Szczegółowe informacje o plikach backupu

16 marzec 2011 autor: **gani**

Wśród danych składowanych w bazie danych Bacula można znaleźć tajemniczą kolumnę LStat zawierającą swego rodzaju hash odpowiadający rekordom każdego zapisanego na woluminie pliku. Artykuł zawiera informacje o tym, jak odczytać ten zakodowany ciąg znaków oraz co on zawiera.

Wstęp

Całkiem niedawno dołączyłem się do dyskusji na grupie dyskusyjnej **bacula-users** na temat próby odczytania rozmiaru plików zapisanych w backupie. O ile odczytanie rozmiarów plików, które dopiero mają zostać zapisane na woluminy, nie wydaje się trudne (komenda **estimate** *[więcej]*), o tyle odczytanie rozmiarów plików już zapisanych na woluminach może okazać się kłopotliwe. Informacje o rozmiarze takich plików oraz o wiele więcej można znaleźć w tajemniczej kolumnie tabeli **File** o nazwie **LStat**. Pokusiłem się o rozkodowanie ciągu znaków z rekordów **LStat**.

Komenda stat

Do zrozumienia wartości **LStat** z bazy danych Bacula warto zaznajomić się z komendą **stat**, która - jak poniżej przedstawię - ma wiele wspólnego z wartością **LStat**.

Przy pomocy UNIX'owej komendy **stat** można uzyskać wiele informacji na temat żądanego pliku. Są to m.in. informacje o czasach pliku (**ATIME, MTIME, CTIME**), właścicielu pliku (wraz z **UID**), grupie systemowej pliku (wraz **GID**), rozmiarze czy też o uprawnieniach. Przykładowe wywołanie komendy stat może mieć postać:

```
$ stat przepis.txt
  File: `przepis.txt'
  Size: 1009          Blocks: 8          IO Block: 4096   zwykły plik
Device: 801h/2049d    Inode: 565633      Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/    gani)   Gid: ( 1000/    gani)
Access: 2011-03-13 00:28:28.000000000 +0100
Modify: 2011-03-13 00:28:28.000000000 +0100
Change: 2011-03-13 00:28:28.000000000 +0100
```

Część danych, które można otrzymać poprzez komendę **stat**, zapisywana jest przez Bacula do bazy danych. Przez umieszczeniem ich w bazie danych, informacje te są kodowane.

Jak odczytać zakodowane dane LStat

Hash **LStat** dla żądanego pliku można pobrać np. przy pomocy poniższego zapytania:

```
SELECT Path.path AS Path, Filename.name AS Filename, File.LStat AS LStat
FROM Path, Filename, File WHERE Path.pathid=File.pathid AND Filename.
filenameid=File.filenameid AND Filename.name='NAZWA_PLIKU'
```

Dla mojego pliku **bacula.sql** wywołanie będzie miało postać:

```
*sql
Entering SQL query mode.
Terminate each query with a semicolon.
Terminate query mode with a blank line.
SELECT Path.path AS Path, Filename.name AS Filename, File.LStat AS LStat FROM Path, Filename, File WHERE
Path.pathid=File.pathid AND Filename.filenameid=File.filenameid AND Filename.name='bacula.sql';
+-------------------------------------+-----------+-----------------------------------------------------------------+
| path                                | filename  | lstat                                                           |
+-------------------------------------+-----------+-----------------------------------------------------------------+
| /usr/local/bacula/var/bacula/working/ | bacula.sql | gB B50z IGg B A y A FzkI BAA uo BLXToi BLXToi BLXToi A A C |
| /usr/local/bacula/var/bacula/working/ | bacula.sql | gB B502 IGg B A y A F0Uv BAA uw BLXTvj BLXTvk BLXTvk A A C |
| /usr/local/bacula/var/bacula/working/ | bacula.sql | gB B5+D IGg B A y A GXM2 BAA zI BLX9m2 BLX9m2 BLX9m2 A A C |
| /usr/local/bacula/var/bacula/working/ | bacula.sql | gB B5+D IGg B A y A GX2z BAA zI BLYI0y BLYI0y BLYI0y A A C |
+-------------------------------------+-----------+-----------------------------------------------------------------+
```

Przyglądając się bliżej otrzymanym ciągom znaków LStat można zauważyć, że każdy z nich składa się z szesnastu wartości oddzielonych spacjami.

```
gB B5+D IGg B A y A GX2z BAA zI BLYI0y BLYI0y BLYI0y A A C
```

Hash ten zakodowany jest przy użyciu kodowania bazującego na **base64**. Można powiedzieć, że jest to nieco niestandardowa wersja kodowania **base64**. Jej algorytm można znaleźć w źródłach Bacula, a dokładniej w pliku:

```
/{lokalizacja _ ze _ źródłami _ bacula}/src/lib/base64.c
```

Na podstawie tego algorytmu napisałem implementację **dekodera wartości LStat Bacula** w języku **PHP**. Dekoder ten dostępny jest w wersji online pod adresem:

http://www.bacula.pl/bacula-lstat-decoder/

Podstawowa wersja funkcji PHP do rozkodowania wartości **LStat** wygląda następująco:

```
function decode _ bacula _ lstat($lstat) {
     $base64   =   'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwx
yz0123456789+/';
     $lstat = trim($lstat);
     $lstat _ fields = explode(' ', $lstat);

     if(count($lstat _ fields) !== 16) {
          die('Błąd! Niepoprawna ilość pól wartości LStat. Proszę upewnić
się, że podany ciąg znaków jest poprawną wartością LStat');
     }

     list($dev, $inode, $mode, $nlink, $uid, $gid, $rdev, $size,
$blocksize, $blocks, $atime, $mtime, $ctime, $linkfi, $flags, $data) =
$lstat _ fields;
     $encoded _ values = array('dev' => $dev, 'inode' => $inode, 'mode'
=> $mode, 'nlink' => $nlink, 'uid' => $uid, 'gid' => $gid, 'rdev' =>
$rdev, 'size' => $size, 'blocksize' => $blocksize, 'blocks' => $blocks,
'atime' => $atime, 'mtime' => $mtime, 'ctime' => $ctime, 'linkfi' =>
$linkfi, 'flags' => $flags, 'data' => $data);

     $ret = array();
     foreach($encoded _ values as $key => $val) {
          $result = 0;
          $is _ minus = false;
```

```
            $start = 0;

            if(substr($val, 0, 1) === '-') {
                  $is_minus = true;
                  $start++;
            }

            for($i = $start; $i < strlen($val); $i++) {
                  $result = bcmul($result, bcpow(2,6));
                  $result +=  strpos($base64, substr($val, $i , 1));
            }
            $ret[$key] = ($is_minus === true) ? -$result : $result;
      }
      return $ret;
}
```

a jej użycie może mieć postać:

```
$lstat = 'gB B5+D IGg B A y A GX2z BAA zI BLYI0y BLYI0y BLYI0y A A C';
$decoded_lstat = decode_bacula_lstat($lstat);
print_r($decoded_lstat);
```

Znaczenie pól wartości LStat

Jak już wspomniałem, każda wartość rekordu LStat składa się z szesnastu pól oddzielonych znakiem spacji. Są to kolejno:

- **Device** - numer urządzenia w systemie dziesiętnym. Odpowiada temu komenda:

```
stat --format='%d' /mnt/sda5/plik.txt
```

- **Inode** - numer węzła (inode). Odpowiada temu komenda:

```
stat --format='%i' /mnt/sda5/plik.txt
```

- **Mode** - typ i uprawnienia pliku. Do odczytania tych wartości z pola **mode** można posłużyć się dowolną funkcją konwertującą wartość w systemie dziesiętnym na wartość w systemie ósemkowym. Dla **PHP** może to być np. (dla wartości 33184):

```
decoct(33184); # co daje wynik 100640 czyli -rw-r----
```

- **Nlink** - ilość twardych dowiązań. Odpowiada temu komenda:

```
stat --format='%h' /mnt/sda5/plik.txt
```

- **UID** - identyfikator użytkownika, który jest właścicielem pliku. Odpowiada temu komenda:

```
stat --format='%u' /mnt/sda5/plik.txt
```

- **GID** - identyfikator grupy, która jest właścicielem pliku. Odpowiada temu komenda:

```
stat --format='%g' /mnt/sda5/plik.txt
```

- **Rdev** - jeśli zapisanym plikiem jest plik specjalny urządzenia (np. przy backupie całej partycji poprzez specjalny plik urządzenia /dev/sda5) to wartość ta reprezentuje numery major i minor urządzenia. Do ”rozpakowania” tej wartości można posłużyć się np. językiem **Python** w następujący sposób (dla wartości 2049):

```
import os
os.major(2049) # zwróci 8
os.minor(2049) # zwróci 1
```

- **Size** - całkowity rozmiar w bajtach. Odpowiada temu komenda:

```
stat --format='%s' /mnt/sda5/plik.txt
```

- **Blocksize** - rozmiar bloku wejścia/wyjścia. Odpowiada temu komenda:

```
stat --format='%o' /mnt/sda5/plik.txt
```

- **Blocks** - liczba zaalokowanych bloków. Odpowiada temu komenda:

```
stat --format='%b' /mnt/sda5/plik.txt
```

- **Atime** - czas ostatniego dostępu podany jako UNIX’owy znacznik czasu. Odpowiada temu komenda:

```
stat --format='%X' /mnt/sda5/plik.txt
```

- **Mtime** - czas ostatniej modyfikacji pliku podany jako UNIX’owy znacznik czasu. Odpowiada temu komenda:

```
stat --format='%Y' /mnt/sda5/plik.txt
```

- **Ctime** - czas ostatniej zmiany pliku (np. zmiana atrybutów) podany jako UNIX’owy znacznik czasu. Odpowiada temu komenda:

```
stat --format='%Z' /mnt/sda5/plik.txt
```

- **LinkFI** - dodatkowa wartość wprowadzona przez twórców Bacula. Oznacza indeks pliku z danymi jeśli plikiem jest dowiązanie twarde. Indeks ten odnosi się do pliku, na który wskazuje dowiązanie twarde.
- **Flags** - flagi z systemu FreeBSD (taki opis widnieje w źródłach Bacula. Niestety nie wiem dokładnie o jakie flagi chodzi).
- **Data** - identyfikator strumienia danych (również nie wiem nic więcej na ten temat).

Podsumowanie

Rozkodowane wartości pola **LStat** dostarczają wiele informacji na temat zapisanych przez Bacula plików. Mogą one przydać się np. podczas diagnozowania jakichś problemów z plikami lub do własnego oskryptowania operacji wykonywanych przez Bacula.

Wątek z grupy dyskusyjnej **bacula-users**, który zainspirował mnie do rozpoznania wartości LStat i jej kodowania można znaleźć *tutaj*. Można tam znaleźć również implementację dekodera pola **LStat** w języku **SQL**.