My capstone project offers a javascript implementation of the Honeywords method of detecting password cracking attempts by malicious users. The system works by storing additional, honeypot passwords alongside each user's real password. When one of the so-called "sweetwords" is used an alarm is triggered and the system can respond appropriately. A "honeychecker" server is used to either verify a login attempt or raise a flag. This is based on the work of Ari Juels (RSA) and Ronald L. Rivest (MIT).

The intent was to show how easy it is to implement a detection scheme to protect against intrusion using an additonal service and some extra storage. The method should scale well to real-world applications and can add an extra layer of security on top of existing measurers without making any parts of the system more vulnerable.

I also explored different methods of generating sweetwords as outlined in the paper. Three points of evaluation must be considered for each method of generating sweetwords; difficulting of divining the true password from the sweetwords, difficultly of generating the sweetwords, and additional storage needs of generating sweet words. Tradeoffs can be made between storage required for additional sweetwords (down to a single hash) and general case "flatness", which is a measure of the likelihood of guessing the true password from among a list of sweetwords.