CSE-C3400 Information security
Examination 2015-10-22

3. User authentication – example solutions

Our immensely popular *potplant* service has one million users, who have to select 12-character passwords. The character set for the passwords is the following:

abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890-+

The service stores the passwords in a database as hash values. The hash function is SHA-256, which is computed on the concatenation of string "potplant" and the password and then truncated to 16 bytes:

hash = leftmostbytes(SHA-256 ("potplant" | password), 16)

The attacker has obtained the user and password database with an SQL injection attack and mounts a brute-force attack on the hashes. The attacker is using an array of top-end GPUs, which each can compute 1000 million (10^9) SHA-256 hashes per second. The price of a GPU day is approximately $1 including the hardware, electricity and other costs. Based on this information, how much does it cost to crack:

(a) the password of the user *alice,*
(b) the password of at least one user,
(c) all the passwords?

Later, we decide to improve the password hash function by adding the username to the hash input and by saving the full 32-byte hash values, and we require all users to log in once so that the hashes can be upgraded to the new version.

hash = SHA-256 ("potplant" | username | password)

(d) How does the cost of the attack change for cases (a)–(c) as the result of this improvement?

Since you do not have a pocket calculator, a rough estimate is ok. However, please write down the intermediate steps of the calculation. (1 day = 86 400 s)

How to solve the above problem?

Notes: Remember that $10^3 \approx 2^{10} = 1$k, $10^6 \approx 2^{20} = 1$M, $10^9 \approx 2^{30} = 1$T, $10^{12} \approx 2^{40} = 1$T etc. As a computing professional, you should be able to convert powers of 2 to powers of 10 and back without a pocket calculator. Mental arithmetic helps to avoid mistakes even though you should do the actual work in Matlab or Excel. Knowing intuitively the difference between G and T (or G&T, for that matter) is no different from an accountant knowing intuitively the difference between a million or billion, or an automobile engineer never making mistakes between 100 km/h and 100 000 km/h.

Solution in base 10:
12 character-long passwords, $64=2^6$ different characters, makes $2^{6*12} = 2^{72} = 4*2^{70} \approx 4*10^{21}$ different passwords.
There are 10^6 users.
It takes 1 hash / tried password.
We boldly approximate $86400 \approx 10^5$.
10^9 hashes/second/gpu $\approx 10^{9+5} = 10^{14}$ hashes/day/gpu = 10^{14} hashes/dollar.

(a) $4*10^{21} / 10^{14} = 4*10^7$ dollars = $40 million

(b) $(4*10^{21} / 10^6) / 10^{14} = 40$ dollars because there are 10^6 winning tickets instead of just one
(c) same as (a), 40 million dollars
(d) No change to case (a). Case (b) becomes like (a) i.e. $40 million because the brute-force search for each user needs to be done separately. Similarly, case (c) becomes 10^6 times harder, costing $\$40*10^{12}$.

Notes:

- It may help to do the calculation with units. In (a), that is ($4*10^{21}$ passwords * 1 hash/tried password) / (10^{14} hashes/dollar) = $4*10^7$ dollars. Just like in physics, units help to detect errors. Type conversions, such as converting seconds to days, also become easy when you write the units down.
- (a) and (b) are approximately 50% less, if you are interested in the average time instead of guaranteed completion time.
- The truncation of the hash in (a)-(c) and increased hash length in (d) has no significance because 16 bytes is sufficiently long to avoid collisions.

The same solution in base 2:
12 character-long passwords, $64=2^6$ different characters, makes $2^{6*12} = 2^{72}$ different passwords.
There are $10^6 \approx 2^{20}$ users.
Again, we approximate $86400 \approx 2^{16}$ (remember, that is 64k).
2^{30} hashes/second/gpu $\approx$ $2^{30+16} = 2^{46}$ hashes/day/gpu = 2^{46} hashes/dollar

(a) $2^{72} / 10^{46} = 2^{26} \approx 64*10^6$ dollars = $64 million
(b) $(2^{72} / 2^{20}) / 2^{46} = 2^6 = 64$ dollars
(c) same as (a), $64 million
(d) No change to case (a). Case (b) becomes like (a) i.e. $64 million because the brute-force search for each user needs to be done separately. Similarly, case (c) becomes 10^6 times harder, costing $64*10^{12}$.