Method for Detection of Tampering with System Password Files

Virtually every operating system utilizes several methods to prevent unauthorized access to the system. While these methods differ in implementation and effectiveness, virtually every publicly utilized computer is secured by a password that restricts access to specific user accounts on that machine.

For password authentication to occur locally on a machine, the password must be stored in some form on the computer itself. Typically, this means the password is stored on a hard drive in the form of a hash that is designed to be compared, not reversed. While this deterrent used to be sufficient, recent advances in technology have made this form of security inefficient. The password can be bypassed by either discovering the password by obtaining the hashes, and comparing them to the hash of every possible password (brute forcing), or by replacing the hash altogether with the pre-calculated hash of a known password. Both of these methods can easily be implemented if an attacker has physical access to a machine, and is able to boot the computer from removable media. Thus, reliance solely on a password hash stored in a password file is not sufficient for securing access to a system that may be stolen or subject to unauthorized access.

While there are some methods that can ensure the security of a system, such as encrypting the entire hard drive, or encrypting portions of the drive, these methods can be cumbersome, as they increase load on the CPU to perform constant decryption and encryption, and can in many cases reduce hard drive speed. Advances in hashing protocols has led to the development of stronger hashes that take much longer to crack, and can be salted to prevent the use of hash comparison lookups against pre-computed tables of hashes (rainbow tables). Unfortunately, once an attacker discovers the salt, which must be stored in some form on the machine, the attacker can then generate a hash with the proper salt for a known password, which will allow him full access to the system.

Detailed Description of Invention:

Although password files are a necessity for securing computer systems, a password file alone is vulnerable to manipulation by an attacker. This invention detects tampering with the password file by signing the password file utilizing a public/private key system. This adds an additional layer of security that ensures that password hashes cannot simply be replaced when the system is offline. After booting, when a user attempts to logon, the system verifies the authenticity of the password file, utilizing a public key that is optimally stored in a manner that it cannot be changed, e.g. within a TPM, a smartcard, or stored on an external machine that is accessible via the internet. While utilizing a public key stored on a hard drive would make the system more secure by requiring the attacker to take the additional step of generating a new key pair to sign their modified password file, this is a somewhat trivial process. Once the authenticity of the password file has been checked, the user's password is then hashed, and compared to the hash stored in the password file. If the hashes match, the user can then logon.

If the password file does not pass verification, the user is notified via visual, auditory, or other warnings. This ensures that the user is aware that an attacker has had access to their computer, and may or may not have tampered with the files system. This also ensures that passwords are changed and created securely via an external machine or device that contains the private key. This can help prevent

unauthorized modification of passwords in environments where the system administrator has issued passwords that are intended to remain the same.

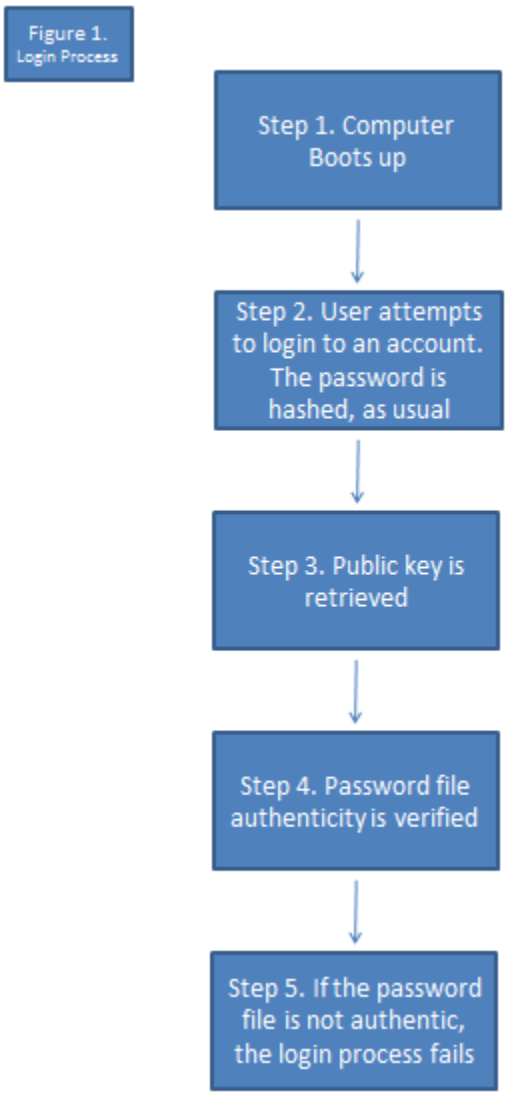

Figure 1 demonstrates the login process. In Step 1, the computer loads the operating system. In step 2, the user enters a password that is hashed to verify against the stored password. In step 3, the public key is retrieved from a location in which it is immutable. This is done to prevent an attacker from replacing the public key as well. The key must be stored in a read-only manner, e.g. on a TPM, or as part of a static hardware id. In step 4, the public key is used to verify the authenticity of the password file. If the password file has been modified, the password file will fail verification, the user will be notified that the password file has been tampered with, and the system will refuse to log the user on.

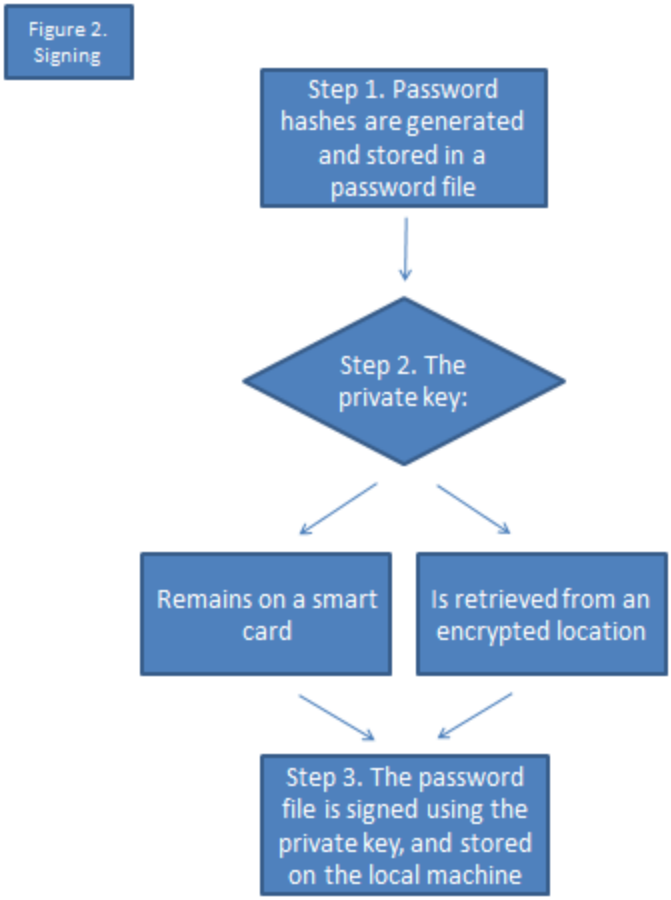

Figure 2 demonstrates the signing process. In step 1, the password file is created by the operating system. Password hashes are stored together in a file. In step 2, the private key is loaded from one of two environments. Ideally, the private key could remain on a smart card that must be inserted to change passwords. Since signing will physically occur on the smart card, the private key never leaves the smart card, and remains secure. The private key can also be retrieved from a location where it has remained encrypted. This would require a master password that would be used to encrypt the private key. This option is less optimal, as after decryption, the private key has entered an unsecure environment. In step 3, the actual signing occurs, and the password file is then stored on the local machine.