

Содержание:

Введение	1
Описание эксперимента по подмене произвольного исполняемого файла операционной системы	2
Разработка bash-сценария для автоматического мониторинга целостности исполняемых файлов с информированием по электронной почте.	3
Описание процесса запуска bash-сценария через планировщик GNU/Linux	4
Описание процесса тестирования системы	5
Заключение	7
Список использованной литературы	8

Введение

Компьютеры в наше время выполняют множество задач. Практически никто сейчас не работает без компьютера. Рынок IT процветает и развивается, появляются новые интернет-проекты и сервисы, люди все больше времени проводят в сети. Сегодня массовое применение персональных компьютеров, к сожалению, оказалось связанным с появлением программ-вирусов, препятствующих нормальной работе компьютера, разрушающих файловую структуру дисков и наносящих ущерб хранимой в компьютере информации.

В вопросах исполняемых файлов, то есть, если по простому, обычных программ, Ubuntu кардинально отличается от Windows. Итак, что же такое исполняемый файл с точки зрения Ubuntu? Фактически это любой файл, который помечен, как исполняемый и который Ubuntu сможет запустить на выполнение. Означает это вот что: у каждого файла есть специальное свойство-переключатель, никак не зависящее ни от имени, ни от содержимого, отвечающее за исполняемость. Если файл помечен, как исполняемый, то вообще говоря он таковым и является, а если не помечен - то это обычный файл с данными и напрямую запустить его на выполнение нельзя. Другое дело, что не любой файл, помеченный как исполняемый, Ubuntu сможет выполнить, хотя в арсенале Ubuntu есть масса методов запуска файлов с совершенно различным содержимым.

Подмена файлов в системе при атаке происходит с помощью:

Паразитические вирусы (parasitic). К таковым относятся все вирусы, которые изменяют содержимое файла, но при этом оставляют его работоспособным. Основными типами таких вирусов являются вирусы, записывающиеся в начало файлов, в конец файлов и в середину файлов.

Вирусы-компаньоны (companion). Данный способ подразумевает создание файла-двойника, при этом код файла-жертвы не изменяется. Обычно вирус изменяет расширение файла (например, с .exe на .com), потом создает свою копию с именем, идентичным имени файла-жертвы, и дает ему расширение, тоже идентичное. Ничего не подозревающий пользователь запускает любимую программу и не подозревает, что это вирус. Вирус, в свою очередь, заражает еще несколько файлов и запускает программу, затребованную пользователем.

Исходя из этого мы приходим к выводу о том, что атаку, направленную на подмену исполняемых файлов нужно как можно быстрее выявить и сообщить об этом администратору сети, что я и постараюсь сделать в своем курсовом проекте.

Описание эксперимента по подмене произвольного исполняемого файла операционной системы

Для проверки работоспособности bash-сценария необходимо провести эксперимент по подмене произвольного исполняемого файла.

Атака, направленная на подмену файлов подразумевает внедрение сторонних команд или данных в работающую систему и осуществляется злоумышленниками при помощи вредоносного ПО.

Вредоносное ПО - любое программное обеспечение, предназначенное для получения несанкционированного доступа к вычислительным ресурсам самой ЭВМ или к информации, хранимой на ЭВМ, с целью несанкционированного использования ресурсов ЭВМ или причинения вреда, ущерба владельцу информации, или владельцу ЭВМ, путем копирования, искажения, удаления или подмены информации.

В нашем случае для воссоздания вторжения в систему и изменения исполняемых файлов был написан сценарий на bash следующего содержания:

```
#!/bin/bash  
file="test"  
fileOrig="/bin/nano"  
fileCopy="/bin/nano1"  
fileChanged=$fileOrig  
sudo mv $fileOrig $fileCopy  
echo $file > $fileChanged
```

В результате мы воссоздали ситуацию, где при атаке на систему был подменен конфигурационный файл и теперь можем приступить к bash-сценарию, который покажет нам изменения файла “ ” и уведомит об этом администратора по электронной почте.

Разработка bash-сценария для автоматического мониторинга целостности исполняемых файлов с информированием по электронной почте.

Для проверки всех исполняемых файлов системы было решено использовать утилиту `debsums`, которая сравнивает контрольные суммы всех исполняемых файлов с теми, что были при установке этих файлов в пакетах. Для отправки почты был создан тестовый почтовый ящик и поставлен пакет `ssmtp` на виртуальную машину.

В итоге был разработан следующий bash-сценарий:

```
#!/bin/bash
mailto="nekeett@gmail.com" - почтовый адрес администратора
theme="Files has been changed" - тема письма
message="This files has been changed:"
file="fileChanged.txt" - имя временного файла
filesChanged=$(debsums -a | grep FAILED | awk -F ' ' '{print $1}') - запуск
утилиты debsums на проверку хэш-сумм всех исполняемых файлов, затем
отбор строк из результатов, в которых хэш-суммы не совпадают и вывод
списка с полным путем до каждого измененного файла.
```

```
    if [ -z '$filesChanged' ] - проверка на наличие данных в переменной
    'filesChanged'
    then
        exit 0 - в случае отсутствия данных в переменной сценарий
завершается
    fi
```

`echo "$message" > $file && echo "$filesChanged" >> $file` - запись заголовка и данных из переменной 'filesChanged' в файл fileChanged.txt

```
if [ ! -f $file ] -проверка на существование временного файла
then
  theme="Something wrong with your script."
  echo "$file does not exists. Check it out." | mail -s "$theme" $mailto -
если файл не существует - тема меняется и отправляется новое письмо на
почту администратора.
  exit 1 - завершение сценария с exitcode=1
fi
cat $file | mail -s "$theme" $mailto - отправка письма со списком
измененных файлов на электронную почту администратора

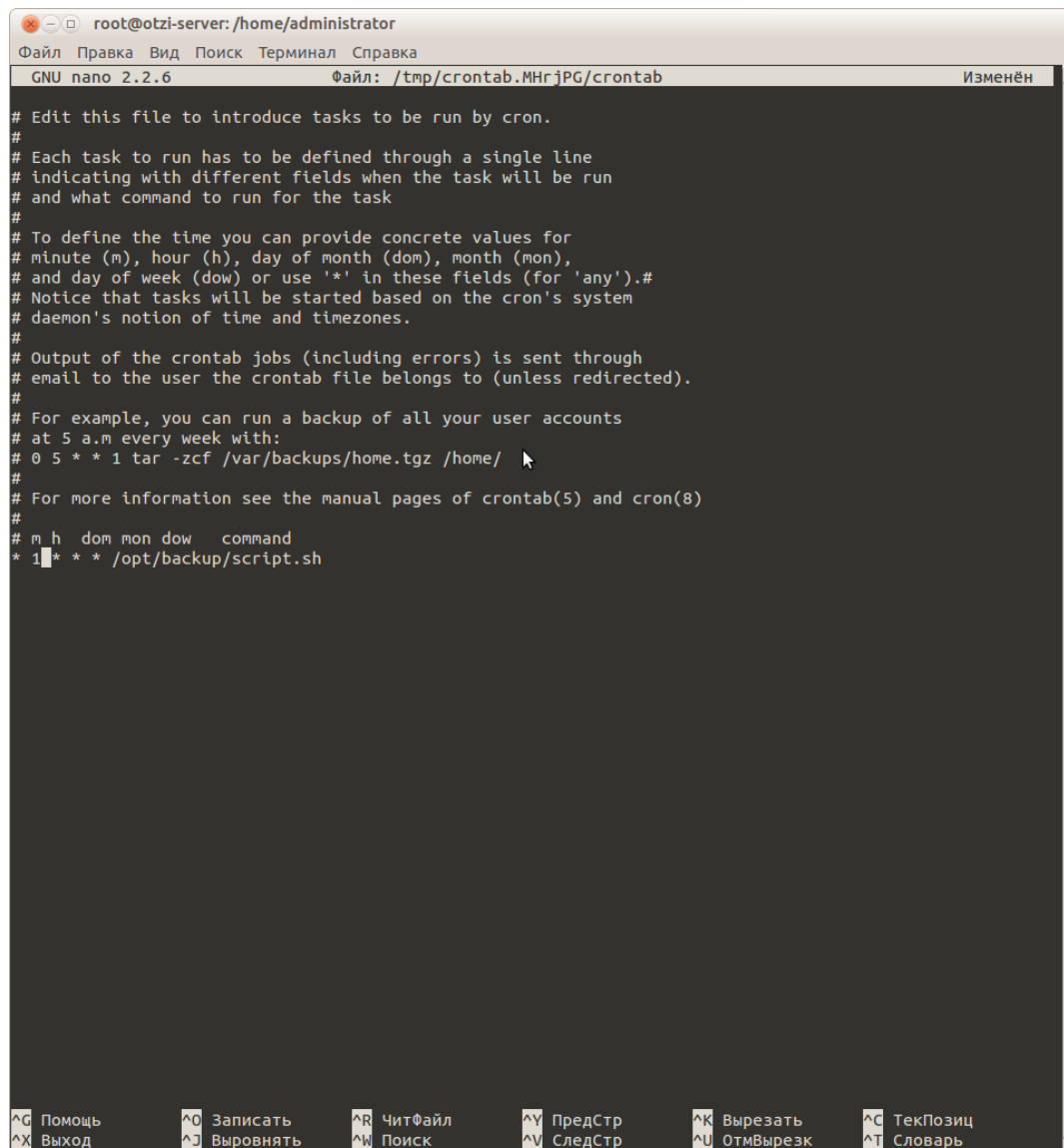
if [ "$?" == "0" ]
then
rm $file - если письмо отправилось успешно, то удалить файл
'fileChanged.txt'
fi
```

Описание процесса запуска bash-сценария через планировщик GNU/Linux

Автоматическое выполнение сценария будет осуществляться при помощи планировщика cron.

cron — классический демон (компьютерная программа в системах класса UNIX), использующийся для периодического выполнения заданий в определённое время. Регулярные действия описываются инструкциями, помещёнными в файлы crontab и в специальные директории.

Чтобы сделать запись о нашем файле воспользуемся утилитой crontab.



```
root@otzi-server: /home/administrator
Файл  Правка  Вид  Поиск  Терминал  Справка
GNU nano 2.2.6      Файл: /tmp/crontab.MHrjPG/crontab      Изменён

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
* 1 * * * /opt/backup/script.sh
```

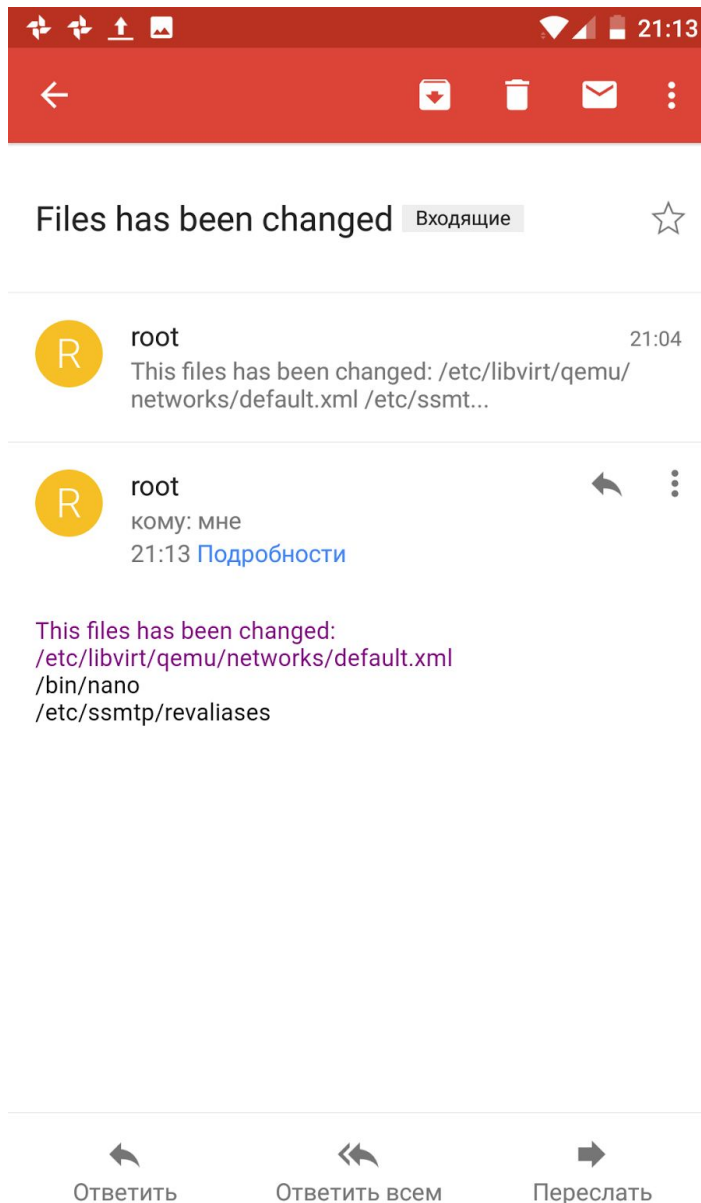
Откроем файл на редактирование командой `crontab -e` и впишем следующую строку в конец файла.

Затем сохраним файл и готово. Теперь планировщик cron будет раз в час запускать наш сценарий автоматически.

Описание процесса тестирования системы

В тестировании системы нам как раз пригодится файл, подмененный нами с помощью сценария.

Для тестирования мы запустим скрипт, в результате которого на электронную почту администратора придет письмо следующего содержания:



Как мы видим, файл “/bin/nano”, который мы подменили, появился в списке. И письмо успешно пришло на почту.

Теперь мы намеренно допустим ошибку в создании файла и протестируем оповещение администратора об ошибках.



После выполнения скрипта получаем на почту следующее сообщение:



Something wrong with your script.



Входящие



root

кому: мне

21:32 [Подробнее](#)



fileChanged.txt does not exists. Check it out.



Ответить



Ответить всем



Переслать

Заключение

В процессе написания курсового проекта был изучен сам процесс подмены исполняемого файла операционной системы GNU/Linux и был разработан универсальный bash-сценарий, задача которого периодически сверять контрольные суммы файлов с исходными и оперативно оповещать администратора об изменениях в исполняемых файлах при помощи сравнения контрольных сумм.

Контрольная сумма — некоторое значение, рассчитанное по набору данных путем применения определенного алгоритма и используемое для проверки целостности данных при их передаче или хранении. Также контрольные суммы могут использоваться для быстрого сравнения двух наборов данных на неэквивалентность: с большой вероятностью различные наборы данных будут иметь неравные контрольные суммы. Это может быть использовано, например, для обнаружения компьютерных вирусов. Несмотря на свое название, контрольная сумма не обязательно вычисляется путем суммирования.

В век информационных технологий защита личной информации и нормальной работоспособности персональных компьютеров сейчас, как никогда, актуальна. Все чаще в средствах массовой информации появляются сообщения о различного рода пиратских проделках компьютерных хулиганов, о появлении все более совершенных вредоносных программ и сценарии, вроде созданного нами, помогут максимально быстро узнать о вторжении в систему и предотвратить печальные последствия взлома.

Список использованной литературы

Андрей Робачевский — Операционная система UNIX

Дэвид Тейнсли - Linux и UNIX: программирование в shell. Руководство разработчика

Даниэл Дж. Баррет - Linux. Основные команды. Карманный справочник

Стивен Спейнауэр, Эллен Сивер - Linux. Справочник