Fireside Fridays

Anatomy of a VPN
Part 1 of 3

Thanks to our sponsors!

ACTIVE COUNTERMEASURES

Antisyphon Training

Lab requirements for this section

- Download a totally safe file:

```
wget https://random-class.s3.us-east-1.amazonaws.com/vpn-lab1.tar.gz
```

- Then open the archive

```
tar xvzf vpn-lab1.tar.gz
```

- You'll find two image files

```
cd hashing
ls -al
```

Anatomy of a VPN

- All VPNs should include:
  - Initial authentication
  - Set up a secure channel over an insecure medium
  - Privacy for all passing data
  - Method of authenticating every packet in the session
- SSH, TLS, IPSec, etc. provide the above
- We'll cover each of these once we know the basics

Quick side trip regarding crypto

- Encryption

  A method of converting plaintext into an alternate format, called ciphertext, that is easy for select individuals but hard for everyone else to convert back into plaintext.

- Hashing

  A method of converting plaintext into an alternate format, called a hash, that should be difficult if not impossible for anyone to return to the original plaintext.

Symmetrical key encryption

- Simplest and most frequently used encryption
- Provides only data privacy
- Same key encrypts as well as decrypts
- Relatively low CPU hit
- Chicken/Egg problem
  - Don't trust the medium so I want to encrypt the data
  - How do I protect the key in transit?
  - I could encrypt it, but that requires another key

Encryption vulnerabilities

- Key size is too small
  - DES is a great example
- Flaw in encryption algorithm
  - Every firewall vendor that thought they were smart enough to create their own VPN crypto without outside scrutiny
- Key is compromised in transit
- Key is not changed frequently enough

Hashing

- Process to convert data to an alternate format
  - Usually, but not always, a fixed length string
- Usually works for any data format, of any size
- One way algorithm, cannot reverse to plaintext
- Same data will always generate the same hash
- Slight changes to the data will result in a drastically different hash value

Where do we use hashes?

- Passwords
  - Avoids clear text storage
  - Should include a random seed
- File downloads
  - If the hash matches, you got the right file
- Network packets
  - CRC check is effectively a hash of the frame data

The purpose of a random seed

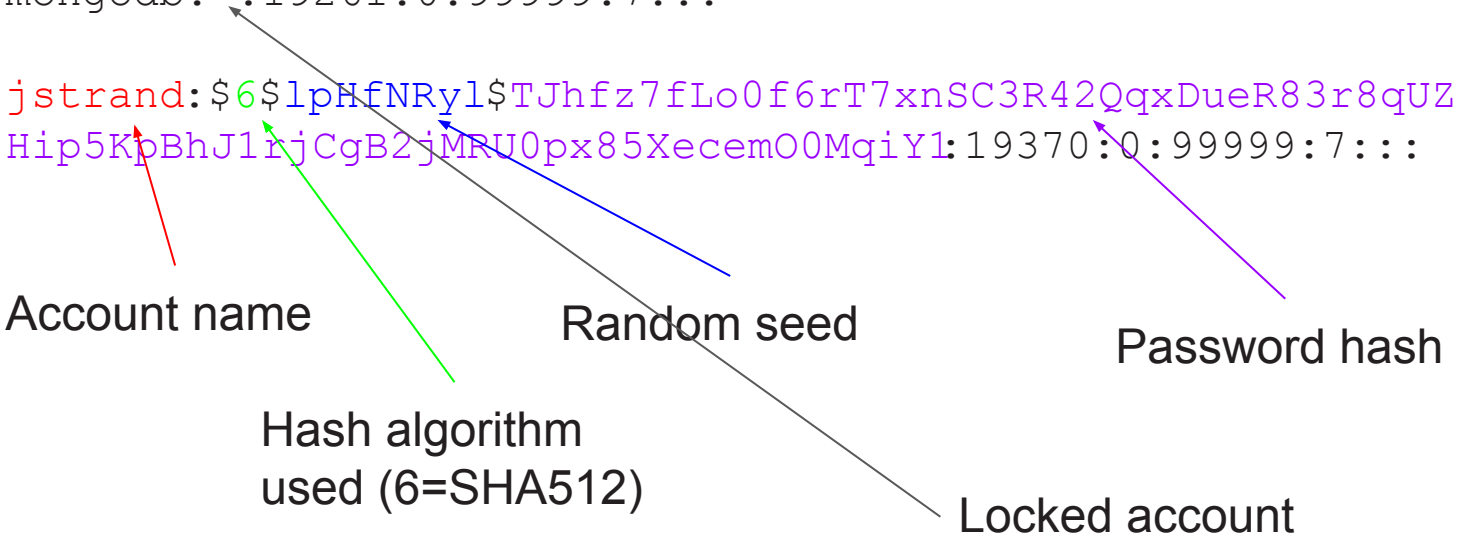
- Hashing is repeatable
  - Same value will always create the same hash
  - This is usually a feature
- Problematic with password hashes
  - 10 people with the same hash have the same password
  - Crack 1 and you know you own them all
- Random seed is semi-random info added to password before hashing
  - Doesn't add strength, just makes the final hash look different when password is the same

Linux shadow file

```
cbrenton:$6$FT5Lri8Q$ilvO0OEtDzBnFzuz5/zL7cUD3RUOwc2rgr1YPH4egJYGgx5P
9bnZBE1dLOwRngPzeKlEorrWJM9c8OBE8ABXU0:18939:0:99999:7:::

mongodb:*:19261:0:99999:7:::

jstrand:$6$lpHfNRyl$TJhfz7fLo0f6rT7xnSC3R42QqxDueR83r8qUZYwfTAJ.7PRhR
Hip5KpBhJ1rjCgB2jMRU0px85XecemO0MqiY1:19370:0:99999:7:::
```



Hash collisions

- Most hash algorithms:
  - Take variable size input, sometimes unlimited
  - Output is usually a fixed length string
- Could have infinite inputs with a fix number of outputs
- Logic dictates there will be collisions
- A "collision" = two datasets create the same hash
- Predictable collisions are a huge problem

Lab time!

- Move to the hashing directory
  - "`cd ~/hashing`" without the double quotes
- There are two files in this directory
- Are they the same files?
  - Size
  - Date/time
  - Hash
- Can you explain any inconsistencies you see?

Hints

- Commands to try:
  - ls -al
  - md5sum *
  - sha1sum *
  - cmp -l image1.jpg image2.jpg
- If you are using SSH
  - Download and view the images
  - Do they look the same?

Answers - they seem similar

```
student@snd:~/hashing$ ls -al
total 664
drwxrwxr-x 2 student student   4096 Apr  1 17:22 .
drwxr-x--- 7 student student   4096 Apr  1 17:23 ..
-rw-r--r-- 1 student student 335104 Apr  1 17:15 image1.jpg
-rw-r--r-- 1 student student 335104 Apr  1 17:15 image2.jpg
student@snd:~/hashing$ md5sum *
253dd04e87492e4fc3471de5e776bc3d  image1.jpg
253dd04e87492e4fc3471de5e776bc3d  image2.jpg
student@snd:~/hashing$
```

But are clearly different

```
student@snd:~/hashing$ md5sum *
253dd04e87492e4fc3471de5e776bc3d  image1.jpg
253dd04e87492e4fc3471de5e776bc3d  image2.jpg
student@snd:~/hashing$ sha256sum *
91e34644af1e6c36166e1a69d915d8ed5dbb43ffd62435e70059bc76a742daa6  image1.jpg
caf110e4aebe1fe7acef6da946a2bac9d51edcd47a987e311599c7c1c92e3abd  image2.jpg
student@snd:~/hashing$ cmp -l image1.jpg image2.jpg | head
   165 331 211
   624 375 370
   625  14 246
   626  16  36
   627 124 265
   628  25 162
   629  43  23
   630 237 200
   631 152 265
   632 156 115
student@snd:~/hashing$
```

Images are not the same

image1.jpg

image2.jpg

What happened?

- These images are similar, but different
  - Date/time is the same
  - Size is the same
- Collision in the hash space when using MD5
- No collision with other hashing algorithms
- File compare (cmp) shows multiple offsets to where different values are stored

Why are collisions bad?

- Two or more values can generate the same hash
- Problematic for passwords
  - System does not check plaintext, just the hash
  - One value per hash, one working password
  - Two or more values per hash, each can access account
- Problematic for verification
  - Really bad if collisions can be predictably manipulated
  - Evil data changes could go undetected

Back to VPNs

- Before the lab we were discussing components of VPNs
- Symmetric key encryption
  - Fast and efficient data privacy
  - Same key to encrypt and decrypt
  - How do we share keys over insecure medium?
- Hashing
  - One way algorithm - cannot reverse
  - Can verify if data has been changed (assuming no collisions)
  - Anyone can generate, so attacker could just generate a new hash after making their evil changes

HMAC

- Hash-based Message Authentication Code
- With a hash, anyone can generate it
- HMAC is a hash plus a symmetric key
  - Limits hash creation and verification to key owners
  - MITM cannot change data and produce a new valid hash without the symmetric key
- Good for authenticating packets as the hash is transmitted with the packet

So where are we at with our VPN?

- Still need to figure out initial authentication
- Still need to figure out setting up a secure channel over an insecure medium
- We can use symmetric key crypto to provide data privacy
- We can use HMAC to authenticate packets
- But we need to figure out the first two before the second two are trustworthy

Next week on Fireside Fridays!

- We still have some components of our VPN to figure out
- We have a number of VPN protocols to discuss
- Next week we'll continue our journey!

Wrap up

- Thank you for attending!
- Certs & video will go out by Monday
- If you have any lingering questions, the Discord channel will remain active
  - Also a good chance to socialize with others in the class
  - Have other tips and tricks? Please share with others!
- **Thank you** for sharing your time with us!