Executive Summary

Consumers continue to increase their reliance on web browsers to conduct their day-to-day activities and business transactions. In those transactions, browsers store and process sensitive user data such as account information and authentication data. Such sensitive data can be susceptible to malware snooping that exploits software vulnerabilities, often without the users’ knowledge. In this paper, we show how commercial web browsers can be enhanced with Intel® SGX to protect sensitive data from malware threats.

“Browsers can be enhanced with Intel® SGX to protect sensitive data from malware threats.”

Srikanth Varadarajan, Security and Privacy Research Lab, Intel

Privacy, User data, and Malware

Authentication Tokens

Internet services such as search engines, social networks, online banking, etc. have security mechanisms in place to protect users’ data such as photos, payment information, bank account balances, etc. in cloud storage and in transit. Users typically authenticate to such services using username and password before they are granted access to the associated resources. Hyper-Text Transfer Protocol (HTTP) is used for web communications between a web server and a client browser, which is a stateless protocol. A secure version of HTTP called HTTPS provides security of the communication channel by encrypting the connection using Transport Layer Security (TLS) or Secure Sockets Layer (SSL) protocol. Before browsers used authentication tokens, users would login with their credentials and the web server would store that state in its memory. This causes scalability problems when many users log in to the same service, and when users’ sessions were migrated between multiple servers for load balancing. Instead, authentication tokens are generated by servers, when users login, and are sent to users’ devices where the browser stores them in cookies.
These tokens are presented by the client browser to the web server in subsequent HTTP transactions, where the token is validated before providing access to the resources. Since the token serves as a proxy, users do not have to enter their login credentials every time they use the service, which improves their user experience.

**Export Attacks**

An export attack is one where the authentication token is exported from a browser’s local storage in a client machine to an attacker’s machine. Since the token represents a successfully logged-in user, the export attack provides the attacker with access to the users’ resources in the web server. As the token is stored as a cookie in the browser’s cookie database, it can be straightforward for malware to steal the cookie, containing the token, from the database and transport it to an attacker’s machine.

**IETF Token Binding Protocol**

The Token Binding [1] protocol specifies a method to cryptographically bind an authentication token to a TLS session between a web server and a client browser. These bindings can be long-lived and can span multiple TLS sessions. If both the client browser and web server support the Token Binding protocol, the browser generates an asymmetric key pair (for each web server). On every TLS connection with that web server, the client proves possession of the private key by signing the ‘tls_unique’ value [RFC 5929] specific to the TLS connection using its private key. The server then verifies the signature with the client browser’s public key. The client browser’s public key serves as token binding identifier.

In subsequent sessions, when the client presents the token to the server, after the server verifies that the client possesses the right private key, it matches the token binding identifier in the token with the value previously established with that client, preventing export and replay attacks.

**Solution Requirements**

The premise of security offered by the Token Binding protocol relies on the fact that the private key is known only to the user agent, the client browser. The generation of the asymmetric keys (which includes public and private keys) and use of private key for signing the TLS unique value must be done in a trusted environment, denying access to private key operations by malware in the system. Furthermore, the performance impact of the key generation and signing must be negligible and not be impactful on the user’s experience, especially when restoring a large number of previously saved sessions during a browser state restore operation.
Intel® Software Guard Extensions

Intel® Software Guard Extensions (SGX) is Intel’s Application Trusted Execution Environment [2]. Intel® 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 attacks from privileged software and a number of hardware based attacks. When the code is executed from within an enclave, that code and data are better protected from disclosure or modification.

In order for an application to use SGX to improve its security, it needs to be re-factored into trusted and untrusted components. Those trusted parts of an application can then be run inside one or more SGX enclaves. All memory associated with an enclave is encrypted at the hardware level within the CPU package. Additional memory access control checks enable code executing within an enclave to be opaque to other software, including other enclaves, 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 the data is sealed using an enclave specific key and later stored on disk or kept in trusted memory. Thus, secrets protected by an enclave can be protected from tampering or theft from outside the enclave both during execution and at rest.

Hardened Private Keys with SGX

To support the Token Binding protocol the Client browser partitions its code base into trusted and untrusted portions. The trusted portion of the browser generates asymmetric key pair for each web server and securely signs the ‘tls_unique’ value using the private key. This trusted code runs within a SGX enclave and the private keys are in encrypted form in main memory. Encryption of enclave code and data is performed using a memory encryption engine [3] within the CPU package. The enclave code and data are resilient to bus and memory snooping attacks. Note that the enclave writer is responsible for taking care of software side channels that might be present in their implementation. The enclave operations are executed by the host CPU with minimal performance overhead, which eliminates the need for a co-processor. The untrusted portion contains the actual browser implementation and relies on the SGX enclave for the private key operations.

Chromium Source Patch in GitHub

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

This document contains information on products, services and/or processes in development. 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 https://software.intel.com/en-us/sgx.

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