VernamFS User Guide

Stuart Maclean*

November 2015

1 Introduction

This guide explains the Vernam File System, or 'VernamFS', a file system whose backing store is a 'one-time-pad' (OTP). It is named after Gilbert Vernam, one of the first to realize the significance of the one-time-pad.

A one-time-pad is simply a sequence of random numbers, in our case a sequence of randomly-valued bytes. In classic OTP usage, a plaintext message M is XOR'ed with a sub-sequence of the OTP to produce an enciphered version C. Without access to the OTP, it is impossible for an adversary to derive M given access only to C. It is impossible in the sense that all possible Ms are equally likely; having C tells us nothing about M. No amount of computing resources can 'crack' this problem.

We apply the properties of a OTP to a file system setting. The VermanFS uses a OTP to keep stored data secure (enciphered), so that only with access to a copy of the original OTP can the stored data be deciphered. The VermanFS is ideally suited to remote deployments, where some 'sensor' (buoy,glider,etc) is to collect and store data, and that data should not be readable by an adversary were they to gain access to the sensor.

The system works thus:

1. Create a one-time-pad file of some desired length, perhaps that of an entire device D (SD card, flash drive, SSD, etc) intended to hold data collected by the sensor. Such a OTP might be 1 or 4GB large, but for testing can be as small as a few kilobytes.

2. Write the OTP to the device D, either onto the whole device, or as a file on the device. Either works.

3. Store the original OTP in a safe place, i.e. in a vault. We call this the 'vault unit' or 'vault copy'. It is crucial that no data is ever written to the vault copy other than the initial OTP.

4. Install the device D into the sensor and deploy it into the field. This is then said to be the 'remote unit'. We shall then create a "VermanFS" on D.

5. In the field, the sensor writes data to D, using regular filesystem operations: open, write, close. This data, via our VermanFS implementation, is completely unreadable by anyone, including the VermanFS code itself! The persistent storage D *is* the OTP. For each byte B of data on D, a user data byte M is XOR'ed with B to produce C and that result written back to the store, overwriting B. Each byte of D can only be used once.

6. At deployment end, recover D, the remote unit. Combined with the original vault copy, the original plaintext data can be recovered.

As a series of Unix commands, the above usage pattern becomes:

*Applied Physics Laboratory, University of Washington

```
// Create the OTP, 1 MB for testing
$ dd if=/dev/random of=OTP count=2048

// Make remote and vault copies
$ cp OTP OTP.V
$ cp OTP OTP.R
$ rm OTP

// Secure the vault copy
$ chmod a-w OTP.V

// Deploy the remote unit
$ scp OTP.R remoteHost:

// At the remote site, store data onto the now-local OTP
// See later for how this is actually done
remoteHost$ vfsWrite OTP.R

// Recover the remote unit
$ scp remoteHost:OTP.R .

// Combine the two OTPs to recover the data
// See later for how this is actually done
$ vfsCombine OTP.R OTP.V
```

2 Implementation

Our Vernam Filesystem implementation makes use of FUSE - the filesystem in user space. Using FUSE makes it very easy for the remote site to use the OTP storage, without really knowing it is even interacting with a OTP.

3 Remote Data Recovery During Deployment

Sometimes we wish to recover data from the remote unit during deployment, i.e. before the remote unit is physically recovered. For example, we might be able to upload parts/all of the remote unit over Iridium satellite comms.

4 Advantages

- Small compute power at remote site, just XOR
- No need for any random number generation on remote unit. Good RNGs are hard to find in the field, expensive.