Start up your Kali VM!
In a browser, go to: http://git.mikeh.am

# Password Cracking

Mike Ham

# Disclaimer

- You'll learn some real attacks, I'll show you how to get close on others
- You will NOT do this on any device that you don't own
  - Not at school, buddies' computers, etc.

# Legitimate Password Cracking

# What makes a strong password?

- Length (total number of characters)
- Complexity
  - Upper/lower case
  - Numbers
  - Special characters
- Frequency of change
  - If it takes you 60 days to crack my password, and I change it every 59…guess what?

# How do attackers break passwords?

- Really not a trick question, just want to know what you guys think.

# Two Main Categories

- Online
  - Trying to authenticate to a real service with a live, active account
  - Sitting down in front of your buddy's computer and trying to log in to their FaceBook account
- Offline
  - Getting a secure password value, taking it back to your own computer, and trying to break it

# How about a brute-force attack?

- Systematically guessing every possible combination until you get the correct answer


Yale
U.S. PAT
5961120
6422051

# Calculating Character Space

- In passwords, we can reuse characters
- Determine the number of allowed characters
  - Just lowercase: 26 total
  - Upper/lower: 52 total
  - Add in numbers (0-9) 62 total
- Figure out how long the password is
- Raise the number of allowed characters to the power of the length of the password
- (Character Space)^Length

# Calculating Character Space

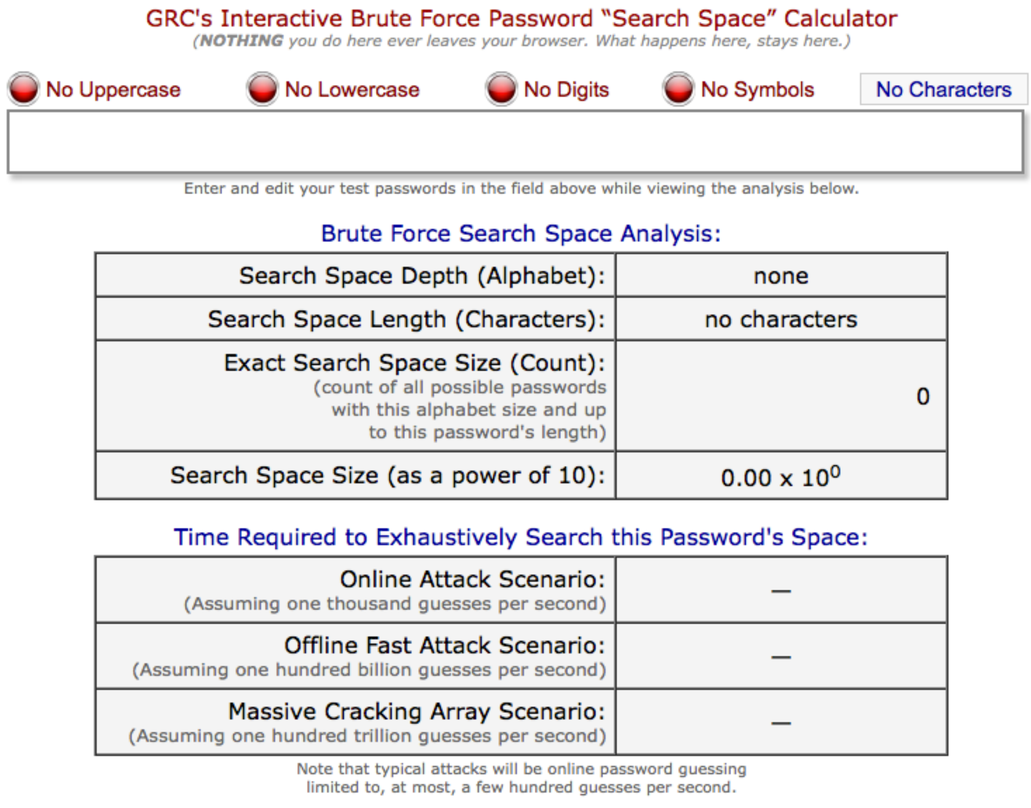
- What if you can only use numbers for your password
  - Numbers 0-9, count them, there’s 10 possibilities
- If your password is 1234 (length of 4)
- We can take 10*10*10*10 or 10^4
- There’s only about 10,000 possibilities

# Calculating Character Space

- A bit more complex…
- 26 upper case + 26 lower case = 52 total
- My password is Password (8 characters long)
- 52*52*52*52*52*52*52*52 or 52^8
- 52^8 = 53,459,728,531,456
  - Fifty-three trillion possible combinations

# https://www.grc.com/haystack.htm

- Password Haystack – how long will it take to brute force your password?



GRC's Interactive Brute Force Password "Search Space" Calculator

*(**NOTHING** you do here ever leaves your browser. What happens here, stays here.)*

No Uppercase | No Lowercase | No Digits | No Symbols | No Characters

Enter and edit your test passwords in the field above while viewing the analysis below.

Brute Force Search Space Analysis:

| | |
|---|---|
| Search Space Depth (Alphabet): | none |
| Search Space Length (Characters): | no characters |
| Exact Search Space Size (Count): (count of all possible passwords with this alphabet size and up to this password's length) | 0 |
| Search Space Size (as a power of 10): | $0.00 \times 10^0$ |

Time Required to Exhaustively Search this Password's Space:

| | |
|---|---|
| Online Attack Scenario: (Assuming one thousand guesses per second) | – |
| Offline Fast Attack Scenario: (Assuming one hundred billion guesses per second) | – |
| Massive Cracking Array Scenario: (Assuming one hundred trillion guesses per second) | – |

Note that typical attacks will be online password guessing limited to, at most, a few hundred guesses per second.

# GRC's Interactive Brute Force Password "Search Space" Calculator

*(**NOTHING** you do here ever leaves your browser. What happens here, stays here.)*

No Uppercase | 5 Lowercase | No Digits | No Symbols | 5 Characters

**cyber**

Enter and edit your test passwords in the field above while viewing the analysis below.

## Brute Force Search Space Analysis:

| | |
|---|---|
| Search Space Depth (Alphabet): | 26 |
| Search Space Length (Characters): | 5 characters |
| Exact Search Space Size (Count): (count of all possible passwords with this alphabet size and up to this password's length) | 12,356,630 |
| Search Space Size (as a power of 10): | $1.24 \times 10^7$ |

## Time Required to Exhaustively Search this Password's Space:

| | |
|---|---|
| Online Attack Scenario: (Assuming one thousand guesses per second) | 3.43 hours |
| Offline Fast Attack Scenario: (Assuming one hundred billion guesses per second) | 0.000124 seconds |
| Massive Cracking Array Scenario: (Assuming one hundred trillion guesses per second) | 0.000000124 seconds |

Note that typical attacks will be online password guessing limited to, at most, a few hundred guesses per second.

# GRC's Interactive Brute Force Password "Search Space" Calculator

*(**NOTHING** you do here ever leaves your browser. What happens here, stays here.)*

1 Uppercase | 4 Lowercase | No Digits | No Symbols | 5 Characters

**Cyber**

Enter and edit your test passwords in the field above while viewing the analysis below.

## Brute Force Search Space Analysis:

| | |
|---|---|
| Search Space Depth (Alphabet): | 26+26 = **52** |
| Search Space Length (Characters): | 5 characters |
| Exact Search Space Size (Count): (count of all possible passwords with this alphabet size and up to this password's length) | 387,659,012 |
| Search Space Size (as a power of 10): | $3.88 \times 10^8$ |

## Time Required to Exhaustively Search this Password's Space:

| | |
|---|---|
| Online Attack Scenario: (Assuming one thousand guesses per second) | 4.49 days |
| Offline Fast Attack Scenario: (Assuming one hundred billion guesses per second) | 0.00388 seconds |
| Massive Cracking Array Scenario: (Assuming one hundred trillion guesses per second) | 0.00000388 seconds |

Note that typical attacks will be online password guessing limited to, at most, a few hundred guesses per second.

# GRC's Interactive Brute Force Password "Search Space" Calculator

*(**NOTHING** you do here ever leaves your browser. What happens here, stays here.)*

1 Uppercase | 3 Lowercase | 1 Digit | No Symbols | 5 Characters

**Cyb3r**

Enter and edit your test passwords in the field above while viewing the analysis below.

## Brute Force Search Space Analysis:

| | |
|---|---|
| Search Space Depth (Alphabet): | 26+26+10 = **62** |
| Search Space Length (Characters): | 5 characters |
| Exact Search Space Size (Count): (count of all possible passwords with this alphabet size and up to this password's length) | 931,151,402 |
| Search Space Size (as a power of 10): | $9.31 \times 10^8$ |

## Time Required to Exhaustively Search this Password's Space:

| | |
|---|---|
| Online Attack Scenario: (Assuming one thousand guesses per second) | 1.54 weeks |
| Offline Fast Attack Scenario: (Assuming one hundred billion guesses per second) | 0.00931 seconds |
| Massive Cracking Array Scenario: (Assuming one hundred trillion guesses per second) | 0.00000931 seconds |

Note that typical attacks will be online password guessing limited to, at most, a few hundred guesses per second.

# GRC's Interactive Brute Force Password "Search Space" Calculator

*(**NOTHING** you do here ever leaves your browser. What happens here, stays here.)*

1 Uppercase | 2 Lowercase | 1 Digit | 1 Symbol | 5 Characters

(Yb3r

Enter and edit your test passwords in the field above while viewing the analysis below.

## Brute Force Search Space Analysis:

| | |
|---|---|
| Search Space Depth (Alphabet): | 26+26+10+33 = **95** |
| Search Space Length (Characters): | 5 characters |
| Exact Search Space Size (Count): (count of all possible passwords with this alphabet size and up to this password's length) | 7,820,126,495 |
| Search Space Size (as a power of 10): | $7.82 \times 10^9$ |

## Time Required to Exhaustively Search this Password's Space:

| | |
|---|---|
| Online Attack Scenario: (Assuming one thousand guesses per second) | 2.98 months |
| Offline Fast Attack Scenario: (Assuming one hundred billion guesses per second) | 0.0782 seconds |
| Massive Cracking Array Scenario: (Assuming one hundred trillion guesses per second) | 0.0000782 seconds |

Note that typical attacks will be online password guessing limited to, at most, a few hundred guesses per second.

# GRC's Interactive Brute Force Password "Search Space" Calculator

*(**NOTHING** you do here ever leaves your browser. What happens here, stays here.)*

3 Uppercase | 9 Lowercase | 1 Digit | 3 Symbols | 16 Characters

**I love G3nCyber!**

Enter and edit your test passwords in the field above while viewing the analysis below.

## Brute Force Search Space Analysis:

| | |
|---|---|
| Search Space Depth (Alphabet): | 26+26+10+33 = **95** |
| Search Space Length (Characters): | 16 characters |
| Exact Search Space Size (Count):<br>(count of all possible passwords with this alphabet size and up to this password's length) | 44,480,886,725,444,<br>405,624,219,204,517,120 |
| Search Space Size (as a power of 10): | $4.45 \times 10^{31}$ |

## Time Required to Exhaustively Search this Password's Space:

| | |
|---|---|
| Online Attack Scenario:<br>(Assuming one thousand guesses per second) | 14.14 million trillion centuries |
| Offline Fast Attack Scenario:<br>(Assuming one hundred billion guesses per second) | 1.41 hundred billion centuries |
| Massive Cracking Array Scenario:<br>(Assuming one hundred trillion guesses per second) | 1.41 hundred million centuries |

Note that typical attacks will be online password guessing limited to, at most, a few hundred guesses per second.

# Brute-force

- Ok, sweet.  We can take every possible combination, but what are some problems with it?
  - Takes forever
  - Account lockouts – does your bank allow you to fail logging in 100 times
  - For most things, it’s really not that practical

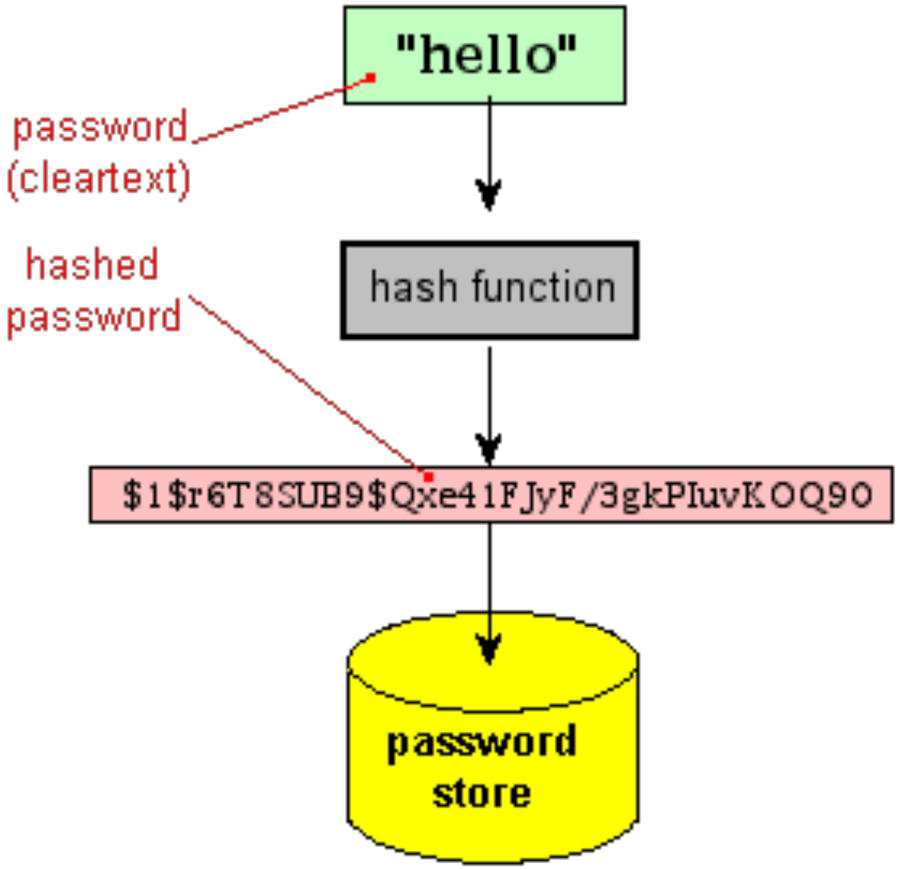
# How are passwords stored?

- Windows and most websites don’t actually store your password as a humanly readable word
    - If the system or their database get’s pwned, they’re in a world of hurt
- Hashes
- Salted Passwords

# Password Hash

- A hash is a one-way, non-reversible way to store some sort of data
  - Anyone used MD5, SHA1
- To authenticate a user, the password presented by the user is hashed and compared with the stored hash
  - Can’t get the password back from a hash

# Hashing


"hello"
password
(cleartext)
hashed
password
hash function
$1$r6T8SUB9$Qxe41FJyF/3gkPIuvKOQ90
password
store

# Windows Password Hashes

- Windows stores user passwords in the system's password database (called "SAM") in an encrypted hash
- LM Hash – old and really broken
  - Windows XP = Easy!
  - Windows Vista and above – turned off by default
- NTLM Hash – new and not as broken
  - Used in Windows Vista and above

# LM Hash Major Flaws

- Password is restricted to a maximum of fourteen characters
- Password is converted to uppercase
- This password is null-padded to 14 bytes
- The “fixed-length” password is split into two seven-byte halves

# NTLM Hash

- NTLM is case sensitive
- Character set is 65,535, and it does not limit stored passwords to two 7-character parts
- Considered a much stronger hashing algorithm
- Same concept in the end, just harder to break

# Hashes

- LM
  - 855c3697d9979e78ac404c4ba2c66533
- NTLM
  - $NT$7f8fe03093cc84b267b109625f6bbf4b
- Bunch of sample hashes:
  - http://openwall.info/wiki/john/sample-hashes

# Getting a Password Hash

- PW Dump
- Cain & Able
- FG Dump
- Get a root shell on the machine remotely, and use some 1337 tools in Kali to extract them

# Ok, we have the hashes…

- Now, it's time to break them
- We can start by brute-forcing some

```
Administrator:500:NO PASSWORD**********************:31D6CFE0D16AE931B73C59D7E0C089C0:::
Guest:501:NO PASSWORD**********************:NO PASSWORD**********************:::
DSU:1000:NO PASSWORD**********************:7FACDC498ED1680C4FD1448319A8C04F:::
Ricky:1098:8C6F5D02DEB21501AAD3B435B51404EE:E0FBA38268D0EC66EF1CB452D5885E53:::
Bubbles:1099:E52CAC67419A9A224A3B108F3FA6CB6D:8846F7EAEE8FB117AD06BDD830B7586C:::
Julian:1100:0222C809FA6E75161CE9CD479353E782:0FB7F09F6B46A6E019B5999572F1AC4E:::
Barb:1101:05083E9CECF981D65E57FC0295493699:2718034A5D4AAE8FD80A621322A0AB51:::
Dennis:1102:NO PASSWORD**********************:0AE8809CAB6D0CF7FC049790D14AA16D:::
```

# John the Ripper

- john --user:<username> <hash file>

- NOTE: The user name is case sensitive!

# Even Easier

# Dictionary Attacks

- Attackers are pretty smart
- Once you break a bunch of passwords, why not write them down
- Compile a huge list of passwords that people actually use
- Try each of those hashes to see if they’re the right one

# RockYou.txt

- In 2009, a major password breach occurred
- 32.6 million stolen passwords of the website's users disclosed
- Attackers have grabbed this, and added to it
- Our dictionary.txt has 109,583

# John the Ripper

- john --user:<username> --wordlist:<dictionary file> --<hash file>

- NOTE: The user name is case sensitive!

# Wordlists

- Sometimes they will have additional passwords you don’t need
  - What if the website requires 8 character passwords?
  - No sense in cracking something with less
- Use our dictionaries and trim them down to meet what we want in them

# pw-inspector

- Reads passwords in and prints those which meet the requirements
- -i FILE
- -o FILE
- -m MINLEN
- -M MAXLEN
- -c MINSETS
- -h, --help

# crunch

- If you know a keyword about a user, it may be good to generate your own wordlist
- crunch is a tool that will allow you to augment a word and find common modifications many users do
- crunch 5 5 ABCDEFGHIJKLMNOPQRSTUVWXYZ -o /root/Desktop/wordlist.txt
- crunch 8 8 ABCDEFGHIJKLMNOPQRSTUVWXYZ -t @@@@1980 -o /root/Desktop/wordlist.txt
- crunch 8 8 -f /usr/share/rainbowcrack/charset.txt mixalpha -o /root/alphawordlist.lst
- crunch 8 8 -t baseword@,%^

# Rainbow Tables

- Precomputed table for reversing cryptographic hash functions, usually for cracking password hashes
- Using less computer processing time and more storage than a brute-force attack
- More processing time and less storage than a simple lookup table with one entry per hash

# Rainbow Tables

- To store every single 7 character password possible for the given example, it would require a least 417 TB of space - rainbow tables only need around 80 GB of storage space, depending on your generation options

- Middle ground between brute force and dictionary attack

- https://www.freerainbowtables.com

# NTLM Rainbow Tables

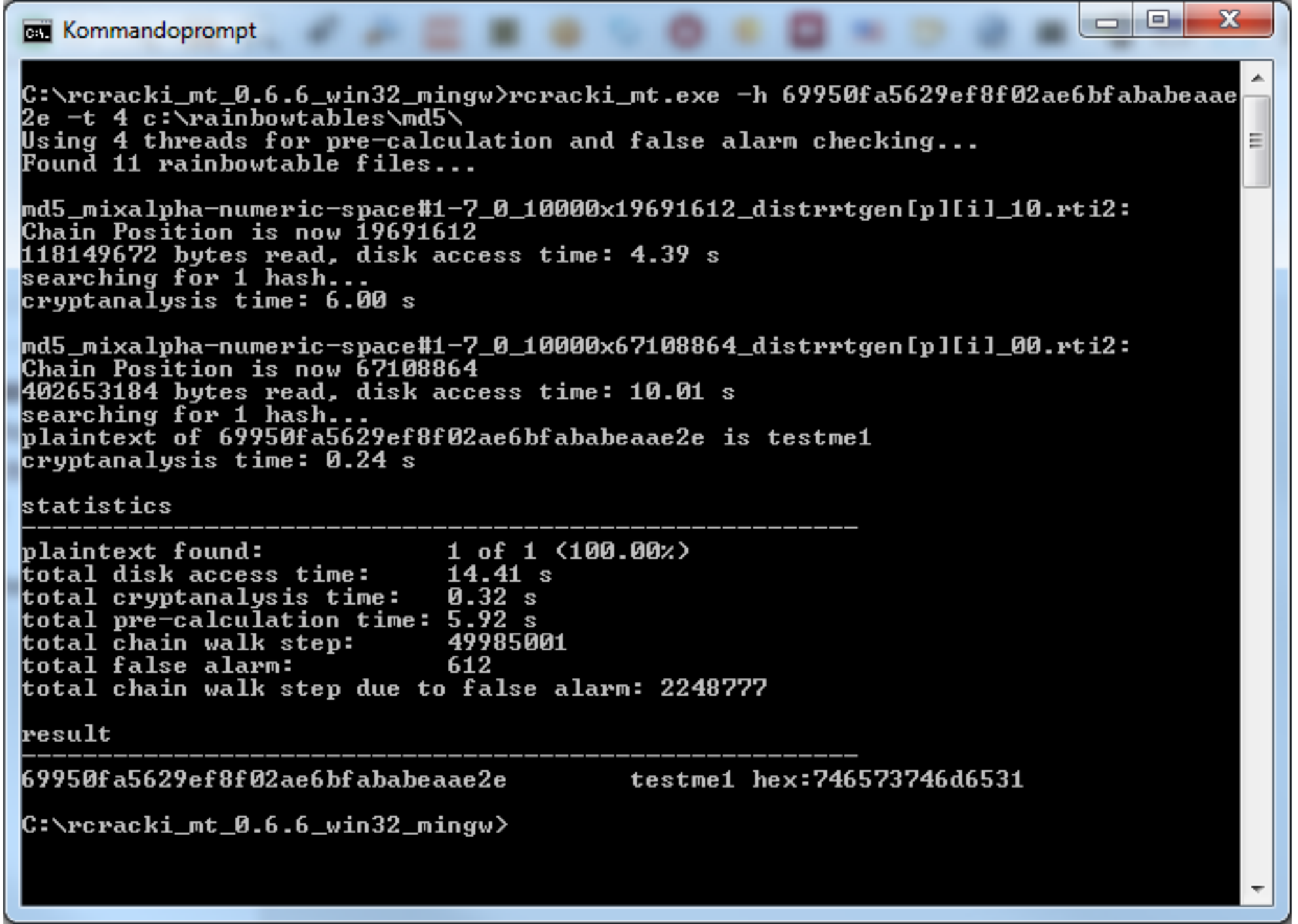
| Table ID | Charset | Plaintext Length | Key Space | Success Rate | Table Size | Files | Performance |
|---|---|---|---|---|---|---|---|
| ntlm_ascii-32-95#1-7 | ascii-32-95 | 1 to 7 | 70,576,641,626,495 | 99.9 % | 52 GB<br>64 GB | Perfect<br>Non-perfect | Perfect<br>Non-perfect |
| ntlm_ascii-32-95#1-8 | ascii-32-95 | 1 to 8 | 6,704,780,954,517,120 | 96.8 % | 460 GB<br>576 GB | Perfect<br>Non-perfect | Perfect<br>Non-perfect |
| ntlm_mixalpha-numeric#1-8 | mixalpha-numeric | 1 to 8 | 221,919,451,578,090 | 99.9 % | 127 GB<br>160 GB | Perfect<br>Non-perfect | Perfect<br>Non-perfect |
| ntlm_mixalpha-numeric#1-9 | mixalpha-numeric | 1 to 9 | 13,759,005,997,841,642 | 96.8 % | 690 GB<br>864 GB | Perfect<br>Non-perfect | Perfect<br>Non-perfect |
| ntlm_loweralpha-numeric#1-9 | loweralpha-numeric | 1 to 9 | 104,461,669,716,084 | 99.9 % | 65 GB<br>80 GB | Perfect<br>Non-perfect | Perfect<br>Non-perfect |
| ntlm_loweralpha-numeric#1-10 | loweralpha-numeric | 1 to 10 | 3,760,620,109,779,060 | 96.8 % | 316 GB<br>396 GB | Perfect<br>Non-perfect | Perfect<br>Non-perfect |

# Chains!

- Hashing the hash, we can save space
- It will take someone time to do, but the storage payoff is worth it

# Making a Chain

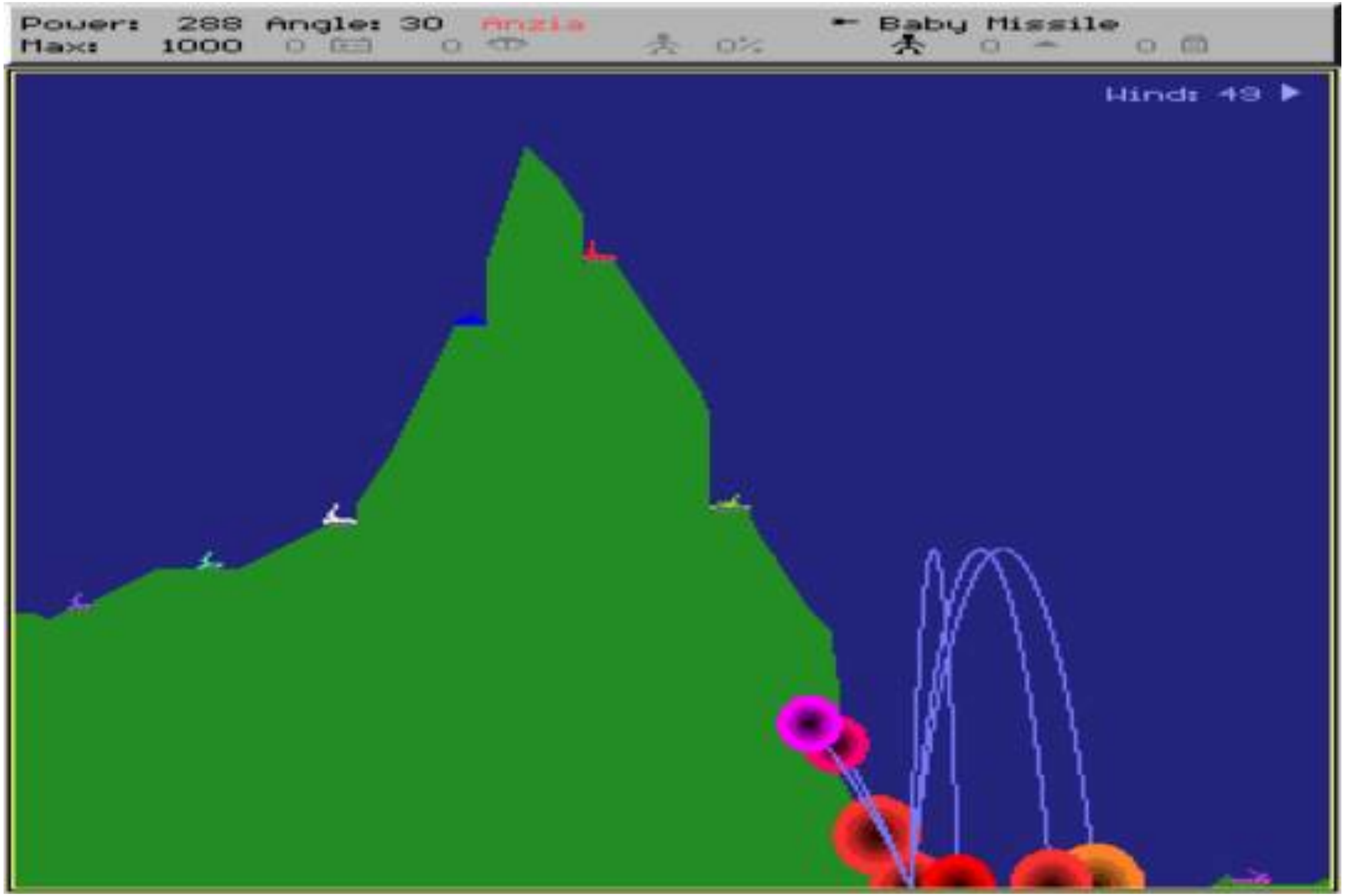
- Suppose you want the first 8 characters because that's what many passwords are and you want it in an MD5
- hashMD5(12345678) = 25d55ad283aa400af464c76d713c07ad
- hashMD5(25d55ad2) = 5c41c6b3958e798662d8853ece970f70
- hashMD5(5c41c6b3) = 4e3d15bea09e7455aa362d3cf89838fc
- hashMD5(4e3d15be) = f439b57135045b9d365dd867fca1e316


Kommandoprompt
C:\rcracki_mt_0.6.6_win32_mingw>rcracki_mt.exe -h 69950fa5629ef8f02ae6bfababeaae
2e -t 4 c:\rainbowtables\md5\
Using 4 threads for pre-calculation and false alarm checking...
Found 11 rainbowtable files...
md5_mixalpha-numeric-space#1-7_0_10000x19691612_distrrtgen[p][i]_10.rti2:
Chain Position is now 19691612
118149672 bytes read, disk access time: 4.39 s
searching for 1 hash...
cryptanalysis time: 6.00 s
md5_mixalpha-numeric-space#1-7_0_10000x67108864_distrrtgen[p][i]_00.rti2:
Chain Position is now 67108864
402653184 bytes read, disk access time: 10.01 s
searching for 1 hash...
plaintext of 69950fa5629ef8f02ae6bfababeaae2e is testme1
cryptanalysis time: 0.24 s
statistics
plaintext found: 1 of 1 (100.00%)
total disk access time: 14.41 s
total cryptanalysis time: 0.32 s
total pre-calculation time: 5.92 s
total chain walk step: 49985001
total false alarm: 612
total chain walk step due to false alarm: 2248777
result
69950fa5629ef8f02ae6bfababeaae2e testme1 hex:746573746d6531
C:\rcracki_mt_0.6.6_win32_mingw>


CloudCracker :: Online Hash Cracker
https www.cloudcracker.com
Reader
CloudCracker
An online password cracking service for penetration testers and network auditors who need to check the security of WPA protected wireless networks, crack password hashes, or break document encryption.
Start Cracking
File Type
WPA/WPA2
Handshake File
Choose File
no file selected
SSID (Network Name)
Next »
Handshake
Dictionary
Delivery
Big. Fast. Cheap.
Run your network handshake against 300,000,000 words in 20 minutes for $17.
"Welcome to the future: cloud-based WPA cracking is here!" -
- TechRepublic
"Low cost service cracks wireless passwords from the cloud..." -
- TheRegister
"This really is a great idea." -
- Hacker News
Save Money. Save Time.
Comprehensive Dictionaries.

# GPU vs. CPU

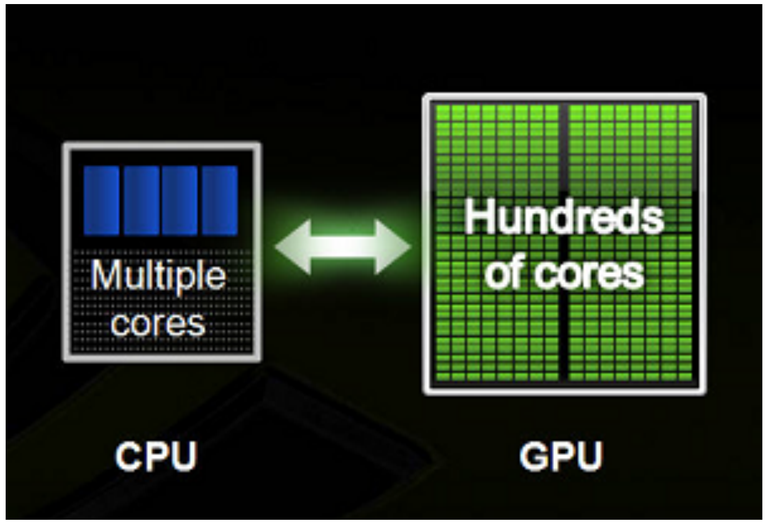
- Any gamers in here?
- I bet if you're 1337, you have a nice graphics card
  - You want full FPS, 1080 resolution, etc.
- What  a GPU really does is alleviate some of the load of the CPU and handles some computation for the output on the screen


Power: 288 Angle: 30 Anzia
Max: 1000
Baby Missile
Wind: 43

# GPU vs. CPU

- CPU has only few cores/multiple cores with lots of cache memory that can handle few software threads at a time

- GPU has hundreds of cores that can handle thousand of threads simultaneously



# GPU vs. CPU

- GPUs are highly specialized in number crunching, something that graphics processing desperately needs
- Multiple GPUs can also be employed to achieve a single goal
- TL;DR; GPUs are really fast, but designed for very specific types of applications

# Not all GPUs are created equally…

- Some graphics cards are more capable than others
- Look at how many cores they have
- How much memory
- Before you go buy a password cracking machine, do your due diligence!


msi
Twin Frozr

# Hurdles

- What about non-English passwords?
- Those that use special character sets
  - Wingdings anybody?
  - W♓▪♑♎♓▪♑’♦ ♋▪♌▫♎✍
- Change frequency
- Locking accounts out

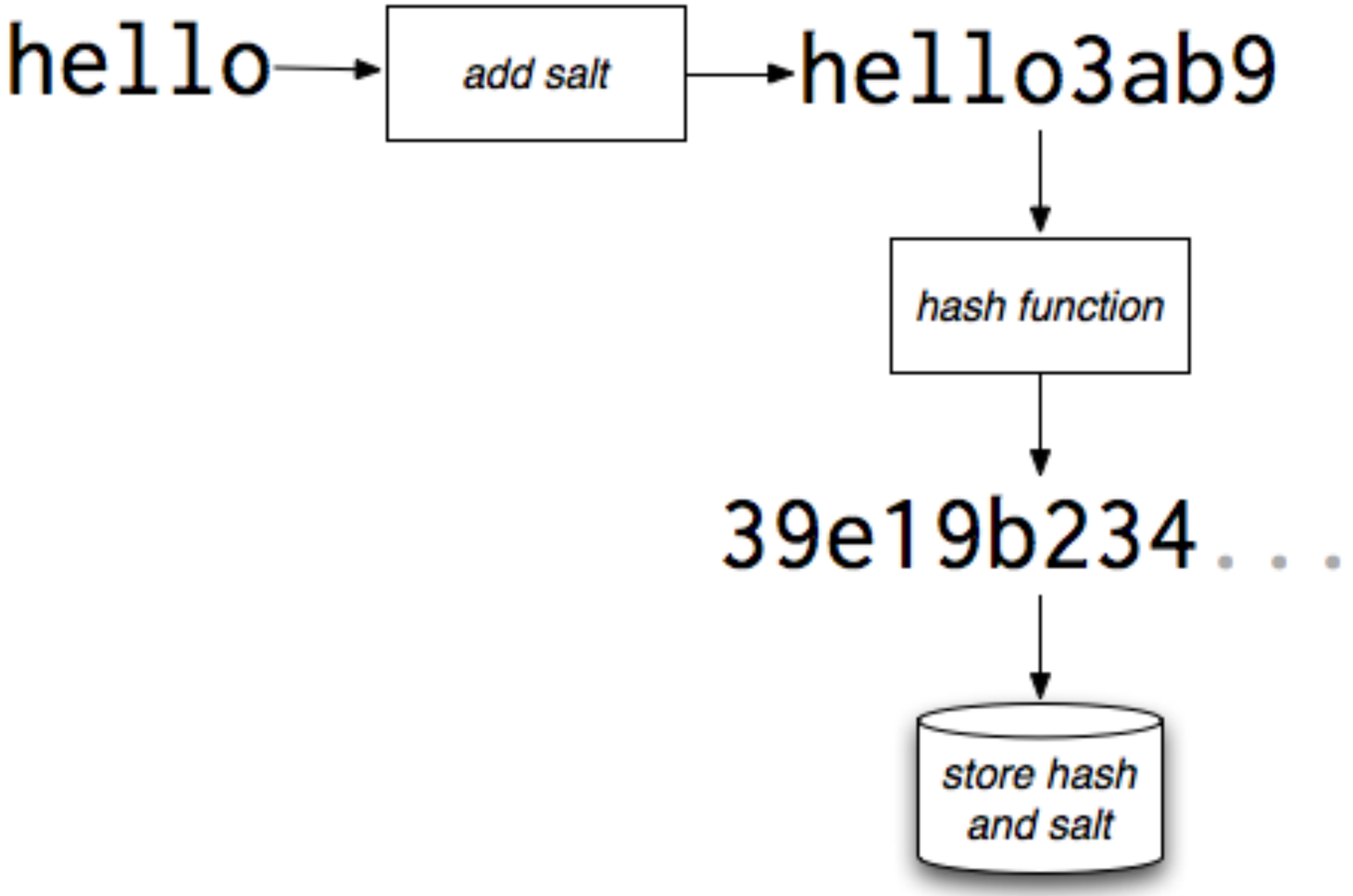
# More Iterations

- Running a hash through the hashing algorithms multiple times
- Running a password through SHA256, for instance, and then feeding the hashed output back through the algorithm another 500 times would certainly make wordlist and brute force attacks slower

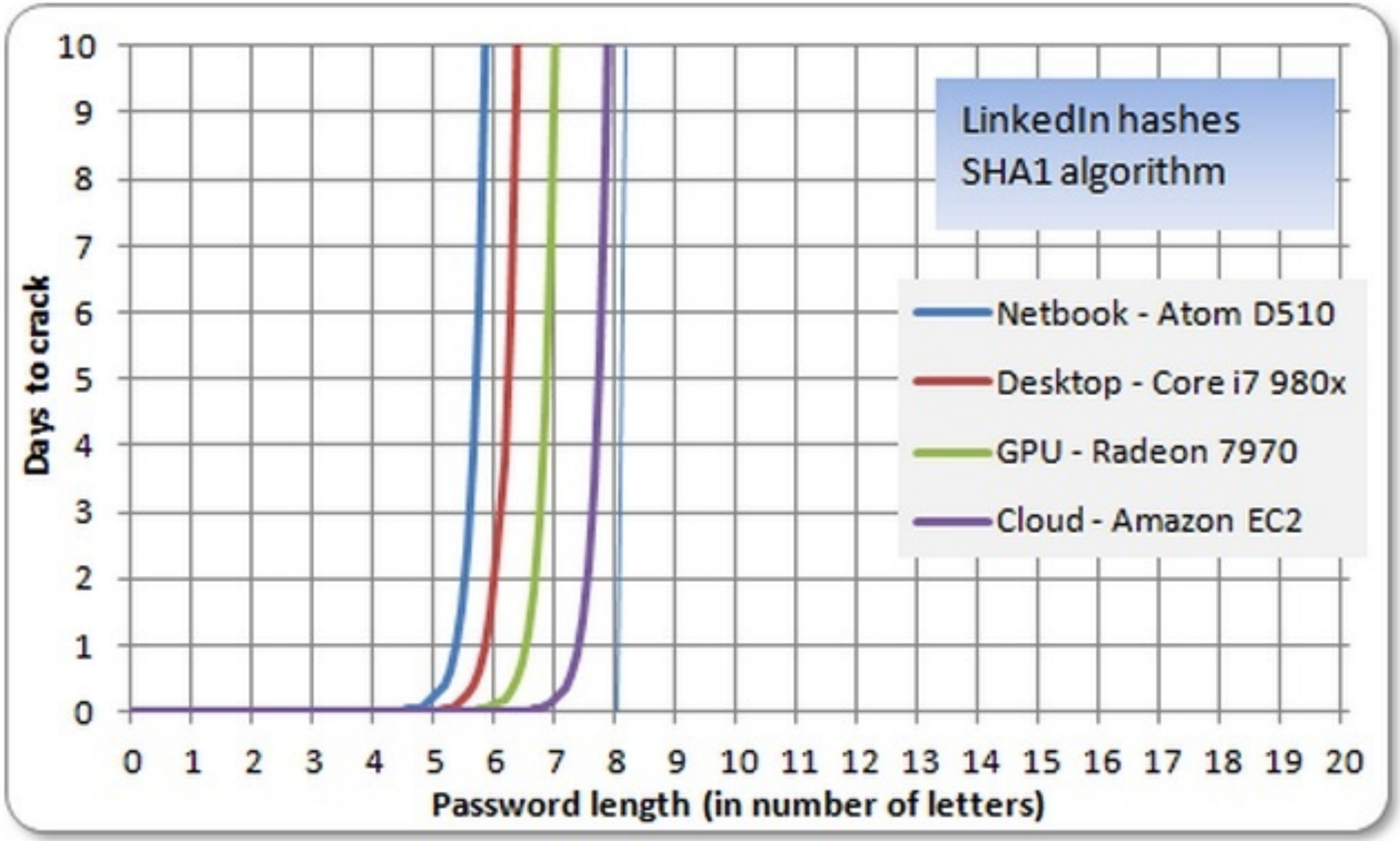
# Salting

- Salting attempts to defeat table lookup attacks by adding several characters to the password before passing it through the hashing algorithm
- Characters don't have to remain secret
- Salting also thwarts cracking techniques that rely on rainbow tables

# Salting


hello
add salt
hello3ab9
hash function
39e19b234...
store hash
and salt

# Length


LinkedIn hashes
SHA1 algorithm
Netbook - Atom D510
Desktop - Core i7 980x
GPU - Radeon 7970
Cloud - Amazon EC2
Days to crack
0 1 2 3 4 5 6 7 8 9 10
Password length (in number of letters)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

# Multifactor Authentication

- Something you know
- Something you have
- Something you are

# Multifactor Authentication


# Google Authentic