Making Secure Easy-to-Remember Passwords

Philip Braunstein

December 14, 2015

Abstract

Three password-making 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

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. Test subjects select one of the ten passwords to use.

Memorable Phrase with Memorable Number

Test subjects choose a memorable phrase and make an acronym of the first letter of each word. Every initial is lower-case in the password. They also choose a particular number and incorporate 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 test subject chooses one. The passwords are written with all lower case letters and no spaces between the words. Test subjects are allowed to alter the password to fix any grammatical mistakes. For example one password generated by `humanPass.py` was `availablenervebitemoleculararradish` (available nerve bite molecular radish). Test subjects are instructed to modify the passwords so that they make grammatical sense, but the exact modification is left to the discretion of the test subject. For example, the previous password could be modified in one of two ways `availablenervebitesmoleculararradish` (available nerve bites molecular radish) or `availablenervesbitemoleculararradish` (available nerves bite molecular radish). Test subjects are encouraged to make whatever grammatical modification makes the most sense to them.

Results

Applications

Conclusion