Brian Malley

Securing Software from Unauthorized Use

Progress Report

CECS 579

CSULB

10/29/2015

Summary:

Securing software from unauthorized users is a historically common problem, with solutions of varying complexity and efficacy. This project plans to provide security from unauthorized use via two avenues; First, an authentication scheme that requires the software to request authentication from a server before use, and second, obfuscation of the software executable file to make it harder to reverse engineer. The result of the project will be two major deliverables, one for the executable obfuscation and the other the libraries for the software client and server.

Project Status:

Compared to the project timeline (found in the GitHub), the project is little bit behind schedule. The timeline had to be revised to reflect the reality of some deadlines that had passed. However, not all of the work has been delayed. The first iteration of planning has been done. The result is a block diagram of the intended client-server interaction. The executable obfuscation portion, being almost entirely separate from the licensing part, is also slightly delayed. Base research on executable obfuscation has been done and the papers added to the GitHub.

Work Done:

The first iteration of the project plan has been fleshed out. Here is a summary: The client will create an encrypted link with the server using openSSL. Over this link, a encrypted and then HMAC message will be sent containing a unique id, a timestamp, and a direction bit. Once the server has decrypted and verified the request, it will respond with a message enabling the software. On the executable obfuscation side, research has been done about the general principles of obfuscation, and currently I am searching for a way to do it.

Challenges:

One of the major challenges for me was creating the authentication scheme from scratch. Since I don't have much information security background it was hard to update the plan with the information I was learning in lecture. As I have learned more things, I have implemented them in

the design. A good example of this is the use of HMAC to authenticate the messages between the server. Before we learned of HMAC in the lecture, it wasn't included in the plan. Since I am alone in this project, another challenge is that I have to do all the work myself, which is both a boon and a hardship. For one thing, nothing gets done unless I personally do it. For another, I have to understand everything myself. However, this makes working on the project simpler, since communicating understanding between members is unnecessary. I have to be careful to not take on more than I can handle since I am alone in this project. At the same time, I alone am responsible for the outcome.

Some delays have occurred due to my underestimating of the time available to work on the project as well as it's difficulty. However, I am happy with my progress thus far and am ready to continue on to the next step. Another problem are the delays i have made in achieving my first draft roadmap. I have found it was overly optimistic and have been to reduce the number of drafts of the project to two to finish on schedule.

Roadmap:

The roadmap, taken from the revised proposal, is as follows:

Timeline of Project Completion:
First Iteration Security Scheme (block diagram, mathematic proofs) - 10/17
First Iteration Implementation (executable, server code) - 11/06
Final Iteration Security Scheme (final block diagram, math) - 11/20
Final Iteration Implementation (final executable, server code) - 12/4
This timeline allows a week for each stage. The first stage is anticipated to have the most work, as everything must be from scratch. The final stage are anticipated to be more difficult as I attempt to protect the implementation from any attacks.

Figures:

Block Diagram of the proposed security scheme:

Program Authentication Scheme