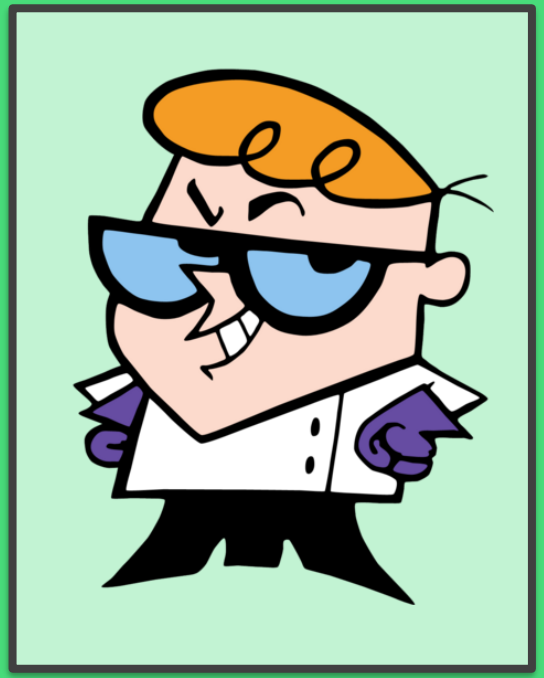
Password Attacks

(To Take Over the World)

******

Shall we play a name?

- woland
- Pentester
- Ex-journo
- I have successfully used password attacks in the field
- @wolandsec



infosec

hacking

In this talk

Beginner friendly!

- Password spraying and guessing
- Wordlists
- Hashing basics and cracking hashes
- Popular tools and how to use them
- Credential stuffing
- Defending against these attacks

Not in this talk

- Phishing and social engineering
- Cloud keys/secrets, and IAM
- Cracking rigs
- Capturing and cracking WiFi handshakes
- Authentication frameworks/protocols

Why do passwords matter?

Uber

Because they're secrets

- ***Shared*** secrets
- Widely used in everyday life
- We create them

This produces security weaknesses.

So you want to break into a system….

- You have permission, *right*?
- Is it in writing?
- The target *belongs* to the those who gave the permission?

First steps

- Default credentials are still a thing
- Are passwords already publicly exposed?

Out-of-Box Role	Out-of-Box User
Administrator	admin
Business Analyst Role	pat
Power User Role	suzy
Report Author Role	tiffany

password: password

Passwords on the internet

- Google:
  site:domain.com intext:password, filtetype:pdf
- https://dorksearch.com
- Instagram/Google photo geolocation:
  Pictures of badges, desks, and desktops (oh my!)

Great for policy info, password list ideas

Password attack basics

Password guessing

- Submitting a list of passwords for 1 or more usernames
- Automated or manual
- No real concern for account lockouts, or noise

Password spraying

- Periodic password guessing using 1 or a limited number of passwords against a large list of usernames
- Used to avoid lockouts, stealthier

Credential stuffiing

- Using publicly available credentials in order to gain access
- Often third-party breach data
- **Be careful** - there are ethical and legal concerns
- Is it effective?
- Resources:

  leak-lookup.com
  dehashed.com
  torrents:
  https://github.com/hmaverickadams/breach-parse

Brute force attacks

- Arduous and time-consuming password guessing
- Every possible combo
- No lockout + allowing infinite login attempts + bad password policy = brute force weakness

Who are the users?

- phonebook.cz (free with account)
- hunter.io (limited info for free)
- Public breach data
- Company website
- Linkedin
- theHarvester

```
theHarvester.py -d domain.com -b all
```

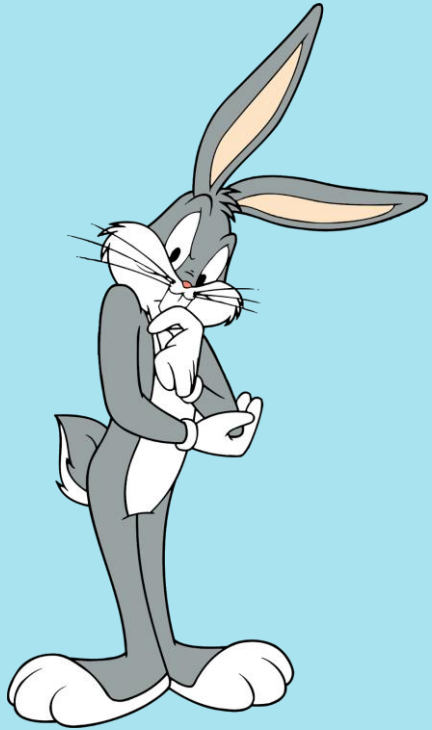
root
admin
guest
user
sysadmin
test
administrator
webadmin
anonymous

User Enumeration

- Discovering valid users of a system
- Submitting input and looking at responses

Logins, password resets, user registration

“The password and/or username is incorrect.”
“Invalid user”
“The user account does not exist”
“The user account already exists”
“If the email exists, a reset link has been sent to your account”

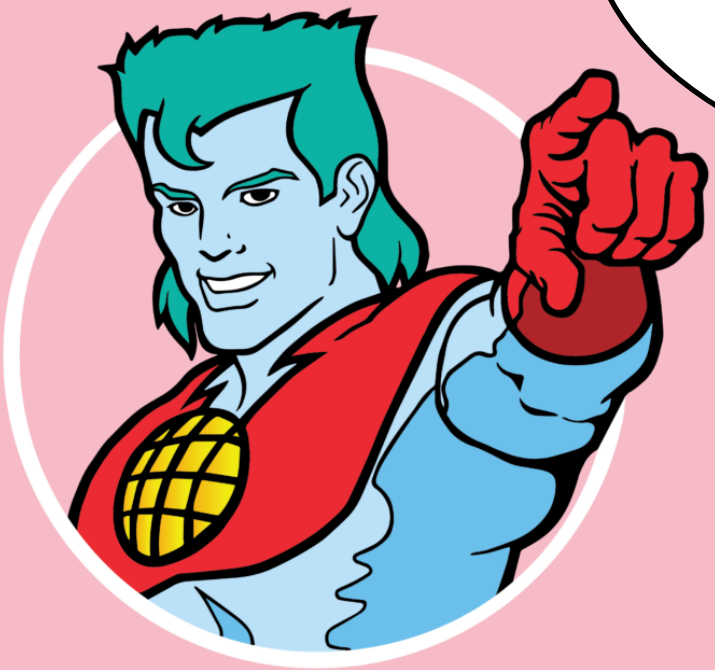


Wordlists

- https://github.com/danielmiessler/SecLists
- http://weakpasswords.net (seasonal password list)
- **Follow the password policy** – length, special characters, numbers?
- **Custom lists**

Seasons, months, regional sports teams, pets, children, school mascots, variations of default passwords

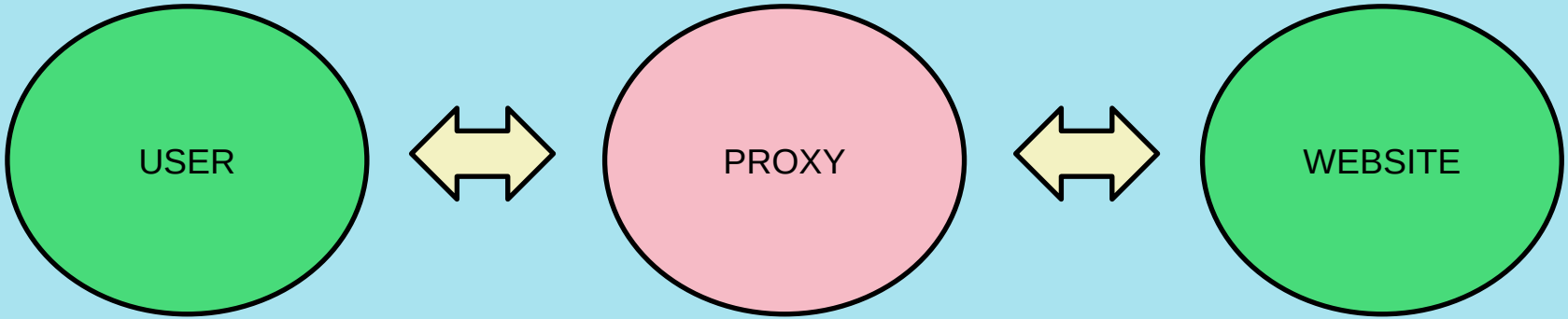
- **CeWL** – Scrape websites for common words and emails
- **crunch** – Generate wordlists and permutations
- **cupp** – Takes interactive input and generates wordlists



Password lists are only as strong as your knowledge of the target. Be good to your lists and they'll be good to you. The power is yours!

Web App Tools

- Proxies!
- Burp Suite Professional Intruder
- OWASP Zap Fuzzer



```
1  POST /rest/user/login HTTP/1.1
2  Host: [redacted]
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:104.0) Gecko/20100101 Firefox/104.0
4  Accept: application/json, text/plain, */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Content-Type: application/json
8  Content-Length: 34
9  Origin: [redacted]
10 Connection: close
11 Referer: [redacted]
12 Cookie: language=en; welcomebanner_status=dismiss
13
14 {"email":"$test$","password":"$test$"}
```

Results | Positions | Payloads | Resource Pool | Options

Filter: Showing all items

Requ...	Payload 1	Payload 2	Status
0			401
1		password	401
2	user	password	401
3	admin	password	401
4		password1	401
5	user	password1	401
6	admin	password1	401
7		qwerty123	401
8	user	qwerty123	401
9	admin	qwerty123	401

Exp

Payloads
admin, password
admin, password1
admin, qwerty123
user, password
user, password1
user, qwerty123
test, password
test, password1
test, qwerty123

Fuzzer

Fuzz Locations | Options | Message Processors

Header: Text | Body: Text

```
POST [redacted] HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:104.0) Gecko/20100101 Firefox/104.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/json
Content-Length: 35
Origin: [redacted]
Connection: keep-alive
Referer: [redacted]
Cookie: language=en; welcomebanner_status=dismiss
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
```

```
{"email":"admin","password":"test"}
```

thc-hydra

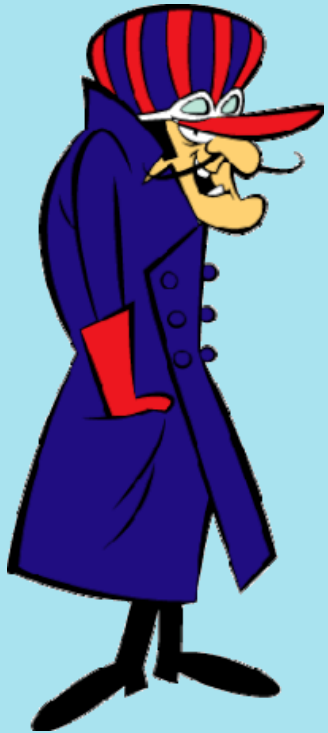
- Command line password guessing and spraying tool
- Supports multiple protocols

```
./hydra -L /home/pentester/users.txt -p
password321! 127.0.0.1 ftp

./hydra -L /home/pentester/users.txt -P
/home/pentester/passwordlist.com
127.0.0.1 ssh
```

o365spray

- For clients that use Office365
- Also made for user enumeration
- Get information about domain and lockout policy:
  `./o365spray.py --validate --domain domain.com`
- Spray with one password every 20 minutes:
  `./o365spray.py --spray -U /home/pentester/useremails.txt -P /home/pentester/weak_passwords.txt --count 1 --lockout 20 --domain domain.com`



Hashes?

Encryption?

Cracking?

Offline password attacks

What the #$%! is a hash?

- The storage problem
- A one-way function that turns text into a unique string of characters

"cartoon" + math stuff = 46e25e123679ab1c022b431dc86ee0a2

- Trap door math functions:

1093 x 1039 = 1,135,627, now reverse that

- Hashing algorithms are similar
- And oftentimes hashes get hashed!
- And there's multiple algorithms for hashing things (MD5, SHA512, and many more)

Hashing is not encryption

- Hashing is a one-way function where it is not expected that clear text can be recovered
- That's the point
- Encryption is a two-way function where clear text can be obtained after decryption

But wait a minute...

Recovering a password from a hash

- We have 46e25e123679ab1c022b431dc86ee0a2

"topsecret" = ea847988ba59727dbf4e34ee75726dc3 ✗
"password" = 5f4dcc3b5aa765d61d8327deb882cf99 ✗
"cartoon" = 46e25e123679ab1c022b431dc86ee0a2 ☑

- As password complexity increases, the harder it is to find a match

Rainbow Tables

crackstation.net
project-rainbowcrack.com

Getting salty

- What if you take "cartoon" and append a string of random characters so it's "cartoon**MOO**"
- What if you do that for every password?

Where do hashes come from?

- Compromise and hashes are dumped or copied
- Intercepted through traffic
- They were left exposed
- You're doing a password audit and extracted them

Linux: A tale of two files

/etc/passwd

```
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
```

/etc/shadow

```
gdm:*:18858:0:99999:7:::
sssd:*:18858:0:99999:7:::
[redacted]:$6$BSPcYYBwsoaUBMWF$1D/wvX5/EVqh2KhPxS6te7.1kLqmt5lWRm00GZ9QgYVXpTo1pbLwNY1LhH4JC2dx3HJAMe1Puez2uWCxa2XUh0:18893:0:99999:7:::
systemd-coredump:!!:18893::::::
```

- **$6** – The hash type (SHA-512)
- **$BSPcYYBwsoaUBMWF** – The hash salt
- **$1D/wvX5/EVqh2KhPxS6te7.1kLqmt5lWRm00GZ9QgYVXpTo1pbLwNY1LhH4JC2dx3HJAMe1Puez2uWCxa2XUh0** – The salted hash

A tale of two Windows hashes

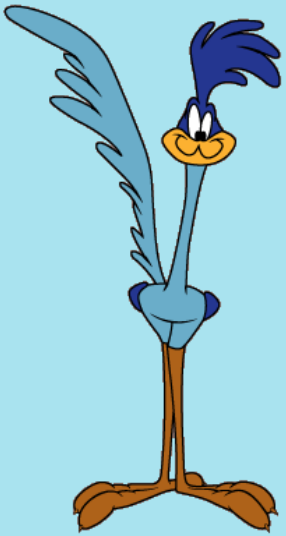
Administrator:500:aad3b435b51404eeaad3b435b51404ee:8118cb8789b3a147c790db402b016a08

- Obtained from a local machine or a domain controller
- Administrator rights needed

Administrator – username
500 – user ID
aad3b435b51404eeaad3b435b51404ee – LANMAN hash
8118cb8789b3a147c790db402b016a08 – NT hash

- LAN Manager (LANMAN) hash: Bad and easily cracked
- NT hash: Better, still crackable

john the ripper



- Solid and easy to use
- Combine two Linux files: **`unshadow /etc/passwd /etc/shadow > passwords.txt`**
- Use wordlist: **`john --wordlist=list.txt passwords.txt`**
- NT cracking: **`john --format=NT --wordlist=list.txt hashes.txt`**
- LANMAN cracking: **`john --format=LM --wordlist=list.txt hashes.txt`**

hashcat

- Powerful, badass, complex
- Uses hardware power well
- Many useful modes and custom rules
- **`-m`**: type of hash being cracked
- **`-a`**: type of attack being used

-m (hash type)

- **0:** MD5
- **1000:** NT
- **1800:** SHA512 (Linux)
- **3000:** LANMAN

-a (attack)

- **0:** Wordlist attack
- **3:** Brute force
- **6:** Wordlist and mask
- **7:** Mask and wordlist

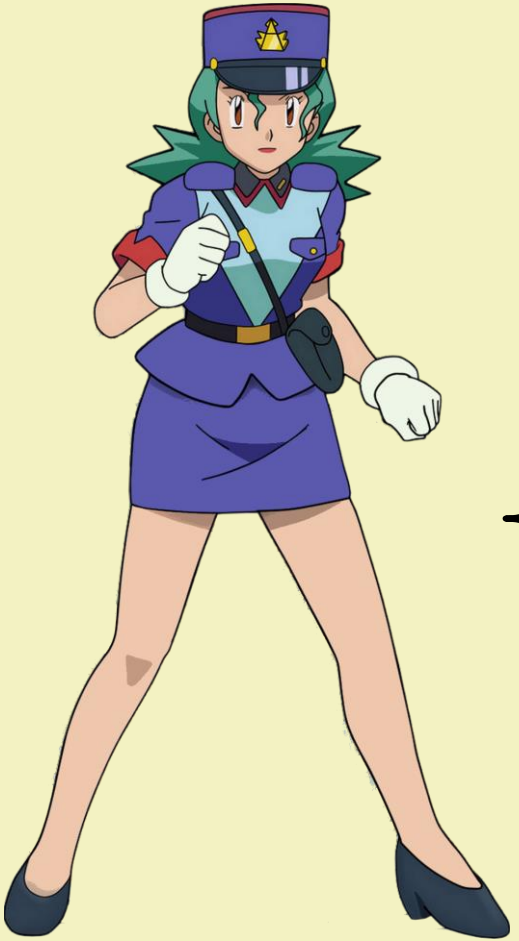
Some hashcat masks

- **?l** = abcdefghijklmnopqrstuvwxyz
- **?u** = ABCDEFGHIJKLMNOPQRSTUVWXYZ
- **?d** = 0123456789
- **?s** = «space»!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

Hashcat command examples

- wordlist attack: **`hashcat -m 1000 -a 0 hashes.txt list.txt`**
- brute force: **`hashcat -m 1800 -a 3 hashes.txt ?u?l?l?l?d?s`**
- wordlist and mask: **`hashcat -m 3000 -a 6 hashes.txt list.txt ?s`**
- mask and wordlist: **`hashcat -m 1000 -a 7 hashes.txt ?d?d?d?d list.txt`**

Best security practices



- Not everyone agrees…
- Unique, strong passwords are key
- Don’t reuse them
- Password managers are nifty
- Multi-factor (MFA) authentication is even better!
- Mandatory password changes aren’t helping
- Audit yourself
- Domain Password Audit Tool (DPAT) - https://github.com/clr2of8/DPAT

A secure and passwordless future

- It’s asymmetric encryption
- FIDO alliance and W3C = FIDO2 project
- WebAuthn is young
- Adoption will take time… and money

Thank You