Using Tcl VFS with Encryption

Phil Brooks

October 20, 2015

Mentor Graphics®

Agenda

- Tcl VFS Overview
- Tcl Encryption Overview
- Mentor Graphics Tcl VFS based decryption system

Tcl VFS Overview

- Provides a low level C api through which all regular Tcl file mechanisms (things like **open, read,** and **seek** as well as higher level commands like **source** and **glob**) are routed.
- This abstraction allows a simple point of access that allows you to make up your own mechanism for supplying the data that any Tcl script can simply view as a bunch of regular file accesses.

Tcl_FS* Functions

- Tcl_FS… Tcl C API functions
  - allow installation, management and destruction of VFS filesystem implementations
  - Provide safe 'C' level access to VFS filesystem
    - For example, using the stat() function call directly from 'C' will not reflect any VFS filesystems installed, while use of TclFSStat will.
  - The Tcl_FS* functions also provide consistent behavior across platforms (to the extent possible).

Tcl_FS... Overview

- ***File Access Functions:***

  Direct access and manipulation of a file and its contents
  - Tcl_FSOpenFileChannel
  - Tcl_FSStat
  - Tcl_FSAccess

Tcl_FS... Overview

- ***Directory Manipulation***

  Directories,their contents, symbolic links, etc. can be accessed and manipulated
  - Tcl_FSMatchInDirectory
  - Tcl_FSGetCwd
  - Tcl_FSChdir
  - Tcl_FSLink
  - Tcl_FSCreateDirectory
  - Tcl_FSUtime.

Tcl_FS... Overview

- ***File Path manipulation and construction***

  Platform independent construction and manipulation of file paths
  - Tcl_FSPathSeparator
  - Tcl_FSNormalizePath
  - Tcl_FSJoinPath
  - Tcl_FSSplitPath

Tcl_FS... Overview

- ***VFS construction, manipulation and maintenance***

  Direct access and manipulation of a file and its contents
  - Tcl_FSRegister
  - TclFSUnregister
  - Tcl_FSData
  - Tcl_FSMountsChanged

struct Tcl_Filesystem

- Contains pointers to functions that implement underlying VFS filesystem operations for a particular VFS implementation
- Many of them correspond directly to Tcl_FS* functions above.
- Tcl_FSPathInFilesystemProc - is the key mechanism by which a VFS filesystem identifies files that belong to it.

  Tcl will only call the rest of the filesystem functions with a path for which this function has returned **TCL_OK**. (Not quite true)

struct Tcl_Filesystem

- ***File Access:***
  - Tcl_FSOpenFileChannelProc, TclFSStatProc, and Tcl_FSAccessProc
  - Tcl_FSOpenFileChannelProc is important to us in that the Tcl channel interface is used to return decrypted data to the end user.
- ***Directory Manipulation:***
  - TclFSMatchInDirectoryProc, Tcl_FSGetCwdProc, Tcl_FSChdirProc, Tcl_FSLinkProc, Tcl_FSCreateDirectoryProc, and Tcl_FSUtimeProc.
- ***File Path manipulation and construction***
  - Tcl_FSPathSeparatorProc, Tcl_FSNormalizePathProc.

::vfs package

- Provided with ActiveTcl
- It consists of three parts:
- vfs – the vfs::filesystem commands – allow mount, unmounts and manipulation of vfs filesystems
- vfs-filesystems: Several useful filesystems: zip, mk4, tar, ftp, ns, webdav, http, urltype
- vfs-fsapi: the interface for creating a filesystem. This allows you to write a command ensemble that supports access to a vfs filesystem

Encryption Overview

- Many encryption packages are available for Tcl. The following are available with Tcllib:
- blowfish
- aes
- des (including triple des)
- rc4
- Cryptkit

Encryption Overview

- Limits of Encryption
  - DVD Problem: If you have a well encrypted file and a machine that can decrypt the file and a person that has enough time and inclination that has access to both, then the person can learn how to decrypt that file and ones similarly encrypted.

Calvfs Encryption

Features and Limitations

- Usage scenario: A supplier creates an archive, ships it to end users who then have runtime application only access to the archive
  - Write once, read many delivery
  - no symbolic links
- workable simple read-only filesystem can be implemented with only the following interfaces:
  - pathInFilesystemProc – identifies files that are part of the VFS
  - statProc – provides information similar to stat()
  - accessProc - provides information similar to access()
  - openFileChannelProc – implements the read channel for files in the filesystem

Mounts?

- The vfs:: package requires an explicit mount
- **Calvfs Automount**
  - Calvfs uses Tcl_FSPathInFilesystemProc - The files are identified by examining the filepath and looking for Calvfs archives along the path.
  - Once a .calzip archive is identified, it is opened by the CalVFS filesystem implementation and its contents are transparently accessed.

Mounts?

- **Nesting**
  - CalVFS archives can also be nested inside of one another.
  - ACLs and permissions on contained archives can be different from one another.
  - The Tcl_interp* argument can be used to restrict encrypted code to execute in a particular interpreter
  - archive demo2.calzip which contains the file dir3/dir4/file is contained inside dir1 of demo1.calzip
  - a regular file foo/bar/file that is outside of any archive

Mounts?

- What to do with paths that pass through multiple filesystems?
- dir1/dir2/foo.zip/bar/../../foo2.calvfs/dat1
- Unsupported for now

Miscellaneous

- **Write/Encode Process**
  - simple serial archival process similar to Zip or Tar
  - Encrypted as they are written into the archive
  - Nested .calzip files are simply copied into a containing archive
- **Read/Decode Process**
  - Files are decoded in 1064 byte chunks
  - Decode state is maintained in conjunction with the channel that is currently reading from them.
  - Offsets and intermediate decryption information are also maintained inside of the channel object.
- **Seek**
  - Seek functionality is limited
  - Seeking can have performance implications
    - decryption state must start from a known point.

Miscellaneous

- **Access Control**
  - Protection between the encrypted .calzip contents in one archive and generally running Tcl in the application
  - Access Control Lists to vary what type of access it will allow to which interpreters in the system.
  - An encrypted .calzip file can get its own dedicated interpreter that has access to the .calzip contents while other interpreters in the same running process are denied access to the same .calzip file

Mentor Graphics®

www.mentor.com