Making Secure Easy-to-Remember Passwords

Philip Braunstein

December 14, 2015

Abstract

Password leaks are a notorious way that systems get compromised. Passwords are often leaked exactly because they are heard to remember, which prompts people to record them in insecure locations and avoid changing them. This report describes and evaluates three password generation strategies. Three password generation strategies are evaluated in this report: random strings of characters, numbers, and symbols; abbreviations of phrases mixed with meaningful numbers; and an adjective-noun-verb-adjective-noun string modeled after xkcd 936 SOURCE: XKCD . Passwords are evaluated on ease of memorization, cryptographic strength against a password cracker, and bits of entropy

Introduction

People like to imagine that computer security is usually compromised by genius hackers exploiting inscrutable vulnerabilities. Often however, an attacker compromises a system because of seemingly stupid reasons like the being written on a note next to the computer. Shoring up technical security is a worthy cause, but this effort is rendered irrelevant unless the impact of social engineering resulting in password leaks is minimized.

People avoid changing their passwords, leave them written in plain text in a document on their computer or even on a sticky note on their computer for one reason only: secure passwords are hard to remember. Insecure and poorly-stored passwords are the cause of many security leaks FIND SOME GENERAL SOURCES . Passwords are a common source of vulnerability because different websites have different requirements for what they consider valid passwords, and passwords that are considered secure are obtuse and hard for people to remember.

In this report, three methods for making passwords are described and evaluated. The security and how easy each type of password to remember is evaluated and described in the Results section.

To the Community

Methods

Overall Description of Procedure

Each of the password generation methods described below were tested on each subject. In order to prevent bias based on subjects getting used to the memorization task, the script `determineOrder.py` was used to dictate the order that each of the password methods was tested.

For each password method, participants chose the password they wanted to use, and immediately wrote it out on paper five times in a row. After this, participants were given 1 minute to remember their password in any way of their choosing. Then all written representations were removed from the vicinity of the participant, and the participant was instructed to wait 1 minute with no distractions (such as using a computer, reading, speaking with others, etc.). After this buffer minute, participants wrote down their password to the best of their ability.

The entire procedure was explained to the participant without hiding anything as it was deemed that knowledge of the goals of the experiment would not damage the outcomes of the experiment.

Password Generation Schemes

Random String

A random string password consists of ten random characters, numbers, and symbols. A random string passwords must have contain at least one of three of the four categories: upper-case letter, lower-case letter, number, or symbol. The script `randomPass.py` generates ten of these random strings passwords. Participants selected one of the ten passwords to use.

Memorable Phrase and Number

Participants chose a memorable phrase and made an acronym of the first letter of each word. Every initial was lower-case in the password. They also chose a particular number and incorporated it after the initials of the memorable phrase. For example, I might choose the hook from a famous Rolling Stones song (**I** **c**an't **g**et **n**o **s**atisfaction, and the month and year of my graduation from college (May, 2012) to generate the following password: icgns0512.

Modified XKCD

Randall Munroe of XKCD suggests using several common words that can be used to create a more coherent memory as opposed to other password generation methods. I have improved on his idea by forcing passwords to obey the form adjective noun verb adjective noun, where the words from these parts of speech are drawn from publicly available word lists CITE WORD LISTS .

The script `humanPass.py` creates ten of these modified XKCD passwords, from which the participant chose one. The passwords were written with all lower case letters and no spaces between the words. Participants were allowed to alter the password to fix any grammatical mistakes. For example one password generated by `humanPass.py` was `availablenervebitemolecularradish` (available nerve bite molecular radish). Participants were instructed to modify the passwords so that they make grammatical sense, but the exact modification was left to the discretion of the participant. For example, the previous password could be modified in one of two ways `availablenervebitesmolecularradish` (available nerve bites molecular radish) or `availablenervesbitemolecularradish` (available nerves bite molecular radish). Participants were encouraged to make whatever grammatical modification makes the most sense to them.

Password Evaluation

Success Remembering Password

Strength Against Password Cracking

Bits of Entropy

Results

Applications

Conclusion