@quasar/tauri - v1.0.0-alpha.0

https://github.com/quasarframework/tauri

TAURI

Create tiny, fast and secure cross-platform native apps with the ease of VueJS and the power of the Quasar Framework

Case Study: Cryptographic Enclave

In the semi-trusted environment of an application on a user-device that communicates with the other computers over the network, guaranteeing that the communication pathway is isolated from high-value algorithms is an essential strategy for the minimization of viable attack vectors.

The approach described here applies the generally accepted best practice of "Security in Breadth". In the tradeoff between security and performance, this **Cryptographic Enclave** prefers security. Nevertheless, it is still quite performant and extensible.

In this study we show a method for building a safe interface to Rust that is functionally isolated from the user interface and applies several important techniques to make attacking this interface virtually impossible:

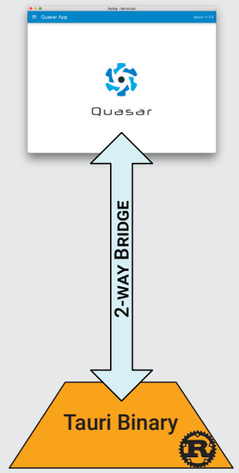
dAoT
By using a dynamic Ahead of Time compiler, we are able to generate code references that are unique for every session.

fASLR
We propose a functional address space layout randomization at boot time and optionally after every execution. Using a UUID for each critical function prevents static attacks.

OTP
Additionally, hashing important messages with OTP, we are further able to encrypt messages between the UI and the Rust backend.

Bridge
Instead of passing potentially unsafe functions, the bridge is used to pass messages and commands to and from brokers. The functionality is mapped to commands at each respective side of the bridge.

Trollbridge



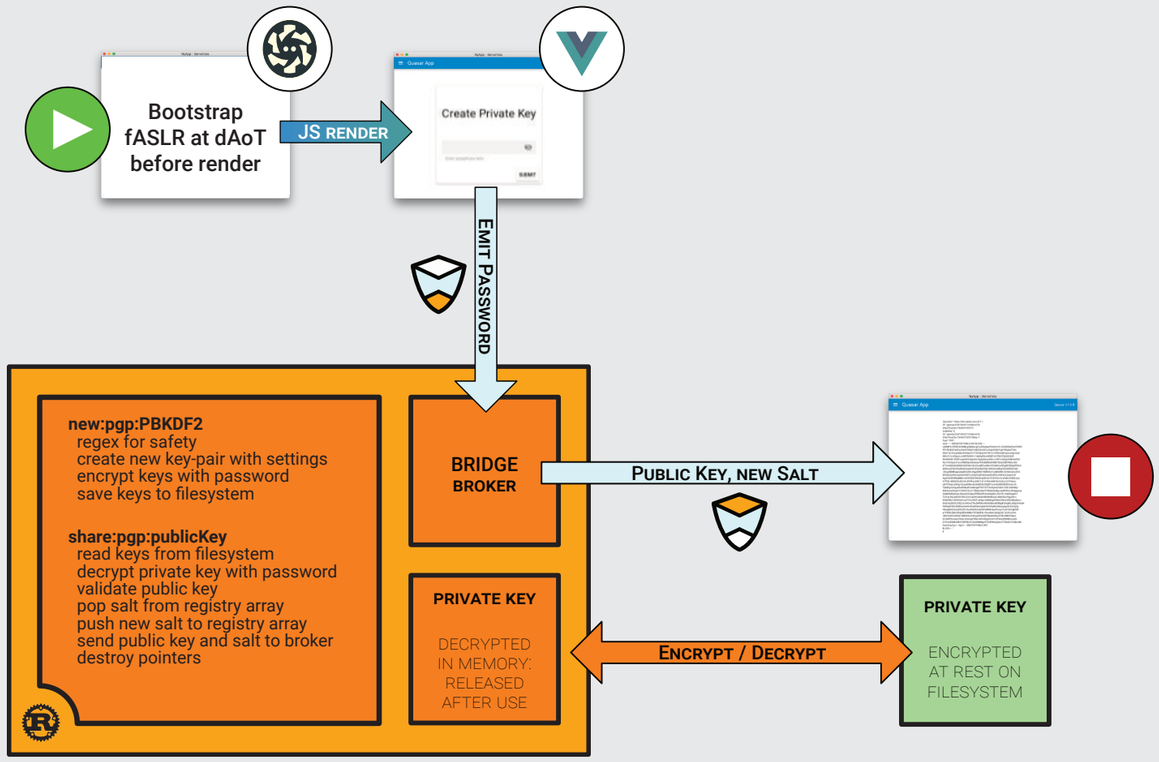
The **Trollbridge** recipe is a pattern for the highest degree of operational security.

Features:
- render UI securely at bootstrap
- promise based message passing
- RW access to filesystem
- STDOUT access to other binaries
- extensible with Rust functions
- whitelist for functional codegen
- runtime message salting
- fASLR & dAoT Compiling

Best When:
- you are paranoid about security but still need the power of Rust.

Pros:
- without a webserver, a whole class of attacks on both the UI and the server itself are mitigated out of existence

Cryptographic Enclave



The **Enclave** is an advanced **Trollbridge** pattern that uses modern PGP encryption and isolates important key generation, validation and signing processes from the UI. It uses message passing (light-blue arrows) to emit a salted and hashed password to the Rust process handler using a pre-registered function that itself is referenced with a UID that is generated by the Rust broker at every message that passes through the bridge.

As this Case Study is a high-level overview, the above flow describes a secure approach that can be used for generating a new key-pair and writing/reading with a secondary layer of encryption. It should be noted that message decryption is possible, but obviously only if the message has been encrypted with the holder's private key. Furthermore, multi-party encryption and even message signing are both trivial to implement and merely a matter of passing the right messages and architecting the right flow.