To: William H. Flathead III

From: Matteo Franzil

Subject: Security issues and SQL injections

Date: October 2, 2021

This memo tries to address some urgent issues regarding our software for the Community Credit Union.

As we discussed before, in the past days there have been several breaches in our system, which resulted in a $32,767 loss for our company. After some extensive research, I've come to the conclusion that our web portal that connects to the database is severely flawed and needs to be fixed as soon as possible.

Indeed, when logging via the FCCU.php portal, we are presented with two HTML form fields that are subsceptible to a category of attacks called 'SQL Injection'. SQL Injections allow attackers to insert data within the fields (unlike the usual usernames and passwords one would write) that allow them to have access to information they shouldn't have.

In this case, it is extremely easy to exploit this flaw, and obtain access as any user in our system without knowing the password! Although this does require the attacker to try users one by one (so no direct access, if they don't know the account number), it still allows them to login as anyone once their account ID is obtained - and that one, that's exposed in plain sight once the login phase is over.

Thankfully, our software installation is new enough that allows us to fix this issue without any updates, for the moment. I've used a feature called "prepared statements" - when an attacker tries to insert malevolent data in the forms, it gets discarded by the program and interpreted as a whole string instead of a command. I've tracked and hunted down all offending usages of non-prepared statements, so for that part, we're set.

However, this does not mean that the issues are far from over. Firstly, I've noticed that the source code would kindly need some more fixes in order to be truly more secure. Indeed, some pieces of code - in particular the ones regarding withdrawals - do not make enough checks when computing balances, and risk introducing integer overflowing. Integer overflowing happens, in case you do not know, when a number gets too large for the system to be processed, so it flips over and starts from the lowest negative number available. The opposite is also true.

Secondly, I'd like to address the issue of the breach itself and how to safely recover from it. Although it was very serious, some limitations in the engine allowed us to get out of the mess safer than expected. Indeed, it was impossible for attackers to edit balances directly or deviate otherwise from the behavior of the program (so no creation or deletion of accounts too). Right now, our top priority should be to secure the privacy of our workers, by changing all of their passwords, implementing my patch, then reducing the permissions for the mysql user tied to the program - just in case. I have to admit, this is not enough to deem our system fully secure. The code needs some deep refactoring, but for the near future, let's stick to the urgent issues.