Fireside Fridays

Anatomy of a VPN
Part 2 of 3

Thanks to our sponsors!

ACTIVE COUNTERMEASURES

Antisyphon Training

Special thanks to the THC mods

- A huge shout out to all of the threat hunter community moderators
- They invest a huge amount of time into answering questions and dealing with spammers
- This would not be such a cool, welcoming place without their constant support
- Please take a moment to thank them for all that they do

Lab requirements for this section

- Linux system running an SSH server
- Sudo access to the SSH server
- Admin access to a Windows system

Where did we leave off last week?

- 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

Asymmetrical encryption

- Sometimes called public key cryptography
- Uses two mathematically related, but unequal keys
  - One key to encrypt - public key
  - One key to decrypt - private key
- Set up a secure channel over an insecure medium
- Popular for securing email
- Also leveraged in VPNs

Asymmetric problems

- Still susceptible to initial man in the middle attacks
  - What if we try to exchange public keys but an adversary replaces each with their public key?
  - They can view and modify all data that is exchanged
  - With the tech we've discussed so far, this would not be detected
- High CPU utilization
  - Would be a huge performance to streaming data
  - Better for store and forward messaging like email

Asymmetric bonus - verification

- Can be used for generating a "signature" hash
- Hash created by the private key
- Public key can verify the hash of the private key
- Public key can only verify hash, it cannot create one
- If you have my public key, you can verify that:
  - The message was encrypted with my private key
  - The message has not been modified in transit
  - Pass/fail test, meaning you can't tell why it failed

Secure comms with just Asymmetric

- We exchange public keys
- I encrypt data with your public key
- I hash ciphertext with my private key
- You check the hash with my public key
- You decrypt the data using your private key
- But this is really CPU intensive

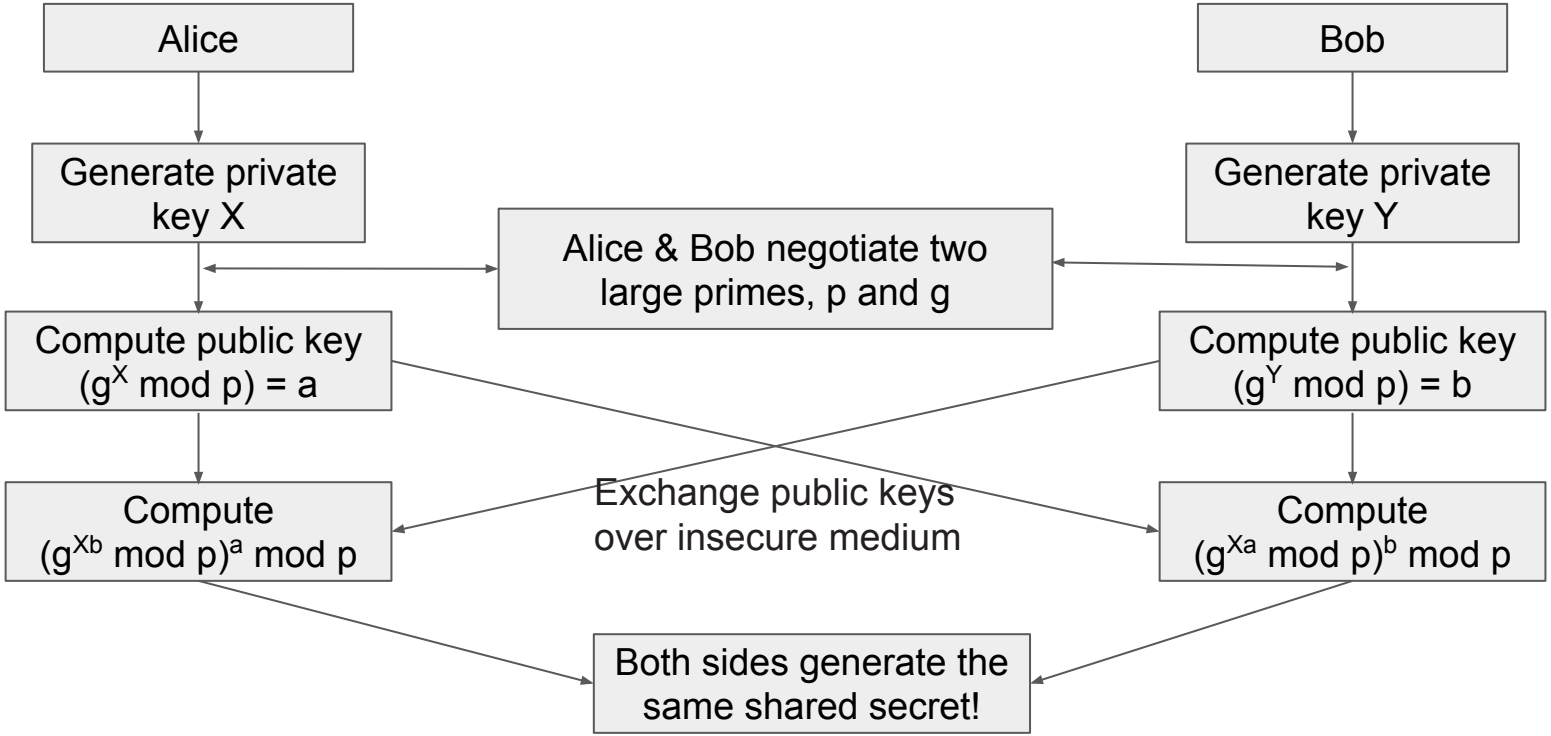
Create a secure channel for symmetric

- You send me your public key
- I generate a secret key
- I encrypt the symmetric key using your public key
- You decrypt the symmetric key using your private key
- We switch over to symmetric key encryption
  - Symmetric key used to encrypt and decrypt
  - More efficient and processor friendly than asymmetric
- Creates predefined roles
- Transfer of symmetric key is not authenticated unless HMAC used

Asymmetric examples

- RSA
  - Based on the difficulty of factoring very large numbers back into the two original prime numbers used to generate the value
  - Generate a public key to encrypt, private key to decrypt
- Elliptic curve
  - Similar to RSA but smaller pimes can be used
  - More efficient and less of a CPU hit than RSA
- Diffie-Hellman
  - Negotiate a shared secret key to be used with symmetric algorithm
  - Does not encrypt, just a key that can be used with symmetric

Diffie-Hellman in action



What we have so far

- Asymmetric encryption - setup a secure channel
- Symmetrical encryption - Efficient data privacy
- HMAC - packet level authentication
- Still missing initial authentication
- How do I know it's really your public key?
- Without initial auth, the dominoes fall…

As simple as a password

- Can use a password or shared secret
- Many proprietary VPNs leverage this
- Simple implementation but problematic
- Everyone usually knows the shared secret
- PITA to cycle to a new value
- How do we protect this secret in transit without encryption?
- This makes it easier to crack

SSH initial auth via private key hash

First login:

```
C:\Users\cbren>ssh cbrenton2@161.35.113.192
The authenticity of host '161.35.113.192 (161.35.113.192)' can't be established.
ECDSA key fingerprint is SHA256:3pIpQ2eFFzvBt23Jy+XahfaxZXQe5PYct2couWeCkc4.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Run from console:

```
root@cb-lab:~# for fp in /etc/ssh/ssh_host_*_key; do ssh-keygen -l -f "$fp"; done
1024 SHA256:JHsFdU5WQ1VGPXJhWmM93sEPen5xakAWb1nYNMcxQi4 root@cd-lab (DSA)
256 SHA256:3pIpQ2eFFzvBt23Jy+XahfaxZXQe5PYct2couWeCkc4 root@cd-lab (ECDSA)
256 SHA256:jxxhpRebcx/ojZVXUkJ5rvUY55x0S+WKhZrAP+aV/J0 root@cd-lab (ED25519)
2048 SHA256:Da0ZzxpQghiSz95+pkFVhryAtUvsNBtIL4SrUJKa2yA root@cd-lab (RSA)
```

Try this yourself!

```
echo 'for pubkey in /etc/ssh/*.pub; do ssh-keygen -lf ${pubkey} ; done' > key

chmod +x key

sudo ./key
```

```
cbrenton@cbrenton-demo:~$
cbrenton@cbrenton-demo:~$ echo 'for pubkey in /etc/ssh/*.pub; do ssh-keygen -lf ${pubkey} ; done' > key
cbrenton@cbrenton-demo:~$ chmod +x key
cbrenton@cbrenton-demo:~$ sudo ./key
256 SHA256:YFxGt2cVVQCcc6ArudxRNma/MdwQdENjf9JARZ+KzRU root@cbrenton-demo (ECDSA)
256 SHA256:aVpYdp2ZrPB+2bXKWla8H/SDOZ95MXonJyoCeuzCn74 root@cbrenton-demo (ED25519)
3072 SHA256:+6tphVagglc72i1VkTXxdHbwPUGZNNhbNWBZJaSFnl4 root@cbrenton-demo (RSA)
cbrenton@cbrenton-demo:~$ _
```

Weaknesses of this initial auth

- Requires another human
  - Someone to check the public key hash
  - Must already have system access
- This makes it cumbersome and impractical
- Come on, be honest, have you ever actually used this feature before?
- We need initial auth that is fully automated by default

What is a digital certificate?

- Usually associated with HTTPS servers
  - But can secure any TCP application (including SSH)
  - Even apps that tunnels IP packets
- Clients can be authenticated via digital certificates as well
- Method of distributing public keys
- Method of validating that the public key you received is in fact associated with the server

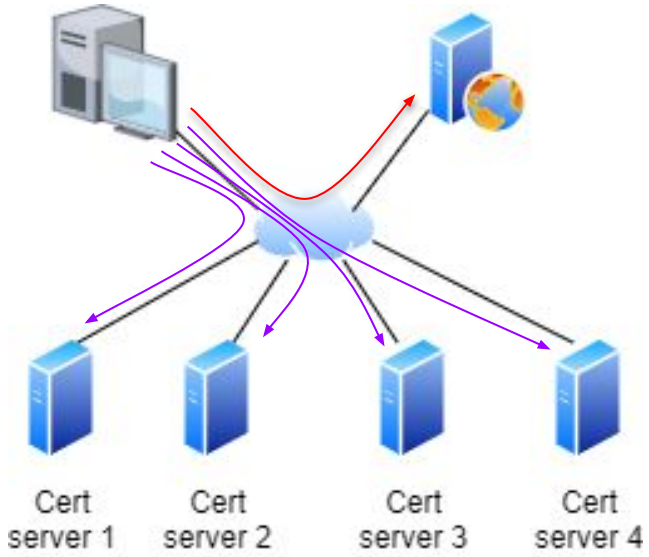
Anatomy of a digital certificate

- Imagine a plaque on the wall
- On that plaque would be:
  - What system was the cert issued to
  - The organization responsible for the system
  - Certificate authority who issued this digital certificate
  - Issue and expiration dates
  - Public key of the identified server
  - The hash created by the private key of the issuing server
  - Other fields we don't need to worry about just yet ;-)

How to validate a digital certificate

- The cert has a hash created with the private key of the issuing server
- Get the issuing server's public key and verify the hash
- Check is pass/fail - If it fails you can't tell why
- If it passes, you know the cert can be trusted
- But wait, I got the issuing server's public key to check the hash. How do I know it's valid and was not intercepted in an initial MiTM attack???
- Let's go down the rabbit hole…

Digital cert check



Get web server certificate issued by server1
Get server1 digital cert to check web cert
Get server2 digital cert to check server1 cert
Get server3 digital cert to check server2 cert
Get server4 digital cert to check server3 cert
Trust server4 because it tells you to

Implementation challenges

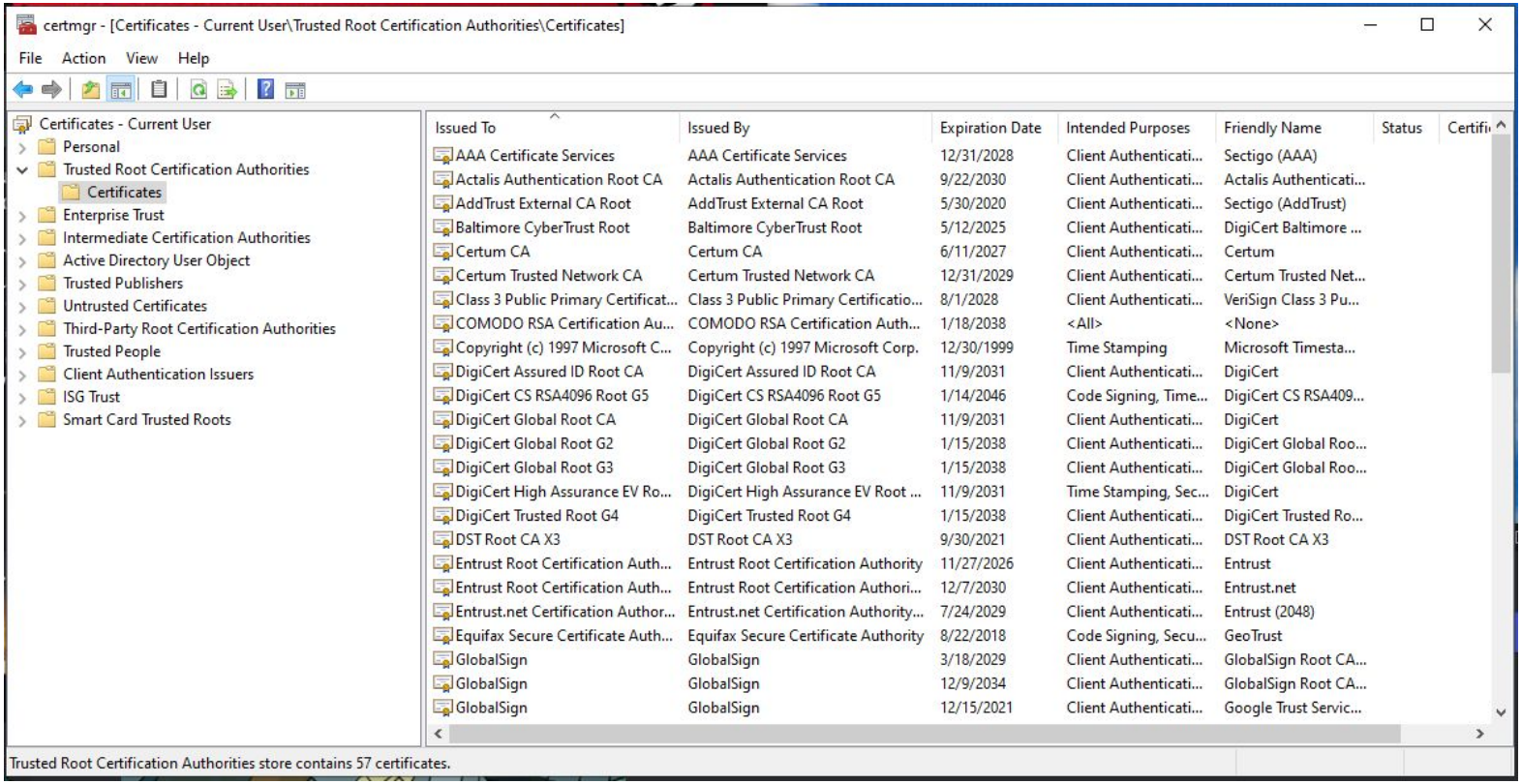
- Most system don't check the chain
  - Cert for for first certificate server saved locally
  - Usually a system check for revocation only
  - May not check to see if it's a recognized cert authority
- At the bottom of the rabbit hole is a server with a self signed certificate
  - Trust it because it says so
  - No math to verify, just warm fuzzy

Self signed certificates

- You can generate your own digital certificates
- Even though you are not a recognize cert authority
- Enables TLS communications
- But initial authentication untrusted since you are not a recognized cert authority

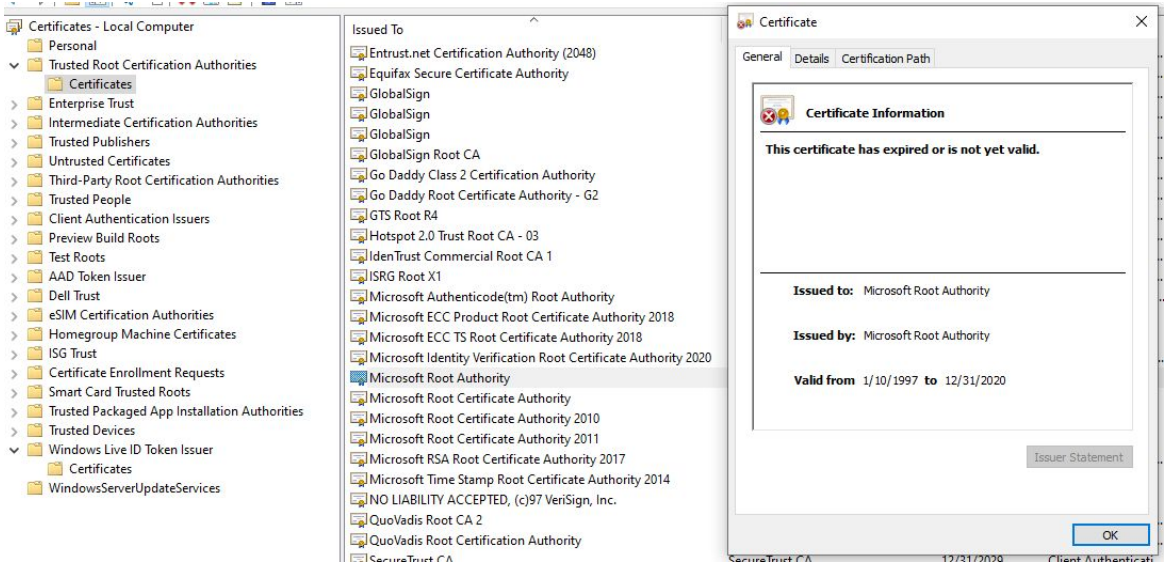
- What makes someone a recognized cert authority?
- Cert for signing server added to OS certificate chain

View root certs with certmgr.msc



Wait…

- Wouldn't this technically be a self signed cert?
- "No" because it's pre-installed in the OS :-)



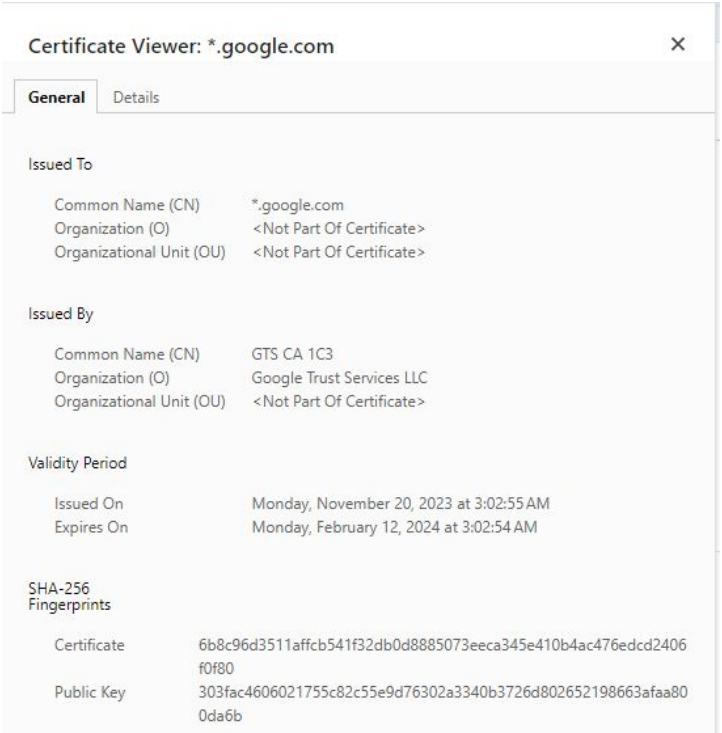
View root certs on linux

```
awk -v cmd='openssl x509 -noout -subject' '
    /BEGIN/{close(cmd)};{print | cmd}' < /etc/ssl/certs/ca-certificates.crt | less
```

```
subject=CN = ACCVRAIZ1, OU = PKIACCV, O = ACCV, C = ES
subject=C = ES, O = FNMT-RCM, OU = AC RAIZ FNMT-RCM
subject=C = IT, L = Milan, O = Actalis S.p.A./03358520967, CN =
Actalis Authentication Root CA
subject=C = US, O = AffirmTrust, CN = AffirmTrust Commercial
subject=C = US, O = AffirmTrust, CN = AffirmTrust Networking
subject=C = US, O = AffirmTrust, CN = AffirmTrust Premium
subject=C = US, O = AffirmTrust, CN = AffirmTrust Premium ECC
subject=C = US, O = Amazon, CN = Amazon Root CA 1
subject=C = US, O = Amazon, CN = Amazon Root CA 2
subject=C = US, O = Amazon, CN = Amazon Root CA 3
```

View current cert in Chrome

Click the lock icon
Click "Connection is secure" (if it appears)
Click "Certificate is valid" or "Certificate details"



Pulling together a VPN

- Digital certificate for initial authentication
- Asymmetric encryption to setup a secure channel
- Symmetric encryption for data privacy
- HMAC to authenticate each packet

Common VPNs

- SecureSHell (SSH)
- IPSec
- TLS
- We'll do a brief overview of each… next week

Next week on Fireside Fridays!

- We'll talk about VPN implementations
- No setup required, it will just be lecture

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!