# INFORMATYKA ŚLEDCZA

Przewodnik po narzędziach open source

Cory Altheide
Harlan Carvey

Helion

Tytuł oryginału: Digital Forensics with Open Source Tools

Tłumaczenie: Grzegorz Kowalczyk

ISBN: 978-83-246-9562-1

Syngressis an imprint of Elsevier
225 Wyman Street, Waltham, MA 02451, USA



Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/infsle.zip

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/infsle
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

# Spis treści

ROZDZIAŁ

# Analiza zawartości dysku i systemu plików

3

**W TYM ROZDZIALE:**

- Podstawowe pojęcia związane z analizą nośników danych
- Pakiet The Sleuth Kit
- Podział na partycje i konfiguracja dysków
- Kontenery specjalne
- Haszowanie
- Data carving — wyszukiwanie i odzyskiwanie danych z niealokowanej przestrzeni nośnika danych
- Tworzenie binarnych kopii nośników danych

## ANALIZA ZAWARTOŚCI NOŚNIKÓW DANYCH — POJĘCIA PODSTAWOWE

W największym uproszczeniu możemy powiedzieć, że informatyk śledczy zajmuje się analizą *plików* zapisanych na różnego rodzaju *nośnikach danych* — mogą to być pliki skasowane przez użytkownika, pliki znajdujące się w folderach, pliki osadzone w innych plikach czy pliki zapisane w takich czy innych kontenerach plików. Zadaniem analizy nośników danych jest identyfikacja, przetwarzanie i analiza zawartości takich plików, jak również analiza samych systemów plików (ang. *file systems*), w których takie pliki są przechowywane. Proces **identyfikacji** obejmuje między innymi określenie, jakie pliki, zarówno istniejące, jak i skasowane, są zapisane na danym nośniku danych. **Przetwarzanie** to proces pozwalający na wyszukiwanie i pozyskiwanie z plików odpowiednich danych i metadanych mających znaczenie dla bieżącej ekspertyzy śledczej. Wreszcie ostatni etap, **analiza**, to proces, w którym informatyk śledczy poddaje analizie zgromadzone dane, stara się znaleźć pomiędzy nimi powiązania i wyciągnąć odpowiednie wnioski, potwierdzające lub odrzucające założoną hipotezę.

Zwróć uwagę, że opisane powyżej etapy postępowania nie mają charakteru dyskretnego, a wręcz przeciwnie, w praktyce bardzo często zdarza się, że dana operacja nie daje się jednoznacznie zakwalifikować — na przykład ***carving***, czyli wyszukiwanie i odzyskiwanie danych z niealokowanej przestrzeni nośnika danych, może być traktowany zarówno jako

identyfikacja, jak i przetwarzanie danych. Niezależnie jednak od tego wydaje nam się, że podział przedstawiony powyżej dobrze oddaje charakter pracy informatyków śledczych i wyjaśnia, *dlaczego* wykonują takie czy inne operacje.

W tym rozdziale skoncentrujemy się przede wszystkim na operacjach związanych z identyfikacją i pozyskiwaniem artefaktów przechowywanych w systemach plików oraz wydobywaniem *informacji o plikach*. Nie znajdziesz tutaj zatem opisów znaczenia samych artefaktów związanych z plikami i systemami plików, ponieważ będziemy się nimi szczegółowo zajmować w rozdziałach od 4. do 8.

Należy tutaj jednak zaznaczyć, że pomimo iż tematy związane z analizą śledczą systemów plików to jedne z najważniejszych zagadnień dla każdego informatyka śledczego, to jednak szczegółowe omawianie wszystkich możliwych artefaktów i niuansów poszczególnych systemów plików zdecydowanie wykraczałoby daleko poza ramy tej książki. Jeżeli jesteś zainteresowany poszerzeniem swojej wiedzy w tym zakresie, to zdecydowanie polecamy lekturę doskonałej książki Briana Carriera, zatytułowanej *File System Forensic Analysis* [1], która jest jedną z najlepszych książek omawiających zagadnienia analizy śledczej systemów plików.

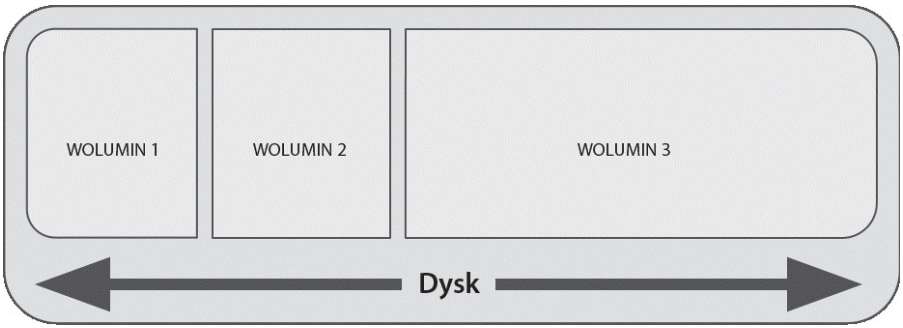
## Abstrakcyjny model systemu plików

We wspomnianej w poprzednim podrozdziale książce *File System Forensic Analysis* autor przedstawia abstrakcyjny model systemu plików, który może być wykorzystywany do opisywania poszczególnych mechanizmów systemów plików i generowanych przez nie artefaktów. Czytelnicy posiadający pewną wiedzę w zakresie sieci komputerowych z pewnością zauważą tutaj pewne podobieństwo do modelu OSI, wykorzystywanego do opisywania systemów komunikacyjnych.

Model systemu plików opisywany przez Briana Carriera wygląda następująco (patrząc od najniższej warstwy do najwyższej):

- Dysk
  Termin **dysk** (ang. *disk*) odnosi się do fizycznego urządzenia będącego nośnikiem danych, takiego jak na przykład dysk twardy SCSI czy SATA, karta pamięci z aparatu cyfrowego czy przenośna pamięć USB. Analiza na tym poziomie znajduje się zazwyczaj daleko poza zasięgiem większości analityków — przeprowadzanie analizy nośników fizycznych (na przykład „talerzy" w dyskach twardych) wymaga bardzo specjalistycznej wiedzy, dostępu do specjalnie przygotowanych, sterylnych pomieszczeń oraz bardzo drogiego, specjalistycznego sprzętu, takiego jak mikroskopy elektronowe. Z drugiej strony jednak, gwałtownie rosnąca popularność przenośnych pamięci USB i dysków SSD (ang. *Solid State Disk*) powoduje, że analiza nośników na tym poziomie przestaje być tylko i wyłącznie domeną wąskiego grona specjalistów.
- Wolumin
  **Wolumin** (ang. *volume*) jest tworzony z wykorzystaniem całości lub części jednego lub więcej dysków. Pojedynczy dysk może zawierać wiele woluminów, ale równie dobrze jeden wolumin może rozciągać się na kilka dysków; wszystko zależy od konfiguracji

danego systemu. Bardzo często wymiennie z terminem *wolumin* używany jest termin *partycja*, choć w swojej książce Brian Carrier rozgranicza te dwa pojęcia: *partycja* jest ograniczona do jednego fizycznego dysku, a *wolumin* to kolekcja składająca się z jednej lub więcej partycji. Krótko mówiąc, wolumin opisuje liczbę sektorów na dysku (lub dyskach) w danym systemie. Na rysunku 3.1 przedstawiamy uproszczony schemat przedstawiający dysk i utworzone na nim woluminy.




**RYSUNEK 3.1.**

Dysk i woluminy

- System plików
  **System plików** (ang. *file system*) jest zapisany na woluminie i opisuje przechowywane w nim pliki oraz powiązane z nimi metadane. Na warstwie systemu plików możesz również znaleźć takie elementy jak metadane specyficzne dla danego typu systemu plików i wykorzystywane wyłącznie do zapewnienia jego poprawnego działania — dobrym przykładem mogą być tzw. superbloki (ang. *superblocks*) występujące w systemie Ext2.
- Jednostka danych
  **Jednostka danych** (ang. *Data Unit*) to najmniejszy niezależny blok danych dostępny w danym systemie plików. W rozwiązaniach wywodzących się z systemu UNIX takie jednostki danych noszą nazwę **bloków** (ang. *blocks*). Takie bloki danych mają zazwyczaj rozmiar będący wielokrotnością rozmiaru pojedynczego sektora. Jeszcze nie tak dawno temu sektory na dyskach miały rozmiar 512 bajtów, ale we współczesnych systemach plików najmniejsze adresowalne jednostki danych mają rozmiar 4096 bajtów (4 kB) lub więcej. Informacje dostępne na tej warstwie modelu systemu plików są proste — jest to zawartość poszczególnych jednostek danych. Na przykład jeżeli wybrana jednostka danych jest przypisana do zdjęcia w formacie JPEG, to taka jednostka danych przechowuje fragment danych obrazu w formacie JPEG. Jeżeli wybrana jednostka danych jest przypisana do pliku tekstowego, to po prostu przechowuje fragment zawartości takiego pliku.

- Metadane
  **Metadane** (ang. *metadata*) to po prostu *dane opisujące inne dane*. Jeżeli przyjmiemy, że *warstwa jednostek danych* w systemie plików przechowuje dane, to na *warstwie metadanych* przechowywane są dane opisujące poszczególne jednostki danych. W rozwiązaniach wywodzących się z systemu UNIX takie jednostki metadanych są nazywane **i-węzłami** (ang. *inodes*). Dokładna zawartość jednostek metadanych zależy od rodzaju systemu plików, ale ogólnie rzecz biorąc, dane przechowywane w tej warstwie reprezentują takie elementy jak znaczniki czasu poszczególnych plików, informacje o właścicielach plików czy lista jednostek danych alokowanych do poszczególnych jednostek metadanych. W kolejnych podrozdziałach będziemy bardziej szczegółowo omawiać poszczególne artefakty dla różnych systemów plików.
- Nazwa pliku
  **Nazwa pliku** to warstwa, na której działa użytkownik. Nie będzie chyba dla nikogo zaskoczeniem, jeżeli napiszemy, że na tej warstwie przechowywane są nazwy plików i katalogów. Podobnie jak w poprzednich przypadkach liczba i rodzaje artefaktów dostępnych dla tej warstwy zmieniają się w zależności od systemu plików. Niezależnie jednak od systemu plików nazwa pliku powinna mieć przypisany wskaźnik prowadzący do odpowiadającej jej struktury metadanych.

Ponieważ przedstawiony abstrakcyjny model systemu plików został zbudowany z myślą o systemach plików wywodzących się z rodziny UNIX, niektóre z jego elementów nie do końca odpowiadają projektom systemów plików na innych platformach. Nie zmienia to jednak w niczym faktu, że dobre poznanie budowy i działania tego modelu jest niezbędnym warunkiem do zrozumienia wagi i znaczenia artefaktów, jakie występują w *dowolnym* systemie plików.

## PAKIET THE SLEUTH KIT

Artefakty systemu plików będziemy przetwarzać za pomocą pakietu The Sleuth Kit (*http://www.sleuthkit.org/*). Jest to zestaw narzędzi przeznaczonych do analizy śledczej, napisany przez Briana Carriera jako zaktualizowana wersja starszego pakietu The Coroner's Toolkit (TCT). Pakiet TCT był dedykowany przeprowadzaniu analizy śledczej systemów wywodzących się z rodziny UNIX. Posiadał ogromne możliwości, ale niestety nie był również pozbawiony wad, takich jak brak możliwości działania w innych systemach czy brak wsparcia dla systemów plików innych niż uniksowe. Brian Carrier zaprojektował nowy pakiet tak, aby można go było używać w różnych systemach i rozszerzać jego funkcjonalność za pomocą wtyczek, dzięki czemu The Sleuth Kit stał się bardzo wygodnym i użytecznym narzędziem dla szerokiej rzeszy informatyków śledczych.

### Instalowanie pakietu The Sleuth Kit

Pakiet The Sleuth Kit posiada wbudowaną obsługę binarnych obrazów dysków typu RAW (jedno- i wieloplikowych), ale po zaimportowaniu odpowiednich mechanizmów z bibliotek LibEWF oraz AFFLib (o których pisaliśmy w rozdziale 2.) potrafi obsługiwać również inne formaty. Warto zauważyć, że dla systemu Ubuntu istnieje gotowa, prekompilowana wersja tego pakietu, którą możemy zainstalować za pośrednictwem menedżera pakietów tego systemu. Pobranie kodu źródłowego i samodzielne skompilowanie programu pozwoli Ci jednak na znaczące zredukowanie liczby pośredników, którzy byli zaangażowani w przygotowanie binarnej, wykonywalnej wersji pakietu. Samodzielna kompilacja gwarantuje również, że będziesz korzystał z najnowszych wersji podstawowych narzędzi i bibliotek, ponieważ repozytoria prekompilowanych pakietów nie zawsze są aktualizowane na bieżąco.

Zwróć uwagę, że po uruchomieniu skryptu konfiguracyjnego pakietu The Sleuth Kit (`.\configure`) powinieneś w końcowej części wyników działania zobaczyć następujące wiersze informacji:

```
checking afflib/afflib.h usability... yes
checking afflib/afflib.h presence... yes
checking for afflib/afflib.h... yes
checking for af_open in -lafflib... yes
checking libewf.h usability... yes
checking libewf.h presence... yes
checking for libewf.h... yes
checking for libewf_open in -lewf... yes
configure: creating ./config.status
```

Wyświetlenie takich komunikatów potwierdza, że biblioteki LibEWF oraz AFFLib są zainstalowane poprawnie i zostaną użyte przez pakiet The Sleuth Kit.

Po zakończeniu przygotowania konfiguracji pakietu możesz dokończyć proces instalacji, wykonując najpierw polecenie `make`, a następnie polecenie `sudo make install`. Po zakończeniu działania tych poleceń pakiet The Sleuth Kit składający się z szeregu narzędzi konsolowych będzie zainstalowany w Twoim systemie.

**OSTRZEŻENIE**

**MASZ UPRAWNIENIA UŻYTKOWNIKA ROOT?**

Jeżeli chcesz użyć pakietu The Sleuth Kit do przeprowadzenia analizy zawartości dysku dołączonego do Twojego systemu (zamiast analizy binarnego obrazu takiego dysku), pamiętaj, że będziesz potrzebował uprawnień użytkownika *root*. Możesz to osiągnąć, przełączając konsolę za pomocą polecenia `su –` lub uruchamiając proces instalacji za pomocą polecenia `sudo`, tak jak to robiliśmy w rozdziale 2.

## Narzędzia pakietu

Poznanie zastosowania i sposobu działania 21 poleceń konsolowych wchodzących w skład pakietu The Sleuth Kit może się wydawać ogromnym wyzwaniem, zwłaszcza jeżeli nie przywykłeś do pracy z konsolą systemu. Poszczególne polecenia posiadają jednak logicznie dobrane nazwy wskazujące warstwę systemu plików, na której pracują, oraz rodzaj danych wyjściowych, których możesz oczekiwać po ich uruchomieniu. Ponieważ pakiet wywodzi się z systemów rodziny UNIX, to użytkownikom mającym doświadczenie w pracy z konsolą systemu Linux przyjęta konwencja nazw będzie się wydawała oczywista.

W nazwach poszczególnych narzędzi pakietu The Sleuth Kit pojawiają się wspólne prefiksy wskazujące warstwę modelu systemu plików, na której działa dane narzędzie:

- *mm-* — to narzędzia pracujące na woluminach dysków (ang. *media management*).
- *fs-* — to narzędzia działające na strukturach systemu plików.
- *blk-* — to narzędzia pracujące na warstwie jednostek danych (warstwie „bloków" danych).
- *i-* — to narzędzia działające na warstwie metadanych (warstwa i-węzłów).
- *f-* — to narzędzia przeznaczone do działania na warstwie nazw plików.

Istnieją jednak dwie dodatkowe warstwy systemu plików, które nie mają swoich bezpośrednich odpowiedników w przedstawionym wcześniej abstrakcyjnym modelu:

- *j-* — narzędzia do pracy z *dziennikami systemu plików*.
- *img-* — narzędzia przeznaczone do pracy z *plikami binarnych kopii nośników danych*.

Oprócz wymienionych wyżej prefiksów w nazwach poleceń pakietu The Sleuth Kit znajdziesz również szereg sufiksów, które wskazują na rolę poszczególnych narzędzi i rodzaj wyników ich działania:

- *-stat* — narzędzia, które wyświetlają ogólne informacje o danym elemencie; podobne do polecenia `stat` w systemach uniksowych.
- *-ls* — narzędzia, które wyświetlają zawartość danej warstwy; podobne do polecenia `ls` w systemach uniksowych.
- *-cat* — narzędzia, które pozwalają na pozyskanie zawartości danej warstwy; podobne do polecenia `cat` w systemach uniksowych.

Oprócz narzędzi, których nazwy są zgodne z przedstawioną wyżej konwencją, w pakiecie The Sleuth Kit znajdziesz również kilka narzędzi, których nazwy nie mają z tą konwencją nic wspólnego. Takie narzędzia zostały opisane w podrozdziale zatytułowanym „Inne narzędzia pakietu The Sleuth Kit".

Aby pokazać zastosowania pakietu The Sleuth Kit, przejdziemy teraz przez kolejne warstwy modelu systemu plików i opiszemy wszystkie narzędzia występujące na poszczególnych warstwach. Dodatkowo szczegółowo omówimy sposób działania najważniejszych narzędzi i opiszemy wyniki ich działania na przykładzie systemu plików Ext3.

**UWAGA**

### NARZĘDZIA WARSTWY DYSKU W PAKIECIE THE SLEUTH KIT

W bieżącej wersji pakietu The Sleuth Kit nie ma żadnych narzędzi działających na warstwie dysku. Ponieważ The Sleuth Kit jest pakietem przeznaczonym do analizy śledczej systemów plików, brak takich narzędzi nie powinien być żadnym zaskoczeniem. Warto jednak zauważyć, że w wersjach pakietu starszych niż 3.1.0 można było znaleźć dwa narzędzia działające na warstwie dysku, które do tej pory możesz odnaleźć w niektórych starszych dystrybucjach Live CD śledczych wersji systemu Linux.

Pierwsze z tych poleceń, `disk_stat`, wyświetlało informacje o tym, czy dany dysk posiada tzw. obszar HPA (Host Protected Area). Zastosowanie HPA to jedna z metod pozwalających na sztuczne ograniczenie liczby sektorów dysku, które mogą być zaadresowane przez system operacyjny.

Drugie polecenie, `disk_sreset`, pozwalało na tymczasowe usunięcie ustawień obszaru HPA z dysku. Taka operacja miała charakter *ulotny* — domyślne ustawienia obszaru HPA były automatycznie przywracane po wyłączeniu i ponownym włączeniu zasilania dysku. Tymczasowe usunięcie ustawień obszaru HPA za pomocą polecenia `disk_sreset` pozwalało na utworzenie pełnej kopii binarnej zawartości dysku, w tym zawartości obszaru, która w normalnych okolicznościach była chroniona.

Inną metodą ograniczenia liczby sektorów widocznych dla systemu operacyjnego jest zastosowanie tzw. obszaru DCO (Device Configuration Overlay). Oba obszary, HPA i DCO, mogą być wykryte i usunięte za pomocą narzędzia `hdparm`, które jest domyślnie wbudowane w zdecydowanej większości współczesnych dystrybucji systemu Linux.

Opisy innych narzędzi niezwiązanych z pakietem The Sleuth Kit, które działają na warstwie dysku, znajdziesz w podrozdziale „Tworzenie binarnych kopii nośników danych" w dalszej części tego rozdziału.

### *Narzędzia warstwy woluminu*

Polecenie `mmstat` wyświetla typ woluminu wykorzystywany w badanym dysku lub binarnym obrazie nośnika danych.

Polecenie `mmls` analizuje i wyświetla informacje o strukturach zarządzających danymi przechowywanymi na dysku lub w obrazie binarnym (np. tablice partycji). Warto zauważyć, że w przeciwieństwie do polecenia `fdisk` polecenie `mmls` wyświetla również informacje na temat przestrzeni niealokowanej, znajdującej się przed, po lub pomiędzy woluminami.

Poniżej przedstawiamy przykład działania takiego polecenia na obrazie pobranym z witryny Digital Forensics Tool Testing (*http://dftt.sourceforge.net/*).

```
user@forensics:~$ mmls 10-ntfs-disk.dd
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors
     Slot    Start        End          Length       Description
00:  Meta    0000000000   0000000000   0000000001   Primary Table (#0)
01:  -----   0000000000   0000000062   0000000063   Unallocated
02:  00:00   0000000063   0000096389   0000096327   NTFS (0x07)
03:  00:01   0000096390   0000192779   0000096390   NTFS (0x07)
04:  -----   0000192780   0000192783   0000000004   Unallocated
```

Jak łatwo zauważyć, partycja podstawowa rozpoczyna się w pierwszym sektorze dysku, a na dysku możemy wyróżnić dwa woluminy — pierwszy rozciąga się od sektora 63 do sektora 96389, a drugi zajmuje sektory od 96390 do 192779. Ponadto wyniki działania polecenia `mmls` jasno wskazują, że na dysku, za końcem drugiego woluminu, znajdują się jeszcze cztery, niealokowane, dodatkowe sektory (oprócz standardowej, niealokowanej przestrzeni o rozmiarze 63 sektorów, znajdującej się przed pierwszym woluminem dysku).

Kolejną ważną zaletą używania polecenia `mmls` zamiast narzędzi takich jak `fdisk` jest to, że `mmls` wyświetla offsety poszczególnych woluminów w postaci liczby 512-bajtowych sektorów. Wartość offsetu wskazującego początek woluminu do analizy może być następnie bezpośrednio przekazana do innych poleceń pakietu The Sleuth Kit, działających na wyższych warstwach modelu systemu plików.

Polecenie `mmcat` przesyła zawartość wskazanego woluminu na standardowe wyjście STDOUT (domyślnie jest to konsola). Możesz to wykorzystać do pozyskania zawartości interesującego Cię woluminu i jego dalszej analizy przy użyciu innych narzędzi, które nie potrafią działać bezpośrednio na danej kopii binarnej lub dysku.

### *Narzędzia warstwy systemu plików*

Polecenie `fsstat` wyświetla informacje o systemie plików. Rodzaje wyświetlanych informacji mogą się różnić w zależności od badanego systemu plików, ale zazwyczaj w wynikach działania tego polecenia znajdziesz takie informacje jak nazwy woluminów, rozmiar jednostek danych oraz informacje statystyczne o systemie plików. Poniżej przedstawiamy przykładowe wyniki działania tego polecenia dla systemu plików Ext3. Więcej szczegółowych informacji na temat analizy systemu plików Ext3 znajdziesz w rozdziale 5.

```
user@forensics:~$ fsstat ubnist1.casper-rw.gen3.aff
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: Ext3
Volume Name:
Volume ID: 9935811771d9768b49417b0b3b881787
Last Written at: Tue Jan 6 10:59:33 2009
Last Checked at: Sun Dec 28 12:37:56 2008
Last Mounted at: Tue Jan 6 10:59:33 2009
Unmounted properly
Last mounted on:
Source OS: Linux
Dynamic Structure
Compat Features: Journal, Ext Attributes, Resize Inode, Dir Index
InCompat Features: Filetype, Needs Recovery,
Read Only Compat Features: Sparse Super, Has Large Files,
Journal ID: 00
Journal Inode: 8
```

Jak widać na tym nieco okrojonym przykładzie, wykonanie polecenia `fsstat` dostarczyło podstawowych informacji o systemie plików, w tym danych, które mogą mieć kluczowe

znaczenie dla prowadzonego dochodzenia, takich jak data i czas ostatniego zapisu czy data i czas ostatniego zamontowania systemu plików. Poza kilkoma informacjami ogólnymi rodzaj danych wyświetlanych przez polecenie `fsstat` jest bardzo mocno uzależniony od typu badanego systemu plików. W przypadku systemu plików Ext3 polecenie to wyświetla szereg informacji na temat metadanych i innych struktur danych znajdujących się na dysku:

```
METADATA INFORMATION
------------------------------------------------
Inode Range: 1 - 38401
Root Directory: 2
Free Inodes: 36976
Orphan Inodes: 35, 20, 17, 16,
CONTENT INFORMATION
------------------------------------------------
Block Range: 0 - 153599
Block Size: 4096
Free Blocks: 85287
...
```

Zwróć uwagę na fakt, że jedną z wyświetlanych informacji jest rozmiar bloku danych wykorzystywany w tym systemie plików. Jest to niezmiernie ważna i przydatna informacja, zwłaszcza jeżeli chcesz wyszukiwać i pozyskiwać informacje z niealokowanych przestrzeni dysku.

### *Narzędzia warstwy jednostek danych*

Polecenie `blkstat` wyświetla informacje o wskazanej jednostce danych. Ogólnie rzecz biorąc, znajdziesz tutaj status alokacji takiej jednostki danych, choć w przypadku systemów plików rodziny Ext dodatkowo wyświetlane są również informacje o grupie bloków, do której przypisana jest badana jednostka danych.

```
user@forensics:~$ blkstat ubnist1.casper-rw.gen3.aff 521
Fragment: 521
Allocated
Group: 0
```

Polecenie `blkls` wyświetla szczegółowe informacje o jednostkach danych, ale może być również wykorzystywane do pozyskania zawartości całego niealokowanego obszaru danych z systemu plików. Taka możliwość może być bardzo użyteczna na przykład w sytuacji, kiedy chcesz wyszukiwać i odzyskiwać dane z niealokowanej przestrzeni dysku. W przykładzie przedstawionym poniżej polecenie `blkstat` zostało użyte do pozyskania zawartości całej, niealokowanej przestrzeni systemu plików z binarnego obrazu nośnika danych i zapisania jej w pliku na dysku.

```
user@forensics:~$ blkls ubnist1.casper-rw.gen3.aff > ubnist1.casper-rw.gen3.unalloc
user@forensics:~$ ls -lath ubnist1.casper-rw.gen3.unalloc
-rw-r----- 1 cory eng 331M Sep 2 20:36 ubnist1.casper-rw.gen3.unalloc
```

Wykonanie polecenia `blkstat` powoduje przesłanie strumienia danych z niealokowanej przestrzeni systemu plików na standardowe wyjście polecenia (STDOUT). Efekt końcowy takiej operacji jest bardzo zbliżony do użycia polecenia `dd` do odczytywania i zapisywania określonego bloku danych. W kolejnym przykładzie użyjemy polecenia `blkcat` do pozyskania zawartości bloku danych o numerze 521, która następnie zostanie wyświetlona na ekranie za pomocą przeglądarki danych binarnych `xxd` (polecenie to jest częścią pakietu edytora `vim`, który jest domyślnie instalowany w zdecydowanej większości dystrybucji systemu Linux).

```
user@forensics:~$ blkcat ubnist1.casper-rw.gen3.aff 521 | xxd | head
0000000: 0200 0000 0c00 0102 2e00 0000 0200 0000 ........
0000010: 0c00 0202 2e2e 0000 0b00 0000 1400 0a02 ........
0000020: 6c6f 7374 2b66 6f75 6e64 0000 0c00 0000 lost+found.....
0000030: 1400 0c01 2e77 682e 2e77 682e 6175 6673 .....wh..wh.aufs
0000040: 011e 0000 1400 0c02 2e77 682e 2e77 682e .........wh..wh.
0000050: 706c 6e6b 015a 0000 1400 0c02 2e77 682e plnk.Z......wh.
0000060: 2e77 682e 2e74 6d70 021e 0000 0c00 0402 .wh..tmp.......
0000070: 726f 6673 025a 0000 0c00 0302 6574 6300 rofs.Z.....etc.
0000080: 045a 0000 1000 0502 6364 726f 6d00 0000 .Z...cdrom......
0000090: 031e 0000 0c00 0302 7661 7200 013c 0000 ......var..<..
```

Polecenie `blkcalc` jest wykorzystywane do analizy pliku niealokowanej przestrzeni systemu plików, który został utworzony za pomocą polecenia `blkls`. Dzięki poleceniu `blkcalc` możemy sprawdzić, w którym miejscu oryginalnego systemu plików znajduje się określony blok danych z pliku zawierającego zrzut niealokowanej przestrzeni systemu plików. Jest to bardzo przydatne zwłaszcza w sytuacji, kiedy w pliku zawierającym zrzut niealokowanej przestrzeni systemu plików znajdziesz dane istotne dla prowadzonego dochodzenia (ciąg znaków, plik czy inny artefakt) i chcesz odnaleźć lokalizację takiego elementu w oryginalnym systemie plików.

### *Narzędzia warstwy metadanych*

Polecenie `istat` wyświetla informacje o wskazanej strukturze metadanych. W praktyce polecenie to wyświetla wszystkie informacje znajdujące się w danej strukturze metadanych (na przykład informacje o właścicielu, znaczniki daty i czasu, informacje o alokacji bloków itp.). Jak zwykle rodzaj i ilość wyświetlanych informacji zależy w dużej mierze od typu badanego systemu plików. Więcej szczegółowych informacji na temat poszczególnych typów systemów plików znajdziesz w kolejnych rozdziałach tej książki.

Poniżej przedstawiamy przykładowe wyniki działania polecenia `istat`, które zostało użyte do wyświetlenia informacji o i-węźle numer 20, znajdującym się w naszym testowym systemie plików Ext3. W wynikach działania możemy znaleźć informacje spotykane w większości innych systemów plików, takie jak status alokacji, informacje o właścicielu, rozmiarze czy znaczniki daty i czasu. Adresy jednostek danych i-węzłów są również często spotykane, ale różne systemy plików wykorzystują je w różny sposób, o czym napiszemy jeszcze nieco później.

```
user@forensics:~$ istat ubnist1.casper-rw.gen3.aff 20
inode: 20
Allocated
Group: 0
Generation Id: 96054594
uid / gid: 0 / 0
mode: rrw-r--r--
size: 123600
num of links: 0
Inode Times:
Accessed: Tue Jan 6 10:59:33 2009
File Modified: Wed Jan 7 07:59:47 2009
Inode Modified: Wed Jan 7 07:59:47 2009
Deleted: Wed Dec 31 16:00:17 1969
Direct Blocks:
28680 0 0 0 0 0 0 28681
0 0 0 0 0 0 0 28683
0 0 0 0 0 0 28684 0
0 0 0 0 0 0 28685
Indirect Blocks:
28682
```

Polecenie `ils` wyświetla struktury metadanych oraz ich zawartość, łącznie z osadzonymi znacznikami daty i czasu, informacjami o właścicielu itp. Jest to jedno z poleceń, którego możesz użyć do utworzenia tak zwanego pliku *bodyfile*, który za pomocą polecenia `mactime` możesz przekonwertować do formatu pozwalającego na analizę zdarzeń w osi czasu (ang. *timeline*) — patrz podrozdział „Inne narzędzia pakietu The Sleuth Kit". Analiza zdarzeń w osi czasu jest jednym z kluczowych elementów podczas prowadzenia analizy śledczej; więcej szczegółowych informacji na ten temat znajdziesz w rozdziale 9.

Jak można się łatwo zorientować po liczbie dostępnych argumentów wywołania, każdy analityk może dostosować sposób działania polecenia `ils` oraz ilość wyświetlanych informacji do własnych potrzeb.

```
user@forensics:~$ ils
Missing image name
usage: ils [-emOpvV] [-aAlLzZ] [-f fstype] [-i imgtype] [-b dev_sector_size]
↪[-o imgoffset] [-s seconds] image [images] [inum[-end]]
-e: Display all inodes
-m: Display output in the mactime format
-O: Display inodes that are unallocated, but were still open (UFS/ExtX only)
-p: Display orphan inodes (unallocated with no file name)
-s seconds: Time skew of original machine (in seconds)
-a: Allocated inodes
-A: Unallocated inodes
-l: Linked inodes
-L: Unlinked inodes
-z: Unused inodes (ctime is 0)
-Z: Used inodes (ctime is not 0)
-i imgtype: The format of the image file (use '-i list' for supported types)
```

```
-b dev_sector_size: The size (in bytes) of the device sectors
-f fstype: File system type (use '-f list' for supported types)
-o imgoffset: The offset of the file system in the image (in sectors)
-v: verbose output to stderr
-V: Display version number
```

Na przykład jeżeli chcesz wyświetlić listę wszystkich i-węzłów, które są alokowane lub które były już kiedyś użyte, możesz to zrobić, dodając w wierszu wywołania polecenia opcje –a oraz –Z:

```
user@forensics:~$ ils -aZ ubnist1.casper-rw.gen3.aff
...
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_crtime|st_mode|
↪st_nlink|st_size
1|a|0|0|1230496676|1230496676|1230496676|0|0|0|0
2|a|0|0|1231268373|1230496676|1231268373|0|755|15|4096
7|a|0|0|1230496676|1230496676|1230496676|0|600|1|4299210752
8|a|0|0|1230496679|0|1230496679|0|600|1|16777216
11|a|0|0|1230496676|1230496676|1230496676|0|700|2|16384
12|a|0|0|1230469846|1230469846|1231311252|0|444|19|0
13|a|0|0|1230615881|1225321841|1230615881|0|755|9|4096
...
```

Polecenie `icat` przesyła zawartość jednostki danych wskazanej przez adres struktury metadanych podany jako argument wywołania polecenia. Na przykład jeżeli plik *file1.txt* wskazuje na i-węzeł 20, który z kolei wskazuje na bloki danych o numerach 30, 31 i 32, to wykonanie polecenia `icat` {*plik-obrazu*} 20 będzie miało taki sam efekt jak wykonanie polecenia `car file1.txt` na zamontowanym systemie plików.

Polecenie `ifind` wyszukuje strukturę metadanych powiązaną z podaną nazwę pliku *lub* strukturę metadanych powiązaną z jednostką danych o podanym adresie. Na przykład aby znaleźć i-węzeł, do którego przypisany jest blok danych o numerze 28680, powinieneś wykonać następujące polecenie:

```
user@forensics:~$ ifind -d 28680 ubnist1.casper-rw.gen3.aff
20
```

### Narzędzia warstwy nazw plików

Polecenie `fls` wyświetla listę nazw plików (skasowanych i alokowanych). Domyślnie polecenie nie przechodzi przez całą strukturę systemu plików, stąd na ekranie pojawi się tylko zawartość głównego katalogu badanego woluminu. Jest to jedno z poleceń, które możesz wykorzystać do utworzenia pliku *bodyfile*, który za pomocą polecenia `mactime` możesz przekonwertować do formatu pozwalającego na analizę zdarzeń w osi czasu (ang. *timeline*) — patrz podrozdział „Inne narzędzia pakietu The Sleuth Kit”. Wykonanie prostego polecenia `fls` {*plik-obrazu*} spowoduje wyświetlenie na ekranie zawartości głównego katalogu systemu plików.

```
user@forensics:~$ fls ubnist1.casper-rw.gen3.aff
d/d 11: lost+found
```

```
r/r 12: .wh..wh.aufs
d/d 7681: .wh..wh.plnk
d/d 23041: .wh..wh..tmp
d/d 7682: rofs
d/d 23042: etc
d/d 23044: cdrom
d/d 7683: var
d/d 15361: home
d/d 30721: tmp
d/d 30722: lib
d/d 15377: usr
d/d 7712: sbin
d/d 13: root
r/r * 35(realloc): .aufs.xino
d/d 38401: $OrphanFiles
```

Zwróć uwagę, że w wierszu opisującym plik *.aufs.xino* znajduje się symbol gwiazdki — oznacza to, że taki plik jest skasowany. Dodatkowo słowo kluczowe *realloc* oznacza, że i-węzeł przypisany początkowo do tego pliku został już realokowany do innego pliku.

Na stronach podręcznika `man` polecenia `fls` znajdziesz szczegółowy opis szeregu innych opcji wywołania tego polecenia. Poniżej przedstawiamy zestaw najważniejszych opcji wykorzystywanych w trybie interaktywnym:

`-d`: *Wyświetla tylko skasowane pliki*
`-l`: *Wyświetla szczegółowe informacje o plikach (podobnie jak polecenie ls -l)*
`-m`: *Wyświetla wyniki w formacie mactime z zachowaniem punktu montowania*
`-p`: *Wyświetla pełną ścieżkę dla każdego pliku*
`-r`: *Wyświetla zawartość podkatalogów*
`-u`: *Wyświetla tylko istniejące pliki*
`-z`: *Pozwala na ustawienie strefy czasowej oryginalnego komputera (np. EST5EDT czy GMT);*
↪*opcja jest użyteczna tylko w połączeniu z opcją* `-l`
`-s liczba_sekund`: *Przesunięcie czasu dla oryginalnego komputera (w sekundach);*
↪*opcja jest użyteczna tylko w połączeniu z opcjami* `-l` *i* `-m`

Zwróć uwagę, że opcja –z pozwalająca na zdefiniowanie strefy czasowej nie ma żadnego znaczenia, jeżeli jednocześnie używasz opcji –m do wygenerowania pliku wejściowego dla polecenia `mactime`; innymi słowy, opcja –z powinna być wykorzystywana tylko wtedy, kiedy wyświetlasz informacje na konsoli.

Polecenie `ffind` wyszukuje nazwy plików odpowiadające strukturom metadanych o podanych numerach. Jak pamiętasz, za pomocą polecenia `ifind` odszukaliśmy i-węzeł o numerze 20, przypisany do bloku danych 28680 w naszym testowym binarnym obrazie systemu plików Ext3. Teraz za pomocą polecenia `ffind` możemy spróbować odszukać nazwę powiązanego z tym i-węzłem pliku.

```
user@forensics:~$ ffind ubnist1.casper-rw.gen3.aff 20
File name not found for inode
```

Jak widać, i-węzeł 20 nie jest obecnie powiązany z żadną nazwą pliku — inaczej mówiąc, jest **osierocony** (ang. *orphaned*). Aby zaspokoić ciekawość, możemy sprawdzić również sąsiednie i-węzły.

```
user@forensics:~$ ffind ubnist1.casper-rw.gen3.aff 19
/root/.pulse-cookie
user@forensics:~$ ffind ubnist1.casper-rw.gen3.aff 21
/root/.synaptic/lock
```

### *Inne narzędzia pakietu The Sleuth Kit*

Polecenie `mactime` generuje plik w formacie pozwalającym na analizę zdarzeń w osi czasu na podstawie plików wejściowych przygotowanych za pomocą polecenia `ils` i (lub) `fls`. Aby wygenerować plik zawierający historię zdarzeń w osi czasu, musisz najpierw przygotować odpowiedni plik *bodyfile*. Jest to prosty plik tekstowy przechowujący metadane systemu plików i nie tylko, zapisany w formacie, w którym poszczególne pola rekordów są od siebie oddzielane symbolem potoku (|), i wykorzystywany jako plik danych wejściowych dla polecenia `mactime`.

```
user@forensics:~$ ils -em ubnist1.casper-rw.gen3.aff > ubnist1.bodyfile
user@forensics:~$ fls -r -m "/" ubnist1.casper-rw.gen3.aff >> ubnist1.bodyfile
```

Wykonanie dowolnego z przedstawionych wyżej poleceń spowoduje utworzenie pliku tekstowego, w którym kolejne wiersze zawierają metadane powiązane z poszczególnymi plikami lub i-węzłami.

```
md5|file|st_ino|st_ls|st_uid|st_gid|st_size|st_atime|st_mtime|st_ctime|st_crtime
0|<ubnist1.casper-rw.gen3.aff-alive-1>
    |1|-/----------|0|0|0|1230496676|1230496676|1230496676|0
0|<ubnist1.casper-rw.gen3.aff-alive-2>
    |2|-/drwxr-xr-x|0|0|4096|1230496676|1231268373|1231268373|0
0|<ubnist1.casper-rw.gen3.aff-alive-3>
    |3|-/----------|0|0|0|0|0|0|0
0|<ubnist1.casper-rw.gen3.aff-alive-4>
    |4|-/----------|0|0|0|0|0|0|0
0|<ubnist1.casper-rw.gen3.aff-alive-5>
    |5|-/----------|0|0|0|0|0|0|0
0|<ubnist1.casper-rw.gen3.aff-alive-6>
    |6|-/----------|0|0|0|0|0|0|0
0|<ubnist1.casper-rw.gen3.aff-alive-7>
    |7|-/rrw-------|0|0|4299210752|1230496676|1230496676|1230496676|0
...
0|/lost+found|11|d/drwx------
    |0|0|16384|1230496676|1230496676|1230496676|0
0|/.wh..wh.aufs|12|r/rr-r-r-
    |0|0|0|1230469846|1230469846|1231311252|0
0|/.wh..wh.plnk|7681|d/drwx------
    |0|0|4096|1230469846|1230469897|1230469897|0
0|/.wh..wh.plnk/1162.7709|7709|r/rrw-r-r-
    |0|0|186|1225322232|1225322232|1230469866|0
```

Kiedy budujesz zestawienie zdarzeń w osi czasu dla prowadzonego dochodzenia, z pewnością będziesz chciał wprowadzić odpowiednie ustawienia strefy czasowej, w której znajdował się badany komputer, czy dodać kilka innych informacji specyficznych dla danego systemu plików. Aby wygenerować proste zestawienie zdarzeń w osi czasu, gdzie poszczególne pola rekordów są od siebie oddzielone przecinkami (format CSV), możesz wykonać następujące polecenie:

```
user@forensics:~$ mactime -b ubnist1.bodyfile -d > ubnist1.timeline.csv
```

Dobrze wykonana analiza zdarzeń w osi czasu może być potężnym narzędziem w arsenale każdego informatyka śledczego. Więcej szczegółowych informacji na ten temat znajdziesz w rozdziale 9.

Polecenie `sigfind` pozwala na wyszukiwanie w obrazach dysków ciągów wartości heksadecymalnych (sygnatur binarnych), począwszy od miejsca o podanym offsecie. Po uruchomieniu polecenie `sigfind` rozpoczyna przeszukiwanie zawartości obrazu dysku pod kątem występowania podanego ciągu wartości heksadecymalnych i wyświetla offsety miejsc, w których takie sygnatury binarne zostały odnalezione. Polecenie `sigfind` może operować zarówno na sektorach, jak i na blokach danych o podanym rozmiarze. Jest to bardzo użyteczne zwłaszcza w sytuacji, kiedy poszukujesz sygnatur binarnych w plikach obrazów o luźnej strukturze danych, takich jak zrzuty zawartości pamięci operacyjnej komputerów czy pliki zawierające zrzut niealokowanej przestrzeni dysku. Polecenie `sigfind` może być przydatne do wyszukiwania plików na podstawie ich sygnatur binarnych (takich jak nagłówki) i pozwala na zminimalizowanie fałszywie pozytywnych trafień, które często mogą zdarzać się w sytuacji, kiedy przeszukujemy takie strumienie danych za pomocą narzędzi podobnych do polecenia `grep`.

Składnia polecenia `sigfind` jest bardzo prosta:

```
sigfind [-b rozmiar] [-o offset] [-t szablon] [-lV] [sygnatura] plik
```

gdzie

`-b` *`rozmiar`*: *rozmiar bloku danych (domyślnie 512 bajtów)*
`-o` *`offset`*: *offset w bloku danych, wyrażony bajtach, gdzie powinna znajdować się poszukiwana sygnatura (domyślnie 0)*
`-l`: *informuje, że sygnatura została podana w formacie little-endian*
`-V`: *wyświetla informacje o wersji*
`-t` *`szablon`*: *nazwa szablonu struktury danych: dospart, ext2, ext3, fat, hfs, hfs+, ntfs, ufs1, ufs2*

Aby zademonstrować działanie polecenia `sigfind`, użyjemy go do odszukania (przynajmniej częściowo) plików PDF znajdujących się w naszym testowym obrazie systemu plików Ext3. Pliki dokumentów zapisanych w formacie PDF rozpoczynają się od ciągu znaków `%PDF`. Jeżeli zamienimy taki ciąg znaków ASCII na ich szesnastkową reprezentację, otrzymamy następujący ciąg wartości heksadecymalnych: `25 50 44 46`. Teraz, korzystając z polecenia `sigfind`, będziemy poszukiwać takiej sygnatury na początku każdego bloku danych w obrazie dysku (rozmiar bloków danych w naszym obrazie odszukaliśmy już wcześniej za pomocą polecenia `fsstat`).

```
user@forensics:~$ sigfind -b 4096 25504446 ubnist1.casper-rw.gen3.aff
Block size: 4096 Offset: 0 Signature: 25504446
Block: 722 (-)
Block: 1488 (+766)
Block: 1541 (+53)
Block: 1870 (+329)
Block: 82913 (+81043)
...
```

Polecenie wyświetla offset w obrazie dysku, wyrażony w liczbie bloków, gdzie znaleziona została poszukiwana sygnatura, oraz offset liczony od poprzedniego trafienia (podany w nawiasach). Polecenie `sigfind` posiada również szereg wbudowanych szablonów struktur danych, co znakomicie ułatwia identyfikację utraconych partycji i innych struktur systemu plików.

Polecenie `hfind` pozwala na wyszukiwanie wartości funkcji skrótu w bazie danych zawierającej listę wartości funkcji skrótów. Takie rozwiązanie jest znacznie szybsze niż wyszukiwanie haszy w pliku tekstowym za pomocą polecenia `grep`.

Polecenie `sorter` wyodrębnia i sortuje pliki według kategorii określonych poprzez analizę ich zawartości. Za pomocą tego polecenia możesz również sprawdzać, czy wartości skrótu poszczególnych plików znajdują się w określonej bazie danych, oraz weryfikować, czy rozszerzenia poszczególnych plików odpowiadają ich rzeczywistej zawartości.

Ostatnie polecenie z tej kategorii, `srch_strings`, to po prostu samodzielna wersja polecenia `strings` stanowiącego część pakietu GNU *binutils*. Polecenie to zostało dołączone do pakietu The Sleuth Kit po to, aby zapewnić mu możliwość wyszukiwania ciągów znaków bez konieczności uprzedniego instalowania całego pakietu *binutils* (co prawda ma to znaczenie tylko w systemach, w których pakiet *binutils* nie jest instalowany domyślnie).

### *Narzędzia warstwy pliku obrazu*

Plik obrazu nośnika danych możemy traktować jako swego rodzaju nową warstwę pośrednią, zastępującą w naszym modelu warstwę dysku. Ze względu na fakt, że warstwa pliku obrazu nośnika danych jest budowana przez analityka, raczej nie powinniśmy się tutaj spodziewać obecności artefaktów istotnych dla prowadzonego dochodzenia. Warto jednak zaznaczyć, że w przypadku kilku formatów kontenerów binarnych obrazów dysków możesz również i tutaj znaleźć kilka interesujących informacji.

Polecenie `img_stat` wyświetla informacje o formacie, w jakim zapisany został dany binarny obraz nośnika danych, łącznie z informacjami o wartościach funkcji skrótu oraz innych metadanych zapisanych w obrazie (o ile oczywiście w danym formacie są one dostępne). W zasadzie polecenie to ma zastosowanie tylko w przypadku plików kontenerów obrazów binarnych. Poniżej przedstawiamy wyniki działania polecenia `img_stat` dla naszego testowego obrazu systemu plików Ext3.

```
user@forensics:~$ img_stat ubnist1.casper-rw.gen3.aff
IMAGE FILE INFORMATION
--------------------------------------------------
Image Type: AFF
Size in bytes: 629145600
```

```
MD5: 717f6be298748ee7d6ce3e4b9ed63459
SHA1: 61bcd153fc91e680791aa39455688eab946c4b7
Creator: afconvert
Image GID: 25817565FO5DFD8CAEC5CFC6B1FAB45
Acquisition Date: 2009-01-28 20:39:30
AFFLib Version: "3.3.5"
```

Polecenie `img_cat` przesyła zawartość pliku obrazu na standardowe wyjście STDOUT. Jest to bardzo wygodny sposób na dokonanie konwersji zawartości kontenera binarnego obrazu nośnika danych na obraz typu RAW.

### *Narzędzia warstwy dziennika systemu plików*

Bardzo wiele współczesnych systemów plików wyposażonych jest w tzw. **mechanizm księgowania transakcji**, czyli inaczej mówiąc, dzienniki, w których przed zapisaniem na dysk rejestrowane są wszystkie zmiany dokonywane w systemie plików. Dzięki takiemu rozwiązaniu znacząco zmniejsza się prawdopodobieństwo utraty danych w wyniku na przykład utraty zasilania — jeżeli utrata zasilania nastąpiła w trakcie zapisu, to informacje zarejestrowane w dzienniku zostaną po przywróceniu zasilania wykorzystane do dokończenia operacji i zapewnienia spójności danych w systemie plików. Biorąc zatem pod uwagę specyfikę działania takiego dziennika, nietrudno zauważyć, że mogą się w nim znajdować informacje, których próżno by szukać w innych miejscach aktywnego systemu plików.

Polecenie `jls` wyświetla wszystkie rekordy i wpisy dziennika systemu plików, a polecenie `jcat` przesyła zawartość wskazanego bloku dziennika na standardowe wyjście STDOUT. Informacje zawarte w dziennikach są silnie uzależnione od typu systemu plików; więcej szczegółowych informacji na ten temat znajdziesz w podrozdziałach poświęconych poszczególnym rodzajom systemów plików w dalszej części książki.

## PODZIAŁ NA PARTYCJE I KONFIGURACJA DYSKÓW

Współcześnie stosowane są dwa główne schematy podziału dysków na partycje: MBR (Master Boot Record) oraz GPT (GUID Partition Table). Schemat GPT został opracowany jako następca starzejącego się schematu MBR. Partycjonowanie metodą MBR pozwalało na utworzenie jedynie czterech partycji podstawowych (ang. *primary partitions*) oraz wykorzystywanie dysków o pojemności maksymalnie 2 TB, a przecież obecnie dyski o większych pojemnościach spotykane są coraz częściej. Format GPT pozwala na wykorzystywanie dysków o pojemnościach maksymalnie 8 ZB[1] i tworzenie do 128 partycji (nie mówiąc już o wielu innych usprawnieniach). W większości przypadków tablice partycji nie zawierają praktycznie żadnych informacji, które miałyby jakieś znaczenie dla prowadzonego dochodzenia. Analiza śledcza tablicy partycji sprowadza się zazwyczaj tylko do pozyskiwania danych niezbędnych do odtworzenia utraconych, uszkodzonych lub celowo usuniętych partycji.

[1] ZB — zettabajt, czyli $10^{21}$ bajtów; dla porównania 1 TB to $10^{12}$ bajtów — *przyp. tłum.*

### Identyfikacja i odtwarzanie partycji

Identyfikacja usuniętych lub utraconych w inny sposób partycji może być przeprowadzona za pomocą polecenia `sigfind`, o którym pisaliśmy już wcześniej. Narzędzie to posiada wbudowany szereg predefiniowanych szablonów struktur danych, które możesz wykorzystać do wyszukiwania i identyfikacji nagłówków tablic partycji czy systemów plików. Aby się o tym przekonać, możemy użyć obrazu testowego numer 10 z projektu Digital Forensics Tool Testing Image (*http://dftt.sourceforge.net/test10/index.html*). Szablon *dospart* pozwala na wyszukiwanie sygnatury składającej się z dwóch bajtów o wartościach `55 AA` (hex) w ostatnich bajtach każdego sektora, co stanowi strukturę charakterystyczną dla partycji MBR.

```
user@ubuntu:~/10-ntfs-autodetect$ sigfind -t dospart 10-ntfsautodetect/
↪10-ntfs-disk.dd
Block size: 512 Offset: 510 Signature: 55AA
Block: 0 (-)
Block: 63 (+63)
Block: 96389 (+96326)
Block: 96390 (+1)
```

Teraz możemy porównać otrzymane wyniki z wynikami działania polecenia `mmls` dla tego samego obrazu dysku:

```
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors
     Slot    Start        End          Length       Description
00: Meta    0000000000 0000000000 0000000001 Primary Table (#0)
01: -----   0000000000 0000000062 0000000063 Unallocated
02: 00:00   0000000063 0000096389 0000096327 NTFS (0x07)
03: 00:01   0000096390 0000192779 0000096390 NTFS (0x07)
04: -----   0000192780 0000192783 0000000004 Unallocated
```

Widzimy, że polecenie `sigfind` zlokalizowało sygnaturę *0x55AA* w sektorze rozruchowym (0), na początku i na końcu pierwszego woluminu (sektory 63 i 96389) oraz na początku kolejnego woluminu (sektor 96390).

> **UWAGA**
> **INNE SCHEMATY PARTYCJONOWANIA DYSKÓW**
>
> Pakiet The Sleuth Kit potrafi również rozpoznawać dwa inne schematy tworzenia woluminów: *slices* firmy Sun, wykorzystywany przez system operacyjny Solaris, oraz *disklabels*, wykorzystywany w systemach operacyjnych bazujących na systemie BSD. W naszej książce nie będziemy się zajmować analizą takich systemów plików, powinieneś jednak pamiętać, że w razie potrzeby możesz do ich analizy używać pakietu The Sleuth Kit.

Warto tutaj wspomnieć, że do odtwarzania partycji utraconych na skutek problemów z dyskiem czy mniej lub bardziej świadomego działania użytkownika możesz użyć znakomitego programu TestDisk, którego autorem jest Christopher Grenier (patrz strona internetowa

*http://www.cgsecurity.org/*). Program obsługuje zarówno surowe obrazy dysków typu RAW, jak i pliki kontenerów zapisane w formacie Expert Witness/E0, wykorzystywane przez pakiet EnCase. Na stronie internetowej CGSecurity znajdziesz doskonałą dokumentację całego pakietu. Program TestDisk możesz zainstalować w systemie Ubuntu za pomocą polecenia `apt-get`. Na stronie CGSecurity znajdziesz również kod źródłowy programu oraz prekompilowane pakiety dla systemów DOS, Windows, OS X oraz Linux.

### Macierze RAID

Macierze RAID (Redundant Array of Inexpensive Disks) grupują wiele dysków fizycznych w jedną logiczną całość, tworząc niskopoziomowe powiązania pomiędzy poszczególnymi dyskami i prezentując je dla systemu operacyjnego w postaci pojedynczego, dużego urządzenia logicznego.

Istnieje kilka podstawowych rodzajów macierzy RAID:

- **RAID 0** — w takiej macierzy dwa lub więcej dysków fizycznych łączy się na poziomie „przeplatanych" bloków danych. Przykładowo: jeżeli mamy dwa dyski, 0 i 1, blok A zostanie zapisany na dysku 0, blok B ma dysku 1, kolejny blok na dysku 0 i tak dalej. Takie rozwiązanie przyspiesza zapisywanie i odczytywanie danych i umożliwia wykorzystanie całej przestrzeni dysków na dane, ale w zamian zwiększa podatność na awarie i utratę danych, gdyż w takiej sytuacji utrata jednego dysku oznacza utratę połowy zapisanych bloków danych.
- **RAID 1** — taka macierz to pewne przeciwieństwo macierzy RAID 0. Poszczególne bloki danych są zapisywane równolegle na dwóch „lustrzanych" dyskach jednocześnie. Takie rozwiązanie przyspiesza zapisywanie i odczytywanie danych, zwiększa odporność na awarie, ale jednocześnie redukuje efektywną pojemność macierzy do połowy całkowitej fizycznej pojemności użytych dysków.
- **RAID 5** — macierz w takiej konfiguracji wymaga użycia co najmniej trzech dysków. Kontroler takiej macierzy dzieli bloki danych i zapisuje poszczególne fragmenty na wszystkich dyskach macierzy, po czym oblicza specjalną *sumę kontrolną* (ang. *parity block*) każdego bloku danych i również fragmentuje ją, zapisując po kolei na wszystkich dyskach macierzy. Dzięki kontroli parzystości zawartość uszkodzonego dysku może być odtworzona na nowym dysku bez utraty danych.

Oprócz wymienionych wyżej podstawowych rodzajów macierzy RAID istnieje również kilka innych, hybrydowych konfiguracji, łączących ze sobą funkcjonalność dwóch wybranych podstawowych konfiguracji. Na przykład RAID 50 (lub jak kto woli, 5+0) to połączenie dwóch macierzy RAID 5 zapisywanych na „przeplatanych" dyskach macierzy RAID 0.

Pakiet The Sleuth Kit nie posiada żadnych wbudowanych mechanizmów pozwalających na obsługę macierzy RAID. W rozdziale 9. przedstawimy pakiet PyFLAG, w którym znajdziesz narzędzie o nazwie *raid_guess.py*, umożliwiające rekonstrukcję macierzy RAID na podstawie istniejącego zestawu obrazów dysków składowych [3]. Jeżeli musisz utworzyć

obraz macierzy, to najlepszym rozwiązaniem będzie skorzystanie z oryginalnej konfiguracji sprzętowej (o ile jest to oczywiście możliwe). Pamiętaj, że istnieje bardzo wiele różnych implementacji macierzy RAID i odtwarzanie ich logicznej struktury wyłącznie na podstawie obrazów dysków może być nie lada wyzwaniem.

## KONTENERY SPECJALNE

Oprócz systemów plików rezydujących w woluminach na dyskach fizycznych możesz również napotkać systemy plików w innych kontenerach. Jednym z dobrych przykładów mogą być kontenery DMG wykorzystywane w systemie Mac OS X, o których wspominaliśmy nieco wcześniej w tym rozdziale. W praktyce najczęściej możesz spotkać się z dwoma innymi rodzajami kontenerów, którymi są obrazy dysków maszyn wirtualnych oraz kontenery obrazów binarnych wykorzystywane przez aplikacje śledcze.

### Obrazy dysków maszyn wirtualnych

Aplikacje wirtualizacyjne, takie jak VMWare, VirtualBox, Virtual PC czy QEMU, pozwalają użytkownikom na uruchamianie na swoich komputerach maszyn wirtualnych często działających pod kontrolą innych systemów operacyjnych niż system gospodarza. Generalnie rzecz biorąc, systemy plików wykorzystywane przez maszyny wirtualne mają postać wirtualnych obrazów dysków — inaczej mówiąc, są to specjalne kontenery, które dla oprogramowania wirtualizacyjnego są „widziane" jako dyski fizyczne. Skoro zatem takie kontenery spełniają rolę dysków dla maszyn wirtualnych, to równie dobrze i my możemy użyć ich jako binarnych obrazów dysków, które możemy poddawać analizie śledczej, i pozyskiwać z nich artefakty. Najpopularniejszym obecnie formatem dysków wirtualnych jest VMDK, wykorzystywany przez oprogramowanie wirtualizacyjne firmy VMWare.

Dyski wirtualne VMWare są definiowane za pomocą specjalnego **pliku deskryptora** (ang. *descriptor file*), w którym zapisana jest konfiguracja pliku lub plików tworzących dany dysk wirtualny oraz specyfikacja tego dysku dla maszyny wirtualnej. Dysk wirtualny składa się z pliku bazowego (lub wielu plików w sytuacji, kiedy został utworzony z szeregu plików składowych o rozmiarze 2 GB każdy). W miarę jak użytkownicy tworzą kolejne migawki maszyny wirtualnej, tworzone są specjalne **pliki różnicowe**, zawierające opis danych, które zostały zmienione w stosunku do pliku bazowego, oraz kolejne pliki deskryptorów, zawierające informacje o pliku bazowym i plikach różnicowych wchodzących w skład danej migawki.

Pełną specyfikację plików VMDK możesz znaleźć w dokumencie PDF na stronie internetowej firmy VMWare, pod adresem *http://www.vmware.com/app/vmdk/?src=vmdk*.

Biblioteka AFFLib posiada wbudowane mechanizmy obsługi kontenerów VMDK, stąd pakiet The Sleuth Kit potrafi zaimportować je do własnych potrzeb (o ile oczywiście został skompilowany z obsługą tej biblioteki). Wszystkie narzędzia z pakietu The Sleuth Kit mogą pracować z kontenerami VMDK po dodaniu w wierszu wywołania specjalnej opcji (`-i`), włączającej mechanizmy biblioteki AFFLib.

**WSKAZÓWKA**

### TWORZENIE KONTENERÓW VMDK Z OBRAZÓW TYPU RAW

W niektórych sytuacjach użytecznym rozwiązaniem może być możliwość uruchomienia obrazu typu RAW bezpośrednio w maszynie wirtualnej. Obecnie istnieją co najmniej dwa projekty udostępniające taką funkcjonalność. LiveView (*http:///liveview.sourceforge.net/*) to aplikacja wyposażona w graficzny interfejs użytkownika, przeznaczona dla systemu Windows (i w ograniczonym zakresie działająca również w systemie Linux), która na podstawie binarnego obrazu typu RAW tworzy wszystkie pliki niezbędne do uruchomienia systemu zawartego w takim obrazie w maszynie wirtualnej VMWare.

Drugim rozwiązaniem jest pakiet Raw2VMDK (*https://github.com/Zapotek/raw2vmdk*), który obecnie nie jest już rozwijany. Jest to wywoływane z wiersza poleceń narzędzie napisane w języku Java, które tworzy plik VMDK powiązany z oryginalnym obrazem typu RAW. Po dokonaniu takiej „konwersji" utworzony plik VMDK możesz wykorzystywać na wiele sposobów, na przykład możesz go dołączyć jako dysk tylko do odczytu do dowolnej istniejącej maszyny wirtualnej.

**UWAGA**

### INNE FORMATY DYSKÓW WIRTUALNYCH

Choć VMDK jest najpopularniejszym obecnie formatem dysku wirtualnego, to oczywiście oprócz niego możesz spotkać również inne formaty, takie jak na przykład:

- **VDI** — to format dysku wirtualnego wykorzystywanego przez pakiet VirtualBox firmy Sun (oprogramowanie typu *open source*).
- **VHD** — to format wykorzystywany przez pakiet Virtual PC firmy Microsoft oraz mechanizmy wirtualizacyjne „wbudowane" w systemy Windows 7 oraz Windows Server 2008
- **QCOW2** — to format dysku wirtualnego wykorzystywany przez pakiet projektu QEMU (oprogramowanie typu *open source*).

Oczywiście w razie potrzeby dyski wirtualne zapisane w takich formatach mogą zostać przekonwertowane do formatu VMDK czy nawet do postaci obrazu typu RAW. Możesz to zrobić na przykład za pomocą polecenia `qemu-img` wchodzącego w skład pakietu QEMU czy też polecenia `vboxmanage` z pakietu VirtualBox.

## Kontenery obrazów binarnych

Do tej pory już całkiem sporo pracowaliśmy z obrazami dysków zapisanymi w specjalnych plikach kontenerów wykorzystywanych przez oprogramowanie śledcze, ale tak naprawdę jeszcze nie do końca wyjaśniliśmy, czym są takie pliki. Ogólnie rzecz biorąc, kontenery obrazów binarnych przeznaczone do zastosowań śledczych w porównaniu z obrazami typu RAW posiadają znacznie bardziej rozbudowaną funkcjonalność. Oprócz kopii binarnej nośnika danych w kontenerach takich stosowane są różne rozwiązania pozwalające na sprawdzanie integralności zawartości kontenera, zapisanie informacji dodatkowych o prowadzonym dochodzeniu, a także skompresowanie i często zaszyfrowanie zawartości kontenera. Oczywiście takie operacje możemy również wykonać na surowym obrazie binarnym typu RAW. Zasadnicza różnica polega jednak na tym, że w przypadku kontenera takie

mechanizmy są już od razu wbudowane w jego format, co zwalnia analityka z konieczności każdorazowego wykonywania takich operacji na surowym obrazie dysku i poświęcania dużej ilości dodatkowego czasu na upewnianie się, że integralność dowodowa obrazu typu RAW pozostała nienaruszona przez cały czas prowadzenia dochodzenia.

### Format EWF/E01

Najpowszechniej spotykanym rodzajem kontenera binarnych obrazów dysków przeznaczonego do zastosowań śledczych jest format EWF (Expert Witness Format), czasami nazywany również formatem E01 z racji domyślnego rozszerzenia nazw plików. Format E01 jest wykorzystywany w popularnym pakiecie śledczym EnCase firmy Guidance Software. Warto tutaj jednak zauważyć, że specyfikacja tego formatu nie jest otwarta, a firma Guidance Software ma tendencje do wprowadzania do niego pewnych zmian wraz z wypuszczaniem na rynek kolejnych edycji pakietu EnCase. Nie zmienia to jednak w niczym faktu, że biblioteka LibEWF posiada wbudowaną obsługę praktycznie wszystkich wariantów plików obrazów generowanych przez różne wersje pakietu EnCase.

Dokumentacja formatu EWF została początkowo udostępniona przez jego autora Andy'ego Rosena z firmy ASRData i rozszerzona przez Joachima Metza podczas jego prac nad projektem LibEWF [4]. Format EWF wspiera kompresję danych, obrazy wieloplikowe (dzielone) i pozwala na zapisywanie w strukturze nagłówka pierwszego segmentu pliku kontenera dodatkowych metadanych (dane analityka, informacje o prowadzonej sprawie czy wartości funkcji skrótu MD5 lub SHA1 obrazu binarnego). Jeżeli jesteś bliżej zainteresowany poznaniem wewnętrznej struktury formatu EWF, powinieneś sięgnąć do jednego z opisanych wyżej źródeł.

### Format AFF

Format AFF (Advanced Forensics Format) to format typu *open source* przeznaczony do przechowywania binarnych obrazów dysków wraz z odpowiednimi metadanymi. Obsługa formatu AFF została zaimplementowana w bibliotece LibAFF, o której już pisaliśmy wcześniej. Pakiet The Sleuth Kit wykorzystuje tę bibliotekę do obsługi kontenerów obrazów zapisanych w formacie AFF. Obrazy zapisane w tym formacie mogą być kompresowane, szyfrowane i podpisywane cyfrowo. Interesującą cechą formatu AFF jest to, że zawartość metadanych zapisanych w obrazie dysku może być rozszerzana — dodatkowe informacje istotne dla prowadzonego dochodzenia mogą być zapisywane bezpośrednio w pliku kontenera.

Obrazy w formacie AFF mogą być zapisywane w trzech różnych wariantach:

- AFF — jest to domyślny format kontenera AFF, składa się z pojedynczego pliku zawierającego zarówno binarną kopię nośnika danych, jak i metadane opisujące daną sprawę.
- AFD — w tej wersji metadane również są zapisywane w kontenerze, który może być podzielony na wiele plików o stałym rozmiarze. Takie rozwiązanie może być bardzo użyteczne w sytuacji, kiedy musimy przenieść obraz dysku do innego systemu, mając do dyspozycji nośniki pamięci masowej o ograniczonych rozmiarach.

- AFM — w tym formacie kontenera binarny obraz dysku zapisany jest w jednym, dużym pliku, ale opisujące go metadane są przechowywane w osobnym pliku.

## HASZOWANIE

Jedną z kluczowych operacji często wykonywanych na różnych etapach dochodzenia jest obliczanie wartości kryptograficznej funkcji skrótu różnych obiektów, czyli inaczej mówiąc, **haszowanie** (ang. *hashing*). Kryptograficzna funkcja haszująca pobiera na wejściu strumień danych (na przykład zawartość pliku) i po dokonaniu skomplikowanych obliczeń zwraca wynik w postaci unikatowego ciągu znaków o stałej długości. Do najpopularniejszych algorytmów haszujących należą MD5 i SHA1. Algorytm MD5 generuje wartości o rozmiarze 128 bitów, a wyniki działania algorytmu SHA1 mają 160 bitów. Istnieją również inne, bardziej złożone algorytmy obliczania funkcji SHA, które zwracają wartości o długościach 256 bitów (SHA256) oraz 512 bitów (SHA512).

Największą zaletą funkcji haszujących jest to, że zmiana nawet jednego bitu w strumieniu danych wejściowych powoduje wygenerowanie zupełnie innej wartości funkcji skrótu. Biorąc pod uwagę tę szczególną właściwość, nietrudno sobie wyobrazić jedno z ich podstawowych zastosowań w informatyce śledczej — weryfikację integralności cyfrowego materiału dowodowego. Wartość funkcji skrótu obliczona dla oryginalnego, cyfrowego materiału dowodowego (np. zawartości dysku) może być porównana z wartością funkcji skrótu obliczoną dla binarnego obrazu takiego nośnika — jeżeli te dwie wartości będą takie same, oznacza to, że zawartość dysku i jego binarnego obrazu są identyczne. Oprócz tego obliczenie wartości funkcji skrótu binarnego obrazu nośnika danych po zakończeniu analizy śledczej może dodatkowo wykazać, że w czasie przeprowadzania ekspertyzy analityk w żaden sposób nie zmienił jego zawartości.

Inne właściwości funkcji haszujących powodują, że mają one szereg kolejnych zastosowań w informatyce śledczej. Ponieważ wartość funkcji skrótu jest obliczana na podstawie zawartości pliku, porównywanie obliczonych wartości skrótów dla różnych plików może pomóc w odnalezieniu kopii takich samych plików o zmienionych nazwach czy w usunięciu dobrze znanych plików (na przykład systemowych) z listy plików przeznaczonych do szczegółowej analizy. Oprócz tego wartości skrótów plików mogą być wykorzystywane do wyszukiwania takich plików na dysku bez konieczności analizy ich zawartości i bez względu na zmiany dokonane w metadanych opisujących takie pliki w systemie plików.

Programy obliczające wartości funkcji skrótu dla popularnych algorytmów, takich jak MD5 czy SHA*, można znaleźć praktycznie na wszystkich platformach. Do prostego obliczania wartości funkcji skrótu pojedynczych plików możesz wykorzystać polecenia `md5sum` czy `sha1sum`, dostępne praktycznie we wszystkich dystrybucjach systemu Linux, jednak używanie takich programów do przygotowania listy wartości funkcji skrótu dla wielu plików czy plików zagnieżdżonych w złożonej strukturze podkatalogów może być zajęciem niezwykle żmudnym i niewdzięcznym. Aby rozwiązać ten problem, Jesse Kornblum utworzył dwa rewelacyjne narzędzia: *md5deep* oraz *hashdeep*.

Pakiet ***md5deep*** składa się z szeregu narzędzi obliczających wartości funkcji skrótu dla wszystkich plików i folderów znajdujących się w podanym katalogu. Sposób prezentacji wyników działania może być dostosowywany do bieżących potrzeb użytkownika. Narzędzia wchodzące w skład pakietu pozwalają na obliczanie funkcji skrótu według wszystkich najpopularniejszych algorytmów haszowania, takich jak MD5, SHA* i innych. Pakiet ***hashdeep*** to nieco nowsze i bardziej zaawansowane narzędzie pozwalające na przeprowadzanie audytów plików opartych na wartościach funkcji skrótu. Program pozwala na wygenerowanie bazowych wartości funkcji skrótów dla danego zestawu plików i następnie raportowanie podczas kolejnych przebiegów haszowania wszystkich wykrytych zmian, takich jak pliki, które zostały usunięte, pliki, których zawartość została zmodyfikowana, pliki, które zostały przeniesione z jednej lokalizacji do innej, czy wreszcie pliki nowe, których nie było w oryginalnym, bazowym zestawie plików. Więcej szczegółowych informacji na temat składni poszczególnych poleceń, obsługiwanych algorytmów haszowania i sposobu ich użycia znajdziesz na stronie internetowej projektu *md5deep*. Znajdziesz tam również kod źródłowy pakietów oraz prekompilowane pliki binarne dla systemu Windows [5].

Jak już wspominaliśmy wcześniej, fakt, że zmiana nawet jednego bita danych w strumieniu wejściowym powoduje wygenerowanie zupełnie różnej wartości funkcji skrótu, jest jedną z najbardziej pożądanych właściwości tego mechanizmu, pozwalających na weryfikację zawartości oraz integralności cyfrowego materiału dowodowego. Jeżeli jednak będziesz chciał wykazać, że dwa pliki są bardzo do siebie *podobne*, ale jednak nie identyczne, to standardowa funkcja skrótu okaże się w takiej sytuacji bezużyteczna — za pomocą funkcji haszującej możesz wykazać, że pliki są różne, ale nie będziesz wiedział, *jak bardzo* się od siebie różnią. Jesse Kornblum opracował rozwiązanie również i tego problemu, tworząc pakiet o nazwie *ssdeep*, który przeprowadza tzw. **haszowanie rozmyte**. Mówiąc w uproszczeniu, program *ssdeep* najpierw dzieli plik wejściowy na szereg bloków danych, następnie oblicza wartości funkcji skrótu dla poszczególnych fragmentów i na koniec używa utworzonej w ten sposób listy skrótów do określenia podobieństwa dwóch plików. Rozmiar okna haszowania (czyli inaczej mówiąc, rozmiar bloków danych, na jakie dzielony jest plik wejściowy) może być ustalany przez użytkownika.

Sposób działania polecenia `ssdeep` przedstawimy na przykładzie poniżej. Aby przygotować dane wejściowe, utworzyliśmy plik o nazwie *lorem1.txt* zawierający akapit tekstu oraz plik o nazwie *lorem2.txt* zawierający dokładnie taki sam tekst, z tym że pierwsze słowo tekstu zostało zapisane z wielkiej litery. Jak się można było spodziewać, proste wartości funkcji skrótu MD5 obu plików znacząco różnią się od siebie:

```
MD5 (lorem1.txt) = ea4884844ddb6cdc55aa7a95d19815a2
MD5 (lorem2.txt) = 9909552a79ed968a336ca3b9e96aca66
```

Teraz możemy rozpocząć obliczanie wartości rozmytych funkcji skrótu. Aby to zrobić, wystarczy wywołać polecenie `ssdeep` bez żadnych opcji, podając jedynie jako argumenty wywołania nazwy obu plików.

```
ssdeep,1.1--blocksize:hash:hash,filename
24:FPYOEMR7SlPYzvH6juMtTtqULiveqrTFIoCPddBjMxiAyejao:
   9YfQ7qYza6MdtiHrTKoCddBQxiwd,"/home/cory/ssdeep-test/lorem1.txt"
24:lPYOEMR7SlPYzvH6juMtTtqULiveqrTFIoCPddBjMxiAyejao:
   dYfQ7qYza6MdtiHrTKoCddBQxiwd,"/home/cory/ssdeep-test/lorem2.txt"
```

Poprzez proste, wizualne porównanie otrzymanych wartości rozmytych funkcji skrótu możemy się zorientować, że oba pliki są niemal identyczne, z wyjątkiem pierwszego bloku, gdzie dokonaliśmy modyfikacji. Zamiast tego możemy również uruchomić program *ssdeep* w trybie analizy katalogu, w którym program dokona porównania zawartości wszystkich plików znajdujących się w podanym katalogu. Aby to zrobić, powinieneś wywołać polecenie `ssdeep` z opcją `–d`, tak jak to zostało przedstawione poniżej:

```
user@ubuntu:~/ssdeep-test$ ssdeep -d *
/home/user/ssdeep-test/lorem2.txt matches /home/user/ssdeeptest/lorem1.txt (99)
```

Pełną dokumentację programu wraz z kodem źródłowym oraz prekompilowanymi plikami instalacyjnymi dla systemu Windows znajdziesz na stronie projektu *ssdeep* [7].

**UWAGA**

### KOLIZJE W ALGORYTMACH HASZOWANIA

W ciągu kilku ostatnich lat w literaturze fachowej pojawiły się doniesienia o przeprowadzeniu kilku udanych ataków na algorytm MD5, w trakcie których dzięki zastosowaniu bardzo wyrafinowanego aparatu kryptologicznego badacze byli w stanie utworzyć dwa pliki o różnej zawartości, dla których algorytm MD5 generował identyczną wartość funkcji skrótu. Warto jednak zauważyć, że wszystkie upublicznione do tej pory ataki polegały na próbach wygenerowania *kolizji MD5* w sytuacji, gdy strona atakująca posiada pełną kontrolę nad obydwoma plikami biorącymi udział w kolizji i może odpowiednio dostosowywać ich zawartość. Taki scenariusz nie znajduje zastosowania w zdecydowanej większości sytuacji, w których analityk prowadzący dochodzenie używa funkcji haszujących do weryfikacji integralności cyfrowego materiału dowodowego, porównywania z bazą wartości skrótów znanych, „dobrych" plików lub do selekcji plików istotnych dla prowadzonego dochodzenia. Co więcej, w przypadku jakichkolwiek wątpliwości zawsze możesz skorzystać z narzędzi takich jak *hashdeep*, które generują wartości skrótów, wykorzystując kilka różnych algorytmów, co w zdecydowany sposób zwiększa zaufanie do otrzymanych rezultatów.

## DATA CARVING — ODZYSKIWANIE DANYCH Z NIEALOKOWANEJ PRZESTRZENI NOŚNIKA DANYCH

Jeden ze znanych specjalistów informatyki śledczej powiedział kiedyś, że „(…) kiedy wszystko inne zawiedzie, zaczyna się dłubanina[2]". Odzyskiwanie informacji istotnych dla prowadzonego dochodzenia z nieuporządkowanych strumieni danych (ang. *data carving*) jest niewątpliwie czymś z pogranicza nauki, sztuki i być może nawet odrobiny (czarnej)

[2] *(...) when all else fails, we carve* — z ang. *to carve* oznacza „rzeźbić, drążyć, dłubać" — *przyp. tłum.*

magii. Zagadnienia związane z pozyskiwaniem danych w taki sposób od wielu lat stanowią przedmiot wielu prezentacji i referatów na konferencjach takich jak Digital Forensics Research Workshop oraz punkt zainteresowania wielu wybitnych specjalistów zajmujących się informatyką śledczą. Mówiąc w uproszczeniu, proces ***carvingu danych*** polega na przeszukiwaniu strumienia danych pod kątem występowania specyficznych ciągów bajtów reprezentujących nagłówki plików, określaniu (lub odgadywaniu) lokalizacji znaczników końca plików (stopek) i zapisywaniu strumienia danych pomiędzy nagłówkiem a znacznikiem końca do osobnego pliku na dysku w nadziei odzyskania użytecznej zawartości. Proces odzyskiwania informacji z nieuporządkowanych strumieni danych jest wciąż otwartym zagadnieniem i jednym wielkim poligonem doświadczalnym. Z tego powodu możesz spotkać bardzo wiele programów wykorzystujących różne eksperymentalne metody carvingu danych, ale istnieje również szereg dobrze udokumentowanych i sprawdzonych programów, których możesz używać w zastosowaniach „produkcyjnych".

**WSKAZÓWKA**

### PROGRAM HACHOIR-SUBFILE

Program *hachoir-subfile* to narzędzie przeznaczone do „inteligentnej" identyfikacji plików w strumieniach danych binarnych, w tym takich źródeł jak zrzuty zawartości pamięci operacyjnej czy pliki zawierające zrzut niealokowanej przestrzeni systemu plików. Program działa w bardzo podobny sposób do omawianego już wcześniej polecenia `sigfind`, z tym że wykorzystuje znacznie lepsze algorytmy identyfikacji plików, które w połączeniu z bazą sygnatur znacząco zwiększają efektywność programu i zmniejszają liczbę fałszywie pozytywnych trafień. Pomimo że program *hachoir-subfile* nie jest narzędziem do carvingu danych, to jednak możesz go wykorzystywać do identyfikacji położenia plików w strumieniach danych, które następnie będą odzyskiwane za pomocą innych metod. Poszczególne programy wchodzące w skład pakietu *hachoir* będziemy szczegółowo omawiać w rozdziale 8.

## Foremost

Foremost to program przeznaczony do carvingu danych, napisany przez Jesse Kornbluma i Krisa Kendalla z agencji Air Force Office for Special Investigations i następnie zaktualizowany przez Nicka Mikusa z Naval Postgraduate School. Program wykorzystuje bazę zdefiniowanych nagłówków i stopek plików oraz wewnętrznych struktur danych do efektywnego wyszukiwania i odzyskiwania plików z nieuporządkowanych strumieni danych. Pełną listę obsługiwanych typów plików możesz znaleźć w dokumentacji programu, ale wystarczy tutaj wspomnieć, że za jego pomocą możesz odzyskiwać pliki takie jak obrazy w formacie JPEG, dokumenty pakietów biurowych, pliki archiwów i wiele innych. Jeżeli jakiś rodzaj pliku nie jest domyślnie obsługiwany, możesz dodać jego definicję w pliku *foremost.conf*. Więcej szczegółowych informacji na temat analizy plików i ich zawartości znajdziesz w rozdziale 8.

Pakiet Foremost możesz zainstalować w systemie Ubuntu za pomocą polecenia `apt-get`, poprzez samodzielne skompilowanie kodu źródłowego lub poprzez zainstalowanie gotowych, prekompilowanych pakietów dostępnych na stronie internetowej projektu *http://foremost.sourceforge.net/*. Do najważniejszych opcji wywołania programu należą:

`-d` – *włącza pośrednie wykrywanie bloków danych (tylko dla uniksowych systemów plików)*
`-i` – *nazwa pliku wejściowego (domyślnie STDIN)*
`-a` – *zapisuje wszystkie znalezione nagłówki bez sprawdzania poprawności*
↪ *(przydatne w przypadku uszkodzonych plików)*
`-w` – *zapisuje tylko raport z wyszukiwania, bez odzyskiwania żadnych plików*
`-o` – *nazwa katalogu wyjściowego (domyślnie STDOUT)*
`-c` – *nazwa pliku konfiguracyjnego, który będzie użyty (domyślnie foremost.conf)*
`-q` – *włącza tryb szybkiego wyszukiwania, w tym trybie program poszukuje sygnatur tylko*
↪ *na początku i na końcu 512-bajtowych bloków danych*

Aby zademonstrować działanie programu, użyjemy pliku obrazu binarnego udostępnionego w ramach konkursu Digital Forensics Research Workshop 2006 Carving Challenge [8]. W naszym przykładzie wykorzystamy opcję –v, która powoduje, że program wyświetla szczegółowe informacje o przebiegu całego procesu.

```
user@ubuntu:~/dfrws $ foremost -v -i dfrws-2006-challenge.raw
Foremost version 1.5.4 by Jesse Kornblum, Kris Kendall, and Nick Mikus
Audit File
Foremost started at Sat Dec 10 21:51:55 2010
Invocation: foremost -v -i dfrws-2006-challenge.raw
Output directory: /home/user/dfrws/output
Configuration file: /usr/local/etc
Processing: dfrws-2006-challenge.raw
|------------------------------------------------------------------
File: dfrws-2006-challenge.raw
Start: Sat Jan 1 21:51:55 2011
Length: Unknown
Num Name (bs=512) Size   File Offset    Comment
0:  00003868.jpg  280 KB 1980416
1:  00008285.jpg  594 KB 4241920
2:  00011619.jpg  199 KB 5948928
3:  00012222.jpg    6 MB 6257664
4:  00027607.jpg  185 KB 14134784
5:  00031475.jpg  206 KB 16115200
6:  00036292.jpg  174 KB 18581504
7:  00040638.jpg  292 KB 20806656
8:  00041611.jpg    1 MB 21304832
9:  00045566.jpg  630 KB 23329792
10: 00094846.jpg  391 KB 48561152
11: 00000009.htm   17 KB 4691
12: 00004456.htm   22 KB 2281535
13: 00027496.htm  349 KB 14078061
14: 00028244.htm   50 KB 14460928
15: 00029529.htm  183 KB 15118957
16: 00032837.doc  282 KB 16812544
17: 00045964.doc   71 KB 23533568
18: 00028439.zip  157 KB 14560768
19: 00030050.zip  697 KB 15385752
20: 00045015.zip  274 KB 23047680
21: 00007982.png    6 KB 4086865       (1408 x 1800)
```

```
22: 00033012.png   69 KB 16902215      (1052 x 360)
23: 00035391.png   19 KB 18120696      (879 x 499)
24: 00035431.png   72 KB 18140936      (1140 x 540)
*|
Finish: Sat Jan 1 21:51:57 2011
25 FILES EXTRACTED
jpg:= 11
htm:= 5
ole:= 2
zip:= 3
png:= 4
```

Warto tutaj zauważyć, że ze względu na specjalnie przygotowaną fragmentację zawartości naszego binarnego obrazu dysku większość odzyskanych plików nie będzie do końca identyczna jak ich oryginały. Na konferencji Digital Forensics Research Workshop w roku 2007 Simson Garfinkel przedstawił raport ze swoich badań, z którego wynikało, że statystycznie rzecz biorąc, większość plików odzyskiwanych z dowolnego woluminu będzie miała zachowaną ciągłość, a pliki pofragmentowane najczęściej są podzielone na dwie części, pomiędzy którymi znajduje się tylko jeden „obcy" blok danych [9].

**WSKAZÓWKA**

### INNE NARZĘDZIA DO CARVINGU DANYCH

Scalpel to program przeznaczony do carvingu danych. Pierwsza wersja tego programu powstała w roku 2005 i była oparta na pakiecie Foremost w wersji 0.69, którego kod został praktycznie napisany od początku na nowo i zoptymalizowany pod kątem zwiększenia wydajności odzyskiwania danych. Najnowsza wersja pakietu, 2.0, pojawiła się w roku 2013 [10].

PhotoRec to zaawansowane, wieloplatformowe narzędzie do carvingu danych, udostępniane jako część pakietu TestDisk, omawianego nieco wcześniej — w podrozdziale „Identyfikacja i odtwarzanie partycji". Szczegółową dokumentację programu znajdziesz na stronie internetowej autora [11].

Najczęściej z koniecznością odzyskiwania danych z niealokowanych przestrzeni dysku spotykamy się w sytuacji, kiedy metadane opisujące interesujące nas pliki w systemie dysków już nie istnieją lub nie są powiązane z tymi plikami. W takich wypadkach bardzo użytecznym rozwiązaniem może być uprzednie zapisanie za pomocą polecenia `blkls` zawartości całej niealokowanej przestrzeni dysku w osobnym pliku, co pozwala na potencjalne wyeliminowanie fragmentacji spowodowanej przez aktualnie alokowane bloki danych.

## TWORZENIE BINARNYCH KOPII NOŚNIKÓW DANYCH

Tworząc kopię nośnika danych przeznaczoną do zastosowań śledczych, staramy się utworzyć binarną reprezentację oryginalnego nośnika, która będzie jak najbardziej zbliżona do oryginału. Jest to proces w pewnym sensie bardzo podobny do sytuacji, w której policjanci

zabezpieczają miejsce zbrodni specjalną taśmą. Zadaniem tej taśmy jest powstrzymanie osób postronnych od wchodzenia na chroniony teren, co przekłada się na zmniejszenie liczby zmian wprowadzanych na miejscu zbrodni oraz zwiększenie ilości i jakości śladów, które mogą zabezpieczyć i wykorzystać wykwalifikowani specjaliści z laboratorium kryminalistycznego.

Wyobraź sobie teraz sytuację, w której taki specjalista może wykonać dokładną *kopię* miejsca zbrodni. Oczywiście w rzeczywistym świecie takie założenie pozostaje tylko mrzonką, ale w świecie informatyki śledczej to jest dokładnie to, co osiągamy poprzez utworzenie binarnej kopii nośnika danych.

Prawidłowo przeprowadzony proces tworzenia kopii binarnej pozwala na uzyskanie dokładnego duplikatu zawartości źródłowego nośnika danych. Poprzez określenie *dokładny duplikat* rozumiemy, że utworzona kopia zawiera dokładną, wykonaną bit po bicie replikę zawartości oryginalnego nośnika danych. Na oryginalnym nośniku nie mogą pozostać żadne dane, które nie miałyby swojej kopii w binarnym obrazie tego nośnika. Idealny proces tworzenia kopii binarnej nie może w żaden sposób zmieniać zawartości oryginalnego nośnika, pomijać pozyskania jakiegokolwiek obszaru danych ani umieszczać w kopii żadnych danych, których nie ma na oryginalnym nośniku.

Specjalista pracujący w laboratorium kryminalistycznym, który bada broń będącą potencjalnym narzędziem zbrodni, pracuje na oryginalnym egzemplarzu tej broni. Dlaczego więc informatyk śledczy nie może postępować w taki sam sposób? Powodów jest co najmniej kilka. Po pierwsze i najważniejsze, informatyk śledczy poddaje badaniu *dokładną, binarną kopię* oryginalnego cyfrowego materiału dowodowego. W przypadku tradycyjnego śledztwa to właśnie badany egzemplarz broni jest najlepszym dostępnym *materiałem dowodowym* i wykonanie jego dokładnej kopii po prostu nie jest możliwe. W przypadku cyfrowego materiału dowodowego możemy utworzyć tyle kopii binarnych, ile będzie nam potrzebne, i każda z tych kopii będzie miała taką samą wartość dowodową jak oryginał. Ponadto praca z oryginalnym cyfrowym materiałem dowodowym może być bardzo niebezpieczna, ponieważ dane cyfrowe mogą być w prosty sposób zmodyfikowane czy nawet skasowane. Jeżeli wykorzystujemy oryginalny, cyfrowy materiał dowodowy tylko raz, do utworzenia kopii binarnej, to w ten sposób zdecydowanie minimalizujemy możliwość przypadkowego zmodyfikowania jego zawartości. Kolejną zaletą pracy na kopii binarnej jest to, że jeżeli popełnimy błąd, w wyniku którego zawartość kopii zostanie w niezamierzony sposób zmieniona lub nawet zniszczona, po prostu możemy wykonać kolejną kopię binarną oryginalnego nośnika.

### Skasowane dane

Kolejnym powodem, dla którego informatycy śledczy pracują wyłącznie na binarnych kopiach oryginalnego materiału dowodowego, jest *kompletność procesu*. Prosta analiza aktywnego systemu plików przeprowadzana na poziomie systemu operacyjnego z punktu widzenia analizy śledczej nie jest w żaden sposób wystarczająca. Na większości badanych woluminów można znaleźć wiele istotnych dla prowadzonego dochodzenia danych, które normalnie

nie są widoczne na poziomie systemu operacyjnego. Przykładem takich artefaktów mogą być różne rodzaje „skasowanych danych".

- **Pliki skasowane** (ang. *deleted files*) — takie pliki można w większości przypadków odzyskać w całości lub przynajmniej części. Generalnie takie określenie odnosi się do plików, które nie są już dla użytkownika widoczne w danym katalogu i których nazwy, struktura metadanych i jednostki danych zostały w systemie plików oznaczone jako wolne. Nie zmienia to jednak faktu, że dla takich plików połączenia pomiędzy warstwami modelu systemu plików pozostają nienaruszone i mogą zostać odtworzone za pomocą odpowiednich technik. Odtwarzanie skasowanych plików polega na odszukaniu powiązań pomiędzy nazwami plików a odpowiednimi strukturami metadanych i następnie na odzyskaniu zawartości przypisanych do nich jednostek danych.
- **Pliki osierocone** (ang. *orphaned files*) — są bardzo podobne do plików skasowanych, z tym że w przypadku takich plików połączenia pomiędzy nazwami plików i strukturami metadanych zostały przerwane. W takim przypadku odzyskiwanie danych (oraz odpowiednich struktur metadanych) nadal jest możliwe, choć mocno utrudnione, ponieważ nie ma żadnej bezpośredniej korelacji pomiędzy nazwą pliku a blokami danych.
- **Pliki niealokowane** (ang. *unallocated files*) — w przypadku takich plików połączenia pomiędzy nazwami plików i strukturami metadanych zostały przerwane, a odpowiadające im wpisy w systemie plików i bloki danych zostały ponownie wykorzystane dla innych plików. W takich sytuacjach jedyną szansą na częściowe odzyskanie zawartości takich plików jest carving danych z niealokowanej przestrzeni nośnika w celu odtworzenia zawartości bloków danych należących do oryginalnych plików (pod warunkiem że nie zostały jeszcze ponownie użyte).
- **Pliki nadpisane** (ang. *overwritten files*) — w przypadku takich plików jedna lub więcej jednostek danych zostało realokowane i nadpisane przez zawartość innego pliku. Pełne odtworzenie zawartości nadpisanych plików jest niemożliwe, a częściowe odzyskanie zawartości jest uzależnione od stopnia nadpisania oryginalnych bloków danych. Pliki, których nazwy i (lub) struktury metadanych pozostały nienaruszone, ale których kilka lub nawet wszystkie bloki danych zostały nadpisane, są często określane jako *usunięte/nadpisane* (ang. *deleted/overwritten*) lub *usunięte/realokowane* (ang. *deleted/reallocated*).

### File slack — niewykorzystana przestrzeń na końcu pliku

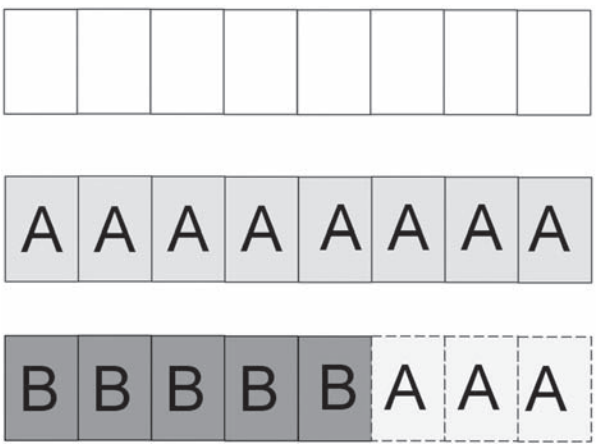
Jak już wspominaliśmy wcześniej, minimalny obszar, jaki może być alokowany dla pliku w danym woluminie, to pojedynczy blok. Jeżeli przyjmiemy, że rozmiar bloku danych to 4 kB (na standardowym dysku z 512-bajtowymi sektorami), oznacza to, że plik tekstowy zawierający tylko jedną literę, na przykład *a*, zajmuje na dysku aż osiem sektorów. Naszym wkładem do tego pliku był jeden bajt, czyli litera *a* — skąd zatem wzięło się pozostałych 4095 bajtów zapisanych na dysku?

Odpowiedź na to pytanie jest taka sama jak zawsze: to zależy. W różnych systemach plików i różnych systemach operacyjnych odbywa się to w różny sposób, ale w ogólnym przypadku działa to tak:

- Klaster, który ma zostać użyty, jest zaznaczany jako „alokowany" i przypisywany do struktury metadanych powiązanej z plikiem.
- Znak *a*, a następnie kolejnych 511 pustych bajtów (hex 00) zostają zapisane w pierwszym sektorze.

Uważny czytelnik zapewne szybko się zorientuje, że nie napisaliśmy ani słowa na temat tego, jak zapisywanych jest na dysku kolejnych 7 sektorów. Nie, to nie jest przeoczenie — one po prostu nie są zapisywane na dysku… W tych kolejnych siedmiu sektorach znajdują się zatem takie dane, jakie zostały w nich zapisane podczas *poprzedniej* alokacji. Taki obszar nazywamy **niewykorzystaną przestrzenią na końcu pliku** (ang. ***file slack*** lub *slack space*).

Na rysunku 3.2 przedstawiono proces powstawania obszaru *file slack* w trzech kolejnych ujęciach tego samego klastra składającego się z ośmiu bloków. W pierwszym rzędzie widzimy osiem pustych, niealokowanych bloków. Teraz tworzony jest plik A. Do niego zostaje alokowanych osiem bloków, które zostają całkowicie wypełnione danymi. Następnie plik A zostaje „skasowany" i jakiś czas później pierwszych pięć bloków jest realokowanych i nadpisanych zawartością pliku B. Taka operacja pozostawia ostatnie 3 bloki tego obszaru wypełnione danymi z pliku A, które są co prawda niealokowane, ale których zawartość możemy odtworzyć.




**RYSUNEK 3.2.**

Obszar file slack — niewykorzystana przestrzeń na końcu pliku

**UWAGA**

### OBSZAR RAM SLACK

Wszystkie współczesne systemy operacyjne dopełniają zapisywane na dysku sektory danych pustymi bajtami (hex 00), ale nie zawsze tak bywało. Na przykład system MS-DOS i wczesne, bazujące na systemie DOS wersje systemu Windows dopełniały zapisywane na dysku sektory danych zawartością tego obszaru pamięci operacyjnej, który akurat sąsiadował z danymi zapisywanymi na dysku. Taki obszar danych, znajdujący się pomiędzy alokowanymi danymi pliku i początkiem kolejnego, poprzednio alokowanego sektora danych, nosił nazwę *RAM slack*. Co ciekawe, ze względu na swoją specyfikę w obszarze *RAM slack* mogły się znajdować dane, które nigdy nie były wprost zapisywane na dysku, takie jak klucze kryptograficzne czy hasła.

**WSKAZÓWKA**

### WOLUMIN CZY DYSK?

Kiedy tworzysz binarną kopię nośnika danych, urządzeniem wejściowym w większości przypadków będzie fizyczny dysk (na przykład */dev/sda*). Zdarzają się jednak sytuacje, w których lepszym rozwiązaniem będzie utworzenie binarnej kopii wybranego woluminu (na przykład */dev/sda1*). Dobrym przykładem takiej sytuacji jest tworzenie binarnej kopii zawartości macierzy RAID. Utworzenie binarnych obrazów poszczególnych dysków fizycznych wchodzących w skład macierzy RAID wymagałoby później odbudowania woluminu macierzy, co jak już opisywaliśmy wcześniej, może się okazać bardzo trudne albo nawet wręcz niemożliwe. Innym przykładem może być praca z woluminem SAN (Storage Area Network) — w przypadku takich urządzeń utworzenie binarnych kopii poszczególnych dysków fizycznych często po prostu nie wchodzi w grę.

## Polecenie dd

Polecenie `dd` to jedno z najbardziej elementarnych narzędzi pozwalających na utworzenie binarnej kopii nośnika danych. Ponieważ polecenie to jest dostępne w niemal wszystkich systemach operacyjnych wywodzących się z rodziny UNIX i jednocześnie stanowi bazę dla wielu innych narzędzi śledczych przeznaczonych do tworzenia kopii binarnych, poznanie zasad jego działania z pewnością przyniesie korzyści każdemu analitykowi. Mówiąc w uproszczeniu, polecenie `dd` kopiuje dane z jednego miejsca do drugiego. Użytkownik może dodawać w wierszu wywołania polecenia różne argumenty i opcje zmieniające sposób jego działania, ale podstawowa składnia wywołania polecenia jest bardzo prosta. W przedstawionym poniżej przykładowym ekranie pomocy tego polecenia najważniejsze opcje zostały wyróżnione pogrubioną czcionką:

```
user@forensics:~$ dd --help
Usage: dd [OPERAND]…
  or: dd OPTION
Copy a file, converting and formatting according to the operands.
  bs=BYTES force ibs=BYTES and obs=BYTES
  cbs=BYTES convert BYTES bytes at a time
  conv=CONVS convert the file as per the comma separated symbol list
  count=BLOCKS copy only BLOCKS input blocks
```

```
ibs=BYTES read BYTES bytes at a time
if=FILE read from FILE instead of stdin
iflag=FLAGS read as per the comma separated symbol list
obs=BYTES write BYTES bytes at a time
of=FILE write to FILE instead of stdout
oflag=FLAGS write as per the comma separated symbol list
seek=BLOCKS skip BLOCKS obs-sized blocks at start of output
skip=BLOCKS skip BLOCKS ibs-sized blocks at start of input
status=noxfer suppress transfer statistics
```

Na przykład chcąc skopiować zawartość jednego dysku na drugi, możesz wykonać następujące polecenie:

```
dd if=/dev/sda of=/dev/sdb bs=4096
```

Po uruchomieniu polecenie `dd` będzie odczytywało dane z pierwszego urządzenia po 4096 bajtów na raz i zapisywało dane na drugim urządzeniu, również po 4096 bajtów na raz. Jeżeli wywołując polecenie, nie zdefiniujesz rozmiaru bloku danych (opcja `bs=`; *block size*), polecenie `dd` domyślnie ustawi rozmiar bloku na 512 bajtów, co w praktyce spowoduje dosyć znaczące spowolnienie procesu kopiowania.

Klonowanie dysków jest ciekawą możliwością, jednak z punktu widzenia informatyka śledczego ma ono dosyć ograniczone zastosowanie. W zdecydowanej większości przypadków znacznie bardziej interesująca dla nas będzie możliwość utworzenia *binarnego obrazu dysku*, czyli pliku zawierającego binarną kopię całej zawartości źródłowego nośnika danych. Wykonanie takiej operacji również jest bardzo proste:

```
user@forensics:~$ sudo dd if=/dev/sdg of=dd.img bs=32K
[sudo] password for user:
60832+0 records in
60832+0 records out
1993342976 bytes (2.0 GB) copied, 873.939 s, 2.3 MB/s
```

Najważniejszymi informacjami wyświetlanymi przez polecenie `dd` na ekranie konsoli są wiersze *records in* oraz *records out*. Po pierwsze, wartości w obu wierszach są identyczne i bardzo dobrze — oznacza to po prostu, że podczas kopiowania nie utraciliśmy żadnych danych (na przykład na skutek awarii dysku, braku możliwości zapisu danych w pliku wyjściowym czy z innych przyczyn). Po drugie, liczba *60832+0* rekordów oznacza, że dokładnie tyle 32-kilobajtowych rekordów zostało odczytanych z dysku źródłowego i następnie zapisanych w pliku obrazu binarnego. Jeżeli tworzylibyśmy obraz dysku o rozmiarze, który nie jest pełną wielokrotnością 32 kilobajtów, to zamiast *+0* wyświetlona zostałaby informacja *+1*, wskazująca, że na koniec został odczytany i zapisany rekord o niepełnych rozmiarach (rekord częściowy).

Jedną z innych bardzo przydatnych opcji polecenia `dd` jest `conv`. Jeżeli próbujesz utworzyć obraz binarny dysku uszkodzonego lub działającego niestabilnie, możesz użyć opcji `conv=noerror,sync`, która spowoduje, że wszelkie błędy odczytu będą ignorowane, a w miejsce danych, których nie udało się odczytać, polecenie `dd` zapisze bloki danych

wypełnione pustymi bajtami (hex 00). Jeżeli dysk źródłowy jest naprawdę w nie najlepszym stanie (czyli mówiąc kolokwialnie, „ledwo zipie"), możesz dołożyć opcję `iflag=direct`, która powoduje włączenie trybu bezpośredniej komunikacji z dyskiem z pominięciem bufora sterownika, oraz zredukować rozmiar kopiowanego bloku do 512 bajtów, dzięki czemu ilość błędnie odczytanych bajtów zostanie zredukowana do minimum.

**OSTRZEŻENIE**
**USZKODZONE SEKTORY**

Pamiętaj, że zastosowanie opcji `conv=noerror` *nie jest* polecanym rozwiązaniem podczas tworzenia obrazu binarnego uszkodzonego dysku twardego. Zamiast tego powinieneś skorzystać raczej z programu GNU `ddrescue`, czyli specjalizowanej wersji polecenia `dd`, zaprojektowanej specjalnie do odzyskiwania danych z dysków, które nie bardzo mają ochotę współpracować. Z drugiej strony, zdarza się jednak i tak, że masz do dyspozycji tylko i wyłącznie polecenie `dd` i możesz albo utworzyć częściowy obraz dysku, albo nie robić go wcale…

## Polecenie dcfldd

Jak już wspominaliśmy, polecenie `dd` może być i w praktyce jest często używane do tworzenia binarnych obrazów nośników danych. Warto jednak także powiedzieć, że istnieją specjalne wersje tego polecenia, zoptymalizowane do zastosowań w informatyce śledczej. Jedną z takich wersji jest pakiet *dcfldd*, napisany przez Nicka Harboura na potrzeby Defense Computer Forensics Laboratory. Projekt *dcfldd* wywodzi się w prostej linii z polecenia `dd`, stąd jego podstawowa funkcjonalność jest bardzo podobna. Poza tym pakiet *dcfldd* posiada szereg bardzo ciekawych możliwości, których próżno by szukać w jego pierwowzorze. Większość z tych funkcji obraca się dookoła obliczania wartości funkcji skrótów, weryfikacji integralności kopii binarnych, logowania wykonanych operacji i dzielenia obrazów binarnych na szereg plików o stałej wielkości. Listę dostępnych opcji polecenia `dcfldd` możesz wyświetlić na ekranie, wywołując je z opcją `--help`.

Nie będzie dla nikogo zaskoczeniem, jeżeli stwierdzimy, że tworzenie prostych obrazów binarnych za pomocą polecenia `dcfldd` jest bardzo podobne do tego, co robiliśmy za pomocą polecenia `dd`. Jeżeli nie będziemy korzystać z żadnych dodatkowych funkcji tego pakietu, to w praktyce podstawowa składnia wywołania polecenia będzie dokładnie taka sama jak w poprzednim przypadku. W przykładzie przedstawionym poniżej utworzymy kopię binarną tego samego urządzenia co w przykładzie z poleceniem `dd`, ale przy okazji dla każdego bloku danych o rozmiarze 512MB obliczone zostaną wartości funkcji skrótu MD5 i SHA1.

```
user@forensics:~$ sudo dcfldd bs=32k if=/dev/sdg of=dcfldd.img
  hashwindow=512M hash=md5,sha1 hashlog=dcfldd.hashlog
60672 blocks (1896Mb) written.
60832+0 records in
60832+0 records out
```

### Polecenie dc3dd

Ostatnim poleceniem, o którym będziemy tutaj pisać, jest `dc3dd`, czyli zorientowana na zastosowania w informatyce śledczej wersja polecenia `dd` napisana przez Jessego Kornbluma na potrzeby agencji Cyber Crime Center departamentu obrony Stanów Zjednoczonych. Pakiet *dc3dd* został zaprojektowany jako aktualizacja polecenia `dd`, dzięki czemu zmiany wprowadzane w bazowym pakiecie `dd` pojawiają się w nim znacznie szybciej niż w konkurencyjnym *dcfldd*. Zestaw rozszerzonych funkcji pakietu *dc3dd* jest bardzo zbliżony do pakietu *dcfldd*.

Pracując z poleceniem `dc3dd`, możesz korzystać z takiego samego zestawu opcji, jakiego używaliśmy dla polecenia `dcfldd`.

```
user@forensics:~$ sudo dc3dd bs=32k if=/dev/sdg of=dc3dd.img
  hashwindow=512M hash=md5,sha1 hashlog=dc3dd.hashlog
[sudo] password for user:
warning: sector size not probed, assuming 512
dc3dd 6.12.3 started at 2010-09-03 17:34:57 -0700
command line: dc3dd bs=32k if=/dev/sdg of=dc3dd.img
  hashwindow=512M hash=md5,sha1 hashlog=dc3dd.hashlog
compiled options: DEFAULT_BLOCKSIZE=32768 sector size: 512 (assumed)
md5 0- 536870912: 07c416f8453933c80319c2d89e5533ad
sha1 0- 536870912: a222f5835ed7b7a51baaa57c5f4d4495b1ca1e79
md5 536870912- 1073741824: acac88a20c6d6b364714e6174874e4da
sha1 536870912- 1073741824: 5b69440a15795592e9e158146e4e458ec8c5b319
md5 1073741824- 1610612736: ed9b57705e7ae681181e0f86366b85e6
sha1 1073741824- 1610612736: bc5369977d9a2f788d910b5b01a9a1e97432f928
md5 1610612736- 1993342976: 812c94592ec5628f749b59a1e56cd9ab
sha1 1610612736- 1993342976: bb789315a814159cdf2d2803a73149588b5290ee
md5 TOTAL: 58e362af9868562864461385ecf58156
sha1 TOTAL: 8eaba11cb49435df271d8bc020eb2b46d11902fe
3893248+0 sectors in
3893248+0 sectors out
1993342976 bytes (1.9 G) copied (??%), 908.424 s, 2.1 M/s
dc3dd completed at 2010-09-03 17:50:06 -0700
```

Zwróć uwagę, że polecenie `dc3dd` wyświetla wyniki działania na ekranie konsoli, ale jednocześnie zapisuje je w pliku dziennika zdefiniowanego w wierszu wywołania polecenia za pomocą opcji `hashlog=`. Warto również zauważyć, że po zakończeniu pracy polecenie wyświetla liczbę skopiowanych sektorów dysku, a nie liczbę bloków danych.

**WSKAZÓWKA**

#### TWORZENIE PLIKÓW W FORMACIE EWF (EXPERT WITNESS FORMAT)

Wcześniej czy później z pewnością staniesz wobec konieczności przekazania cyfrowego materiału dowodowego innemu analitykowi. Jednym z możliwych rozwiązań będzie udostępnienie binarnej kopii nośnika danych w postaci pliku kontenera zapisanego w formacie EWF (ang. *Expert Witness Format*), o którym pisaliśmy już wcześniej w podrozdziale zatytułowanym „Kontenery obrazów binarnych". Pakiet EnCase co prawda bez problemów radzi sobie z obrazami binarnymi typu RAW, ale jeżeli otrzymasz polecenie dostarczenia cyfrowego materiału dowodowego w formacie EWF, powinieneś wiedzieć, jak sobie poradzić z takim zadaniem przy użyciu narzędzi typu *open source*.

Polecenie `ewfacquire` jest częścią pakietu LibEWF i pozwala na tworzenie plików w formacie EWF bezpośrednio z poziomu konsoli systemu. Aby uruchomić pakiet, wystarczy wykonać polecenie `ewfacquire` i jako argument wywołania podać nazwę nośnika źródłowego. Program poprosi o podanie kilku informacji niezbędnych do utworzenia pliku obrazu.

Program *guymanager* to narzędzie wyposażone w graficzny interfejs użytkownika, przeznaczone do tworzenia binarnych obrazów nośników danych w formatach RAW, AFF i EWF. Warto zauważyć, że pakiet *guymanager* do obsługi plików w formacie EWF wykorzystuje bibliotekę LibEWF, zatem podczas generowania plików EWF jest to praktycznie funkcjonalny odpowiednik polecenia `ewfacquire`.

## PODSUMOWANIE

W tym rozdziale omówiliśmy podstawowe zagadnienia związane z analizą zawartości dysków i systemów plików. Oprócz tego przedstawiliśmy również najważniejsze zagadnienia z zakresu analizy śledczej, takie jak tworzenie binarnych kopii nośników danych, praca z plikami kontenerów obrazów binarnych czy haszowanie, tworząc fundament dla tematów, które będziemy omawiać w kolejnych rozdziałach.

## BIBLIOGRAFIA

[1] B. Carrier, *File System Forensic Analysis*, Addison-Wesley, Boston 2005.
[2] „TestDisk Step By Step" — CGSecurity; *http://www.cgsecurity.org/wiki/TestDisk_Step_By_Step*.
[3] „RAID Reassembly — A forensic Challenge" RAID Recovery — PyFLAG; *http://pyflag.sourceforge.net/Documentation/articles/raid/reconstruction.html*.
[4] Biblioteka LibEWF; *http://www.forensicswiki.org/wiki/Libewf*.
[5] „Getting Started with Hashdeep"; *http://md5deep.sourceforge.net/start-hashdeep.html*.
[6] J. Kornblum, „Identifying almost identical files using context triggered piecewise hashing"; *http://dfrws.org/2006/proceedings/12-Kornblum.pdf*.
[7] „Getting Started with ssdeep"; *http://ssdeep.sourceforge.net/usage.html*.
[8] Digital Forensics Research Workshop 2006 File Carving Challenge; *http://www.dfrws.org/2006/challenge/dfrws-2006-challenge.zip*.
[9] S.L. Garfinkel, „Carving contiguous and fragmented files with fast object validation"; *http://www.dfrws.org/2007/proceedings/p2-garfinkel.pdf*.
[10] „Scalpel: A Frugal, High Performance File Carver"; *http://dfrws.org/2005/proceedings/richard_scalpel.pdf*.
[11] „PhotoRec Step By Step" — CGSecurity; *http://www.cgsecurity.org/wiki/PhotoRec_Step_By_Step*.

# Skorowidz

## D



## E



## F

## G



## H



## I



## J



## K

## L



## Ł



## M

## N



## O



## P

## R



## S

## Ś



## T



## U



## V



## W



## Z



## Ź

## *Twoja przepustka do świata informatyki śledczej!*

Współczesne systemy informatyczne przetwarzają gigantyczne ilości bezcennych danych. Numery kart płatniczych, przelewy bankowe, historie pacjentów to tylko niektóre z nich. Ich zniszczenie lub kradzież to niewyobrażalne straty, za które winni muszą zostać ukarani. Dlatego coraz większą popularność zdobywa nowa gałąź nauki — informatyka śledcza. Jej celem jest dostarczenie dowodów przestępstw popełnionych w cyfrowym świecie.

Jeżeli chcesz poznać tę dziedzinę, to trafiłeś na doskonałą książkę. Dowiesz się z niej, jak wykorzystać darmowe narzędzia do analizy zawartości dysków twardych oraz odzyskiwania usuniętych danych w systemach operacyjnych Windows, Linux oraz Mac OS X. Ponadto nauczysz się rozpoznawać w systemie te miejsca, w których można znaleźć informacje na temat aktywności użytkowników internetu, oraz zobaczysz, jak uzyskać dostęp do skrzynek pocztowych. Na sam koniec nauczysz się automatyzować najczęściej wykonywane zadania. Książka ta jest doskonałą lekturą dla pasjonatów bezpieczeństwa systemów informatycznych.

**Dzięki tej książce:**

- przygotujesz funkcjonalne stanowisko badawcze
- poznasz system plików i artefakty systemów operacyjnych Windows, Linux, Mac OS X
- nauczysz się analizować zawartość plików
- zaznajomisz się z najlepszymi narzędziami przydatnymi w codziennej pracy



Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl



ISBN 978-83-246-9562-1
9 788324 695621
cena: 54,90 zł