tar to BRU Contrast

Introduction

`tar` (Tape ARchive) is a widely used backup utility that is included in most UNIX-based operating systems. This white paper compares the ability of the `tar` utility and TOLIS' BRU™ (Backup & Restore Utility) technology to reliably protect critical data that is valuable enough to back up.

This paper also presents a simple test that can be performed to prove the inherent inability of tar to reliably protect data.

BRU Was Developed Because of tar

In 1985, Fred Fish was using `tar` to protect his business data and discovered he was unable to successfully recover it. Burned by the loss of the irrecoverable information, he analyzed the backup and restore data flows and discovered the process was fraught with pitfalls. Mr. Fish wrote an elegant backup utility that implemented the necessary mechanisms to assure data could still be recovered when a problem arose. The BRU format and first BRU product was born.

From its inception to this day, the development of BRU has been a classic example of "form follows function." Mr. Fish's design focus centered on the ability to recover data, and the backup function simply became the form to support the recovery function. While this design philosophy is profoundly different from all other backup tools to deliver reliability, operationally BRU uses the familiar `tar` syntax.

Because of Mr. Fish's vision and resolve, countless users around the world rest easy knowing that their data is safely protected by BRU. The excellence of his code, continually updated and enhanced over the last 27 years, has stood the test of time.

After all, it's all about the restore, not the backup.

BRU and tar Compared

Feature	**BRU**	tar
1. Works with any device*	Y	N
2. Graphical Interface Support	Y	N
3. Comparison Verification	Y	Y
4. Checksum-based Verification	Y	N
5. Recover from Corrupted Media	Y	N
6. Backup Special Files (pipes, fifos, symlinks)	Y	N[1]
7. Backup Macintosh Finder Info and Resource Forks*	Y	N
8. Supports Raw Disk Backup and Restore	Y	N
9. Recognition of Media Errors During Backup	Y	N
10. File by File Data Compression	Y	N
11. Files over 2 GB Supported	Y	N
12. Tape Labeling	Y	N
13. Overwrite Protection	Y	N
14. Smart Restore on Live Systems	Y	N
15. Translate Files on Restore	Y	N
16. Restore Control for Overwrite of Existing Disk Files	Y	N
17. Full Operational Logging	Y	N
18. Automatically Fully Backward Compatibility	Y	N
19. Technical Support Ownership	Y	N

* Macintosh related

[1] Some versions of tar back up special files

BRU and tar Compared - Details

1. Works with any device – `tar` is not device aware. This is bad on the Macintosh since there are no device nodes for tape drives. This prevents `tar` from working with tape devices on a Mac OS X system. BRU is fully integrated with the Mac OS X SCSI manager and supports all forms of tape drives and libraries, as well as disk files and disk volumes.

2. Graphical interface support – The use of `tar` is limited to command line operation only. The populace's ability to operate via command line is fast waning, and therefore reduces the available number of people capable of managing tar backup and restore operations. BRU supports command line operation and graphical interfacing to provide access to BRU's robust features.

3. Comparison verification – Both BRU and `tar` support file-by-file comparison to verify the backup. File-by-file verification is only useful immediately following the backup and keeps the system's filesystems in use for twice as long as required when using checksum verify (see following).

4. <u>Checksum-based verification</u> – BRU's checksum-based verification provides a method to check the validity if an archive's contents at any time, even on a different system.

5. <u>Recover from corrupted media</u> – Any number of events can corrupt data on media following an accurate backup. `tar` will abort a restore completely if an archive error occurs during a restore, leaving any remaining data irretrievable. BRU has the logic and control capabilities to seek beyond the corrupt segment and resume the restore when the next file is located on the media.

6. <u>Backup special files</u> – `tar` will skip over fifos, pipes, or empty directories. This can result in incorrect functioning of many apps if a restore is required. BRU properly tracks, backs up and restores all file types.

7. <u>Backup Macintosh finder info and resource forks</u> – `tar` does not understand nor track the special elements of the Mac filesystems - finder info and resource forks will not be included in the backup. This renders applications and data useless after a restore.

8. <u>Supports raw disk backup and restore</u> – BRU will backup and restore raw disk volumes, enabling the backup of custom partitions and filesystems used by database platforms such as Sybase and Oracle. `tar` does not have the ability to process this data.

9. <u>Recognition of media errors during backup</u> – `tar` does not track device or media errors during backup (aside from a complete write failure), so high levels of rewrites or pending device failure can go completely undetected until a restore is required and the archive cannot be read.

10. <u>File by file data compression</u> – BRU's software compression method backs up each file as a separate entity within the resulting archive while tar compresses the entire stream via its connection to gzip. This results in a `tar` archive becoming unrecoverable if a single bit is corrupted early in the backup. A corrupted compressed file in BRU does not affect other files within the archive.

11. <u>Files over 2 GB supported</u> – BRU is fully 64-bit aware, so files larger than 2GB are not a problem. Depending on the version of `tar` used, the file size limit is 2GB minus 1 byte. Should a newer version of tar be used to write an archive containing 64-bit files, older versions of `tar` will fail when attempting to restore the data.

12. <u>Tape labeling</u> – BRU maintains a human-defined and readable label for all archives, thereby simplifying tape access. Also, each BRU archive has a unique archive ID making it easier to track tapes using something besides the human generated label. `tar` offers no mechanism for either method of media identification.

13. <u>Overwrite protection</u> – BRU checks media before overwriting the contents (unless you explicitly select overwrite as an option). `tar` will blindly overwrite existing backups with no warning.

14. <u>Smart restore on live systems</u> – `tar` will gladly overwrite in-use files during a restore (such as libc) with no concern for the result of that restore. BRU will properly handle in-use files and restore without crashing your system.

15. <u>Translate files on restore</u> – BRU will allow you to define a translation table and automatically relocate files during a restore.

16. Restore control for overwrite of existing disk files – BRU allows you to choose whether a restore will replace existing files on your disks. You may choose to never replace, replace only if the file on tape is newer than the file on disk, and many other variations.

17. Full operational logging – All BRU operations are reported in the /var/log/bruexeclog file. This allows an administrator to track all BRU operations, regardless of who runs BRU. `tar` offers no reporting.

18. Automatically fully backward compatibility – All versions of BRU automatically support the reading, verification, and recovery of older BRU formatted tapes. Some versions of `tar` can create archives that can only be recovered by that particular version of `tar`.

19. Technical support ownership – When you have a problem with `tar` and your `tar` archives: "Who are you going to call?"

Disproving tar's Reliability - a Simple Test

BRU implements multiple techniques during the backup and restore operations to assure that the data returned is accurate.

The test that follows, using a tar format example, is but one example that verifies the robustness of the BRU technology and the vulnerability of `tar`. Prove it yourself!

```
[root@pc-00067 /root]# tar -cvf test.tar /etc/hosts*
tar: Removing leading `/' from member names
etc/hosts
etc/hosts.allow
etc/hosts.deny
[root@pc-00067 /root]# bru -cvf test.bru /etc/hosts*
/
/etc
/etc/hosts
/etc/hosts.allow
/etc/hosts.deny
```

Here is the /etc/hosts file before we tar'd or BRU'd it.

```
--snip--
127.0.0.1               localhost.localdomain   localhost
192.168.1.75            accounting
192.168.1.1             gw
209.149.147.185         serve1
66.1.56.100             bleaf
66.1.47.197             fwc
--snip--

[root@pc-00067 /root]#

[root@pc-00067 /root]# vi test.tar
```

Skip past the `tar` header and modify one byte or character i.e. change one of the IP's in the hosts file. We changed 209.149.147.185 to 209.349.147.185 in the 'serve1' entry.

```
[root@pc-00067 /root]# vi test.bru
```

Skip past the BRU header and modify one byte or character ie. change one of the IP's in the hosts file, we changed 209.149.147.185 to 209.349.147.185 in the 'serve1' entry.

PLEASE NOTE: we made the same change to both the test.tar and the test.bru, changing the 149 to a 349 in the '209.149.147.185 'serve1' entry.

<u>Now let's try to extract the files from the tar archive first.</u>

```
[root@pc-00067 /root]# tar -xvf test.tar
etc/hosts
etc/hosts.allow
etc/hosts.deny
```

<u>It extracted normally, now let's look at the file after it's been modified.</u>

```
--snip--
127.0.0.1                 localhost.localdomain localhost
192.168.1.75              accounting
192.168.1.1               gw
209.349.147.185           serve1
66.1.56.100               bleaf
66.1.47.197               fwc
--snip--
```

The 'tar' utility returned inaccurate data without notification.

<u>Now let's try to extract it with BRU and see what the result is.</u>

```
[root@pc-00067 /root]# bru -xvvvvvf test.bru
archive ID = 3c9a03a856e2
buffer size = 20k bytes
media size =
bru: [I245] "/": skipped file, directory exists
bru: [I245] "/etc": skipped file, directory exists
bru: [W011] warning - file synchronization error; attempting recovery ...
x   8K [1] -rw-r--r--  1 root      root      161 Jan 12 2000 292040 /etc/hosts.allow
x  10K [1] -rw-r--r--  1 root      root      347 Jan 12 2000 292041 /etc/hosts.deny
bru: [I181] read 10 blocks (20 KBytes) on volume [1], 0:00:00, 9999 Kb/sec

    **** bru: execution summary ****

    Started:            Thu Mar 21 09:11:23 2002
    Completed:          Thu Mar 21 09:11:23 2002
    Archive id:         3c9a03a856e2
    Messages:           1 warnings,  0 errors
    Archive I/O:        0 blocks (0Kb) written
    Archive I/O:        10 blocks (20Kb) read
    Files written:      0 files (0 regular, 0 other)
    Files read:         2 files (2 regular, 0 other)
    Files skipped:      2 files
    Write errors:       0 soft,  0 hard
    Read errors:        0 soft,  0 hard
    Checksum errors:    1

    [root@pc-00067 /root]#
```

BRU recognizes the data has been corrupted.

Conclusion

If it were not for the inherent vulnerabilities of the `tar` format, BRU would never have been developed.

The structure of traditional `tar`, and cpio archive formats, do not support the integrity of data during the backup or restore operation. In contrast, the BRU format accounts for every bit being backed-up and restored—reflecting backup science at its best. The protection of data, whether in the form of traditional text/number information or works of art, is too important to trust to a flawed approach.

In the objective example above, `tar` fails the reliability test while BRU recognizes and reports the corrupted data. During actual operation, BRU would then advance the tape until the next good BRU header was read and the restore would continue. `tar`, and other tools will abort the restore, thereby rendering the remaining data that has not already been restored irretrievable.

Most people embrace the concept of insurance (auto, home, life) to provide an acceptable level of protection. The concept should also apply to your data. BRU insures your data is safely protected - `tar` cannot.