COSC 5010-03 (23194) Spring 2023 Threat Modeling

Michael Elgin

April 20, 2023

1 Context

This Threat Modeling document is in reference to the Project Design Document from early April, in which an application to create secure data files was described.

2 Threats

2.1 Discovering the data file(s)

As a first step in accessing any data that someone is not supposed to access, they first must actually get possession of it. Someone could stumble upon it by dint of happenstance, or they could intentionally seek it.

Mitigation: As a first priority, the encrypted data files should be relatively hidden. Forcing the files to have obscure names is somewhat of an overkill. Not to mention the user may very well want to give the files their own names for the sake of staying organized. By default, they will be secluded in a non-obvious location (folder) with an inconspicuous name. This "security through obscurity" is not the backbone making the application secure. It is merely the icing on the cake.

Update May 1, 2023: GUI users are now required to set their own location as opposed to being given one by default.

2.2 Determining the plaintext

Assuming the obscurity countermeasure has been defeated, the next logical question then becomes: is an encrypted data file still secure? Is there some way for an attacker to discover the plaintext from the ciphertext? Since the encryption key will be based off the password, methods for which an attacker can abuse a password-related system apply here.

Mitigation: To help prevent the end-user from being their own worst enemy, three password foot-guns shall be mitigated: First, warning that the password is not long enough to prevent brute-force attacks. Second, warning that the password is one of the most commonly used passwords. Third: warning if the password does not contain capitals, numbers, and/or special characters. Furthermore, a strong encryption algorithm will be used.

2.3 Dependencies

The software will undoubtedly make use of various dependencies. This introduces the possibility for security vulnerabilities stemming from those dependencies.

Mitigation: All dependencies shall be examined to confirm they have certain characteristics. As a minimum, they should be reputable, popular, and open-source. As an additional measure, dependabot security alerts should be enabled in GitHub to notify of any security vulnerabilities in the dependencies.