Accountabilibuddy Passwords System

Philip Robinson

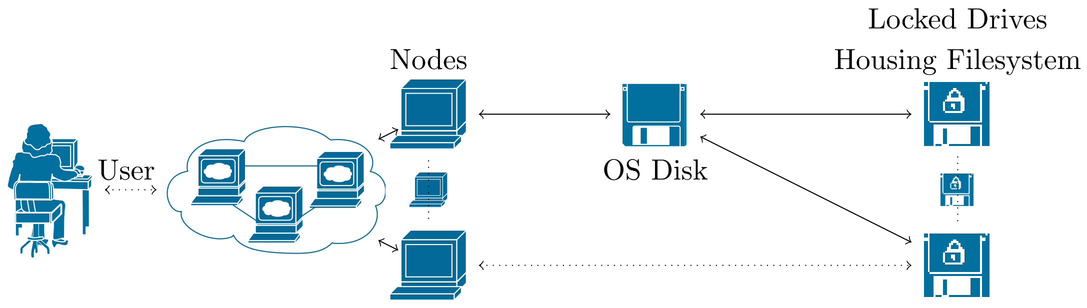
May 13, 2015

Abstract

The purpose of this paper is to clearly explain the design, advantages, and restrictions to the Accountabilibuddy Password System. Note that all new terms perhaps found in this abstract will be defined in the paper as needed. Accountabilibuddy Password System acts as a networked password manager that, instead of having a single remote point of failure, has a threshold failure based on the consistency and up time of the participating computers sharing a network. This is accomplished by the use of Shared Secrets Cryptography to split and distribute sensitive information across, what we will call, stable network vectors.

Introduction

To make this problem more accessible I am going to first address the system architecture that inspired this idea. I was working with a security sensitive team whose goal was to provide a password manager used in unlocking self encrypting drives on a a distributed file system (FreeBSD derivative).

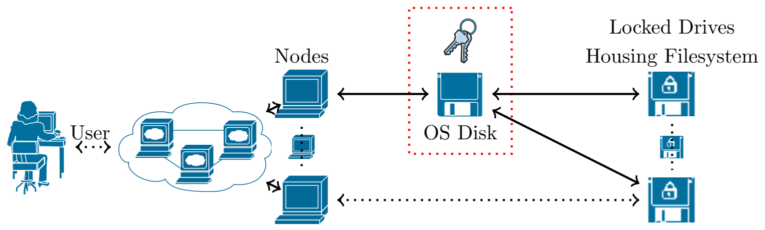


Every node in the system owned an unencrypted 'Operating System Disk' that was tasked with locking and unlocking self encrypting drives where the file-system was stored. We were to provide an environment where, if a person walked in and stole a harddrive at random, they would be unable to retreave any data from that drive.

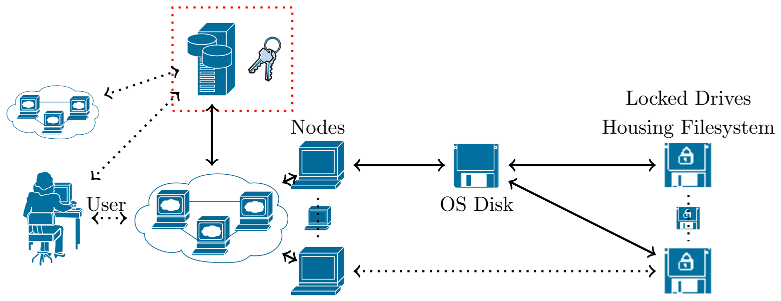
Self Encrypting Drives and Password Managers

Usually, a password manager is used to administrate passwords when it is unreasonable to have a human keep track and enter the passwords used by a system. In the case of self encrypting drives, data is decrypted and encrypted on disk read and write at the hardware level, respectively. In order to accomplish this, a password handshake must happen every time a self encrypting drive is mounted to the operating system. When a self encrypting drive is unmounted or, it's power is lost, the drive is re-locked; resulting in the inability to read or write useful data from the disk.

The password manager acts like a database that can provide the passwords as needed to unlock the drives. The use case that many people encounter is a local password manager. This is often used on personal computers when your internet browser automatically fills your password field on login to a website. In the case of our distributed system, this would mean that the passwords stored at the 'OS Disk' level and would only need to house the passwords for that node's self encrypting drives.



In contrast a networked password manager provides a separate appliance to support a great diversity of password types/clients, and in much greater volume. networked password managers are much more common as industry solutions. In our example, the networked password manager would house and administrate the self encrypting drives' passwords, and possibly passwords for other use cases as well.



Simple Threat Modeling for Password Managers

Using a password manager brings in discussions of how we choose to define security. To claim a system is completely secure, is not a realistic statement. When talking about the security of a system, we also are required to define our threat model. This means that we are choosing what the possible attacks are based on our system's needs and environment. In this, we will talk about vulnerabilities when in data transit and in physical theft. There are many other concerns when writing a formal threat model, but these will be sufficient for this discussion.

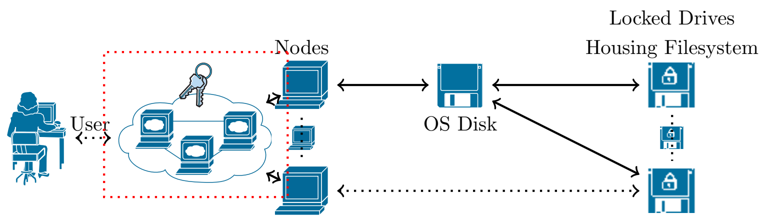
As we can see with the local password manager: if a single self encrypting drive is removed from any node, then that drive's data remains uncompromised. Secondly, there is no point where the passwords are in transit over a network connection; so the ability for someone to steal the passwords while in transit is very unlikely. However, if the 'OS Disk' and some of it's managed 'Drives' were to be removed from our system, then we would have given up the keys and their corresponding data for that node.

We can increase our security in local password managers by, instead of storing passwords on the 'OS Disk,' introducing the use of stable system vectors. This is a term used to describe the sampling of (non-harddrive) hardware, with unique and consistant profiles, within a node that cannot change (even on reboot and physical maintenance) to construct consistent unique passwords. This would then require the whole node to be stolen to compromise our data (rather than just the drives). This strategy is only really helpful if it takes significantly more time to steal a node than a series of harddrives and may be brittle to replacing hardware in use.

In the networked password manager our cluster's data isn't compromised by physical theft of nodes or any drives in the node; unless the password manager appliance is compromised as well. For this reason it is often the case that a networked password manager is stored in a physically different location. This does mean that if we assume that the networked password manager is stored securely. This model often increases the number of devices networked to your cluster, making it a higher value target.

Thousand Mile Accountabilibuddy Password System

The goal of Shared Secrets Cryptography is to take a secret '**message**' and split the contents among a series of participants, such that no individual participant can compromise the security of the system; or more correctly, that given N participants in total, it would require at least T participants ($T < N$) to recreate the original '**message**'. The maths & technical details of this process will be discussed in a later section.



We choose this '**message**' to act as the password that is split among the N participating nodes in a system, under the assumption that it is difficult for an adversary to gain control of T nodes. What is accomplished with this design is virtualizing the stable system vectors (as exampled in the local password manager) into what we call stable network vectors, and distributing them across the nodes. This improves to our local password managers threat model such that an adversary would need to compromise a substantial subset of the participating nodes in order to recover any data.

In practice we can actually use this '**message**' to act as the key for decrypting the storage unit for a local password manager. This design decision can significantly reduce our network traffic for a distributed file system, as we will only need to unlock our keystore, not each drive indivisually over the network.

Implementation