Exercises on Password Security

Solve the following exercises and submit them until the communicated date.

LB-PS 01.

a) In the *Attacker VM*, generate Linux (SHA-512-crypt) password hashes for the passwords `1234`, `password` and one password of your own choosing with a maximum length of six characters. Compute and discuss the key space of every password by assuming the password's length and an (for the attacker) optimistic character set, i.e., only decimal digits for the password `1234` and only lower-case letters from the alphabet for the password `password`. To generate a password hash and save it to a file, use the command (without line breaks or dollar signs!) `mkpasswd -m sha-512 "$password" > $filename`.

b) Analogous to a), generate Windows (NTLM) password hashes for the same passwords, but compute and discuss the key space of every password under a more (for the attacker) pessimistic assumption for the password length and character set. To generate a password hash and save it to a file, use the command (without line breaks or dollar signs!) `smbencrypt "$password" 2>&1 | tail -n 1 | cut -f2 > $filename`.

LB-PS 02.

a) Use *hashcat* to brute-force **all** passwords corresponding to the hashes generated in exercise 01., if this is possible within a reasonable time limit, e.g., 10 minutes per password. Compare the key space size (contained in the *progress* status output) to your own computation from exercise 01. Save the status output as you will need it for the following exercises. A tutorial on how to use *hashcat* is available in the How to use *hashcat* Section.

b) Analyze the speed and time (the total time in case of successful brute-forcing, and the estimated time otherwise) of the status outputs from exercise a), with a special focus on the difference between Linux (SHA-512-crypt) and Windows (NTLM) password hashes. Describe how long each password can be used safely given a brute-forcing adversary with your measured speed (computing power) as well as a better-equipped adversary whose computing power is a thousandfold of your own.

c) Download the list of the 100000 most common passwords from `https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/xato-net-10-million-passwords-100000.txt` on your host computer and copy it into your VM via the shared folder. Repeat exercises a) and b) with this password list instead of brute force and compare your new results to the old ones. Discuss the security implications of using a common password in general. In addition, provide a **well justified** recommendation for a safe, long-lasting password based on the observed results.

How to use *hashcat*

The tool *hashcat* computes hashes of a defined pattern or list of passwords to find the one that matches a given input hash. The general recommended invocation command in the context of this course looks like this[1] (without line breaks or dollar signs!):

```
hashcat --potfile-disable --status -O -m $hashtype -a $attackmode
$filename $parameter
```

`$filename` specifies the file name containing the target hash, `$hashtype` has to be set to 1800 for Linux (SHA-512-crypt) password hashes and to 1000 for Windows (NTLM) password hashes, respectively. `$attackmode` has to be set to 3 for brute force and to 0 for password lists, respectively (note that the `-a` parameter and its value may be omitted entirely in the latter case).
When using password lists, `$parameter` needs to be set to the path of the password list file. When using brute force, `$parameter` needs to specify the pattern of the password which consists of a single-quoted string of `?$charactertype` character specifications which is as long as the password and where `?l` denotes lower-case letters, `?u` denotes upper-case letters, `?d` denotes decimal digits, `?s` denotes a selection of common special characters and `?a` denotes a combination of all of the above. For example, the string `'?d?d?l?l'` specifies a four-character password consisting of two decimal digits at the beginning and two lower-case letters at the end.
If the pre-defined character types are insufficient, a new character type may be introduced by adding the `-1` parameter after `$attackmode` in the command line arguments, followed by a single-quoted string denoting the associated combination of pre-defined character types as described above. For example, `-1 '?l?u'` defines a new character type containing both, lower-case and upper-case letters. The new character type can be referred to as `?1` in character type specifications.
More information on the use of *hashcat* may be found via `hashcat --help` as well as in the man pages.

Notes

[1] `--potfile-disable` prevents *hashcat* from storing successfully determined passwords permanently to allow for repeated processing of the same file; `--status` shows periodic status updates in the output; `-O` enables the use of optimized *OpenCL* kernels which are usually faster.