

Decentralized Hacker Net

Eijah

2.0.0
July 1st, 2018

Who am I?

- Founder
- Programmer
- Hacker
- CEO

PROMETHER

THE END OF SURVEILLANCE

demonsaw





Goals

- Create a decentralized hacker network
- Perform analysis, design, implementation
- Protect confidentiality and authenticity of data
- Remain anonymous and unseen

Prologue

- What does it mean "to be secure"?
- Confidentiality, Integrity, and Availability
- Authenticity and Non-repudiation
- Free Software
  - Open Source
  - Open Infrastructure (Walled Gardens)
  - Runtime verification/Sandboxing
- Evade & Subvert: Our new best friends
  - Escape or avoid, especially by cleverness or trickery
  - Undermining of the power and authority of an established system
- Hiding in plain sight



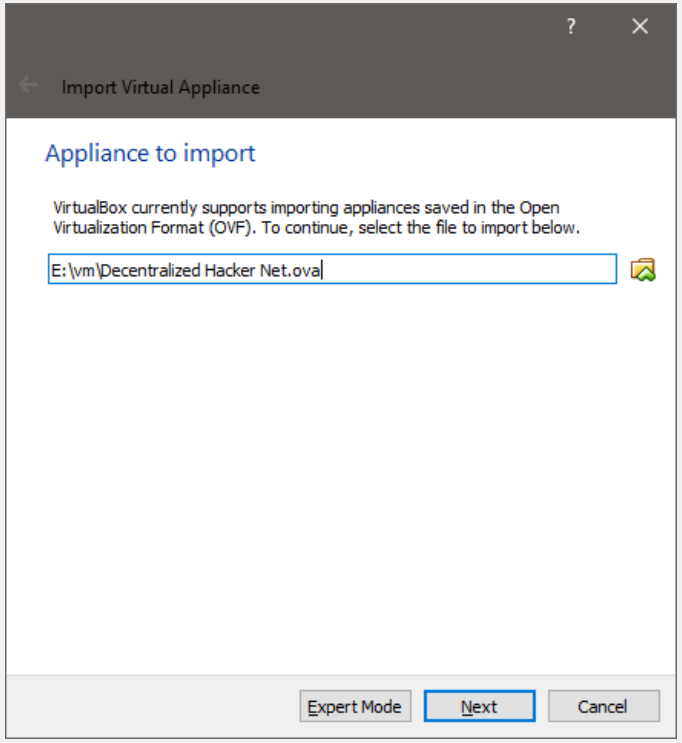
Configuration

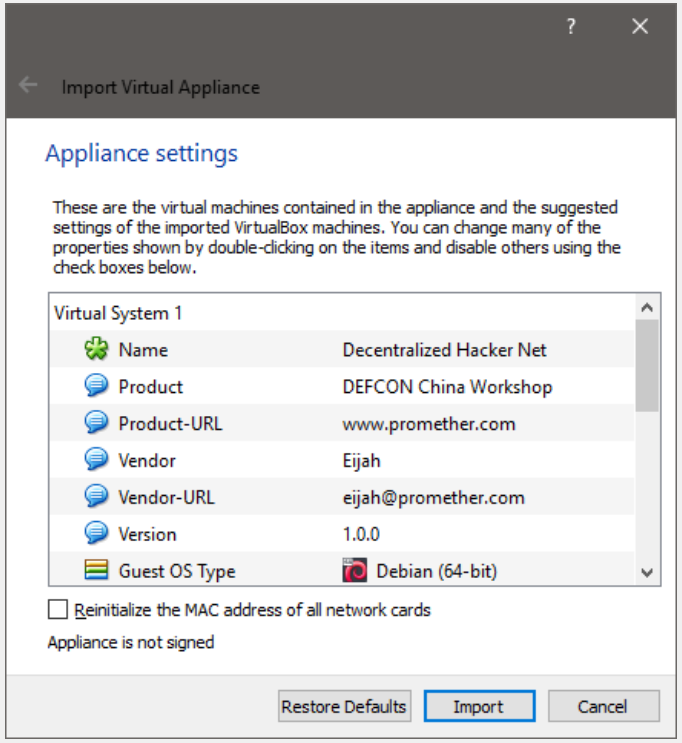
Prerequisites

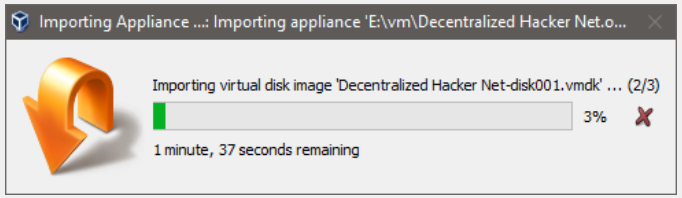
- Laptop
- Virtual Box (www.virtualbox.org) or any other virtualization software that supports *.ova files
- Prior experience in at least 1 programming language

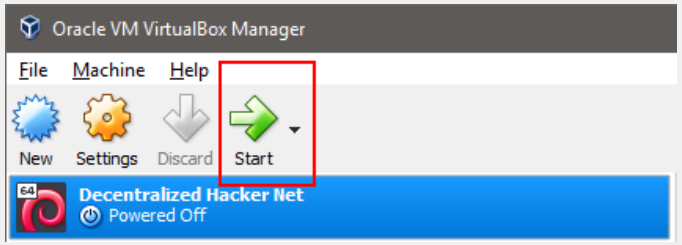
Virtual Machine

- Open Oracle VM VirtualBox Manager (www.virtualbox.org)
- Click on “File” → “Import Appliance” in the toolbar









Building the Code

- Start the Virtual Machine
- Linux
  - root
  - password
- Wireless
  - Decentralized Hacker Net
  - Promether!
- Navigate to /defcon
- Run ./configure
- Run make
- Navigate to /defcon/_lib/
- Verify that the workshop libraries were created



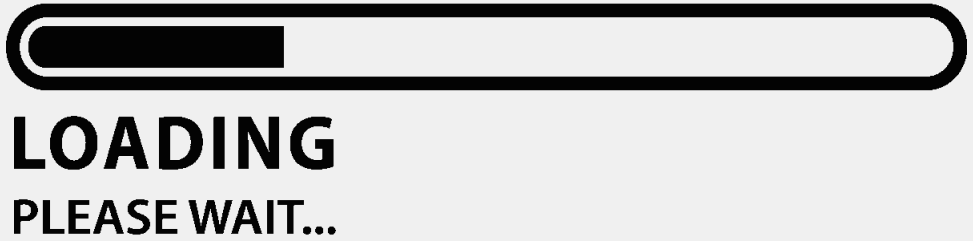
Lessons

Overview

- Lesson 1 – Intro
- Lesson 2 – Config
- Lesson 3 – Crypto
- Lesson 4 – Architecture
- Lesson 5 – Network
- Lesson 6 – Data
- Lesson 7 – Protocol
- Lesson 8 – Advanced

Lesson 1 – Intro

- C++
  - main.cpp
- Boost
  - https://www.boost.org
- Crypto++
  - https://www.cryptopp.com
- Message Pack
  - https://msgpack.org
- Promether
  - https://www.promether.com



Lesson 1 – Intro

- Navigate to /defcon/lesson_1
- Open main.cpp and modify the code as necessary
- Run ./configure
- Run make
- Navigate to /defcon/lesson_1/_bin
- Run ./lesson_1



Lesson 2 – Config

- Parse config file
- Tom's Obvious, Minimal Language (TOML)
  - Read in strings and integers
  - https://github.com/toml-lang/toml
- If unspecified, set default IP address and port number with Boost Asynchronous IO (ASIO)
- Print name, IP address, and port number to the console
- Open lesson_2.toml and modify config parameters



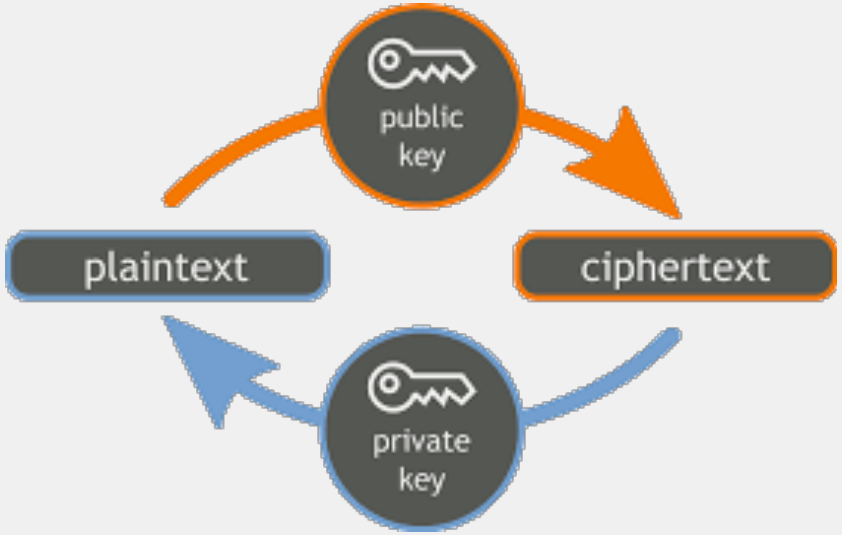
Lesson 2 – Config

- Navigate to /defcon/lesson_2
- Open main.cpp and modify the code as necessary
- Run ./configure
- Run make
- Navigate to /defcon/lesson_2/_bin
- Run ./lesson_2



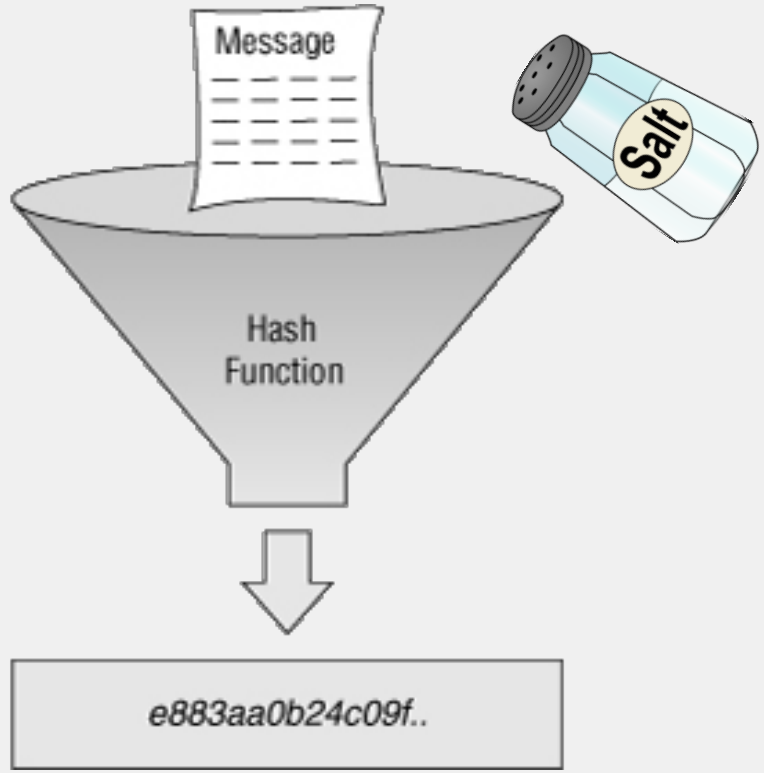
Lesson 3 – Crypto

- Rivest–Shamir–Adleman (RSA) Encryption
  - https://en.wikipedia.org/wiki/RSA_(cryptosystem)
  - https://www.cryptopp.com/wiki/RSA_Cryptography
- Elliptic Curve Integrated Encryption Scheme (ECIES)
  - https://en.wikipedia.org/wiki/Integrated_Encryption_Scheme
  - https://www.cryptopp.com/wiki/ECIES
- Create RSA private/public key pair
  - 1024, 2048, 3072, 4096 bits
- Create ECIES private/public key pair
  - secp256r1, secp384r1, secp521r1
  - secp256k1 (Koblitz curve)



Lesson 3 – Crypto

- Save private and public keys to a file
- Load private and public keys from a file
  - dumpasn1
- Encrypt data with the public key
- Decrypt data with the private key
- Hash the public key to create a unique identifier
  - Message Digest (MD-5)
  - Secure Hash Algorithm (SHA-1, SHA-3)
  - Cyclic Redundancy Check (CRC)
  - Adler-32
  - xxHash



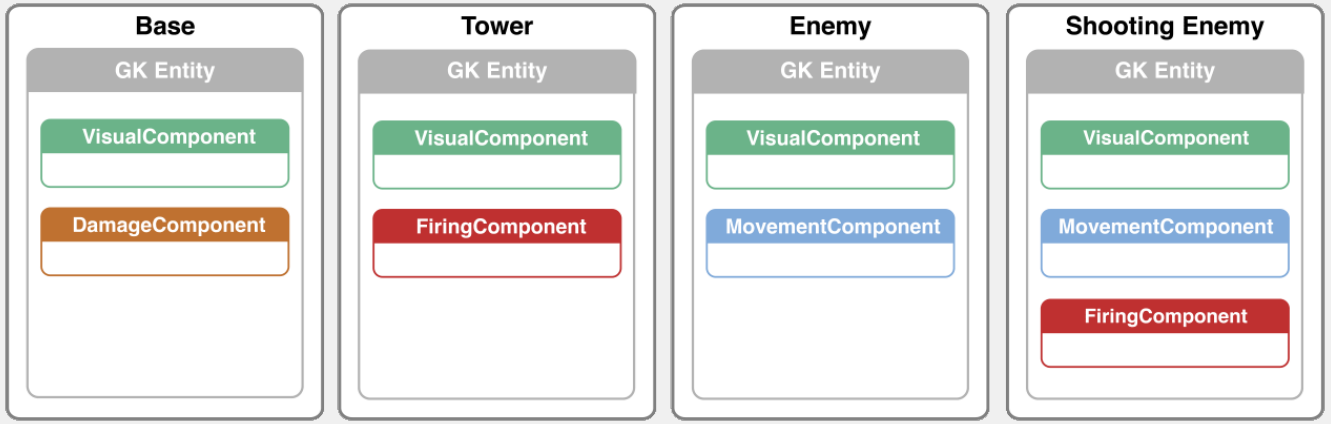
Lesson 3 – Crypto

- Navigate to /defcon/lesson_3
- Open main.cpp and modify the code as necessary
- Run ./configure
- Run make
- Navigate to /defcon/lesson_3/_bin
- Run ./lesson_3



Lesson 4 – Architecture

- Entity Component System (ECS)
  - Architecture Pattern mostly used in game programming
  - Composition over inheritance allows greater flexibility
  - https://en.wikipedia.org/wiki/Entity%E2%80%93component%E2%80%93system
- Boost ASIO
  - https://www.boost.org/doc/libs/1_67_0/doc/html/boost_asio.html
  - Asynchronous Callbacks
  - Multi-threaded



Lesson 4 – Architecture

- Entities, components, and data
- Multi-threaded, asynchronous state machines
- Main window and input processing loop

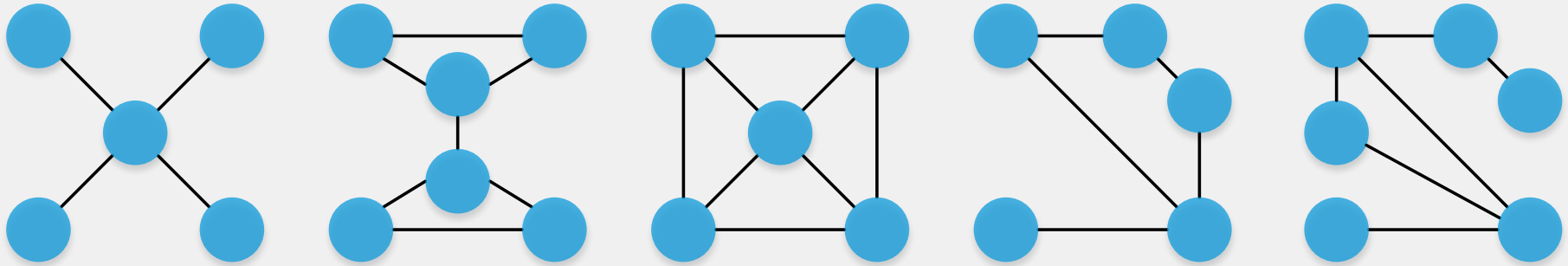
Lesson 4 – Architecture

- Navigate to /defcon/lesson_4
- Open main.cpp and modify the code as necessary
- Run ./configure
- Run make
- Navigate to /defcon/lesson_4/_bin
- Run ./lesson_4



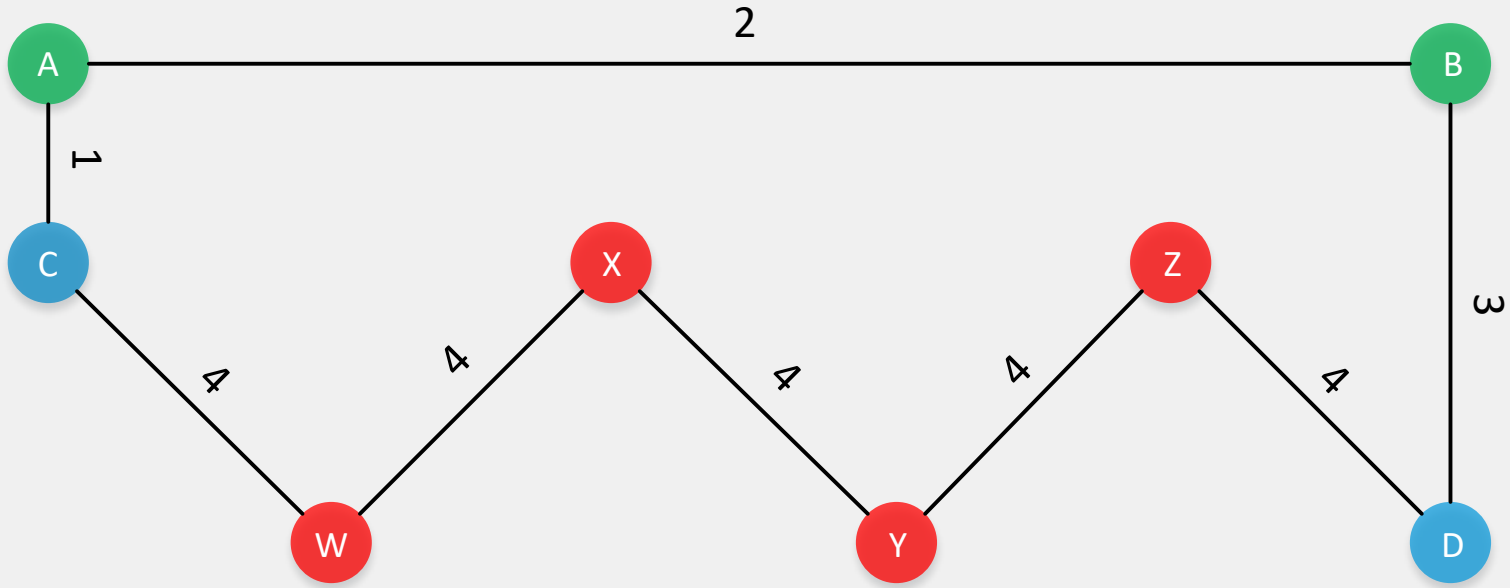
Lesson 5 – Network

- The Internet is already decentralized
- There are many types of network topologies
- Nodes are endpoints in the network that produce or consume packets
- The pros and cons of security, anonymity, and decentralization.
- Does network equality and balance matter?
- What are the features that we want our hacker network to support?
  - Routing, randomization, hopping, padding, sharding, reordering, spoofing?



Lesson 5 – Network

- Listen for incoming connection on a specific (or random) port
- Connect to one (or more) nodes on our network
- Call a handshake function to announce our presence
- Encrypt and decrypt via ECIES (or RSA)



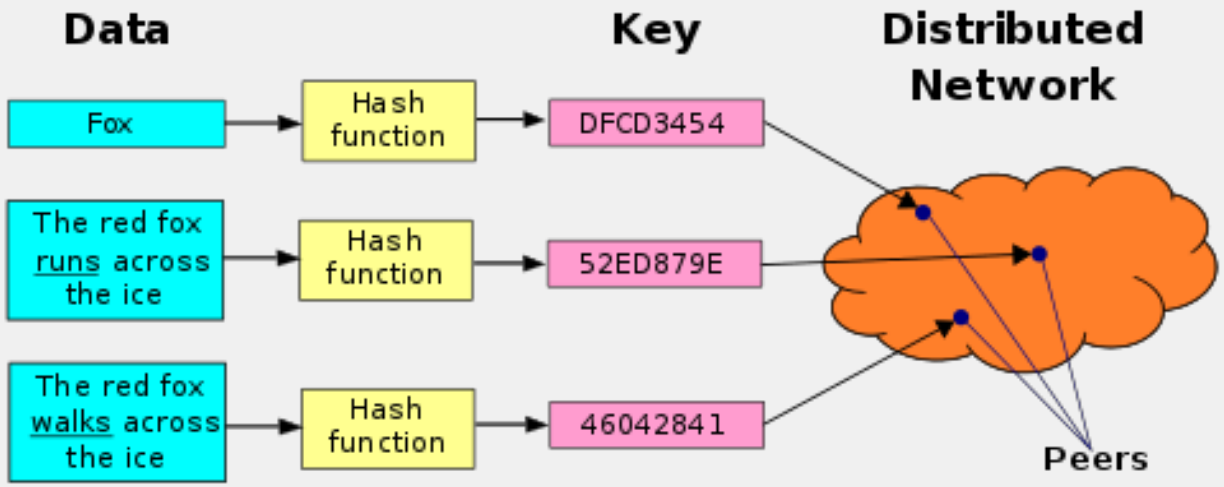
Lesson 5 – Network

- Navigate to /defcon/lesson_5
- Open main.cpp and modify the code as necessary
- Run ./configure
- Run make
- Navigate to /defcon/lesson_5/_bin
- Run ./lesson_5



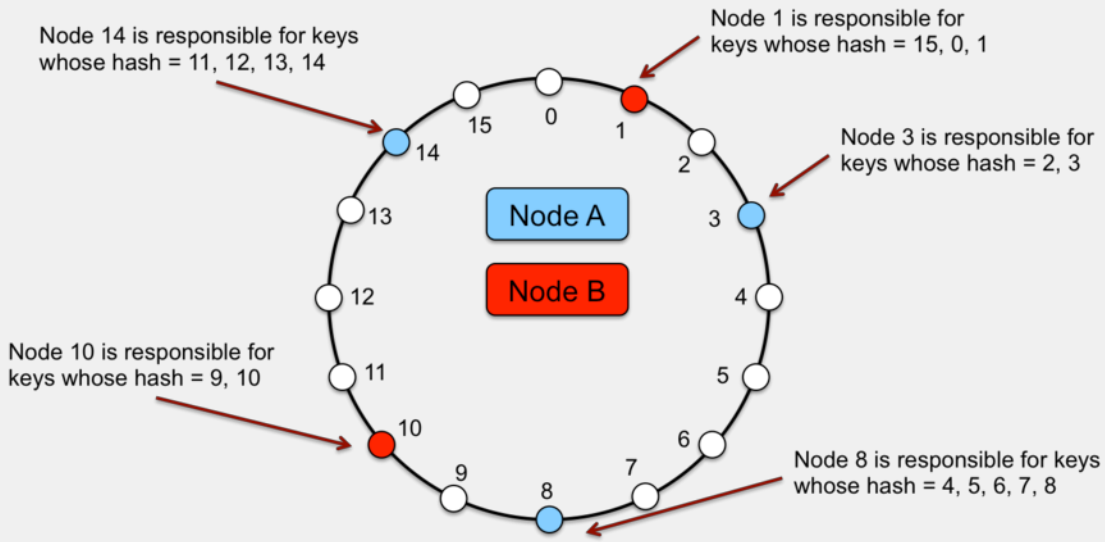
Lesson 6 – Data

- Distributed Hash Table (DHT)
  - Distributed storage
  - Support complex architectures
  - Consistent, Rendezvous hashing
  - Add, remove rebalancing
- Overlay Network
  - Node identity (via public key)
  - Data identity (via content)
- Features
  - Autonomy, decentralization
  - Fault tolerance
  - Scalability



Lesson 6 – Data

- Add a user to the network
  - Name
  - Public Key
  - IP Address
  - Port
- Remove a user from the network



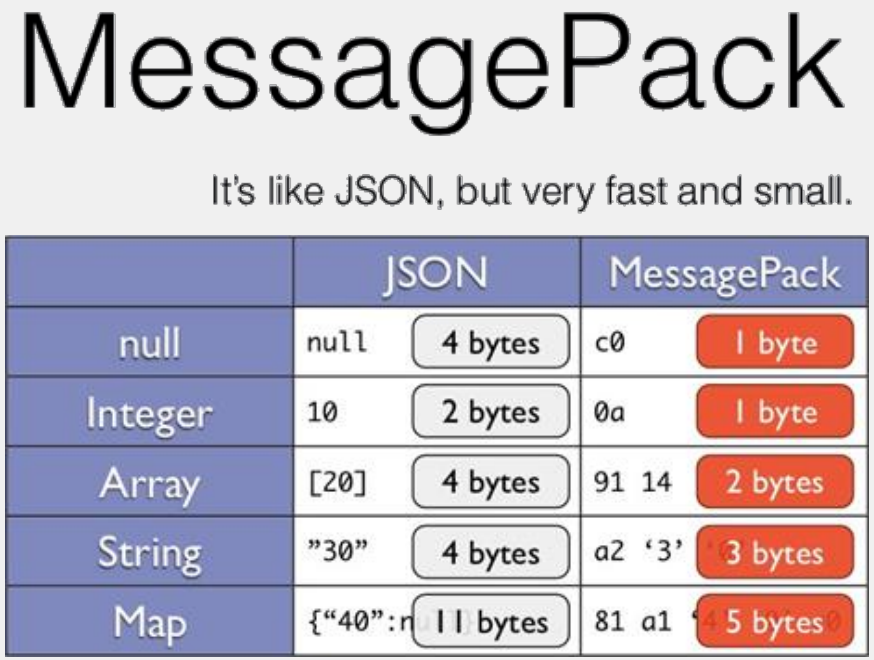
Lesson 6 – Data

- Navigate to /defcon/lesson_6
- Open main.cpp and modify the code as necessary
- Run ./configure
- Run make
- Navigate to /defcon/lesson_6/_bin
- Run ./lesson_6



Lesson 7 – Protocols

- What types of messages?
  - Message Pack (binary, packed format)
- Protect against as many attack vectors as possible
  - ECIES Encryption
- We want speed and flexibility
  - User Datagram Protocol (UDP)
  - Routed transmissions
  - Peer-to-peer (P2P)



Lesson 7 – Protocols

- Chat
  - Individual
  - Group
- Search by name



Lesson 7 – Protocols

- Navigate to /defcon/lesson_7
- Open main.cpp and modify the code as necessary
- Run ./configure
- Run make
- Navigate to /defcon/lesson_7/_bin
- Run ./lesson_7



Lesson 8 – Advanced

- Let’s take a look at the server code
  - Keep in mind that the server code could easily be integrated with the client
  - We’ve only separated them to keep things as simple as possible
  - There is value to keeping the Distributed Hash Table (DHT) constrained to as few nodes as possible (e.g. Master Node architecture)
- What other features do you think we could add to the code?
  - File sharing?
  - File sync?
  - VOIP?



Lesson 8 – Features

- Navigate to /defcon/server
- Open main.cpp and modify the code as necessary
- Run ./configure
- Run make
- Navigate to /defcon/server/_bin
- Run ./server



Summary

- Books
- Online resources
- Open source communities

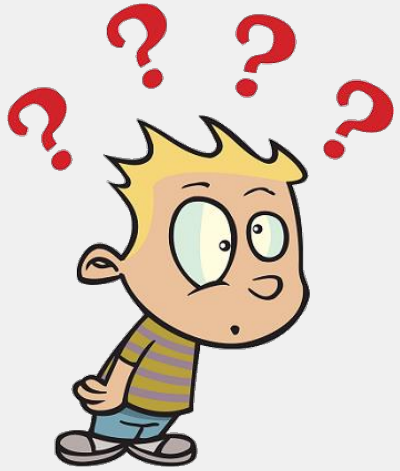


Questions?

www.promether.com

eijah@promether.com

@demon_saw



Eijah