Password Security

CSCE 499 Technical Presentation 2

Daniel Case: BA Computer Science/BS Mathematics
Faculty Advisor: Dr. George Hauser

Overview of Project (Progress)

- Research of Password Security
  - Three methods of password encryption
  - Three methods of password cracking

- Today's cracking technique: *Rainbow Tables*
- Time/Space Trade-Off

Rainbow Table (Pre) Construction

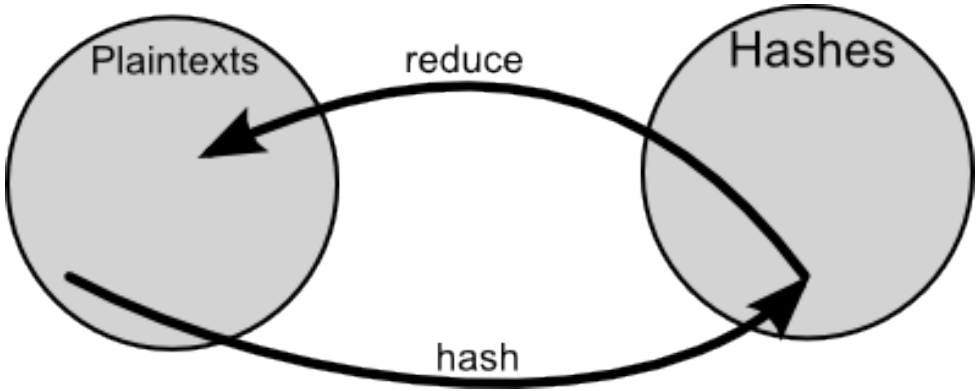
- Begins with a large *input file*
- Input file is either
  - Dictionary based
  - Built ground-up from a character set (alphabet)
- Rainbow Tables *always* based on chosen hashing algorithm

Rainbow Table Construction (Key Concepts)

- Two important concepts:
  - Reduction Functions
  - Hash Chains

- Let’s talk about Reduction Functions

Reduction Functions



- Reduction Functions map hashes to plaintext based on the *contents of input file*
- R1: Take the first 8 characters in hash, R2: Take the last 8 characters in hash, R3: R1 XOR R2

Rainbow Table Hash Chains

- Start with generated *plaintext* from *input file*
- Hash each plaintext via chosen hash function

p1	**h1=H(p1)**
3	**3955**
10	**0823**
68	**3131**
91	**2554**

R1: Last 2 digits

R2: First 2 digits

H: Map to 4 digits

Rainbow Table Hash Chains (cont)

- Apply the reduction functions to the columns
- These are the Hash Chains

p1	h1=H(p1)	p2=R1(h1)	h2=H(p2)	p3=R2(h2)	h3=H(p3)
3	3955	55	4532	45	3706
10	0823	23	5603	56	5850
25	2059	59	3626	36	4202
68	3131	31	3790	37	5520
91	2554	54	3213	32	5109

Properties of RT

- Rainbow Tables can have many Reduction Functions
- Each RF makes a longer Hash Chain
- Hash Chains are calculated - not all are stored

Properties of RT (cont)

- Time/Space Trade-Off
- Table size < number of lookups and calculations
- Rainbow Tables only store the *First Plaintext Col and Last Hash Col in Hash Chain*

p1	h3
3	3708
10	5850
25	4202
68	5520
89	5109

The Algorithm

1) Search for hashed value in Rainbow Table

    If found: goto Step 2. If not:

A) Starting with the last reduction function, *reduce* the hashed value to gain a new plaintext. (When Step 1 is repeated, use next lowest reduction function)

B) Hash new plaintext and repeat Step 1 with new hash value

2) Take the corresponding plaintext value in the RT and hash it

3) Compare the target hash

    If they match - then stop

    If not: goto Step 4

4) Apply the current reduction function, get plaintext, goto Step 2

Example

- Try it with target hash: 3626

p1	h1=H(p1)	p2=R1(h1)	h2=H(p2)	p3=R2(h2)	h3=H(p3)
3	3955	55	4532	45	3706
10	0823	23	5603	56	5850
25	2059	59	3626	36	4202
68	3131	31	3790	37	5520
91	2554	54	3213	32	5109

RT Pros and Cons

- Extremely powerful
- Can be configured for almost any password combo or hash function

- Salts make the computation and lookup of RT slow
- Better off with Brute Force, unless Salt is known
- Need decent amount of space on HD

Last Thoughts

In terms of the project…

- Gotta work on that paper

Questions?

Comments?

Algorithm explanation and example courtesy of stitchintime.wordpress.com, kestas.kuliukas.com, RainbowCrack Project, and ophcrack