

# ЛАБОРАТОРНАЯ РАБОТА №1.

Свидиров Кирилл, 11-902

27.04.2022

## Содержание

1	Общая информация	3
2	Постановка задачи	4
2.1	Суть задания	4
2.2	Цель упражнения	4
2.3	Используемые средства	4
3	Исходное состояние	5
4	Выполнение задания	6
4.1	Создание виртуальной машины Metasploitable	6
4.2	Смена паролей в Metasploitable	6
4.2.1	Вход в Metasploitable	6
4.2.2	Смена пароля для пользователя msfadmin	6
4.3	Обновление источников пакетов	6
4.3.1	Создание backup-файла репозитория	6
4.3.2	Просмотр списка репозитория	6
4.3.3	Отключение репозитория	6
4.3.4	Добавление репозитория из файла deb.txt в список репозитория	6
4.3.5	Обновить список репозитория для менеджера пакетов apt	7
4.3.6	Установка заголовочных файлов ядра	7
4.3.7	Установка утилиты Zip	7
4.4	Установка LiME	7
4.4.1	Скачивание архива с исходными текстами Lime	7
4.4.2	Распаковка и сборка Lime	7
4.5	Установка вспомогательных библиотек	7
4.5.1	Установка вспомогательных библиотек	7
4.5.2	Скачивание, настройка и установка библиотеки libdwarf	7
4.6	Скачивание, настройка и установка фреймворка Volatility	8
4.6.1	Скачивание Volatility	8
4.6.2	Создание файла с отладочной информацией модуля ядра Metasploitable – module.dwarf	8
4.6.3	Создание профиля Volatility для Metasploitable	8
4.7	Форензика	8
4.7.1	Создание дампа памяти с помощью LiME	8
4.7.2	Просмотр активности в памяти в момент снятия дампа с помощью Volatility	9
4.8	Оформление результатов работы	10

## 1 Общая информация

- ✦ Работу выполнил Свидиров Кирилл Андреевич, 11-902 группа.
- ✦ Название лабораторной работы – “ЛАБОРАТОРНАЯ РАБОТА №1. LIME И VOLATILITY. ФОРЕНЗИКА”.

## 2 Постановка задачи

### 2.1 Суть задания

С помощью LiME и libdwarf снять дамп оперативной памяти и проанализировать его, используя Volatility.

### 2.2 Цель упражнения





На практике увидеть, как и что можно узнать, имея физический доступ к консоли (с правами супер-пользователя). В этом задании мы поняли, что через консоль можно получить доступ ко всему, что в данный момент происходит на устройстве, анализируя оперативную память.

### 2.3 Используемые средства

- ✈ Vmware Workstation 16 Player - для запуска виртуальной машины с уязвимым образом Linux – Metasploitable.
- ✈ Metasploitable – устаревший и уязвимый образ Linux. Используется, чтобы свободно изучать существующие уязвимости операционной системы.
- ✈ LiME – загружаемый модуль ядра, позволяющий сделать дамп (содержимое рабочей памяти системы) операционной системы.
- ✈ libdwarf – библиотека для чтения дампа. Привносит свой формат файлов - .dwarf.
- ✈ Volatility – инструмент, позволяющий проанализировать дампы огромного количества профилей (систем, с которых был снят дамп).
- ✈ Zip – утилита для разархивации архивов. Нужна для установки используемых утилит / программ / библиотек.

### 3 Исходное состояние

Чистый (исходный) образ Metasploitable. В процессе выполнения лабораторной работы будут изменены:

-  Пароль пользователя
-  Список репозитория для загрузки библиотек (образ устаревший, так что многие ссылки уже недействительны)
-  Установка дополнительных модулей и утилит
-  Создание файлов

## 4 Выполнение задания

### 4.1 Создание виртуальной машины Metasploitable

- ✓ Предварительно был установлен VMware Workstation 16 Player.
- ✓ Далее была создана виртуальная машина с использованием файла виртуального жесткого диска Metasploitable.vmdk.

### 4.2 Смена паролей в Metasploitable

#### 4.2.1 Вход в Metasploitable

- ✓ Был осуществлён вход через пользователя msfadmin с паролем msfadmin.

#### 4.2.2 Смена пароля для пользователя msfadmin

- ✓ С помощью команды `[sudo su]` были получены суперправа для дальнейшей работы
- ✓ С помощью команды `[passwd msfadmin]` пароль пользователя был обновлён

### 4.3 Обновление источников пакетов

#### 4.3.1 Создание backup-файла репозитория

- ✓ С помощью команды `[cd /etc/apt]` был совершён переход в нужную директорию.
- ✓ Командой `[cp sources.list sources.list.BKP]` была создана копия файла `sources.list` с названием `sources.list.BKP`.
- ✓ Через команду просмотра и регулярного выражения `[ls -l sources.list*]` мы убедились, что копия действительно была создана.

#### 4.3.2 Просмотр списка репозитория

- ✓ С помощью команды `[grep “#” -v sources.list | head -20]` было выведено содержимое файла `sources.list` (первые 20 строк).

#### 4.3.3 Отключение репозитория

- ✗ Данный шаг был пропущен, т.к. на следующем шаге мы будем менять содержимое, и нам не нужно в итоге комментировать строки.

#### 4.3.4 Добавление репозитория из файла `deb.txt` в список репозитория

- (!) Данный файл отсутствует, поэтому были изменены существующие ссылки на актуальные.
- ✓ Командой `[vim sources.list]` был открыт файл репозитория.
- ✓ С помощью регулярного выражения `[%s/us.archive/old-releases/]` были изменены ссылки на актуальные.

#### 4.3.5 Обновить список репозиториев для менеджера пакетов apt

- ✓ С помощью команды `[apt-get update]` был обновлён список репозиториев для менеджера пакетов apt.

#### 4.3.6 Установка заголовочных файлов ядра

- ✓ С помощью команды `[uname -r]` была выведена версия ядра.
- ✓ С помощью команды `[apt-get install linux-headers-$(uname -r)]` были обновлены заголовки ядра linux для соответствующей версии.

#### 4.3.7 Установка утилиты Zip

- ✓ С помощью команды `[apt-get install zip]` была установлена соответствующая утилита.

### 4.4 Установка LiME

#### 4.4.1 Скачивание архива с исходными текстами Lime

- (!) Установить командой `wget` из лабораторной работы не получилось, так как ссылка в репозитории осталась устаревшей.
- ✓ С GitHub был скачан архив утилиты и передан на виртуальную машину через `scp`.

#### 4.4.2 Распаковка и сборка Lime

- ✓ С помощью команды `[unzip -x LiME-master.zip]` архив был распакован.
- ✓ С помощью команды `[cd LiME-master]` был осуществлён переход в распакованную директорию утилиты.
- ✓ С помощью команды `[cd src/]` был осуществлён переход в папку с ресурсами утилиты.
- ✓ С помощью команды `[make]` была выполнена сборка.

### 4.5 Установка вспомогательных библиотек

#### 4.5.1 Установка вспомогательных библиотек

- ✓ С помощью команды `[apt-get install libelfg0-dev]` через пакетный менеджер была установлена библиотека `libelfg0-dev`.

#### 4.5.2 Скачивание, настройка и установка библиотеки libdwarf

- (!) Установить командой `wget` из лабораторной работы не получилось, так как ссылка в репозитории осталась устаревшей.
- ✓ С данного сайта был скачан архив библиотеки и передан на виртуальную машину через `scp`.
- ✓ С помощью команды `[tar xf libdwarf-20140208.tar.gz]` архив был распакован.
- ✓ С помощью команды `[cd /var/tmp/dwarf-20140208]` был осуществлён переход в директорию с исходными файлами.

- ✓ С помощью команды `./configure` были сконфигурированы исходники для дальнейшей сборки.
- ✓ С помощью команды `make` была выполнена сборка.
- ✓ С помощью команды `cp dwarfdump/dwarfdump /usr/bin` файлы утилиты были перемещены в каталог исполняемых файлов.
- ✓ С помощью команды `which dwarfdump` было получено `[/usr/bin/dwarfdump]` => утилита была перенесена верно.

## 4.6 Скачивание, настройка и установка фреймворка Volatility

### 4.6.1 Скачивание Volatility

- (!) Установить командой `wget` из лабораторной работы не получилось, так как ссылка в репозитории осталась устаревшей.
- ✓ С данно го сайта был скачан архив библиотеки и передан на виртуальную машину через `scp`.
- ✓ С помощью команды `[tar zxvf volatility-2.3.1.tar.gz]` архив был распакован.

### 4.6.2 Создание файла с отладочной информацией модуля ядра Metasploitable – module.dwarf

- ✓ С помощью команды `[cd /var/tmp/volatility-2.3.1/tools/linux]` был осуществлён переход в директорию с утилитами фреймворка.
- ✓ С помощью команды `[make]` была выполнена сборка.
- ✓ С помощью команды `[ls -l module.dwarf]` была проверена успешность получения структуры ядра.

### 4.6.3 Создание профиля Volatility для Metasploitable

- ✓ С помощью команды `[cd /]` был совершён переход в корневую папку.
- ✓ С помощью команды `[zip /var/www/UBUNTU-MSF804.zip /var/tmp/volatility-2.3.1/tools/linux/module.dwarf /boot/System.map-2.6.24-16-server]` был подготовлен архив с профилями ядра Metasploitable
- ✓ С помощью команды `[ls -l /var/www/UBUNTU-MSF804.zip]` была проверена успешность создания архива.

## 4.7 Форензика

### 4.7.1 Создание дампа памяти с помощью LiME

- ✓ С помощью команды `[cd /var/tmp/LiME-master/src]` был совершён переход в директорию с собранным для ядра модулем.
- ✓ С помощью команды `[insmod lime-2.6.24-16-server.ko "path=/var/tmp/LiME-master/src/mem.img format=lime"]` был создан дамп памяти.



## 4.7.2 Просмотр активности в памяти в момент снятия дампа с помощью Volatility

- ✓ С помощью команды `[cd /var/tmp/volatility-2.3.1]` был осуществлён переход в каталог с фреймворком.
- ✓ С помощью команды `[mkdir ./volatility/profiles/]` была создана отдельная директория для архива с файлами профиля ядра.
- ✓ С помощью команды `[cp /var/www/UBUNTU-MSF804.zip /var/tmp/volatility-2.3.1/volatility/profiles/]` архив был перемещён в новый каталог.
- (!) Для следующего шага было необходимо обновить питон до версии 2.6, поэтому шаги будут расписаны ниже.
- ✓ официального сайта был скачан архив библиотеки и передан на виртуальную машину через scp.
- ✓ С помощью команды `[tar xf Python-2.6.tgz]` архив был распакован.
- ✓ С помощью команды `[cd Python-2.6]` был осуществлён переход в директорию распакованного архива.
- ✓ С помощью команды `[./configure]` были сконфигурированы исходники для дальнейшей сборки.
- ✓ С помощью команды `[make make install]` была выполнена сборка.
- ✓ С помощью команды `[ - update-alternatives --install /usr/bin/python python $(which python2.6) 1]` новая версия питона была установлена версией по умолчанию.
- ✓ С помощью команды `[./configure.]` были сконфигурированы исходники для дальнейшей сборки.
- ✓ С помощью команды `[./configure.]` были сконфигурированы исходники для дальнейшей сборки.
- (!) Для следующего шага было необходимо установить PyCrypto. Шаги установки будут представлены ниже.
- ✓ официального сайта был скачан архив библиотеки и передан на виртуальную машину через scp.
- ✓ С помощью команды `[tar xf pycrypto-2.6.1.tar.gz]` архив был распакован.
- ✓ С помощью команды `[cd pycrypto-2.6.1]` был осуществлён переход в директорию распакованного архива.
- ✓ С помощью команды `[./configure]` были сконфигурированы исходники для дальнейшей сборки.
- ✓ С помощью команды `[python ./setup.py build]` была выполнена сборка.
- ✓ С помощью команды `[python ./setup.py install]` была выполнена установка.
- ✓ С помощью команды `[python ./vol.py --plugins=/var/tmp/volatility-2.3.1/volatility/profiles/-info | grep -i profile | grep -i linux]` я убедился, что Volatility распознал директорию с профилями.

- ✓ С помощью команды `[python ./vol.py -plugins=/var/tmp/volatility-2.3.1/volatility/profiles/ -profile=<результат выполнения предыдущего пункта> linux|sof-f|var/tmp/src/mem.img|tail-20]` `âúëâüññëâíáíáëëcâàìíàììððè.`

## 4.8 Оформление результатов работы

```
root@metasploitable: /var/tmp/volatility-2.3.1
root@metasploitable:/var/tmp/volatility-2.3.1#
root@metasploitable:/var/tmp/volatility-2.3.1#
root@metasploitable:/var/tmp/volatility-2.3.1#
root@metasploitable:/var/tmp/volatility-2.3.1#
root@metasploitable:/var/tmp/volatility-2.3.1#
root@metasploitable:/var/tmp/volatility-2.3.1#
root@metasploitable:/var/tmp/volatility-2.3.1#
root@metasploitable:/var/tmp/volatility-2.3.1#
root@metasploitable:/var/tmp/volatility-2.3.1#
root@metasploitable:/var/tmp/volatility-2.3.1#
root@metasploitable:/var/tmp/volatility-2.3.1#
root@metasploitable:/var/tmp/volatility-2.3.1#
root@metasploitable:/var/tmp/volatility-2.3.1#
root@metasploitable:/var/tmp/volatility-2.3.1# grep "password changed" /var/log/auth.log
Apr 18 13:57:13 metasploitable passwd[5371]: pam_unix(passwd:chauthtok): password changed for msfadmin
root@metasploitable:/var/tmp/volatility-2.3.1# ls -l /var/www/UBUNTU-MSF804.zip
-rw-r--r-- 1 root root 345639 2022-04-18 14:37 /var/www/UBUNTU-MSF804.zip
root@metasploitable:/var/tmp/volatility-2.3.1# date
Mon Apr 18 14:51:12 EDT 2022
root@metasploitable:/var/tmp/volatility-2.3.1# echo "Свидилов Кирилл Андреевич"
Свидилов Кирилл Андреевич
root@metasploitable:/var/tmp/volatility-2.3.1# free -m
              total          used         free       shared    buffers     cached
Mem:           503            436             66              0           9        264
-/+ buffers/cache:           162          340
Swap:            0              0              0
root@metasploitable:/var/tmp/volatility-2.3.1# du -sh /var/tmp/LiME-master/src/mem.img
512M   /var/tmp/LiME-master/src/mem.img
kuxUBUNTU-MSF804x86 linux_lsof -f /var/tmp/LiME-master/src/mem.img | tail -20
Volatility Foundation Volatility Framework 2.3.1
14916      2 /dev/null
14916      3 socket:[45273]
14916      4 socket:[45299]
14916      5 socket:[45303]
14916      6 pipe:[45304]
14916      7 pipe:[45304]
14916      8 /dev/ptmx
14916      9 /dev/ptmx
14916     10 /dev/ptmx
14917      0 /dev/pts/1
14917      1 /dev/pts/1
14917      2 /dev/pts/1
14917    255 /dev/pts/1
14920      0 /dev/pts/1
14920      1 /dev/pts/1
14920      2 /dev/pts/1
14920    255 /dev/pts/1
23107      0 /dev/pts/1
23107      1 /dev/pts/1
23107      2 /dev/pts/1
root@metasploitable:/var/tmp/volatility-2.3.1#
```