Executive Summary

User privacy is becoming increasingly important for applications, especially in open execution environments \(^1\). Keeping user data secure locally on the user’s machine so that only the user can access it is a strong value proposition for an application vendor. Unfortunately, there is a dearth of tools for accomplishing this task effectively. This document describes the challenges faced by any application attempting secure local data storage and how Intel technology can help address them. For the purposes of illustration, this document will approach the problems from the perspective of a password manager operating on the Windows desktop. The mechanisms described herein can be generalized to other applications.

“Hardware empowered Software Guard Extension enclaves provide the protection needed to fix the potential vulnerabilities for user data stored on the local machine.”

Paul Carlson, Software and Services Group, Intel

Privacy, User data, and Malware

Hide and Seek Security

In order to preserve user privacy, data must be protected when in use and when in storage. For storage, applications will typically use a key to encrypt the user data and keep it on the local disk \(^2\). Therefore, protecting that key is tantamount to protecting the data. Some applications achieve key protection by simply hiding the key (e.g. the password stores in most web browsers). However, this can be easily compromised by malware. Applications such as password managers that wish to take things a step further generally operate on the ‘master password’ principle. That is, they encrypt all the passwords with a key derived from a master password and ask the user for the master password when they wish to open the encrypted database.

Passwords vs the World

While the master password approach sounds fine in theory, there are practical issues which can undermine its security in the real world. Users of password managers want more than just the ability to keep a list of encrypted passwords; they also want convenience. Master password based password managers improve usability by not requiring the user to enter the master password each time the user requests one of their passwords. To make this possible, those managers must either (1) keep the database encryption key alive (in memory) or (2) keep a decrypted copy of the database alive. In either case, the situation is downgraded back to security by obscurity.
Memory is an Open Book

Unfortunately, this is not the only way the data or database key can be stolen. No matter what method is used to hide the key, it must be unmasked in order to be used - leaving both the key and the data being decrypted vulnerable to theft. In an open execution environment, malicious applications can force a password manager to ‘break’ at the exact point the data is exposed by attaching a debugger. Finding the exact point to ‘break’ may be difficult but hackers have unlimited access to their local copy of the password manager application. Thus, there exist multiple attack surfaces for malware to attempt to obtain a user’s data from a local application.

Biometric Authentication

Although the master password approach is common today, biometrics are a promising addition for user authentication in the future. Unlike passwords, biometrics cannot be lost, forgotten, poorly “chosen”, or easily brute-forced. However, there is an important aspect to consider about the use of biometrics for password management. With master password, the encryption key could be removed from the system when a user session is not open. With biometrics, that key needs to exist somewhere at all times so that the database can be decrypted when a biometric provider authenticates the user. The reason for this is that a biometric match cannot be used to deterministically create a secret such as an encryption key. Without this capability, the encryption key must pre-exist on the system in order for local secure data storage to work. Thus, use of biometrics requires a solution which can successfully protect the encryption key (and possibly the biometric match itself).

Requirements for a Solution

There is an important assumption about open execution environments in the above reasoning. Namely, that there is no truly effective means of protecting the password manager’s bits and bytes from malware. Although performing some attacks may require elevated privileges, phishing software or users that are unaware of the risks are likely to grant malware the required elevated privileges via a permission request dialogue. The security must be stronger than a simple privilege elevation dialog with unclear consequences from user’s perspective.

What would a better solution require? Application authors need to be able to specify what parts of their codebase need to be protected and how those protected sections communicate with the rest of the codebase. Next, those specifications need to be enforced by a trusted entity. Finally, the data on disk representing the protected sections and interfaces need to be protected against alteration. All of these security capabilities are provided by Intel’s Software Guard Extensions.

Intel® Software Guard Extensions

Intel® Software Guard Extensions (SGX) is Intel’s Trusted Execution Engine. SGX consists of a set of CPU instructions and platform enhancements that enable applications to create private trusted environments called enclaves. Within these enclaves, code and associated data can be protected from compromise due to attacks from privileged software and many hardware based attacks. When the code is executed from within an enclave, that code and data are better protected from disclosure or modification.

How can Intel SGX be used to improve application security? An application that wants to use SGX to improve its security needs to be re-factoried into trusted and untrusted components. Those trusted parts of an application can then be run inside one or more SGX enclaves. Because all memory associated with an enclave is encrypted at the hardware level, code executing within an enclave is opaque to other software, regardless of privilege level. Likewise, data associated with code executing within an enclave is also opaque to other software (including the “untrusted” part(s) of the same application that set up the enclave(s)) regardless of whether it is sealed on disk or kept in trusted memory. Thus, all secrets protected by an enclave are sealed and protected from tampering or theft from outside the enclave both during execution and at rest.

Password Management Protection with SGX

Hardware empowered SGX enclaves provide the protection needed to fix the potential vulnerabilities for user data stored on the local machine. First, SGX hardware sealing allows secrets or keys to be encrypted such that only the SGX enclave which sealed it can open it again. This allows for safe persistence of encryption keys or other data between sessions. Second, SGX enclaves provide a protected memory space to safeguard a cryptographic key (whether pre-existing or derived from a master password). By sending the master password (or other key) down into an enclave and keeping it there, the key can both be kept alive for user convenience and protected against theft by a means much stronger than obfuscation.[3]

Case in Point

True Key, Intel Security’s no password application launched in 2015 and leverages Intel Software Guard Extensions (SGX) to better protect against sophisticated attacks on its data. This technology and methodology can be used by other ISVs with password managers to provide added security for their end-users.
Biometrics with SGX

Beyond protecting current password-based models, SGX also enables the secure addition of biometric authentication. Different SGX enclaves can securely communicate with each other over encrypted channels. Thus, a biometric match algorithm could run in one enclave and pass the result to a second enclave which uses a sealed encryption key to unlock the user's private data. In this way, biometric authentication could be used to safely unlock an encrypted database without risking the match itself being tampered with, the match result being forged, or the encryption key being stolen.

1 An open execution environment is one where it is expected that the end users are at liberty to introduce any new application or system software (e.g. driver) into their system and potentially impact the security posture of the execution environment. On such systems, applications typically have access to each other’s memory and files such that a malicious program could potentially steal secrets from another application.

2 Some applications may use a server to help protect the key but that violates the “local” clause of local secure storage and will not work offline.

3 Malware could also attack the UI which gathers the master password from the user or the endpoint application the password manager injects the password into. Both cases are issues of third party application security and are out of the scope of this article. For a completely protected ecosystem, those elements would need to be protected as well. This does not undermine the security improvements suggested above.

This paper describes how ISVs can use Intel SGX to better protect password managers’ password databases. If you are interested in protecting your application’s data with Intel® SGX please contact the Intel Developer Relations Division at https://software.intel.com/en-us/sgx-sdk-support.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

Intel, the Intel logo and Intel SGX are trademarks of Intel Corporation in the U.S. and/or other countries.

* Other names and brands may be claimed as the property of others.

© Intel Corporation