III. Защита локальных вычислительных сетей образовательных учреждений от интернет-угроз

1. Введение

1.1. Информационная безопасность и Интернет

Классификация угроз при работе в сети Интернет

Если ранее подавляющее большинство атак сводилось к проникновению на компьютер пользователя той или иной вредоносной программы, сегодня множество угроз – хакерская атака, реклама, фишинг, реализуются удаленно и не нуждаются в дополнительных модулях на компьютере-жертве. Поэтому, при проведении классификации угроз в первую очередь следует разделить угрозы на два больших подмножества:

- угрозы, реализация которых требует внедрения программных компонентов на компьютере-жертве, иначе именуемых *вредоносные программы*:
  - ✓ компьютерные вирусы;
  - ✓ черви;
  - ✓ троянские программы;
  - ✓ потенциально опасное ПО – рекламные утилиты и т.д.;
- угрозы, реализация которых внедрения программных компонентов не требует:
  - ✓ спам;
  - ✓ фишинг;
  - ✓ внешняя, применительно к компьютеру, реклама, к примеру – баннеры и всплывающие окна;
  - ✓ хакинг.

Ущерб от вредоносных программ

Для вредоносных программ характерны следующие формы вредоносных действий:

- *перегрузка каналов связи* – вид ущерба, связанный с тем, что во время масштабных эпидемий по Интернет-каналам передаются огромные количества запросов, зараженных писем или непосредственно копий вредоносного кода;
- *DDoS (Distributed Denial-of-service) атаки*, распределённые атаки «отказ в обслуживании» – вид вредоносного воздействия, выражающийся в одновременном

обращении большого количества, иногда до миллиона и более, инфицированных систем к определенному Интернет-ресурсу, что приводит к полному его блокированию;

- *потеря данных* – поведение вредоносного кода, связанное с намеренным уничтожением определенных данных на компьютере пользователя;
- *нарушение работы ПО* – из-за ошибок в самом вредоносном ПО, зараженные приложения могут работать с ошибками или не работать вовсе;
- *загрузка ресурсов компьютера* – интенсивное использование ресурсов компьютера вредоносными программами ведет к снижению производительности как системы в целом, так и отдельных приложений.

Помимо этого заражение компьютера и включение его в состав ботнета приводит к тому, что в журналах атакуемых систем как источник атаки значится именно этот компьютер, что может привести к судебным искам против владельца зараженного компьютера или блокированию доступа к Интернет-ресурсу с этого компьютера.

Угрозы, не требующие внедрения программных компонентов

Все угрозы этого класса, не считая хакерских, преимущественно используют методы социальной инженерии (социотехники) для достижения результата.

Спам

Спам – это анонимная массовая незапрошенная рассылка.

Данное определение довольно хорошо соотносится с мировой практикой и определениями спама, положенными в основу американского и европейского законодательства о спаме. Кроме того, это определение можно эффективно использовать на практике.

Анонимная рассылка: все страдают в основном именно от автоматических рассылок, со скрытым или фальсифицированным обратным адресом. В настоящее время не существует спамеров, которые не скрывали бы своего адреса и места рассылки.

Массовая рассылка: именно массовые рассылки, и только они, являются настоящим бизнесом для спамеров и настоящей проблемой для пользователей. Небольшая рассылка, сделанная по ошибке человеком, не являющимся профессиональным спамером, может быть нежелательной почтой, но не спамом.

Незапрошеная рассылка: очевидно, что подписные рассылки и конференции не должны попадать в категорию «спама» (хотя условие анонимности и так в значительной мере это гарантирует).

Пять основных тематик покрывают около 50% всего потока спама, как в Рунете, так и в Интернете в целом. Состав тематических «лидеров» практически не менялся за

последние годы и не зависит от региона распространения спама. Лидирующие тематики спама следующие:

- спам «для взрослых»;
- здоровый образ жизни и медикаменты;
- компьютеры и Интернет;
- личные финансы;
- образование.

Очень большая часть спама не преследует рекламных или коммерческих целей. Существуют рассылки политического и агитационного спама, есть также «благотворительные» спамерские письма (призывающие помочь каким-нибудь несчастным). Отдельную категорию составляют мошеннические письма, а также письма, направленные на кражу паролей и номеров кредитных карт. Еще бывают так называемые «цепочечные письма», то есть письма с просьбой переслать их знакомым («страшилки», «письма счастья») и т. п. Есть также вирусные письма, содержащие завлекательный текст и вирусы под видом игрушек, картинок, программ.

Сбор и верификация списков адресов

Для рассылки спама необходимо иметь список адресов электронной почты потенциальных получателей («спам-базу», email database). Сбор адресов осуществляется следующими методами:

- подбор по словарям имен собственных, «красивых слов», частых сочетаний «слово-цифра» (например, «john@», «cool@», «alex-2@»);
- метод аналогий — если существует адрес Masha.Orlova@mail.ru, то вполне резонно поискать Masha.Orlova в почтовых доменах inbox.ru, gmail.ru, yandex.ru и т.п.;
- сканирование всех доступных источников информации — веб-сайтов, форумов, чатов, досок объявлений, Usenet, баз данных Whois на сочетание « слово1@слово2.слово3» (при этом на конце такого сочетания должен быть домен верхнего уровня — com, ru, info и т.д.);
- воровство баз данных сервисов, провайдеров и т.п.;
- воровство персональных данных пользователей при помощи компьютерных вирусов и прочих вредоносных программ.

При сканировании доступных источников информации (способ 3) можно пытаться определить «круг интересов» пользователей данного источника, что дает возможность получить тематические базы данных. В случае воровства данных у провайдеров

достаточно часто имеется дополнительная информация о пользователе, что тоже позволяет провести персонализацию.

Фишинг

Фишинг (англ. phishing) — вид Интернет-мошенничества, цель которого — получить идентификационные данные пользователей. Фишинг – это компьютерное преступление, мошенничество, основанное на принципах социотехники. Злоумышленником создается практически точная копия сайта выбранного Интернет-ресурса: банка, платёжной системы и т.п. Затем, при помощи спам-технологий рассылается письмо, составленное таким образом, чтобы быть максимально похожим на настоящее письмо от выбранной организации. Используются логотипы организации, имена и фамилии реальных руководителей. В таком письме, как правило, сообщается о том, что из-за смены программного обеспечения в системе оказания Интернет-услуг пользователю необходимо подтвердить или изменить свои учетные данные. В качестве причины для изменения данных могут быть названы выход из строя ПО или же нападение хакеров. Во всех случаях цель таких писем одна — заставить пользователя нажать на приведенную ссылку, а затем ввести свои конфиденциальные данные на ложном сайте.

Как выглядят phishing-атаки?

Мошенники со временем становятся все более изощренными, то же самое происходит и с phishing-сообщениями электронной почты и всплывающими окнами. Они часто используют официальные логотипы реальных организаций и другую идентифицирующую информацию, взятую непосредственно с надежного веб-узла.

Ниже приведен пример phishing-сообщения электронной почты (Рис. 1.). Для придания phishing-сообщению еще более надежного вида мошенники могут поместить в него ссылку, на первый взгляд ведущую к надежному веб-узлу (1), хотя на самом деле она ведет на поддельный веб-узел мошенников (2) или вызывает всплывающее окно, которое выглядит так же, как официальный веб-узел. Эти фальшивые узлы называют также подложными веб-узлами. Попав на такой веб-узел, вы можете предоставить мошенникам личную информацию. Полученная информация чаще всего используется для приобретения товаров, получения новой кредитной карточки или хищения личных данных.

В некоторых случаях злоумышленники размещают на подобных сайтах различные эксплойты уязвимостей MS Internet Explorer для побочной установки на компьютер пользователей каких-либо троянских программ. Появление в конце 2003 года экплойта уязвимости с подменой реального URL привело к появлению новой разновидности

фишинга, получившего название «спуфинг» (spoofing). В случае использования данной уязвимости атакуемый пользователь визуально может наблюдать настоящий адрес банковского сайта в адресной строке браузера, но находиться сам при этом будет на сайте поддельном. Успеху фишинг-афёр способствует низкий уровень осведомленности пользователей о правилах работы компаний, от имени которых действуют преступники. В частности, почти половина пользователей не знают простого факта: банки не рассылают писем с просьбой подтвердить в онлайне номер своей кредитной карты и её PIN-код. Для защиты от фишинга производители основных Интернет-браузеров договорились о применении одинаковых способов информирования пользователей о том, что они открыли подозрительный сайт, который может принадлежать мошенникам. Новые версии браузеров обладают такой возможностью.

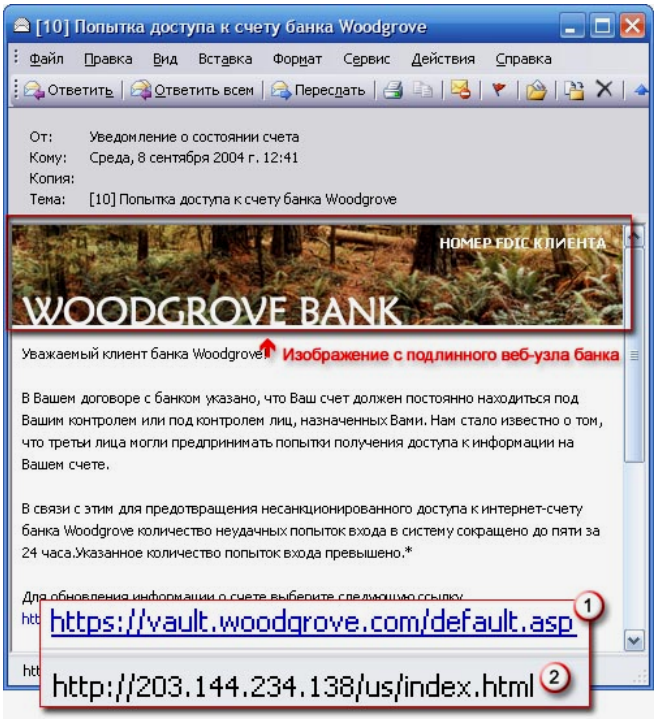

Рис. 1. Пример phishing-сообщения электронной почты со ссылкой на мошеннический веб-узел

Реклама

Реклама сегодня присутствует везде и постоянно – на улице, на телевидении, в журналах и газетах и, разумеется, в Интернет. Рекламу в Интернет можно разделить на внешнюю и внутреннюю применительно к компьютеру. Внутренней в этом случае будет самореклама программ, установленных на компьютере, реклама, демонстрируемая в качестве дополнения к основному функционалу загруженных из Интернет бесплатных программ, а также разного рода скрытые и полускрытые утилиты для демонстрации рекламы, установленные на компьютере. Последние три типа программ относятся к

категории adware и причислены к потенциально опасному ПО, так как, не нанося явного вреда системе, они, тем не менее, загружают канал передачи данных и отнимают время у пользователя.

Внешней, применительно к целевому компьютеру, рекламой, обычно считаются всплывающие окна на сайтах, иногда без возможности закрытия и многочисленные банеры, показываемые в процессе интерактивной работы с Интернет, а также спамовые рекламные рассылки, уже рассмотренные ранее. В отличие от платного телевидения, покупка чистого от рекламы Интернета в настоящий момент не представляется возможной, а количество рекламы в Интернет постоянно возрастает, что автоматически поднимает вопрос о необходимости уменьшения этого количества. С рекламой, приходящей по электронной почте, очевидно, должны бороться антиспамовые фильтры, тогда как для уменьшения количества всплывающих окон и демонстрируемых баннеров нужны другие специальные средства.

Хакерские атаки

Под хакерскими атаками подразумевается два вида атак – с автоматическим внедрением вредоносных программ стандартными способами и реализуемые вручную, когда целью злоумышленника является взлом отдельной системы. Наилучшее средство борьбы с атаками второго рода – разработка и реализация правил безопасности, четко описывающих допустимые и недопустимые информационные потоки в сети. Для практической реализации политики безопасности в случае хакерских атак, как правило, используются пакетные фильтры, межсетевые экраны и системы обнаружения вторжений (Intrusion Detection System, IDS).

1.2. Понятие многоуровневой системы защиты сети образовательного учреждения

Современные информационные системы имеют сложную структуру. Они содержат пользовательские приложения, работающие во взаимодействии с различными операционными системами, установленными на компьютерах, объединенных в локальную сеть, часто связанную тем или иным образом с сегментом глобальной сети. Обеспечение безопасности такой системы требует проведения целого комплекса мероприятий в соответствии с разработанной на предприятии политикой информационной безопасности.

Существует два возможных направления политики информационной безопасности. В первом случае (ограничительная политика), пользователь имеет право использовать любые ресурсы, кроме тех, доступ к которым ограничен или закрыт. Во втором случае

(нормативная политика), пользователь имеет право использовать только те ресурсы, которые ему явным образом выделены.

Первая схема политики безопасности применяется, как правило, на предприятиях с большим набором функционально различных групп достаточно квалифицированных пользователей. Вторая схема применима там, где производится много действий с одним и тем же набором приложений, причем круг этих приложений может быть очерчен заранее, а любое нестандартное действие рассматривается как попытка нарушения режима информационной безопасности.

В соответствии с принятой терминологией, информационная безопасность обеспечена в случае, если для любых информационных ресурсов в системе поддерживается определенный уровень конфиденциальности (невозможности несанкционированного получения какой-либо информации), целостности (невозможности несанкционированной либо случайной ее модификации) и доступности (возможности за разумное время получить требуемую информацию). При этом должна учитываться не только вероятность нарушения какого-либо из аспектов безопасности в результате умышленных либо неумышленных действий пользователей, но и вероятность выхода из стоя каких-либо узлов информационной системы. В этом смысле средства повышения надежности также входят в комплекс информационной безопасности предприятия.

Многоуровневая защита информационной системы

Невозможно анализировать безопасность информационной системы без рассмотрения ее структуры, хотя бы в самом общем виде. Подобная структура достаточно хорошо описывается четырехуровневой моделью.

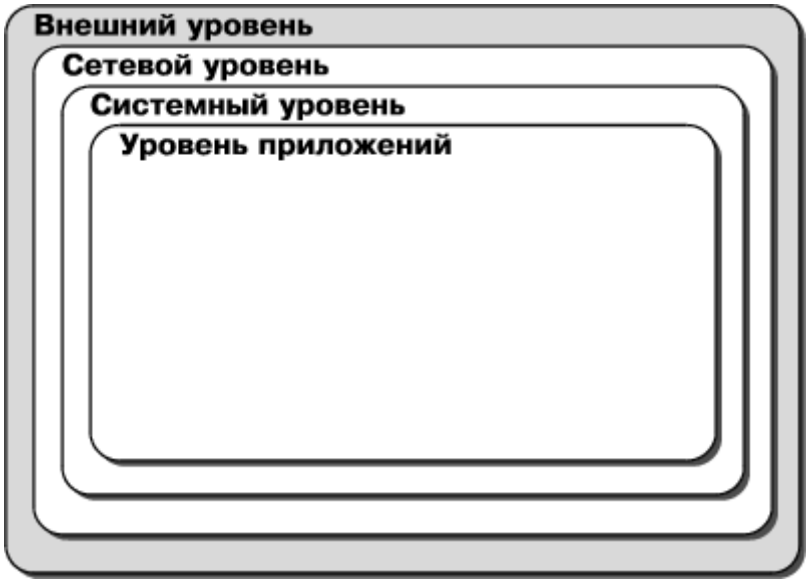

Рис. 2. Четырехуровневая модель информационной системы

Ниже приведена краткая характеристика каждого из уровней:

1. Внешний уровень определяет взаимодействие информационной системы организации с глобальными ресурсами и системами других организаций. Функционально этот уровень характеризуется, с одной стороны, информационными (главным образом, сетевыми) сервисами, предоставляемыми данной организацией, с другой стороны, аналогичными сервисами, запрашиваемыми из глобальной сети. На этом уровне должны отсекаться, как попытки внешних пользователей несанкционированно получить доступ к внутренним сервисам, так и попытки собственных пользователей осуществить подобные операции по отношению к внешним сервисам или несанкционированно переслать информацию в глобальную сеть.

2. Сетевой уровень связан с доступом к информационным ресурсам внутри локальной сети организации. Безопасность информации на этом уровне обеспечивается средствами проверки подлинности пользователей и разграничением доступа к ресурсам локальной сети (аутентификация и авторизация).

3. Системный уровень связан, прежде всего, с управлением доступом к ресурсам ОС. Именно на этом уровне происходит непосредственное взаимодействие с пользователями, запускаются приложения, и, самое главное, определяются «правила игры» между информационной системой и пользователем (задается либо изменяется конфигурация системы). В этой связи, естественно понимать защиту информации на данном уровне, как четкое разделение, к каким ресурсам ОС, какой пользователь и когда может быть допущен. Важность именно этого уровня можно проиллюстрировать тем фактом, что долгое время вопросы информационной безопасности сводились исключительно к рассмотрению защиты на уровне и средствами ОС.

4. Уровень приложений связан с использованием прикладных ресурсов информационной системы. Поскольку именно приложения на содержательном уровне работают с пользовательскими данными, для них, вообще говоря, нужны собственные механизмы обеспечения информационной безопасности.

Многоуровневая защита с точки зрения сетевой архитектуры

Концепция многоуровневой системы безопасности состоит в наращивании числа «линий обороны» с целью повышения степени защищенности охраняемого объекта. Такая система защиты многократно увеличивает затраты, необходимые для проведения атаки. Она воздвигает множество препятствий на пути злоумышленника. Эшелонированная оборона предотвращает как прямые атаки на критически важные системы, так и попытки «прощупывания» вашей сети. Кроме того, такое построение системы защиты

предполагает и реверсный эффект – предотвращение несанкционированных действий собственных пользователей по отношению к внешним ресурсам. С точки зрения сети образовательного учреждения такой подход представляется наиболее актуальным.

При организации защиты от Интернет-угроз важным является понятие *периметра* – укреплённой границы сети. Периметр может состоять из различных подсистем, как представленных различными аппаратными и программными средствами, так и объединёнными в единый программно-аппаратный комплекс. Такими подсистемами обычно являются:

- маршрутизаторы (routers);
- межсетевые экраны (брандмауэры, firewalls);
- прокси-серверы;
- системы обнаружения вторжений (IDS);
- средства создания виртуальных частных сетей (VPN);
- антивирусные средства;
- экранированные подсети.

Пограничный маршрутизатор

Маршрутизаторы (routers) осуществляют управление трафиком, поступающим в сеть, выходящим из сети или трафиком внутри самой сети. Пограничный маршрутизатор (border router) является последним маршрутизатором непосредственно перед выходом во внешнюю сеть. В силу того, что весь Интернет-трафик организации проходит через этот маршрутизатор, последний часто функционирует в роли первой и последней линии защиты сети, обеспечивая фильтрацию входящего и исходящего трафика.

Межсетевой экран

Межсетевой экран — комплекс аппаратных или программных средств, осуществляющий контроль и фильтрацию проходящих через него сетевых пакетов на различных уровнях модели OSI в соответствии с заданными правилами.

Основной задачей межсетевого экрана является защита компьютерных сетей или отдельных узлов от несанкционированного доступа. Также межсетевые экраны часто называют фильтрами, так как их основная задача — не пропускать (фильтровать) пакеты, не подходящие под критерии, определённые в конфигурации.

Некоторые межсетевые экраны также позволяют осуществлять трансляцию адресов — динамическую замену адресов назначения или источника – Network Address Translation (NAT).

Другие названия

Брандма́уэр (нем. *Brandmauer*) — заимствованный из немецкого языка термин, являющийся аналогом английского *firewall* в его оригинальном значении (стена, которая разделяет смежные здания, предохраняя от распространения пожара). Интересно, что в области компьютерных технологий в немецком языке употребляется слово «*firewall*».

Файрво́лл, файрво́л, файерво́л, фаерво́л — образовано транслитерацией английского термина *firewall*, эквивалентного термину межсетевой экран, в настоящее время не является официальным заимствованным словом в русском языке.

Разновидности межсетевых экранов

Межсетевые экраны подразделяются на различные типы в зависимости от следующих характеристик:

- обеспечивает ли экран соединение между одним узлом и сетью или между двумя или более различными сетями;
- происходит ли контроль потока данных на сетевом уровне или более высоких уровнях модели OSI;
- отслеживаются ли состояния активных соединений или нет.

В зависимости от охвата контролируемых потоков данных межсетевые экраны делятся на:

- *традиционный межсетевой экран* — программа (или неотъемлемая часть операционной системы) на шлюзе (сервере передающем трафик между сетями) или аппаратное решение, контролирующие входящие и исходящие потоки данных между подключенными сетями;
- *персональный межсетевой экран* — программа, установленная на пользовательском компьютере и предназначенная для защиты от несанкционированного доступа только этого компьютера.

Вырожденный случай — использование традиционного межсетевого экрана сервером, для ограничения доступа к собственным ресурсам.

В зависимости от уровня модели OSI, на котором происходит контроль доступа, существует следующее разделение межсетевых экранов:

- функционирующие на *сетевом* уровне, когда фильтрация происходит на основе адресов отправителя и получателя пакетов, номеров портов транспортного уровня модели OSI и статических правил, заданных администратором;
- функционирующие на *сеансовом* уровне (также известные как *stateful*) — отслеживающие сеансы между приложениями, не пропускающие пакеты нарушающих спецификации TCP/IP, часто используемых в злонамеренных операциях —

сканировании ресурсов, взломах через неправильные реализации TCP/IP, обрыв/замедление соединений, инъекция данных;

- функционирующие *на уровне приложений*, фильтрация на основании анализа данных приложения, передаваемых внутри пакета: такие типы экранов позволяют блокировать передачу нежелательной и потенциально опасной информации, на основании политик и настроек.

Некоторые решения, относимые к межсетевым экранам уровня приложения, представляют собой прокси-серверы с некоторыми возможностями межсетевого экрана, реализуя прозрачные прокси-серверы, со специализацией по протоколам. Возможности прокси-сервера и многопротокольная специализация делают фильтрацию значительно более гибкой, чем на классических межсетевых экранах, но такие приложения имеют все недостатки прокси-серверов (например, анонимизация трафика).

В зависимости от возможности отслеживания активных соединений межсетевые экраны подразделяются на:

- *stateless* (простая фильтрация), которые не отслеживают текущие соединения (например, TCP), а фильтруют поток данных исключительно на основе статических правил;
- *stateful*, *stateful packet inspection* (SPI) (фильтрация с учётом контекста), с отслеживанием текущих соединений и пропуском только таких пакетов, которые удовлетворяют логике и алгоритмам работы соответствующих протоколов и приложений: такие типы межсетевых экранов позволяют эффективнее бороться с различными видами DoS-атак и уязвимостями некоторых сетевых протоколов, кроме того, они обеспечивают функционирование таких протоколов, как H.323, SIP, FTP и т.п., которые используют сложные схемы передачи данных между адресатами, плохо поддающиеся описанию статическими правилами, и, зачастую, несовместимых со стандартными, stateless межсетевыми экранами.

Прокси-серверы

Прокси-сервер (от англ. *proxy* — «представитель, уполномоченный») — 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 или веб-сайт), 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.

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;
- 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.

Многие прокси-серверы используются для нескольких целей одновременно. Некоторые прокси-серверы ограничивают возможность использования только нескольких наиболее популярных протоколов: http - *tcp/80*, https – *tcp/443* (шифрованное соединение *http*), ftp – *tcp/20* и *tcp/21*.

В отличие от шлюза, прокси-сервер чаще всего не пропускает ICMP-трафик (невозможно проверить доступность машины командами *ping* и *tracert*).

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

Прозрачный прокси — это такой прокси-сервер, который принимает трафик от клиентов сети через маршрутизатор, способный различать какой трафик должен быть

направлен через прокси-сервер. То есть клиенту не нужно проводить конфигурацию ПО для работы через прокси-сервер, маршрутизатор сам направит трафик клиента на прокси-сервер.

Системы обнаружения вторжений

IDS (Intrusion Detection System) – система обнаружения вторжений выполняет роль охранной сигнализации сети и применяется для обнаружения и извещения обо всех сетевых пакетах, являющихся частью вредоносного или потенциально опасного трафика. Система, как правило, содержит множество детекторов различного типа, размещенных в стратегических точках сети. Детекторы IDS ищут заданные сигнатуры нежелательных событий и могут выполнять заранее предопределённые действия, подобно тому, как антивирус определяет наличие вируса в файле.

Виртуальные частные сети

VPN (Virtual Private Network) – виртуальная частная сеть представляет собой защищенный сеанс, для организации которого используются незащищенные каналы, например, Интернет.

Антивирусные средства

Антивирусные средства предназначены для борьбы с вредоносными программами и иными угрозами и, как правило, интегрируются с другими подсистемами сетевого периметра.

Экранированная подсеть

Представляет собой изолированную сеть, соединенную с определенным интерфейсом межсетевого экрана или другого фильтрующего трафик устройства. Сеть образовательного учреждения, защищённая с помощью продуктов из комплекта поставки СПО, может рассматриваться как экранированная подсеть.

Организация многоуровневой защиты сети образовательного учреждения с использованием комплекта ПСПО

В комплекте ПСПО поставляется несколько продуктов, предназначенных для построения защищённого периметра сети образовательного учреждения:

- межсетевой экран netfilter/iptables;
- модульный конфигуратор Alterator, используемый, в том числе, и для управления работой межсетевого экрана netfilter/iptables;
- антивирус ClamAV;
- графический интерфейс настройки и управления антивирусом KlamAV;
- кэширующий прокси-сервер Squid;

- система контентной фильтрации (СКФ).

Общая схема совместной работы указанных компонентов представлена на диаграмме:

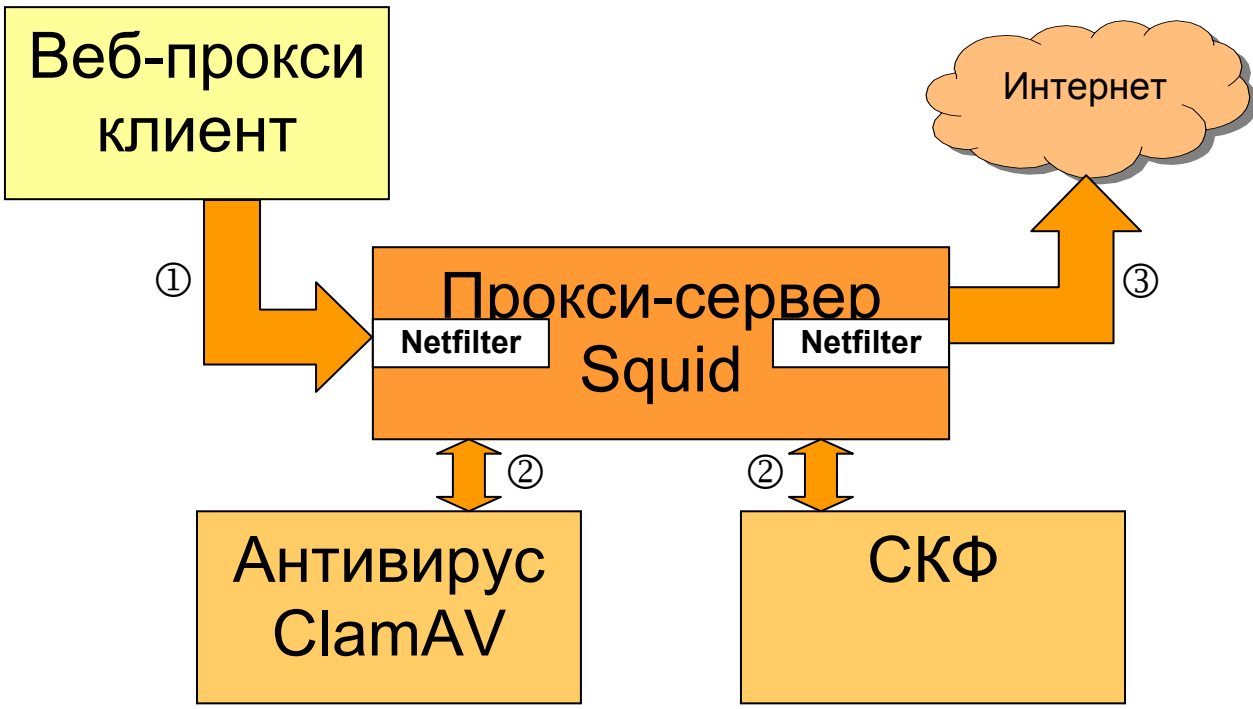

Рис. 3. Схема обработки исходящих запросов из сети образовательного учреждения

Порядок взаимодействия компонентов в вышеприведённой схеме следующий:

1. Ученик обращается с компьютера, сконфигурированного как клиент прокси-сервера Squid к Интернет-ресурсу.

2. Межсетевой экран Netfilter принимает запрос и, опираясь на список своих правил, сформированных для внутреннего интерфейса, принимает решение о разрешении или запрете дальнейшей обработки запроса. При положительном решении запрос попадает к прокси-серверу Squid.

3. Прокси-сервер Squid, опираясь на собственный список правил, также принимает решение о возможности дальнейшей обработки запроса. В качестве критериев принятия решения выступают также ответы от антивируса и СКФ, к которым прокси-сервер Squid обращается в процессе обработки запроса. При положительном решении запрос перенаправляется на внешний интерфейс прокси-сервера.

4. Межсетевой экран Netfilter обрабатывает запрос, опираясь на список своих правил, сформированных для внешнего интерфейса. При положительном решении запрос передаётся в Интернет.

5. Обработка ответа происходит в обратном порядке. При этом, если антивирус обнаружит в ответе вредоносный код, он передаст отрицательный ответ прокси-серверу, и компьютер ученика не получит несущую угрозу информацию с Интернет-ресурса.

2. Программное обеспечение для защиты от вирусов и всех других типов вредоносных программ, а также от хакерских атак и спама (KlamAV+ClamAV, alterator-firewall)

В предыдущем разделе была приведена схема совместного использования продуктов из комплекта ПСПО. Для её практического использования требуется установить и настроить каждый из компонентов.

2.1. Антивирус ClamAV

Clam AntiVirus — это антивирусный набор с открытым исходным кодом (GPL) для UNIX, предназначенный, прежде всего, для сканирования электронной почты на почтовых шлюзах. Он предоставляет некоторое количество утилит, включая гибкий и масштабируемый многопоточный демон, сканер командной строки и продвинутый инструмент для автоматических обновлений баз данных. Ядром набора является антивирусный механизм, доступный в форме разделяемой библиотеки.

Общие сведения

Вот список основных возможностей:

- сканер командной строки;
- быстрый, многопоточный демон с поддержкой сканирования при доступе;
- milter-интерфейс для sendmail;
- модуль обновления баз сигнатур угроз с возможностью получения частичных обновлений и использованием цифровых подписей файлов обновлений;
- C-библиотека вирусного сканера;
- сканирование при доступе (Linux® и FreeBSD®);
- вирусная база данных, обновляемая несколько раз в день (смотрите домашнюю страницу относительно общего числа сигнатур);
- встроенная поддержка различных архивных форматов, включая Zip, RAR, Tar, Gzip, Bzip2, OLE2, Cabinet, CHM, BinHex, SIS и другие;
- встроенная поддержка почти всех форматов почтовых файлов;

- встроенная поддержка выполняемых файлов ELF и Portable Executable, сжатых UPX, FSG, Petite, NsPack, wwpack32, MEW, Upack и замаскированных SUE, Y0da Cryptor и другими;
- встроенная поддержка общераспространённых форматов документов, включая файлы MS Office и MacOffice, HTML, RTF и PDF.

Принцип функционирования антивируса схематично изображён на следующем рисунке:

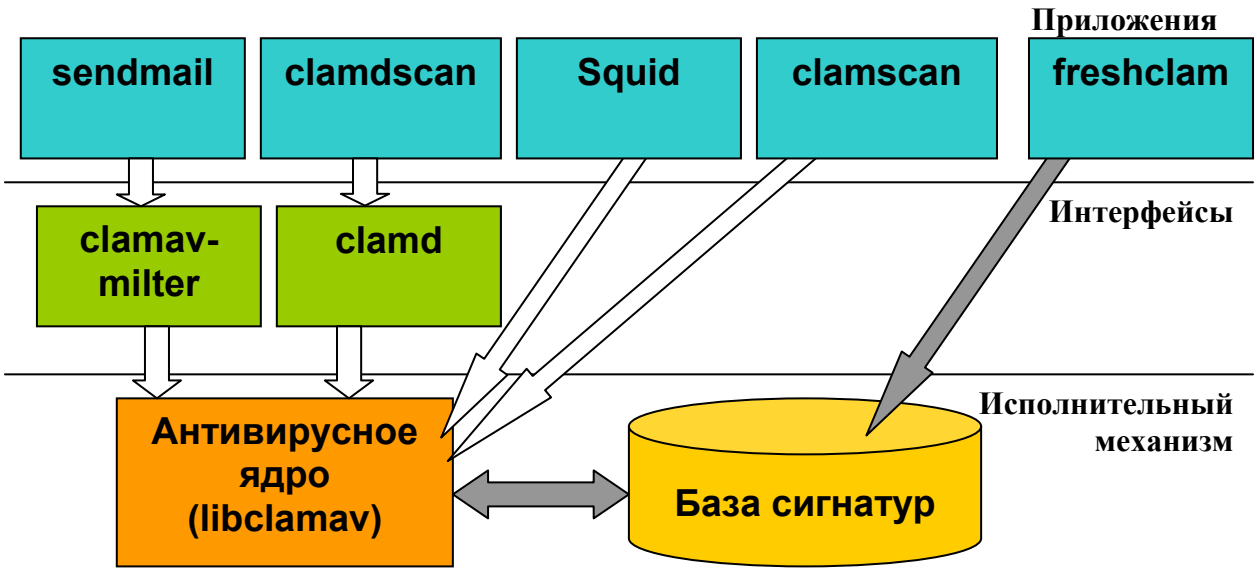

Рис. 4. Принципиальная схема функционирования антивируса

Процесс поддержания баз сигнатур угроз в актуальном состоянии

База сигнатур угроз поддерживается в актуальном состоянии, не в последнюю очередь, благодаря помощи сообщества СПО. Группа создателей сигнатур старается не отставать от создателей и распространителей новых угроз, выпуская обновление базы сигнатур меньше, чем через час после начала распространения нового вредоносного кода. При обнаружении нового вируса, который не смог обнаружить ClamAV, необходимо заполнить форму на сайте http://www.clamav.net/sendvirus. Группа создателей сигнатур проверит полученную информацию и при необходимости обновит базу сигнатур.

2.1.1. Установка ClamAV

Устанавливать приложения из комплекта ПСПО лучше всего с использованием менеджера пакетов Synaptic, т.к. это гарантирует совместимость всех программным модулей комплекта друг с другом. Если вы планируете произвести установку из rpm-пакетов, то обратитесь к документации разработчика: **http://www.clamav.net/doc/latest/clamdoc.pdf**

Для установки из репозитария необходимо запустить менеджер пакетов, выбрать пакет *clamav* и отметить его для установки. После чего применить изменения, и антивирус будет установлен. Если антивирус уже установлен, то этот шаг можно пропустить.

По окончании установки в системе будут присутствовать следующие исполняемые файлы:

- **/usr/sbin/clamd** – антивирусный демон, прослушивающий подключения к UNIX или TCP-сокетам и сканирующий каталоги по требованию;
- **/usr/bin/clamscan** – утилита командной строки, предназначенная для проверки файлов и каталогов на предмет наличия вирусов;
- **/usr/bin/clamdscan** – простой интерфейс к демону clamd, позволяет также сканировать файлы и каталоги, при этом используются те же параметры, что и в clamscan;
- **/usr/bin/freshclam** – утилита автоматического обновления вирусной базы данных через Интернет, позволяющая держать ее в актуальном состоянии;
- **/usr/bin/sigtool** – генерирует вирусную сигнатуру, используя внешний антивирусный сканер, который способен обнаружить вирус. Может создавать шестнадцатеричный дамп и формировать и распаковывать CVD-базу данных (ClamAV Virus Database).

2.1.2. Настройка ClamAV

Для успешного функционирования исполняемых модулей, входящих в состав антивируса *ClamAV*, потребуется создать в системе учетные записи пользователя и группы с именем **clamav**. Именно эти учётные записи используются для доступа к директориям и файлам со служебной информацией. Их можно создать, например, с помощью следующей последовательности команд:

```
# groupadd clamav
# useradd -g clamav -s /bin/false -c "Clam AntiVirus" clamav
```

Необходимо заблокировать несанкционированный доступ к вновь созданным учетным записям.

Если антивирус был установлен из репозитария, то он уже настроен на использования учётной записи пользователя mail, входящего в состав группы mail.

Антивирус ClamAV использует для настройки параметров функционирования своих компонентов два конфигурационных файла: **clamd.conf** для демона **clamd** и **freshclam.conf** для утилиты обновления баз сигнатур freshclam. Если установка производилась средствами менеджера пакетов Synaptic, то эти файлы будут расположены в директории /etc/clamav, и их редактирование без необходимости не требуется.

При установке из rpm-пакетов с параметрами по умолчанию конфигурационные файлы располагаются в директории **/usr/local/etc** и требуют обязательного первоначального редактирования. Редактирование заключается в комментировании строки со словом **Example**. Прочие параметры настраиваются по необходимости.

Файл **clamd.conf** содержит следующие основные настройки:

1. **User** – имя учётной записи, используемой для работы демоном clamd, например:

```
User clamav
```

2. **LogFile** – полный путь к файлу журнала демона **clamd**, например:

```
LogFile /var/log/clamav/clamd.log
```

Для правильного и корректного функционирования демона clamd необходимо вручную создать указанный файл и сделать его владельцем пользователя, указанного в параметре User.

3. **DatabaseDirectory** - полный путь к директории, в которой хранятся файлы базы сигнатур, например:

```
DatabaseDirectory /var/lib/clamav
```

Необходимо, чтобы пользователь, указанный в параметре User, имел доступ на чтение к указанной директории.

4. **TemporaryDirectory** - полный путь к директории, в которую демон clamd будет сохранять временные файлы, например:

```
TemporaryDirectory /var/tmp
```

Необходимо, чтобы пользователь, указанный в параметре User, имел доступ на чтение и запись к указанной директории.

5. **LocalSocket** - полный путь к файлу сокета, который демон clamd будет использовать для взаимодействия с другими программами, например:

```
LocalSocket /var/lib/clamav/clamd.socket
```

Для правильного и корректного функционирования демона clamd необходимо вручную создать указанный файл и сделать его владельцем пользователя, указанного в параметре User.

6. **LogFileMaxSize** – максимальный размер файла журнала, значение «0» означает отсутствие ограничений. Например:

```
LogFileMaxSize 1M
```

7. **TCPAddr** – IP-адрес сетевого интерфейса компьютера, по которому демон принимает запросы на проверку файлов от внешних приложений, например:

```
TCPAddr 192.168.10.1
```

По умолчанию запросы принимаются через любой интерфейс.

8. **TCPSocket** – порт протокола TCP, по которому демон принимает запросы на проверку файлов от внешних приложений, например:

```
TCPSocket 3310
```

9. **AlgorithmicDetection** – указание использовать эвристический алгоритм для обнаружения вредоносного кода, например:

```
AlgorithmicDetection yes
```

10. Указания проверять различные типы файлов, используя собственные алгоритмы анализа структуры соответствующих типов файлов. Например:

```
ScanPE yes
ScanELF yes
ScanOLE2 yes
ScanPDF yes
```

11. Указания проверять присутствие атаки *фишинг* в сообщениях электронной почты и способы реакции на такое событие:

```
PhishingSignatures yes
PhishingScanURLs yes
PhishingRestrictedScan yes
PhishingAlwaysBlockSSLMismatch no
PhishingAlwaysBlockCloak no
```

12. **ScanHTML** – указание нормализовать HTML-заголовки с целью выявления злонамеренно модифицированных заголовков, например:

```
ScanHTML yes
```

13. **ScanArchive** – указание проверять файлы архивных форматов, например:

```
ScanArchive yes
```

С прочими настройками можно ознакомиться в конфигурационном файле, созданном программой установки антивируса. В нём содержится полный список всех параметров и достаточно подробное описание синтаксиса и назначения каждого из них. Неиспользуемые параметры закомментированы.

2.1.3. Настройка получения обновлений антивирусных баз

Обновление антивирусных баз может осуществляться в двух режимах: *интерактивном* – путём однократного запуска утилиты freshclam командной строки, и в *автоматическом* – путём запуска утилиты freshclam как демона. Утилита freshclam поддерживает возможность частичного получения обновлений антивирусных баз, что позволяет не передавать полный CVD-файл при каждом обновлении, а только изменения между текущим состоянием антивирусных баз на сервере и на обновляемом компьютере.

Для настройки утилиты обновления баз сигнатур **freshclam** используется конфигурационный файл **freshclam.conf**. Он располагается в той же директории, что и файл **clamd.conf**, и принципы его редактирования те же.

Файл **freshclam.conf** содержит следующие основные настройки:

1. **DatabaseDirectory -** полный путь к директории, в которой хранятся файлы базы сигнатур, например:

```
DatabaseDirectory /var/lib/clamav
```

Путь к этой директории должен быть тем же самым, что и в файле clamd.conf.

2. **DatabaseOwner** – имя учётной записи, используемой для записи файлов в директорию, указанную в параметре **DatabaseDirectory**, например:

```
DatabaseOwner clamav
```

Этой учётной записи необходимо предоставить права чтения и записи в директорию, указанную в параметре **DatabaseDirectory**. Лучше использовать ту же учётную запись, что и в настройках демона clamd.

3. **UpdateLogFile** – полный путь к файлу журнала утилиты freshclam, например:

```
UpdateLogFile /var/log/clamav/freshclam.log
```

Для правильного и корректного функционирования утилиты freshclam необходимо вручную создать указанный файл и сделать его владельцем пользователя, указанного в параметре DatabaseOwner.

С прочими настройками можно ознакомиться в конфигурационном файле, созданном программой установки антивируса. В нём содержится полный список всех параметров и достаточно подробное описание синтаксиса и назначения каждого из них. Неиспользуемые параметры закомментированы.

Проверка правильности функционирования

1. Для проверки правильности функционирования утилиты freshclam необходимо в консольном окне выполнить команду:

```
# freshclam
```

Если всё настроено правильно, то утилита подключится к серверу обновлений и скопирует последние файлы сигнатур угроз.

2. Для запуска утилиты freshclam в режиме демона необходимо выполнить команду:

```
# freshclam -d
```

Проверить, что утилита успешно запустилась можно командой:

```
# ps fealxw | grep freshclam
```

3. Для проверки возможности обнаружения вирусов с помощью утилиты сканирования нужно загрузить с **сайта http://eicar.org** тестовые образцы «вируса» **Eicar** в директорию **/tmp**. Затем выполнить команду:

```
# clamscan -r -l scan.txt /tmp
```

Антивирус должен найти тестовые файлы и сохранить результаты сканирования в файл scan.txt.

4. Для проверки работоспособности демона clamd необходимо запустить его и использовать команду **clamdscan**:

```
# clamdscan -l scan.txt /tmp
```

Обратите внимание, что сканируемые файлы должны быть доступны учетной записи демона clamd, иначе будет сгенерировано сообщение об ошибке при попытке доступа.

Антивирусная проверка файлов при обращении

Для организации расширенной обработки событий обращения к файлам со стороны процессов, исполняющихся в среде ОС Linux, был разработан дополнительный модуль **Dazuko**, который доступен для скачивания на сайте **http://dazuko.org/.** По сути, он выполняет функцию файлового монитора, перехватывающего обращения к файлам и передающего их на обработку сторонним программам, которые зарегистрированы у него как дополнительные обработчики данного события. Такие обработчики проверяют параметры запроса по своим правилам и возвращают своё решение о допустимости или недопустимости разрешить данное обращение.

Текущая версия демона clamd позволяет встроиться в цепочку обработчиков события доступа к файлам в списке модуля Dazuko, тем самым реализуя совместно с ним функционал антивирусного монитора. Модуль Dazuko является экспериментальным и не обязательным для запуска и успешного функционирования демона clamd.

Для взаимодействия с модулем Dazuko в демоне clamd используется специальный интерфейс – **Clamuko**, реализованный в форме отдельного потока. Этот поток принимает от модуля Dazuko сигнал о попытке получить доступ к файлу и передаёт информацию о файле основному потоку демона, который с помощью антивирусных баз проверяет файл на наличие вирусов. Если вирус не был обнаружен, то через интерфейс Clamuko передаётся положительное решение о доступе к файлу, в противном случае - отрицательное.

Необходимо соблюдать следующие важные правила при интеграции демона clamd с модулем Dazuko:

- демон требуется останавливать корректно, т.е. используя команду **SHUTDOWN** или сигнал **SIGTERM**: в противном случае может произойти потеря доступа к файлам, которые проверялись в момент некорректного завершения работы демона, до перезагрузки системы;
- не включать в список защищаемых демоном директорию, используемую почтовым сканером для распаковки вложений: доступ ко всем зараженным файлам будет автоматически заблокирован и сканер (включая *clamd*!) не сможет определить наличие вируса во вложении, как результат заражённое письмо может быть доставлено получателю.

В качестве примера, для защиты полностью всей системы в файл *clamd.conf* должны быть добавлены следующие строки:

```
ClamukoScanOnAccess
ClamukoIncludePath /
ClamukoExcludePath /proc
ClamukoExcludePath /temporary/dir/of/your/mail/scanning/software
```

2.2. Графическая оболочка KlamAV

KlamAV — программа KDE для обнаружения вирусов, которая позволяет осуществлять сканировать файлы и директории на наличие вирусов, в том числе и по расписанию, получать сведения о вирусах, обновлять антивирусную базу данных и антивирусное программное обеспечение. Другими словами, это графический интерфейс для взаимодействия с антивирусом ClamAV. Данный продукт, как и ClamAV распространяется по лицензии GNU GPL.

Установку приложения KlamAV, как и антивируса ClamAV, необходимо осуществлять с использованием менеджера пакетов Synaptic. Название пакета: *klamav*.

Перед началом работы с KlamAV необходимо указать два основных параметра: где будут располагаться карантин и антивирусная база данных (Рис. 5):

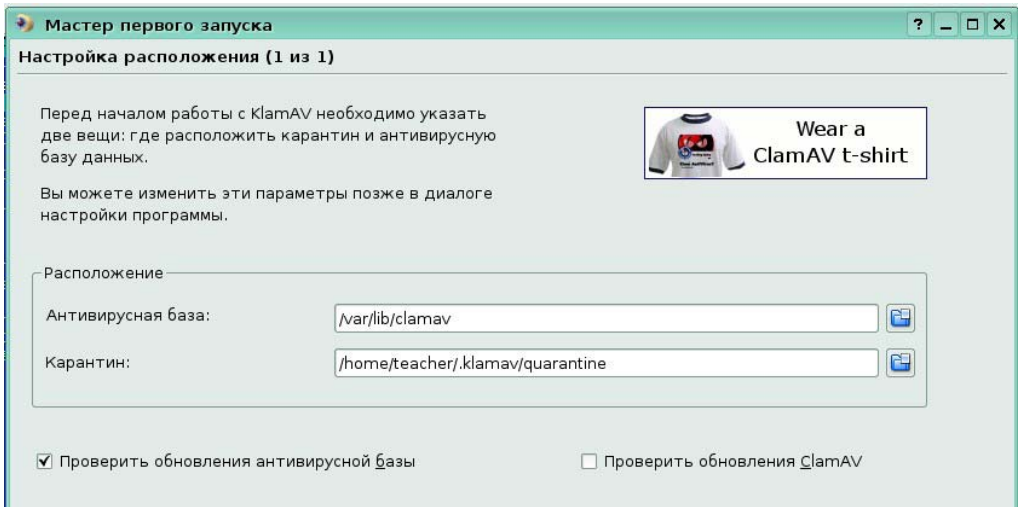

Рис. 5. Страница Мастера первоначальной настройки KlamAV

Интерфейс Klamav состоит из ряда страниц-вкладок, каждая из которых имеет свое предназначение. Ниже приведен внешний вид и указано предназначение каждой из вкладок.

На первой вкладке – **«Проверка»** (Рис.6.) настраиваются задачи проверки различных областей системы по требованию. Настраиваемые здесь действия аналогичны действию утилиты clamscan.

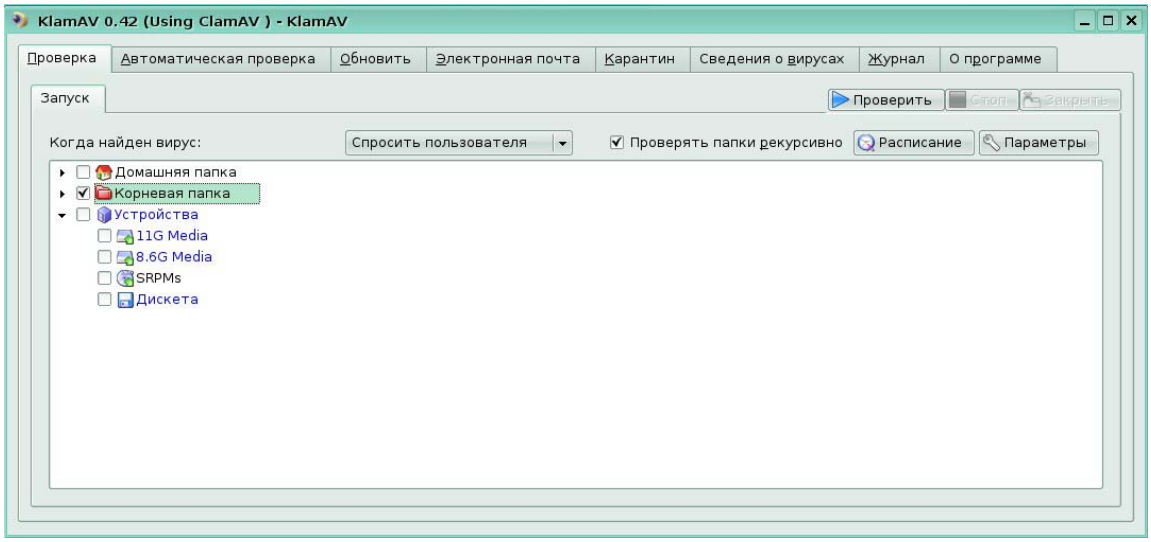

Рис. 6. Закладка настройки задач проверки по требованию (антивирусного сканера)

При необходимости регулярного выполнения настраиваемых действий можно создать расписание запуска созданных задач (Рис 7.).

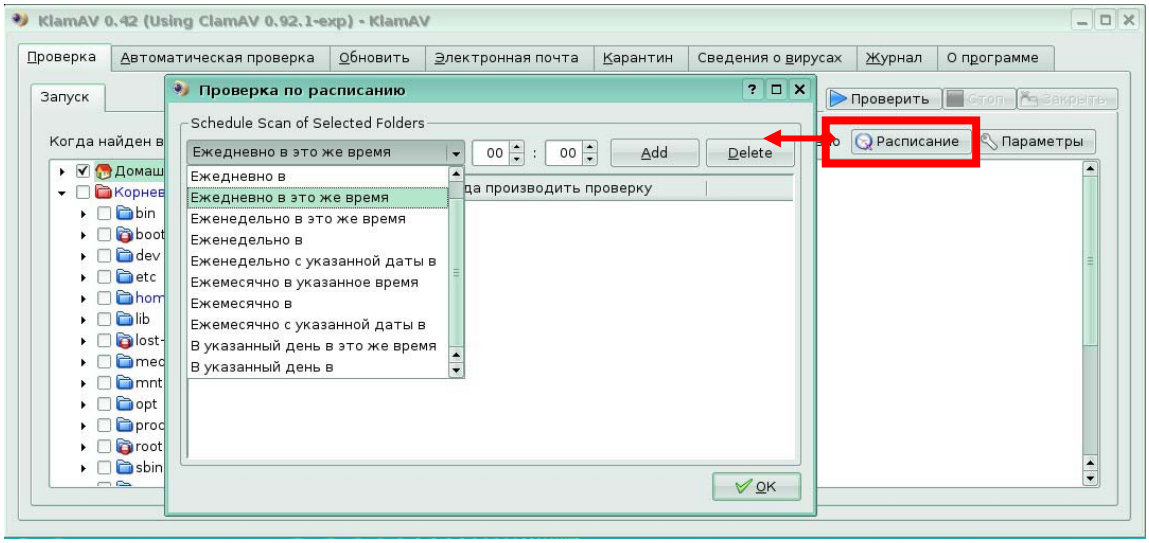

Рис. 7. Настройка расписания запуска задач проверки по требованию

Используя кнопку **«Параметры»** можно вызвать окно общих настроек антивируса (Рис. 8).

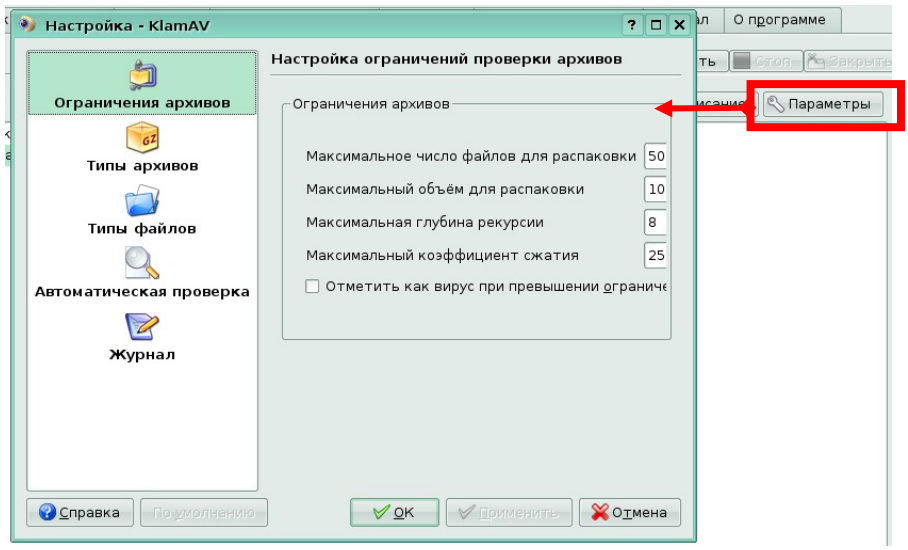

Рис. 8. Настройка параметров проверки архивов

В окне настроек можно сконфигурировать следующие параметры:

- ограничения по проверке файлов архивов (Рис. 8.);
- типы архивов, формат которых понимает и может проверить антивирус (Рис. 9);
- типы файлов специального формата, которые должен обрабатывать антивирус (Рис. 10);
- действия с объектами файловой системой, которые должны предваряться проверкой антивирусом запрошенных объектов (Рис. 11);
- указание типов событий, которые должны фиксироваться в журнале (Рис. 12).

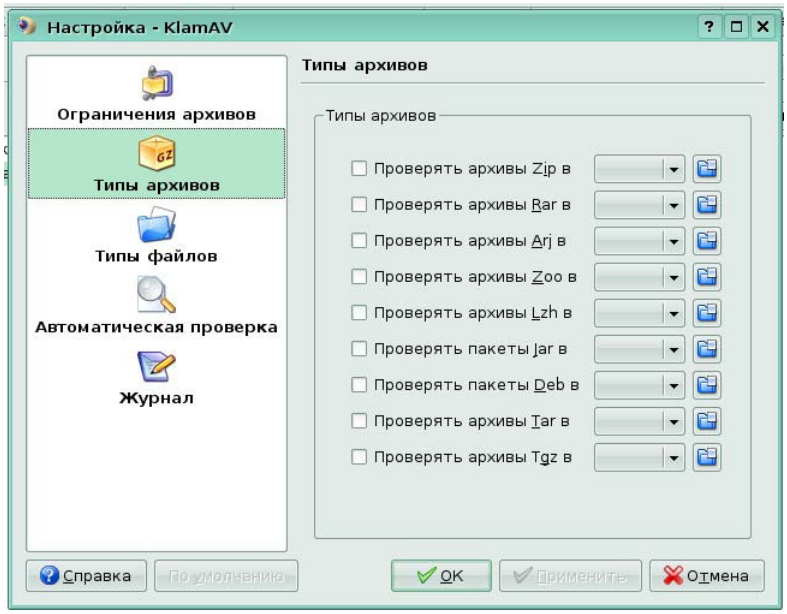

Рис.9.Настройка типов архивов, проверяемых антивирусом

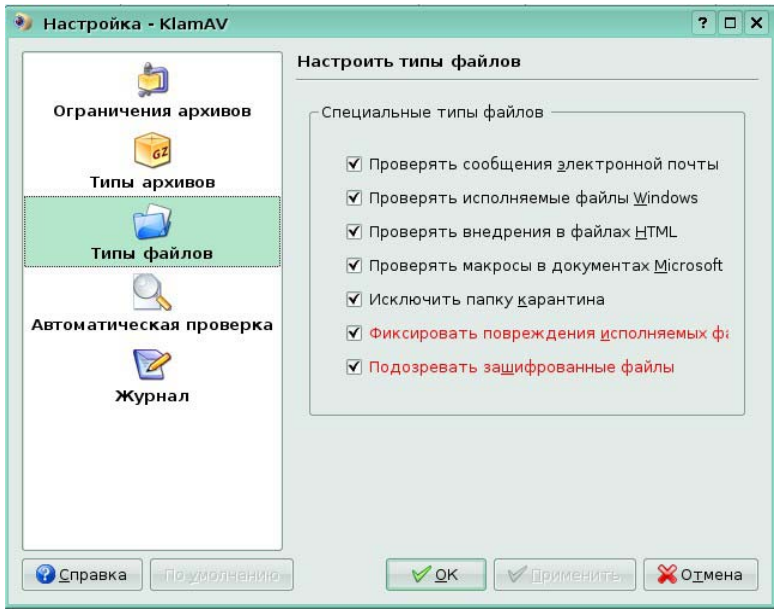

Рис.10.Настройка специальных типов файлов, проверяемых антивирусом

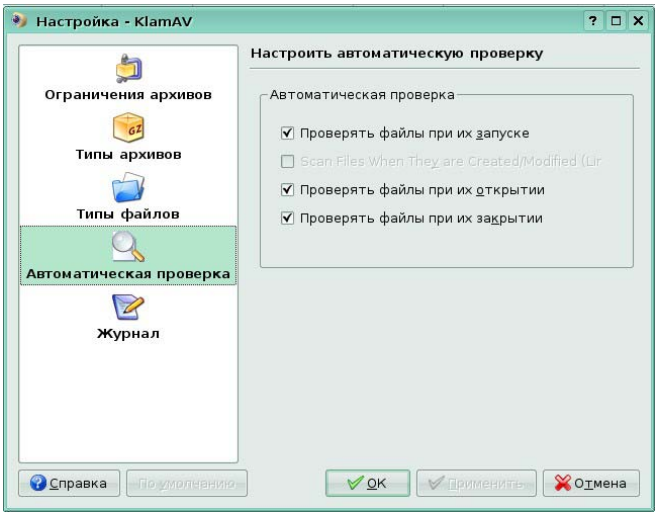

Рис.11. Настройка действий, при которых запускается проверка при доступе (антивирусный монитор)

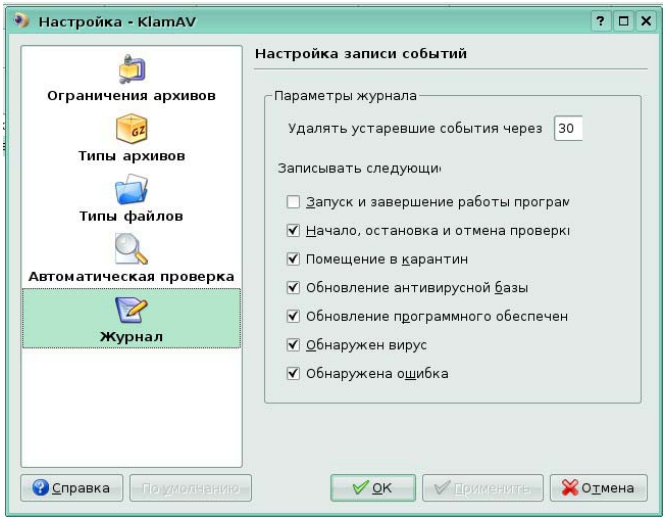

Рис.12. Настройка параметров ведения журнала

На закладке **«Автоматическая проверка»** (Рис. 13) настраивается интерфейс *Clamuko*, а именно, за какими директориями вести постоянное наблюдение и какие действия предпринимать при обнаружении вируса.

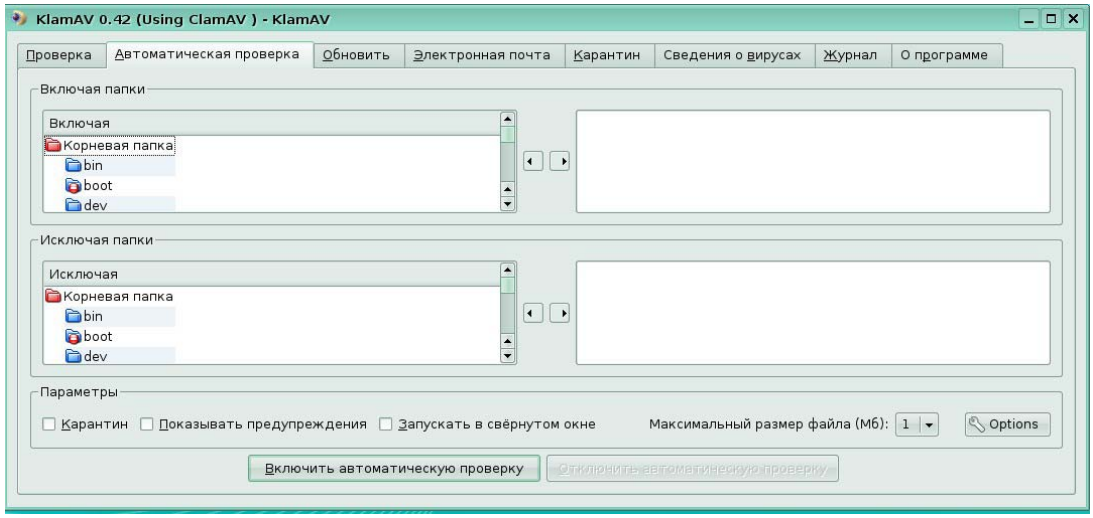

Рис. 13. Закладка настройки автоматической проверки файлов (антивирусного монитора)

Закладка **«Обновить»** (Рис. 14) служит для настройки процесса обновления антивирусных баз и модулей приложения. Здесь можно указать путь к директории, где должны располагаться антивирусные базы, параметры доступа к серверу обновлений через прокси-сервер, а также частоту проверки наличия обновлений. Т.е. производится настройка параметров модуля **freshclam**.

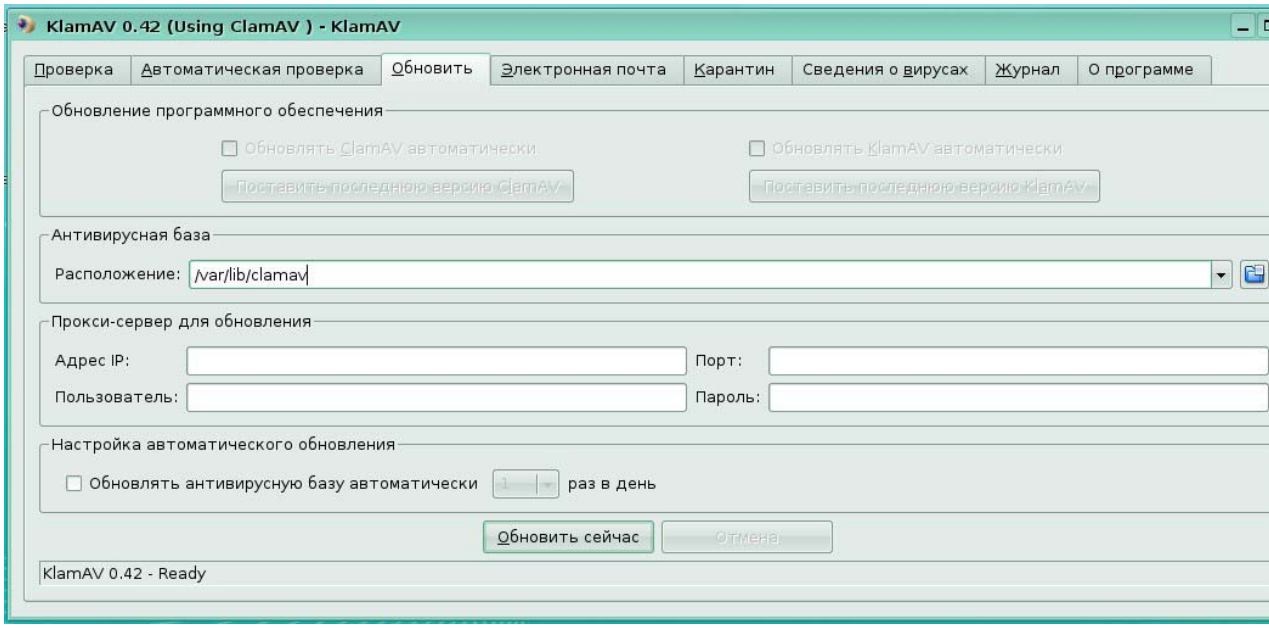

Рис. .14. Закладка настройки задачи автоматического получения обновлений

Закладка **«Электронная почта»** (Рис. 15) служит для выбора типа почтового клиента с целью интеграции с ним антивируса и запуска процесса проверки входящей/исходящей почты.

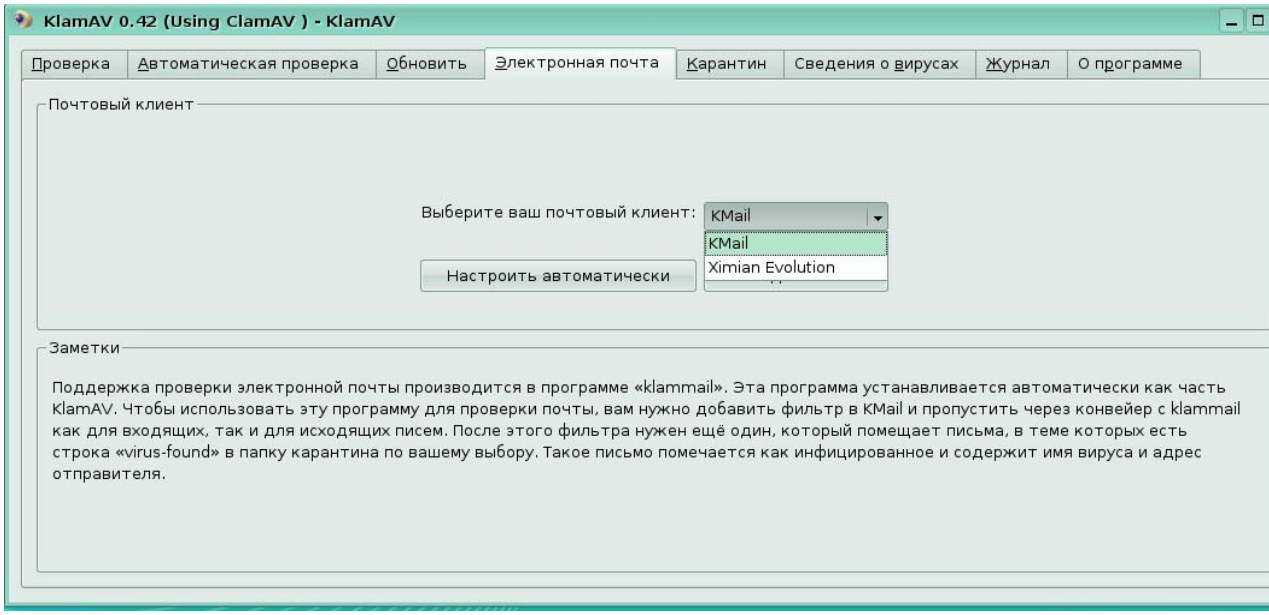

Рис. 15. Закладка настройки интеграции с клиентом электронной почты

Закладка **«Карантин»** (Рис. 16) служит для указания директории, используемой в качестве хранилища, в которое перемещаются обнаруженные вредоносные объекты, и для работы с этими объектами.

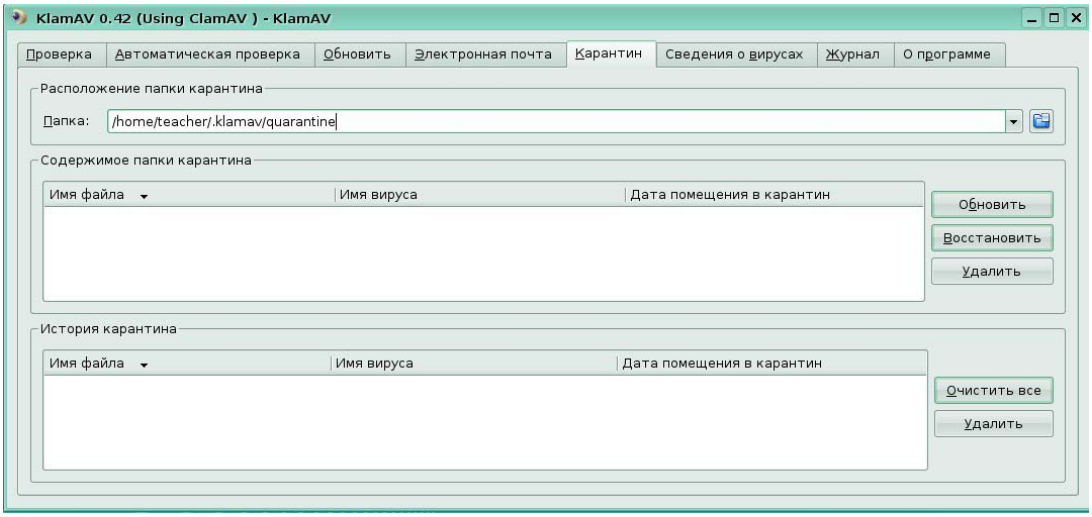

Рис. 16. Закладка настройки параметров карантина

Закладка **«Журнал»** (Рис. 17) служит для просмотра событий и результатов выполнения задач, запущенных программой KlamAV.

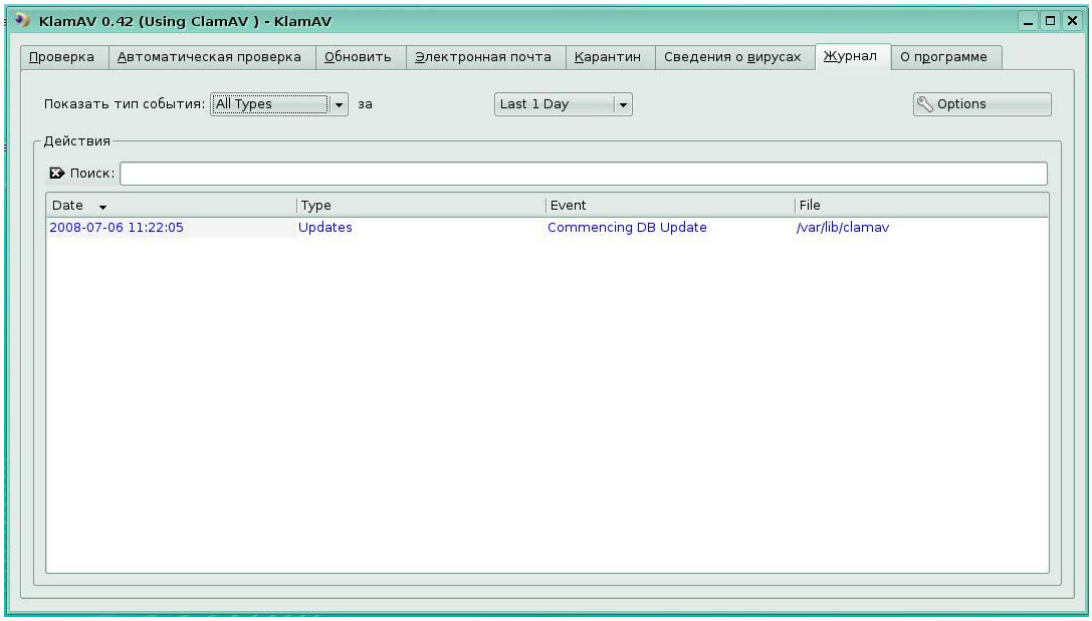

Рис. 17. Закладка просмотра журнала работы KlamAV

Закладки **«Сведения о вирусах»** (Рис. 18) и **«О программе»** (Рис. 19) позволяют посмотреть справочную информацию о вирусах, которые в состоянии обнаружить антивирус, в вирусной энциклопедии и информацию о самом продукте, в т.ч. на сайте проекта.

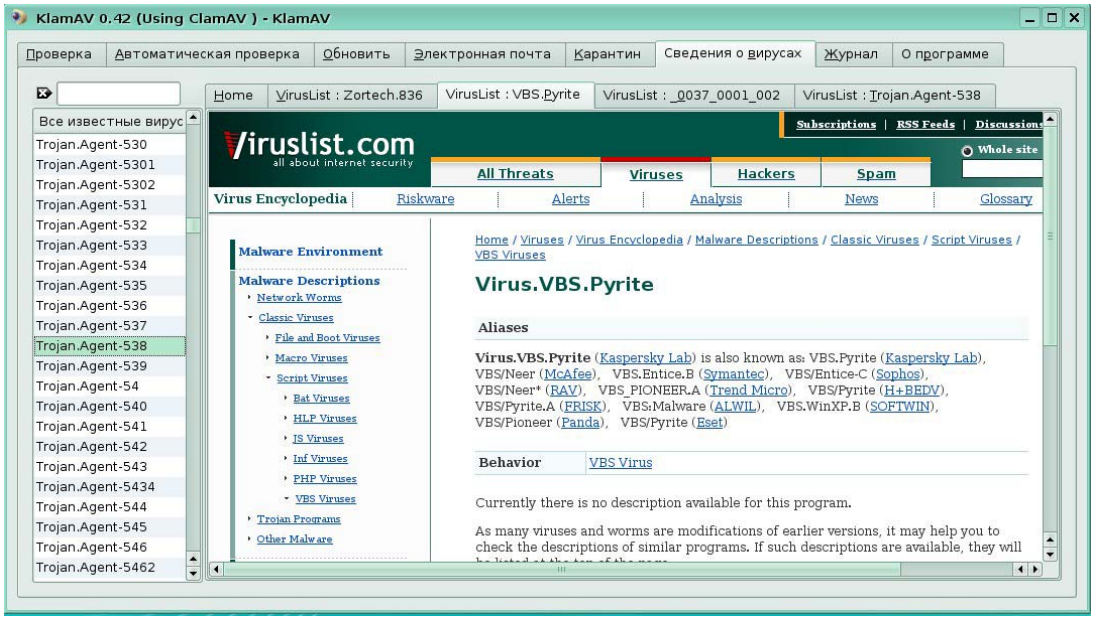

Рис. 18. Закладка просмотра информации о вирусах, из числа имеющихся в вирусной базе, в вирусной энциклопедии

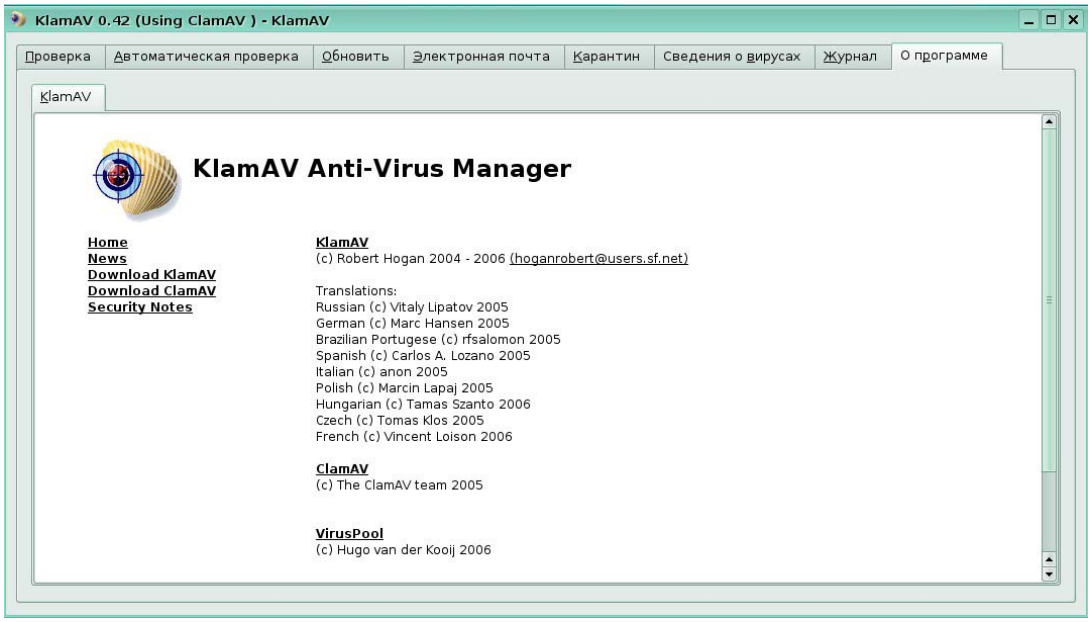

Рис. 19. Закладка просмотра информации о продукте KlamAV

2.3. Межсетевой экран netfilter/iptables

Netfilter — межсетевой экран (брандмауэр), встроенный в ядро Linux версий 2.4 и 2.6.

Iptables — название пользовательской утилиты (запускаемой из командной строки) предназначенной для управления *netfilter*. С её помощью администраторы создают и изменяют правила, управляющие фильтрацией и перенаправлением пакетов.

Некоторые авторы под словом *netfilter* имеют в виду только те элементы межсетевого экрана, которые непосредственно являются частью стека протоколов ядра Linux, а всё прочее (систему таблиц и цепочек) называют *iptables*, другие под термином *iptables* подразумевают и сам межсетевой экран *netfilter*. Поэтому, из-за не совсем ясной терминологии, весь проект (внутриядерный межсетевой экран вместе с пользовательской утилитой) просто именуется *netfilter/iptables*.

Межсетевой экран *netfilter* присутствует в ядре ОС изначально, а утилиту *iptables* возможно придётся инсталлировать отдельно. Как и раньше, инсталляцию лучше осуществлять с использованием менеджера пакетов Synaptic. Название пакета: *iptables*.

Вопреки очень распространённому мнению, ни *iptables*, ни *netfilter* не производят маршрутизацию пакетов и никак ей не управляют. Netfilter только фильтрует и модифицирует (в том числе, для NAT) пакеты по правилам, заданным администратором через утилиту *iptables*.

История проекта

Проект netfilter/iptables был основан в 1998. Автором является Расти Расселл (англ. Rusty Russell), он же автор проекта-предшественника *ipchains*. По мере развития проекта, в 1999 г. образовалась команда Netfilter Core Team (сокращено *coreteam*). Разработанный межсетевой экран получил официальное название netfilter. В марте 2000 г. он был включен в ядро Linux 2.3.

Изначально разработка *netfilter* и *iptables* шла совместно, поэтому в ранней истории этих проектов есть много общего. Предшественниками *iptables* были проекты *ipchains* (применялась для администрирования файрвола ядрах Linux версии 2.2) и *ipfwadm* (аналогично для ядер Linux версий 2.0). Последний был основан на BSD-утилите *ipfw*.

Iptables сохраняет идеологию, ведущую начало от *ipfwadm*: функционирование файрвола определяется набором правил, каждое из которых состоит из критерия и действия, применяемого к пакетам, подпадающим под этот критерий. В *ipchains* появилась концепция цепочек — независимых списков правил. Были введены отдельные цепочки для фильтрации входящих (INPUT), исходящих (OUTPUT) и транзитных (FORWARD) пакетов. В продолжении этой идеи, в *iptables* появились таблицы — независимые группы цепочек. Каждая таблица решала свою задачу — цепочки таблиц filter отвечали за фильтрацию, цепочки таблиц nat — за преобразование сетевых адресов (NAT), к задачам цепочки таблиц mangle относились прочие модификации заголовков пакетов (например, изменение TTL или TOS). Кроме того, была слегка изменена логика работы цепочек: в *ipchains* все входящие пакеты, включая транзитные, проходили цепочку INPUT. В *iptables* через INPUT проходят только пакеты, адресованные самому хосту.

Такое разделение функционала позволило *iptables* при обработке отдельных пакетов использовать информацию о соединениях в целом (ранее это было возможно только для NAT). В этом *iptables* значительно превосходит *ipchains*, так *iptables* может отслеживать состояние соединения (сеанса) и перенаправлять, изменять или отфильтровывать пакеты, основываясь не только на данных из их заголовков (источник, получатель) или содержимого пакетов, но и на основании данных о соединении. Такая возможность файрвола называется *stateful*-фильтрацией, в отличие от реализованной в *ipchains* примитивной *stateless*-фильтрации.

В будущем, разработчики *netfilter* планируют заменить *iptables* на *nftables* — инструмент нового поколения, пока находящийся в ранней стадии разработки.

Архитектура

Ключевыми понятиями *iptables* являются:

- *критерий* — логическое выражение, анализирующее свойства пакета и/или соединения и определяющее, подпадает ли данный конкретный пакет под действие текущего правила;
- *действие* — описание единичной операции, которую нужно произвести над пакетом и/или соединением в том случае, если они подпадают под действие этого правила;
- *счётчик* — компонент правила, обеспечивающий учет количества пакетов, которые попали под критерий данного правила, также счётчик учитывает суммарный объем таких пакетов в байтах;
- *правило* — состоит из критерия, действия и счётчика, если пакет соответствует критерию, к нему применяется действие, и он учитывается счётчиком; критерия может и не быть — тогда неявно предполагается критерий «все пакеты», указывать действие тоже не обязательно: в отсутствие действия правило будет работать только как счётчик;
- *цепочка* — упорядоченная последовательность правил, цепочки можно разделить на пользовательские и базовые;
- *базовая цепочка* — цепочка, создаваемая по умолчанию при инициализации таблицы; каждый пакет, в зависимости от того, предназначен ли он самому хосту, сгенерирован им или является транзитным, должен пройти положенный ему набор базовых цепочек различных таблиц; кроме того, базовая цепочка отличается от пользовательской наличием «действия по умолчанию» (*default policy*), это действие применяется к тем пакетам, которые не были обработаны другими правилами этой цепочки и вызванных из нее цепочек; имена базовых цепочек всегда записываются в верхнем регистре: PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING;
- *пользовательская цепочка* — цепочка, созданная пользователем, может использоваться только в пределах своей таблицы; рекомендуется не использовать для таких цепочек имена в верхнем регистре, чтобы избежать путаницы с базовыми цепочками и встроенными действиями;
- *таблица* — совокупность базовых и пользовательских цепочек, объединенных общим функциональным назначением; имена таблиц записываются в нижнем регистре, так как в принципе не могут конфликтовать с именами пользовательских цепочек: при вызове команды *iptables* таблица указывается в формате **`-t имя_таблицы`**, при отсутствии явного указания, используется таблица **filter**.

Описание функционирования

Для понимания функционирования *netfilter/iptables* далее будет использоваться следующая схема обработки IP-пакета, наложенная на модель OSI:

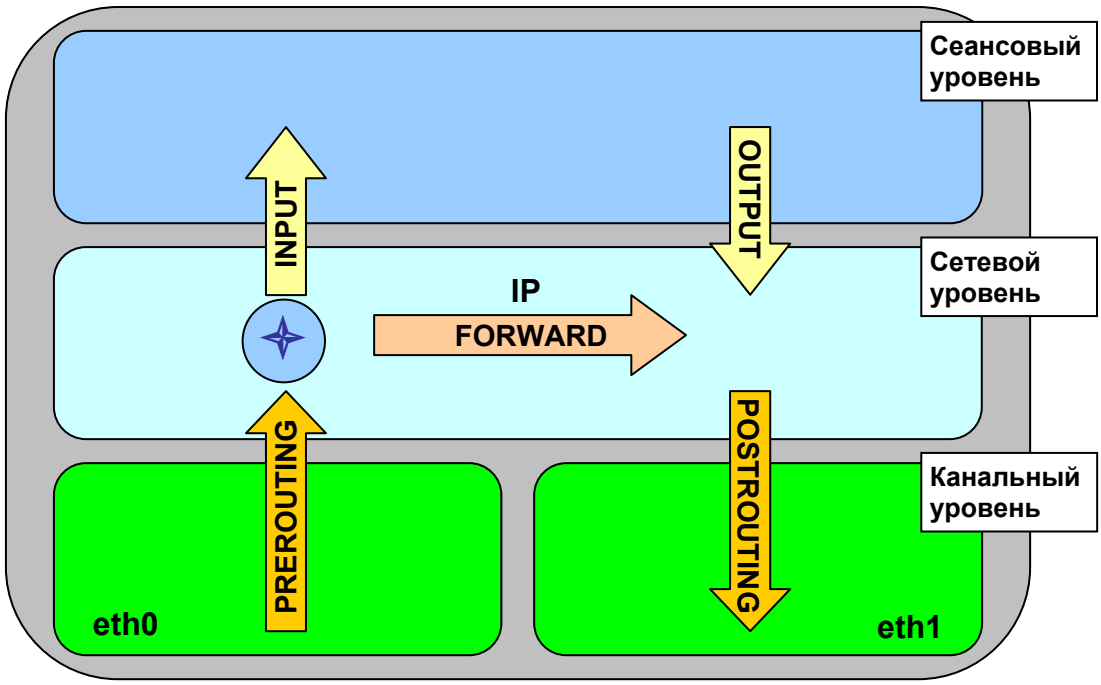

Рис. 20. Схема обработки IP-пакетов в стеке TCP/IP

В рассматриваемой схеме на канальном уровне функционируют два сетевых интерфейса: через *eth0* в систему поступает входящий трафик, а через *eth1* исходящий трафик покидает систему. На сетевом уровне функционирует протокол IP и происходит принятие решения о маршрутизации входящего трафика. На сеансовом и более высоких уровнях функционируют прикладные протоколы, принимающие поступающие пакеты и генерирующие исходящие пакеты.

Ключевым для понимания функционирования сетевого фильтра *netfilter/iptables* является понятие цепочки. Цепочка — это часть пути, проходимого пакетом в системе, где могут применяться те или иные правила обработки (фильтрации). На приведённой схеме цепочки изображены стрелками.

Изначально в систему встроены пять типов цепочек, называемых базовыми цепочками:

- **PREROUTING** — цепочка, предназначенная для обработки только что поступивших пакетов, по которым не принято никакого решения по маршрутизации, т.е. ещё неизвестно адресованы они процессу, функционирующему на данном компьютере или должны быть переданы обратно в сеть через другой сетевой интерфейс;

- **INPUT** — цепочка, предназначенная для обработки пакетов, адресованных процессу, функционирующему на данном компьютере (клиенту или серверу);
- **FORWARD** — цепочка, предназначенная для обработки пакетов, адресованных процессу, функционирующему на другом компьютере, т.е. транзитных пакетов;
- **OUTPUT** — цепочка, предназначенная для обработки пакетов, сгенерированных локальными процессами, независимо от того адресованы они процессу на этом же компьютере или будут переданы в сеть на другой компьютер;
- **POSTROUTING** — цепочка, предназначенная для окончательной обработки исходящих пакетов, независимо от того транзитные это пакеты или сгенерированные на том же самом компьютере.

Помимо понятия цепочки используется понятие таблица. Каждая таблица применяется для осуществления вполне определенных действий с пакетами и может быть частью тех или иных цепочек. Таблицы бывают четырёх типов:

- таблица *filter* содержит правила, которые занимаются собственно фильтрацией пакетов, т.е. выполняет функции межсетевого экрана; этот тип таблиц может быть использован в трёх цепочках: INPUT, FORWARD и OUTPUT;
- таблица *mangle* предназначена для внесения изменений в заголовки пакетов, например, для изменения параметров QoS, ToS, TTL, и нестандартных действий с пакетами, свойственных ОС Linux; этот тип таблиц может быть использован во всех пяти цепочках: PREROUTING, POSTROUTING, INPUT, FORWARD и OUTPUT;
- таблица *nat* используется для преобразования IP-адресов по технологии Network Address Translation (NAT), что, вообще говоря, тоже является изменением заголовков, но поскольку это затрагивает адреса отправителя и получателя (а следовательно, непосредственно влияет на решение о маршрутизации), то вынесено в отдельную таблицу; этот тип таблиц может быть использован в трёх цепочках: PREROUTING, POSTROUTING и OUTPUT;
- таблица *raw* существует для обработки пакетов без отслеживания IP-соединения, по которому они проходят и позволяет, соответственно, обрабатывать произвольные пакеты до передачи их системе определения состояний; ввиду узкой специфики использования эта таблица по умолчанию пуста; этот тип таблиц может быть использован в двух цепочках: PREROUTING и OUTPUT.

Порядок обработки сетевых пакетов

Несмотря на то, что в *netfilter* используется несколько таблиц и цепочек, пакет не может параллельно обрабатываться ими всеми. Он проходит через таблицы и цепочки последовательно. Запомнить порядок прохождения таблиц несложно, т.к. он одинаков для любого типа цепочки: raw → mangle → nat → filter. В зависимости от типа цепочки не используются те типы таблиц, которые в них неприменимы (см. описание типов таблиц, приведённое выше).

Обработка сетевого пакета может происходить по одному из трёх возможных сценариев:

- пакет поступает на сетевой интерфейс компьютера и в качестве получателя выступает процесс этого же компьютера, т.е. пакет является входящим;
- пакет передаётся в сеть через сетевой интерфейс компьютера и в качестве отправителя выступает процесс этого же компьютера, т.е. пакет является исходящим;
- пакет должен быть передан с одного сетевого интерфейса компьютера на другой, т.к. и отправителем и получателем выступают процессы, функционирующие на других компьютерах, т.е. пакет является транзитным.

Необходимо отметить, что когда происходит пересылка пакета между процессами одного и того же компьютера, то пакет уходит в интерфейс **loopback** и тут же из него появляется (в этом случае задействуются сразу и первый, и второй пункты приведённого списка). Последовательность обработки пакетов таблицами и цепочками, задействованными в каждом из сценариев, приведена в таблице:

Таблица 1. Порядок использования таблиц и цепочек при обработке различных типов пакетов

Таблица	*Цепочка*
Транзитные пакеты	
raw	PREROUTING
mangle	PREROUTING
nat	PREROUTING
mangle	FORWARD
filter	FORWARD
mangle	POSTROUTING
nat	POSTROUTING
Входящие пакеты	
mangle	PREROUTING

nat	PREROUTING
mangle	INPUT
filter	INPUT
Исходящие пакеты	
raw	OUTPUT
mangle	OUTPUT
nat	OUTPUT
filter	OUTPUT
mangle	POSTROUTING
nat	POSTROUTING

Каждая таблица представляет собой упорядоченный набор (список) правил вида «если для пакета выполняется такое-то условие, сделать то-то», т.е. это просто пары «условие-действие». Принятие решения по пакету при прохождении его по цепочкам и таблицам осуществляется следующим образом:

- пакет поступает на обработку в цепочку и последовательно проверяется на соответствие правилам всех включённых в цепочку таблиц;
- если на основании какого-либо правила текущей таблицы по пакету было принято решение: отбросить, то любая дальнейшая обработка пакета немедленно прекращается, пакет уничтожается и никуда больше не передаётся;
- если на основании какого-либо правила текущей таблицы по пакету было принято решение: пропустить, то обработка пакета *данной таблицей* немедленно прекращается, указанное в правиле действие применяется, и пакет передаётся следующей таблице текущей цепочки;
- если список правил текущей таблицы был исчерпан и решение по пакету не было принято, то для *базовых цепочек* применяется *действие по умолчанию*, а для *пользовательских цепочек* никакого действия над пакетом не производится, и пакет передаётся следующей таблице текущей цепочки;
- если в текущей цепочке больше нет таблиц, то пакет передаётся на обработку в следующую цепочку.

Следует отметить, что не может сложиться ситуация, что к пакету не будет применено хотя бы одно правило, т.к. в *базовых цепочках* всегда будет применено действие по умолчанию - *default policy*.

Т.о алгоритм обработки пакета сетевым фильтром *netfilter/iptables* может быть представлен так, как это изображено на рисунке (Рис. 21):

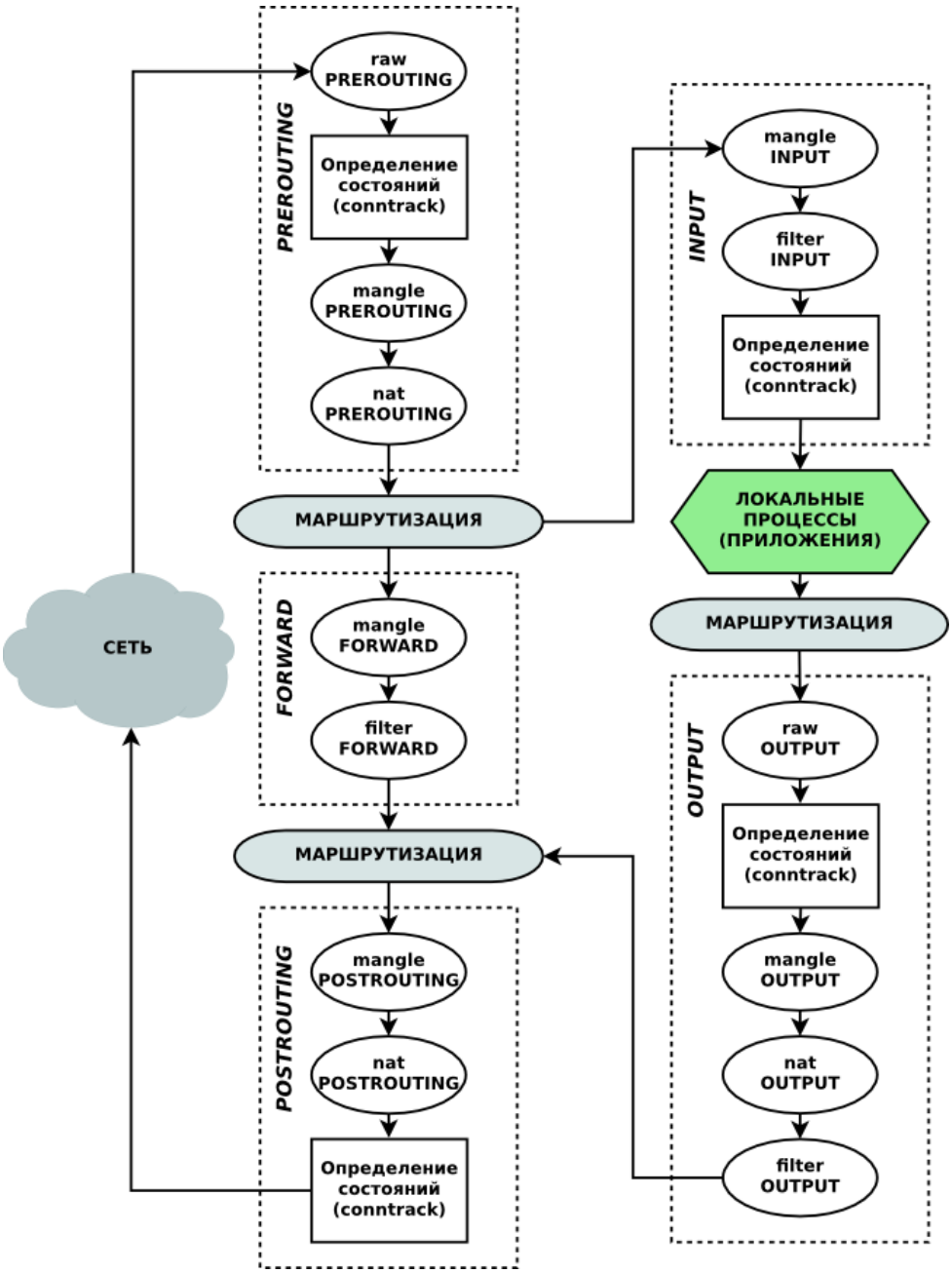

Рис. 21. Общая схема проверки пакета в netfilter

Необходимо отметить, что пользовательская документация *iptables* рассматривает взаимосвязь таблиц и цепочек с точки зрения программной реализации, а именно, основной логической единицей является таблица, в которой записаны правила с указанием, в какой цепочке это правило применять. Это неудобно при планировании применения и настройке продукта. Поэтому правильнее говорить о цепочках таблиц, т.к. пакет обрабатывается правилами, помещёнными в таблицу, в пределах текущей цепочки. И невозможно перейти в другую цепочку принудительно, пока полностью не завершится обработка пакета правилами текущей цепочки. Эта концепция отражена и в приведённой выше схеме (Рис. 21).

Управление цепочками с помощью iptables

Управление цепочками производится с помощью программы *iptables*. Для её использования привилегии суперпользователя (root).

Чтобы посмотреть, какие правила действуют в настоящий момент в системе, необходимо выполнить команду:

```
[root@mlinux ~]# iptables-save
# Generated by iptables-save v1.3.7 on Tue Sep 22 14:36:40 2009
*filter
:INPUT ACCEPT [2998:1441761]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [2617:220737]
:stdin - [0:0]
COMMIT
# Completed on Tue Sep 22 14:36:40 2009
```

В приведённом примере никаких заданных администратором правил нет.

Как строить правила

Каждая строка, которую вы вставляете в ту или иную цепочку, должна содержать отдельное правило. Каждое правило - это строка, содержащая в себе критерии определяющие, подпадает ли пакет под действие данного правило, и действие, которое необходимо выполнить в случае выполнения критерия. В общем виде команда создания правила выглядит следующим образом:

```
iptables [-t table] command [match] [target/jump]
```

Нигде в документации не утверждается, что описание действия (target/jump) должно стоять последним в строке, однако, такая нотация наиболее удобна. Как бы то ни было, но чаще всего встречается именно такой способ записи правил.

Если в правило не включается спецификатор **-t table**, то по умолчанию предполагается использование таблицы *filter*, если же предполагается использование другой таблицы, то это требуется указать явно. Спецификатор таблицы так же можно указывать в любом месте строки правила, однако стандартом считается указание таблицы в начале правила.

Непосредственно за именем таблицы, должна стоять команда. Если спецификатора таблицы нет, то команда всегда должна стоять первой. Команда определяет действие для программы iptables, например: вставить правило, или добавить правило в конец цепочки, или удалить правило и т.п.

Раздел match задает критерии проверки, по которым определяется подпадает ли пакет под действие данного правила или нет. Можно указать различные критерии: IP-

адрес источника пакета или сети, IP-адрес получателя пакета, порт, протокол, сетевой интерфейс и т.д.

Раздел target указывает, какое действие должно быть произведено при условии выполнения критериев, указанных в правиле. Межсетевой экран *netfilter* может передать пакет в другую цепочку правил, отбросить пакет, отправить источнику пакета сообщение об ошибке и т.п.

Таблицы

Опция **-t** указывает на используемую таблицу. По умолчанию используется таблица *filter*. Описание значений параметра –t и их назначение приведено в следующей таблице:

Таблица 2. Таблицы

Таблица	*Описание*
nat	Таблица *nat* используется главным образом для преобразования сетевых адресов (Network Address Translation). Через эту таблицу проходит только первый пакет из потока. Преобразования адресов автоматически применяется ко всем последующим пакетам, поэтому не следует осуществлять какую-либо фильтрацию в этой таблице. В цепочке PREROUTING таблица используется для внесения изменений в пакеты на входе в брандмауэр. В цепочке OUTPUT таблица используется для преобразования адресов в пакетах, созданных приложениями внутри брандмауэра, перед принятием решения о маршрутизации. И в цепочке POSTROUTING - для преобразования пакетов перед передачей их в сеть.
mangle	Эта таблица используется для внесения изменений в заголовки пакетов. Примером может служить изменение полей TTL или TOS. В цепочке PREROUTING таблица используется для внесения изменений на входе в брандмауэр, перед принятием решения о маршрутизации. В цепочке POSTROUTING - для внесения изменений на выходе из брандмауэра, после принятия решения о маршрутизации. В цепочке INPUT - для внесения изменений в пакеты перед тем как они будут переданы локальному приложению внутри брандмауэра. В цепочке OUTPUT - для внесения изменений в пакеты, поступающие от приложений внутри брандмауэра. В цепочке FORWARD - для внесения изменений в транзитные пакеты после первого принятия решения о

	маршрутизации, но перед последним принятием решения о маршрутизации. Таблица *mangle* ни в коем случае не должна использоваться для преобразования сетевых адресов или маскарадинга (Network Address Translation, Masquerading), поскольку для этих целей существует таблица *nat*.
filter	Таблица *filter* используется, главным образом, для фильтрации пакетов. Например, в этой таблице допустимо применять действия DROP, LOG, ACCEPT или REJECT без каких-либо ограничений, которые существуют в других таблицах. В цепочке FORWARD таблица используется для фильтрации пакетов, идущих транзитом через брандмауэр. В цепочке INPUT - для фильтрации пакетов, которые предназначены локальным приложениям. В цепочке OUTPUT - для фильтрации исходящих пакетов, сгенерированных приложениями на самом брандмауэре.

Команды

Ниже приводится список команд и порядок их использования. Посредством команд программе iptables сообщается, какую работу необходимо выполнить. Обычно предполагается одно из двух действий - добавление нового правила в цепочку или удаление существующего правила из той или иной таблицы. Список команд, которые используются в iptables приведён в следующей таблице:

Таблица 3. Команды

Команда	`-A, --append`
Пример	`iptables -A INPUT ...`
Описание	Добавляет новое правило в конец заданной цепочки.
Команда	`-D, --delete`
Пример	`iptables -D INPUT --dport 80 -j DROP` `iptables -D INPUT 1`
Описание	Удаление правила из цепочки. Команда имеет два формата записи, первый -- когда задаётся критерий сравнения с опцией -D (см. первый пример), второй - порядковый номер правила. Если задаётся критерий сравнения, то удаляется правило, которое имеет в себе этот критерий, если задаётся номер правила, то будет удалено правило с заданным номером. Счет правил в цепочках начинается с 1.
Команда	`-R, --replace`

Пример	`iptables -R INPUT 1 -s 192.168.0.1 -j DROP`
Описание	Эта команда заменяет одно правило другим. В основном она используется во время отладки новых правил.
Команда	`-I, --insert`
Пример	`iptables -I INPUT 1 --dport 80 -j ACCEPT`
Описание	Вставляет новое правило в цепочку. Число, следующее за именем цепочки указывает номер правила, перед которым нужно вставить новое правило, другими словами число задает номер для вставляемого правила. В примере выше, указывается, что данное правило должно быть 1-м в цепочке INPUT.
Команда	`-L, --list`
Пример	`iptables -L INPUT`
Описание	Вывод списка правил в заданной цепочке, в данном примере предполагается вывод правил из цепочки INPUT. Если имя цепочки не указывается, то выводится список правил для всех цепочек. Формат вывода зависит от наличия дополнительных ключей в команде, например -n, -v, и пр.
Команда	`-F, --flush`
Пример	`iptables -F INPUT`
Описание	Очистка, т.е. удаление всех правил из заданной цепочки (таблицы). Если имя цепочки и таблицы не указывается, то удаляются все правила из всех цепочек.
Команда	`-Z, --zero`
Пример	`iptables -Z INPUT`
Описание	Обнуление всех счётчиков в заданной цепочке. Если имя цепочки не указывается, то подразумеваются все цепочки. Допускается совместное использование команд -L и -Z. В этом случае будет сначала выведен список правил со счётчиками, а затем произойдет обнуление счётчиков.
Команда	`-N, --new-chain`
Пример	`iptables -N allowed`
Описание	Создается новая цепочка с заданным именем в заданной таблице. В приведенном примере создается новая цепочка с именем *allowed*. Имя цепочки должно быть уникальным и не должно совпадать

	с зарезервированными именами цепочек и действий (такими как DROP, REJECT и т.п.)
Команда	`-X, --delete-chain`
Пример	`iptables -X allowed`
Описание	Удаление заданной цепочки из заданной таблицы. Удаляемая цепочка не должна иметь правил и не должно быть ссылок из других цепочек на удаляемую цепочку. Если имя цепочки не указано, то будут удалены все цепочки заданной таблице, кроме встроенных.
Команда	`-P, --policy`
Пример	`iptables -P INPUT DROP`
Описание	Задает *политику по умолчанию* для заданной цепочки. Политика по умолчанию определяет действие, применяемое к пакетам, не попавшим под действие ни одного из правил в цепочке. В качестве политики по умолчанию допускается использовать только DROP и ACCEPT.
Команда	`-E, --rename-chain`
Пример	`iptables -E allowed disallowed`
Описание	Команда -E выполняет переименование пользовательской цепочки. В примере цепочка *allowed* будет переименована в цепочку *disallowed*. Эти переименования не изменяют порядок работы, а носят только косметический характер.

Команда должна быть указана всегда. Список доступных команд можно просмотреть с помощью команды iptables -h или, что тоже самое, iptables --help. Некоторые команды могут использоваться совместно с дополнительными ключами. Ниже приводится список дополнительных ключей и описывается результат их действия за исключением ключей, используемых при построении критериев (matches) или действий (targets).

Таблица 4. Дополнительные ключи

Ключ	`-v, --verbose`
Команды, с которыми используется	`--list, --append, --insert, --delete, --replace`
Описание	Используется для повышения информативности вывода и, как правило, используется совместно с командой --list. В

	случае использования с командой --list, в вывод этой команды включаются так же имя интерфейса, счётчики пакетов и байт для каждого правила. Если ключ -v, --verbose используется с командами --append, --insert, --delete или --replace, то будет выведен подробный отчет о произведенной операции.
Ключ	`-x, --exact`
Команды, с которыми используется	`--list`
Описание	Для всех чисел в выходных данных выводятся их точные значения без округления и без использования множителей K, M, G. Этот ключ используется только с командой --list и не применим с другими командами.
Ключ	`-n, --numeric`
Команды, с которыми используется	`--list`
Описание	Заставляет *iptables* выводить IP-адреса и номера портов в числовом формате, предотвращая попытки преобразовать их в символические имена. Данный ключ используется только с командой --list.
Ключ	`--line-numbers`
Команды, с которыми используется	`--list`
Описание	Ключ --line-numbers включает режим вывода номеров строк при отображении списка правил командой --list. Номер строки соответствует позиции правила в цепочке. Этот ключ используется только с командой --list.
Ключ	`-c, --set-counters`
Команды, с которыми используется	`--insert, --append, --replace`

Описание	Этот ключ используется для установки начального значения счётчиков пакетов и байт в заданное значение при создании нового правила. Например, ключ --set-counters 20 4000 установит счётчик пакетов равным 20, а счётчик байт равным 4000.

Критерии

Ниже рассмотрены критерии выбора пакетов, к которым должно быть применено правило, разбитые на группы:

- общие критерии, которые могут использоваться в любых правилах;
- TCP-критерии, которые применяются только к TCP-пакетам;
- UDP-критерии, которые применяются только к UDP-пакетам;
- ICMP-критерии для работы с ICMP-пакетами.

Общие критерии

Общие критерии допустимо употреблять в любых правилах, они не зависят от типа протокола.

Таблица 5. Общие критерии

Критерий	`-p, --protocol`
Пример	`iptables -A INPUT -p tcp`
Описание	Основные значения: tcp, udp, icmp, или all. Протокол можно задать с помощью номера, или названия, указанного в файле /etc/protocols. Знак «!» перед именем протокола изменяет критерий на противоположенный (например, !tcp означает: «любой протокол, кроме TCP»). Значение «Любой протокол» можно указать с помощью слова all или числа 0. Если протокол не указан, то подразумевается «Любой протокол».
Критерий	`-s, --src, --source`
Пример	`iptables -A INPUT -s 192.168.1.1`
Описание	IP-адрес(а) источника пакета. Можно использовать единственный адрес, как показано в примере. Но допускается указывать подсеть в виде *address/mask*, например, 192.168.0.0/24 или 192.168.0.0/255.255.255.0. Знак «!» перед адресом изменяет критерий на противоположенный, т.е. запись --source ! 192.168.0.0/24 означает: «любой адрес, кроме подсети 192.168.0.0/24».
Критерий	`-d, --dst, --destination`

Пример	`iptables -A INPUT -d 192.168.1.1`
Описание	IP-адрес(а) получателя. Использует синтаксис, аналогичный критерию --source.
Критерий	`-i, --in-interface`
Пример	`iptables -A INPUT -i eth0`
Описание	Указывает интерфейс, через который получен пакет. Использование этого критерия допускается только в цепочках INPUT, FORWARD и PREROUTING, в любых других случаях будет вызывать сообщение об ошибке. Знак «!» перед адресом изменяет критерий на противоположенный. Если указанное имя интерфейса кончается на «+», то критерию соответствуют все интерфейсы, чьи имена начинаются с заданной строки, например, -i PPP+ означает «любой PPP-интерфейс», а запись -i ! eth+ означает «любой интерфейс, кроме eth». Если параметр --in-interface не указан, то критерию соответствуют пакеты из любого сетевого интерфейса.
Критерий	`-o, --out-interface`
Пример	`iptables -A FORWARD -o eth0`
Описание	Задает имя выходного интерфейса. Этот критерий допускается использовать только в цепочках OUTPUT, FORWARD и POSTROUTING, в противном случае будет генерироваться сообщение об ошибке. Использует синтаксис, аналогичный критерию --in-interface.
Критерий	`-f, --fragment`
Пример	`iptables -A INPUT -f`

Описание	Критерию соответствуют только фрагментированные пакеты, начиная со второго фрагмента. Знак «!» перед адресом изменяет критерий на противоположенный. У таких фрагментов, начиная со второго, нет заголовка с портами отправителя и получателя (или с типом ICMP). Следовательно, такие фрагменты не соответствуют критериям, содержащим номера портов. С помощью фрагментированных пакетов могут производиться атаки на ваш компьютер, так как фрагменты пакетов могут не определяться другими правилами.

Неявные критерии

Эти критерии так названы потому, что они подгружаются неявно и становятся доступны при указании соответствующего критерия --protocol. В настоящий момент существует три автоматически подгружаемых расширения, это TCP-критерии, UDP-критерии и ICMP-критерии. Загрузка этих расширений может производиться и явным образом с помощью ключа -m, -match, например -m tcp.

TCP критерии

Этот набор критериев используется при указании в правилах критерия **--protocol tcp**.

Таблица 6. TCP критерии

Критерий	`--sport, --source-port`
Пример	`iptables -A INPUT -p tcp --sport 22`
Описание	Определяет порт, с которого был отправлен пакет. Порт можно задать с помощью номера, или названия, указанного /etc/services. Номера портов могут задаваться в виде интервала из минимального и максимального номеров, например, --source-port 22:80. Если опускается минимальный номер порта, то в качестве начала диапазона принимается число 0, например, --source-port :80. Если опускается максимальный номер порта, то в качестве конца диапазона принимается число 65535, например, --source-port 22:. Знак «!» перед номером порта изменяет критерий на противоположенный, например, запись --source-port ! 22 означает «любой порт, кроме 22».

Критерий	`--dport, --destination-port`
Пример	`iptables -A INPUT -p tcp --dport 22`
Описание	Определяет порт, которому адресован пакет. Использует синтаксис, аналогичный критерию --source-port.
Критерий	`--tcp-flags`
Пример	`iptables -p tcp --tcp-flags SYN,FIN,ACK SYN`
Описание	Позволяет указать список установленных и снятых TCP-флагов. В маске перечисляются (через запятую, без пробелов) все проверяемые флаги, далее, после пробела, перечисляются (также через запятую) те из них, которые должны быть установлены. Все остальные, перечисленные в маске, флаги должны быть сняты. Возможные флаги: SYN, ACK, FIN, RST, URG, PSH. Также можно использовать псевдофлаги ALL и NONE, обозначающие «все флаги» и «ни одного флага», соответственно.

UDP критерии

Этот набор критериев используется при указании в правилах критерия **--protocol udp**.

Таблица 7. UDP критерии

Критерий	`--sport, --source-port`
Пример	`iptables -A INPUT -p udp --sport 53`
Описание	Определяет порт, с которого был отправлен пакет. Порт можно задать с помощью номера, или названия, указанного /etc/services. Номера портов могут задаваться в виде интервала из минимального и максимального номеров, например, --source-port 22:80. Если опускается минимальный номер порта, то в качестве начала диапазона принимается число 0, например, --source-port :80. Если опускается максимальный номер порта, то в качестве конца диапазона принимается число 65535, например, --source-port 22:. Знак «!» перед номером порта изменяет критерий на противоположенный, например, запись --source-port ! 22 означает «любой порт, кроме 22».

Критерий	`--dport, --destination-port`
Пример	`iptables -A INPUT -p tcp --dport 22`
Описание	Определяет порт, которому адресован пакет. Использует синтаксис, аналогичный критерию --source-port.

ICMP критерии

Этот набор критериев используется при указании в правилах критерия **--protocol icmp**.

Протокол ICMP используется для передачи сообщений об ошибках и для управления соединением. Он не является подчиненным протоколу IP, но тесно с ним взаимодействует, поскольку помогает обрабатывать ошибочные ситуации. Заголовки ICMP-пакетов очень похожи на IP-заголовки, но имеют и отличия. Главное свойство этого протокола заключается в типе заголовка, который содержит информацию о том, что это за пакет. Например, при попытке соединиться с недоступным хостом, генерируется ответное сообщение *ICMP host unreachable*.

Таблица 8. ICMP критерии

Критерий	`--icmp-type`
Пример	`iptables -A INPUT -p icmp --icmp-type 8`
Описание	Тип сообщения ICMP определяется номером или именем. Числовые значения определtys в RFC 792. Чтобы получить полный список допустимых значений критерия выполните команду iptables --protocol icmp --help. Знак «!» перед номером типа сообщения изменяет критерий на противоположенный, например, запись --icmp-type ! 8 означает «любое ICMP-сообщение, кроме echo-запросов».

Действия

Действие - это указание, каким образом необходимо обработать пакет, если он совпал с заданным в правиле критерием. Например, можно применить действие DROP или ACCEPT к пакету, в зависимости от наших нужд. Существует и ряд других действий, которые описываются ниже.

В результате выполнения одних действий, обработка пакета в текущей цепочке прекращается, например, DROP и ACCEPT, в результате других, после выполнения некоторых операций, обработка пакета продолжается, например, LOG, в результате

работы третьих пакет видоизменяется, например, DNAT и SNAT, TTL и TOS, но так же продолжает продвижение по цепочке.

Действие ACCEPT не имеет дополнительных ключей. Если над пакетом выполняется действие ACCEPT, то пакет прекращает движение по цепочке (и всем вызвавшим цепочкам, если текущая цепочка была вложенной) и считается ПРИНЯТЫМ, тем не менее, пакет продолжит движение по цепочкам в других таблицах и может быть отвергнут там. Действие задается с помощью ключа -j ACCEPT.

Действие DROP просто отбрасывает обрабатываемый пакет. Обработка пакетов, к которым применено действие DROP прекращается полностью, т.е. они не передаются в другие таблицы, как это происходит в случае с действием ACCEPT.

Действие REJECT используется, как правило, в тех же ситуациях, что и DROP, но в отличие от DROP, команда REJECT выдает сообщение об ошибке на хост, передавший пакет. Действие REJECT может использоваться только в цепочках INPUT, FORWARD и OUTPUT (и во вложенных в них цепочках). Пока существует только единственный ключ, управляющий действием REJECT.

Таблица 9. Действие REJECT

Ключ	`--reject-with`
Пример	`iptables -A FORWARD -p TCP --dport 22 -j REJECT --reject-with tcp-reset`
Описание	Указывает, какое сообщение необходимо передать в ответ, если пакет совпал с заданным критерием. При применении действия REJECT к пакету, сначала на хост-отправитель будет отослан указанный ответ, а затем пакет будет отброшен. По умолчанию передается сообщение *port-unreachable*.

Действие RETURN вызывает прекращение обработки пакета в текущей цепочке правил и возврат в вызывающую цепочку, если текущая цепочка была вложенной, либо применение к пакету политики по умолчанию, в противном случае. Обычно, в качестве политики по умолчанию используются действия ACCEPT или DROP .

Действие LOG служит для занесения в журнал (протоколирования) информации из отдельных пакетов и о произошедших событиях. В журнал могут заноситься заголовки IP-пакетов и другая интересующая информация. Информация из журнала может быть затем извлечена и обработана с помощью программ dmesg, syslogd или иных, предназначенных для работы с системными журналами.

Действием LOG управляют пять ключей, которые перечислены ниже.

Таблица 10. Ключи действия LOG

Ключ	`--log-level`
Пример	`iptables -A FORWARD -p tcp -j LOG --log-level debug`
Описание	Используется для задания уровня протоколирования (log level). Полный список уровней перечислен в руководстве по syslog.conf.
Ключ	`--log-prefix`
Пример	`iptables -A INPUT -p tcp -j LOG --log-prefix "INPUT packets"`
Описание	Ключ задает текст (префикс), которым будут предваряться все сообщения iptables. Сообщения со специфичным префиксом затем легко можно найти, к примеру, с помощью grep. Префикс может содержать до 29 символов, включая пробелы.
Ключ	`--log-tcp-sequence`
Пример	`iptables -A INPUT -p tcp -j LOG --log-tcp-sequence`
Описание	Используется для записи в журнал поля TCP Sequence из TCP-заголовка пакета.
Ключ	`--log-tcp-options`
Пример	`iptables -A FORWARD -p tcp -j LOG --log-tcp-options`
Описание	Используется для записи в журнал поля Options из TCP-заголовка пакета.
Ключ	`--log-ip-options`
Пример	`iptables -A FORWARD -p tcp -j LOG --log-ip-options`
Описание	Используется для записи в журнал поля Options из IP-заголовка пакета.

Следующие действия: SNAT, DNAT, MASQUERADE и REDIRECT могут быть использованы только в таблице *nat*. NAT(Network Address Translation)-преобразования позволяют заменить в заголовках пакетов IP адрес источника или назначения, а также

порт источника или назначения. В *netfilter* существуют два основных действия, позволяющие осуществить NAT-преобразования: SNAT — для замены адреса и/или порта источника и DNAT — для замены адреса и/или порта назначения.

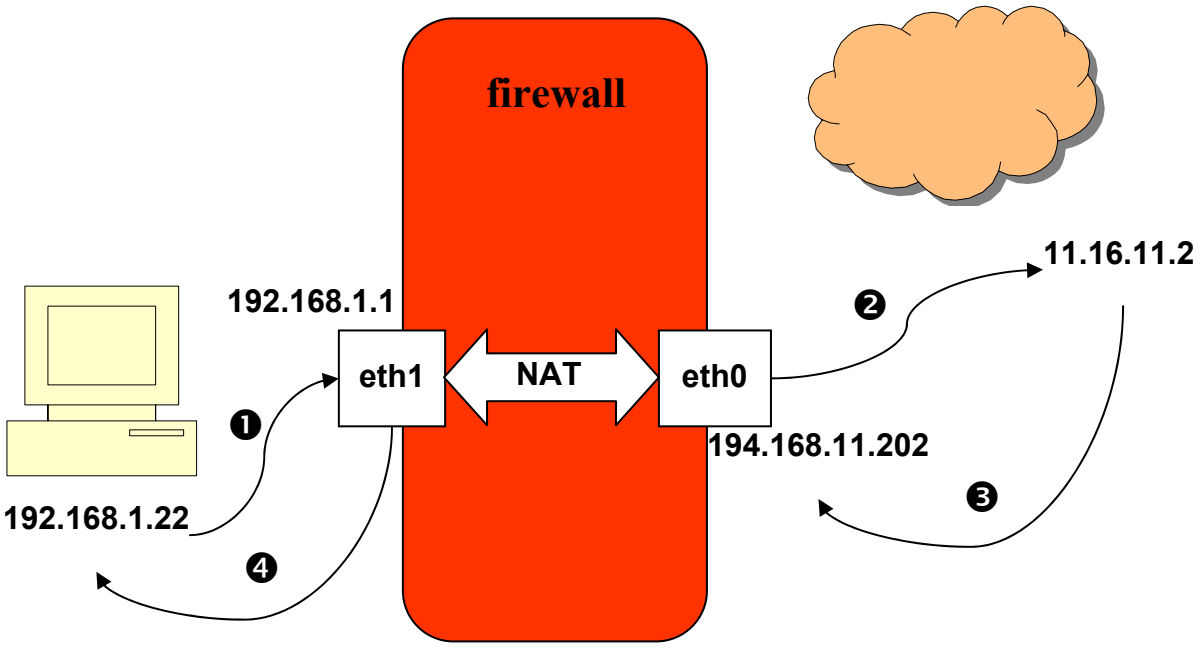

Рис. 22. Общая схема функционирования NAT

Кроме того, у SNAT и DNAT существуют частные случаи: MASQUERADE — частный случай SNAT, REDIRECT — частный случай DNAT.

На Рис. 22 приведена общая схема работы NAT. При этом при работе NAT в памяти межсетевого экрана создаётся таблица, в которой хранятся параметры текущего транслируемого сеанса:

до NAT		после NAT	
Source IP	Port source	Source IP	Port source
192.168.1.22	2023	194.168.11.202	2023

Если случится так, что два хоста из внутренней сети обратятся к одному и тому же ресурсу и у них будет использован один и тот же порт отправителя, то NAT произведёт также замену номера порта:

до NAT		после NAT	
Source IP	Port source	Source IP	Port source
192.168.1.22	2023	194.168.11.202	2023
192.168.1.25	2023	194.168.11.202	2024

Не смотря на то, что сетевой фильтр и NAT являются частями *netfilter* — это разные подсистемы, предназначенные для разных целей. Для функционироания сетевого фильтра используется таблица filter, для NAT преобразований — таблица *nat*.

Настоятельно не рекомендуется помещать правила фильтрации в цепочки таблицы *nat*.

Действие SNAT (Source Network Address Translation) используется для преобразования сетевых адресов отправителей, т.е. изменение исходящего IP-адреса в IP-заголовке пакета, что было показано выше. Это действие используется для предоставления доступа в Интернет компьютерам локальной сети при наличии только одного публичного IP-адреса. Для этого необходимо включить пересылку пакетов (*forwarding*) в ядре и затем создать правила, которые будут транслировать исходящие IP-адреса локальной сети в реальный внешний адрес. В результате все хосты в Интернете будут считать, что источником запросов является межсетевой экран *netfilter*.

SNAT допускается выполнять только в таблице *nat* в цепочке POSTROUTING. Другими словами, только здесь допускается преобразование исходящих адресов. Если первый пакет в соединении подвергся преобразованию исходящего адреса, то все последующие пакеты из этого же соединения будут преобразованы автоматически без повторного применения этой цепочки правил.

Таблица 11. Действие SNAT

Ключ	`--to-source`
Пример	`iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT --to-source 194.236.50.155-194.236.50.160:1024-32000`
Описание	Ключ --to-source используется для указания адреса, присваиваемого пакету. Иными словами, этот ключ указываете IP-адрес, который будет подставлен в заголовок пакета в качестве исходящего. Если вы собираетесь перераспределять нагрузку между несколькими брандмауэрами, то можно указать диапазон адресов, где начальный и конечный адреса диапазона разделяются дефисом, например: 194.236.50.155-194.236.50.160. Тогда, конкретный IP-адрес будет выбираться из диапазона случайным образом для каждого нового потока. Дополнительно можно указать диапазон портов, которые будут использоваться только для нужд SNAT.

Действие DNAT (Destination Network Address Translation) используется для преобразования адреса назначения в IP-заголовке пакета. Если пакет подпадает под критерий правила, выполняющего DNAT, то этот пакет, и все последующие пакеты из этого же потока, будут подвергнуты преобразованию адреса назначения и переданы требуемому устройству, хосту или сети.

Действие DNAT используется в тех случаях, когда необходимо предоставить доступ к ресурсам, расположенным во внутренней сети, клиентам, находящимся в

Интернете. Т.к. клиенты имеют возможность обратиться только к реальному внешнему адресу межсетевого экрана, то действие DNAT позволяет ретранслировать такой запрос путём замены адреса получателя со своего адреса на реальный внутренний адрес ресурса.

Действие DNAT может выполняться только в цепочках PREROUTING и OUTPUT таблицы *nat*, и во вложенных цепочках. Важно помнить, что вложенные цепочки, реализующие DNAT не должны вызываться из других цепочек, кроме PREROUTING и OUTPUT.

Таблица 12. Действие DNAT

Ключ	`--to-destination`
Пример	`iptables -t nat -A PREROUTING -p tcp -d 15.45.23.67 --dport 80 -j DNAT --to-destination 192.168.1.1-192.168.1.10`
Описание	Ключ --to-destination указывает, какой IP-адрес должен быть подставлен в качестве адреса назначения. В выше приведенном примере во всех пакетах, пришедших на адрес 15.45.23.67, адрес назначения будет изменен на один из адресов диапазона 192.168.1.1÷192.168.1.10. Все пакеты из одного потока будут направляться на один и тот же адрес, а для каждого нового потока будет выбираться один из адресов в указанном диапазоне случайным образом. Можно также определить единственный IP-адрес. Можно дополнительно указать порт или диапазон портов, на который (которые) будет перенаправлен трафик. Для этого после IP-адреса через двоеточие необходимо указать порт, например, --to-destination 192.168.1.1:80, либо --to-destination 192.168.1.1:80-100. Указание портов допустимо только при работе с протоколами TCP или UDP, и при наличии опции --protocol в критерии.

Действие MASQUERADE в является частным случаем SNAT, когда нецелесообразно использовать ключ --to-source. Это происходит в тех случаях, когда IP-адрес внешнему интерфейсу межсетевого экрана присваивается устройству динамически, как в распространённых в настоящее время ADSL-подключениях.

Masquerade подразумевает получение IP-адреса от заданного сетевого интерфейса вместо прямого его указания. Действие MASQUERADE имеет хорошее свойство – «забывать» параметры соединения при остановке сетевого интерфейса, т.к. при использовании динамического IP-адресом есть вероятность при следующем запуске интерфейса получить другой IP-адрес, что в любом случае приведёт к необходимости

повторной установки соединений, и поэтому нецелесообразно хранить устаревшую информацию.

Действие MASQUERADE может быть использовано вместо SNAT, даже если вы имеете постоянный IP-адрес, но его не следует считать предпочтительным, поскольку оно интенсивнее использует системные ресурсы.

Действие MASQUERADE допускается указывать только в цепочке POSTROUTING таблицы *nat*, так же как и действие SNAT.

Действие REDIRECT выполняет перенаправление пакетов и потоков на другой порт той же самой машины. К примеру, можно пакеты, поступающие на HTTP порт перенаправить на порт HTTP proxy. Действие REDIRECT очень удобно для создания «прозрачного» прокси-сервера (transparent proxy), когда компьютеры в локальной сети даже не подозревают о существовании прокси.

REDIRECT может использоваться только в цепочках PREROUTING и OUTPUT таблицы *nat*. Для действия REDIRECT предусмотрен только один ключ:

Таблица 13. Действие REDIRECT

Ключ	`--to-ports`
Пример	`iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080`
Описание	Ключ --to-ports определяет порт или диапазон портов назначения. При отстутствии ключа --to-ports перенаправления не происходит, т.е. пакет передаётся порту назначения, явно указанному в TCP- или UDP-заголовке.

Полное описание синтаксиса команды *iptables* можно найти на сайте проекта и на русскоязычных ресурсах, посвящённых этому проекту.

Использование конфигуратора Alterator

Настройка межсетевого экрана *netfilter/iptables* процедура достаточно кропотливая и трудоёмкая, особенно для начинающего администратора, и цена ошибки достаточно велика. Поэтому в образовательных учреждениях рекомендуется использовать решение, входящее в комплект ПСПО: модульный конфигуратор, построенный по технологии Alterator. Конфигуратор может быть использован как в процессе установки системы, так и для настройки уже установленной системы.

Alterator можно разделить на два слоя: *backend* и *frontend*. Первый слой реализует низкоуровневое взаимодействие с системой, второй — высокоуровневый интерфейс с пользователем (Рис. 23).

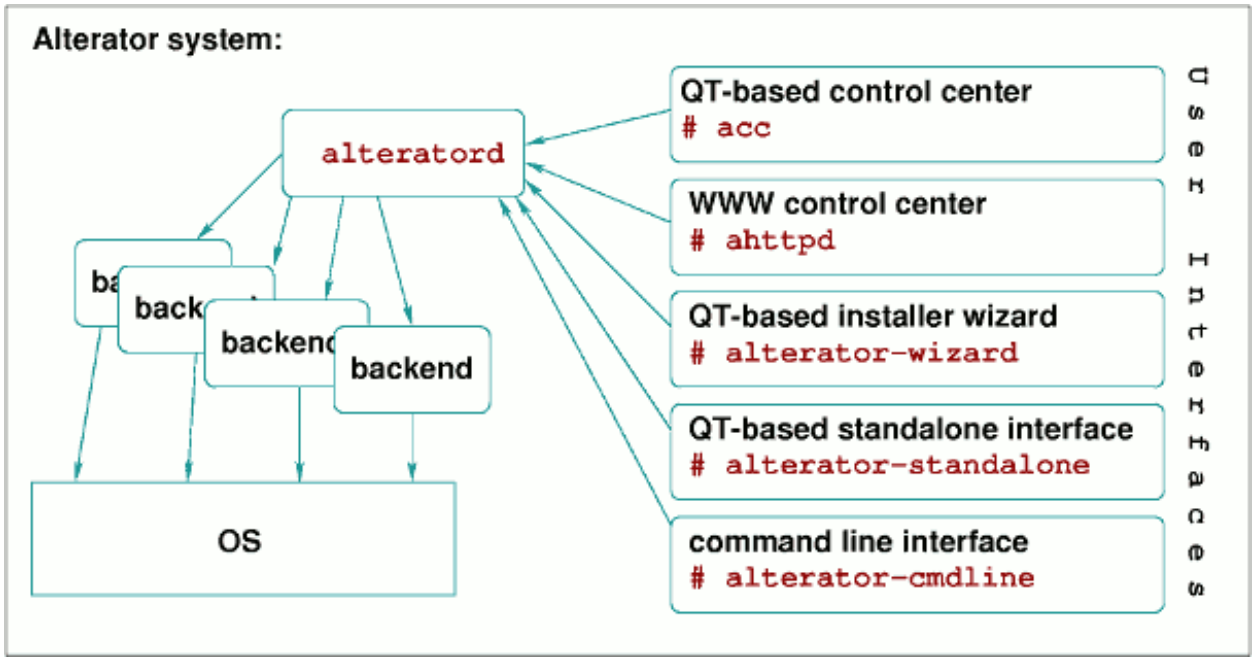

Рис. 23. Архитектура конфигуратора Alterator

Технология Alterator разрабатывалась для управления сервером по сети, поэтому было необходимо обеспечить максимально возможный уровень безопасности при передаче аутентификационных данных, т.к. подключаться к конфигуратору необходимо от имени суперпользователя. Для того, чтобы пароль не передавался по сети открытым текстом, и используется зашифрованное соединение. Необходимо отметить, что для подключения используется нестандартный порт – 8080, что также повышает защищённость интерфейса взаимодействия администратора с серверной частью «Центра управления системой», т.к. подключиться к нему случайно невозможно.

При настройке серверной части веб-интерфейса Alterator генерируется сертификат компьютера, подтверждающий его подлинность и подписанный им самим, а не в сертификационном центре. Существуют значительные различия между сертификатом, который выпущен в сертификационном центре, и сертификатом, который выписан компьютером самостоятельно (самоподписанным). Изначально неизвестно, подключение производится к доверенному сайту или недоверенному, который может заниматься перехватом вводимых паролей, то требуется проверить его подлинность. Чтобы быть полностью быть уверенными в подлинности сайта, необходимо, чтобы сертификат был подписан в сертификационном центре. Однако выпуск сертификата является платной услугой, и теоретически имеется возможность подмены корневые сертификаты. Поэтому при обнаружении самоподписанного сертификаты современные веб-браузеры генерируют соответствующее предупреждение, решение по которому должен принять пользователь.

Управление различными компонентами ОС Linux становится возможным после установки и подключения к конфигуратору соответствующих модулей. Для управления межсетевым экраном в системе должен быть установлен пакет *alterator-firewall*.

Начало работы с конфигуратором

Запустите веб-браузер и наберите в адресной строке: *https://localhost:8080*, обратите внимание, что работа с центром управления осуществляется по *https* (Рис. 24).

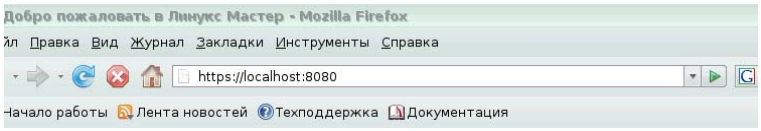

Рис. 24. Адресная строка подключения к конфигуратору

Т.к. веб-сайт конфигуратора использует самоподписанный сертификат, то появится предупреждение (Рис. 25). В рассматриваемом сценарии следует выбрать пункт «Всегда принимать этот сертификат», тогда при следующем подключении предупреждение не будет сгенерировано и соединение по протоколу *https* будет установлено автоматически.

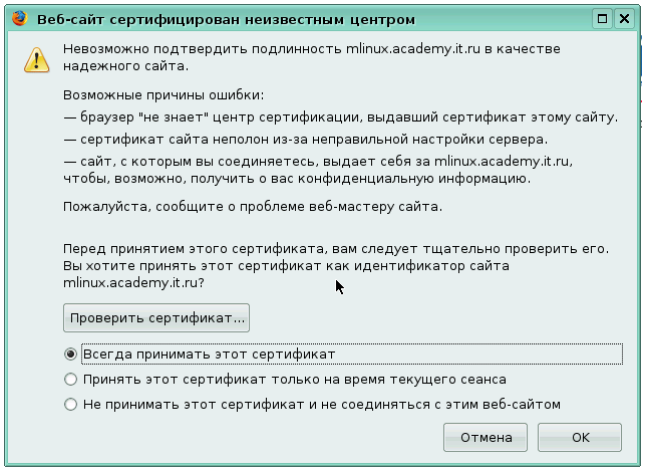

Рис. 25. Окно предупреждения Mozilla Firefox о невозможности проверить подлинность сертификата

Т.к. подключение происходит к сайту с именем localhost, а при генерации сертификата использовалось полное доменное имя компьютера, указанное в момент установки системы, то появится следующее предупреждение – о несоответствии имён в адресной строке и в сертификате, предъявленном сервером (Рис. 26). В рассматриваемом сценарии работы надо разрешить данное несоответствие.

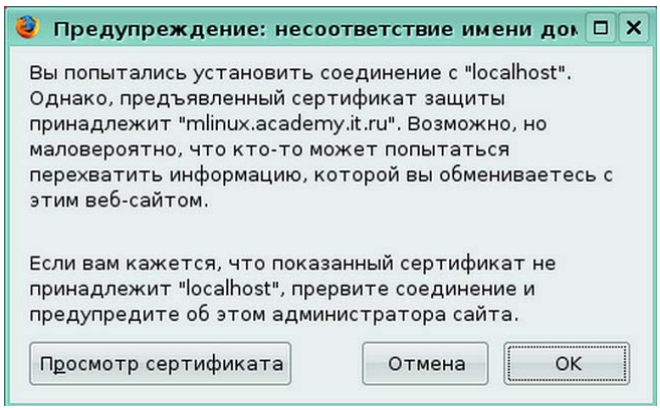

Рис. 26. Окно предупреждения Mozilla Firefox о возможном нарушении безопасности

После того, как защищенный сеанс будет установлен, появится окно с запросом аутентификационных данных, в котором следует ввести имя и пароль суперпользователя (Рис. 27).

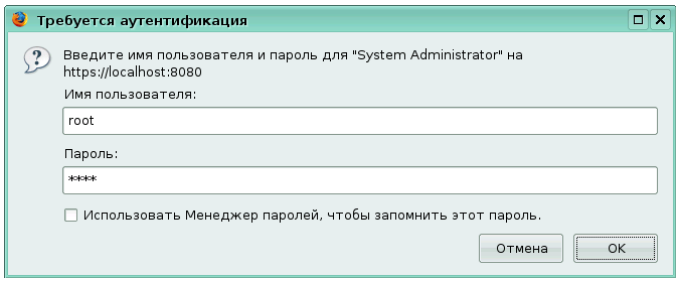

Рис. 27. Окно запроса аутентификационных данных веб-браузера Mozilla Firefox

Если аутентификационные данные введены без ошибок, появится основное окно Центра управления системой (Рис. 28).

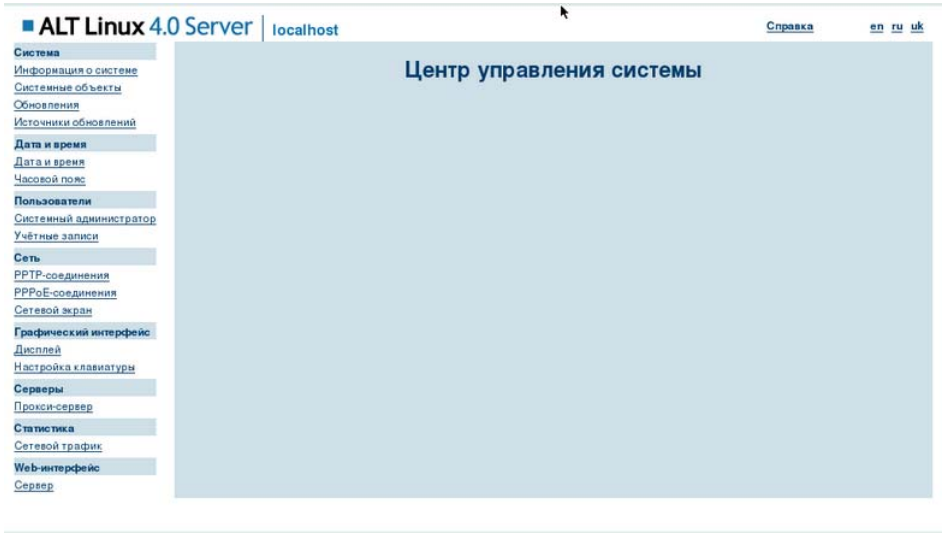

Рис. 28. Главное окно веб-интерфейса конфигуратора Alterator

Для перехода к настройкам межсетевого экрана необходимо в списке слева выбрать ссылку **«Сетевой экран»** в разделе **«Сеть»**. Отобразится окно настройки правил межсетевого экрана в упрощённом режиме (Рис. 29).

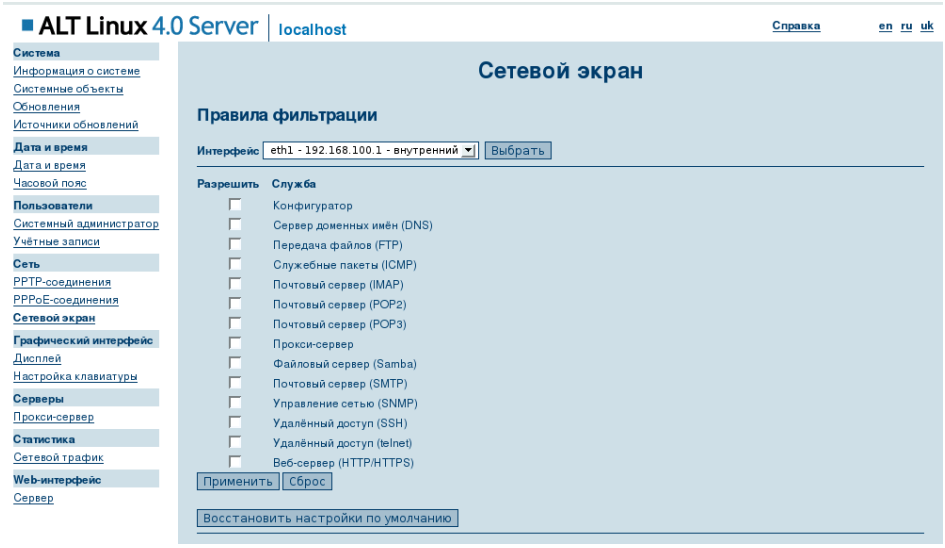

Рис. 29. Окно управления сетевым экраном веб-интерфейса конфигуратора Alterator

В этом режиме для того, чтобы разрешить компьютеру принимать запросы с использованием определённого протокола по определённому сетевому интерфейсу, необходимо сначала выбрать соответствующий интерфейс в поле Интерфейс и отметить требуемые протоколы (Рис. 30). Следует помнить, запросы с использованием неотмеченных протоколов будут автоматически отвергаться. В этом режиме невозможно управлять доступом с использованием протоколов, отсутствующих в списке, а также настраивать правила для транзитных пакетов.

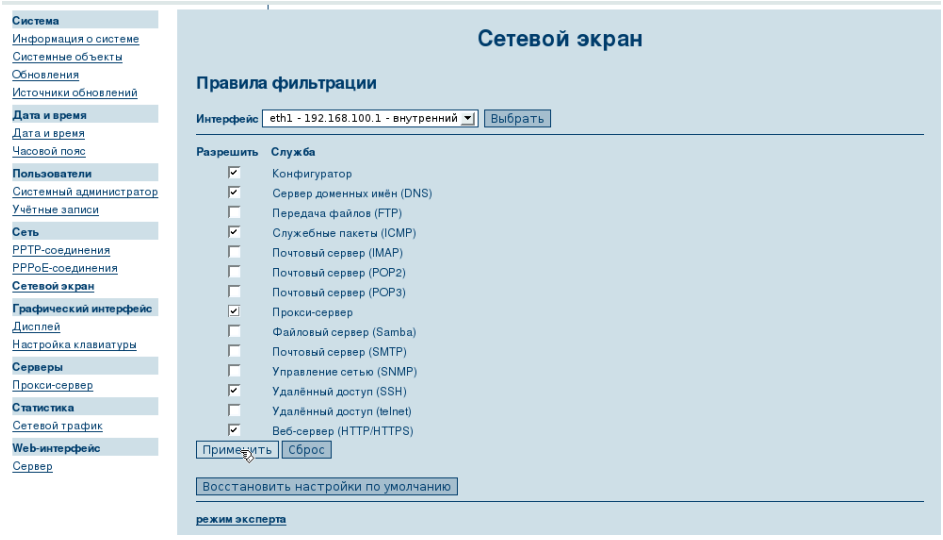

Рис. 30. Окно упрощенных настроек межсетевого экрана

Если выполнить команду просмотра списка действующих правил, то результат будет аналогичен приведённому ниже:

```
[root@mlinux squid]# iptables-save
# Generated by iptables-save v1.3.7 on Wed Sep 23 23:37:30 2009
*mangle
:PREROUTING ACCEPT [2116:680158]
:INPUT ACCEPT [2116:680158]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [2486:1278103]
:POSTROUTING ACCEPT [2534:1281807]
COMMIT
# Completed on Wed Sep 23 23:37:30 2009
# Generated by iptables-save v1.3.7 on Wed Sep 23 23:37:30 2009
*nat
:PREROUTING ACCEPT [114:12619]
:POSTROUTING ACCEPT [97:6936]
:OUTPUT ACCEPT [97:6936]
COMMIT
# Completed on Wed Sep 23 23:37:30 2009
# Generated by iptables-save v1.3.7 on Wed Sep 23 23:37:30 2009
*filter
:INPUT ACCEPT [2035:670601]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [2227:1122980]
:stdin - [0:0]
-A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 8080 -j REJECT --reject-with icmp-
port-unreachable
-A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 53 -j ACCEPT
-A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 53 -j ACCEPT
-A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 20 -j REJECT --reject-with icmp-
port-unreachable
-A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 20 -j REJECT --reject-with icmp-
port-unreachable
-A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 21 -j REJECT --reject-with icmp-
port-unreachable
-A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 21 -j REJECT --reject-with icmp-
port-unreachable
-A stdin -d 10.8.128.0 -i eth0 -p icmp -j ACCEPT
-A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 143 -j REJECT --reject-with icmp-
port-unreachable
-A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 143 -j REJECT --reject-with icmp-
port-unreachable
-A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 220 -j REJECT --reject-with icmp-
port-unreachable
-A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 220 -j REJECT --reject-with icmp-
port-unreachable
-A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 993 -j REJECT --reject-with icmp-
port-unreachable
-A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 993 -j REJECT --reject-with icmp-
port-unreachable
-A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 109 -j REJECT --reject-with icmp-
port-unreachable
```

```
        -A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 109 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 110 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 110 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 995 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 995 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 3128 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 137 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 137 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 138 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 138 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 139 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 139 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 445 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 445 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 25 -j ACCEPT
        -A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 25 -j ACCEPT
        -A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 465 -j ACCEPT
        -A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 587 -j ACCEPT
        -A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 587 -j ACCEPT
        -A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 161 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 161 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 162 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 22 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 22 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 23 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 23 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 80 -j ACCEPT
        -A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 80 -j ACCEPT
        -A stdin -d 10.8.128.0 -i eth0 -p tcp -m tcp --dport 443 -j ACCEPT
        -A stdin -d 10.8.128.0 -i eth0 -p udp -m udp --dport 443 -j ACCEPT
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 8080 -j ACCEPT
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 53 -j ACCEPT
```

```
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 53 -j ACCEPT
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 20 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 20 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 21 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 21 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p icmp -j ACCEPT
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 143 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 143 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 220 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 220 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 993 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 993 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 109 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 109 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 110 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 110 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 995 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 995 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 3128 -j ACCEPT
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 137 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 137 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 138 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 138 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 139 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 139 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 445 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 445 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 25 -j REJECT --reject-with icmp-
port-unreachable
```

```
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 25 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 465 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 587 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 587 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 161 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 161 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 162 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 22 -j ACCEPT
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 22 -j ACCEPT
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 23 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 23 -j REJECT --reject-with icmp-
port-unreachable
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 80 -j ACCEPT
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 80 -j ACCEPT
        -A stdin -d 192.168.100.1 -i eth1 -p tcp -m tcp --dport 443 -j ACCEPT
        -A stdin -d 192.168.100.1 -i eth1 -p udp -m udp --dport 443 -j ACCEPT
        COMMIT
        # Completed on Wed Sep 23 23:37:30 2009
```

Обратите внимание, что конфигуратор помещает сгенерированные правила в отдельную таблицу *stdin*, которая вложена в таблицу *filter*.

Если возможностей упрощённого режима недостаточно, то можно переключиться в режим эксперта, перейдя по соответствующей ссылке. В режиме эксперта (Рис.31) становятся доступны к полноценному редактированию таблицы и цепочки.

Рис. 31. Окно настроек межсетевого экрана для эксперта

Для редактирования цепочки необходимо перейти по ссылке редактировать напротив её названия. Будет отображено окно (Рис. 32), в котором можно управлять списком правил и выбирать действие по умолчанию.

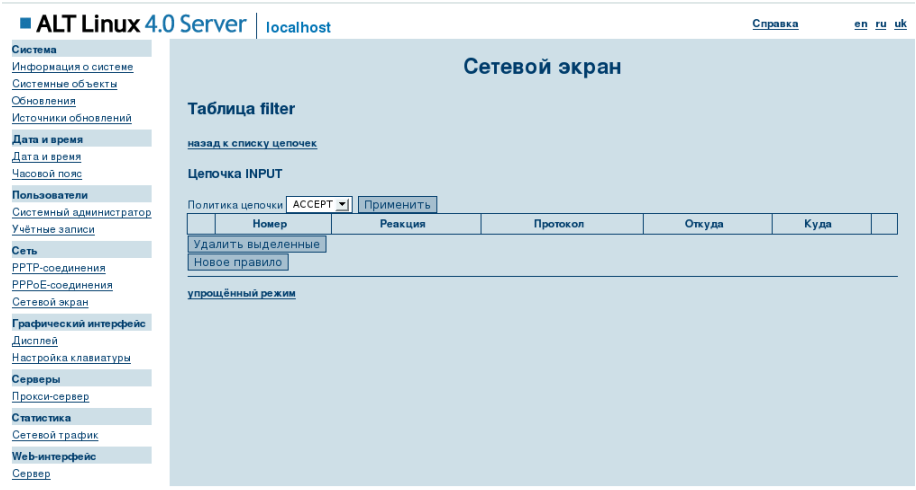

Рис. 32. Окно управления правилами в таблице в экспертном режиме

Для создания нового правила необходимо перейти по ссылке Новое правило. Будет отображено окно (Рис. 33), в котором можно указать все необходимые компоненты правила.

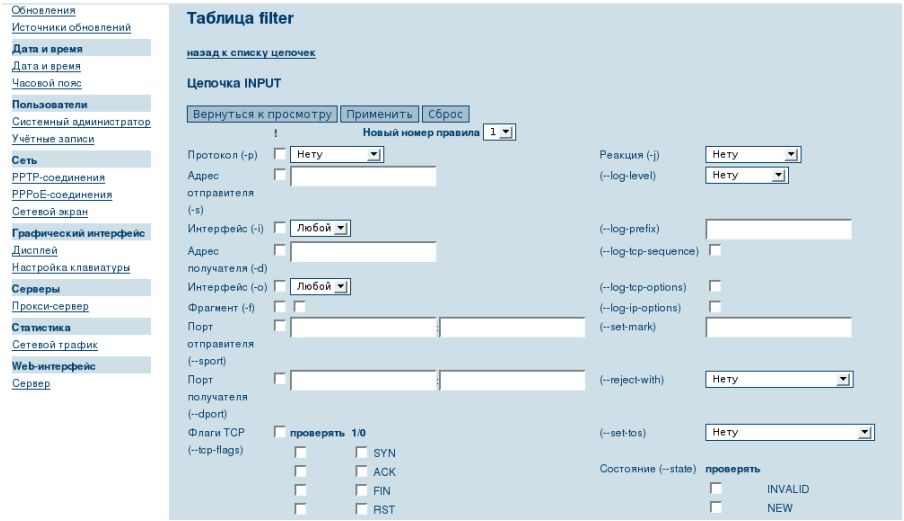

Рис. 33. Окно создания нового правила в экспертном режиме

Следует отметить, что из любого окна режима эксперта можно всегда переключиться в упрощённый режим, перейдя по соответствующей ссылке.

3. Программное обеспечение для исключения доступа учащихся к интернет-ресурсам, несовместимым с задачами их воспитания (Squid и модуль сопряжения с федеральным сервером фильтрации контента)

3.1. Кэширующий прокси-сервер Squid

Общие сведения

Squid - это высокопроизводительный кэширующий прокси-сервер для веб-клиентов, поддерживающий протоколы FTP, gopher и HTTP. В отличие от традиционных кэширующих программ, Squid все запросы выполняет как один неблокируемый процесс ввода/вывода. Squid сохраняет часто запрашиваемые данные в ОЗУ, кэширует DNS запросы, не блокируется при выполнении DNS запросов, и не кэширует неудавшиеся запросы. Также поддерживает SSL, расширенный контроль доступа и полную регистрацию запросов. Используя Internet Cache Protocol (ICP), можно создать иерархическую структуру из прокси-серверов, получая тем самым дополнительный выигрыш в использовании пропускной способности канала.

Squid состоит из:

- основной программы squid;
- программы обработки DNS запросов dnsserver;
- программы скачивания FTP данных ftpget;
- инструментов управления.

Когда Squid запускается, он запускает заданное число процессов dnsserver, каждый из которых работает самостоятельно, блокируя только DNS запросы. Таким образом, уменьшается общее время ожидания ответа DNS.

Squid берет свое начало с основанного ARPA проекта Harvest: **http://harvest.cs.colorado.edu/**. Кэширование объектов Интернет - это способ хранения запрошенных из Интернет объектов (например, данных доступных по HTTP, FTP и gopher протоколам) на сервере, находящемся ближе к запрашивающему компьютеру, нежели исходный. Браузеры могут потом использовать сервер Squid как HTTP прокси-сервер, уменьшая как время доступа, так и загрузку канала.

Squid, как и большинство подобных продуктов, распространяется по лицензии **GNU GPLv2 (General Public License Version 2).**

Squid может функционировать в следующих операционных средах:

- Linux;
- FreeBSD;
- NetBSD;

- OpenBSD;
- BSDI;
- Mac OS/X;
- OSF/Digital Unix/Tru64;
- IRIX;
- SunOS/Solaris;
- NeXTStep;
- SCO Unix
- AIX;
- HP-UX;
- Microsoft Windows Cygwin and MinGW;
- OS/2.

Установка

Получить исходные коды продукта можно с одного из зеркал официального сайта проекта, список которых доступен по адресу: **http://www.squid-cache.org/Mirrors/http-mirrors.html.** Также можно воспользоваться основными сайтами проекта: **http://www.squid-cache.org или ftp://ftp.squid-cache.org.**

Как и в случае, ранее рассмотренных продуктов, инсталляцию прокси-сервера Squid предпочтительнее осуществлять с использованием менеджера пакетов Synaptic. Название пакета: *squid*.

Общие сведения об управлении сервером Squid

Squid традиционно настраивается с помощью конфигурационного файла **squid.conf**, расположенного в директории **/etc/squid**.

Все параметры файла конфигурации разбиты на несколько групп:

- **NETWORK OPTIONS** – общие настройки параметров сетевого взаимодействия Squid-сервера с другими хостами;
- **OPTIONS WHICH AFFECT THE NEIGHBOUR SELECTION ALGORITHM –** настройка алгоритма выбора соседних серверов при использовании распределенного кэша;
- **OPTIONS WHICH AFFECT THE CACHE SIZE** – размер и параметры использования кэша Squid-сервером;
- **LOGFILE PATHNAMES AND CACHE DIRECTORY** – пути к директориям лог-файлов и кэша;

- **OPTIONS FOR EXTERNAL SUPPORT PROGRAMS** – в этой группе находятся параметры взаимодействия в внешними сетевыми сервисами (ftp, dns и проч.);
- **OPTIONS FOR TUNING THE CACHE** – параметры тонкой настройки алгоритма кэширования;
- **TIMEOUTS** – настройки различных временных задержек;
- **ACCESS CONTROLS** – параметры управления доступом;
- **ADMINISTRATIVE PARAMETERS** – учетные записи, используемые различными компонентами Squid для доступа к информации, и другие общие административные параметры;
- **OPTIONS FOR CACHE REGISTRATION SERVICES** – настройка оповещений при организации иерархической системы кэширования (используется не часто, поэтому по умолчанию все параметры закомментированы);
- **MISCELLANEOUS** – разнообразные дополнительные параметры;
- **DELAY POOL PARAMETERS** – настройка пулов задержки.

Для начинающих администраторов можно порекомендовать использовать конфигуратор Alterator. Для управления прокси-сервером Squid должен быть установлен пакет *alterator-squid*.

Настройка сервера Squid

В установочном пакете продукта находится файл с именем **quickstart**. В этом файле содержатся рекомендации по быстрой начальной настройке продукта для запуска его в эксплуатацию. Но в дальнейшем не рекомендуется использовать эти настройки как штатные.

Для начала использования прокси-сервера Squid достаточно настроить следующие параметры:

1. **http_port** - указывает номер порта, который сервер Squid будет прослушивать, ожидая запросы от клиентов, например:

```
http_port 3128
```

Данный параметр по умолчанию имеет значение «3128». По умолчанию сервер Squid ожидает запросы клиентов, поступающие с любого сетевого интерфейса. При необходимости указать конкретный сетевой интерфейс, который должен использовать сервер Squid для обслуживания клиентов данный параметр следует указать в форме:

```
http_port interface_ip_address:port
```

2. **cache_dir** - указывает местонахождения и размера локального кэша, например:

```
cache_dir ufs /var/spool/squid 100 16 256
```

Значениями данного параметра являются (по порядку): тип файловой системы, полный путь к директории кэша, максимальный размер кэша на диске в мегабайтах, количество директорий первого уровня, количество директорий второго уровня (не рекомендуется указывать значения меньшие, чем в приведённом выше примере).

3. **cache_mgr -** указывает почтовый адрес администратора кэша, например:

```
cache_mgr admin@mlinux.academy.it.ru
```

Задаваемый этим параметром адрес будет подставляться в генерируемые сервером Squid сообщения об ошибках.

4. **cache_effective_user** – указывает имя учётной записи пользователя, в контексте которой сервер Squid должен обслуживать кэш, например:

```
cache_effective_user nobody
```

Демон *squid* спроектирован для запуска от имени суперпользователя root, но далее с целью повышения уровня безопасности переключается в контекст указанного здесь пользователя для обслуживания кэша. Рекомендуется использовать учётную запись пользователя **nobody**.

5. **cache_effective_group** – указывает имя учётной записи группы, в контексте которой сервер Squid должен обслуживать кэш, например:

```
cache_effective_group nogroup
```

Данный параметр аналогичен предыдущему, указывая имя учётной записи группы для обслуживания кэша. Рекомендуется использовать учётную запись группы nogroup.

6. **visible_hostname** – указывает отображаемое имя хоста, на котором функционирует сервер Squid, например:

```
visible_hostname school-skf.edu.local
```

Задаваемое этим параметром имя будет подставляться в генерируемые сервером Squid сообщения об ошибках. Имеет смысл использовать данный параметр при создании кластера, чтобы все серверы в кластере имели одно и то же отображаемое имя. В прочих случаях используется имя хоста, сконфигурированное в настройках операционной системы.

7. **acl** – определяет списки доступа в следующем формате:

```
acl имя тип строка
```

Тип - это тип объекта, а строка - регулярное выражение. Список объектов, принадлежащих к одному и тому же типу, можно передать в файле (таким образом, поступает, например, интерфейс настройки сервера Squid в конфигураторе Alterator), в этом случае формат параметра будет следующим:

```
acl имя тип <имя файла>
```

Типы перечислены в следующей таблице:

Таблица 14. Описание значения тип параметра acl

`src ip-address/netmask`	IP-адрес подсети, где расположены клиенты
`src addr1-addr2/netmask`	диапазон IP-адресов клиентов
`dst ip-address/netmask`	IP-адрес подсети, где расположены серверы
`time [day-abbrevs] [h1:m1-h2:m2]`	время, где день это одна буква из SMTWHFA
`port`	порт или список портов
`port port1-port2`	диапазон портов
`proto`	протокол - HTTP/FTP
`method`	метод - GET/POST
`browser [-i] regexp`	сравнивается заголовок User-Agent, [-i] - игнорируется регистр букв

8. **http_access** – определяет возможность доступ к прокси-серверу через порт, указанный параметром http_port, используя протокол *http*, в следующем формате:

```
http_access allow|deny aclname
```

Значение **aclname** должно соответствовать одному из имён списков доступа, определённых параметром **acl**.

Назначение, формат и область применения других настроек можно узнать из документации по серверу Squid и многочисленных ресурсов, посвящённых этому продукту. Также все настройки подробно описаны в комментариях в файле **squid.conf**.

Минимальный конфигурационный файл сервера Squid

Конфигурационный файл сервера Squid в минимально необходимом имеет вид:

```
http_port 3128
cache_dir ufs /var/spool/squid 100 16 256
acl our_networks src 192.168.20.0/24
http_access allow our_networks
visible_hostname school.edu.local
```

Первая строка указывает на необходимость прослушивания порта 3128 на всех сетевых интерфейсах.

Вторая строка указывает, что кэш размером 100 мегабайт будет находиться в директории **/var/spool/squid**, при этом будет создано 16 поддиректорий первого уровня и по 256 поддиректорий второго уровня.

Третья строка описывает внутреннюю сеть организации. В данном примере IP-адреса всех рабочих станций принадлежат IP-сети 192.168.20.0/24.

Четвертая строка разрешает полный доступ по протоколу http из внутренней сети.

Пятая строка задает отображаемое имя прокси-сервера – **school.edu.local**.

Использование конфигуратора Alterator

Для настройки основных параметров функционирования сервера Squid можно использовать *Alterator*. Подключение к нему через веб-интерфейс осуществляется, как и в случае конфигурирования межсетевого экрана по адресу: **https://localhost:8080**.

После появления основного окна Центра управления системой (Рис.28) для настройки сервера Squid требуется перейти по ссылке Прокси-сервер. Появится окно с настройками сервера Squid (Рис. 34):

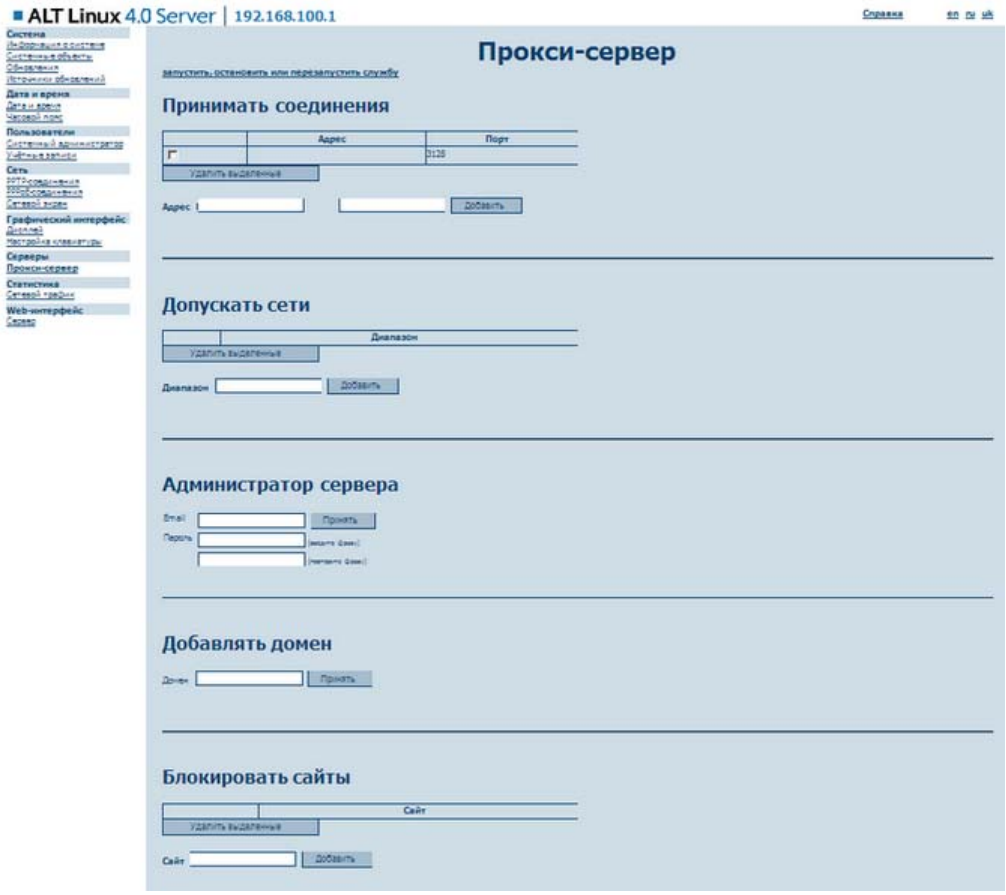

Рис. 34. Окно настроек сервера Squid

Если из окна настройки сервера Squid перейти по ссылке **Запустить, остановить или перезапустить службу**, то отобразится окно управления демоном squid (Рис..35):

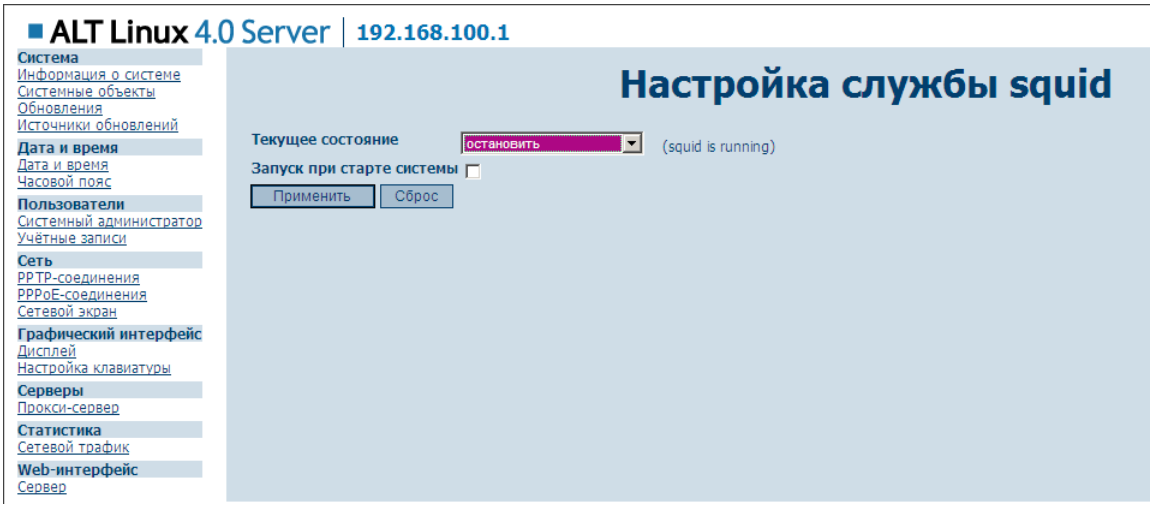

Рис. 35. Окно управления демоном squid

Защита рабочих директорий

Демон squid спроектирован для запуска от имени суперпользователя root, но для работы с кэшем он использует отдельную учетную запись с именем squid. Поэтому необходимо удостовериться в том, что права доступа к рабочим директориям настроены корректно, а при необходимости сконфигурировать их самостоятельно.

Например, файл конфигурации выглядит следующим образом:

```
cache_dir ufs /var/spool/squid 800 16 256
access_log /var/log/squid/access.log
cache_log /var/log/squid/cache.log
cache_effective_user squid
cache_effective_group squid
```

Если директория **/var/spool/squid/** отсутствует, создавать её необходимо вручную. Права доступа к ней также потребуется назначить вручную. Вот пример последовательности команд, решающих указанную задачу:

```
# mkdir /var/spool/squid
# chown -R squid /var/spool/squid
# chgrp -R squid /var/spool/squid
# chmod -R ug=rwx /var/spool/squid
# ls -ld /var/spool/squid
drwxrwx--- 18 squid squid 4096 Jul 9 19:01 /var/spool/squid
# mkdir /var/log/squid
# chown -R squid /var/log/squid
# chgrp -R squid /var/log/squid
# chmod -R ug=rwx /var/log/squid
# ls -ld /var/log/squid
drwxrwx--- 18 squid squid 4096 Jul 9 19:03 /var/log/squid
```

Надо заметить, что при инсталляции прокси-сервера Squid с использованием менеджера пакетов Synaptic, все необходимые директории будут созданы автоматически и права доступа к ним будут назначены в соответствии с требованиями документации. Иными словами, описанную выше процедуру проделывать не придётся, хотя удостовериться в том, что всё настроено правильно, необходимо.

Первый запуск Squid

По завершении редактирования конфигурационного файла, а также создания и защиты рабочих директорий, необходимо выполнить команду:

```
[root@mlinux spool]# squid -z
2009/09/24 12:10:29| Creating Swap Directories
```

Результатом выполнения данной команды является создание структуры директорий для хранения кэшируемых данных в соответствии со значением параметра **cache_dir** в конфигурационном файле. Необходимо заметить, что данную операцию требуется выполнить вручную независимо от способа инсталляции сервера Squid.

Теперь необходимо запустить squid и проверить, что запуск прошел успешно:

```
[root@mlinux spool]# squid -D
[root@mlinux spool]# netstat -an | grep 3128
tcp            0          0  0.0.0.0:3128                        0.0.0.0:*
LISTEN
[root@mlinux spool]# ps aux|grep squid
root        8011   0.0   0.0    1968   1020 ?          Ss    11:12    0:00
/bin/sh /usr/lib/alterator/backend3/template-squid
root        8038   0.0   0.1    2124   1116 ?          Ss    11:12    0:00
/bin/awk -f /usr/lib/alterator/backend3/squid
root       9700  0.0  0.1    6716  1152 ?        Ss    12:13    0:00 squid
-D
squid      9702   1.0   0.5    8952   5908 ?          S     12:13    0:00
(squid) -D
squid      9703   0.0   0.0    1384    300 ?          Ss    12:13    0:00
(unlinkd)
squid      9704   0.0   0.0    1880    612 ?          Ss    12:13    0:00
(pinger)
root       9710  0.0  0.0    1988    636 pts/3    R+    12:13    0:00 grep
squid
[root@mlinux spool]# tail -f /var/log/messages
Sep 24 12:13:12 mlinux squid[9700]: Squid Parent: child process 9702
started
```

Полезные ключи командной строки

- **-a** *<номер порта>* - позволяет динамически переопределить номер прослушиваемого порта;
- **-f** *<имя файла конфигурации>* - позволяет указать серверу Squid на необходимость использования альтернативного файла конфигурации;
- **-h** – запрос полного списка ключей командной строки;
- **-k** – посылка сигнала работающему серверу Squid:

✓ reconfigure – заново прочитать файл конфигурации и применить указанные в нем параметры;

✓ shutdown – завершить работу, обслужив все запросы в очереди;

✓ interrupt – прервать работу;

✓ kill – принудительное немедленное завершение работы;

✓ parse - проверить синтаксис squid.conf;

- **-v** – вывести версию и параметры сборки;
- **-z** - создать дисковый кэш при первом запуске;
- **-D** - не запускать DNS-тест при старте.

ACL (списки управления доступом)

Управление доступом является наиболее важной функцией Squid. Если не настроить должным образом механизм управления доступом, система будет открыта для любого доступа извне. Даже если отсутствует необходимость изменять настройки по умолчанию, заданные в конфигурационном файле **squid.conf**, требуется настроить раздел параметров управления доступом.

Управление доступом включает в себя две составляющие, которые комбинируются для того, чтобы разрешить или запретить доступ к самому серверу Squid или определенным ресурсам (URL). Первой частью системы управления доступом являются списки управления доступом ACL (Access Control List). Запись, определяющая список управления доступом, начинается со служебного слова **acl,** за которым следует уникальное имя (два разных списка acl не могут обладать одним и тем же именем). После имени следует спецификация, за которой указывается один или несколько аргументов. Аргументы спецификации логически складываются, и если они являются IP-адресами, вместе с ними необходимо указывать сетевую маску.

В качестве аргументов можно использовать имена сетевых узлов (при этом подразумевается сетевой узел, обладающий указанным и никаким другим именем), однако делать это следует с осторожностью, так как если требуется заблокировать доступ к узлу с

указанным именем, доступ к этому узлу можно будет получить, указав другое имя или IP-адрес. Вообще говоря, параметр **acl** можно рассматривать как спецификацию, которая будет использована на этапе принятия решения о разрешении или запрете доступа.

После того как будут сформированы списки acl, их имена можно использовать в следующих далее списках прав *доступа ARL* (Access Rights List). Список прав доступа — это группа прав на доступ к ресурсу (в качестве ресурса выступает и сам сервер Squid) по определенному протоколу. За идентификатором протокола следует одно из ключевых слов — **allow** (разрешить) или **deny** (запретить), а затем перечень списков ACL, которые логически перемножаются. Иными словами, для того чтобы список ARL был использован при обработке поступившего запроса, необходимо, чтобы параметры этого запроса удовлетворяли условиям всех ACL, перечисленых в ARL.

Для каждого протокола, по которому произошло обращение, Squid ищет список ARL, параметрам которого соответствует запрос, и как только такой список ARL обнаруживается, Squid прекращает дальнейший поиск и предпринимает действие, указанное в ARL: либо allow - разрешить, либо deny - запретить. Если для запроса не удалось найти ни одного подходящего ARL, Squid применяет действие противоположное указанному в последнем правиле для протокола, по которому пришёл запрос. Такое поведение может привести к непредсказуемым результатам, поэтому на самой последней позиции следует расположить ARL, который будет соответствовать всем соединениям, для которых не удалось подобрать соответствия ранее. Формат такого ARL, как правило, следующий:

```
http_access deny all
```

Также следует отметить, что для сети образовательного учреждения при построении ARL потребуется использовать только протокол доступа к прокси-серверу через порт, указанный параметром http_port. Прочие возможности, скорее всего, не понадобятся.

Примеры построения списков доступа

В рассматриваемых ниже примерах номера строк используются для удобства ссылок на них при описании их назначения. В конфигурационном файле номера строк отсутствуют.

Пример 1. Наиболее простой набор правил:

```
1. acl all src 0.0.0.0/0.0.0.0
2. http_access allow all
```

Данный набор списков ACL и ARL является наиболее простым из всех возможных. Он разрешает абсолютно все, т.е. полный доступ для всех: в первой строке описываются

абсолютно все возможные IP-адреса, во второй строке для всех этих адресов разрешается доступ через Squid по протоколу http. Очевидно, что приведённый набор правил не может быть признан удовлетворительным, т.к. любой желающий в сети Интернет сможет использовать такой сервер Squid для доступа к веб-сайтам. Существуют программы, которые специально предназначены для обнаружения таких открытых прокси-серверов.

Чтобы избежать этого, следует ограничить доступ к прокси-серверу.

Пример 2. Расширенная базовая конфигурация Squid, более соответствующая действительности:

```
1. acl all src 0.0.0.0/0.0.0.0
2. acl manager proto cache_object
3. acl allowed_hosts src 192.168.0.0/255.255.0.0 127.0.0.0/255.0.0.0
4. http_access allow allowed_hosts
5. http_access deny all
```

Первая строка полностью совпадает с предыдущим примером, описывая все доступные IP-сети.

Вторая строка указывает на протокол доступа к кэшу Squid.

В третьей строке идентифицируются узлы, которым требуется разрешить доступ. В этой строке к узлам сети 192.168.0.0/16 добавлен локальный узел localhost, который в большинстве примеров идентифицируется IP-адресом 127.0.0.1, однако на самом деле для узла localhost можно использовать любой IP-адрес из диапазона сети 127.0.0.0/8.

Четвертая строка разрешает группе узлов allowed_hosts обмен данными по протоколу http.

Пятая строка запрещает доступ по протоколу http для всех остальных узлов. Таким образом, ни один узел сети Интернет не сможет использовать такой сервер Squid в качестве открытого прокси-сервера

Пример 3. Списки доступа, сформированные в конфигурационном файле сервера Squid по умолчанию в одной из версий (этот список не обязательно будет совпадать со списком, сформированным при установке ПСПО):

```
1. acl all src 0.0.0.0/0.0.0.0
2. acl manager proto cache_object
3. acl localhost src 127.0.0.1/255.255.255.255
4. acl SSL_ports port 443 563
5. acl Safe_ports port 80 21 443 563 70 210 1025-65535
6. acl CONNECT method CONNECT
7. http_access allow manager localhost
8. http_access deny manager
9. http_access deny !Safe_ports
10.      http_access deny CONNECT !SSL_ports
```

```
11.      http_access deny all
12.      miss_access allow all
```

Первая строка совпадает с первой строкой из предыдущих примеров. В ней определяется группа узлов all, в которую входят все допустимые IP-адреса.

Вторая строка определяет протокол, который используется для доступа к кэшу Squid.

В третьей строке определяется локальный узел localhost - его адрес 127.0.0.1.

Четвертая строка определяет номера портов, которые могут использоваться протоколом SSL/TLS.

Пятая строка определяет номера прочих портов, допустимых к использованию.

Шестая строка определяет метод для семейства протоколов SSL/TLS.

Седьмая строка разрешает обращение к кэшу Squid только для локального узла. Другими словами, данный сервер не будет обеспечивать доступ к своему кэшу для соседних кэширующих систем (это хороший подход для сетей, в которых используется только один сервер Squid).

Восьмая строка запрещает доступ к кэшу для всех систем, для которых доступ еще не был разрешен (на текущий момент доступ к кэшу разрешен только для одной системы — localhost).

Девятая строка запрещает использование каких-либо портов, которые не входят в группу Safe_ports (группа портов Safe_ports определена ранее при помощи соответствующей записи acl). Таким образом, Squid не сможет обслужить запросы для telnet (порт 23), sendmail (порт 25) и т. д.

Десятая строка запрещает новые входящие соединения к каким-либо портам, не являющимся портами SSL. Вы не сможете запустить какой-либо сервер, принимающий соединения из Интернета, если эти соединения не являются соединениями SSL.

Одиннадцатая строка запрещает все остальные виды доступа.

Двенадцатая строка разрешает всем желающим доступ к кэшу Squid на данном сервере (возможно, это не самая лучшая идея).

Настройка клиентов

Для использования сервера Squid клиентскими машинами требуется настроить на них параметры подключения к серверу.

Ниже приведены примеры настроек для веб-браузера Internet Explorer (Рис. 36) и веб-браузера Mozilla Firefox (Рис.37).

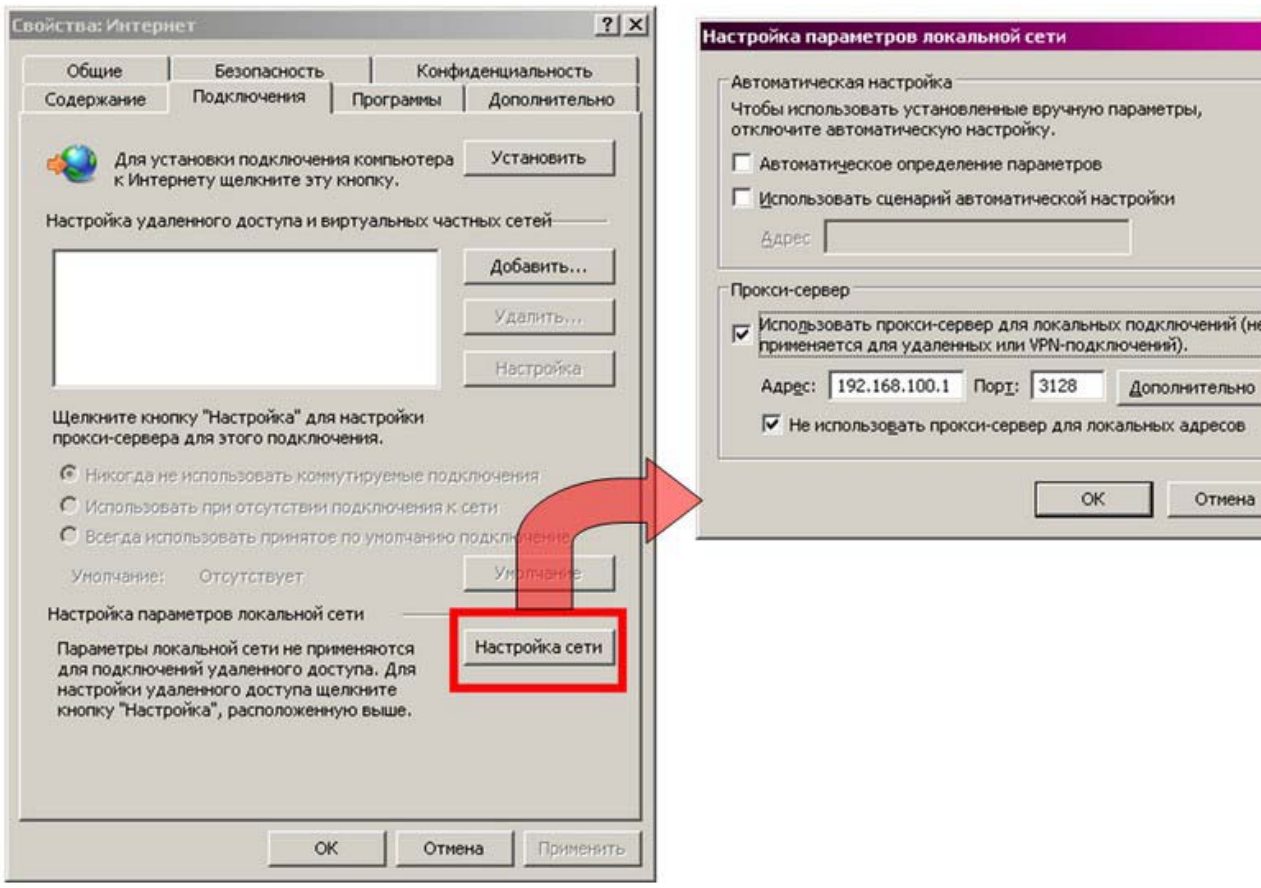

Рис. 36. Настройка параметров прокси-сервера в браузере Internet Explorer

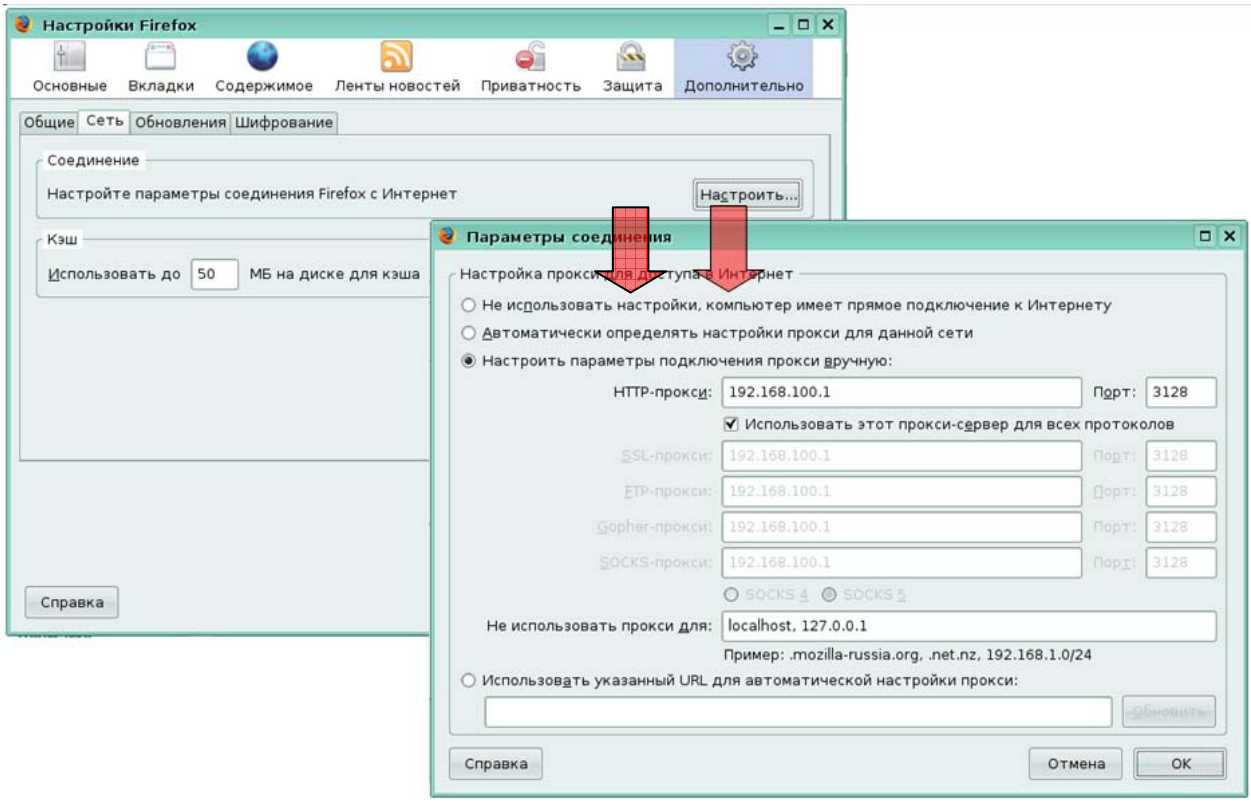

Рис. 37. Настройка параметров прокси-сервера в браузере Mozilla Firefox

3.2. Модуль сопряжения с федеральным сервером фильтрации контента

Контроль над использованием интернет-трафика является одной из основных задач системного администратора в любой организации. Для образовательных учреждений особую остроту приобретает вопрос контроля и регулирования обращений учащихся к Интернет-ресурсам, несовместимым с задачами их воспитания.

Проблему нагрузки на внешний канал решает использование кэширующего прокси-сервера, но остается проблема с динамически генерируемыми ссылками, которые используются в форумах, чатах и в баннерных сетях. С другой стороны, стоит запретить учащимся посещать страницы с определенным содержанием, а также скачивать музыку, фильмы и другие файлы, что может нарушать авторские права их создателей и/или владельцев. Использование Access Control List (ACL) не всегда может решить эту проблему, к тому же этот процесс достаточно трудоемкий (примеры для конфигурирования прокси-сервера Squid можно найти на сайтах из приведенного списка литературы).

Выходом из сложившейся ситуации является фильтрация по содержимому. Для образовательных учреждениях различного уровня предлагается к использованию Система Контентной Фильтрации (СКФ).

Общие сведения

Система Контентной Фильтрации (СКФ) предназначена для использования в образовательных учреждениях Российской Федерации различного уровня с целью:

- ограничения доступа к Интернет-ресурсам, содержание которых несовместимо с задачами воспитания и образования или нежелательно;
- сбора статистических сведений об использовании ресурсов Интернет в образовательном учреждении.

Для выполнения вышеперечисленных задач СКФ взаимодействует с программным комплексом Сервер тематической категоризации (СТК), служащим для выполнения задачи категоризации Интернет-ресурсов (Рис. 38).

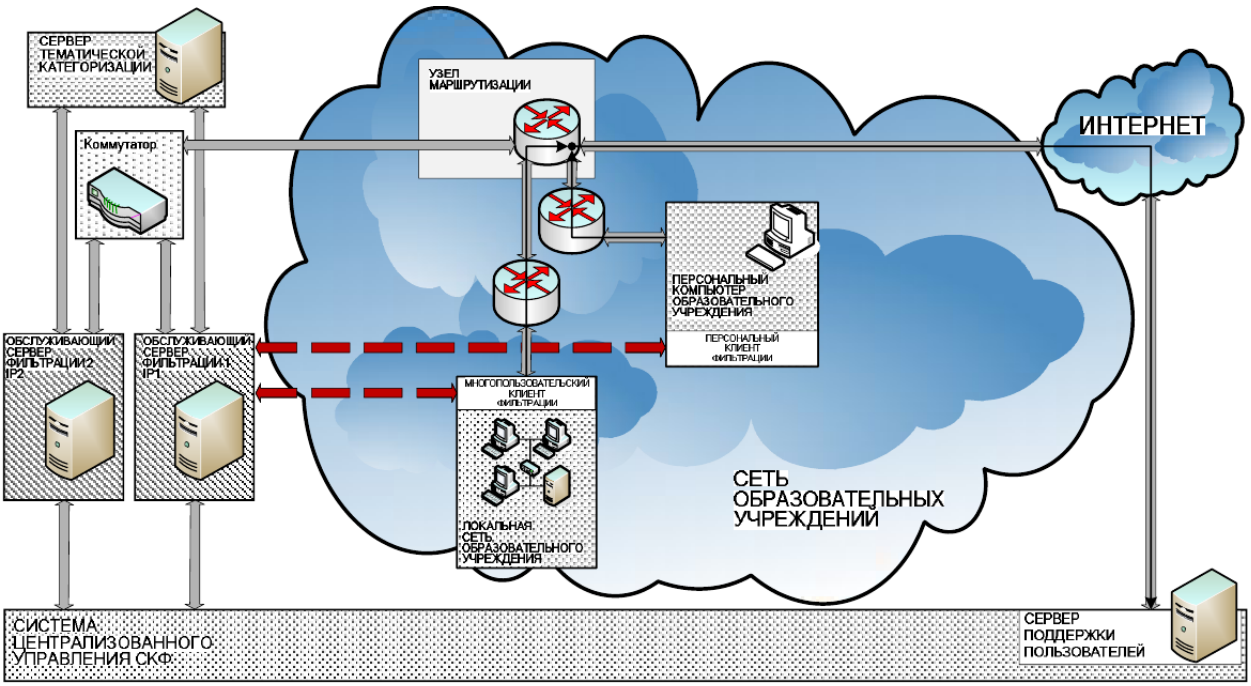

Рис. 38. Архитектура Системы Контентной Фильтрации

Система Контентной Фильтрации предназначена для выполнения следующих задач:

- идентификации пользователя клиента СКФ;
- назначения прав доступа пользователей клиента СКФ к Интернет-ресурсам;
- фильтрации обращений пользователей клиента СКФ к Интернет-ресурсам;
- накопления статистики обращений к Интернет-ресурсам с отправкой сведений по некатегоризированным ресурсам программному комплексу СТК;

обновления системной информации.

3.2.1. Принцип функционирования МКФ

Многопользовательский клиент фильтрации (МКФ) – это технологическое решение, основной задачей которого является предоставление сервиса управления доступом пользователей к ресурсам Интернет. Система контентной фильтрации (СКФ), частью которой является МКФ, настроена на русскоязычный контент и использует эффективные алгоритмы URL фильтрации. СКФ отвечает за управление доступом пользователей к ресурсам Интернет в зависимости от категории запрашиваемых ресурсов и принятой организационной политики.

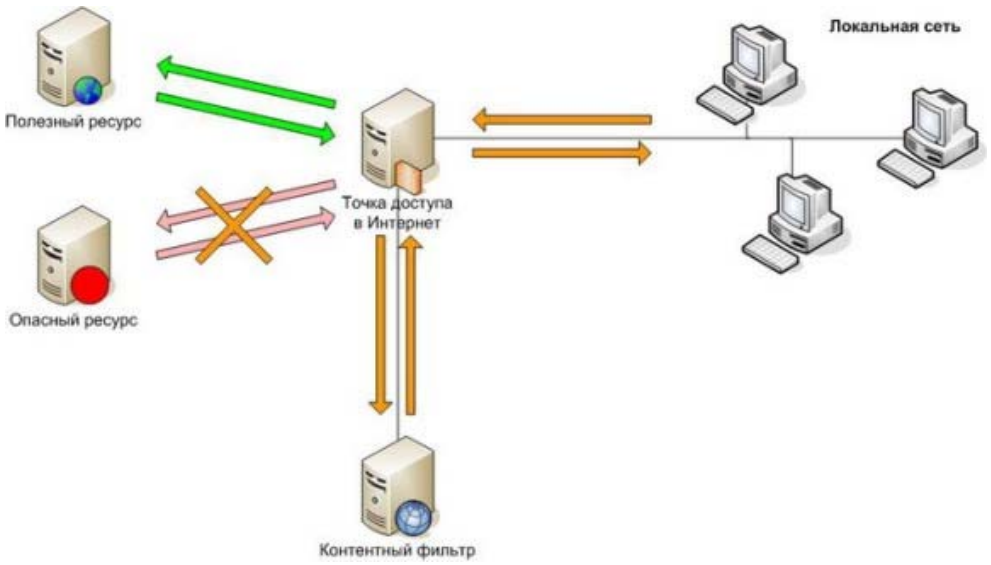

Рис. 39. Схема обработки запроса к категоризированному ресурсу

МКФ проверяет запросы пользователей на принадлежность к конкретной категории. В случае запрещенной категории запрос к ресурсу блокируется. Одновременно запрос к ресурсу, не относящемуся к запрещенной категории будет одобрен контентным фильтром, как показано на рисунке 39.

В случае если запрашиваемый Интернет-ресурс на данный момент не содержится в базе данных, то пользователь получает доступ к этому ресурсу, а контентный фильтр передает новую ссылку в центральную базу. Там ресурс категоризируется, и соответствующая информация немедленно поступает в базу ресурсов контентного фильтра (Рис. 40).

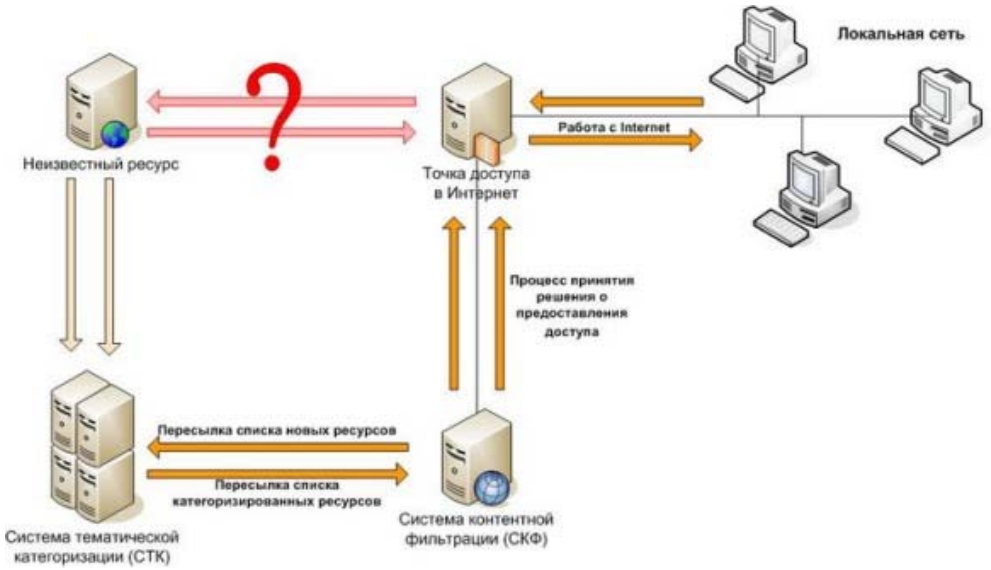

Рис..40. Схема обработки запроса к некатегоризированному ресурсу

Еще одной отличительной чертой МКФ является гибкость и масштабируемость системы. Улучшенный механизм мониторинга, сбора и представления статистических данных по всей организации реализован с помощью специализированного модуля сбора и

обработки статистики обращений пользователей, собираемых локальными контентными фильтрами. Принцип работы вышеуказанного модуля показан на примере связки Региональный департамент образования, Комитеты образования и групп подведомственных школ, представленном на рисунке 41.

Рис. 41. Схема работы механизма сбора и обработки статистики

Возможности МКФ

МКФ предоставляет организациям, его использующим следующие возможности:

- работа с ресурсами на русском и иностранных языках: наравне с отличной способностью анализировать русскоязычный контент, МКФ также работает с другими основными языками мира;
- гибкие правила фильтрации МКФ предоставляют широкие возможности для настройки правил фильтрации: настройка правил доступа для отдельного пользователя или групп пользователей, управление доступом к ресурсам на уровне категорий и отдельных сайтов, «чёрные» и «белые» списки, календарь применения правил доступа; регулируемый уровень управления доступом: мониторинг, предупреждение о запросе к опасному ресурсу, блокирование опасного ресурса.
- отчеты о доступе пользователей: различные типы отчетов позволяют получить детальную статистику об использовании Интернет от детализированного отчета по сайту до общих отчетов об активности использования Интернет; модуль сбора и обработки статистики. МКФ представляет уникальный инструмент для сбора и представления статистики в больших территориально-распределенных организациях с помощью специализированного модуля сбора и обработки статистики обращений пользователей, собираемых локальными контентными фильтрами, установленными в удаленных подразделениях организации.

- автоматическое обновление базы URL: МКФ избавляет от дополнительной работы по обслуживанию, производя автоматическое обновление базы URL, списка категорий и других параметров.

Реализация МКФ в комплекте ПСПО

МКФ на сервере образовательного учреждения реализован в виде набора взаимодействующих друг с другом модулей:

- фильтр реализован в виде двух серверов: icap-сервера и прокси-сервера Squid, который передаёт все запросы от клиентов на обработку icap-серверу с посредством протокола icap (Internet Content Adaptation Protocol), описанного в документе RFC3507;
- модуль принятия решения icap-сервера реализован: в виде модуля host2cat - ядра системы фильтрации, который обращается к заданному ОСФ (обслуживающему серверу фильтрации) для получения категории ресурса и модуля memcached, предназначенного для оперативного кэширования ответов;
- базы данных программного комплекса МКФ, который реализован модулем opendbx и представляет собой СУБД SQL, работающую локально на том же компьютере и закрытую для внешних взаимодействий.

Все указанные модули, включая базу данных, должны размещаются на одной ПЭВМ.

Взаимодействие между модулями программного комплекса МКФ осуществляется следующим образом:

- прокси-сервер Squid принимает запрос от клиента;
- посредством протокола icap прокси-сервер Squid передаёт запрос icap-серверу;
- icap-сервер посредством протокола icap передаёт запрос дальше модулю принятия решения, а также, используя стандартные интерфейсы операционной системы, антивирусу ClamAV;
- модуль принятия решения обращается к оперативному кэшу модуля memcached для определения категории, к которой необходимо отнести запрашиваемый Интернет-ресурс, если в кэше такая информация отсутствует, то сконфигурированному в его настройках ОСФ;
- модуль принятия решения принимает от ОСФ категорию, к которой отнесён запрашиваемый ресурс, и передаёт ответ модулю memcached для оперативного кэширования;

- модуль принятия решения обращается к SQL-серверу, обслуживающему базу данных программного комплекса, чтобы определить допустимость обращаться к данной категории ресурсов клиенту, сгенерировавшему запрос;
- модуль принятия решения возвращает ответ icap-серверу;
- icap-сервер, суммируя ответы, полученные от модуля принятия решения и антивируса ClamAV, принимает окончательное решение по запросу и возвращает его серверу Squid;
- сервер Squid реализует необходимое действие, опираясь на полученный от icap-сервера ответ.

Взаимодействие программного комплекса МКФ с серверами ОСФ не требует постоянного соединения и оно устанавливается с помощью протокола TCP/IP по мере необходимости следующими программными модулями:

- модулем принятия решений – ядром системы фильтрации;
- модулем сервиса поддержки МКФ.

Периодическое взаимодействие необходимо для получения списка категорий ресурсов, а также описаний предопределённых ролей пользователей.

В настоящее время существуют следующие категории ресурсов:

Таблица 15. Категории ресурсов, определённые в ОСФ

1	Пропаганда войны, разжигание ненависти и вражды, пропаганда порнографии и антиобщественного поведения
2	Экстремизм / Злоупотребление свободой СМИ
3	Наркотические средства / Злоупотребление свободой СМИ
4	Информация с ограниченным доступом / Злоупотребление свободой СМИ
5	Скрытое воздействие / Злоупотребление свободой СМИ
6	Экстремистские материалы или экстремистская деятельность (экстремизм)
7	Вредоносные программы
8	Преступления
9	Ненадлежащая реклама
10	Информация с ограниченным доступом
11	Алкоголь
12	Баннеры и рекламные программы
13	Вождение и автомобили

14	Досуг и развлечения
15	Здоровье и медицина
16	Компьютерные игры
17	Корпоративные сайты, Интернет -представительства негосударственных учреждений
18	Личная и немодерируемая информация
19	Отправка SMS с использованием Интернет-ресурсов
20	Модерируемые доски объявлений
21	Нелегальная помощь школьникам и студентам
22	Неприличный и грубый юмор
23	Нижнее белье, купальники
24	Обеспечение анонимности пользователя, обход контентных фильтров
25	Онлайн - казино и тотализаторы
26	Платные сайты
27	Поиск работы, резюме, вакансии
28	Поисковые системы
29	Религии и атеизм
30	Системы поиска изображений
31	СМИ
32	Табак, реклама табака, пропаганда потребления табака
33	Торговля и реклама
34	Убийства, насилие
35	Чаты

3.2.2. Установка компонентов МКФ

Порядок установки СКФ следующий:

- в менеджере пакетов Synaptic подключаем репозитарии:
  - ✓ ftp://ftp.altlinux.ru/pub/distributions/ALTLinux/4.0/school/branch i586 classic;
  - ✓ ftp://ftp.altlinux.ru/pub/distributions/ALTLinux/4.0/school/branch noarch classic;
- устанавливаем необходимые пакеты, если они ещё не установлены:
  - ✓ squid;
  - ✓ memcached;
  - ✓ host2cat;
  - ✓ c-icap;
  - ✓ opendbx.

Следующим шагом необходимо убедиться в активности всех необходимых служб. Например, выполнив следующую последовательность команд:

```
[root@mlinux ~]# /sbin/service memcached restart
Stopping memcached service: Stopping memcached service:                [
DONE ]
Starting start-memcached service:                                      [
DONE ]
[root@mlinux ~]# /sbin/service c-icap restart
Service           c-icap           is           not           running.
[PASSED]
Starting c-icap service:                                               [
DONE ]
[root@mlinux ~]# /sbin/service host2cat restart
Stopping host2cat service:                                             [
DONE ]
Starting host2cat service:                                             [
DONE ]
[root@mlinux ~]# /sbin/service httpd2 restart
Service           httpd2           is           not           running.
[PASSED]
Checking configuration sanity for httpd2:  Syntax OK
                                                                       [
DONE ]
Starting httpd2 service:                                               [
DONE ]
[root@mlinux ~]# /sbin/service squid restart
Service           squid           is           not           running.
[PASSED]
Starting squid service:                                                [
DONE ]
```

3.2.3. Настройка МКФ

Основные параметры функционирования МКФ принадлежат icap-серверу и находятся в файле **/etc/c-icap.conf.** Параметры и формат файла описаны в документации проекта **c-icap**.

Для подключения МКФ к icap-серверу в конфигурационном файле необходимо указать имя подключаемого модуля:

```
Service url_filter_module srv_url_filter.so
```

Специфические параметры, необходимые для настройки функционирования МКФ, описаны ниже:

1. **url_filter.MemcachedServers** – указывает имя хоста и порт подключения, на котором запущен memcached сервер, например:

```
url_filter.MemcachedServers "memcached:11211"
```

Для большей эффективности (при наличии ресурсов) возможно использование нескольких memcached серверов. В этом случае они перечисляются через запятую.

2. **url_filter.DBName** – указывает полное имя (полный путь и имя файла) файла базы данных с настройками МКФ, например:

```
url_filter.DBName "/var/cache/host2cat/filter.db"
```

Этот файл должен существовать в системе и быть доступен на чтение/запись. Формирование этого файла возможно при помощи скрипта **/usr/local/share/c-icap/initdb.pl.**

3. url_filter.Host2CatServer – указывает хост и порт, используемый СКФ для приёма запросов, например:

```
url_filter.Host2CatServer "127.0.0.1:6666"
```

4. **url_filter.RedirectUrl** - указывает URL ресурса, на который происходит перенаправление запроса пользователя в случаях, если именно данное действие определено в политике, например:

```
url_filter.RedirectUrl "http://www.content-filtering.ru/filter.html"
```

Следующие параметры определяют действие фильтра в случае сбоев и в неопределённых ситуациях. Возможные значения параметров:

- 0 - разрешает обслуживание запроса;
- 1 - сбрасывает запрос, объявляя ресурс недоступным;
- 2 - перенаправляет запрос на URL задаваемый параметром **url_filter.RedirectUrl;**

5. **url_filter.DBOpenErrorPolicy** - определяет действие фильтра при ошибке открытия файла БД, хранящего политику доступа.

6. **url_filter.DBQueryErrorPolicy** - определяет действие фильтра, если при запросе к БД, хранящей политику доступа, произошла ошибка.

7. **url_filter.UnknownRolePolicy** - определяет действие фильтра, если указанная роль не описана в политиках.

8. **url_filter.MemcachedConnectErrorPolicy** - определяет действие фильтра, если произошла ошибка доступа к базе данных memcached.

9. **url_filter.MemcachedMissPolicy** - определяет действие фильтра, если соответствующие данные отсутствуют в базе данных **memcached**.

10. **url_filter.EmptyCategoryPolicy** - определяет действие фильтра, если категоризирующий сервер возвращает пустую запись.

11. **url_filter.CategoryNotFoundPolicy** -определяет действие фильтра, если категоризирующий сервер не находит соответствующую запись.

Политика доступа, пользователи и роли

Политика доступа пользователя к ресурсам определяется на основе ролей и описания возможностей доступа к запрашиваемому ресурсу для каждой роли. Пользователь в сети идентифицируется по имени, используемому при авторизации на сервере Squid (этот функционал настраивается в файле **squid.conf**), и/или IP-адресом компьютера пользователя. Из имени и адреса формируется полное имя пользователя, например: `teacher@10.12.19.5` - это пользователь **teacher**, работающий на компьютере с IP-адресом 10.12.19.5.

Вся информация о правах доступа хранится в файле базы данных, полный путь к которому определяется параметром **url_filter.DBName** в файле конфигурации. Для формирования БД используется программа **/usr/lib/host2cat/initdb.pl**. Для тестирования сформированных политик используется **/usr/lib/host2cat/check.pl**.

Описание программы формирования БД политик

Программа **/usr/lib/host2cat/initdb.pl** позволяет сформировать БД политик. Для формирования БД политик используются следующие конфигурационные файлы:

- *users* - содержит описание соответствия пользователей назначаемым ролям;
- *custom_roles, generic_roles* - содержат описание действий фильтра для различных ролей.

Синтаксис, используемый для запуска программы, следующий:

```
initdb.pl [-h] [-v] [-c] [-w] [-d <каталог>] <имя файла БД>
```

Параметры запуска имеют следующее значение:

- **-h** - выводит на экран описание параметров запуска;
- **-v** - выводит на экран список SQL-команд, сформированных в результате работы программы;
- **-c** – проверка непротиворечивости описаний в конфигурационных файлах, БД при этом не создается;
- **-w** - запрещает создавать БД, если в результате работы программы было выведено предупреждающее сообщение;
- **-d <каталог>** - имя каталога в котором располагаются файлы users, custom_roles и generic_roles.

Файл *users* содержит записи, описывающие соответствие идентификатора пользователя той или иной роли. Пустые строки, строки из пробелов и знаков табуляции,

а также строки, начинающиеся со знака «#», считаются комментариями и в процессе обработки игнорируются. Формат записей файла users следующий:

```
[<имя пользователя>]@[<IP адрес>[/<маска сети>]]     <роль>
```

- **<имя пользователя>** - имя учетной записи пользователя, предъявляемое для аутентификации; допускаются только символы латинского алфавита и цифры, цифра не должна быть первым символом; если этот параметр не задан, то указанная в данной записи роль сопоставляется с любым пользователем;
- **<IP адрес>** - IP-адрес отдельной рабочей станции или целой IP-сети, если этот параметр не задан, то он принимается равным (вместе со следующим параметром - <маска сети>) «0.0.0.0/0», т.е. все возможные IP-адреса и сети;
- **<маска сети>** - целое число от 0 до 32, указывающее количество старших бит IP-адреса, которые трактуются как номер IP-сети, если этот параметр не задан, то он принимается равным «/32», т.е. один-единственный компьютер.

Если характеристики источника запроса – пользователя соответствуют нескольким ролям из списка ролей, то выбирается роль, наиболее полно соответствующая пользователю. Пример файла **users**:

```
user@                          student5
@192.168.0.0/24                student4
user@192.168.0.0/24            student3
user@192.168.0.5               student2
@                              student1
```

На основании приведенного выше описания различным пользователям, работающим за теми или иными компьютерами, будут сопоставлены следующие роли:

- пользователю user, работающему на компьютере с IP-адресом 192.168.0.5 - student2;
- пользователю user, работающему на компьютере с IP-адресом 192.168.0.4 - student3;
- пользователю user, работающему на компьютере с IP-адресом 192.168.1.15 - student5;
- пользователю user1, работающему на компьютере с IP-адресом 192.168.0.4 - student4;
- пользователю user1, работающему на компьютере с IP-адресом 192.168.1.15 – student1.

Файл *generic_roles* содержит информацию о действии системы фильтрации для различных ролей, определенных администратором системы. Как и в файле users пустые строки, строки из пробелов и знаков табуляции, а также строки, начинающиеся со знака

«#», считаются комментариями и в процессе обработки игнорируются. Формат строк в файле следующий:

```
role <имя роли>
     <ключевое слово> <значение>[ <еще значение>]
     <ключевое слово> <значение>[ <еще значение>]
     <ключевое слово> <значение>[ <еще значение>]
```

- **<имя роли>** - уникальный идентификатор роли. По сути, является именем списка действий, предпринимаемых системой фильтрации. Следовательно, определение роли с тем или иным уникальным именем в файле generic_roles может быть только однократным.

Если ключевое слово (по сути, тип действия системы фильтрации) может содержать более одного значения, то все эти значения можно указать в одной строке, разделив их пробелами.

Допустимые ключевые слова, их назначение и возможные значения приведены ниже:

- **parent** - значением является имя другой, родительской, роли; используется для включения в текущую роль полного списка действий, уже описанного для другой роли, чтобы не описывать их повторно;
- **accept** - значением является номер категории; данной роли разрешается доступ к сайтам указанных категорий;
- **reject** - значением является номер категории; данной роли запрещается доступ к сайтам указанных категорий;
- **redirect** - значением является номер категории; при доступе к сайтам указанных категорий для данной роли осуществляется перенаправление запроса на сайт, заданный параметром url_filter.RedirectUrl конфигурационного файла **c-icap.conf**;
- **blacklist** - значением является URL (допускается использование подстановочного символа «*», обозначающего любую последовательность символов); если URL запрашиваемого ресурса совпадает с заданным (с учетом обработки символа «*»), то доступ пользователя к данному ресурсу запрещается независимо от его категории;
- **whitelist** - значением является URL (допускается использование подстановочного символа «*», обозначающего любую последовательность символов); если URL запрашиваемого ресурса совпадает с заданным (с учетом обработки символа

«*»), то доступ пользователя к данному ресурсу разрешается независимо от его категории;

- **exactmatch** - флаг, принимающий значение «истина», если указано одно из ключевых слов: «*on*», «*true*» или «*1*», или значение «ложь», если указано одно из ключевых слов: «*off*», «*false*» или «*0*»; если ни одно из ключевых слов не указано, то значение флага принимается равным «истина». В случае значения флага равного «истина» для всех последующих за ним директив blacklist и whitelist при обработке в конец каждого значения URL автоматически дописывается «*».

Файл *custom_roles* имеет тот же формат, что и generic_roles, но в нем не допускается использование директивы **accept**.

Описание программы проверки БД политик

Программа проверки БД политик **/usr/lib/host2cat/check.pl** позволяет проверить работоспособность сформированной БД политик.

Запуск программы осуществляется из командной строки со следующими параметрами:

```
check.pl [-h] <имя файла БД> <имя пользователя> <IP адрес> <URL запрашиваемого ресурса>
```

Результат работы программы выводиться на экран в виде:

```
Uname: <имя пользователя>
IP: <IP адрес>
URL: <URL запрашиваемого ресурса>
Host: <Имя сервера>
Role: <Назначенная роль пользователю>
Cats: <Категория ресурса>
Action: <действие> (допустимые значения - ACCEPT, REJECT, REDIRECT)
```

3.2.4. Веб-интерфейс настройки МКФ

Настройка МКФ может быть произведена через веб-интерфейс. Для этого в адресной строке веб-браузера необходимо ввести адрес сайта управления МКФ, например: http://192.168.100.1:80/cgi-bin/login.cgi. Появится окно с веб-формой аутентификации, в котором при первом входе необходимо ввести учётные данные суперпользователя (Рис. 42).

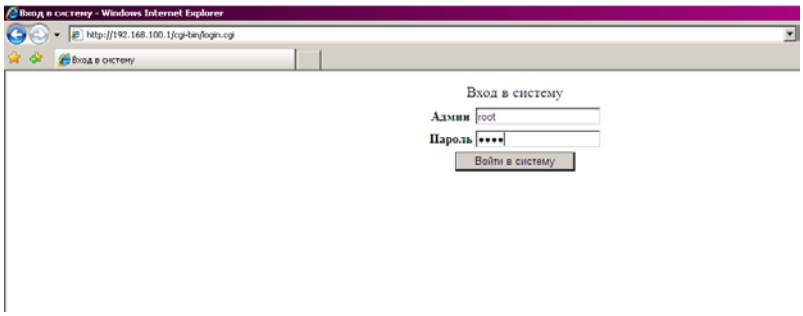

Рис. 42. Веб-форма аутентификации сайта управления МКФ

Появится основное окно сайта управления МКФ (Рис.43).

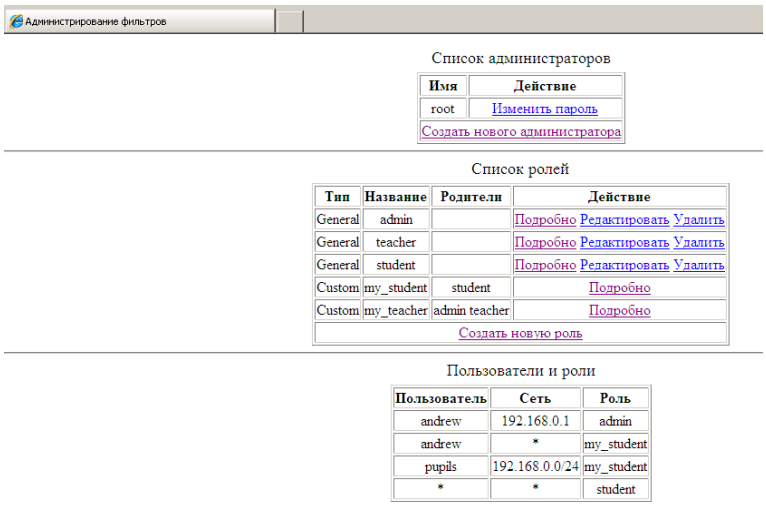

Рис. 43. Основное окно сайта управления МКФ

Следующим шагом требуется создать учётную запись администратора МКФ, чтобы не задействовать суперпользователя для этих целей. Для этого необходимо перейти по ссылке **Создать нового администратора и ввести имя и пароль для новой учётной записи** (Рис 44).

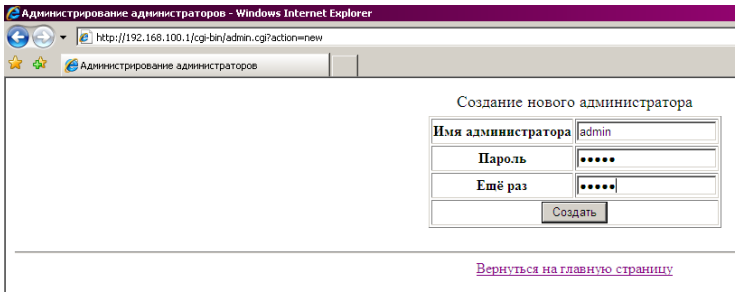

Рис. 44. Окно создания администраторов сайта управления МКФ

После создания учётной записи администратора МКФ необходимо повторно набрать в адресной строке браузера адрес веб-формы аутентификации сайта управления МКФ и зарегистрироваться от имени вновь созданного администратора. Основное окно управления МКФ будет несколько отличаться от предыдущего (Рис. 45).

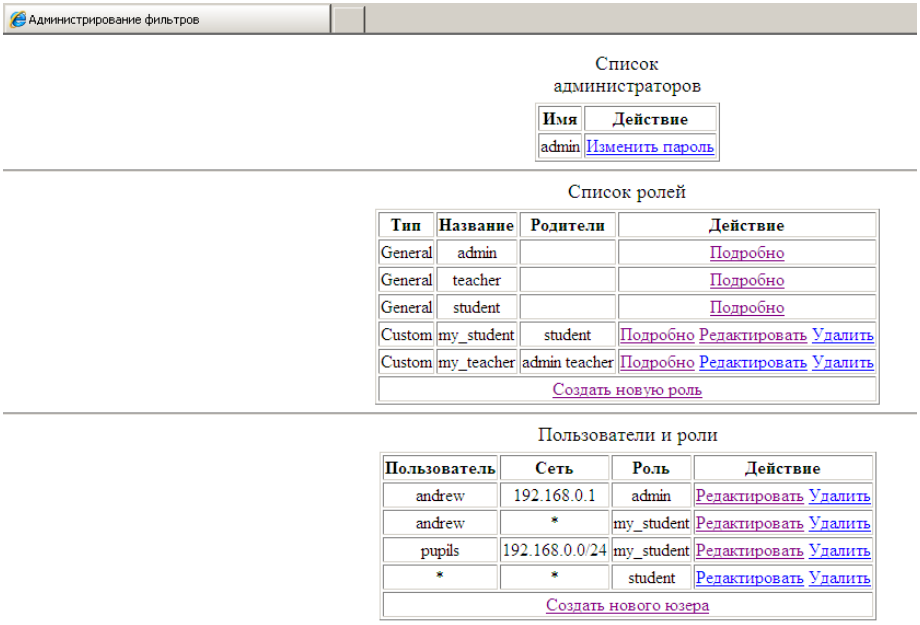

Рис. 45. Основное окно сайта управления в режиме администратора МКФ

Если в основном окне в списке ролей перейти по ссылке **Создать новую роль**, то откроется окно создания новой роли (Рис. 46).

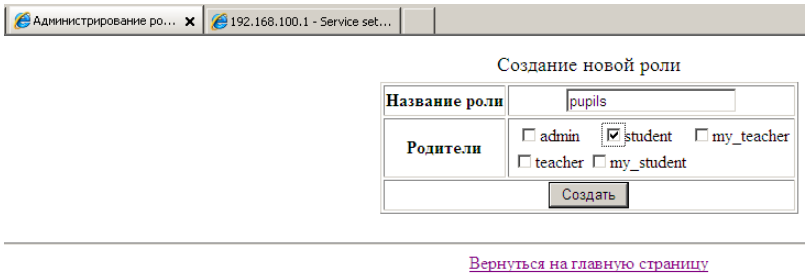

Рис. 46. Окно создания новой роли

Если в основном окне в списке пользователей перейти по ссылке **Создать нового юзера**, то откроется окно создания нового пользователя (Рис. 47).

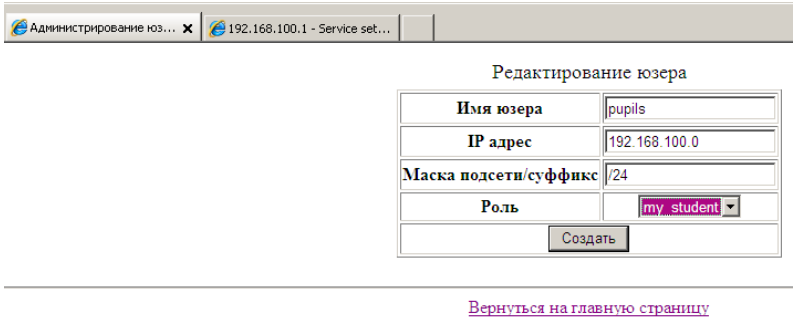

Рис. 47. Окно создания нового пользователя

Если в основном окне в списке ролей перейти по ссылке **Редактировать** напротив выбранной роли, то откроется окно редактирования роли (Рис. 48).

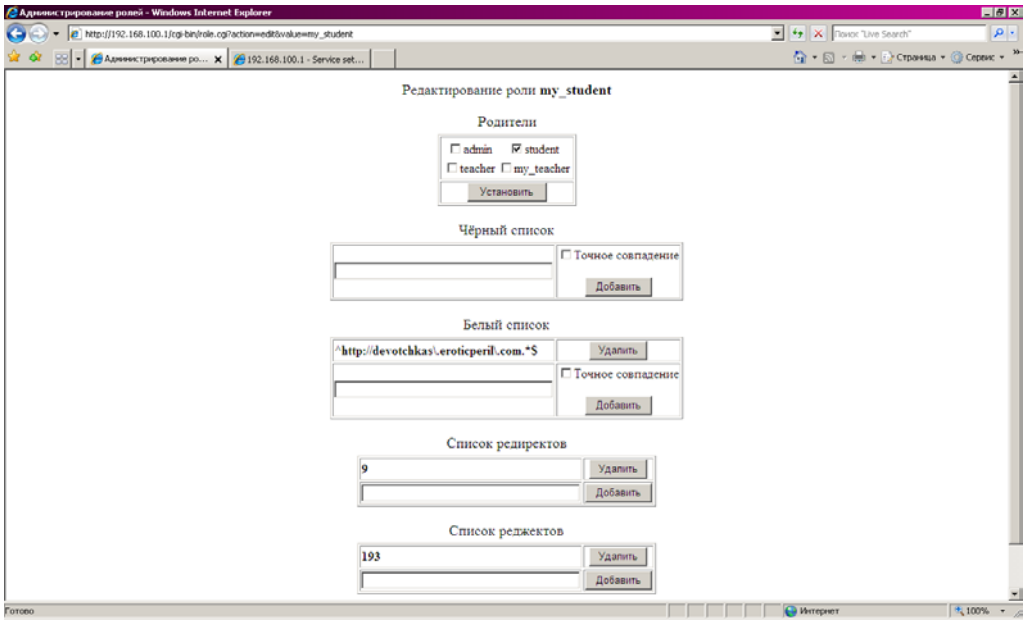

Рис. 48. Окно редактирования роли

Если в основном окне в списке ролей перейти по ссылке **Подробно** напротив выбранной роли, то откроется окно свойств роли (Рис. 49).

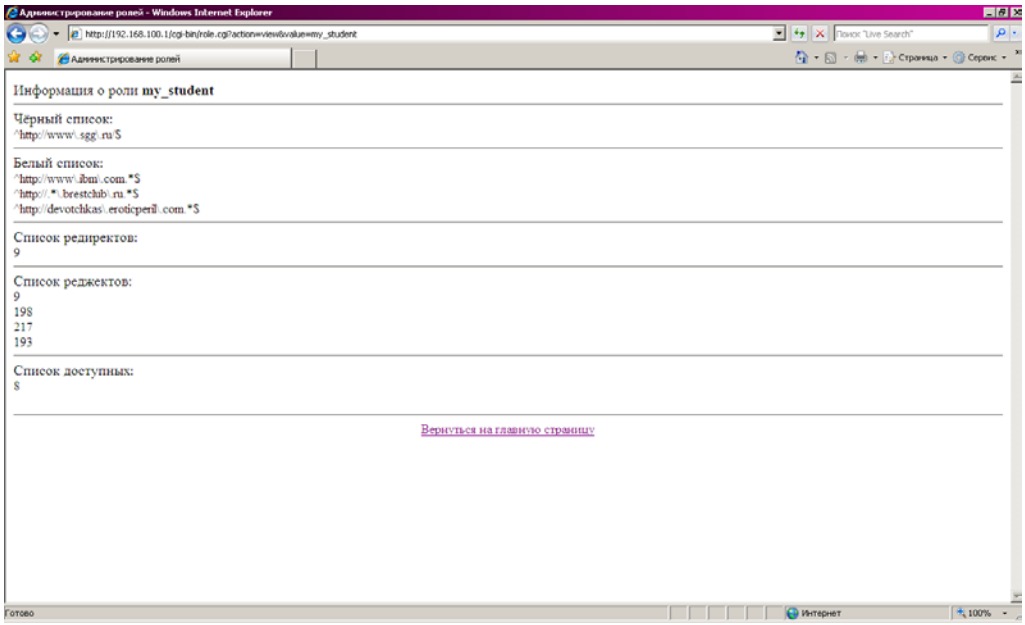

Рис. 49. Окно свойств роли

Параметры, указываемые при создании и редактировании ролей и пользователей полностью аналогичны описанным выше при создании файлов users, custom_roles и generic_roles.

После внесения любых изменений через веб-интерфейс необходимо перезапустить **icap-сервер**:

```
service c-icap restart
```

Если все настройки сделаны правильно, то при попытке пользователя обратиться к запрещённой странице будет выведена веб-страница (Рис. 50):

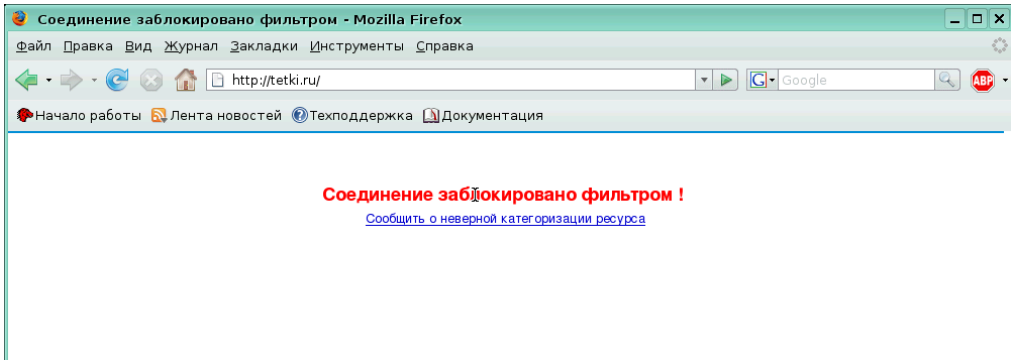

Рис. 50. Окно, информирующее о запрете доступа к запрашиваемому ресурсу

3.2.5. Упрощённая настройка МКФ

Если отсутствует необходимость организовывать ролевой доступ к категорированным сайтам, то можно подключить DNS-фильтр. Для этого необходимо сконфигурировать на компьютере, на котором функционирует прокси-сервер Squid, следующие адреса DNS-серверов:

```
81.176.72.82
81.176.72.83
```

Эти DNS-серверы принадлежат компании-разработчику СКФ/МКФ – *Центру Анализа Интернет-ресурсов (ЦАИР)*. Они же используются и при функционировании всего комплекса СКФ.

При использовании этого сценария приходится полностью полагаться на решение, принятое разработчиками СКФ/МКФ.