Владимир Ануфриев

Интеграция системы обнаружения и предупреждения вторжений Snort в среду OpenSuse 11 и разработка для нее новой консоли управления

Москва
2009

Vladimir Anufriev

Web-based Control Console for Snort-Inline Intrusion Detection System

First Edition (November 2009)

This edition applies to Release 1.0 of AVN Console for Snort IDS

Comments may be addressed to: avn@avnsite.com

Введение

Как известно, для системы обнаружения вторжений (Intrusion detection system – IDS) Snort[1] давно существует ставшая уже стандартом консоль управления, называющаяся BASE[2]. Она в целом позволяет отслеживать результаты фильтрации трафика и предоставляет большое количество статистических отчетов. Пример начальной страницы этой программы представлен на рисунке 1.

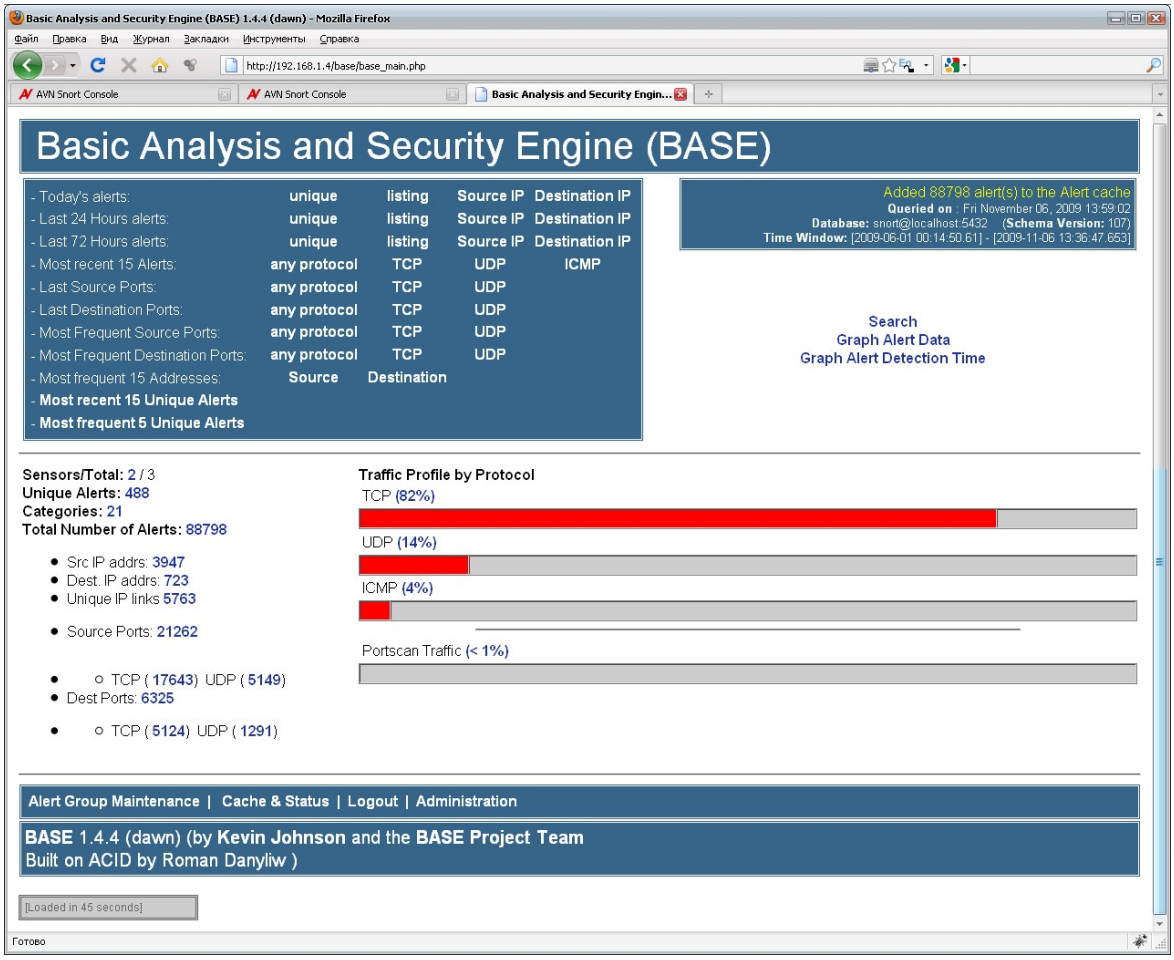

Рис. 1. Начальная страница консоли BASE

Начальная страница дает мало информации офицеру информационной безопасности, если он должен хотя бы периодически отслеживать происходящее на внешнем периметре корпоративной сети. Одна из множества страниц с более содержательной информацией представлена на рисунке 2.

---

1 http://www.snort.org – разработана компанией Sourcefire. Распространяется на основе лицензии GPL. Набор периодически обновляемых сигнатур фильтрации трафика, поддерживаемый компанией-разработчиком, распространяется на платной основе, но с некоторой задержкой выкладывается в открытый доступ для зарегистрировавшихся на сайте производителя пользователей.

2 http://base.secureideas.net/ - Basic Analysis and Security Engine (BASE). Распространяется на основе лицензии GPL.

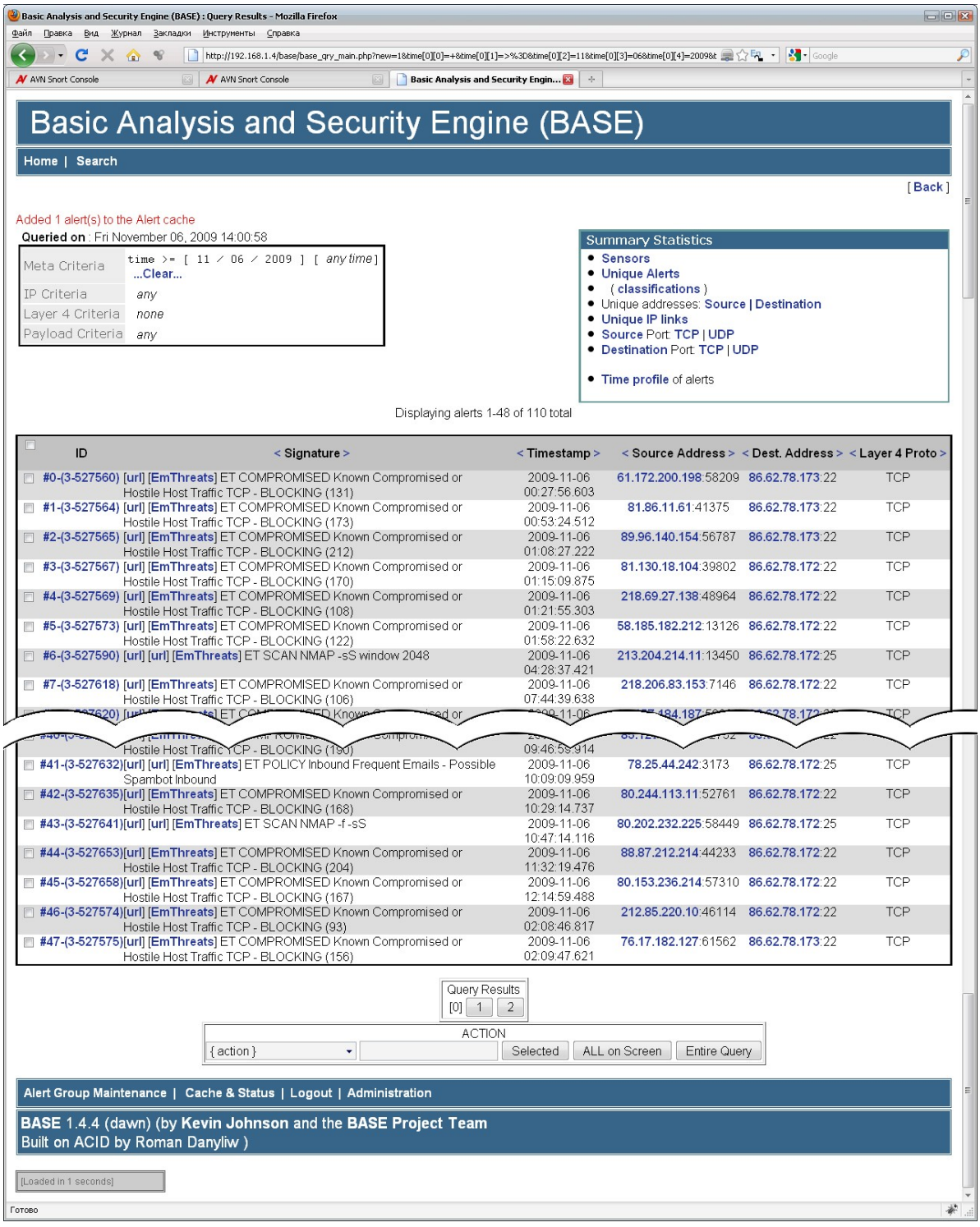

Рис. 2. Страница BASE со списком последних событий

К сожалению, более детальную информацию нам получить не удалось, так как дальше последняя версия системы (1.4.4) выдает ошибки (см. рис. 3), что было характерно и для ее более старых версий, на доработку и адаптацию которых мы когда-то потратили

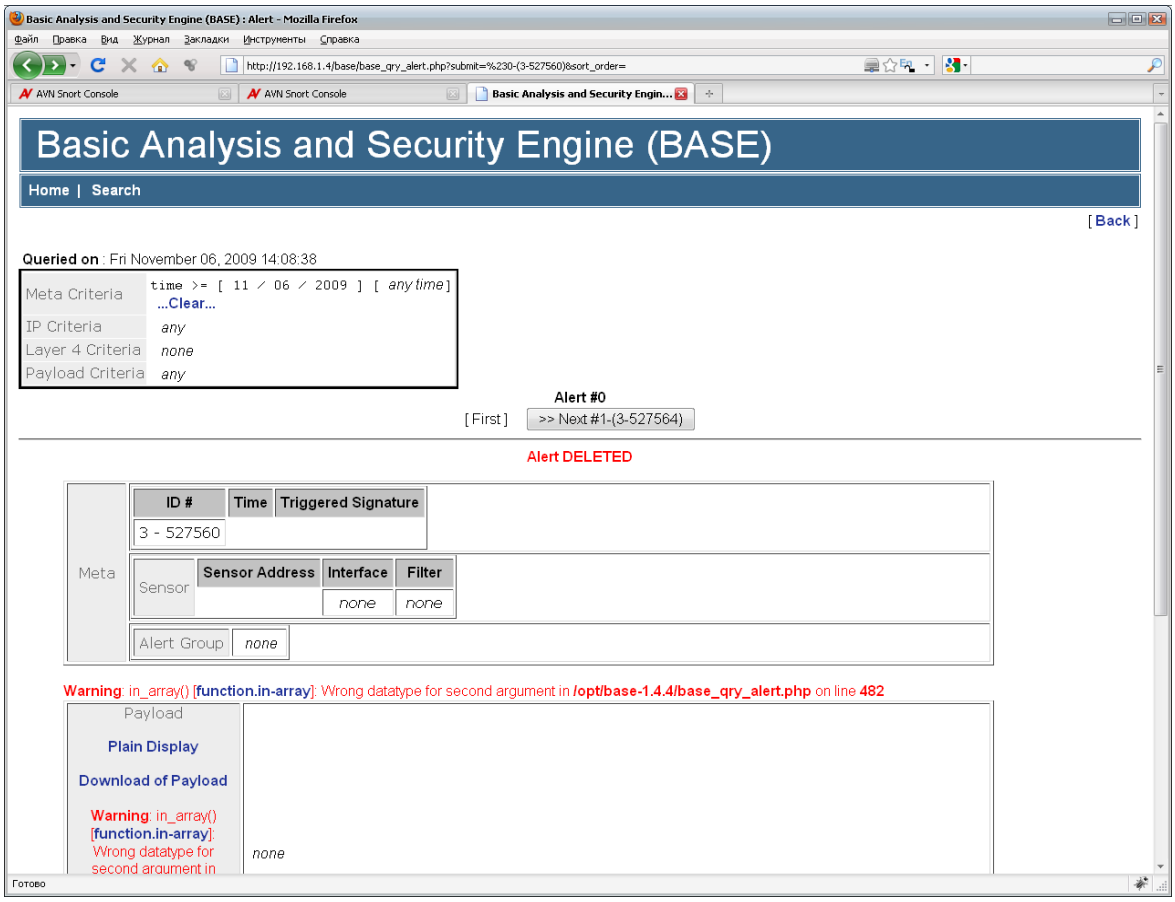

Рис. 3. Результат попытки посмотреть подробности события

значительное время. В любом случае, **BASE имеет ряд существенных недостатков**, делающих ее трудноприменимой в реальных приложениях. В частности:

- BASE не позволяет управлять набором сигнатур, загруженных в систему обнаружения вторжений, количество же сигнатур в типовой установке измеряется десятками тысяч, что требует обязательной автоматизации;
- BASE не позволяет загружать обновления сигнатур и перезагружать Snort;
- BASE не имеет никакой связи с другими системами обеспечения безопасности, в частности, с брандмауэром (в Linux 2.6 в качестве брандмауэра используется модуль ядра netfilter с интерфейсом iptables[3]);
- BASE не поможет вам найти, какая машина из внутренней (локальной или DMZ) сети посылает пакеты, на которые реагирует IDS;
- BASE не различает типы сигнатур (реакций на сигнатуры) Snort (pass, drop, sdrop, reject, alert, log) и, соответственно, не может определить и показать оператору реакцию IDS на то или иное событие, но это последнее обстоятельство не столько её вина, сколько недоработка в модуле вывода информации в реляционную СУБД (систему управления базами данных) самой IDS;

[3] http://www.netfilter.org/

– предыдущий недостаток не позволит вам эффективно применять BASE при включении Snort в режиме системы предупреждения вторжений (IPS Snort-Inline).

Таким образом, BASE требует серьезной доработки, которая существенно затруднена в связи с довольно большим объемом этой программы. Её исходный текст на PHP[4] составляет порядка 2.4Мб. Мы решили, что быстрее будет написать систему заново, чем разбираться в этом нагромождении кода. В результате уже через пару недель у нас появился действующий прототип, который в дальнейшем был постепенно отшлифован и насыщен всеми функциями, которые необходимы в повседневной работе. Статистические отчеты нужны крайне редко и их можно всегда получить сторонними средствами с помощью запросов на SQL, поэтому мы не стали нагружать ими интерфейс, что позволило его существенно упростить и ускорить его работу. Исходный код получившейся системы практически в 20 раз (!) меньше, чем у BASE, а функциональность вы сможете оценить по нижеизложенному[5].

Далее мы рассмотрим основные элементы комплексной системы обеспечения информационной безопасности периметра вычислительной сети предприятия, состав задействованных программных средств, их доработки и разработанный нами интерфейс, реализующий управляющую консоль системы обнаружения и предупреждения вторжений.

Общая схема комплекса обеспечения безопасности

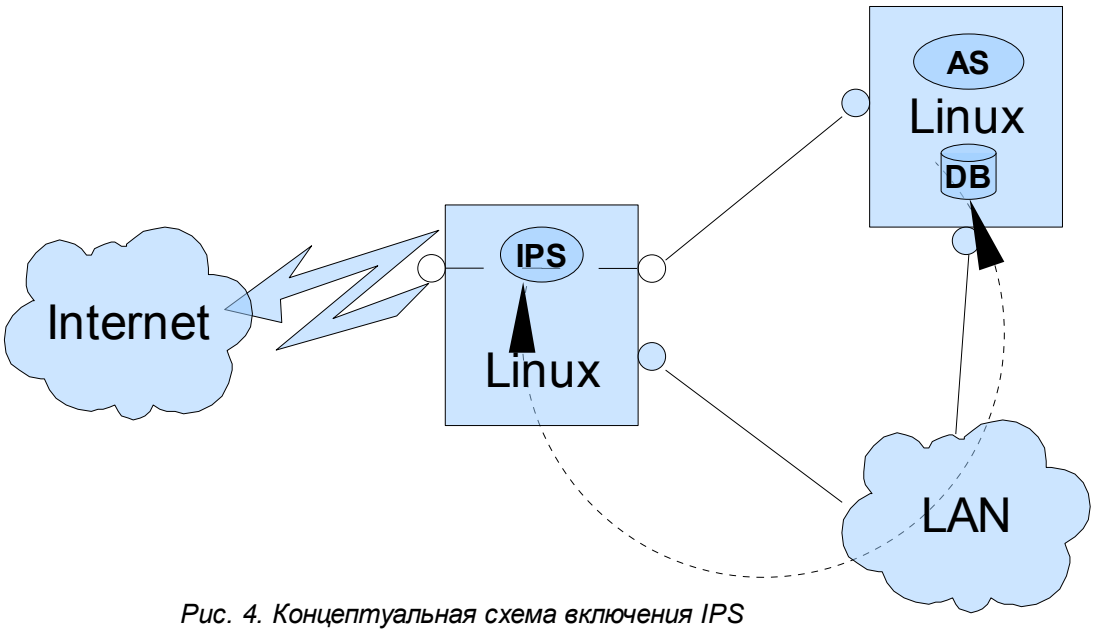

Рис. 4. Концептуальная схема включения IPS

В ходе реализации мы принципиально использовали исключительно бесплатное программное обеспечение, за исключением HTTP-сервера, в качестве которого для

[4] PHP: Hypertext Preprocessor – http://php.net/index.php

[5] Попутно заметим, что полученная разница в объеме исходного кода не может определяться исключительно снижением функциональной нагрузки, так как снижение количества статистической информации, в том числе в графической форме, в некоторой степени компенсируется дополнительными функциями, которые отсутствуют у BASE. Мы относим такое колоссальное уменьшение объема исходного текста на разницу в уровнях применяемых языков визуализации данных (технологии PHP и JSP, соответственно). Затраты на поддержку, обеспечение переносимости и развитие программы, с нашей точки зрения, также должны существенно сократиться, по крайней мере, в части затрачиваемого на это времени.

уменьшения сложности системы мы задействовали имеющийся IBM Lotus Domino. Общая типовая схема системы представлена на рисунке 4.

Представленная схема сильно упрощена по сравнению с реальной, но дает общее представление, так как не загромождена избыточными подробностями. Возможен вариант, когда все компоненты системы расположены на одном сервере. В этом случае сервер должен иметь четыре сетевых интерфейса.

В качестве операционных систем мы везде, где можно, задействовали Linux OpenSuse последней доступной версии.

В связи с деликатностью обсуждаемой темы некоторые подробности мы опустим, но обо всех существенных моментах расскажем.

Установка IPS

Итак, Snort IDS последней версии включается в режиме IPS («прозрачного моста») или, по терминологии Sourcefire, в варианте Snort-Inline. На рисунке 4 интерфейсы прозрачного моста оставлены без заливки цветом, таким образом мы пометили, что они не имеют IP-адресов и не могут адресоваться из-вне ближайшего сегмента сети провайдера связи. В целом, установка типовая, в соответствующей документации есть почти исчерпывающая информация на этот счет. В конце концов мы написали довольно сложный скрипт для запуска сервера IPS, но его «навороты» требуются, главным образом, для обеспечения непрерывности связи и вывода диагностики при рестарте IDS через консоль управления, а также средств, повышающих надежность (безотказность) работы IDS.

Как уже было отмечено выше, модуль (plugin) вывода информации о событиях в базу данных spo_database не сохраняет необходимой для работы консоли информации, хотя в модуль текстового журнала spo_alert_fast нужная функциональность заложена[6].

Для начала нужно слегка усовершенствовать схему базы данных Snort, добавив поле в одну из основных таблиц, чтобы создать место, куда будет записываться необходимая для управления IPS информация:

```
ALTER TABLE event ADD COLUMN flags integer;
ALTER TABLE event ALTER COLUMN flags SET STORAGE PLAIN;
COMMENT ON COLUMN event.flags IS 'Packet flags if it is not a fragment';
```

Также незначительно модифицируем еще одну таблицу:

```
ALTER TABLE reference_system ADD COLUMN url text;
ALTER TABLE reference_system ALTER COLUMN url SET NOT NULL;
ALTER TABLE reference_system ALTER COLUMN url SET DEFAULT 'http://'::text;
COMMENT ON COLUMN reference_system.url IS 'Base part of URL';
```

Заметим, что наши изменения не влияют на функциональность стандартного варианта Snort и консоли BASE. Громоздить новые таблицы, ради того, чтобы оставить нетронутыми стандартные, из-за вышеприведенных модификаций, с нашей точки зрения, нецелесообразно.

Теперь можно доработать исходный текст модуля spo_database:

```
--- src/output-plugins/spo_database.c       2009-10-03 00:29:59.000000000 +0400
+++ src/output-plugins/spo_database.c       2009-11-06 18:17:07.000000000 +0300
```

[6] `if( p != NULL && p->packet_flags & PKT_INLINE_DROP )`
`  TextLog_Puts(data->log, " [Drop]");`

```
@@ -1660,33 +1660,65 @@
     if ( (data->shared->dbtype_id == DB_ORACLE) &&
          (data->DBschema_version >= 105) )
     {
+#if defined( LOGPACKET_FLAGS )
+       if( !p )
+#endif
         ret = SnortSnprintf(query->val, MAX_QUERY_LENGTH,
                             "INSERT INTO "
                             "event (sid,cid,signature,timestamp) "
                             "VALUES (%u, %u, %u, TO_DATE('%s', 'YYYY-MM-DD HH24:MI:SS'))",
                             data->shared->sid, data->shared->cid, sig_id, timestamp_string);
-
+#if defined( LOGPACKET_FLAGS )
+       else
+         ret = SnortSnprintf(query->val, MAX_QUERY_LENGTH,
+                             "INSERT INTO "
+                             "event (sid,cid,signature,timestamp,flags) "
+                             "VALUES (%u, %u, %u, TO_DATE('%s', 'YYYY-MM-DD HH24:MI:SS'), %u)",
+                                  data->shared->sid, data->shared->cid, sig_id, timestamp_string, p-
>packet_flags);
+#endif
         if (ret != SNORT_SNPRINTF_SUCCESS)
             goto bad_query;
     }
     else if(data->shared->dbtype_id == DB_ODBC)
     {
+#if defined( LOGPACKET_FLAGS )
+       if( !p )
+#endif
         ret = SnortSnprintf(query->val, MAX_QUERY_LENGTH,
                             "INSERT INTO "
                             "event (sid,cid,signature,timestamp) "
                             "VALUES (%u, %u, %u, {ts '%s'})",
                             data->shared->sid, data->shared->cid, sig_id, timestamp_string);
+#if defined( LOGPACKET_FLAGS )
+       else
+         ret = SnortSnprintf(query->val, MAX_QUERY_LENGTH,
+                             "INSERT INTO "
+                             "event (sid,cid,signature,timestamp,flags) "
+                             "VALUES (%u, %u, %u, {ts '%s'}, %u)",
+                                  data->shared->sid, data->shared->cid, sig_id, timestamp_string, p-
>packet_flags);
+#endif

         if (ret != SNORT_SNPRINTF_SUCCESS)
             goto bad_query;
     }
     else
     {
+#if defined( LOGPACKET_FLAGS )
+       if( !p )
+#endif
         ret = SnortSnprintf(query->val, MAX_QUERY_LENGTH,
                             "INSERT INTO "
                             "event (sid,cid,signature,timestamp) "
                             "VALUES (%u, %u, %u, '%s')",
                             data->shared->sid, data->shared->cid, sig_id, timestamp_string);
+#if defined( LOGPACKET_FLAGS )
+       else
+         ret = SnortSnprintf(query->val, MAX_QUERY_LENGTH,
+                             "INSERT INTO "
+                             "event (sid,cid,signature,timestamp,flags) "
+                             "VALUES (%u, %u, %u, '%s', %u)",
+                                  data->shared->sid, data->shared->cid, sig_id, timestamp_string, p-
>packet_flags);
+#endif

         if (ret != SNORT_SNPRINTF_SUCCESS)
             goto bad_query;
@@ -2190,9 +2222,8 @@
 /* Historically these were together in a common "else".
  * Keeping it that way until somebody complains...
  */
-#if defined(ENABLE_MYSQL) || defined(ENABLE_POSTGRESQL)
-    if (data->shared->dbtype_id == DB_MYSQL ||
-        data->shared->dbtype_id == DB_POSTGRESQL)
```

```
+#if defined(ENABLE_POSTGRESQL)
+    if ( data->shared->dbtype_id == DB_POSTGRESQL)
     {
       for(end=from+from_length; from != end; from++)
       {
@@ -2233,8 +2264,13 @@
             *to++= '\\';
             *to++= '\\';
             break;
-          case '\'':            /* '  -->  \' */
-            *to++= '\\';
+          case '\'':
+#if 0
+// standard_conforming_strings = off
+            *to++= '\\';            /* '  -->  \' */
+#else
+            *to++= '\'';            /* '  -->  '' */
+#endif
             *to++= '\'';
             break;
           case '"':             /* "  -->  \" */
```

Поскольку мы включаем Snort в режиме IPS, то есть в режиме предупреждения, а не только обнаружения вторжений, то желательно задействовать механизм связи IDS с брандмауэром, возможности которого учитываются в сигнатурах из набора Emerging Threats[7]. Эта функциональность требует установки дополнительного сервера и добавления в исходный текст Snort специального модуля вывода. Речь идет о продукте SnortSam. Он тоже требует доработок, если вы хотите использовать рассматриваемую связку программ не только для лабораторных опытов. Этому вопросу посвящена отдельная небольшая статья, которую можно взять на сайте автора[8]. После того как все патчи сложены в отдельный каталог, можно начать сборку Snort.

Перейдем в каталог, в который мы сложили все патчи, применим их, запустим процедуру конфигурирования с указанными ниже параметрами, а затем программу сборки make из каталога с исходным текстом. Примерные команды, которые для этого потребуются приведены ниже. Необходимые каталоги можно переназначить, исходя из особенностей вашей установки. В нашем случае написан отдельный скрипт:

```
VERSION=2.8.5.1
PATCHDIR=`pwd`
PREFIX=/opt/snort/snort-$VERSION
grep x86_64 /etc/SuSE-release > /dev/null
if [ $? -eq 0 ] ; then
  LIBX=lib64
else
  LIBX=lib
fi
cd $PREFIX
for f in $PATCHDIR/*.patch ; do
  patch $DRY -p0 < $f
done
find $PATCHDIR -name \*.diff -exec patch -p1 -i \{\} \;
#
if [ -f autojunk.sh ] ; then
  sh autojunk.sh
fi
export LIBS=-lsupc++
export LDFLAGS="-L/usr/local/$LIBX -export-dynamic"
export CFLAGS="-g -I/usr/local/include -DLOGPACKET_FLAGS -rdynamic"
./configure --prefix=/opt/snort \
--enable-inline \
--enable-dynamicplugin \
--enable-decoder-preprocessor-rules \
```

[7] http://emergingthreats.net/

[8] V. Anufriev. SnortSam Implementation in Complex Environment. http://www.avnsite.com/avn/publications/SnortSam.pdf

```
--with-postgresql=/usr \
--with-pgsql-includes=/usr/include/pgsql \
--with-dnet-includes=/usr/local/include \
--with-dnet-libraries=/usr/local/$LIBX \
--with-libnet-libraries=/usr/local/$LIBX

cd $PREFIX/src
make
```

Конфигурирование и запуск Snort описан в документации на эту программу, поэтому мы не будет сейчас подробно на этом останавливаться. Не забудем, что нам нужно создать «прозрачный» мост, чего стандартная программа yast из OpenSuse делать не умеет. Yast требует, чтобы у моста был IP-адрес, а нам он не нужен. Поэтому создаем мост, как получается, задав какой-нибудь «левый» статический IP, а затем слегка исправляем настройки в /etc/sysconfig/network, удаляя упоминание об IP-адресе. То же необходимо обеспечить и для обоих интерфейсов, входящих в мост. Можно, конечно, вообще все делать вручную, но, как показала практика, с помощью yast получается быстрее.

Настройка брандмауэра

Snort в режиме IPS не работает без брандмауэра. Брандмауэр направляет пакеты, которые, как он, точнее, конечно, настраивающий его системный программист, предполагает должны быть обработаны Snort'ом, в специальную цепочку QUEUE, для чего ядро операционной системы должно поддерживать эту функциональность, а необходимый модуль ядра ip_queue быть загружен (команда modprobe ip_queue обеспечит это).

Подробному описанию настройки брандмауэра под OpenSuse, наверное, придется посвятить отдельное изложение, потому что это дело довольно сложное и объемное. В общем случае нужно применять внешние ресурсы для ограничения трафика с нежелательных, а точнее, гарантированно скомпрометированных узлов. И никакими средствами визуальной интерактивной настройки правил, типа webadmin или того же yast, здесь отделаться не получится. Наш действующий набор правил брандмауэра netfilter (команд iptables), например, в настоящее время насчитывает 2530 штук. Поэтому задействовать придется профессиональные механизмы, к счастью, заложенные в скриптах OpenSuse. Придется писать десятки строк собственного кода в файл /etc/sysconfig/scripts/SuSEfirewall2-custom и аккуратно прописывать параметры в /etc/sysconfig/SuSEfirewall2, в том числе, в дополнительных нестандартных зонах. Описанию этого увлекательного процесса можно посвятить не один десяток страниц, а время, потраченное на самостоятельное изучение этого вопроса, можно сравнить со временем разработки основного предмета настоящей статьи.

Собственно Snort требует всего нескольких команд. Минимальный набор состоит из одного правила, которое приведено в документации производителя. Из общего действующего списка команд мы выбрали несколько, существенных для пояснения взаимодействия брандмауэра и Snort. Они выглядят так:

```
# ...
iptables -A FORWARD -m physdev --physdev-is-bridged -j ACCEPT
# ...
iptables -A forward_bridge -j ETBLOCKLIST
iptables -A forward_bridge -j forward_fwsam
iptables -A forward_bridge -m physdev  --physdev-in eth0 -j forward_bridge_ext
iptables -A forward_bridge -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN -m limit --limit 3/min -j
LOG --log-prefix "BR-QUE-TCP
iptables -A forward_bridge -p tcp -m tcp -j QUEUE
iptables -A forward_bridge -p udp -m udp -j QUEUE
iptables -A forward_bridge -p icmp -m icmp --icmp-type any -j QUEUE
iptables -A forward_bridge -m state --state INVALID -m limit --limit 3/min -j LOG --log-prefix "BR-
DROP-DEFLT-INV " --log-tcp-op
iptables -A forward_bridge -j DROP
# ...
```

```
iptables -I FORWARD -i $INTERFACE -j forward_bridge
# ...
```

Здесь переменная окружения $INTERFACE берет свое значение из файла /etc/sysconfig/snort, которое задает имя интерфейса сконфигурированного «прозрачного» моста. Согласно рекомендациям из вышеупомянутой статьи (см. сноску 9), мы создали специальную цепочку правил forward_bridge. Многоточиями помечены пропуски в данной цитате из общего списка. Важно отметить, что в последней команде, которая и переключает входящий трафик на Snort, используется ключ -I, который ставит соответствующее правило в начало цепочки FORWARD. Следующим за ним правилом в нашем случае является то, что стоит первым в вышеприведенной цитате.

Теперь, чтобы отключить Snort, и пустить весь трафик в обход его («замкнуть» мост), нужно дать команду:

```
iptables -D FORWARD -i $INTERFACE -j forward_bridge
```

Следующим за исключенным данной командой из цепочки FORWARD правилом будет правило `-A FORWARD -m physdev --physdev-is-bridged -j ACCEPT`, которое просто пропускает все пакеты через интерфейсы, включенные в мост. Таким образом, мы можем включать и отключать IPS, не выгружая Snort из памяти. Его загрузка занимает приличное время, так как типовой набор сигнатур, как уже было сказано, насчитывает порядка двух десятков тысяч штук.

В правилах цепочки forward_bridge мы проходимся по списку нежелательных адресов, который поддерживается сервером SnortSam: `-A forward_bridge -j forward_fwsam`. Три правила, оканчивающихся на -j QUEUE, собственно, и передают трафик на проверку IPS. Из можно объединить в одно, но мы оставили три для общности, предусмотрев дополнительно, чтобы информация о пакетах TCP сохранялась в журнале для отладки и возможного анализа. Остальные тысячи правила будем, как и условились выше рассматривать в другой раз.

Установка сервера СУБД, HTTP-сервера и сервера приложений

В качестве сервера СУБД мы используем PostgreSQL одной из последних, вышедших к настоящему моменту, версий. Разработанная нами новая консоль управления изначально ориентирована на работу именно с этой СУБД и отвязка от нее потребует некоторых изменений в коде. При наличии достаточных ресурсов на сервере с IPS, можно устанавливать СУБД на него, как и вообще все упоминаемые здесь приложения.

Установка и настройка PostgreSQL выполняются стандартным способом. Единственное, что нужно отследить, это настройки конфигурации соответствующей базы данных (файл data/postgresql.conf):

```
default_with_oids = off
escape_string_warning = off
standard_conforming_strings = on
```

Внимание, последний параметр требует изменений в `src/output-plugins/spo_database.c`.

Сервером приложений для новой консоли является Tomcat[9]. Вся интерактивная часть консоли написана на технологии JSP[10]. Установка и настройка Tomcat версии 6 не требуют специальных пояснений, если вы, конечно, не используете как и мы в качестве HTTP-сервера Domino, а не Apache. В этом случае одними «пояснениями» не отделаешься, а требуется

[9] Apache Tomcat 6.0 - http://tomcat.apache.org/

[10] JavaServer Pages Technology - http://java.sun.com/products/jsp/

написание довольно сложного шлюза, который описан в серии посвященных ему статей, которые можно вместе с одной из старых, но вполне работающих, версий взять на сайте автора[11]. Настройка Tomcat в комплексе с Apache многократно описана в соответствующей документации, статьях и книгах, поэтому этот вопрос здесь дополнительно обсуждать не будем. О настройке приложения будет сказано чуть ниже.

Для работы приложения создаются ряд дополнительных, по отношению к схеме Snort 1.06, таблиц. Идея хранить правила в таблицах, их первоначальная структура и скрипты загрузки заимствованы нами из давно заброшенного проекта, который называется Snort Rules Remote Access Management (SRRAM)[12]. Мы постарались сохранить преемственность в названиях таблиц и полей из уважения к авторам SRRAM. Поскольку проект был заброшен в конце 2002 года и изначально строился на отличной от нашей технологии, мы смогли использовать из него только некоторые идеи и отдельные небольшие куски кода на Perl. В текущей версии нашей консоли используются следующие таблицы:

categories	категории сигнатур (базовые имена файлов, в которые изначально помещены сигнатуры Snort)
flags	SnortDB Extra Table: комбинации флагов протокола TCP
messages	информационные сообщения для диагностики работы внешних, по отношению к серверу приложений, скриптов
protocols	SnortDB Extra Table: список поддерживаемых системой протоколов (импортируется из /etc/protocols)
services	SnortDB Extra Table: список поддерживаемых системой сервисов (импортируется из /etc/services)
sig_desc	описания сигнатур
sig_affected	список систем, подверженных атакам, связанным с соответствующими сигнатурами
sig_contrib	авторы и источники сигнатур
snort_rules	список сигнатур
snort_rules_load	источники сигнатур (URL и др.)
sqacclog	журнал кэширующего прокси сервера (в нашем случае Squid)

В отличие от авторов BASE мы делаем попытку синтаксического разбора описаний сигнатур из каталога doc/signatures, включаемого в архив с сигнатурами. К сожалению, строгих правил авторы этих описаний, как нам кажется, не придерживаются, что создает определенные сложности. Однако результат приложенных усилий оправдывается существенным повышением информативности сообщений о событиях и позволяет сделать их гипертекстовыми.

Таким образом, после установки СУБД необходимо создать структуру базы данных (БД), с перечисленными выше таблицами, индексами и правилами связанности, а также несколькими хранимыми функциями, упрощающими код.

Приведем здесь пару примеров. В качестве примера хранимой функции покажем текст enabled2str, преобразующий код типа сигнатуры из вида, хранимого в БД, в читаемый человеком-оператором консоли:

```
CREATE OR REPLACE FUNCTION enabled2str(flag character)
  RETURNS text AS
$BODY$
  BEGIN
```

[11] http://www.avnsite.com/downloads/domino/Domino2Tomcat.pdf
http://www.avnsite.com/downloads/domino/domino-1.13.1.tar.gz
http://www.avnsite.com/avn/publications/Domino2TomcatAuth.pdf

[12] http://sourceforge.net/projects/srram/

```
    IF ( flag = 'Y' ) THEN
      RETURN 'ALERT';
    ELSIF ( flag = 'N' ) THEN
      RETURN 'DISABLED';
    ELSIF ( flag = 'A' ) THEN
      RETURN 'ALERT';
    ELSIF ( flag = 'D' ) THEN
      RETURN 'DROP';
    ELSIF ( flag = 'S' ) THEN
      RETURN 'SDROP';
    ELSIF ( flag = 'R' ) THEN
      RETURN 'REJECT';
    ELSIF ( flag = 'P' ) THEN
      RETURN 'PASS';
    ELSIF ( flag = 'L' ) THEN
      RETURN 'LOG';
    ELSE
      RETURN 'UNKNOWN';
    END IF;
  END;
  $BODY$
  LANGUAGE 'plpgsql' IMMUTABLE STRICT;
```

Как пример описания таблицы приведем скрипт для создания таблицы snort_rules (имя и большинство полей унаследованы от программы SRRAM):

```
CREATE TABLE snort_rules
(
  rule_uid     serial NOT NULL,
  rule_sid     integer,
  rule_rev     integer,
  "rule"       text NOT NULL,
  enabled      character(1),
  rule_cat_1   character varying(255),
  rule_cat_2   character varying(255),
  mtime        timestamp,
  state        text DEFAULT 'NEW'::text,
  CONSTRAINT snort_rules_pkey PRIMARY KEY (rule_uid),
  CONSTRAINT snort_rules_category_fk FOREIGN KEY (rule_cat_1)
      REFERENCES categories (category) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE RESTRICT
)
WITH (OIDS=TRUE);
```

Кроме этого мы задействуем все три таблицы, помеченные на стандартной схеме БД (Snort Database Schema v1.06[13]) как необязательные (SnortDB Extra Table). Для автоматического заполнения этих таблиц разработаны специальные скрипты. Обычно эта процедура проводится однократно и далее информация в этих таблицах не меняется. Прежде чем запускать упомянутые выше скрипты необходимо убедиться, что файлы /etc/protocols и /etc/services в вашей системе содержат актуальную информацию. Это далеко не всегда так! Часто системные программисты применяют нестандартные порты, и далеко не все приложения, существующие в мире и применяемые в корпоративной среде, отражены в /etc/services. Файл, входящий в операционную систему по-умолчанию, требует обязательного исправления в виду наличия дублирующихся ключей (в частности, это касается протокола HTTP).

Новая консоль управления для своей работы требует наличия ряда специальных программ, выполненных в виде скриптов на языках Bash и Perl на сервере IDS и на сервере приложений. XML-файл настройки приложения META-INF/context.xml в случае разнесения сервера приложений и сервера IDS между разными машинами может выглядеть так:

```
<Context path="/snort" debug="0" reloadable="true" crossContext="false" privileged="true">
  <Valve className="com.avn.tomcat.authenticator.DominoAuthenticator" cache="false" debug="0"/>
```

[13] Файл doc/snort_schema_v106.pdf включается в дистрибутив Snort

```
  <Resource name="jdbc/snort"
            auth="Application"
           type="javax.sql.DataSource"
            maxActive="10"
           maxIdle="4"
           maxWait="100"
            username="snort"
           driverClassName="org.postgresql.Driver"
            url="jdbc:postgresql:snort"
  />
  <Environment name="dataSource" value="jdbc/snort" type="java.lang.String" description="Data
source"/>
  <Environment name="snortHost" value="192.168.1.100" type="java.lang.String" description="Host IP
address running Snort IPS"/>
  <Environment name="restartCmd" value="/usr/bin/ssh 192.168.1.100 /usr/bin/sudo
/opt/snort/bin/snort.sh > /tmp/restart.log 2>&amp;1 &amp;"
     type="java.lang.String" description="Restart command"/>
  <Environment name="etblockUpdateCmd" value="/usr/bin/ssh 192.168.1.100 /usr/bin/sudo
/opt/snort/iptables/update_iptables.sh > /dev/null 2>&amp;1 &amp;"
     type="java.lang.String" description="ET Block rules update command"/>
  <Environment name="srramImportCmd" value="/usr/bin/sudo /opt/tomcat6/webapps/snort/WEB-
INF/cgi/srram_import.sh > /dev/null 2>&amp;1 &amp;"
     type="java.lang.String" description="Download rules command"/>
  <Environment name="rulesImportCmd" value="/usr/bin/sudo /opt/tomcat6/webapps/snort/WEB-
INF/cgi/rules_import.pl > /dev/null 2>&amp;1 &amp;"
     type="java.lang.String" description="Import downloaded rules command"/>
  <Environment name="descImportCmd" value="/usr/bin/sudo /opt/tomcat6/webapps/snort/WEB-
INF/cgi/loaddesc.pl > /dev/null 2>&amp;1 &amp;"
     type="java.lang.String" description="Import downloaded rules descriptions command"/>
  <Environment name="checkBlockCmd" value="/usr/bin/ssh 192.168.1.100 /usr/bin/sudo
/opt/snort/bin/checkblock.sh"
     type="java.lang.String" description="Check block host netfilter rule command before
parameter"/>
  <Environment name="blockCmd" value="/usr/bin/ssh 192.168.1.100 /usr/bin/sudo
/opt/snort/bin/block.sh"
     type="java.lang.String" description="Block host by netfilter rule command before parameter"/>
  <Environment name="unblockCmd" value="/usr/bin/ssh 192.168.1.100 /usr/bin/sudo
/opt/snort/bin/unblock.sh"
     type="java.lang.String" description="Unblock host by deleting netfilter rule command before
parameter"/>
  <Environment name="loadSqAccLogCmd" value="/usr/bin/sudo /opt/tomcat6/webapps/snort/WEB-
INF/cgi/loadacclog.pl > /dev/null 2>&amp;1 &amp;"
     type="java.lang.String" description="Load Squid access log into database command"/>
</Context>
```

Для обеспечения соединения приложения, работающего на сервере приложений, с сервером IDS по протоколу SSH требуется настройка SSH на аутентификацию по ключам или иным способом, не требующим интерактивного ввода пароля авторизованного на управление демоном Snort пользователя. Поскольку пользователь, под учетной записью которого работает сервер приложения, обычно не является суперпользователем, то необходимо использовать какой-то механизм эскалации полномочий, обычно это программа sudo, также требующая специальных настроек.

В связи с широким разнообразием возможных конфигураций и большим количеством окружающего программного обеспечения, функциональность которого в той или иной мере задействована, мы пока не создали установочной программы. Вероятно, это вообще неподъемная задача. Поэтому специалист, проводящий установку, помимо знания собственно Snort, должен быть знаком с общей структурой каталогов используемой операционной системы, схемой автозагрузки подсистем и демонов (в случае OpenSuse это /etc/init.d[14], /etc/sysconfig), SSH, sudo, iptables, основными командами Linux, такими как wget, curl, sed, а также более сложными приложениями, такими как Squid, Tomcat, Apache или другим применяемым HTTP-сервером, PostgreSQL, и хотя бы в общих чертах разбираться в следующих языках разметки: XML (схема JSP и др.), HTTP и языках программирования: Bourne Shell (вариант bash), Perl, Java, JavaScript, на которых написан весь комплекс наших

[14] Comment Conventions for Init Scripts
http://refspecs.freestandards.org/LSB_3.1.0/LSB-Core-generic/LSB-Core-generic/initscrcomconv.html

приложений, включая интерактивную часть — новую консоль управления IDS Snort.

Описание реализации

Начальная страница новой консоли управления IDS (см. рис. 5), на которую выведено меню системы, дает представление об основной заложенной в это приложение функциональности.

Мы вкратце перечислим функции системы:

- загрузка сигнатур из Интернета (источники — snort.org, emergingthreats.net и др.), источники задаются в отдельной таблице с именем snort_rules_load и могут быть произвольно включены или выключены через интерактивную форму ввода;
- импорт разархивированных сигнатур в БД с учетом уже имеющейся в ней информации, что позволяет отслеживать изменения в наборах сигнатур;
- импорт описаний сигнатур;
- обновление правил брандмауэра из набора EmergingThreats;
- просмотр и редактирование сигнатур, изменение типа реакции IDS на них, с сортировкой и отбору по категориям, а также произвольным фильтрам с синтаксисом SQL;
- выгрузка отредактированного списка сигнатур и перезагрузка IDS непосредственно из консоли; контроль за ходом загрузки и возможными ошибками; предупреждение ситуаций прекращения доступа к внешней сети в результате ошибок Snort;

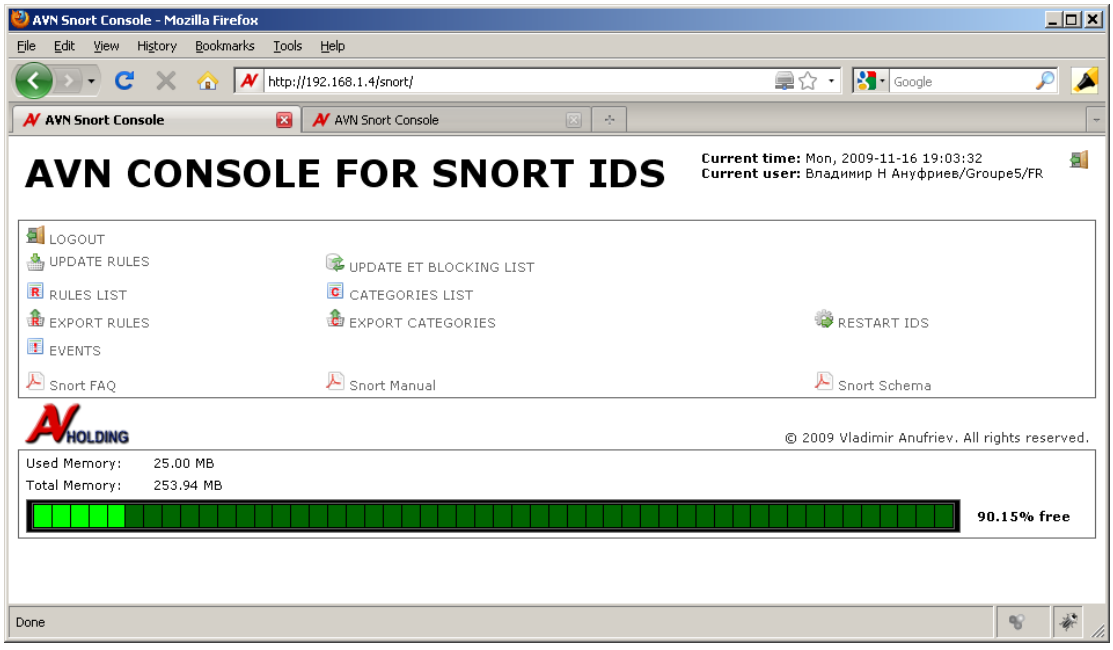

Рис. 5. Стартовая страница новой консоли управления IDS с меню

- просмотр информации о событиях, отбор событий по произвольному фильтру с синтаксисом SQL; получение информации о задействованных узлах и сигнатурах; подробные сведения о пакетах, на которые отреагировала IDS, блокировка и разблокировка узлов на брандмауэре непосредственно из консоли; автоматический поиск узлов в локальной сети по перехваченным пакетам и журналам прокси серверов.

Основная рабочая страница нашего приложения показывает автоматически или вручную обновляемый список последних событий, считываемых из БД. Пример приведен на рисунке 6. Мы постарались не перегружать это окно избыточной информацией, так как подробности можно получить, кликнув по полю с описанием сигнатуры. На следующем рисунке 7 представлена информация о событии из категории EMERGING-COMPROMIZED-BLOCK. На этом примере можно рассмотреть значительную часть функциональности, заложенной в систему. Во-первых, как можно увидеть, данная сигнатура задействует связь с брандмауэром через сервер SnortSam, на что указывает фраза «fwsam: src, 24 hours;». В результате срабатывания этой сигнатуры серверу SnortSam была передана информация, на основе которой были созданы новые правила в брандмауэре сервера, на котором выполняется демон snortsam (в общем случае серверов snortsam может быть несколько и работать они могут не только на машине, на которой работает IDS):

```
-A forward_fwsam -d 61.151.238.131/32 -i br0 -j fwsam_drop
-A input_fwsam -s 61.151.238.131/32 -i br0 -j fwsam_drop
```

Эти правила заблокируют любой трафик между корпоративной сетью и хостом 61.151.238.131 на 24 часа. О том что блокировка в тот момент, когда мы просматриваем информацию о событии, действует, свидетельствует появление кнопки Unblock в правой части окна рядом с меткой Src. Если нажать на эту кнопку, то правила будут удалены, и трафик будет разблокирован. При нажатии на кнопку Permanent Block трафик будет заблокирован без ограничения по времени. Программа проверяет блокируемые адреса на принадлежность к собственной сети, таким образом, заблокировать собственные серверы не удастся.

На этой же странице можно изменить реакцию на данную сигнатуру, в частности деактивировать ее (DISABLE)[15]. С некоторыми ограничениями можно произвольным образом отредактировать сигнатуру, но ошибки в этом процессе крайне нежелательны, так как Snort практически не умеет обрабатывать ошибок в сигнатурах и конфигурационных файлах. Для того чтобы изменения вступили в силу, нужно перегрузить IDS (это делается в отдельном окне программы), так как Snort не умеет динамически изменять загруженный в него набор сигнатур.

Дополнительной функцией в данном окне является изменение реакции на все сигнатуры данной категории одновременно. Повторим, что для того, чтобы изменения вступили в силу, требуется перезагрузка Snort.

Эта статья не является документацией на систему, поэтому не на всей функциональности будем останавливаться. Естественно, что по всем, участвующим в событиях IP-адресам можно получить справку через сервис whois, а также проверить их на включенность в черные списки. Так же мы видим максимально подробную информацию о сигнатуре и можем перейти по представленым в описании сигнатуры ссылкам для получения дополнительных сведений. Выбранная сигнатура недостаточно показательна в этом смысле, так как сигнатуры из набора EmergingThreats не содержат описаний, в отличие от поставляемых Sourcefire VRT (Vulnerability Research Team).

[15] Для того чтобы сигнатура с атрибутом fwsam сохраняла работоспособность необходимо, чтобы ее тип реакции был alert.

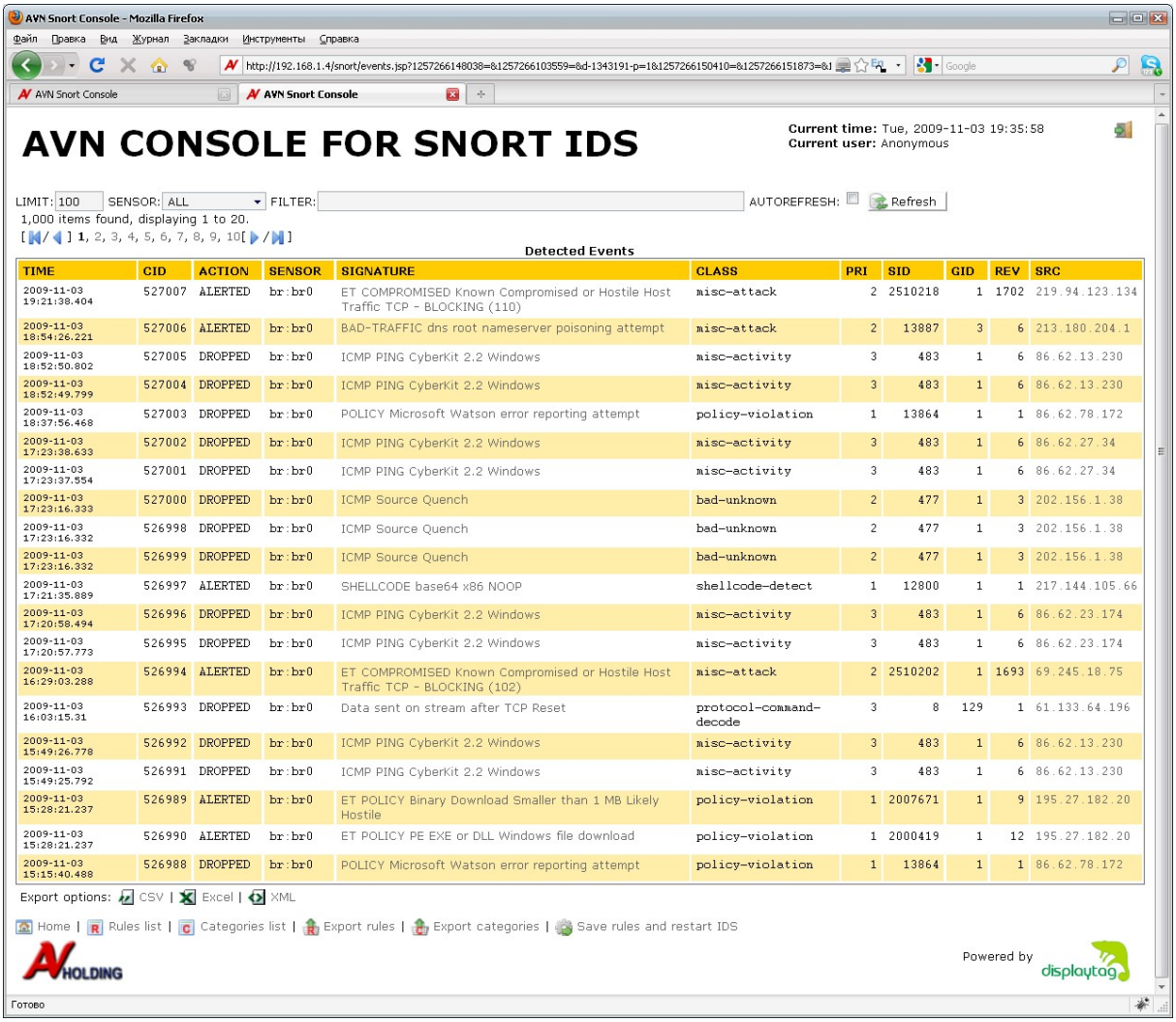

Рис. 6. Основная страница со обновляемым списком событий

Для получения информации сервиса whois и о черных списках мы пока используем функциональность внешних сайтов. Пример вывода такой информации для IP 161.151.238.131 представлен на рисунке 8.

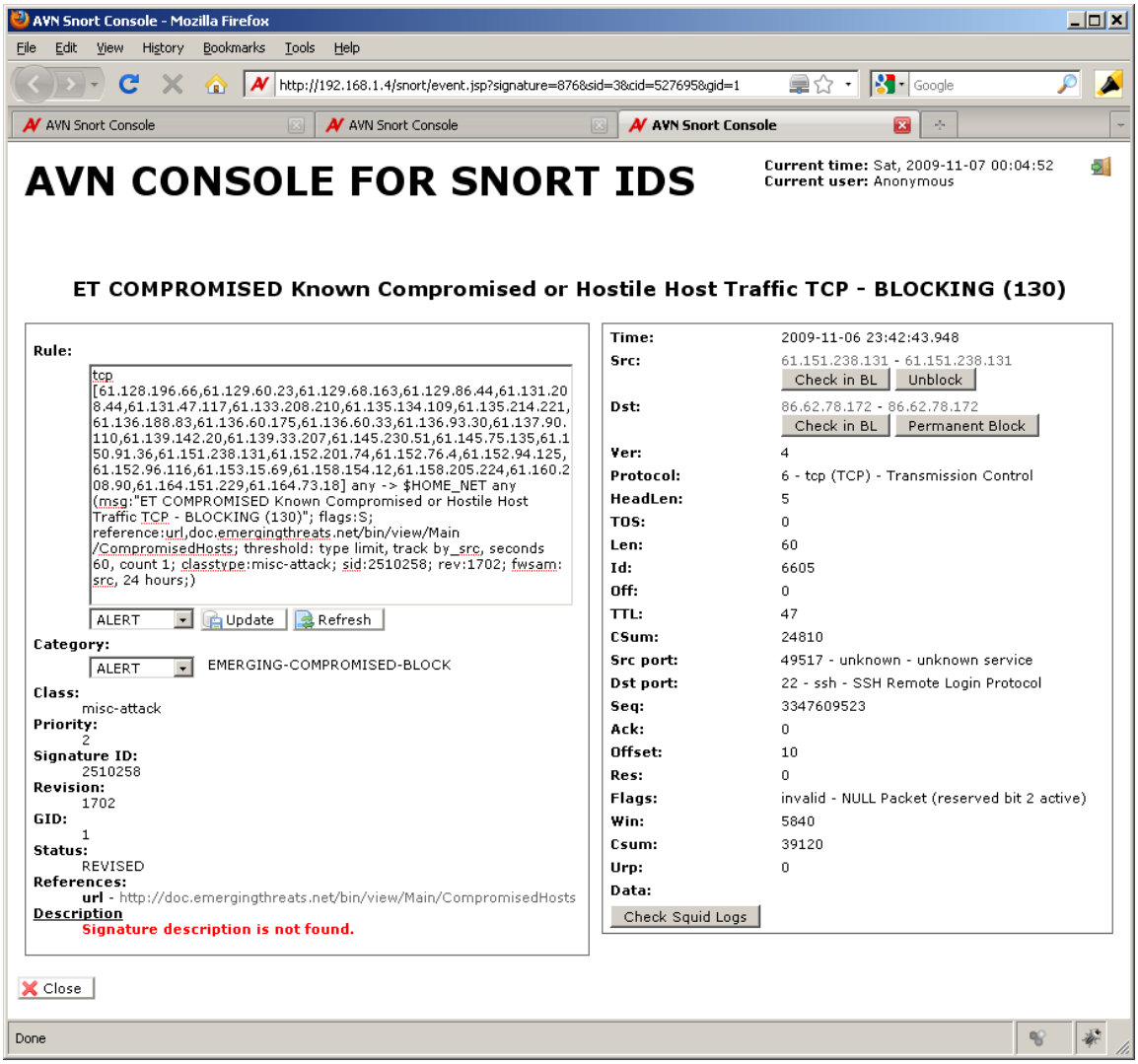

Рис. 7. Пример окна с информацией о событии

Для дополнения общей картины приведем еще пару иллюстраций. На рисунке 9 отражена информация о событии, связанном с сигнатурой из набора Sourcefire VRT. Здесь выведена подробная информация о потенциальной уязвимости, почерпнутая из соответствующего файла описания, импортированного в БД. Видно, также, что реакцией Snort на это событие была блокировка пакета (drop), то есть пакет был отброшен и не передался на интерфейс другого конца моста. Такая реакция при импорте новых сигнатур является реакцией по-умолчанию. Однако, данная реакция применима далеко не ко всем сигнатурам. Выше мы рассмотрели сигнатуру, срабатывание которой требует не отбрасывания пакета, а добавления нового правила в цепочку правил брандмауэра. Такие сигнатуры характеризуются наличием фразы с ключевым словом fwsam. Есть и другие исключения. Например, сигнатура может не свидетельствовать о пакете, несущем потенциальную опасность, а всего лишь устанавливать некоторый флаг, необходимый для анализа последущих пакетов. На основании таких сигнатур IDS выставляет соответствующий флаг, но других активных действий не предпринимает. Эти сигнатуры стандартно помечаются флагом noalert (flowbits: noalert;). Таким сигнатурам программа импорта по-умолчанию устанавливает правило обработки pass.

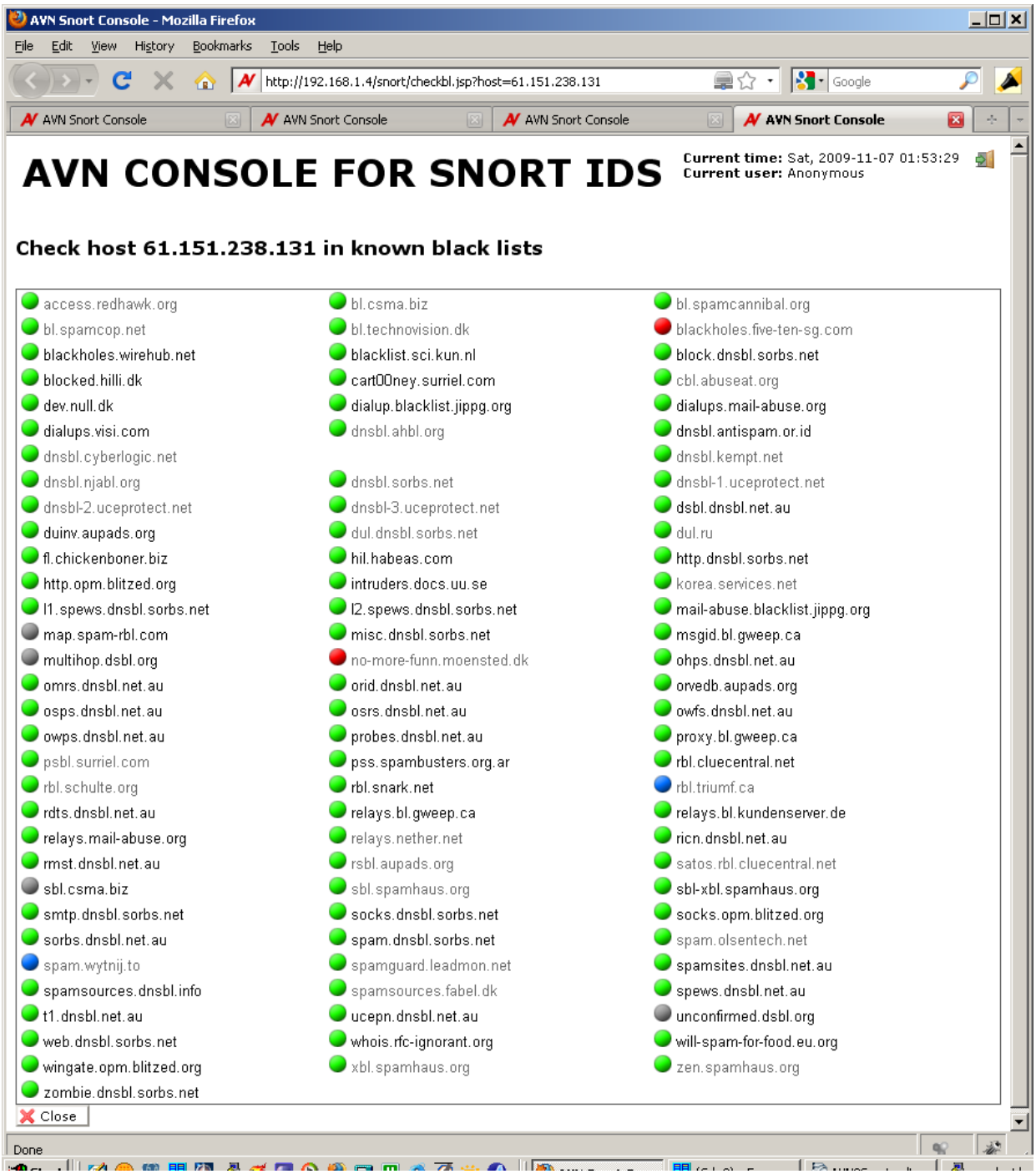

Рис. 8. Страница, показывающая информацию о включенности заданного IP в известные черные списки

На рисунке 10 представлена выборка из загруженного в БД набора сигнатур. Новые сигнатуры помечены как NEW, неизмененные со времени предыдущего обновления — как WORKING, удаленные (отсутствующие в последнем обновлении) — как DELETED, а новые версии сигнатур — как REVISED. На основании этих пометок, используя механизм фильтрации, основанный на синтаксисе SQL фразы WHERE оператора SELECT, можно проанализировать, какие изменения сделаны во взятом из Интернета обновлении набора сигнатур.

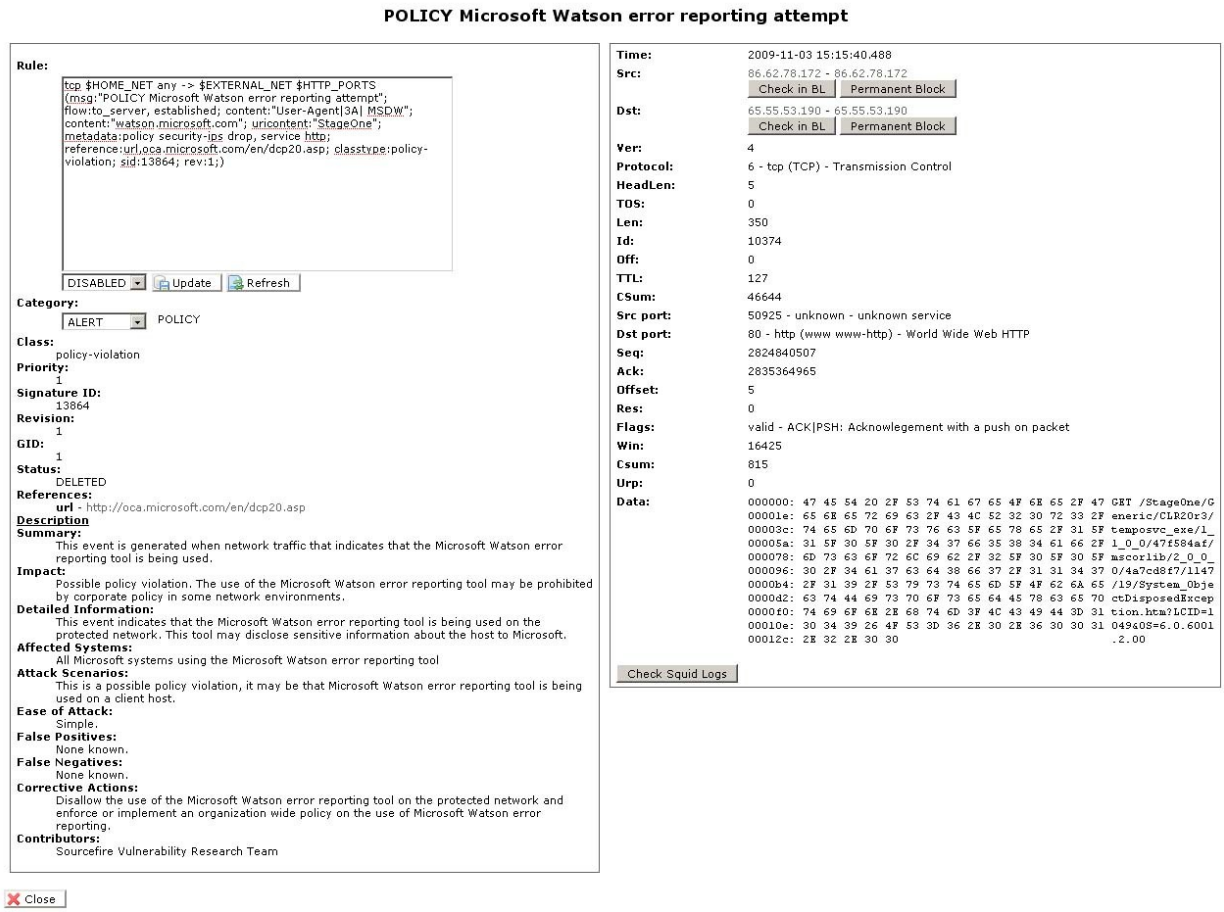

Рис. 9. Пример события, связанного с сигнатурой из набора Sourcefire VRT

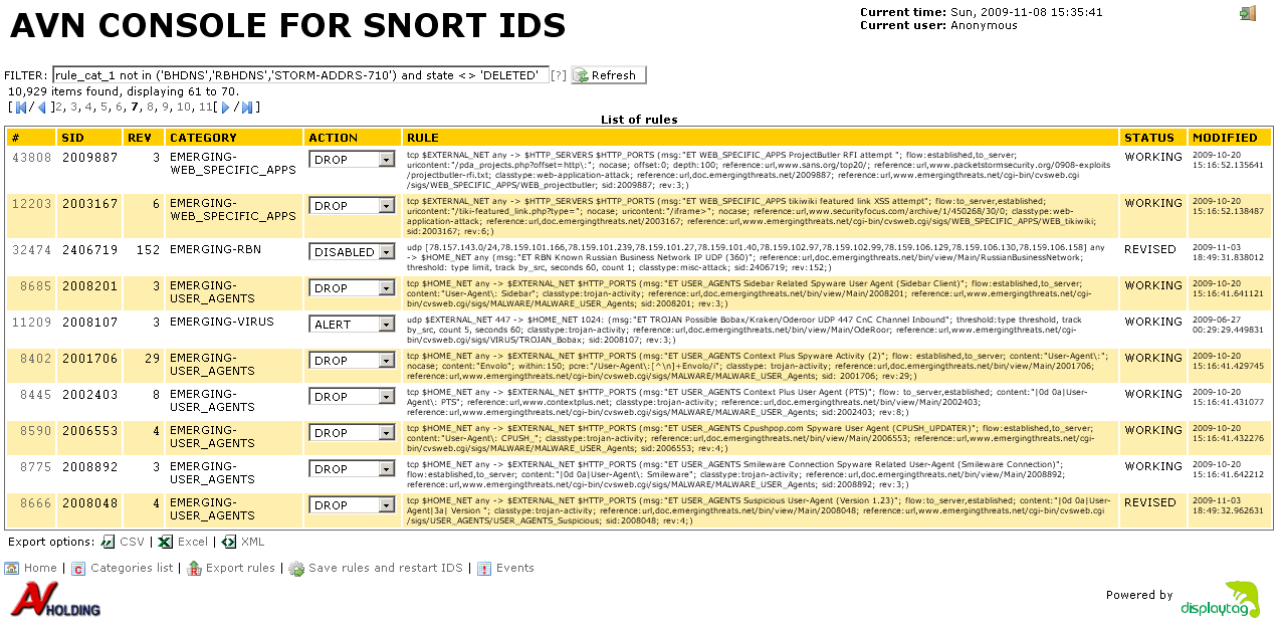

Рис. 10. Окно с отфильтрованным фрагментом списка сигнатур

Здесь же можно в некоторой степени отредактировать сигнатуры, используя возможности, описанные выше при рассмотрении информации о событии. После загрузки новых версий сигнатур можно посмотреть на их изменения в разрезе категорий. Категория — это имя файла, содержащего сигнатуры, сгруппированные по области действия. Страница со списком категорий представлена на рисунке 11.

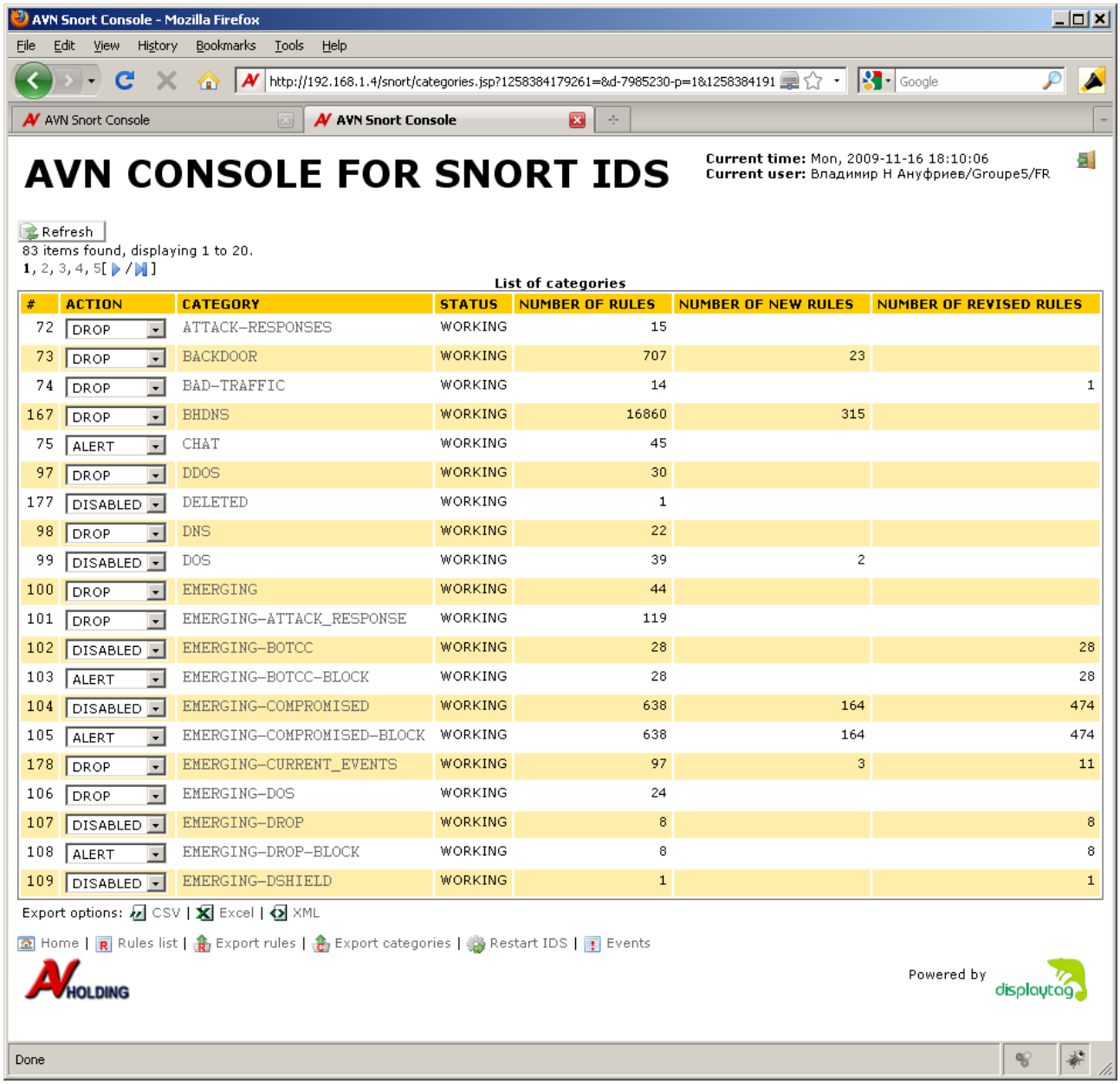

Рис. 11. Страница со списком категорий сигнатур

Для полноты картины нужно показать какую-нибудь из технических страниц, связанных с управлением IDS. На рисунке 12 представлена страница с протоколом перезагрузки IDS. Отметим, что все страницы в нашей консоли запрограммированы с применением технологии Ajax, и информация отображается по мере ее поступления от управляющих скриптов.

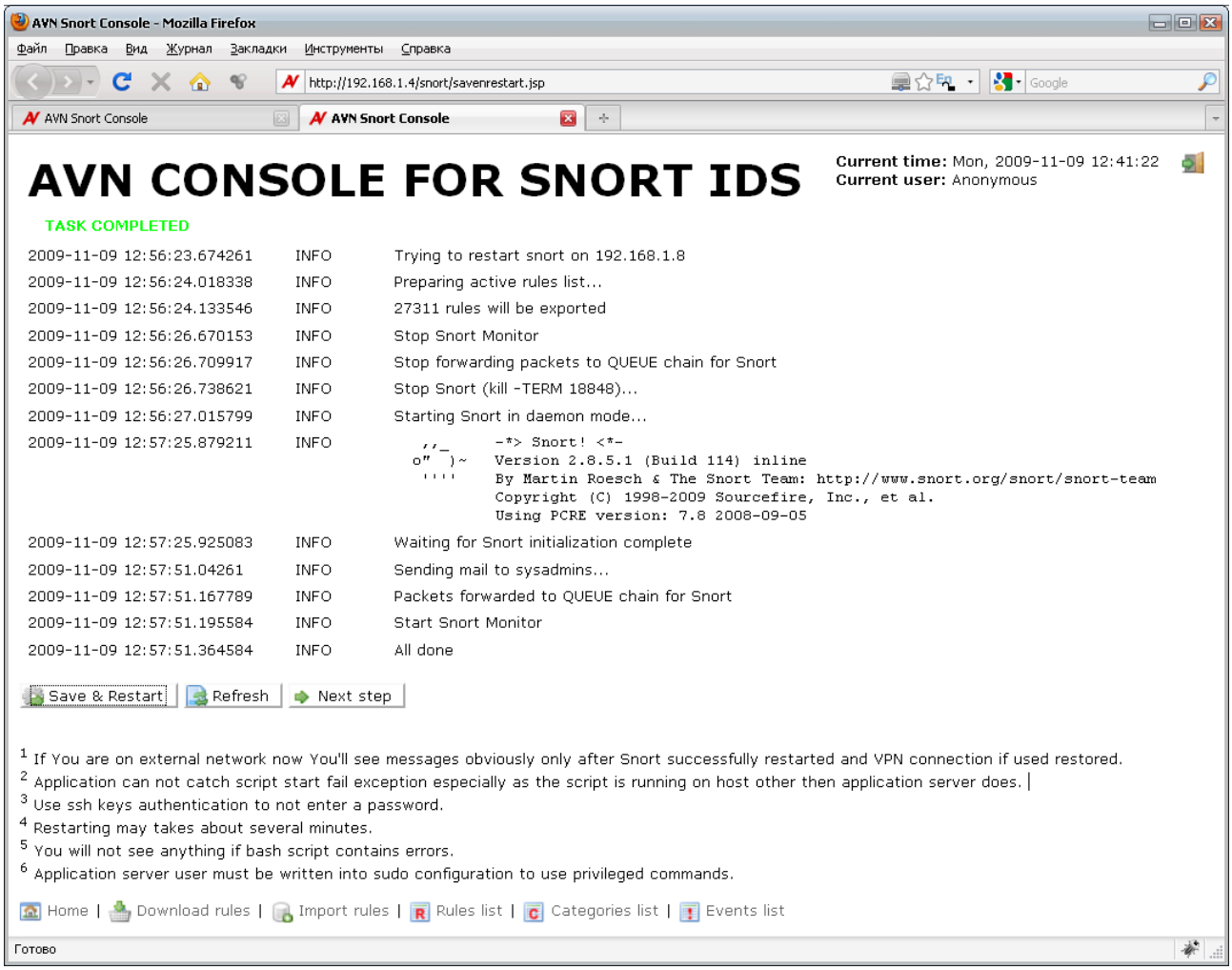

Рис. 12. Страница с протоколом перезагрузки IDS

Заключение

В данной статье рассмотрена реализация системы обнаружения и предупреждения вторжений на базе IDS Snort в комплексе с брандмауэрами Linux, специально разработанной консолью управления и другими программными средствами. Новая консоль управления реализует функциональность, необходимую в ежедневной работе службы информационной безопасности, и обладает следующими основными преимуществами перед стандартной консолью BASE:

- позволяет управлять набором сигнатур, загруженных в систему обнаружения вторжений, произвольно изменять выбранные сигнатуры;
- позволяет загружать обновления сигнатур и перезагружать Snort;
- в некоторой степени позволяет управлять другими системами обеспечения безопасности, в частности, брандмауэром Linux, главным образом, в связи с событиями IDS;
- осуществляет поиск в журнале кэширующего прокси сервера на базе Squid для определения, какая машина из внутренней (локальной или DMZ) сети посылает пакеты, на которые реагирует IDS;

- различает типы сигнатур (реакций на сигнатуры) Snort (pass, drop, sdrop, reject, alert, log), показывает оператору реакцию IDS на то или иное событие (drop/alert) в случае, когда используется слегка модифицированный модуль spo_database;
- наилучшим образом вписывается в решение на базе IDS Snort, включенной в режиме «прозрачного» моста.

avn@avnsite.com
2009-11-06