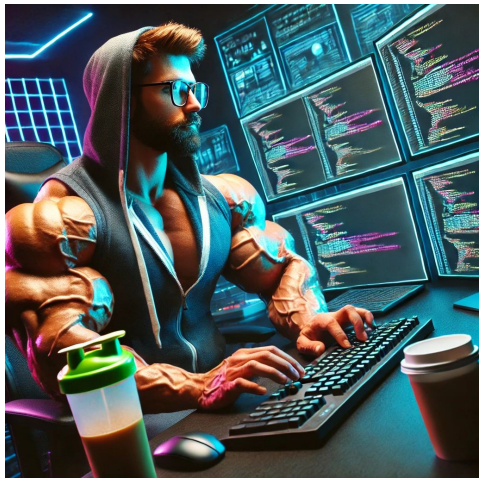
Covert Channel Presentation

Max, Carter, Charlie

Why Use File Size?

- Seemingly just a bunch of empty (or almost empty) files
  - Hard to detect unless looking for it
- Simple to use
- Perfect for smaller messages
- Works with anything that can be converted to binary



How It Works

Sender Tool

Convert Message to Binary

File Creation and Storage Based on Binary.

```
# Sender tool
def covert_channel_send_size(message, directory):
    binary_message = ''.join(format(ord(c), '08b') for c in message)

    for bit in binary_message:
        filename = os.path.join(directory, f"file_{time.time()}.txt")
        with open(filename, 'w') as f:
            if bit == '1':
                f.write('A')  # Append a small amount of data for '1'
            else:
                f.write('')  # Leave blank for '0'
        time.sleep(0.1)
```

Receiver Tool

Collect Files

Reconstruct Binary

Decode Binary

```python
# Receiver tool
def covert_channel_receive_size(directory):
    files = sorted(os.listdir(directory))  # Sort files by created time to ensure message is in order
    message_binary = []

    for filename in files:
        file_path = os.path.join(directory, filename)
        size = os.path.getsize(file_path)
        if size > 0:  # If size is greater than 0, it's a '1'
            message_binary.append('1')
        else: # It's a '0'
            message_binary.append('0')

    # Convert binary back to string to reveal message
    message = ''.join(message_binary)
    decoded_message = ''.join(chr(int(message[i:i+8], 2)) for i in range(0, len(message), 8))
    print(f"Decoded message: {decoded_message}")
```

Demonstration

Advantages

- Simple to understand source code
- Ease of access to messages if you are sender or receiver
- Small messages/conversations are quick and easy
- Works on different systems due to the python conversion tool, can be done on command line for ease of access

Security Implications

- Only those who need to have access to the system can access the source code
- A person who understands network traffic and is willing to watch for messages can try and decode the messages, meaning a secure network is strongly advised but not required if a person isn't looking for your messages
- It is implied that each the sender and the receiver are trusted and sending from a secure machine/location

Conclusions

- The overall architecture of the program is easy to understand and use
- Security of the receivers and senders machines is implied and important before use of the program, as well as a secure network
- Python takes care of the binary conversions for ease of understanding, simple to read and use
- Best used when limited people know about what is going on, use for large groups of people is less helpful
- There are only 2 ways a file can be changed, by having data or lack of data, which may become problematic or predictable for large messages or conversations if an adversary gains access to the network