8. Computer Security (Tutorial 9)

Slides: Lecture 8 - CompSec.pdf

Notes: Lecture 8 - CompSec - Notes.pdf

Tutorial 9: Tut9.pdf

Tutorial 9 Solution: tut9_solutions.pdf

More information on Rainbow Table

https://www.geeksforgeeks.org/understanding-rainbow-table-attack/

This approach is used to find inputs that will result in a given hash value. So if you have a hash and you need to find an input that will result in this hash - this was an approach used for password searching (and other hash searches). The idea is that you do not necessarily find the password but and input that the verifier will hash to the correct value, i.e. you password is y and system stores h(y) - I find x so that h(x)=h(y) - it does not matter that I enter x as the result will be the same as if I entered y (the system will let me in).

This approach has advantages when storage was more limited and we cannot store all hash results. It is argued that with today's resources brute force/dictionary is improved.

We would start by hashing value a.

h1= h(a), h2= h(h1), h3= h(h2), h4= h(h3) ...

h100 = h(h99) ... h200 = h(h199) etc.

We would now store only some points: h1 h100 h200

now we find a candidate hash z = h(?)

We start hashing z

z1 = h(z) z2 = h(z1) z3 = h(z2)

Now we see that z3 is equal to h100. So we go back to the stored hash point prior to h100, which is h1. We start doing the hashes and find z= h(97), so a valid input? that will result in z if hashed is h(96).