WHY FORMAL ANALYSIS

Modern day code is becoming increasingly more complex with each passing year. As programs and protocols become more complex, they become harder and harder to test. Exhaustive testing of sufficiently detailed software is near impossible to do. This can lead to disastrous results. Buggy and incomplete software isn’t only an annoyance. Depending on the use, it can be a massive security risk. Formal analysis can serve as another layer of validation to ensure the code being written does not contain any critical errors. By abstracting a piece of software to basic components and interactions, mathematical proofs can be generated to prove that certain conditions, like data integrity or security, hold.

What formal analysis helps us to find are attacks centered around the unexpected and unwanted knowledge, replication, combination, and forwarding of pieces of information. For example, a message may be checked to see if it is from a specific user by signing it with their key. If at some point the key was broadcast unencrypted on the network, any attackers could use it to sign messages and impersonate the user who it belonged to. By combining known pieces of data (the key and the message) and forwarding them, the attacker successfully impersonates someone on the network.

PURPOSE AND METHODS

The purpose of this project was to investigate the tools used to preform formal verification, specifically in the field of internet security protocols. On top of analysing a selected and unverified protocol, the other goal of this project was to test and introduce an introduction to formal analysis using the chosen tool for anyone interested in getting started.

One of the earliest decisions was which tool to proceed with for the project. After preliminary research of ProVerif and Tamarin, Tamarin was chosen. Tamarin was chosen for its use of multiset rewriting for specification, which was more familiar to the author than process calculus. This was to ensure a quicker learning curve. As far as features and testing environments, ProVerif and Tamarin are nearly identical and it is recommended to choose whichever one is most comfortable with. Tamarin is a security verification tool where protocols are specified as multiset rewriting systems and analysed with respect to first-order properties. Tamarin (along with ProVerif) use the Dolev-Yao adversary model.

To research and learn Tamarin, many online resources were used. The Tamarin manual, open source toy protocols, related papers which aided in the process are all apart of the tutorial. The official Tamarin manual has some structural issues that the new introduction sought to address and fix.

SAFEKEEPER PROTOCOL

The protocol chosen for analysis was the SafeKeeper password protection protocol. Developed in 2018, this relatively new protocol seeks to protect user passwords for websites against threat models that include compromised servers and phising sites. This is done by using a keyed one-way function within a code enclave, such as the one provided by Intel SGX. This way even a server which was owned by a malicious agent could not retrieve the plain text passwords at any point of the verification. This protocol was chosen for its relative new creation, lack of published formal verification, and for its unique solution to the password storage problem.

The SafeKeeper protocol was implemented in Tamarin and ran against multiple lennas to verify the security properties. Other papers have chosen to model the browser as a separate entity to the user, but for the sake of simplicity only three entities were modelled: the user, the trusted execution environment, and the web server. Anything passed between the web server and the TEE was visible to the attacker and added to the threat model.

RESULTS AND FUTURE WORK

The results of modelling reveal no exposure of sensitive information. However, this can only be said for lenmas which completed. Tamarin uses unbounded verification which is an undesirable problem and not guaranteed to terminate. Future work could expand on the current model which was implemented, by adding more granularity. As of now, only the server, the enclave and the user were modelled. Future models could add the browser as a separate entity to explore attacks related to the browser side. Another possible extension to the model could be the modelling of the secure channel used to communicate between the enclave and the user. The secure channel is modelled as a given between the two entities, but one could add the setup stage and exchange of shared key to the protocol.

https://www.researchgate.net/profile/SjoukeMauw/publication/221024606/figure/fig2/AS:6529445632493878@1532685523/Man-in-the-middle-attack-on-the-Needham-Schroeder-protocol.png


https://tamarin-prover.github.io/