Криминалистический анализ работающей компьютерной системы под управлением ОС Linux

Часть 1

Автор: Mariusz Burdach
Перевод: Дмитрий Цирлин

---

1. Введение

В процессе расследования и ликвидации последствий компьютерных инцидентов часто приходится сталкиваться с ситуацией, когда пользователь или администратор оставляет скомпрометированную компьютерную систему во включенном состоянии. Это дает замечательную возможность получить важную информацию, которая была бы безвозвратно утеряна при выключении системы. Я говорю о таких вещах как запущенные процессы, открытые **TCP/UDP** порты, образы программ, которые были удалены с носителей, но все еще активны в ОЗУ, содержимое буферов, очередь запросов на установление сетевых соединений, список установленных соединений и программные модули, загруженные в области виртуальной памяти, зарезервированные для ядра ОС. Все эти данные могут помочь исследователю при дальнейшем поиске и анализе криминалистических следов в лабораторных условиях. Более того, если компьютерный инцидент произошел относительно недавно, мы можем восстановить почти все данные, использовавшиеся злоумышленником, и последовательность его действий.

В некоторых случаях описанные в данной системе процедуры анализа работающих систем являются единственным способом собрать информацию об инциденте, так как некоторые виды вредоносных программ, таких, например, как средства несанкционированного доступа на основе технологии подгружаемых модулей ядра операционной системы **(LKM rootkit)**, загружаются непосредственно в память, не модифицируя какие-либо файлы или каталоги. Аналогичные ситуации возможны и в компьютерных системах, работающих под управлением ОС **Windows**. Сетевой червь **Code Red** является хорошим примером таких вредоносных программ, которые не записываются предварительно в файл, а загружаются непосредственно в ОЗУ системы и запускаются на исполнение.

С другой стороны, описанные ниже методы имеют серьезные ограничения и нарушают основное требование, предъявляемое к процедуре сбора доказательств при расследовании компьютерных инцидентов – требование, которое не может быть выполнено достаточно простыми способами. Имеется в виду следующее обстоятельство: любая утилита для сбора данных, запущенная на уровне пользователя или ядра операционной системы, естественным образом изменяет состояние исследуемой системы. Запуская любую утилиту в работающей системе, мы создаем как минимум один процесс, который может быть записан в область, где до этого возможно хранились данные, имеющие доказательственное значение. При создании нового процесса менеджер памяти операционной системы резервирует для этого процесса область памяти, при этом данные которые находились в этот момент в свободной области памяти или в области подкачки файловой системы могут быть в дальнейшем перезаписаны данными нового процесса.

Другие проблемы появляются, когда по результатам расследования в дальнейшем планируется принятие мер юридического характера, и, таким образом, необходимо обеспечить полное соответствие нормам местного законодательства. Следы злоумышленной деятельности, обнаруженные в ОЗУ системы могут быть признаны недостоверным, так как могли появиться вследствие работы использованных исследователем средств сбора и фиксации данных. Таким образом, перед выполнением каких-либо работ по поиску, сбору и фиксации доказательств, исследователь должен определить, возможно ли использовать для целей расследования работающую систему или нет. Достаточно часто преимущества сбора данных в работающей системе оказываются весьма ощутимыми. При исследовании копии содержимого ОЗУ могут быть обнаружены, например, пароли или дешифрованные файлы. Используя псевдо-файловую систему **/proc**, можно восстановить программы, файлы которых были стерты, но соответствующие

процессы все еще занимают выделенные под них области оперативной памяти.
Идеальным решением данных проблем представлялось бы наличие определенного устройства для компьютеров, построенных на платформе **Intel**, которое позволяло бы записать образ содержимого оперативной памяти на внешний носитель без использования функций операционной системы. Подобное решение существует для компьютерных систем на платформе **Sparc**, где копия содержимого всего ОЗУ может быть создана с помощью встроенного программного обеспечения **OpenBoot**. К сожалению, в настоящее время не существует подобных решений для компьютеров, построенных на базе платформ **Intel** или **AMD**.
Несмотря на указанные выше проблемы, использование программных средств для целей криминалистических исследований имеет определенные достоинства, которые я постараюсь раскрыть в данной статье. Основное внимание в данной статье будет сосредоточено на методах, используемых на этапе сбора и фиксации криминалистически значимой информации. Все собранные данные могут быть затем использованы при проведении более детального криминалистического анализа в лабораторных условиях. Некоторые из представленных методов могут также использоваться на этапах подготовки и обнаружения, представляющих собой в соответствии с классификацией, приведенной в руководстве **"Incident Handling Step by step"**, изданном институтом **SANS**, две из шести стадий процесса предупреждения компьютерных инцидентов и реагирования на них.

2. Криминалистический анализ

Данная статья состоит из четырех разделов:

- **2.1** Ознакомление с окружающими условиями и подготовка
- **2.2** Подготовка сменного носителя с набором инструментов для криминалистических исследований
- **2.3** Пошаговое описание процедуры сбора данных на работающей компьютерной системе
- **2.4** Предварительный анализ собранной информации и поиск по ключевым словам

Разделы **2.1**, **2.2** и частично **2.3** будут представлены в данной статье; остальные процедуры, а также некоторые вопросы исследования собранных данных в лабораторных условиях будут обсуждаться во второй части статьи, которая выйдет в следующем месяце.

2.1 Ознакомление с окружающими условиями и подготовка

Перед началом сбора данных непосредственно с работающей компьютерной системы нам необходимо ознакомиться с окружающими условиями. В первую очередь необходимо с помощью сниффера исследовать информацию, принимаемую и передаваемую скомпрометированной компьютерной системой. Этот шаг является абсолютно необходимым. Некоторые виды злонамеренных действий могут быть обнаружены уже на этом этапе путем записи и анализа в реальном времени передаваемой в сети информации. Утилита **tcpdump** замечательно подходит для решения этих задач. Рекомендуется выполнять запись сетевых пакетов в необработанном виде **(raw format)**, поскольку в противном случае могут возникать проблемы, связанные с недостаточной производительностью.
Перед тем как выполнять какие-либо действия непосредственно с скомпрометированной компьютерной системой, мы должны составить на бумаге план действий и описание процедуры сбора информации. Образец типового описания процедуры приведен в третьей части данной статьи. Наличие такого описания помогает избежать случайных ошибок при проведении криминалистического исследования компьютерных инцидентов. Дополнительные сведения должны вноситься в описание после выполнения каждого этапа исследований, а также в случае возникновения нештатных ситуаций. Скрупулезное документирование процесса исследований является очень важным условием, особенно, если в дальнейшем планируется передача полученных материалов в судебные органы.

Следующим шагом является подготовка к фиксации результатов выполнения команд на скомпрометированной системе в процессе дальнейшего сбора информации. Для этого нам необходимо подключить дополнительный компьютер, на который будут пересылаться все данные, к локальной сети, в которую включена скомпрометированная компьютерная система. Запись данных непосредственно на носители скомпрометированной системы может повлечь за собой уничтожение следов несанкционированной деятельности. Для минимизации воздействия проводимых исследований на состояние скомпрометированной системы мы должны обеспечить передачу всей полученной компьютерной информации и ее фиксацию на каком-либо внешнем носителе или компьютере. Это является одним из наиболее важных правил при проведении криминалистических исследований компьютерных систем, и, как уже указывалось ранее, обеспечить выполнение этого правила не всегда возможно достаточно простыми средствами.
Если у нас отсутствует заранее подготовленный сменный носитель, на котором записаны необходимые для проведения криминалистических исследований программные средства и который может быть установлен в скомпрометированную систему, нам необходимо подготовить его теперь. Указанные программные средства будут использоваться для сбора важной информации, при этом в первую очередь должны собираться и фиксироваться данные, наиболее подверженные изменению, а затем уже постепенно все более и более устойчивые данные. В следующем разделе описана методика подготовки сменных носителей, содержащих необходимые инструменты для проведения криминалистических исследований.

2.2 Подготовка сменного носителя с набором инструментов для криминалистических исследований

Необходимо помнить, что в процессе последующего сбора данных, должны выполняться следующие условия:

- Следует по возможности избегать запуска программ, установленных на скомпрометированной системе, так как злоумышленник мог модифицировать системные команды **(**например, **netstat)** или системные библиотеки **(**например, **libproc)** и полученные результаты могут оказаться недостоверными. Для того чтобы обеспечить выполнение данного условия, программные средства, входящие в подготовленный нами набор для криминалистических исследований, должны быть скомпилированы и скомпонованы статически.
- Следует по возможности избегать запуска программ, которые могут вызвать изменения метаданных файлов и каталогов.
- Все результаты исследований должны записываться на внешний носитель. Чтобы обеспечить выполнение этого условия, для записи результатов исследований будет использоваться дополнительный компьютер. Для передачи данных по сети будет применяться утилита **netcat.**
- Необходимо использовать программные средства, обеспечивающие подсчет контрольных значений **(hash)** для собираемых компьютерных данных. Наличие вычисленных контрольных значений позволяет убедиться, что собранные данные не подверглись изменению. Общепризнанные правила требуют убедиться, что данные не подвергались изменению в процессе сбора и записаны в неизменном виде на внешний носитель. Для этого необходимо сравнить контрольные значения, вычисленные для исходных данных и данных, записанных на внешнем носителе. Иногда невозможно вычислить контрольные значения исходных данных непосредственно на скомпрометированной системе. Примером этого может служить вычисление контрольных значений содержимого оперативной памяти. Если с помощью утилиты **md5sum** попытаться два раза подряд вычислить контрольные значения для устройства **/dev/mem**, то полученные в обоих случаях контрольные значения будут отличаться друг от друга. Это происходит вследствие того, что каждый раз эта программа загружается в память **(**т.е. создается новый процесс, для работы которого необходима оперативная память**)** и, таким образом, происходит

изменение содержимого ОЗУ. В нашей процедуре расчет контрольных значений будет выполняться непосредственно по завершении сбора данных, при этом, если возможно, расчет будет выполняться как для исходных данных, так и для данных, записанных на внешний носитель. Для расчета контрольных значений и обеспечения целостности данных будет использоваться утилита **md5sum.**

- Упоминавшееся требование, касающееся того, что используемые при проведении исследований инструменты не должны изменять содержимого ОЗУ или области подкачки скомпрометированной системы, не может быть выполнено полностью на некоторых этапах исследований. Это будет обсуждаться более подробно в разделе **2.3.** Пока же давайте убедимся, что все необходимые для проведения криминалистических исследований программные средства записаны на съемный носитель, как указано в Таблице **1.**

Таблица 1: Требования к программным средствам для криминалистических исследований, записанным на съемный носитель.

	Название программы	Источник и метод создания
1	**nc**	**http://www.atstake.com/research/tools/network_utilities/nc110.tgz** команды компоновки: **$tar zxvf nc110.tgz; make linux** команды проверки: **file nc** или **ldd nc**
2	**dd**	**http://www.gnu.org/software/fileutils/fileutils.html** **(**включена в набор основных утилит ОС**)**
3	**datecat**	**http://www.gnu.org/software/coreutils/** команды компоновки: **$ tar zxvf coreutils-5.0.tar.gz; configure CC="gcc -static", make** команды проверки: **file date cat** или **ldd date cat**
4	**pcat**	**http://www.porcupine.org/forensics/tct** команды компоновки: **$tar zxvf tct-1.14.tgz; make CC="gcc -static"** команды проверки: **file pcat** или **ldd pcat**
5	**Hunter.o**	**http://www.phrack.org/phrack/61/p61-0x03_Linenoise.txt** Чтобы сделать данный модуль более **«**независимым**»,** необходимо удалить следующие строки исходного текста: **#ifdef CONFIG_MODVERSIONS** **#define MODVERSIONS** **#include <linux/modversions.h>** **#endif** Удалив **MODVERSIONS,** мы обеспечиваем возможность загрузки данного модуля с другим ядром ОС. команды компоновки: **$ gcc -c hunter.c -I/usr/src/linux/include/**
6	**insmod**	**http://www.kernel.org/pub/linux/utils/kernel/modutils/** для ядра версии **2.4** команды компоновки: **$./configure-enable-insmod_static; make** команды проверки: **file insmod.static** или **ldd insmod.static**
7	**NetstatArproute**	**http://freshmeat.net/projects/net-tools/** команды компоновки: **$bzip2 -d net-tools-1.60.tar.bz2; tar xvf net-tools-1.60.tar.bz2; make config; make CC="gcc -static"** команды проверки: **file netstat arp route** или **ldd netstat arp route**
8	**dmesg**	**http://ftp.cwi.nl/aeb/util-linux/util-linux-2.12.tar.gz** команды компоновки: **$./configure; make CC="gcc -static"** команды проверки: **file dmesg** или **ldd dmesg**

После того как все утилиты успешно скомпонованы, мы можем записать их на сменный носитель **(**например, на диск **CD-RW).**

2.3 Пошаговое описание процедуры сбора данных на работающей системе

Одно из очень важных требований при проведении криминалистических исследований заключается в том, что процесс сбора данных должен выполняться в определенной последовательности, начиная с наиболее подверженных изменению данных и постепенно

переходя к сбору более устойчивых данных. Необходимо помнить об этом в процессе выполнения исследований.

Шаг 1: Сфотографируйте экран монитора скомпрометированной компьютерной системы

Это достаточно простая задача и для ее выполнения может использоваться цифровая фотокамера.

Перед тем как перейти к шагу **2 –** монтированию нашего съемного носителя в файловую систему скомпрометированного компьютера **–** давайте задумаемся об эффекте, который эти действия окажут на скомпрометированную систему. Что произойдет в результате наших действий**?** Пока давайте оставим за рамками обсуждения результаты этого воздействия на состояние ОЗУ скомпрометированной системы.

Очевидно, что мы должны смонтировать сменный носитель в файловую систему скомпрометированной компьютерной системы. Для того чтобы сделать это, нам потребуется воспользоваться командой **mount** в скомпрометированной системе, которая возможно подверглась модификации. Скорее всего, это единственная ситуация, когда мы будем вынуждены воспользоваться системной командой скомпрометированной системы. Если команда **mount** будет выполнена как положено, все последующие команды будут использовать утилиты, записанные на нашем сменном носителе.

Мы должны выяснить какой эффект окажет на систему выполнение команды **mount**. Я провел некоторые эксперименты на компьютере, результаты которых, показывающие зафиксированные изменения файлов и каталогов, представлены в Таблице **2**.

strace /bin/mount /mnt/cdrom

Таблица 2: Файлы, к которым осуществлялся доступ при выполнении команды mount.

Файл	Метаданные, измененные при выполнении команды **mount**
/etc/ld.so.cache	**atime**
/lib/tls/libc.so.6	**atime**
/usr/lib/locale/locale-archive	**atime**
/etc/fstab	**atime**
/etc/mtab*	**atime, mtime, ctime**
/dev/cdrom	**atime**
/bin/mount	**atime**

*****Доступ к данному файлу может быть предотвращен при использовании опции **"-n"** в командной строке.

Можно представить ситуацию, когда команда **mount** была модифицирована злоумышленником. В этом случае, когда происходит попытка выполнить данную команду, вместо монтирования файловой системы может быть запущен специальный процесс, удаляющий все следы на скомпрометированной системе. Такой процесс называют **«**стоп-кран**» ("deadman switch").** Однако, давайте предположим, что в нашем случае такого не происходит, и вернемся к обсуждению процесса сбора данных.

Я рекомендую вам исследовать аналогичным образом все утилиты, которые вы намерены включить в свой криминалистический набор, записать на съемный носитель и использовать в дальнейшем для сбора доказательств на скомпрометированной системе.

Необходимо также остановиться и задуматься о потенциальных проблемах, которые могут подстерегать нас в процессе монтирования сменного носителя:

- После установки сменного носителя в дисковод, процесс менеджера томов файловой системы **(Volume Manager)** автоматически смонтировал сменный носитель. Какие файлы и каталоги оказались при этом модифицированы**?** Перечислены ли все эти файлы в Таблице **1?**

- Предположим, что посторонний сменный носитель уже смонтирован в файловую систему скомпрометированного компьютера. При этом первым делом необходимо демонтировать этот посторонний носитель. Как выполнить это наиболее безопасным способом? Я могу предложить два возможных решения. Мы можем использовать не заслуживающую полного доверия команду **unmount** скомпрометированной системы, либо мы можем статически скомпоновать команду **unmount** на заведомо надежной системе и записать ее на флоппи диск. Затем мы можем использовать ненадежную команду **mount** в скомпрометированной системе, чтобы смонтировать флоппи диск, а затем уже использовать записанную на нем надежную команду **unmount**. Это немного сложный, но эффективный путь – мы по-прежнему используем только одну ненадежную команду скомпрометированной системы.
- Администратор не зарегистрирован в системе или, более того, пароль учетной записи администратора был изменен злоумышленником. Если администратор не зарегистрирован в системе, мы должны будем выполнить регистрацию. Какие файлы будут модифицированы, и к каким файлам будет осуществляться доступ в процессе регистрации? Сколько дополнительных процессов будет при этом запущено? Если пароль учетной записи администратора был изменен, какие еще учетные записи существуют в системе и могут быть использованы? Какие подверженные изменению данные могут быть собраны при отсутствии доступа к шеллу? Открытые **TCP/UDP** порты, установленные сетевые соединения, что еще?
- Есть ли еще какие-нибудь непредвиденные проблемы?

Step 2: Монтирование сменного носителя

Давайте смонтируем наш сменный носитель, в данном случае **CD-ROM** с записанным набором утилит для криминалистических исследований.

```
# mount -n /mnt/cdrom
```

Если монтирование прошло успешно, мы можем приступать к наиболее важной стадии сбора данных. Напоминаю, что все результаты, полученные при выполнении команд на скомпрометированной системе, должны пересылаться на отдельный компьютер. Для этого я использую утилиту **netcat** и конвейерное перенаправление данных. В дальнейших пояснениях, чтобы различать какие команды выполняются на каком из компьютеров, все команды, выполняемые на скомпрометированной системе, будут предваряться словом **(compromised)**, заключенным в скобки. Все команды, выполняемые на отдельном компьютере, предназначенном для записи и хранения полученных результатов, будут предваряться словом **(remote)**, заключенным в скобки. Рассмотрим следующий пример.
Чтобы передать информацию о текущих показаниях часов скомпрометированной системы и записать эти данные в файл на другом компьютере **(**положим, что этому компьютеру присвоен **IP** адрес **192.168.1.100)**, нам необходимо открыть **TCP** порт на этом компьютере с помощью следующих команд:

```
(remote)# nc -l -p 8888 > date_compromised
```

Далее на скомпрометированной системе необходимо выполнить следующие команды:

```
(compromised))# /mnt/cdrom/date | /mnt/cdrom/nc 192.168.1.100 8888 -w 3
```

Для контроля целостности собранных данных надо обеспечить вычисление контрольных значений для записываемых файлов, а также вести бумажный отчет, четко описывая все шаги выполняемые в рамках этой документированной процедуры исследований.

```
(remote)# md5sum date_compromised > date_compromised.md5
```

В некоторых случаях значения контрольных сумм для собранных данных могут также вычисляться непосредственно на скомпрометированной системе и затем пересылаться на другой компьютер. Дополнительные соображения по поводу проблем, которые могут при этом возникать, уже обсуждались в данной статье.

(compromised)# /mnt/cdrom/md5sum /etc/fstab | /mnt/cdrom/nc 192.168.1.100 8888 -w 3

Step 3: Текущие показания часов

Результат данных команд представлен в формате **UTC (Coordinated Universal Time)**
(remote)# nc -l -p *port* > date_compromised

(compromised)# /mnt/cdrom/date -u | /mnt/cdrom/nc *(remote) port*

(remote)# md5sum date_compromised > date_compromised.md5

Step 4: Динамические таблицы

Прежде всего, мы должны сохранить содержимое различных динамических таблиц, так как срок жизни данных в таких таблицах весьма ограничен. Ниже представлены процедуры сохранения содержимого динамических таблиц **arp** и роутинга.
Динамическая таблица **MAC-**адресов**:**
(remote)# nc -l -p *port* > arp_compromised

(compromised)# /mnt/cdrom/arp -an | /mnt/cdrom/nc *(remote) port*

(remote)# md5sum arp_compromised > arp_compromised.md5

Динамическая таблица роутинга**:**
(remote)# nc -l -p *port* > route_compromised

(compromised) # /mnt/cdrom/route -Cn | /mnt/cdrom/nc *(remote) port*

(remote)#md5sum route_compromised > route_compromised.md5

Step 5: Установленные соединения, запросы на соединения и открытые TCP / UDP порты.

Теперь мы можем сохранить информацию об установленных соединениях и открытых **TCP/UDP** портах. Процедура сбора информации обо всех активных сокетах будет представлена при описании восьмого шага описываемой процедуры.
(remote)#nc -l -p *port* > connections_compromised

(compromised)# /mnt/cdrom/netstat -an | /mnt/cdrom/nc *(remote) port*

(remote)#md5sum connections_compromised > connections_compromised.md5

В данном случае вместо команды **netstat** могла быть также использована команда **cat.** Информация об открытых портах хранится в псевдо-файловой системе **/proc (**файлы **/proc/net/tcp** и **/proc/net/udp).** Информация о всех установленных соединениях хранится в файле **/proc/net/netstat.** Все данные в этих файлах представлены в шестнадцатиричном формате.
Пример**: 0100007F:0401** соответствует десятичной записи **127.0.0.1:1025.**
Как указывалось ранее, информация об установленных соединениях может быть также получена с помощью записи и анализа сетевого трафика. Необходимо отметить, что существует простой метод обнаружения средств несанкционированного доступа, использующих технологию подгружаемых модулей ядра ОС **(LKM)** и маскирующих наличие открытых портов. Для этого необходимо провести с помощью отдельного компьютера сканирование по сети скомпрометированной системы на наличие открытых портов, и сравнить полученный список со списком открытых портов, выдаваемым утилитой **netstat.** Однако данный метод приводит к весьма значительным изменениям состояния скомпрометированной системы, при описании шага **7** процедуры исследования я представлю альтернативный метод детектирования маскирующихся средств несанкционированного доступа на основе технологии **LKM.**

Заключение к части 1

После того как зафиксированы показания часов и данные о состоянии сетевой активности, можно выполнить еще ряд дополнительных шагов по исследованию скомпрометированной системы, прежде чем выключать ее. В следующем месяце, во второй части данного цикла статей основное внимание будет уделено сбору дополнительных данных с целью поиска вредоносных программ. Мы также обсудим некоторые методы поиска криминалистических следов, которые могут быть использованы при более детальном анализе собранных данных в лабораторных условиях.

Список литературы

- **Alessandro Rubini, Jonathan Corbet. Linux Device Drivers, 2nd Edition. O'Reilly; 2001.**
- **Dan Farmer, Wietse Venema. Column series for the Doctor Dobb's Journal. http://www.porcupine.org/forensics/column.html.**
- **Daniel P. Bovet, Marco Cesati. Understanding the Linux Kernel, 2nd Edition. O'Reilly; 2002.**
- **Kernel source code. http://www.kernel.org**
- **Linux manual pages.**
- **National Institute of Standards and Technology. Computer Security Incident Handling Guide. http://csrc.nist.gov.**
- **PHRACK #61. Finding hidden kernel modules (the extrem way) by madsys. http://www.phrack.org.**
- **RFC 3227. Guidelines for Evidence Collection and Archiving.**
- **Smith Fred, Bace Rebecca. A guide to forensic testimony. Addison Wesley; 2003.**
- **Symantec Corporation. CodeRed Worm. http://securityresponse.symantec.com.**
- **The Honeynet Project. Scan 29. http://www.honeynet.org**
- **The SANS Institute. Incident Handling step by step. http://www.sans.org**

Примечание: Оригинал статьи находится по адресу - http://www.securityfocus.com/infocus/1769