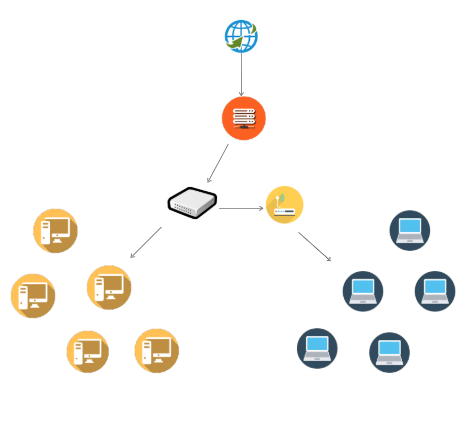
**Лабораторная работа № 9.**

Системы обнаружения вторжений.

1.Постановка задачи: Спроектировать физическую сеть для заданного расположения узлов

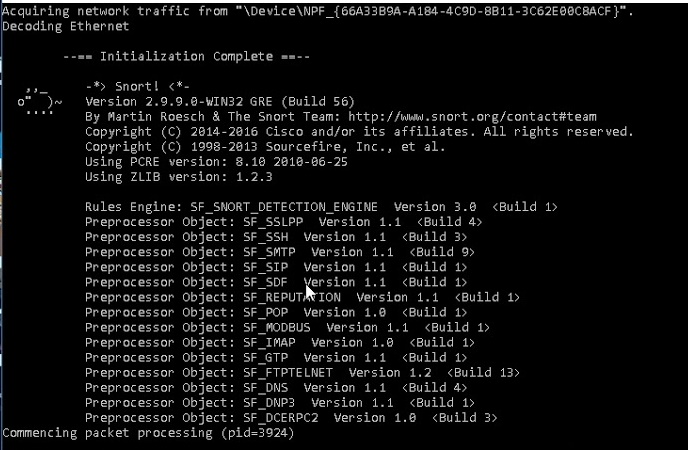
Результат выполненной работы:

Схема компьютерного класса:



2.Постановка задачи: Установка и настройка системы IDS (системы обнаружения вторжений) Snort.

Результат выполненной работы:



Для начала скачаем Snort с сайта www.snort.org.Вот прямая ссылка на последнюю версию на данный момент http://www.snort.org/dl/binaries/linux/snort-1.9.1-1snort.i386.rpm. Также существуют различные модификации Snort, например с поддержкой MySQL, postgresql, snmp, все это вы можете скачать с этого же сайта, а наш вариант программы я выбрал, как самый легкий в установке.

Установка:

rpm –i snort-1.9.1-1snort.i386.rpm

После этого все необходимые файлы будут скопированы в систему.

Настройка системы:

Перейдем в директорию /etc/snort, здесь вы можете найти базы сигнатур (точнее их можно назвать правилами, по которым Snort определяет вредный трафик) и несколько файлов конфигурации, нам нужен snort.conf. Тут настраиваем переменные переменные типа HOME\_NET, EXTERNAL\_NET и другие... В самом конце конфигурационного файла идут подключаемые сигнатуры, ненужные можно закомментировать, для повышения производительности.

Пример конфига:

###################################################

# Шаг #1: Установка переменных касающихся сети

# Измените IP, на адреса вашей локальной сети

# Можно указать несколько диапазонов, разделяя их запятыми

var HOME\_NET 192.168.168.0/24

var EXTERNAL\_NET !$HOME\_NET

var DNS\_SERVERS $HOME\_NET

var SMTP\_SERVERS $HOME\_NET

var HTTP\_SERVERS $HOME\_NET

var SQL\_SERVERS $HOME\_NET

var TELNET\_SERVERS $HOME\_NET

var ORACLE\_PORTS 1521

var HTTP\_PORTS 80

var SHELLCODE\_PORTS !80

# Путь к сигнатурам

var RULE\_PATH /etc/snort

#Подключаем необходимые файлы содержащие классификацию обнаруженной атаки и ссылки на

# багтраки

include classification.config

include reference.config

###################################################

# Шаг #2: Настройка механизма обнаружения атак

preprocessor frag2

preprocessor stream4: detect\_scans, disable\_evasion\_alerts

preprocessor stream4\_reassemble

preprocessor http\_decode: 80 unicode iis\_alt\_unicode double\_encode iis\_flip\_slash full\_whitespace

preprocessor rpc\_decode: 111 32771

preprocessor portscan: $HOME\_NET 4 3 portscan.log

# Мне пришлось добавить эту опцию, из за некоторых специфических программ, используемых в моей

# сети, из за которых часто случались ложные срабатывания

preprocessor portscan-ignorehosts: 192.168.168.0/24

preprocessor arpspoof

preprocessor conversation: allowed\_ip\_protocols all, timeout 60, max\_conversations 32000

preprocessor portscan2: scanners\_max 3200, targets\_max 5000, target\_limit 5, port\_limit 20, timeout 60

####################################################################

# Шаг #3: Указываем какие сигнатуры нам нужны

include $RULE\_PATH/bad-traffic.rules

include $RULE\_PATH/exploit.rules

include $RULE\_PATH/scan.rules

include $RULE\_PATH/finger.rules

include $RULE\_PATH/ftp.rules

include $RULE\_PATH/dos.rules

include $RULE\_PATH/ddos.rules

include $RULE\_PATH/dns.rules

include $RULE\_PATH/web-cgi.rules

Запись Snort в inittab для того, чтобы Snort запускался вместе с системой

2. Добавление собственных сигнатур

Snort – это очень гибкий и удобный в настоке IDS. Одно из его качеств позволяет нам самим добавлять сигнатуры атак (или как я уже говорил, это больше похоже на правила). Такие правила у нас лежат в файлах \*.rules. Синтаксис правил довольно прост:

ACTION PROTO IP\_ADDR1 PORT1 DIRECTION IP\_ADDR2 PORT2 [ (OPTIONS) ]

ACTION:

Поле Action имеет три основных директивы, которые определяют действия, при обнаружении сетевого пакета, соответствующего некоторому правилу: pass, log и alert.

pass - игнорировать пакет

log - пакет должен быть передан процедуре журналирования, для записи в файл журнала

alert генерирует уведомление об обнаружении пакета, удовлетворяющего правилу

PROTO:

Протокол пакета, может иметь значения tcp,udp,icmp

IP\_ADDR:

Как это понятно из названия опции, это поле означает IP адрес. any позволяет задать все возможные адреса. Символ ! инвертирует условие, т.е. !192.168.168.0/24 означает любой не принадлежащий подсети 192.168.168.0/24. Можно перечислять несколько IP адресов через запятую

PORT:

Кроме единственного номера порта можно задать диапазон портов через двоеточие, например, 6000:6010, символ ! инвертирует условие, а any обозначает все порты

DIRECTION:

Определяет направление движения пакета:

-> (одностороннее) - правило будет применяться только к пакетам, идущим с IP\_ADDR1 на IP\_ADDR2;

<> (двустороннее) - направление движения пакета роли не играет

OPTIONS

Заключаемые в круглые скобки параметры являются необязательной частью правила, но они определяют текст уведомляющего об угрозе сообщения, задают дополнительные действия при срабатывании правила и дополнительные условия на соответствие анализируемых пакетов данному правилу. Параметры отделяются друг от друга точкой с запятой, а ключевое слово параметра отделяется от его аргумента двоеточием.

Параметры, задающие дополнительные условия на соответствие правилу:

ttl - задает значение поля TTL в заголовке IP-пакета;

tos - задает значение поля TOS в заголовке IP-пакета;

id - задает значение поля номера фрагмента в заголовке IP-пакета;

ipopts - задает значение поля параметров IP-пакета;

fragbits - задает биты фрагментации IP-пакета;

dsize - задает условия на размер IP-пакета;

flags - задает условия на наличие или отсутствие определенных TCP-флагов;

seq - задает номер сегмента TCP-пакета в последовательности;

ack - задает значение поля подтверждения в TCP-пакете;

itype - задает значение поля типа ICMP-пакета;

icode - задает значение поля кода ICMP-пакета;

icmp\_id - задает значение поля ICMP ECHO ID в ICMP-пакете;

icmp\_seq - задает номер ICMP ECHO пакета в последовательности;

content - задает искомый шаблон в содержимом пакета, а не в заголовке (шаблон можно задавать как в текстовом виде, так и в шестнадцатеричном);

content-list - этот параметр аналогичен параметру content за исключением того, что список искомых шаблонов берется из заданного файла;

offset - работает совместно с опцией content для определения смещения в пакете, с которого будет производиться анализ содержимого;

depth - аналогичен параметру offset и определяет положение в пакете, до которого будет производиться анализ содержимого;

nocase - отключает чувствительность к регистру при анализе содержимого пакета;

rpc - этот параметр позволяет более точно задать характеристики программных или процедурных вызовов RPC-сервисов.

Как можно заметить, перечисленные параметры позволяют создавать правила для перехвата практически любых пакетов, которые как-то могут угрожать безопасности. А если учесть, что Snort может перехватывать пакеты на канальном уровне, то его применение особенно интересно на хостах, защищенных файрволлом, так как отбрасываемые файрволлом пакеты все равно будут находиться в поле зрения Snort.

Параметры, значения которых имеют смысл при соответствии анализируемого пакета всем условиям:

msg - содержит текст сообщения;

logto - задает альтернативный файл для записи в него содержимого пакета;

session - этот параметр позволяет включить очень интересную возможность Snort - извлечение пользовательских данных из TCP-сессии, например, для последующего анализа того, какие команды вводил пользователь во время telnet-сессии;

resp - если пакет соответствует правилу, то Snort выполнит одно из указанных действий - например, закроет соединение, отправив TCP-RST-пакет одному из хостов.

react - блокирует заданные в правиле web-сайты, закрывая соединение с ними и/или отправляя заданное сообщение браузеру, с которого была предпринята попытка зайти на сайт.

Вот пара примеров построения собственных правил:

alert tcp !192.168.1.45/32 any -> 192.168.1.45/32 80 (msg:"IIS-\_vti\_inf";flags:PA; content:"\_vti\_inf.html"; nocase;)

Адресованные web-серверу пакеты, содержащие в себе запрос к файлу \_vti\_inf.html, рассматриваются как попытка воспользоваться одной из уязвимостей Internet Information Server, что вызовет при обнаружении таких пакетов генерацию сообщения об этом событии, а сам пакет запишется в лог-файл.

alert tcp any any <> any 6688 (msg:"Napster Client Data"; flags:PA; content:".mp3"; nocase; resp: rst\_all)

В случае обнаружения запроса к Napster-серверу соединение принудительно закрывается. Как видно, при помощи Snort организовать фильтрацию нежелательного трафика можно более эффективно, чем просто закрывая соответствующие порты на файрволе, поскольку имеется возможность ввести дополнительное условие на содержимое пакетов.