификация и аутентификация, управление сеансами.

Основным недостатком этого инструмента является то, что от разработчика модели требуется детальное знание семантики функций безопасности. Кроме того, возникает сложность отображения информации, если тестируемый продукт имеет сложные интерфейсы.

MAST

Инструмент MAST (Mobile Agent-based Security Tool) [CCS03] предназначен для автоматизированной поддержки безопасности всех систем в отдельном домене или во всей сети. Инструмент проверяет, произведены ли все вышедшие обновления систем в сети, а также выявляет ошибки в конфигурациях систем, которые могут привести к их уязвимостям. Инструмент MAST интегрирует две ключевые технологии – систему мобильных агентов NOMADS [SBB00] и инструменты для концептуальных отображений CmapTools [CFN03]. Концептуальные отображения Новака [Novak] представляют собой графическое представление концепций и их взаимосвязей, которое дает строгое и сжатое описание конкретной области знания. Инструменты CmapTools дают возможность работать с концептуальными отображениями в оперативном режиме. С их помощью можно создавать хранилища концептуальных отображений (которые называются моделями знаний) и разделять их в среде Интернет. Система мобильных агентов NOMADS, основанная на Java-агентах, дает высокую степень мобильности агентов в распределенной сети и обеспечивает защиту системы от различных форм атак.

2 Мобильность

Мобильные системы состоят из компонентов, которые могут перемещаться в физическом или логическом пространстве. Если компоненты, которые перемещаются, являются главными компьютерами (хостами), система проявляет физическую мобильность. Если компоненты являются фрагментами кода, говорят, что система проявляет логическую мобильность, также называемую переносимостью кода. Код по требованию, удаленные вычисления (remote evaluation), и мобильные агенты – типичные формы мобильности кода. Конечно, многие системы обладают комбинацией и логической, и физической мобильности. В данной работе под мобильностью понимается только физическая мобильность и рассматривается только этот вид мобильности.

Мобильные системы можно считать разновидностью распределенных систем, однако поскольку связь в мобильных системах осуществляется через беспроводные сети, в мобильных системах появляются еще более критичные проблемы, чем в распределенных системах. В беспроводных сетях сетевые узлы могут двигаться и подключаться с перерывами, следовательно, топология сети больше не определена статически. Как следствие, некоторые из основных принципов исследования распределенных систем не применимы к мобильным системам, и необходимо приспосабливать и расширять существующие теоретические и технологические результаты для этого нового сценария.

Мобильная беспроводная ad-hoc сеть (MANET) состоит из набора равноправных (peer) мобильных узлов, которые способны взаимодействовать друг с другом без помощи со стороны какой-либо фиксированной инфраструктуры. Взаимосвязи между узлами способны к изменению на непрерывном и произвольном основании. Узлы в пределах радио диапазона друг друга взаимодействуют непосредственно через беспроводные связи, в то время как те, которые находятся далеко друг от друга, используют другие узлы как ретрансляторы. Узлы обычно разделяют одни и те же физические носители; они передают и принимают на той же самой полосе частот, и подчиняются одной и той же топологии сети при распространении кода. Функция уровня передачи данных управляет беспроводными ресурсами связи и координирует доступ к носителю среди соседних узлов. Функция сетевого уровня поддерживает многозвенные маршруты коммуникации в сети; все узлы должны функционировать как маршрутизаторы, которые обнаруживают и поддерживают маршруты к другим узлам в сети. Мобильность и изменчивость скрыты от приложений так, чтобы любой узел мог взаимодействовать с любым другим узлом, как будто они все находятся в фиксированной обычной сети.

2.1 Безопасность в мобильных системах

Количество беспроводных пользователей в 2003 году превысило 35 миллионов человек, и специалисты предсказывают рост числа пользователей на 50-200% каждый год до 2006. Беспроводные сети обладают рядом неоспоримых преимуществ – они гибки, мобильны, удобны в использовании, обладают высокой производительностью при относительно малой цене. Те же аспекты, которые делают беспроводные сети привлекательными в использовании, могут стать и их ахиллесовой пятой в случае ненадлежащей защиты. Беспроводные сети обеспечивают пользователей огромным количеством свободы. К сожалению, эта технология имеет огромное количество недостатков безопасности.

Безопасность MANET – существенный компонент для основных сетевых функций, таких как отправление пакета и маршрутизация: сетевая операция легко подвергается опасности, если контрмеры заранее не были встроены в базовые сетевые функции на ранних стадиях проектирования. В отличие от сетей, использующих специализированные узлы для поддержки базовых функций, в ad-hoc сетях эти функции выполняются всеми доступными

узлами. Это существенное различие лежит на уровне ядра проблем защиты, которые являются специфическими в ad-hoc сетях. В противоположность специализированным узлам классической сети, узлам ad-hoc сети нельзя доверять, и невозможно гарантировать, что они будут правильно выполнять критические сетевые функции.

Если между узлами ad-hoc сети априорно существуют доверительные отношения, идентификация сущности может быть достаточной для гарантии правильного выполнения критических сетевых функций. Априорное доверие может существовать только в нескольких специальных сценариях как военные и корпоративные сети, где управляет сетью некая власть, которой доверяют, что для реализации критических функций требует аппаратных средств, защищенных от неумелого обращения. Идентификация сущности в большой сети, с другой стороны, поднимает ключевые требования управления. Среду, где существует общая власть, которой доверяют, называют управляемой средой.

Когда аппаратные средства, защищенные от неумелого обращения, и строгая опознавательная инфраструктура не доступны, например, в открытой среде, где общая власть, которая регулирует сеть, не существует, любой узел ad-hoc сети, может подвергнуть опасности надежность базовых функций типа маршрутизации. Корректная операция сети требует не только правильного выполнения критических сетевых функций каждым участвующим узлом, но она также требует, чтобы каждый узел выполнял четко определенную долю этих функций. Угрозы, которые рассматриваются в MANET, таким образом, не ограничиваются злонамеренностью, и новый тип аномального поведения, называемого эгоизмом (selfishness), нужно также принимать во внимание, чтобы предотвратить ситуации, при которых узлы просто не вступают во взаимодействие.

При отсутствии априорного доверия классические сетевые механизмы безопасности, основанные на аутентификации и управлении доступом, не могут справиться с эгоизмом, и совместные схемы безопасности кажутся единственным разумным решением. В совместной схеме безопасности, аномальное поведение узла может быть обнаружено через сотрудничество между множеством узлов, предполагающих, что большинство узлов не ведет себя аномально.

Существующие ad-hoc протоколы маршрутизации в основном подвергаются двум типам атак: активным атакам и пассивным атакам. Атака считается активной, когда аномально ведущий себя узел несет некоторые энергетические затраты при выполнении угрозы, в то время как пассивные нападения происходят главным образом из-за отсутствия сотрудничества с целью сохранения энергии. Узлы, которые выполняют активные атаки с целью повреждения других узлов, что ведет к выходу сети из строя, рассматриваются как злонамеренные, в то время как узлы, которые выполняют пассивные атаки с целью сохранения срока службы аккумулятора для своих собственных взаимодействий, рассматриваются как эгоистичные.

Злонамеренные узлы могут разрушить корректное функционирование протокола маршрутизации, изменяя маршрутизацию информации, фабрикуя ложную машрутизацию и играя роль других узлов. Кроме того, существуют также атаки, получившие название червоточины. С другой стороны, эгоистичные узлы могут существенно ухудшить сетевую производительность и, в конечном счете, разделить сеть, просто не участвуя в сетевых операциях.

Существует множество потенциальных угроз MANET. Отказ в обслуживании (DoS), захват сеанса и сниффинг (sniffing – прослушивание сетевого трафика). Утилиты для сниффинга появились с первых дней появления самих локальных сетей и предназначались для облегчения сетевого администрирования. Однако в соответствующих руках эти утилиты – снифферы – стали мощным инструментом хакеров, позволяющие перехватывать пароли и другую информацию, передаваемую по локальной сети. Традиционно снифферы считаются довольно сложными утилитами, требующими определенного умения для работы с ними,

зачастую еще и с непростыми руководствами. Все это изменилось в последние несколько лет, когда появились и стали широко применяться легкие в использовании специализированные снифферы паролей. Многие из этих утилит нового поколения свободно доступны в Интернете. Имея встроенную базу данных, позволяющую понимать многие сетевые протоколы, снифферы фильтруют сетевой трафик на лету, выделяя только требуемую информацию (такую как связку usernames-passwords).

Пока многие из нападений на беспроводные сети похожи на нападения против обычных сетей, сети MANET находятся в большой опасности. Одной из самых больших проблем является WEP (Wired Equivalent Privacy), стандарт шифрования данных для беспроводных сетей. В WEP обнаружено несколько уязвимостей, которые позволяют его взломать всего за пару часов. Другая проблема связана с открытой природой MANET. Из-за особенностей беспроводных сетей, в них сложно контролировать область доступности сигнала. В отличие от простых сетей, это введёт к тому, что хакер может управлять или подслушивать сеть из областей, которые не были предназначены для обслуживания, когда создавалась сеть. MANET также могут использоваться для создания backdoors к обычным сетям. Многие организации тратят тысячи и даже миллионы долларов для обеспечения защиты обычных сетей, но достаточно всего лишь одного недобросовестного или недоученного беспроводного пользователя, связанного с обычной сетью, и можно легко создать backdoor, обходящий все сложные и дорогостоящие системы защиты, тем самым, позволяя хакеру получить доступ к тщательно закрытой сети.

Итак, все свойства безопасности, свойственные обычным сетям, присущи и беспроводным сетям.

Установление подлинности (аутентификация). Установление подлинности играет важную роль в беспроводных сетях. Все пользователи MANETs обязаны подтверждать подлинность прежде, чем получат доступ к сети. Установление подлинности поможет ограничить доступ к закрытым ресурсам. Политика должна учитывать стандарт аутентификации, метод, реализацию и требования по обслуживанию. Политика для беспроводного установления подлинности должна соответствовать последним стандартам безопасности. Должна быть определена форма взаимной аутентификации. При взаимном установлении подлинности и клиент, и сервер авторизуют друг друга. Взаимное установление подлинности, прежде всего, повышает безопасность, устанавливая подлинность сервера, а также уменьшает возможности мошенничества в сети.

Конфиденциальность. Шифрование может обеспечить безопасный канал связи, в котором беспроводная передача данных может происходить без угрозы подслушивания. Передача данных без шифрования – это лёгкая добыча для хакера. Он легко может собрать чувствительную информацию, передающуюся в/из беспроводной сети. Беспроводной сигнал помогает хакерам незаметно собрать важные данные. Основной метод шифрования, используемый в MANET сетях – Wired Equivalent Privacy (WEP). К сожалению, как оказалось, WEP-алгоритм может быть легко расшифрован и, поэтому, не является надёжным. Однако WEP обеспечивает некоторую защиту и должен использоваться, если другое шифрование невозможно. Для борьбы с ненадежностью WEP, в настоящее время существуют другие методы шифрования. Группа стандартов IEEE 802.11i в настоящее время развивает стандарт для 802.11 безопасности.

Доступность. Доступность MANET существенна, т.к. производительность – функция времени простоя. Беспроводные испытания работоспособности должны проводиться перед развертыванием MANET сети, чтобы гарантировать охват сигналом нужной территории, которая является свободной от конфликтных RF (Radio Frequency – радиочастотных) сигналов. Сети MANET действительно подвержены некоторым RF конфликтам и препятствиям. Если испытания работоспособности не выполнены своевременно, то сеть может страдать от плохого сигнала очень долго.

Мобильные ad-hoc сети являются ключевым фактором в развитии беспроводных коммуникаций. Эти сети наследуют традиционные проблемы беспроводной и мобильной связи, такие как оптимизация пропускной способности, контроль мощности, улучшения качества передачи данных. Кроме того, их многозвенная природа и вероятное отсутствие стабильной структуры вводят новые аспекты исследований, такие как конфигурация сети, обнаружение устройства, поддержка топологии, а также ad-hoc-адресация и собственное управление маршрутизацией.

2.2 Свойства мобильности

Существует несколько видов объектов, которые могут быть мобильны, как например данные, код, агенты, операционные среды, поэтому при описании мобильности требуется не только уметь правильно описывать различные мобильные сущности, но также располагать средствами обоснования, почему и когда они должны передвигаться. В работе [Mas99a] вводится классификация мобильных сущностей, описывается ряд архитектурных стилей программного продукта, включающих мобильные компоненты, и даются конкретные примеры их использования:

1. **Стиль данных**: это самый простой вид мобильности. Мобильные сущности являются данными от поставщика для запрашивающей стороны. Типичный пример – клиент-серверная архитектура, основанная на протоколе как HTTP: HTTP-серверы посылают данные HTTP-клиентов в форме страниц HTML.
2. **Стиль ссылки**: в этом случае только ссылка к некоторой сущности перемещается от поставщика к запрашивающей стороне, а не данные или программа. Пример мобильной ссылки – адрес URL: запрашивающая сторона получает URL, и может переместить свой браузер на страницу с полученной ссылкой.
3. **Стиль кода**: в этом случае некоторый исполнимый код может передвигаться от одного сайта к другому. Java-апплеты основаны на мобильности кода.
4. **Стиль код&память**: в этом случае и некоторый код, и значения переменных, к которым он обращается, перемещаются. Этот вид мобильности обычно называют *мобильностью агента*, и эта форма мобильности в настоящее время предлагается в Java-системах мобильных агентов.
5. **Стиль код, память&состояние**: в этом случае код, память, и состояние (потоки) агента могут передвигаться. Прозрачность ссылок – основная цель этого вида мобильности, поскольку выполнение приостанавливается на одном сайте и продолжается на другом сайте. Этот стиль мобильности был реализован в Telescript.
6. **Стиль замыкания (closure)**: В этом случае агент (code + store) перемещает и поддерживает свои линии связи с ресурсами, которые были локальными к его исходному местоположению и стали удаленными в его месте назначения. Примером служит язык Obliq [Car95] который использует именно этот вид мобильности.
7. **Стиль окружения (ambient**): этот стиль описывает перемещение всего амбиента (окружения), участвующего в вычислениях. Амбиенты могут содержать другие амбиенты, которые также перемещаются. Таким образом, можно моделировать, например, перемещение ряда программ от рабочей станции до ноутбука. В настоящее время не существует спецификационных языков, позволяющих описывать этот вид мобильности. Однако, Карделли (Cardelli) и Гордон (Gordon) предложили язык программирования, основанный на этой парадигме [CG98].

В работе [FR02] предлагается использовать для изучения мобильности разновидность свойства безопасности BNDC (бисимуляционная NDC), в которой каждое достижимое состояние является BNDC-безопасным. Это свойство называется постоянным BNDC (PBNDC) и подходит для формального описания мобильности. Если процесс переходит из

одной среды выполнения в другую (например, меняется основная машина) во время выполнения, необходимо гарантировать, что такое промежуточное состояние все еще остается безопасным. Обеспечение свойства PBNDC с самого начала гарантирует, что каждая возможная миграция будет находиться в безопасном состоянии.

2.3 Методы спецификации мобильности

2.3.1 Спецификационные языки

Bauhaus [CGZ95] и **KLAIM** [DFP98] – спецификационные языки, основанный на модели кортеж-область (tuple-space), как и MobiS. Области в Bauhaus могут быть вложенными, как и в MobiS, а в KLAIM – нет. KLAIM включает алгебру процессов и использует систему типов для выполнения некоторых проверок безопасности. В отличие от MobiS, Bauhaus и KLAIM не поддерживают мобильность областей.

Obliq [Car95] – не типизированный, интерпретируемый язык лексической области действия (lexically-scoped), который поддерживает распределенные объектно-ориентированные вычисления. Объекты Obliq имеют состояние и являются локальными по отношению к узлу. Вычисления Obliq могут бродить по сети, поддерживая подключения к сети. Распределенная лексическая область действия – ключевой механизм для того, чтобы управлять распределенными вычислениями.

MobiS [Mas99a] – расширенная версия PoliS [Mas99b], язык спецификации, основанный на модели кортеж-область (tuple-space) [CFM98], которая специфицирует согласование (coordination) с помощью мультимножественного переписывания. Следует отметить, что MobiS допускает спецификацию вышеприведенных архитектурных стилей для мобильности. Спецификации MobiS иерархически структурированы: спецификация MobiS обозначает дерево вложенных областей, которое динамически развивается во времени. Области – это сущности первого класса, и могут передвигаться, а именно, они могут изменять свою позицию в дереве.

Типичная структура MobiS изображена на Рис. 6. Структура дерева представляет иерархическую структуру вложенных кортеж-областей. Окружение содержит два хоста (*Host*1 и *Host*2). *Host*1 содержит агента. Формально область MobiS может содержать три типа кортежей: обычные кортежи, которые являются упорядоченной последовательностью значений, кортежи программ, которые представляют код, и кортежи областей, которые представляют области.

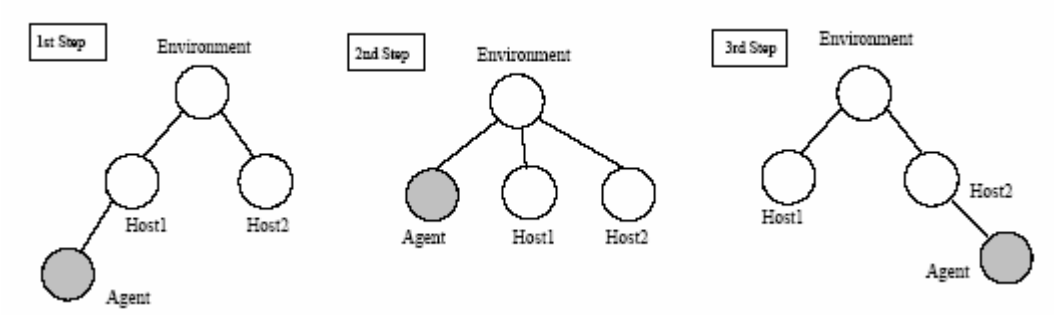

Рис. 6. Движения агентов.

Кортеж программы может изменить область, удаляя и добавляя кортежи (и, следовательно, области). Однако, кортеж программы может только управлять кортежами области, к которой он принадлежит, и кортежами ее родительской области. Это ограничение определяет и "input", и "output" среду любого кортежа. Действие кортежа программы выполняется атомарно. Кортеж программы ("r": R) обращается к правилу R, которое определяет, какая реакция может иметь место. Правило может действовать или на кортежи области, в которой он постоянно находится, или на кортежи родительской области: авторы

называют эти области областью действия правила. Правило читает и потребляет кортежи в своей области действия, выполняет последовательные вычисления, и производит новые кортежи в своей области действия. Формально, правило состоит из предактивации, локального вычисления и постактивации. Предактивация – мультимножество кортежей, которые должны быть найдены в его области действия; локальное вычисление – любые последовательные вычисления, которые не изменяют области кортежа; постактивация состоит из мультимножества кортежей, которые должны быть произведены в его области действия. Предактивация может включать формальные кортежи, поля которых могут быть идентификаторами; кроме того, она включает примитив ***ask***, который разрешает проверять значения, которые присвоены идентификаторам формального кортежа. Когда правило активизировано в области, реакция имеет место: кортежи, подлежащие локальной обработке, удаляются, кортежи, подлежащие внешней обработке, удаляются из родительской области, локальные вычисления выполняются, и, наконец, некоторые кортежи создаются. Кортеж в предактивации должен быть прочитан, если символ "? " помещен перед ним, иначе он должен быть обработан; чтение или обработка включают родительскую область, если символ "↑" помещен перед кортежем, и включают локальную область, если символ отсутствует; кортеж в постактивации должен быть создан в родительской области, если символ "↑" помещен перед ним, иначе – в локальной области.

Области представлены кортежами областей ("*name*" * *SP*), где *name* – название области, *SP* – спецификация содержимого области. Область – мультимножество кортежей. Области являются сущностями первого класса в MobiS. Их можно читать, обрабатывать или создавать точно так же, как обычные кортежи. Кортеж программы имеет форму ("*rule_id*": *rule*), где *rule_id* – идентификатор правила и *rule* – правило MobiS. В то время как кортеж области имеет форму ("*name*" * *SP*), где *name* – имя, которое идентифицирует область, и *SP* – конфигурация областей. Кортежи программ и областей имеют идентификатор, который упрощает чтение или обработку кортежей программ и областей ("*rule_id*" и "*name* "). Всякий раз, когда непересекающиеся мультимножества кортежей удовлетворяют предусловиям активации набора правил, такие правила могут быть выполнены независимо и одновременно: каждое правило изменяет только часть области, содержащую кортежи, которые должны быть прочитаны или обработаны, и поэтому другие правила могут изменить другие кортежи в области или в других областях. Кортежи, представляющие сообщения, помещаются в область, разделяемую компонентами, которые должны взаимодействовать. Следовательно, коммуникация расцеплена, потому что компоненты не знают друг друга, так как они обращаются к кортежам с помощью сопоставления с шаблоном. Так как сообщения не имеют никакого адреса назначения, их содержимое определяет набор возможных получателей, (коммуникация – управляемое свойство).

Мобильность. Поскольку области – сущности первого класса, они могут передвигаться. Фактически, целое поддерево (называемое деревом областей спецификации MobiS) может мигрировать от одного узла к другому. Мобильность в MobiS состоит из обработки и создания кортежей областей. Поскольку область действия правил – локальная и родительская области, перемещение выполняется "step by step", из некоторой области к ее родителю и так далее.

2.3.2 Алгебры процессов

π-исчисление [MPW92] (Milner, Parrow, Walker 1989) – алгебра процессов, которая ведет свое происхождение от CCS. Примитивы исчисления просты и выразительны: имена каналов могут создаваться, передаваться (таким образом предоставляя возможность динамического реконфигурирования ознакомлений процесса), и подвергаются сложным правилам области действия. π-исчисление – это архетип механизма передачи имен или номинальная алгебра процессов. Алгебра процессов с передачей имен подчеркнула принцип, что имена (генерация локальных имен, обмены имен, и т.д.) обеспечивают подходящую

абстракцию для формального объяснения широкого диапазона явлений в глобальной вычислительной системе. Алгебра процессов с передачей имен имеет большую выразительность, чем обычные алгебры процессов, но возможность динамической генерации новых имен приводит к более сложной теории. В частности стандартные модели конечных автоматов становятся автоматами с бесконечным состоянием и бесконечными ветвлениями, таким образом, делая верификацию трудной задачей.

Пусть N – бесконечное перечислимое множество имен (обозначенных через a, . . . , z), тогда множество агентов в π-исчислении над N определяется синтаксисом:

$$P ::= \text{nil} \mid \alpha.\, P \mid P_1 \parallel P_2 \mid P_1 + P_2 \mid (\text{x})\, P \mid [\text{x} = \text{y}]P \mid A(x_1, \ldots, x_{\text{r(A)}})$$
$$\alpha ::= \text{tau} \mid x!y \mid x?(y),$$

где $r(A)$ –это диапазон идентификатора агента A. Появление y в $x?(y).P$ и в $(y)P$ ограничено; свободные имена определяются как обычно, и fn(P) указывает множество свободных имен агента P. Для каждого идентификатора A существует определение $A(y_1, \ldots, y_{\text{r(A)}}) := P_A$ (где все y_i различны и $\text{fn}(P_\text{A}) \subseteq \{y_1 \ldots y_{\text{r}(A)}\}$), и предполагается, что каждый идентификатор в P_A находится в области действия префикса (защищенная рекурсия – guarded recursion).

Наблюдаемые воздействия, которые агент может совершать, определяются следующим синтаксисом:

$$\mu ::= \text{tau} \mid x!y \mid x!(z) \mid x?y;$$

где x и y являются свободными именами для μ (fn(μ)), а z есть пограничное имя (bn(μ)); в заключение

$$\text{n}(\mu) = \text{fn}(\mu) \cup \text{bn}(\mu).$$

Конструктивные правила для опережающей (early) операционной семантики определяются в таблице 1.

TAU $\mathtt{tau}.P \xrightarrow{\mathtt{tau}} P$	OUT $x!y.P \xrightarrow{x!y} P$	IN $x?(y).P \xrightarrow{x?z} P\{z/y\}$
SUM $\dfrac{P_1 \xrightarrow{\mu} P'}{P_1 + P_2 \xrightarrow{\mu} P'}$	PAR $\dfrac{P_1 \xrightarrow{\mu} P_1'}{P_1 \| P_2 \xrightarrow{\mu} P_1' \| P_2}$ if $\mathtt{bn}(\mu) \cap \mathtt{fn}(P_2) = \emptyset$	
COM $\dfrac{P_1 \xrightarrow{x!y} P_1' \quad P_2 \xrightarrow{x?y} P_2'}{P_1 \| P_2 \xrightarrow{\mathtt{tau}} P_1' \| P_2'}$	CLOSE $\dfrac{P_1 \xrightarrow{x!(y)} P_1' \quad P_2 \xrightarrow{x?y} P_2'}{P_1 \| P_2 \xrightarrow{\mathtt{tau}} (y)(P_1' \| P_2')}$ if $y \notin \mathtt{fn}(P_2)$	
RES $\dfrac{P \xrightarrow{\mu} P'}{(x)P \xrightarrow{\mu} (x)P'}$ if $x \notin \mathtt{n}(\mu)$	OPEN $\dfrac{P \xrightarrow{x!y} P'}{(y)P \xrightarrow{x!(z)} P'\{z/y\}}$ if $x \neq y, z \notin \mathtt{fn}((y)P')$	
MATCH $\dfrac{P \xrightarrow{\mu} P'}{[x = x]P \xrightarrow{\mu} P'}$	IDE $\dfrac{P_A\{y_1/x_1, \ldots, y_{r(A)}/x_{r(A)}\} \xrightarrow{\mu} P'}{A(y_1, \ldots, y_{r(A)}) \xrightarrow{\mu} P'}$	

Таблица 1. Опережающая операционная семантика для π-исчисления.

Для π-исчисления было введено несколько бисимуляционных эквивалентностей [SW02], они основаны на непосредственном сравнении наблюдаемых воздействий, совершаемых π-агентами. Они могут быть слабыми и сильными, опережающими, запаздывающими или открытыми. Существует несколько диалектов π-исчисления, в том числе и асинхронное [Pis99]. Для примера приводим определение опережающей бисимуляции.

Бинарное отношение $\mathcal{B}$ на множестве агентов есть строгая опережающая бисимуляция, если она симметрическая, и если $P\ \mathcal{B}\ Q$, тогда имеем:

если $P \xrightarrow{\mu} P'$ и $\text{fn}(P,Q) \cap \text{bn}(\mu) = \varnothing$, тогда существует Q′, такое что $Q \xrightarrow{\mu} Q'$ и $P'\ \mathcal{B}\ Q'$.

Говорят, что два агента строго опережающе бисимулянтны, записывается как P $\simeq$ Q, если существует бисимуляция, такая что $P \mathcal{B} Q$.

Распределенное Join-исчисление [FGLMR96] (Fournet and Gonthier 1996) является исчислением мобильных агентов, расширяет π-исчисление и вводит явную нотацию именованных локальностей и неисправностей в распределенных окружениях.

Seal-исчисление. Seal-исчисление [VC99] – распределенное исчисление процессов с местоположением и мобильностью вычислительных объектов, названных изоляциями (seal). Seal также является рамочной концепцией для написания безопасных распределенных приложений для крупномасштабных открытых сетей типа Интернет. Представлен синтаксис и семантика редукции для Seal-исчисления.

Nomadic Pict. В [SW00] вводится язык программирования агента – Nomadic Pict. Он разработан для того, чтобы выражать алгоритмы инфраструктуры как можно яснее при трансляции с языка высокого уровня в низкий уровень. Уровни основаны на строго-определенном исчислении процессов, которое обеспечивают отчетливые уровни абстракции.

Ambient calculus [CG98] (Cardelli and Gordon 1998). Ambient-исчисление – это формализм для описания мобильности как программного обеспечения, так и аппаратных средств. Амбиент – именованный кластер выполняющихся процессов и вложенных подамбиентов. Каждое состояние вычисления имеет пространственную структуру, дерево, порождаемое вложением амбиентов. Мобильность абстрактно представляется переупорядочиванием этого дерева: амбиент может передвигаться внутри или снаружи другого амбиента. Ambient-исчисление – это алгебра процессов, в которой амбиенты моделируют некий спектр концепций, таких как сетевые узлы, пакеты, каналы и программные агенты.

Ambient-исчисление основано на миграции областей (ambient). Всякий раз, когда возникает необходимость миграции, создается амбиент, и все, что должно переместиться, помещается внутрь его. Перемещается сам амбиент. Как говорят авторы, амбиент есть именованное, ограниченное место, где выполняются вычисления, и служит амбиент как для обеспечения мобильности, так и для соблюдения безопасности. Стандартные методики моделирования распределенных систем, такие как алгебра процесса CSP [Hoa80] и π-исчисление [MPW92], не предлагают специальных конструкций для того, чтобы выразить свойства мобильности, и, таким образом, локальность (формирование административных доменов) и перемещение объектов между ними должны моделироваться явно без поддержки исчисления, что является особенно неудобным при моделировании сложных систем. Чтобы преодолеть этот дефицит, Карделли (Cardelli) и Гордон (Gordon) ввели мобильное ambient-исчисление [CG98], расширяющее π-исчисление. Ambient-исчисление вводит специфические элементы, называемые мандатами (capabilities), которые позволяют описывать такие понятия, как мобильная миграция агентов и системы агентов. Безопасность амбиентов основана на управляемом распределении способностей, право войти в амбиент не подразумевает право выйти из него. Ambient-исчисление имеет следующие главные особенности: иерархическая локальность, миграция, локальная коммуникация, контекстно-зависимая эквивалентность, и семантика редукции.

Boxed Ambient calculus является расширением ambient-исчисления [BCC01], которое определяет более практические шаблоны коммуникации. В этом исчислении коммуникация допускается также с родителем и потомками текущего амбиента (ambient), а рассеивание (dissemination) амбиентов становится невозможным. Однако это исчисление оставляет за пределами своего применения проблемы, связанные с интеграцией состояния и вычислениями, выполняемыми внутри процессов.

Пусть n – это имена, P и Q – процессы и M – способности. BoxedAmbient-исчисление предлагает следующие элементы: ограничение $(\nu n)P$, пустой процесс 0, композиция $P \mid Q$,

репликация !P, амбиент $n[P]$, проявление способности $M.P$, ввод $(x)^a.P$ и вывод $\langle x \rangle^a$, где $a \in \{\star, \uparrow, n\}$ и $\star$ означает локальную коммуникацию, $\uparrow$ означает коммуникацию с родительским амбиентом, и n означает коммуникацию с подамбиентом с именем n. Мандатами M являются *in n* и *out n* для входа в амбиент n и выхода из него соответственно.

HOPLA [NW02] (Higher-Order Process LAnguage [NW02]) является языком для описания недетерминированных процессов высокого порядка. Он имеет непосредственную операционную семантику, поддерживающую обычную бисимуляционную конгруентность, и в нем можно непосредственно кодировать такие исчисления, как CCS, CCS высокого порядка и мобильные амбиенты с публичными именами. Язык возник из работ по теории линейных доменов для параллельности, основанной на категориальной модели линейной логики и связанных комонад (associated comonads) [NW02a]. Денотационная семантика, данная в [NW02], интерпретирует процессы как пред-пучки (presheaves).

В работе [NW03], которая называется полной семантикой HOPLA, рассматривается семантика маршрутов ("path semantics") для HOPLA, которая характеризует контекстуальную и логическую эквивалентность, причем последняя связывается с бисимуляцией. Эта семантика является ясным теоретическим описанием процессов как нисходящих замкнутых множеств маршрутов вычислений: операции HOPLA возникают как синтаксические кодировки канонических конструкций на таких множествах; полная абстракция есть прямое следствие выразительности по отношению к маршрутам вычисления. Простое доказательство семантической непротиворечивости и адекватности показывает соответствие между денотационной и операционной семантикой. Семантика маршрута похожа на семантику трасс в том, что процессы указывают нисходящие замкнутые множества маршрутов вычисления, и соответствующая нотация эквивалентности процессов, называемая эквивалентностью маршрутов, дается с помощью равенства таких множеств.

Будучи языком процессов высокого уровня, HOPLA позволяет описывать передачу процесса и, следовательно, выражать некоторые формы мобильности, в частности те, которые присутствуют в ambient-исчислении с публичными именами [CG00]. Другой вид мобильности, мобильность коммуникационных связей, возникает из генерации имен, как в π-исчислении [MPW92]. Свойства бисимуляции и семантическое обоснование находятся в процессе разработки.

MetaKlaim. Работа [FMP03] описывает дизайн и семантику MetaKlaim – распределенного исчисления процессов высокого порядка, оснащенного механизмами ступенчатости (staging). MetaKlaim интегрирует MetaML (расширение SML для многоступенчатого программирования) и Klaim [DFP98], чтобы разрешить интерливинг мета-программных деятельностей (как ассемблирование и связывание фрагментов кода), динамическую проверку политик безопасности в административных границах и "традиционные" вычислительные деятельности в глобальной сети (удаленные взаимодействия и мобильность). MetaKlaim применяет мощную систему типизации (включая полиморфические типы), чтобы иметь дело с высоко параметризованными мобильными компонентами и динамически предписывать политики безопасности: типы являются метаданными, которые извлекаются из кода во время выполнения и используются, чтобы выразить гарантии доверительности. Динамическая проверка типов обеспечивает гарантии доверительности сетевых приложений, которые взаимодействуют с потенциально ненадежными компонентами.

2.3.3 Модальные логики

Mobile UNITY. Mobile UNITY [MR99] является формализмом, позволяющим специфицировать и верифицировать мобильные приложения. Эта система включает нотацию программирования и логику доказательств, которая является некоторой специализацией темпоральной логики. Mobile UNITY основана на модели UNITY [CM88], предложенной

Шанди (Chandy) и Мисра (Misra). Mobile UNITY расширяет и нотацию, и логику для описания мобильности. В UNITY программа содержит секции **declare, always, initially,** и **assign**. Секция **declare** объявляет переменные, которые будут использоваться в программе, причем каждой из них присваивается имя и тип. Секция **always** содержит определения, которые могут быть использованы далее в программе или в доказательствах. Секция **initially** содержит набор предикатов состояний, которые должны быть истинными перед началом программы (т.е. это, по сути, предусловие). Секция **assign** содержит операторы присваивания. Оператор присваивания сопровождается предикатом состояния, называемым защитой (*guard*), который проверяется при каждом выполнении присваивания. В Mobile UNITY были добавлены секция **Components**, позволяющая создавать различные экземпляры программы (компоненты), и секция **Interactions**, дающая возможность описать взаимодействие компонентов. Кроме того, в Mobile UNITY вводятся дополнительные операторы: *reactive* и *inhibit*. В отличие от UNITY, все переменные в Mobile UNITY являются локальными для каждой компоненты.

В дополнение к ее программной нотации Mobile UNITY также обеспечивает логику доказательства, некую специализацию темпоральной логики. Как в UNITY, свойства безопасности указывают, что определенные переходы между состояниями невозможны, в то время как свойства развития (progress property) указывают, что определенные действия будут в конечном счете иметь место. Свойства безопасности включают **unless**, **invariant**, и **stable**:

- p **unless** q утверждает, что если программа достигает некоторого состояния, в котором предикат $(p \wedge \neg q)$ выполняется, тогда p будет продолжать выполняться по крайней мере так долго, пока q не выполняется, что, возможно, будет всегда.
- **stable** p определяется как p **unless** *false*, что говорит о том, что если p выполняется, это будет продолжаться всегда.
- **Inv** p означает ((**INIT** $\Rightarrow p$) $\wedge$ **stable** p), т.е. выполняется в течение всего прогона программы, причем **INIT** характеризует начальное состояние программы.

Основные свойства развития включают **ensures**, **leads-to**, **until**, и **detects**:

- p **ensures** q устанавливает, что если программа достигает состояния, в котором предикат p истинен, p остается истинным до тех пор, пока q не станет ложным, и существует одно предложение, которое при условии его выборки будет гарантировать истинность предиката q. Это используется для определения основных свойств развития программы.
- p **leads-to** q устанавливает, что если программа достигает состояния, в котором предикат p истинен, она к конце концов достигнет состояния, в котором q истинен. Нужно отметить, что для p не требуется, чтобы он оставался истинным до тех пор, пока q не станет истинным.
- p **until** q определяется как ((p **leads-to** q) $\wedge$ (p **unless** q)) используется для описания условия развития, которое требует, чтобы p удерживалось истинным до тех пор, пока q не станет истинным.
- p **detects** q определяется как (($p \Rightarrow q$) $\wedge$ (q **leads-to** p)).

Автор (McCann) называет Mobile UNITY спецификационным языком.

Ambient-логика [CG00] – модальная логика, разработанная авторами ambient-исчисления для описания свойств распределенных и мобильных вычислений, выполняемых в ambient-исчислении. Так же как и стандартные темпоральные модальности для описания эволюции амбиент-процессов, данная логика включает новые пространственные модальности для описания древовидной структуры окружающих амбиент-процессов.

π-логика – темпоральная логика для агентов π-исчисления. Несколько логик было предложено для выражения свойств агентов π-исчисления [Dam93], [MPW92]. В работе

[FGMP03] представлена логика для спецификации поведенческих свойств агентов π-исчисления. Эта логика, названная π-логикой, расширяет модальную логику, введенную в [MPW92], следующими модальностями. Помимо строгой модальности **next**, обозначаемой $EX\{\mu\}$, которая была введена в [MPW92], π-логика также включает слабую модальность **next**, обозначаемую $<\mu>$, чье значение состоит в том, что перед μ может быть выполнен ряд ненаблюдаемых воздействий. Кроме того, для описания свойств живучести и надежности вводится темпоральный оператор *eventually* (нотация $EF\ \phi$). Значение $EF\ \phi$ состоит в том, что ϕ должен стать истинным когда-нибудь в возможном будущем. Синтаксис:

$$\phi ::= \text{true} \mid \ \mid \phi\ \&\ \phi' \mid EX\{\mu\}\phi \mid <\mu>\phi \mid EF\ \phi.$$

Интерпретация логических формул:

– $P \mid= true$ выполняется всегда;
– $P \mid= \sim\phi$, если и только если не выполняется $P \mid= \phi$;
– $P \mid= \phi\ \&\ \phi'$, если и только если $P \mid= \phi$ и $P \mid= \phi'$;
– $P \mid= EX\{\mu\}\phi$, если и только если существует P', такое что $P \xrightarrow{\mu} P'$ и $P' \mid= \phi$;
– $P \mid= <\mu>\phi$, если и только если существует $P_0, \ldots, P_n$, $n \geq 1$, такое что
$P = P_0 \xrightarrow{\mathtt{tau}} P_1 \ldots \xrightarrow{\mathtt{tau}} P_{n-1} \xrightarrow{\mu} P_n$ и $P_n \mid= \phi$;
– P $\mid= EF\ \phi$, если и только если существует $P_0, \ldots, P_n$ и $\mu_1, \ldots, \mu_n$, где $n \geq 0$, такое что
$P = P_0 \xrightarrow{\mu 1} P_1 \ldots \xrightarrow{\mu n} P_n$ и $P_n \mid= \phi$.

Могут быть определены следующие производные операторы:

– $\phi \mid \phi'$ означает $\sim(\sim\phi\ \&\ \sim\phi')$;
– $AX\{\mu\}\phi$ означает $\sim EX\{\mu\}\sim\phi$, обратный оператор к строгому оператору **next**;
– $[\mu]\phi$ означает $\sim<\mu>\sim\phi$, обратный оператор к слабому оператору **next**;
– $AG\ \phi$ означает $\sim EF\ \sim\phi$, это оператор **always**, который означает, что ϕ истинно и сейчас, и всегда в будущем.

Как было доказано в [GR00] π-логика адекватна по отношению к строгой опережающей бисимуляционной эквивалентности. Это значит, что два агента π-исчисления опережающе бисимулянтны, если они удовлетворяют одним и тем же свойствам, которые могут быть выражены в π-логике. Итак, теперь становится возможным использовать π-логику в алгоритме проверки модели для того, чтобы определить, удовлетворяют ли свойства, выраженные в π-логике спецификациям π-исчисления.

Работа [DL00] представляет темпоральную логику для специфицирования свойств программ на языке Klaim [DFP98]. Логика основана на логике HML (Hennessy-Milner Logic) и на μ-исчислении. Новыми особенностями данной логики является возможность оперировать свойствами состояний для описания эффекта воздействия на различные сайты. Логика снабжена согласованной и полной системой доказательств, которая дает возможность доказывать свойства мобильных систем.

В работе [Zap02] предлагается подход к определению пространственно-темпоральной логики для спецификации и анализа поведения мобильных систем. Эта логика основана на логике TLA (Temporal Logic of Actions) [Lam94] и расширяет ее операторами для описания свойств деревьев и изменений в их структуре.

Логика MTLA [MZW03] также является расширением темпоральной логики действий (Temporal Logic of Actions) [Lam94]. Расширение предназначено для спецификации мобильных систем. В то время как в ambient-исчислении каждый узел дерева конфигурации связывается с процессом, MTLA связывает с каждым узлом локальное состояние. MTLA содержит как темпоральные, так и пространственные модальности. Ее формулы вычисляются во время выполнения в конкретном местоположении.

2.3.4 Конечные автоматы

Амбиентные интерактивные конечные автоматы (Ambient Interacting State Machines – AmbISMs). В работе [KO03] представлены две конкретизации обобщенных интерактивных конечных автоматов (ISMs) с мобильными особенностями, которые являются полезными для моделирования и верификации динамически изменяющихся мобильных систем. ISMs – автоматы с локальным состоянием, обменивающиеся сообщения одновременно на многочисленных буферизованных портах. Система обобщенных ISMs также имеет дело с глобальным состоянием, используемым, например, для описания топологии их взаимодействия. Вводится понятие амбиентных (Ambient) ISMs (AmbISMs), особенности которых включают иерархическую структуру среды окружения, миграцию, и ограничения локальности на коммуникацию. Дается альтернативная операционная семантика по отношению к операционной семантике (ограниченного – boxed) ambient-исчисления. Кроме того, AmbISMs комбинируются с динамическими ISMs, которые вводят динамические структуры коммуникации, а также активацию и дезактивацию ISM. Все варианты ISM определяются формально в пределах инструмента автоматического доказательства теорем Isabelle/HOL. На примере из области систем мобильных агентов демонстрируются преимущества данного подхода по сравнению с ограниченным ambient-исчислением.

В статье [OL03] вводится понятие обобщенной ISM, приводится иерархия конкретизацией ISMs, представленная на Рис. 7, и дается описание динамической ISM.

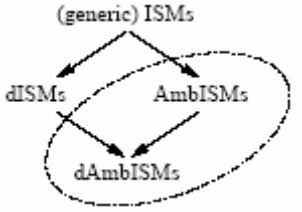

Рис.7. Иерархия ISMs [KO03]

Система обобщенных (generic) ISMs располагает глобальным состоянием и командами для изменения этого состояния. Динамический ISM (dISM) является конкретизацией обобщенного ISM, который обладает динамическим созданием, передачей, открыванием и закрыванием портов, а также динамическим созданием и уничтожением ISM. Система dISMs использует глобальное состояние, чтобы держать трек выполняющихся dISMs, доступных портов и принадлежности портов. Изменения этого состояния производятся участниками системы, выдающими соответствующие команды: dISM может запросить, чтобы dISM, еще не выполняющийся был активизирован, или чтобы выполняющийся dISM (включая себя) был остановлен. Кроме того, dISM может создать новый порт и стать его начальным владельцем. Владелец порта может принимать ввод на этот порт, позволить или запретить другим выводить на этот порт, или передать этот порт любому другому dISM. Возможность открывать или закрывать порты может использоваться для моделирования, например, управления потоком.

Автоматы AmbISMs дают новую форму операционной семантики для ambient-исчисления [CG98]. Вводится концепция состояния процесса с помощью комбинации ambient-процессов с ISMs. Амбиенты (ambients – можно понимать как оболочки) являются вложенными административными доменами, которые содержат процессы, являющимися ISMs. Амбиентная структура определяет возможность процессов общаться друг с другом. В оригинальном ambient-исчислении, только процессы в пределах того же самого амбиента могут обмениваться сообщениями. Это довольно строгое понятие локальной коммуникации расширяется по причинам, приведенным в [BCC01], до родительских амбиентов и амбиентов-потомков. Амбиенты являются мобильными в том смысле, что ISM может

переместить амбиент, которому он принадлежит, вместе со всеми ISMs и подамбиентами, содержавшимися в этом амбиенте, за пределы родительского амбиента или в амбиент брата. Кроме того, амбиент может быть удален ("вскрыт"), так что его содержимое вливается в окружающий амбиент, или новый амбиент может быть создан как потомок данного, где для симметрии дается возможность специфицировать как подмножества ISMs, находящиеся в данное время на том же самом уровне, так и другие амбиенты-потомки, которые должны сразу же переместиться в новый амбиент. Наконец, (новые) ISMs могут быть присвоены амбиентам. В литературе об амбиентах операции амбиентов называются мандатами (capabilities), т. к. обладание ими можно рассматривать, как квалификацию совершать соответствующие действия. Семантически квалификация просто сводится к знанию имени соответствующего амбиента.

Формально глобальное состояние амбиентов и их команды описываются следующим образом. Пусть $\aleph$ – это тип имен амбиентов. Иерархическая структура амбиентов дается частичной функцией типа $\aleph \rightsquigarrow \aleph$, отображающей каждое имя амбиента n в имена его родителя m (если они есть) или в специальное значение $\perp$, указывающее, что нет никакого родителя, т. е. амбиент n – в корне дерева. Это отношение можно представить в виде леса из деревьев амбиентов. Кроме того, назначение ISM к его базовому (home) амбиенту, описываемое частичной функцией типа $\Im \rightsquigarrow \aleph$, где $\Im$ – тип идентификатора ISM. Состояние амбиента *aSTATE*, уточняющее обобщенное глобальное состояние, есть Декартово произведение двух частичный функций, т. е. оно имеет форму α = (*parent*(α), *home*(α)). Пусть *aCMD* – это набор команд, воздействующих на глобальное состояние, тогда *aCMD* = {*Assign* (*j*, *n*) | $j \in \Im \wedge n \in \aleph$} $\cup$ {*In*(*n*) | $n \in \aleph$} $\cup$ {*Out*(*n*) | $n \in \aleph$} $\cup$ {*Del* (*n*) | $n \in \aleph$} $\cup$ {*Ins* (*n*, *ns*, *is*) | $n \in \aleph \wedge ns \in \wp(\aleph) \wedge is \in \wp(\Im)$}. Отношение глобального перехода *AmbTrans*(*i*) определяется как {(α, *acmds*, α' } | $i \in$ *dom*(*home*(α)) $\wedge$ $\alpha \xrightarrow{i:acmds}{}^{*} \alpha'$ }, где отношение выполнения одношаговой команды $\alpha \xrightarrow{i:acmds}{}^{*} \alpha'$ означает, что команда *acmd*, выданная *i*, переводит состояние α в α'.

HD-автоматы (History Dependent Automata) были введены в [MP95] с именем π-автоматы, как удобная структура для компактного описания операционного поведения агентов π-исчисления. Далее HD-автоматы были обобщены для алгебры процессов с передачей имен, алгебры процессов с локализацией (location), с причинностью (causality) и для сетей Петри [Pis99], [MP00]. Из-за механизма входных воздействий обычная операционная семантика π-исчисления требует бесконечного числа состояний даже для простого агента. При создании нового имени количество переходов возрастает до бесконечного множества: по одному на каждый выбор нового имени. Для решения этой проблемы в HD-автоматах имена появляются явно в состояниях, переходах и метках. Кроме того, удобно предположить, что имена, появляющиеся в состоянии, переходе или метке являются локальными именами и не требуют глобальной идентификации. При таком подходе для представления всех состояний системы, которые различаются только биективным (bijective – взаимно однозначный) переименованием, можно использовать единственное состояние HD-автомата. Однако, для каждого перехода требуется явное представление соответствия между именами исходного состояния, целевого состояния и метки.

Определение HD-автомата выглядит следующим образом. Автомат, зависящий от истории (history-dependent), HD-автомат, есть структура $A = (Q, q^0, L, \omega, q \xrightarrow{\lambda}{}_{\sigma} q')$, где

– Q – конечное множество состояний;
– q^0 – начальное состояние;
– L – множество меток воздействий;
– ω – функция, связывающая имена (конечное множество локальных имен) с состояниями:

$\omega : Q \to P_f(N)$;

– $q \xrightarrow[\sigma]{\lambda} q'$ – отношение перехода, где $\sigma : \omega(q') \to \omega(q) \cup \{*\}$ – (инъективная) встроенная функция, а символ * используется для управления созданием нового имени: имя, создаваемое во время перехода, связывается с *. Функция σ встраивает имена целевого состояния в имена исходного состояния перехода. Имена, которые появляются в исходном состоянии, но не появляются в целевом состоянии для некоторого перехода, отбрасываются в ходе развития. В HD-автоматах создание имен управляется явно, с использованием *, тогда как отбрасывание имени может происходить неявно. Как указывается в [MP95], использование локальных имен позволяет моделировать выполнение входных префиксных воздействий с помощью конечного числа переходов: этого достаточно для того, чтобы рассматривать в качестве входных значений все имена, которые появляются как свободные в исходном состоянии плюс только одно новое имя. Другими словами для HD-автомата не имеет смысла добавлять переходы, которые отличаются только выбором нового имени.

HD-автоматы можно понимать абстрактно как автоматы над моделью перестановки (permutation), компоненты которой – множества имен и их перестановок (подстановки переименования) на этих именных множествах.

2.3.5 Сети Петри

Сети Петри и их расширения широко используются для формализации параллельно выполняющихся процессов.

В [AB96] описываются **мобильные сети Петри** и **динамические сети Петри.** Мобильные сети Петри оперируют мобильностью наподобие π-исчисления, т.е. позволяют описывать изменяющуюся конфигурацию каналов коммуникации между процессами. Мобильные сети Петри являются вариантом сетей позиция/переход (place/transition) с раскрашенными маркерами, причем маркерами могут быть имена позиций внутри самих позиций. В π-исчислении имена каналов передаются по каналам.

Динамические сети Петри – мобильные сети Петри, имеющие возможность создавать новую подсеть, когда переход активизируется. В таких сетях постусловие перехода может быть целой сетью, а не только множеством позиций с постусловием. Активизация динамических сетей Петри зависит от функции подстановки для переменных. Активизация переходов удаляет объединенные маркеры, созданные на основе функции подстановки и предусловий переходов, и добавляет к позициям переходы и пред-/постусловия, возникающие в подсетях, которые появляются в результате подстановок переменных и обработки постусловий переходов.

2.3.6 Графическое моделирование

Мобильный UML [BKKW02], расширяет язык UML [UMLS03] концепциями для моделирования мобильных вычислений. Расширение описано в терминах UML с использованием стереотипов и помеченных значений (tagged) как мета-моделирующих средств. Особенно важно, что экземпляры классов стереотипа <location>, обозначают местоположения, где другие объекты могут располагаться. Мобильные объекты являются экземплярами классов со стереотипом <mobile>, и они могут изменять свое местоположение. Действительное движение мобильного объекта выполняется с помощью действия *move*, параметром которого является целевое местоположение..

В работе [LMBW03] предлагается расширение поведенческого подмножества UML-ских диаграмм состояний (statecharts) для мобильных вычислений. Изучаются коллекции объектов UML, поведение которых описывается с помощью диаграмм состояний. Каждый объект постоянно находится в данном месте, а коллекция таких мест формирует сеть. Объекты осведомлены о местоположениях других объектов, т.е. они знают логические, но не

физические, имена мест, где находятся другие объекты. В дополнение к их обычным возможностям, типа посылки сообщений и т.п., объекты могут передвигаться и создавать и разрушать места, которые могут приводить к реконфигурации сети. Для такого мобильного расширения представлена формальная семантика. Это расширение основано на семантике определения диаграмм состояний без мобильности.

2.3.7 Другие подходы

Работа [ABB02] описывает проект AGILE и некоторые результаты, полученные в течение первого года ведения проекта. Подходы на основе архитектуры были продвинуты как средство управления сложными системными конструкциями и их развитием, в особенности для того, чтобы предоставить системам маневренность (agility), необходимую для оперирования в турбулентных средах, и дать им возможность быстро приспосабливаться к изменениям в мире бизнеса. Последние технологические достижения в коммуникации и распределении сделали мобильность дополнительным фактором сложности, для которого текущие архитектурные концепции и методики едва ли могут быть приспособлены. Проект AGILE разрабатывает архитектурный подход, в котором аспекты мобильности могут моделироваться явно и отображаться на топологию распределения и коммуникации, причем эта топология доступна на физических уровнях. В целом подход разрабатывается в единообразной математической рамочной концепции, основанной на граф-ориентированных методиках, которые поддерживают семантически непротиворечивые методологические принципы, формальный анализ и уточнение.

2.4 Методы верификации мобильности

Глобальная вычислительная система определяется как сеть стационарных и мобильных компонентов. Основополагающие особенности глобальной вычислительной системы состоят в том, что ее компоненты автономны, программные средства разнообразны и динамичны, границы и масштаб сети изменчивы, некоторые компоненты постоянно находятся в узлах сети (Web-services), количество участников меняется динамически, зачастую без участия централизованных служб. Глобальные вычислительные системы должны быть очень устойчивыми, так как они предназначены для оперирования в потенциально враждебной динамической среде. Это означает, что их трудно конструировать корректно, и еще больше проблем возникает при попытках осуществлять контроль во время тестировании таких систем. В этой связи возрастает роль формальных методик анализа таких систем. Соответствующие технологии верификации должны давать уверенность в правильном поведении и устранении ошибок и рисков в обеспечении безопасности систем еще до того, как система поступит в эксплуатацию. Несмотря на то, что существенные успехи уже достигнуты в обеспечении основополагающих моделей и эффективных методик верификации для формальной верификации глобальной вычислительные системы, текущие технологии разработки программного обеспечения обеспечивают ограниченные решения некоторых из проблем, упомянутых выше. Проблема формальной верификации глобальной вычислительные системы все еще требует значительных усилий по ее исследованию и распространению.

В настоящее время проводятся исследования, направленные на поиск подходов к верификации свойств безопасности систем, специфицированных в исчислении мобильных амбиентов. В работе [BCFLP03] предложен новый алгоритм для анализа возможного гнездования в исчислении мобильных амбиентов на основе изучения анализа потока управления (CFA – Control Flow Analysis) такого исчисления [BCF02]. Этот алгоритм улучшает и временные, и пространственные характеристики ранее предложенных методик. Улучшения достигаются благодаря более совершенному представлению структур данных и сведения вычислений к ограничениям CFA.

2.4.1 Проверка модели

Как было описано выше, методика проверки модели применялась к протоколам защиты для формальной верификации их корректности. К сожалению, применение этой методики к глобальной вычислительной системе сталкивается со значительными сложностями, т.к. даже самые простые системы требуют исследования бесконечного числа состояний. В основном, исследования в этой области направлены на поиск приемлемых ограничений на поведение системы, которые позволили бы применять к мобильным системам уже прочно зарекомендовавшие себя методики и инструменты верификации.

Проверка модели ambient-исчисления с помощью ambient-логики. Задача проверки модели должна решить, удовлетворяет ли данный объект данной формуле. В работе [CG00] Карделли (Cardelli) и Гордон (Gordon) показали разрешимость проблемы проверки модели для фрагмента ambient-исчисления с конечным числом состояний по отношению к фрагменту ambient-логики без параллельного дополнения. Это ambient-исчисление с конечным числом состояний опускает конструкции полного исчисления для неограниченного дублирования и динамической генерации имен. Авторы не дают никакого анализа сложности их алгоритма. Оценки работы этого алгоритма по времени и пространству являются экспоненциальными.

В работе [CZGMT01] устанавливаются границы сложности проблемы проверки модели для ambient-исчисления без дублирования с публичными именами с помощью ambient-логики без параллельного дополнения. Показывается, что проблема PSPACE-полна, т.е. она может быть сведена к проблеме к другой проблеме, которая разрешима за полиномиальное время. Для верхней границы сложности предлагается новое представление процессов, которое остается в пределах полиномиального размера во время выполнения процесса; что позволяет выполнять процедуру проверки модели в полиномиальном пространстве. Кроме того, доказывается PSPACE-жесткость проблемы для нескольких весьма простых фрагментов исчисления и логики. Делается вывод, что не существует никаких содержательных фрагментов с полиномиальным временем для алгоритмов проверки модели.

Работа [VM94] посвящена интегрированной среде Mobility Workbench (MWB), в которой верификация бисимуляционной эквивалентности межу агентами π-исчисления осуществляется оперативно (on the fly) [FM91], т.е. пространство состояний агентов строится во время конструирования бисимуляционного отношения. Проверка бисимулярности является, в худшем случае, экспоненциальной. Функциональность проверки модели, предлагаемая в MWB основана на реализации табличной системы доказательств [Dam93], [Dam01] для пропозиционального μ-исчисления с передачей имен (расширения μ-исчисления, в котором можно выражать параметризацию и квантификацию имен).

HD-автоматы и π-исчисление. Работа [FGMP03] фокусируется на применении HD-автоматов в качестве теоретической основы для подхода, основанного на конечных автоматах, к верификации систем, специфицированных в π-исчислении. Интегрированная среда для верификации называется HAL (HD Automata Laboratory). HD-автоматы обеспечивают промежуточный, синтаксически независимый формат для представления исчисления, позволяющего описывать мобильность и примитивы распределения. Важным аспектом является тот факт, что для широкого класса процессов результирующие автоматы HD-автоматы являются конечными автоматами. Приводится процедура трансформации HD-автомата в обычный конечный автомат, вследствие чего появляется возможность адаптировать эффективные верификационные методики, разработанные для конечных автоматов, к верификации мобильных процессов. Описывается процедура автоматической верификации на основе проверки модели, в которой проверяется, удовлетворяются ли формулы π-логики для спецификаций, написанный в π-исчислении. Процесс сводится к трансляции формул π-логики в формулы логики, для которой уже существуют хорошо развитые методики верификации, основанные на проверке модели, представленной в виде

обычных конечных автоматов, относительно свойств, выраженных в этой логике. Приводится функция трансляции формул π-логики в формулы логики ACTL [DV90], для которой реализован эффективный анализатор моделей [Fer94] и для которой существует семантически непротиворечивая (sound) трансляция в обычные конечные автоматы. В итоге осуществляется проверка модели в виде обычного конечного автомата относительно свойств, выраженных в логике ACTL.

2.5 Тестирование мобильности

Тестирование и оценивание алгоритмов MANET в реальных условиях необходимы для успеха их применения. Однако, тестирование систем MANET больших размеров – дорогостоящее занятие из-за сложности необходимых аппаратных средств и невозможности протестировать широкий диапазон сценариев мобильности. Имитация (simulation) и эмуляция являются основными способами тестирования мобильных ad-hoc сетей (MANETs) и играют значительную роль в сетевых исследованиях. Несмотря на большое количество усилий, сделанных для развития исследований MANETs, существует только несколько систем, которые был доведены до реализации, и еще меньше систем, масштаб которых превышает дюжину узлов. Одной из причин такого положения является сложность реализации и тестирования реальных мобильных сетей. Не только из-за того, что реализация потребует изощренного программирования на системном уровне, но также из-за того, что тщательное тестирование требует развертывания крупномасштабного тестового стенда для различных схем мобильности. Без комплексных инструментальных средств и поддержки системы отладки тестирование MANET сведется к упрощенной задаче и вряд ли даст значимые результаты [MBJ99].

Имитация. Имитация дает возможность оценивать сетевые протоколы при изменяющихся сетевых условиях. Высокая начальная стоимость препятствует приобретению средств, необходимых для создания всесторонней продвинутой сетевой среды имитации. Это ограничение часто приводит к тому, что разные стадии имитации осуществляется различными группами, что в свою очередь ведет к недостаточной стандартизации и воспроизводимости. Появление интегрированных сред для имитации мультипротокольной сети обеспечивает богатые возможности для эффективного экспериментирования.

Имитация сети имеет свою историю. Первыми разработками в этой области можно считать REAL (Realistic and Large) [Kes88] и NEST (Network Simulation Testbed) [Dup90]. Имитаторы различаются по сфере их применения. Некоторые из них предназначены для исследований определенных типов сетей, другие созданы для широкого диапазона протоколов. Наиболее общие имитаторы поддерживают имитационные языки и библиотеки сетевых протоколов. Существуют имитаторы, которые используют процессор дискретных событий в качестве ядра, и/или адаптируют подходы для улучшения производительности и масштабируемости. Некоторые имитаторы дополняют обработку событий аналитическими моделями транспортного потока или поведением очередей. Параллельные и распределенные имитации обеспечивают еще большую производительность. Иногда имитаторы поддерживают мультипроцессорность или сети рабочих станций. Абстракция является основным понятием в имитаторах, т.к. все они поддерживают тот или иной уровень абстракции при выборе того, что необходимо имитировать. Имитаторы обеспечивают разнообразные интерфейсы от возможности программирования на языке подготовки сценариев до возможности использовать графический интерфейс, иногда допуская выполнение кода на реальной сети.

Самые значительные исследования ad-hoc сетей проводятся в CMU (университете Карнеги-Меллона) [MBJ01]. Был создан тестовый стенд (testbed), состоявший из 5 мобильных узлов, реализованных в виде движущихся автомобилей со скоростью приблизительно в 25-40 км/ч с двумя постоянными узлами, разделенными расстоянием приблизительно в 700m (2-3 радио-звена). Позже, там же был создан больший тестовый

стенд, состоявший из 8 движущихся вокруг сайта узлов в пределах 700m. Исследователи в BBN (Bible Broadcasting Network) также реализовали в реальном эксперименте ad-hoc сеть с 10 узлами [RH00]. Один очевидный недостаток такой реальной стратегии тестового стенда – то, что тестовые модели не являются ни масштабируемыми, ни воспроизводимыми.

Эмуляция. Эмуляция явилась следующим шагом на пути к повторимому и масштабируемому тестированию. Исследователи в университете Карнеги-Меллона использовали режим эмуляции имитатора ns2 для изучения беспроводных сетевых приложений [Fal99]. В этом эксперименте каждый пакет от реальной машины посылался централизованной машине, на которой выполнялась имитация ns2 тестируемой сети, и пакет откладывался или уничтожался согласно поведению, полностью определенному в имитации. Если пакет не уничтожался в пределах имитации, тогда он пересылался в реальную сеть в соответствующее время к месту его реального назначения. Эмуляция в ns2 хорошо подходит для тестирования приложений конечного хоста в ad-hoc среде, но не для тестирования ad-hoc сети или протоколов машрутизации.

Инструмент **Macfilter** университета Карнеги-Меллона [MBJ99] – инструмент для эмуляции, основанной на трассах. Подобная идея также используется в другом инструменте для тестирования ad-hoc сети с названием APE (Ad-hoc Protocol Evaluation) [LLNTN02]. В дополнение к функции подавления пакетов, APE также обеспечивает сбор данных и поддерживает анализ тестирования и оценивания протоколов маршрутизации для ad-hoc сетей. Инструмент визуализации сценария университета Карнеги-Меллона (ad-hockey [MBJ99]) создает и запускает сценарий без прямого взаимодействия с имитацией или эмуляцией. Инструменты эмуляции, основанной на трассах, такие как Macfilter и инструмент, разработанный в Беркли [NSNK97], не поддерживают мультизвенность ad-hoc сети.

В работе [KR01] описана стратегия “testbed on a desktop" для обеспечения эмуляции реальных беспроводных сетей в удобной маленькой беспроводной тестовой среде. “Testbed on a desktop" является независимым от операционной системы реализационной платформ и работает с самыми современными беспроводными сетевыми интерфейсами.

В работе [Zap02] описывается интегрированная среда для разработки и тестирования ad-hoc сетей, которая включает эмулятор мобильности MobiEmu. Эмулятор MobiEmu позволяет тестировать ad-hoc сети виртуально любого масштаба и с любым сценарием мобильности без физического перемещения ad-hoc узлов. Он эмулирует физическое движение узла и изменения топологии сети с реальной реализацией всего, что выше уровня канала передачи данных. MobiEmu является программной платформой для тестирования и анализа реальных ad-hoc сетевых протоколов и приложений (предмет тестирования). При этом используется фиксированная сеть из *n* linux машин, которая эмулирует мобильную ad-hoc сеть из *n* узлов. В реальной ad-hoc сети топология обеспечения связи между узлами является динамической, так как узлы то входят, то выходят из диапазона коммуникации друг друга. MobiEmu подражает этой реальной ситуации на сети тестового стенда, динамически устанавливая или удаляя фильтры пакетов. Цель состоит в том, чтобы создать ту же самую сетевую динамику для предмета тестирования, так, чтобы тестирование и анализ ad-hoc сетей могли быть проведены в лабораторной установке. Программное обеспечение MobiEmu выполняется на каждой машине тестового стенда. Эмуляция управляется сценарием, где входным воздействием является хронология местоположений и движений каждого узла. MobiEmu также имеет компонент пользовательского графического интерфейса, с помощью которого пользователь может просматривать, управлять, и визуализировать ad-hoc сеть во время выполнения. Чтобы тестировать реализацию ad-hoc сети, пользователь может запустить “live" сетевое программное обеспечение (например, предмет тестирования) на машинах тестового стенда. Альтернативно, пользователь может реализовать ad-hoc сеть (предмет тестирования) на совокупности мобильных компьютеров, затем загрузить программное обеспечение MobiEmu на каждый компьютер, чтобы установить удобную среду

тестирования. В любом случае, программное обеспечение MobiEmu гарантирует, что предмет тестирования не ощутит различия, и будет работать так, как будто он находится в реальной ad-hoc сети. Система MobiEmu оперирует в архитектуре главный/подчиненный (master/slave). Главный контроллер выполняется на специализированном хосте вне сети тестового стенда. Подчиненный контроллер выполняется на каждом хосте тестового стенда. Главный контроллер управляет всеми подчиненными и синхронизирует их действия: главный диктует, когда топология обеспечения связи должна измениться, а подчиненные выполняют такие изменения. Коммуникация главный/подчиненный находится на канале управления, который должен быть отделен от сети тестового стенда (например, вторая сеть Ethernet), чтобы избежать перемешивания с операциями ad-hoc сети. Файл сценария, который загружается через GUI, содержит список местоположений с временными метками и описание перемещений для всех узлов.

Исследователи HRL Laboratories разработали эмулятор ad-hoc сети следующего поколения, WiNE (Wireless Network Emulator) [Holland]. Он моделирует как физическую среду (включая модель распространения, постепенное изменение и потерю маршрута, помехи и интерференцию, коэффициент усиления антенны и чувствительность получателя, и т.д.), так и уровень канала передачи данных. Новая архитектура состоит из двух основных аппаратных компонентов: конечный кластер высокоскоростных процессоров для обработки физического мира, и входной кластер из персональных компьютеров, которые похожи на узлы тестового стенда MobiEmu. Детальная имитация в реальном масштабе времени основной физической среды выполняется в конечном кластере с активно-сетевыми и параллельными технологиями имитации. Связь двух кластеров осуществляется через скоростное соединение, позволяющее выполнять управление в реальном масштабе времени (Myrinet). Трафик данных обрабатывается через отдельную сеть передачи данных, подобной сети испытательного стенда MobiEmu. Трафик данных обработан отдельной сетью передачи данных, подобно сети тестового стенда в MobiEmu. Кроме того, WiNE дает возможность тестировать свойства безопасности MANET [HZR]. Платформа WiNE включает инструменты для имитации атак злоумышленников, инструменты для измерения эффективности и производительности решений для обеспечения безопасности. Платформа WiNE также включает модели для анализа таксономии атак и для разработки сценарии атак.

APE - Ad hoc Protocol Evaluation testbed

Интегрированная среда APE (Ad hoc Protocol Evaluation testbed) [LLN02] предназначена для тестирования протоколов маршрутизации в сети с большим количеством узлов в реальных условиях, а не с помощью симуляции. Узлы в тестируемой мобильной сети перемещаются согласно заранее определенным сценариям, что позволяет обеспечивать воспроизводимые тестовые прогоны. APE имеет модульную архитектуру, которая позволяет подключать к экспериментам различные протоколы маршрутизации и генераторы потока информации в сети. атак.

Заключение

Компьютерную безопасность как область специализации довольно трудно точно определить. Даже профессионалы, работающие в области компьютерной безопасности, испытывают трудности при определении этого понятия. Отчасти причина заключается в том, что безопасность сложно описать, поскольку это понятие затрагивает большое количество сфер деятельности в компьютерных технологиях. Ближе всего, по всей вероятности, безопасность касается разработки программного обеспечения – компьютерная безопасность призвана гарантировать, что программное и аппаратное обеспечение удовлетворяет своим спецификациям и требованиям при использовании его в потенциально враждебной среде. Компьютерная безопасность, таким образом, включает вопросы спецификации, верификации, тестирования, валидации, надежности и живучести компьютерной системы. Однако безопасность охватывает гораздо больший круг проблем, касающихся проектирования операционных систем, архитектурного проектирования, информационной безопасности, анализа и предсказания рисков, организации баз данных, шифрования и кодирования, формальной модели вычислений, отказоустойчивости, проектирования интерфейсов, постановлений и политики государства, административных решений, компетентность в вопросах безопасности, а также соответствующее образование.

Для более точного определения круга проблем, относящихся к безопасности, в начале обзора рассматриваются стандарты безопасности, как наборы требований к «безопасным» система. Перечисляются свойства безопасности. Для каждого свойства безопасности рассматриваются различные его модели – как набор относящихся к проблеме аспектов явления. Эти модели позволяют абстрагироваться от не относящихся к проблеме, хотя и важных с других точек зрения, частностей, сконцентрироваться на наиболее существенном. Как отдельный круг проблем, имеющих наиболее давнии традиции в изучении и решении, рассматриваются криптографические протоколы.

Затем рассматриваются способы формального описания моделей: спецификационные языки (универсальные и специального назначения), алгебры процессов, конечные автоматы , модальные логики, сети Петри, графическое моделирование.

Этими способами формального описания определяются методы формальной верификации свойств безопасности, описываемые в следующей главе. К ним относятся: верификация на основе конечных автоматов, проверка модели (model-checking), доказательство теорем (theorem proving), метод проверки типа (type checking). Рассматриваются и другие подходы.

Как необходимое дополнение формальной верификации рассматривается и тестирование свойств безопасности. Оно может производиться как на на основе спецификаций, так и на других основах.

В этой же главе описаны и наиболее характерные инструменты, поддерживающие верификацию свойств безопасности.

Ускоряющееся внедрение мобильных устройств в бизнесс-процессы и повседневную жизнь требует внимательного изучения концепции мобильности. Очевидное преимущество переносимых компьютеров общепризнанно, поскольку они резко улучшают эффективность служащего, который, вооружившись таким компьютером с простым подключением к беспроводной сети, теперь способен оставаться на связи постоянно, независимо от его передвижения по земному шару. Появление небольших карманных компьютеров и смартфонов еще более увеличило интерес к мобильности. Достаточно мощные для выполнения основных корпоративных приложений, эти доступные переносные устройства позволяют профессионалам увеличивать продолжительность активного состояния, улучшать продуктивность и расширять связи с заказчиками. Но увеличение мобильности влечет за собой повышение рисков.

Доля мобильных устройств в общем распределении компьютерного оснащения растет лавинообразно. Организации внедряют мобильные и беспроводные устройства быстрее, чем любую другую платформу. Гартнер (Gartner), аналитик в области индустрии, предсказывает, что к 2007 году во всем мире будет около 120 000 выходов в беспроводную сеть, что будет обеспечивать доступ к частным и общественным сетям более 200 миллионам мобильных устройств, используемых в бизнесе. Гартнер также предсказывает, что более 60% сотрудников в 2000 мировых компаниях будут иметь мобильный доступ к корпоративным приложениям, и что 40% корпоративных данных будут располагаться на переносных устройствах к 2005 году.

Проблемы безопасности становятся еще более острыми системах мобильных устройств. Но кроме этого появляются проблемы, свойственные этому классу распределенных систем. Во второй части обзора рассматриваются именно проблемы мобильности.

Структура второй части повторяет структуру первой: рассматриваются модели мобильности, способы их формального описания, верификации и тестирования.

Общая, качественная оценка состояния и перспективности различных направлений исследований в области верификации функций безопасности и мобильности, основанная на анализе состояния исследований и разработок в этой области (всего было проанализировано свыше 280 работ), представляется следующей:

- Ни одна из методик и, соответственно, ни один из инструментов верификации функций безопасности и мобильности не играет доминирующую роль, используются как методики аналитического анализа, так динамического – тестирование.
- Инструментов и методик, которые позволяют провести строгую аналитическую верификацию реальных функций безопасности и мобильности пока нет. Аналитические подходы, включая аналитическую проверку моделей, применяются только для упрощенных моделей или для крайне простых реализаций. Достаточно общих подходов, предлагающих синтетические решения, которые совместно используют аналитические и динамические методы верификации пока нет.

  Но, тем не менее, работы ведутся чрезвычайно активно.

Библиография

[AB96] A. Asperti and N. Busi. Mobile Petri Nets. Technical Report UBLCS9610, University of Bologna, 1996.

[Aba99] M. Abadi. Secrecy by typing in security protocols. Journal of the ACM, 46(5):749–786, September 1999.

[ABB02] L. Andrade, P. Baldan, H. Baumeister, R. Bruni, A Corradini, R. De Nicola, J. L. Fiadeiro, F. Gadducci, S. Gnesi, P. Hoffman, N. Koch, P. Kosiuczenko, A. Lapadula, D. Latella, A. Lopes, M. Loreti, M. Massink, F. Mazzanti, U. Montanari, C. Oliveira, R. Pugliese, A. Tarlecki, M. Wermelinger, M. Wirsing, and A. Zawlocki. AGILE: Software architecture for mobility. In Martin Wirsing, Dirk Pattinson, and Rolf Hennicker, editors, Recent Trends in Algebraic Develeopment Techniques, 16th International Workshop, WADT 2002, Frauenchiemsee, Germany, Sept. 24-27, 2002, volume 2755 of LNCS, Springer, November 2003.

[ACG03] A. Armando, L. Compagna, and P. Ganty. SAT-based Model-Checking of Security Protocols using Planning Graph Analysis. In Proc. FM, FM 2003: the 12th International Formal Methods Europe Symposium Pisa, Italy, 2003.

[AG97] M. Abadi and A. D. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. In Proc. Fourth ACM Conference on Computer and Communications Security. ACM Press, April 1997.

[AL00] R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In CONCUR00, number 1877 in LNCS. Springer, 2000.

[AS85] B. Alpern and F. B. Schneider. Defining liveness. Information Processing Letters, 21(4):181-185, October 1985.

[AS86] B. Alpern and F. B. Schneider. Recognizing safety and liveness. Technical Report TR86-727, Cornell University, Computer Science Department, January 1986.

[AT91] M. Abadi and M. Tuttle. A semantics for a logic of authentication. In Proc. Tenth Annual ACM Symposium on Principles of Distributed Computing, pp. 201-216, 1991.

[BAN90] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. ACM Transactions in Computer Systems, 8(1):18–36, February 1990.

[Bas99] D. Basin. Lazy infinite-state analysis of security protocols. In Proc. CQRE'99, LNCS 1740, pp. 30–42. Springer, 1999.

[BB92] G. Berry and G. Boudol. The chemical abstract machine. Theoretical Computer Science, 96(1):217–248, April 1992.

[BC94] P. Bieber and N. B. Cuppens. Formal Development of Authentication Protocols. In Proc. BCS-FACS 6th Refinemment Workshop on software Engineering and its Applications, January 1994.

[BCC01] M. Bugliesi, G. Castagna, and S. Crafa. Boxed ambients. In TACS 2001, 4th. International Symposium on Theoretical Aspects of Computer Science, number 2215 in LNCS. Springer-Verlag, 2001.

[BCF02] C. Braghin, A. Cortesi, and R. Focardi. Control Flow Analysis of Mobile Ambients with Security Boundaries. In B. Jacobs and A. Rensink, editors, Proc. of Fifth Int. Conf. on Formal Methods for Open Object-Based Distributed Systems, pp. 197–212. Kluwer Academic Publisher, 2002.

[BCFLP03] C. Braghin, A. Cortesi, R. Focardi, F. Luccio, and C. Piazza. Complexity of Nesting Analysis in Mobile Ambients. In L. D. Zuck, P. C. Attie, A. Cortesi, and S. Mukhopadhyay eds., Proc. of Int. Conference on Verification Model Checking and Abstract Interpretation (VMCAI'03), pp. 86-101, vol. 2575, LNCS, Springer-Verlag, Berlin, 2003.

[BCLW93] P. Bieber, N. B. Cuppens, T. Lehman, and E. van Wickeren. Abstract Machines for Communication Security. In Proc. IEEE Computer Security Foundation Workshop VI, June 1993

[BDNN01] C. Bodei, P. Degano, F. Nielson, and H. R. Nielson. Static analysis for the π-calculus with applications to security. Information and Computation, vol. 168, pp. 68-92, 2001.

[BDNN01a] C. Bodei, P. Degano, H. R. Nielson, and F. Nielson. Static analysis for secrecy and non-interference in networks of processes. In Proc. PACT'01, Sept. 2001, vol. 2127 of LNCS, pp. 27–41, Springer-Verlag.

[Beiz90] B. Beizer. Software Testing Techniques. Second Edition, New York, Van Nostrand Reinhold, 1990.

[BFPR02] A. Bossi, R. Focardi, C. Piazza, S. Rossi, A Proof System for Information Flow Security, in: M. Leuschel (Ed.), Proc. of Int. Workshop on Logic Based Program Development and Transformation (LOPSTR'02), Vol. 2264 of LNCS, Springer-Verlag, 2002, pp. 199-218.

[BFPR04] A. Bossi, R. Focardi, C. Piazza, and S. Rossi. Verifying Persistent Security Properties. Computer Languages Systems and Structures. In Press, 2004.

[Bib77] K. J. Biba. Integrity considerations for secure computer systems. Tech. Rep. ESD-TR-76-372, USAF Electronic Systems Division, Bedford, MA, April 1977.

[Bie90] P. Bieber. A logic of communication in a hostile environment. In Proc. IEEE CS Computer Security Foundations Workshop, pp. 14-22, June 1990.

[BJBG04] R. Bhatti, J. B. D. Joshi, E. Bertino, A. Ghafoor. XML-Based Specification for Web Services Document Security. IEEE Computer, Vol. 37, Number 4, April 2004, pp. 41-49.

[BKKW02] H. Baumeister, N. Koch, P. Kosiuczenko, and M. Wirsing. Extending activity diagrams to model mobile systems. In M. Aksit, M. Mezini, and R. Unland, editors, Objects, Components, Architectures, Services, and Applications for a Networked World. International Conference NetObjectDays, NODe 2002, Erfurt, Germany, Oct. 7-10, 2002. Revised Papers, volume 2591 of LNCS, pp. 278-293. Springer, 2003.

[BL75] D.E. Bell and L.J. LaPadula. Secure computer systems: Unified exposition and Multics interpretation. Technical Report MTR-2997, Mitre, Bedford, MA, 1975.

[Bla03] B. Blanchet. Automatic Verification of Cryptographic Protocols: A Logic Programming Approach (invited talk). In 5th ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP'03), pp. 1-3, Uppsala, Sweden, August 2003. ACM.

[BLP04] M. ter Beek and G. Lenzini and M. Petrocchi. Team Automata for Security Analysis. Centre for Telematics and Information Technology, Univ. of Twente, The Netherlands, Technical report (TR-CTIT-04-13), Feb 2004.

[BM93] C. Boyd and W. Mao. On a limitation of BAN logic. In Advances in Cryptology – EUROCRYPT'93, pp. 240-247, 1993.

[BMV03] D. Basin, S. Mödersheim, L. Viganò. An On-The-Fly Model-Checker for Security Protocol Analysis. in Proc. of ESORICS'03, LNCS 2808, pp. 253-270, Springer-Verlag, 2003.

[Bol96] D. Bolignano. An Approach to the Formal Verification of Cryptographic Protocols. In Proc. Third ACM Conference on Computer and Communications Security, CCS'96, New Delhi, India, pp. 106-118. ACM Press, 1996.

[Bou98] A. Bouali, XEVE, an ESTEREL verification environment, in: A. J. Hu, M. Y. Vardi (Eds.), Proc. of Int. Conference on Computer Aided Verification (CAV'98), Vol. 1427 of LNCS, Springer-Verlag, 1998, pp. 500-504.

[Boy90] C. Boyd. Security Architectures Using Formal Methods. Journal on Selected Areas in Communications, 11(5):694-701, June 1990.

[Boy92] C. Boyd. A Framework for Authentication. In Proc. European Symposuim on Research in Computer Security, ESORICS 92, volume 648 of Lecture Notes in Computer Science, pp. 273-292. Springer Verlag, November 1992.

[Bra96] S. Brackin. A HOL Extension of GNY for Automatically Analyzing Cryptographic Protocols, In Proc. 1996 IEEE Computer Security Foundations Workshop IX, (1996) 62-76, IEEE Computer Society Press.

[Bra96a] S. Brackin. Automatic Formal Analyses of Cryptographic Protocols, In Proc. 19th National Conference on Information Systems Security, MD, (1996), IEEE Computer Society Press.

[Bra97] S. Brackin. An Interface Specification Language for Automatically Analyzing Cryptographic Protocols, In Proc. 1997 Symposium on Network and Distributed System Security, (1997) 40-51, IEEE Computer Society Press.

[Bra98] S. Brackin. Evaluating and improving protocol analysis by automatic proof. In Proc. 11th IEEE Computer Security Foundations Workshop. IEEE Computer Society Press, June 1998.

[BS01] M. Broy and K. Stolen, editors. Specification and Development of Interactive Systems. Springer, 2001.

[But99] L. Buttyan. Formal methods in the design of cryptographic protocols. Technical Report SSC/1999/038, Swiss Federal Institute of Technology, Institute for computer Communications and Applications (ICA), November 1999.

[BW00] M. Broy, M. Wirsing. Invited Talk: Algebraic State Machines. AMAST 2000: 89-188.

[Car95] L. Cardelli. A language with distributed scope. In Proc. 22nd ACM Symposium on Principles of Programming Languages (POPL), pp. 286-298, 1995.

[CASPR] CASPR http://www.caspr.org/.

[CB04] R. Chandramouli, M. R. Blackburn. Automated Testing of Security Functions Using a Combined Model and Interface-Driven Approach. Proc. of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04), January 05 - 08, 2004.

[CC99] Common Criteria for Information Technology Security Evaluation (CC), Version 2.1, 1999. ISO/IEC 15408. http://csrc.nist.gov/cc/CC-v2.1.html.

[CCS03] M. Carvalho, T. Cowin, N. Suri. MAST - A Mobile Agent-based Security Tool. Proc. of the 7th World Multiconference on Systemics, Cybernetics and Informatics. Orlando, FL, October 2003.

[CDL00] I. Cervesato, N. A. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. Relating strands and multiset rewriting for security protocol analysis. In Proc. of the 13th IEEE Computer Security Foundations Workshop (CSFW '00), pp. 35-51. IEEE, 2000.

[CDM01] A. Chander, D. Dean, J. Mitchell. A State-transition Model of Trust Management and Access Control. In Proc. 14th Computer Security Foundations Workshop (CSFW), June 2001.

[CEC93] The commission of the European Communities CEC DG-XIII. Security Investigation Final Report. Technical Report S2011/7000/D010 7000 1000, CEC, September 1993.

[CFN03] A. J. Canas, K. M. Ford, J. D. Novak, P. Hayes, T. Reichherzer, N. Suri., Using Concept Maps with Technology to Enhance Collaborative Learning in Latin America, February 2003, http://www.coginst.uwf.edu/users/acanas/Publications/QuorumSoupST/SoupsST.htm

[CFM98] P. Ciancarini, F. Franze, and C. Mascolo. A Coordination Model to Specify Systems including Mobile Agents. In Proc. 9th IEEE Int. Workshop on Software Specification and Design (IWSSD), pp. 96-105, Japan, 1998.

[CG00] L. Cardelli and A.D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In Proc. POPL'00, pp. 365-377. ACM, 2000.

[CG90] P.-C. Cheng, V. D. Gligor: On the Formal Specification and Verification of a Multiparty Session Protocol. IEEE Symposium on Security and Privacy 1990: 216-233.

[CG98] L. Cardelli and A. D. Gordon. Mobile ambients. In Proc. FOSSACS '98, vol. 1378 of LNCS, pp. 140-155, Springer-Verlag, 1998.

[CGZ95] N. Carriero, D. Gelernter, and L. Zuck. Bauhaus Linda. In P. Ciancarini, O. Nierstrasz, and A. Yonezawa, editors, Object-Based Models and Languages for Concurrent Systems, volume 924 of Lecture Notes in Computer Science, pp. 66-76. Springer-Verlag, Berlin, 1995.

[CJM00] E. M. Clarke, S. Jha, and W. Marrero. Verifying security protocols with Brutus. ACM Transactions on Software Engineering and Methodology, 9(4):443-487, 2000.

[CJRTV] Y. Chevalier, F. Jacquemard, M. Rusinowitch, M. Turuani, and L. Vigneron. CASRUL web site. http://www.loria.fr/equipes/protheo/SOFTWARES/CASRUL/.

[CM03] I. Cervesato and C. Meadows. A fault-tree representation of NPATRL security requirements. Workshop on Issues in the Theory of Security (WITS'03), 2003.

[CM88] K. Chandy and J. Misra. Parallel Program Design: A Foundation. Addison-Wesley, Reading, MA, 1988.

[Coh00] E. Cohen. TAPS: a first-order verifier for cryptographic protocols. In Proc. 13th IEEE Computer Security Foundations Workshop, pp. 144–158. IEEE Computer Society Press, June 2000.

[CS96] R. Cleaveland, S. Sims. The NCSU Concurrency Workbench, in: R. Alur, T. Henzinger (Eds.), Proc. of Int. Conference on Computer Aided Verification (CAV'96), Vol. 1102 of LNCS, Springer-Verlag, 1996, pp. 394-397.

[CSP92] E. A. Campbell, R. Safavi-Naini, and P. A. Pleasants. Partial belief and probabilistic reasoning in the analysis of secure protocols. In Proc. Computer Security Foundation Workshop V, pp. 84-91, Washington, 1992.

[CV01] Y. Chevalier and L. Vigneron. A tool for lazy verification of security protocols. In Proc. 16th IEEE International Conference Automated Software Engineering, 2001.

[CW01] F. Crazzolara and G. Winskel. Petri nets in cryptographic protocols. In Proc. 16th International Parallel & Distributed Processing Symposium – 6th International Workshop on Formal methods for Parallel Programming: Theory and Practice, San Francisco, April 2001. IEEE Press.

[CZGMT01] W. Charatonik, S. Dal Zilio, A. Gordon, S. Mukhopadhyay, and J.M. Talbot. The complexity of model checking mobile ambients. In FoSSaCS, April 2001.

[Dam01] M. Dam. Proof systems for π-Calculus Logics. To appear on Logics for Concurrency and Synchronization. Studies in Logics and Computation, Oxford University Press, 2001.

[Dam93] M. Dam. Model checking mobile processes. In Proc. CONCUR'93, LNCS 715. Springer Verlag, 1993.

[DD77] D. E. Denning and P. J. Denning. Certification of programs for secure information flow. Comm. of the ACM, vol. 20, no. 7, pp. 504–513, July 1977.

[DEK82] D. Dolev, S. Even, and R. Karp. On the Security of Ping-Pong Protocols. Information and Control, pp. 57-68, 1982.

[DFG99] A. Durante, R. Focardi, and R. Gorrieri. CVS: A compiler for the analysis of cryptographic protocols. In Proc. 12th IEEE Computer Security Foundations Workshop, pp. 203-212. IEEE Computer Society Press, June 1999.

[DFP98] R. DeNicola, G. Ferrari, and R. Pugliese. KLAIM: A kernel Language for Agents. Interaction and Mobility. IEEE Transactions on Software Engineering, 24(5):315-330, 1998.

[Dier99] T. Dierks, C. Allen. The TLS Protocol. Version 1.0. – RFC 2246, January 1999.

[DIJK76] E. W. Dijkstra. A Discipline of Programming, Prentice Hall Series in Automatic Computation, Prentice-Hall Inc. Englewood Cliffs, NJ, 1976.

[DK97] Z. Dang and R. A. Kemmerer. Using the ASTRAL model checker for cryptographic protocol analysis. In Proc. DIMACS Workshop on Design and Formal Verification of Security Protocols, September 1997.

[DL00] R. DeNicola and M. Loreti. A modal logic for Klaim. In T. Rus, editor, Proc. Algebraic Methodology and Software Technology (AMAST 2000), volume 1816 of Lecture Notes in Computer Science, pp. 339–354, Iowa, 2000. Springer-Verlag.

[DM98] W. Du and A. P. Mathur. Vulnerability Testing of Software System Using Fault Injection. Department of Computer Sciences, Purdue University; Coast TR 98-02; 1998.

[DMR00] G. Denker, J. Millen, and H. Ruess. The CAPSL integrated protocol environment. Technical Report SRI-CSL-2000-02, SRI International, 2000.

[DoD85] Department of Defense Trusted Computer System Evaluation Criteria. – DoD 5200.28 – STD, December 26, 1985.

[DS81] D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. Communications of the ACM, 24(8):533-536, August 1981.

[DS97] B. Dutertre and S. Schneider. Using a PVS embedding of CSP to verify authentication protocols. In TPHOLS'97, 1997.

[Dup90] A. Dupuy et al. NEST: A Network Simulation and Prototyping Testbed, Comm. ACM, Oct. 1990, pp. 64-74.

[DV90] R. DeNicola and F. W. Vaandrager. Action versus state based logics for transition systems. In Proc. Ecole de Printemps on Semantics of Concurrency, LNCS 469. Springer Verlag, 1990.

[DY83] D. Dolev and A. Yao. On the Security of Public Key Protocols. IEEE Transactions on Information Theory, 29(2):198-208, March 1983.

[EG83] S. Even and O. Goldreich. On the security of multi-party ping-pong protocols. In Proc. 24th IEEE Symposium on the Foundations of Computer Science, pp. 34-39. IEEE Computer Society Press, 1983.

[FA01] M. Fiore and M. Abadi. Computing symbolic models for verifying cryptographic protocols. In 14th IEEE Computer Security Foundations Workshop, pp. 160-173. IEEE Computer Society, 2001.

[Fal99] K. Fall. Network emulation in the VINT/ns simulator. In Proc. 4th IEEE Symposium on Computers and Communications (ISCC'99), July 1999.

[FB97] G. Fink and M. Bishop. Property Based Testing: A New Approach to Testing for Assurance. ACM SIGSOFT Software Engineering Notes, 22(4) (July 1997).

[Fei80] R. J. Feiertag. A technique for proving specifications are multilevel secure. Tech. Rep. CSL-109, SRI International Computer Science Lab, Menlo Park, California, Jan. 1980.

[Fer94] G. Ferro. AMC: ACTL Model Checker. Reference Manual. IEI-Internal Report, B4-47, 1994.

[FG01] R. Focardi, R. Gorrieri, Classification of Security Properties (Part I: Information Flow), in: R. Focardi, R. Gorrieri (Eds.), Foundations of Security Analysis and Design, Vol. 2171 of LNCS, Springer-Verlag, 2001.

[FG94] R. Focardi, R. Gorrieri, A Classification of Security Properties for Process Algebras, Journal of Computer Security 3 (1) (1994/1995) 5-33.

[FG97] R. Focardi, R. Gorrieri, The Compositional Security Checker: A Tool for the Verification of Information Flow Security Properties. IEEE Transactions on Software Engineering 23 (9) (1997) 550-571.

[FGLMR96] C. Fournet, G. Gonthier, J. Levy, L. Maranget, and D. Remy. A Calculus of Mobile Agents. In U. Montanari and V. Sassone, editors, Proc. 7th Int. Conf. on Concurrency Theory (CONCUR), volume 1119 of Lecture Notes in Computer Science, pp. 406-421, Pisa, Italy, August 1996. Springer-Verlag, Berlin.

[FGMP03] G. Ferrari, S. Gnesi, U. Montanari, M. Pistone. A model checking verification environment for mobile processes. Transactions on Software Engineering and Programming Methodologies, pp. 1-37, 2003.

[FK92] D. F. Ferraiolo and D. R. Kuhn. Role-Based Access Controls. In Proc. 15th NIST-NSA National Computer Security Conference, Baltimore, Maryland, October 13-16, 1992.

[FM91] J.-C. Fernandez and L. Mounier. “On the fly” verification of behavioral equivalences and preorders. In Proc. CAV’91, LNCS 575. Springer Verlag, 1991.
[FMP03] G. Ferrari, E. Moggi, and R. Pugliese. Higher-order types and meta-programming for global computing. Mathematical Structures in Computer Science, 2003.
[FR02] R. Focardi, S. Rossi, Information Flow Security in Dynamic Contexts, in: Proc. of the 15th IEEE Computer Security Foundations Workshop (CSFW'02), IEEE Computer Society Press, 2002, pp. 307-319.
[GA98] A. Gordon and M. Abadi. A bisimulation method for cryptographic protocols. In Proc. ESOP'98, Springer LNCS, 1998.
[GD72] G. Graham and P. Denning. Protection – principles and practice. In Proc. Spring Joint Computer Conference. AFIPS Press, 1972.
[GJ01] A. Gordon and A. Jeffrey. Authenticity by typing in security protocols. In Proc. 14th IEEE Computer Security Foundations Workshop. IEEE Computer Society Press, June 2001.
[GJR96] P. Gardiner, D. Jackson, and B. Roscoe. Security Modelling in CSP and FDR: Deliverable Bundle 3. Technical report, Formal Systems (Europe) Ltd, July 1996.
[GK00] T. Genet and F. Klay. Rewriting for cryptographic protocol verification. In D. McAllester, editor, Automated Deduction (CADE-17), volume 1831 of Lecture Notes in Artificial Intelligence. Springer-Verlag, 2000.
[GK91] C. Ghezzi, R. Kemmerer. ASTRAL: An assertion language for specifying real-time systems, In Proc. Third European Software Engineering Conference, (1991) 122-146.
[GL00] F. Germeau, G. Leduc. Verification of security protocols using lotosmethod and application. In Computer Communications, volume 23, pp. 1089-1103. Elsevier Science B.V., 2000.
[Gli83] V. Gligor. A note on the denialofservice problem. In Proc. 1983 IEEE Symposium on Research on Security and Privacy. IEEE Computer Society Press, 1983.
[GLL02] S. Gnesi and D. Latella and G. Lenzini. Towards Model Checking a Spi-Calculus Dialect, Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" (ISTI-CNR), Pisa, Italy, Technical report (2002-TR-10), Jul 2002.
[GM82] J. Goguen and J. Meseguer. Security policies and security models. In Proc. IEEE Symposium on Research in Security and Privacy. IEEE Computer Society Press, pp. 11-20, April 1982.
[GM84] J. Goguen and J. Meseguer. Unwinding and inference control. In Proc. 1984 IEEE Symposium on Research on Security and Privacy. IEEE Computer Society Press, 1984.
[GM93] Gordon M., Melham T., Introduction to HOL: A Theorem Proving Environment for Higher Order Logic, (1993), Cambridge University Press, UK.
[GNY90] L. Gong, R. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In Proc. IEEE Computer Society Symposium on Security and Privacy, pp. 234-248, May 1990.
[GOR02] S. Gürgens, P. Ochsenschläger, C. Rudolph: 'Role based specification and security analysis of cryptographic protocols using asynchronous product automata', GMD-Report 151, March 2002.
[Gou00] J. Goubault-Larrecq. A method for automatic cryptographic protocol verification. In Dominique Mery Beverly Sanders, editor, Beverly Sanders, Dominique M Fifth International Workshop on Formal Methods for Parallel Programming: Theory and Applications (FMPPTA 2000), number 1800 in Lecture Notes in Computer Science. Springer-Verlag, 2000.
[GR00] S. Gnesi and G. Ristori. A Model Checking Algorithm for π-calculus agents. In Advances in Temporal Logic. Kluwer Academic Publishers, pp.339-357, 2000.
[GS91] K. Gaarder and E. Snekkenes. Applying a formal analysis technique to the CCITT X.509 strong two-way authentication protocol. Journal of Cryptology, 3:81-98, 1991.

[GSG99] S. Gritzalis, D. Spinellis, and P. Georgiadis. Security protocols over open networks and distributed systems: Formal methods for their analysis, design, and verification. Computer Communications Journal, 22(8):695-707, 1999.

[Gur97] Y. Gurevich. Draft of the asm guide. Technical Report CSE-TR-336-97, EECS Dept., University of Michigan, 1997.

[HKLB98] C. Heitmeyer, J. Kirby, B. Labaw and R. Bharadwaj. SCR: A toolset for specifying and analyzing software requirements, Proc. 10th Annual Conference on Computer-Aided Verification, Vancouver, Canada, 1998.

[HKMNSSW88] J.H. Howard, M.L. Kazar, S.G. Menees, D.A. Nichols, M. Satyanarayanan, R.N. Sidebotham, and M.J. West. Scale and performance in a distributed file system. ACM Transactions on Computer Systems, 6(1):51-81, February 1988.

[Hoa80] C. A. R. Hoare. Communicating sequential processes. In R. M. McKeag and A. M. Macnaghten, editors, On the construction of programs – an advanced course, pp. 229-254. Cambridge University Press, 1980.

[Holland] G. Holland. Operational Effectiveness Evaluation with In-House, High-Fidelity, and Real-Time Wireless Network Emulator (WiNE). http://darpa.srs.com/ato/files/Ryu,Holland%20HRL.pdf.

[HRU76] M. Harrison, W. Ruzzo, and J. Ullman. Protection in operating systems. Communications of the ACM, 19(8):461-471, August 1976.

[HSSS96]. F. Huber, B. Schatz, A. Schmidt, and K. Spies. Autofocus – a tool for distributed systems specification. In Proc. FTRTFT'96 – Formal Techniques in Real-Time and Fault-Tolerant Systems, volume 1135 of LNCS, pp. 467-470. Springer-Verlag, 1996.

[Hui99] A. Huima. Efficient infinite-state analysis of security protocols. Presented at FLOC'99 Workshop on Formal Methods and Security Protocols, July 1999.

[HZR] G. Holland, Y. Zhang. B. Ryu. Wireless Security Testbed based on WiNE. http://darpa.srs.com/ato/files/HRL%20Ryu.pdf

[Intrus] http://advice.networkice.com/advice/Intrusions/.

[JRV00] F. Jacquemard, M. Rusinowitch, and L. Vigneron. Compiling and verifying security protocols. In M. Parigot and A. Vonkorov, editors, Proc. LPAR: International Conference on Logic for Programming and Automated Reasoning, number 1995 in LNCS, pp. 131–160. Springer-Verlag, 2000.

[JRV01] J. B. D. Joshi, W. G. Aref, A. Ghafoor and E. H. Spafford. Security models for web-based applications, Communications of the ACM, 44, 2 (Feb. 2001), pp. 38-44.

[JT88] D. M. Johnson and F. J. Thayer. Security and the Composition of Machines. In Proc. Security Foundations Workshop, Franconia, NH, pp. 72-89, June 1988.

[Jur01a] J. Jurjens. Developing secure systems with UMLsec – from business processes to implementation. In VIS 2001. Vieweg-Verlag, 2001.

[Jur01b] J. Jurjens. Towards development of secure systems using UMLsec. In H. Hubmann, editor, Fundamental Approaches to Software Engineering (FASE/ETAPS, International Conference), volume 2029 of LNCS, pp. 187-200. Springer-Verlag, 2001.

[Jur01c] J. Jürjens, Secure Java Development with UML, I-NetSec 01 – First International IFIP TC-11 WG 11.4 Working Conference on Network Security, Leuven (Belgium), November 26-27, 2001.

[Jur02] J. Jürjens, UMLsec: Extending UML for Secure Systems Development, UML 2002, Dresden, Sept. 30 – Oct. 4, 2002, LNCS, © Springer-Verlag.

[Jur03] J. Jürjens. Algebraic state machines: Concepts and applications to security. In Andrei Ershov 5th International Conference "Perspectives of System Informatics" (PSI'03), LNCS. Springer-Verlag, 2003.

[JW01] J. Jürjens and G. Wimmel. Specification-based testing of firewalls. In Andrei Ershov 4th International Conference "Perspectives of System Informatics" (PSI'01), LNCS. Springer, 2001.

[Kar98] T. Karygiannis. Network Security Testing Using Mobile Agents, The Third International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agent Technology, London, UK, March 1998.

[Kem89] R. A. Kemmerer. Analyzing encryption protocols using formal verification techniques. IEEE Journal on Selected areas in Communications, 7(4):448-457, May 1989.

[Ken98] S. Kent, R. Atkinson. Security Architecture for the Internet Protocol. – RFC 2401, November 1998.

[Kes88] S. Keshav. REAL: A Network Simulator, tech. report 88/472, Univ. California, Berkeley, 1988.

[Kin99] D. Kindred. Theory generation for security protocols. Technical Report CMU-CS-99130, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1999. Ph.D. thesis.

[KLT00] R. Kaksonen, M. Laakso, A. Takanen. Vulnerability Analysis of Software through Syntax Testing. Technical Research Centre of Finland, 2000.

[KO03] T. A. Kuhn, D. von Oheimb: Interacting State Machines for Mobility. FME 2003: 698-718.

[Koh93] J. Kohl. The Kerberos Network Authentication Service (V5). – Request for Comments: 1510, September 1993.

[Koz83] D. Kozen, Results on the Propositional μ-calculus, Theoretical Computer Science 27 (1983) 333-354.

[KR01] J. Kaba and D. Raichle. Testbed on a desktop: Strategies and techniques to support multi-hop manet routing protocol development. Proc. of the 2001 ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'01), Long Beach, California, Oct. 2001.

[KS99] P. Karn and W. Simpson. Photuris: Session-key management protocol. RFC 2522, Internet Engineering Task Force, March 1999.

[Lam71] B. Lampson. Protection. In 5th Princeton Symposium on Information Sciences and Systems, March 1971. Reprinted in ACM Operating System Review, 8(1), pp. 18-24, 1974.

[Lam73] B. W. Lampson. A note on the confinement problem. Comm. of the ACM, vol. 16, no. 10, pp. 613–615, Oct. 1973.

[Lam94] L. Lamport. The Temporal Logic of Actions. ACM Transactions on Programming Languages and Systems, vol. 16(3), pp. 872-923, May 1994.

[LG00] G. Leduc and F. Germeau. Verification of security protocols using LOTOS-method and application. Computer Communications, 2000.

[LGL03] G. Lenzini, S. Gnesi, D. Latella. Spider: a Security Model Checker., 1st Int. Workshop on Formal Aspect of Security and Trust (FAST), published by Istituto di Informatica e Telematica (IIT-CNR), Pisa, Italy, held in Pisa, Italy, Sep., 2003, pp. 163-180.

[LHM84] C. Landwehr, C. Heitmeyer, and J. McLean. A security model for military message systems. ACM Transactions of Computer Systems, 2(3):198-222, August 1984.

[Lin00] J. Linn. Generic Security Service Application Program Interface. Version 2, Update 1. – RFC 2743, January 2000.

[LLNTN02] H. Lundgren, D. Lundberg, E. Nordstrm, C. Tschudin, and J. Nielsen. A large-scale testbed for reproducible ad hoc protocol evaluations. In Proc. IEEE Wireless Communications and Networking Conference (WCNC 2002), Orlando, Florida, Mar. 2002.

[LMBW03] D. Latella, M. Massink, H. Baumeister, and M. Wirsing. Mobile UML statecharts with localities. Technical Report 2003-TR-37, CNR-ISTI, 2003.

[Low96] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In Proc. TACAS, (1996) 147-166, Springer Verlag.

[Low97] G. Lowe. Casper: A compiler for the analysis of security protocols. Proc. of the 1997 IEEE Compiler Society Symposium on Research in Security and Privacy, pp. 18-30. 1997.

[Low99] G. Lowe. Towards a completeness results for model checking security protocols. Journal of Computer Security, 7(2,3):89–146, 1999.

[LR92] D. Longley and S. Rigby. An automatic search for security flaws in key management schemes. Computers and Security, 11(1):75–90, 1992.

[LT89] N. Lynch and M. Tuttle. An introduction to input/output automata. CWI Quarterly, 2(3):219–246, 1989.

[LTR99] M. Laakso, A. Takanen, J. Röning. The Vulnerability Process: a tiger team approach to resolving vulnerability cases. Proc. of the 11th FIRST Conference on Computer Security Incident Handling and Response, Brisbane. 13-18 June, 1999.

[LVH03] S. Lukell, C. Veldman, and A. C. M. Hutchison. Automated attack analysis and code generation in a multi-dimensional security protocol engineering framework. In Southern African Telecommunication Networks and Applications Conference, George, South Africa, September 2003. SATNAC.

[Man02] H. Mantel. On the Composition of Secure Systems. In Proc. of the IEEE Symposium on Security and Privacy (SSP'02), IEEE Computer Society Press, 2002, pp. 88-101.

[Mar01] W. Marrero. BRUTUS: A Model Checker for Security Protocols, Ph.D. Thesis, CMU-CS-01-170, School of Computer Science, Carnegie Mellon University, December 2001.

[Mar97] W. Marrero. A model checker for authentication protocols. In Proc. DIMACS Workshop on Design and Formal Verification of Security Protocols, September 1997.

[Mar98] F. Martinelli, Partial Model Checking and Theorem Proving for Ensuring Security Properties. In Proc. of the IEEE Computer Security Foundations Workshop (CSFW'98), IEEE Computer Society Press, 1998, pp. 44-52.

[Mas99a] C. Mascolo. Mobis: A specification language for mobile systems. In LNCS. Springer-Verlag, 1999.

[Mas99b] C. Mascolo. Specification, analysis, and prototyping of mobile systems. In Doctoral Symposium of the 21st International Conference on Software Engineering. Los Angeles, CA. IEEE, 1999.

[MB93] W. Mao and C. Boyd. Towards formal analysis of security protocols. In Proc. Computer Security Foundation Workshop VI, pp. 147-158, June 1993.

[MBJ01] D. A. Maltz, J. Broch, and D. B. Johnson. Lessons From a Full-Scale Multi-Hop Wireless Ad Hoc Network Testbed. In IEEE Personal Communications, 8(1):8-15, February 2001.

[MBJ99] D. Maltz, J. Broch, and D. Johnson. Experiences designing and building a multi-hop wireless ad hoc network testbed. Technical Report CMU-CS-99-116, School of Computer Science, Carnegie Mellon Univ. Mar.1999.

[McC87] D. McCullough. Specifications for Multi-Level Security and a Hook-Up Property. In Proc. 1987 IEEE Symposium on Research on Security and Privacy. IEEE Press, May 1987, pp. 161-166.

[McC88] D. McCullough. Noninterference and the composability of security properties. In Proc. IEEE Symposium on Security and Privacy, pages 177–186. IEEE Computer Society Press, May 1988.

[MCF87] J. Millen, S. Clark, and S. Freedman. The Interrogator: Protocol security analysis. IEEE Transactions on Software Engineering, SE-13(2):274-288, February 1987.

[McL90] J. McLean. The specification and modeling of computer security. Computer, 23(1):9-16, January 1990.

[McL92] J. McLean. Proving noninterference and functional correctness using traces. Journal of Computer Security, 1(1):37-57, January 1992.

[McL94] J. McLean. A general theory of composition for trace sets closed under selective interleaving functions. In Proc. IEEE Symposium on Research on Security and Privacy, pp. 79-93, May 1994.
[McL96] J. McLean. A General Theory of Composition for a Class of "Possibilistic" Security Properties. IEEE Transactions on Software Engineering 22 (1) (1996) 53-67.
[McLean90] J. McLean. Security models and information flow. In Proc. 1990 IEEE Symposium on Research in Security and Privacy. IEEE Computer Society Press, 1990.
[McLean94] J. McLean. Security Models, Encyclopedia of Software Engineering. J. Marciniak, ed. John Wiley & Sons, 1994.
[Mea92] C. Meadows. Applying formal methods to the analysis of a key management protocol. Journal of Computer Security, 1(1):5-35, 1992.
[Mea96] C. Meadows. The NRL Protocol Analyzer: An overview. Journal of Logic Programming, Vol. 26, No. 2, (1996) 113-131.
[Mea98] C. A. Meadows. Formal verification of cryptographic protocols: A survey. Technical report, Center for High Assurance Computer Systems, Naval Research Laboratory, 1998.
[MCF87] J. K. Millen, S. C. Clark, and S. B. Freedman. The Interrogator: Protocol Security Analysis. IEEE Transactions on Software Engineering, SE-13(2), 1987.
[MG85] J. McHugh and D. I. Good. An information flow tool for Gypsy. In Proc. IEEE Symp. on Security and Privacy, Apr. 1985, pp. 46–48.
[Mil89] R. Milner. Communication and Concurrency. Prentice-Hall, 1989.
[Mil92] J. Millen. A resource allocation model for denial of service. In Proc. 1992 IEEE Symposium on Research on Security and Privacy, IEEE Computer Society Press, 1992.
[Mil95] J. Millen. The Interrogator Model. In Proc. 1995 IEEE Symposium on Security and Privacy, (1995) 251-260, IEEE Computer Society Press.
[MMS97] J. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using Murϕ. In Proc. 1997 IEEE Symposium on Security and Privacy, pp. 141-151, IEEE Computer Society Press, May 1997.
[MNZZ01] A. C. Myers, N. Nystrom, L. Zheng, and S. Zdancewic. Jif: Java information flow. Software release. http://www.cs.cornell.edu/jif, July 2001.
[Mon99] D. Monniaux: Abstracting cryptographic protocols with tree automata. In Static Analysis Symposium, pp. 149-163, 1999.
[Mos89] L. Moser. A logic of knowledge and belief for reasoning about computer security. In Proc. IEEE CS Computer Security Foundations Workshop, pp. 57-63, 1989.
[MP00] U. Montanari and M. Pistore. π-calculus, structured coalgebras and minimal HD-automata. In Proc. MFCS'00, LNCS 1893. Springer, 2000.
[MP95] U. Montanari and M. Pistore. Checking bisimilarity for finitary π-calculus. In Proc. CONCUR'95, LNCS 962. Springer Verlag, 1995.
[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes (Parts I and II). Information and Computation, 100:1-77, 1992
[MR99] P. J. McCann and G.-C. Roman. Modeling Mobile IP in Mobile UNITY. ACM Trans. on Software Engineering and Methodology, 8(2):115–146, 1999.
[MS01] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In 8th ACM Conference on Computer and Communication Security, pp. 166-175. ACM SIGSAC, November 2001.
[Mye99] A. C. Myers. JFlow: Practical mostly-static information flow control. In Proc. ACM Symp. on Principles of Programming Languages, Jan. 1999, pp. 228–241.
[MZW03] S. Merz, J. Zappe, and M. Wirsing. A spatio-temporal logic for the specification and refinement of mobile systems. In M. Pezz`e, editor, Fundamental Approaches to Software Engineering (FASE 2003), volume 2621 of Lecture Notes in Computer Science, pp. 87-101, Warsaw, Poland, April 2003. Springer-Verlag.

[Nes90] D. M. Nessett. A critique of the Burrows, Abadi and Needham logic. Operating System Review, 24(2):35-38, April 1990.
[Novak] J.D.Novak. The Theory Underlying Concept Maps and How to Construct Them. Cornell University. http://cmap.coginst.uwf.edu/info/
[NS78] R. M. Needham and M. D. Schroeder. Using Encryption for authentication in large networks of computers. Communications of the ACM, 21(12):993–999, December 1978.
[NS87] R.M. Needham and M.D. Schroeder. Authentication revisited. Operating Systems Review, 21:7, January 1987.
[NSNK97] B. Noble, M. Satyanarayanan, G. Ngyuen, and R. Katz. Trace-based mobile network emulation. In Proc. SIGCOMM'97, Cannes, France, Sept.1997.
[NT92] B. Nieh and S. Tavares. Modeling and analyzing cryptographic protocols using Petri nets. In Proc. AUSCRYPT'92, 1992.
[NW02] M. Nygaard and G. Winskel. HOPLA – a higher-order process language. In Proc. CONCUR'02, LNCS 2421.
[NW02a] M. Nygaard and G. Winskel. Linearity in process languages. In Proc. LICS'02.
[NW03] M. Nygaard, G. Winskel. Full Abstraction for HOPLA. CONCUR 2003: 378-392.
[OH90] C. O'Halloran. A Calculus of Information Flow. In Proc. European Symposium on Research on Computer Security, Toulouse, France, 1990.
[Ohe02] D. von Oheimb. Interacting State Machines: a stateful approach to proving security. In Ali Abdallah, Peter Ryan, and Steve Schneider, editors. In Proc. BCS-FACS International Conference on Formal Aspects of Security 2002, volume 2629 of LNCS. Springer-Verlag, 2002.
[OL03] D. von Oheimb and V. Lotz. Generic Interacting State Machines and Their Instantiation with Dynamic Features. ICFEM 2003: 144-166.
[ORRN99] P. Ochsenschlager, J. Repp, R. Rieke, and U. Nitsche. The SH-Verification Tool – Abstraction-Based Verification of Co-operating Systems. Formal Aspects of Computing, The International Journal of Formal Method, 11:1-24, 1999.
[OSM00] S.Osborn, R. Sandhu, Q. Munawer. Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies. ACM Transactions on Information and System Security, vol.3(2) (2000) 85-106.
[OSSTMM] OSSTMM http://www.isecom.org/osstmm/.
[OWASP] OWASP http://www.owasp.org/index
[OXL99] J. Offutt, Y. Xiong, and S. Liu. Criteria for Generating Specification-based Tests. In IEEE Conf. on Engineering of Complex Computer Systems, 1999.
[Pau94] L. C. Paulson. Isabelle: A Generic Theorem Prover. Volume 828 of LNCS. Springer-Verlag, 1994. For an up-to-date description, http://isabelle.in.tum.de/.
[Pau97] L. Paulson. Mechanized proofs for a recursive authentication protocol. In Proc. 10th Computer Security Foundations Workshop, pp. 84-95, IEEE Computer Society Press, June 1997.
[Pau98] L. C. Paulson. The inductive approach to verifying cryptographic protocols. Journal of Computer Security, 6:85-128, 1998.
[Pis99] M. Pistore. History Dependent Automata. PhD. Thesis TD5/99, Universit'a di Pisa, Dipartimento di Informatica, 1999.
[RH00] R. Ramanathan and R. Hain. An ad hoc wireless testbed for scalable, adaptive qos support. In Proc. IEEE Wireless Communications and Networking Conference (WCNC 2000), vol. 3, pp. 998-1002, Chicago, Illinois, Sept. 2000.
[RH93] A. D. Rubin, P. Honeyman. Formal method for the analysis of authentication protocols. Technical report, Bellcore – Center for Info. Technology Integration, 1993.
[Ros95] A. W. Roscoe. Modeling and verifying key-exchange protocols using CSP and FDR. In Proc. 8th IEEE Computer Security Foundations Workshop, pp. 98-107, IEEE Computer Society, 1995.

[Ros98] A. W. Roscoe. The Theory and Practice of Concurrency. Series in Computer Science, Prentice Hall, 1998.

[RS99] P. Ryan and S. Schneider. Process algebra and noninterference. In Proc. 12th IEEE Computer Security Foundations Workshop. IEEE Computer Society Press, 1999.

[RT01] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In 14th IEEE Computer Security Foundations Workshop, pp. 174-190. IEEE Computer Society, 2001.

[Saf00] E.L. Safford. Key applications of Test Automation Framework (TAF). In Proc 12th Annual Software Technology Conference, April 30 - May 5, 2000.

[San92] R. S. Sandhu. The Typed Access Matrix Model. In Proc. IEEE Symposium on Security and Privacy, 1992.

[San96] R. S. Sandhu et. al. Role-based Access Control Models. IEEE Computer, 29(2):38-47 Feb. 1996.

[SBB00] N. Suri, J.M. Bradshaw, M.R. Breedy, P.T. Groth, G.A. Hill, R. Jeffers, and. T.S. Mitrovich. An Overview of the NOMADS Mobile Agent System. 6th ECOOP Workshop on Mobile Object Systems, France, June 2000.

[SBP01] D. Song, S. Berezin, and A. Perrig. Athena: a novel approach to efficient automatic security protocol analysis. Journal of Computer Security, 9(1):47–74, 2001.

[Sch87] F. B. Schneider. Decomposing properties into safety and liveness. Technical Report TR87-874, Cornell University, Computer Science Department, October 1987.

[Sch96] S. Schneider. Security Properties and CSP. In Proc. IEEE Symposium on Security and Privacy, pp. 174-187. IEEE Computer Society Press, May 1996.

[Sch99] F. B. Schneider. Enforceable security policies. Technical Report TR99-1759, Cornell University, Computer Science Department, July 1999.

[Schn00] B. Schneier. Secrets & Lies (Digital Security in a Networked World). John Wiley & Sons, 2000, ISBN 0-471-25311-1.

[SH88] J. Scheid and S. Holtsberg. Ina Jo Specification Language Reference Manual. Systems Development Group, Unisys Corporation, September 1988.

[SI94] B. Steffen, A. Ingolfsdottir. Characteristic Formulae for Processes with Divergence. Information and Computation 110 (1) (1994) 149-163.

[Sid86] D. P. Sidhu. Authentication protocols for computer networks: I. Computer Networks and ISDN Systems, 11:297-310, 1986.

[SM89] Y. Shoham and Y. Moses. Belief as defeasible knowledge. In Proc. 11th International Joint Confernce on Artificial Intelligence, pp. 1168-1173, August 1989.

[SM93] P. Syverson and C. Meadows. A Logical Language for Specifying Cryptographic Protocol Requirements. In Proc. 1993 IEEE Computer Society Symposium on Research in Security and Privacy, pp.165-177. IEEE Computer Society Press, Los Alamitos, California, 1993.

[Smi01] G. Smith. A New Type System for Secure Information Flow. In Proc. 14th IEEE Computer Security Foundations Workshop (CSFW'01), June 11-13, 2001.

[Sne90] E. Snekkenes. Authentication in Open Systems. In Proc. 10th IFIP WG 6.1 Symposium on Protocol Specification, Testing and Verification, pp. 313-324, June 1990.

[Sne91] E. Snekkenes. Exploring the ban approach to protocol analysis. In Proc. 1991 IEEE Computer Society Symposium on Research in Security and Privacy, pp. 171-181, May 1991.

[Sne95] E. Snekkenes. Formal Specification and Analysis of Cryptographic Protocols. PhD thesis, Faculty of Mathematics and Natural Sciences, University of Oslo, Norwegian Defense Research Establishment, P.O. Box 25, N-2007, Kjeller, Norway, January 1995.

[SNS88] J.G. Steiner, B.C. Neuman, and J.I. Schiller. Kerberos: An authentication service for open network systems. In Proc. Usenix Conference, pp. 191-202, Dallas, Texas, February 1988.

[Sto99] S. Stoller. A bound on attacks on authentication protocols. Presented at 1999 Workshop on Formal Methods and Security Protocols, available at http://www.cs.indiana.edu/hyplan/stoller/ , July 1999.
[SU99] R. Sekar and P. Uppuluri. Synthesizing Fast Intrusion Prevention/Detection Systems from High-Level Specifications. USENIX Security Symposium, 1999.
[Sut86] D. Sutherland. A model of information. In Ninth National Computer Security Conference. National Bureau of Standards/National Computer Security Center, 1986.
[SvO94] P. Syverson and P. van Oorschot. On unifying some cryptographic protocol logics. In Proc. IEEE CS Symposium on Research in Security and Privacy, pp. 14-28, 1994.
[SW00] P. Sewell and P.T. Wojciechowski. Nomadic Pict: Language and infrastructure design for mobile agents. IEEE Concurrency, 8(2):42–52, April/June 2000.
[SW02] D. Sangiorgi and D. Walker. The πcalculus: a Theory of Mobile Processes. Cambridge University Press, 2002
[Syv90] P. Syverson. Formal semantics for logics of cryptographic protocols. In Proc. IEEE CS Computer Security Foundations Workshop, pp. 32-41, June 1990.
[Syv91] P. Syverson. The use of logic in the analysis of cryptographic protocols. In Proc. 1991 IEEE Computer Society Symposium on Research in Security and Privacy, pp. 156-170, May 1991.
[Syv92] P. F. Syverson. Knowledge, belief, and semantics in the analysis of cryptographic protocols. Journal of Computer Security, 1:317-334, 1992.
[THG98] F. Thayer Fabrega, J. Herzog, and J. Guttman. Strand spaces: Why is a security protocol correct. In Proc. 1998 IEEE Symposium on Security and Privacy, pp. 160-171. IEEE Computer Society Press, May 1998.
[THG98a] F. J. Thayer Fabrega, J. Herzog, and J. Guttman. Strand space pictures. In Proc. Workshop on Formal Methods and Security Protocols, 1998.
[UMLS03] Object Management Group. Unified Modeling Language Specification, Version 1.5. Specification, OMG, 2003. http://cgi.omg.org/cgi-bin/doc?formal/03-03-01.
[US01] P. Uppuluri and R. Sekar. Experiences with Specification-Based Intrusion Detection. Recent Advances in Intrusion Detection (RAID), 2001.
[Var89] V. Varadharajan. Verification of network security protocols. Computers and Security, 8(8):693-708, 1989.
[Var90] V. Varadharajan. Use of a formal description technique in the specification of authentication protocols. Computer Standards and Interfaces, 9:203-215, 1990.
[VC99] J. Vitek and G. Castagna. Seal: A framework for secure mobile computations. In Proc. Workshop on Internet Programming Languages, volume 1686 of Lecture Notes in Computer Science, pp. 47–77. Springer-Verlag, 1999.
[VM94] B. Victor and F. Moller. The Mobility Workbench – A tool for the π-calculus. In Proc. CAV'94, LNCS 818. Springer Verlag, 1994.
[VM98] J. Voas and G. McGraw. Software Fault Injection: Incoculating Programs Against Errors. John Wiley & Sons, Inc., 1998.
[vOor93] P. van Oorschot. Extending cryptographic logics of belief to key agreement protocols. In Proc. ACM Conference on Computer and Communications Security, pp. 232-243, 1993.
[Wes78] C. H.West. General technique for communications protocol validation. IBM Journal of Research and Development, 22:393-404, 1978.
[WJ02] G. Wimmel, J. Jürjens. Specification-based Test Generation for Security-Critical Systems Using Mutations. In 4th International Conference on Formal Engineering Methods (ICFEM), 2002. © Springer Verlag.
[WJ90] J.T. Wittbold and D. Johnson. Information flow in nondeterministic systems. In Proc. 1990 IEEE Symp. Research in Security and Privacy. IEEE Press, 1990.
[WL92] T. Y. C. Woo and S. S. Lam. Authentication for Distributed Systems. Computer, 25(1):39-52, January 1992.

[WL93] T.Y.C. Woo and S.S. Lam. A semantic model for authentication protocols. In Proc. IEEE Symposium on Security and Privacy, pp. 178-194, 1993.

[WT01] J. Wack and M. Tracey. DRAFT Guideline on Network Security Testing: Recommendations of the National Institute of Standards and Technology. NIST Special Publication 800-42, 2001.

[X.800] Security Architecture for Open Systems Interconnection for CCITT Applications. Recommendations X.800. – CCITT, Geneva, 1991.

[Yas96] A. Yasinsac. A Formal Semantics for Evaluating Cryptographic Protocols. Ph.D. Dissertation, University of Virginia, January 1996.

[YG90] CF. Yu and V. Gligor. A specification and verification method for preventing denial of service. IEEE Transactions on Software Engineering, 16(6):581-592, June 1990.

[YKB93] R. Yahalom, B. Klein, and T. Beth. Trust relationships in secure systems: A distributed authentication perspective. In Proc. IEEE CS Symposium on Research in Security and Privacy, pp. 150-164, 1993.

[YW01] A. Yasinsac and Wm. A. Wulf. A framework for a cryptographic protocol evaluation workbench. The International Journal of Reliability, Quality and Safety Engineering (IJRQSE), 8(4):373-89, 1 December 2001.

[Zap02] J. Zappe. Towards a mobile TLA. In Proc. 7th ESSLLI Student Session, 14th European Summer School in Logic, Language and Information, Trento, Italy, 2002.

[Zap02] Y. Zhang, W. Li. An Integrated Environment for Testing Mobile AdHoc Networks. In Proc. 3rd ACM international symposium on Mobile ad hoc networking & computing, pp. 104-111, 2002.

[ZL97] A. Zakinthinos and E.S. Lee. A General Theory of Security Properties. In Proc. IEEE Symposium on Security and Privacy, pp. 74-102, Oakland, CA, May 4-7 1997.

[Гал03] В. А. Галатенко. Основы информационной безопасности. Курс лекций, Интернет-Университет Информационных технологий, Москва, 2003.

[Гал04] В. А. Галатенко. Стандарты информационной безопасности. Курс лекций, Интернет-Университет Информационных технологий, Москва, 2004.

[ГТКР02] Гостехкомиссия России. Руководящий документ. Безопасность информационных технологий. Критерии оценки безопасности информационных технологий. – Москва, 2002.