# Stretch Project

BLUE SENTINEL PERSONAL IDS

MOLLER.R

# Analysis

## Problem definition

I intend to develop a desktop GUI-based Intrusion detection system (IDS) which provides features to detect malicious content on the host (HIDS) as well as on the network (NIDS).

This system will focus more on the detection of malicious content rather than prevention but will provide notifications for users and possible actions to deal with malware most typically to delete/remove the found malware.

In order to detect malicious traffic, I will be using these three main approaches:

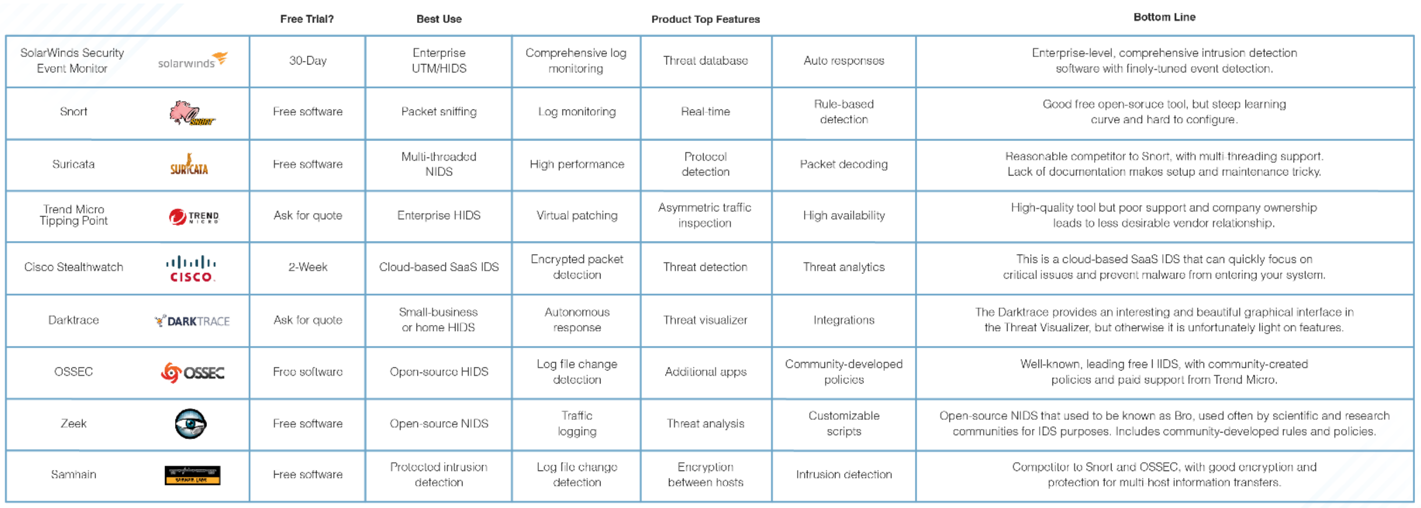
- Signature-based detection
- Policy-based detection
- Anomaly-based detection

*References:*

- *https://en.wikipedia.org/wiki/Intrusion_detection_system*
- *CompTIA, Network+, N10-007 – Page. 456*

## Research

### Current observations



| | | Free Trial? | Best Use | Product Top Features | | | Bottom Line |
|---|---|---|---|---|---|---|---|
| SolarWinds Security Event Monitor | solarwinds | 30-Day | Enterprise UTM/HIDS | Comprehensive log monitoring | Threat database | Auto responses | Enterprise-level, comprehensive intrusion detection software with finely-tuned event detection. |
| Snort | | Free software | Packet sniffing | Log monitoring | Real-time | Rule-based detection | Good free open-soruce tool, but steep learning curve and hard to configure. |
| Suricata | SURICATA | Free software | Multi-threaded NIDS | High performance | Protocol detection | Packet decoding | Reasonable competitor to Snort, with multi-threading support. Lack of documentation makes setup and maintenance tricky. |
| Trend Micro Tipping Point | TREND MICRO | Ask for quote | Enterprise HIDS | Virtual patching | Asymmetric traffic inspection | High availability | High-quality tool but poor support and company ownership leads to less desirable vendor relationship. |
| Cisco Stealthwatch | CISCO | 2-Week | Cloud-based SaaS IDS | Encrypted packet detection | Threat detection | Threat analytics | This is a cloud-based SaaS IDS that can quickly focus on critical issues and prevent malware from entering your system. |
| Darktrace | DARKTRACE | Ask for quote | Small-business or home HIDS | Autonomous response | Threat visualizer | Integrations | The Darktrace provides an interesting and beautiful graphical interface in the Threat Visualizer, but otherwise it is unfortunately light on features. |
| OSSEC | OSSEC | Free software | Open-source HIDS | Log file change detection | Additional apps | Community-developed policies | Well-known, leading free HIDS, with community-created policies and paid support from Trend Micro. |
| Zeek | | Free software | Open-source NIDS | Traffic logging | Threat analysis | Customizable scripts | Open-source NIDS that used to be known as Bro, used often by scientific and research communities for IDS purposes. Includes community-developed rules and policies. |
| Samhain | | Free software | Protected intrusion detection | Log file change detection | Encryption between hosts | Intrusion detection | Competitor to Snort and OSSEC, with good encryption and protection for multi-host information transfers. |

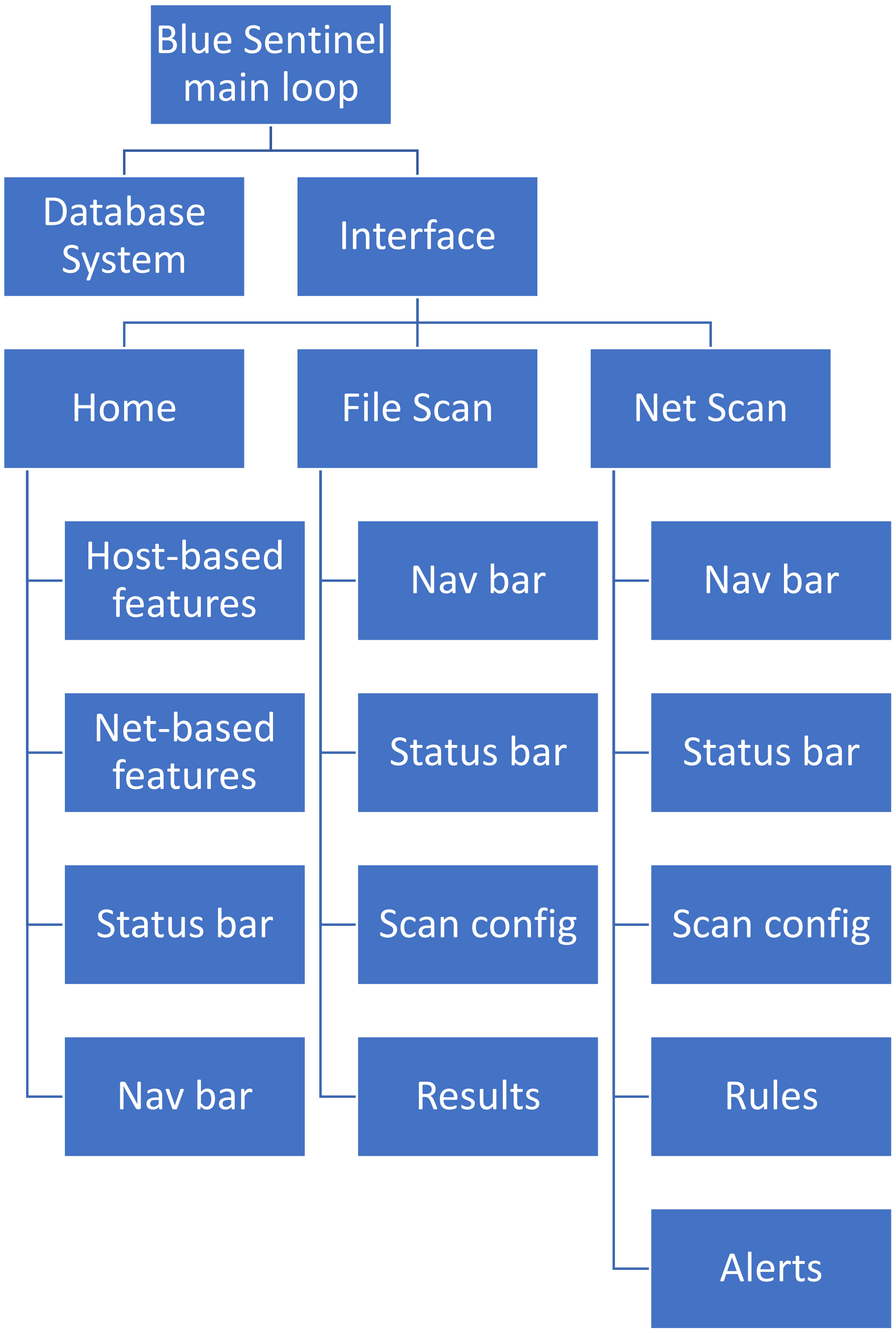
*https://www.dnsstuff.com/wp-content/uploads/2019/10/top-ids-software.png*

## Requirements Specification / Features I intend to implement

| **FEATURE** | **DESCRIPTION** | **PRIORITY** |
|---|---|---|
| **Minimalistic and responsive UI** | The GUI for the application will have a simple minimalistic design with a consistent design scheme.<br>The GUI will also be responsive and not hang during operations. | HIGH |
| **Network intrusion detection system** | Network scan which will inspect packets against a set of rules, alerting user if they are broken. | HIGH |
| **Host intrusion detection system** | File system scan for malicious files by checking hashes of files on system against a malicious has database, | HIGH |
| **Notifications/alert system** | A mechanism for alerting user if any rules are broken or malicious files are found. | HIGH |
| **Report generation** | Formatting results in an exportable medium for user convenience. | MEDIUM |

## Design

### Top down modular design – system diagram



## Technologies

### Programming language

For the development of this project I intend to use Python for the backend and frontend.

The reason for doing so is that Python is lightweight has a variety of built-in libraries which will make the development of this project faster and easier.

Furthermore, our application is not reliant on speed or uses a whole lot of resources and so a lower level language with more control over memory management is not necessary.

### Operating System

I will specifically develop this program for Windows, I am doing this due to time constraints as I would have to put more resources into testing and implementing features differently if it were to be cross-platform.

### OOP approach

In order to create my program, I will be using an object orientated approach. Object orientated programming involves creating solutions using objects that interact with each other. In order to create objects a template known as a class is used. Classes have attributes (variables associated with an object) and methods (functions that the object carries out). The act of creating an object using a class is known as instantiation.
I chose to use an object orientated approach over a procedural approach due to 3 main features: Inheritance, Encapsulation and Polymorphism. Inheritance allows derived classes to inherit the attributes and methods of super classes as well as having their own attributes and methods; furthermore, instances of the same class can share variables between them. Encapsulation prevents attributes of a class from being manipulated by external methods and classes, therefore upholding the integrity of the system; attributes can only be altered using methods of their respective class. Finally, OOP allows objects to be process differently depending on their data type or class through the feature of polymorphism.

### Agile methodology

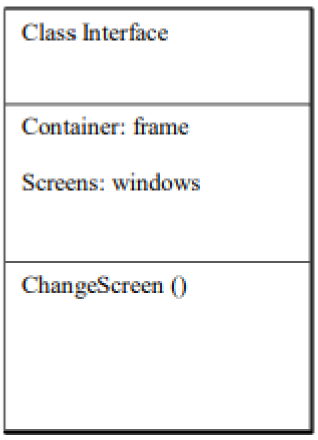
During the development phase of my project, I have decided to switch from the waterfall methodology to an agile approach. Using an agile approach in the development phase will allow me to perform multiple iterations of testing and evaluating on a piece of code/

prototype in the hopes of improving my program to follow and meet the requirements that

were stated in the analysis section which will allow me to make any modifications/improvements before I move on to the next prototype.

## Interface

The interface module will act as a control unit which will link all the individual screens together. The interface module is essential as it will allow each screen and their respective functions to interact with other screens (Share variables and methods, change screens). The interface will consist of 3 screens: Home, FileScan, NetScan. Using OOP the interface module can be modelled as a class.



Attributes:

- Container – This is a frame inside a window which will be used to hold the screens.
- Screens – These are window objects created by their respective classes (Home class, Profile class…)

Methods:

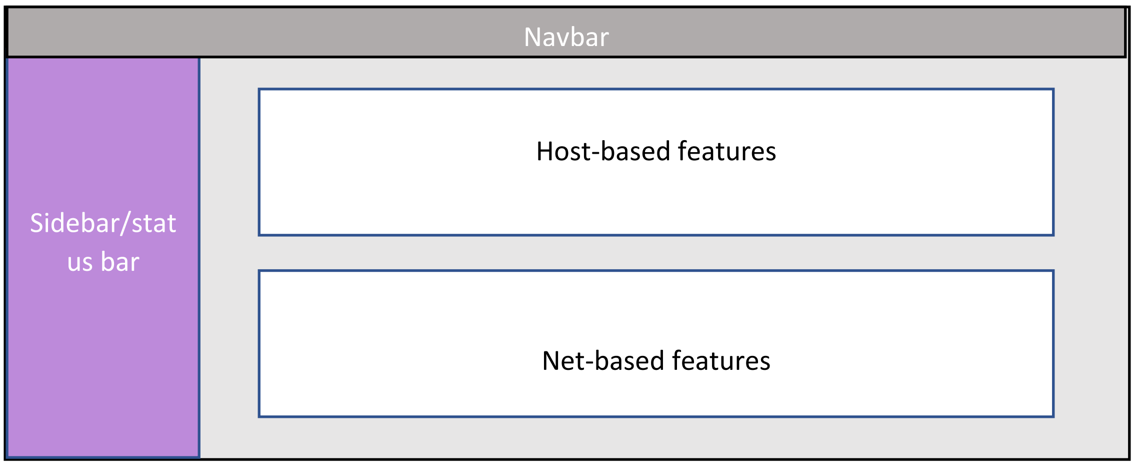
- ChangeScreen () - This method will be used to show the current screen by placing it at the top of the Container.

(The screen at the top of the container is the screen that will be displayed)

## Home

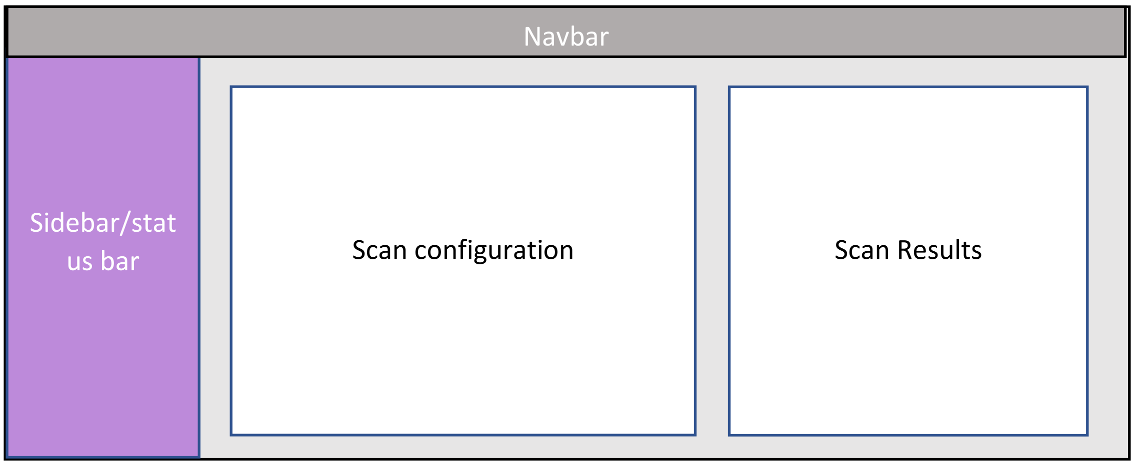
The home screen will be the first screen displayed to the user, and will consist of a navigation bar, sidebar, host-based security features and net-based security features.

Proposed home screen mock-up:



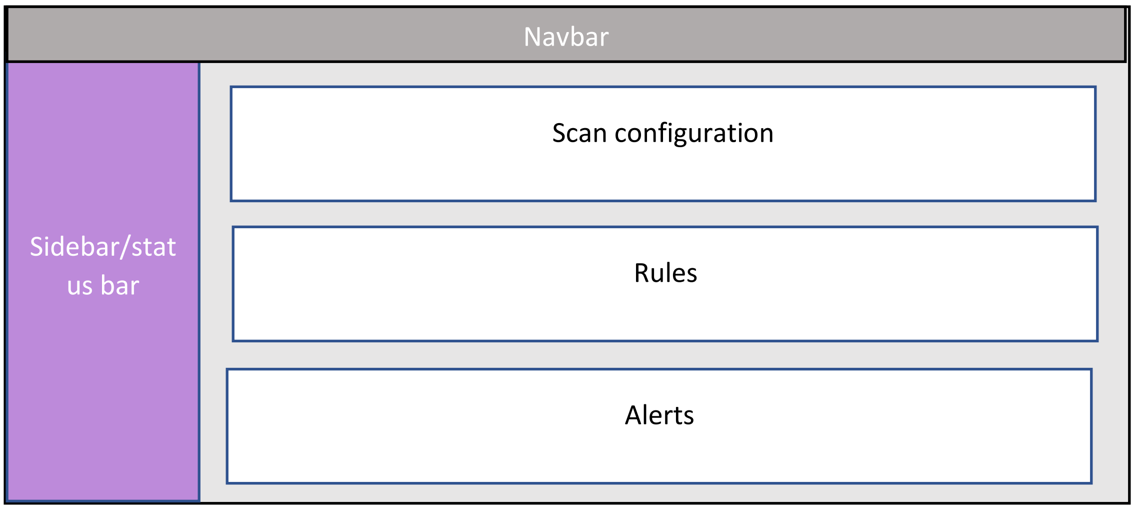
## File Scan

The file scan screen will allow the user to scan the host system for malicious content based on signature IDs in the form of hash comparison.



## Net Scan

The net scan screen will allow the user to scan their network based on a set of network rules. When these rules are broken an alert is displayed to the user.



## Core functionality breakdown / simple pseudocode

### File scan

- Allow user to configure scan path/type
- Use threading to avoid application hanging
- Iterate through each file in path (including sub directories)
    - Get md5 hash of file
    - Compare hash with malicious hash database
    - If match alert user
- Display results to user
- Give user actions to deal with results

### Net scan

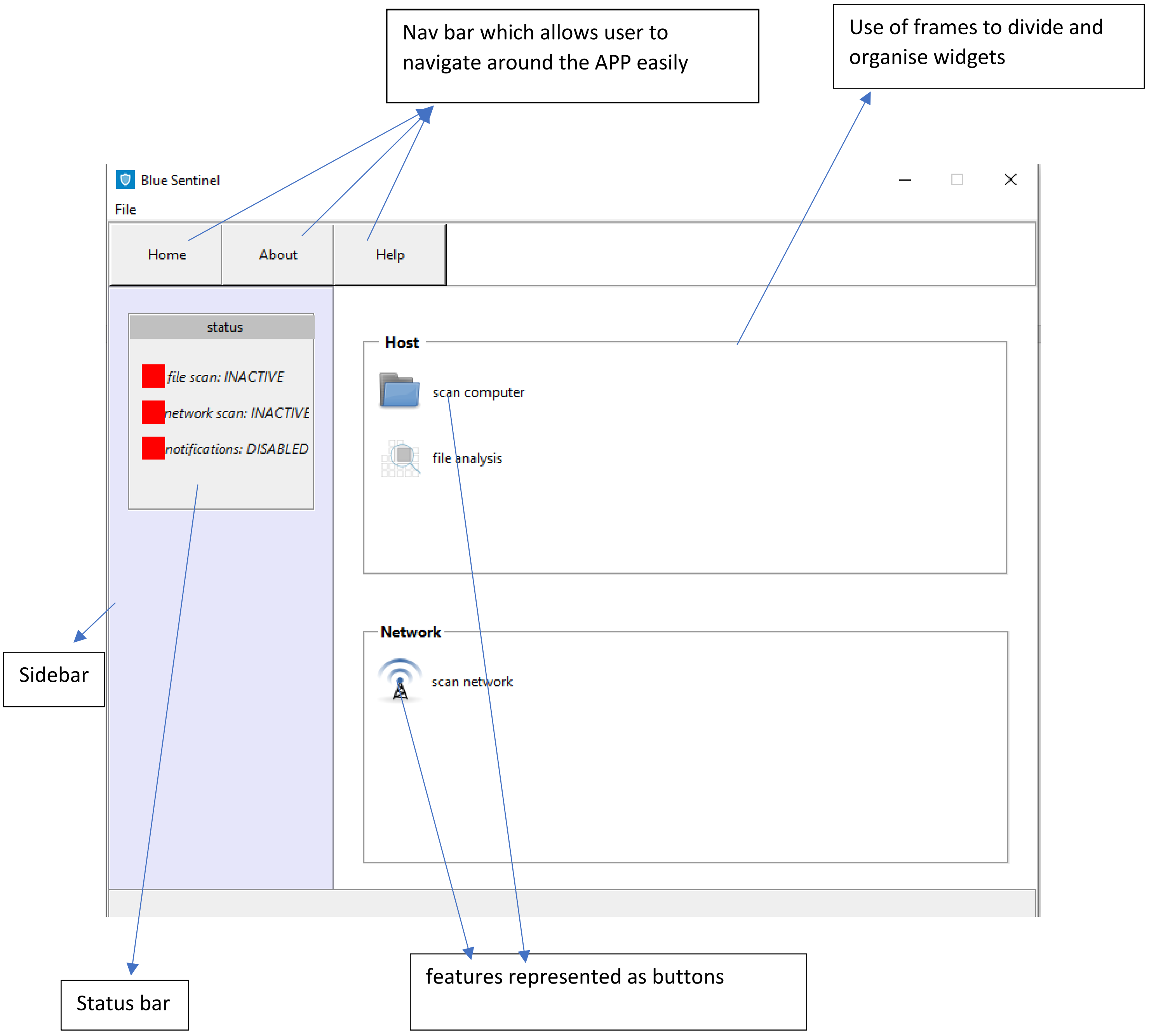
- Capture ethernet frame packets and decode/format correctly
- Decode/format IPv4 packets
- Decode/format TCP segment
- Decode/format UDP segment
- Decode/format data
- Covert IP and MAC addresses to appropriate format
- Check packets against rule set
- If rules broken alert user

## Development

### GUI

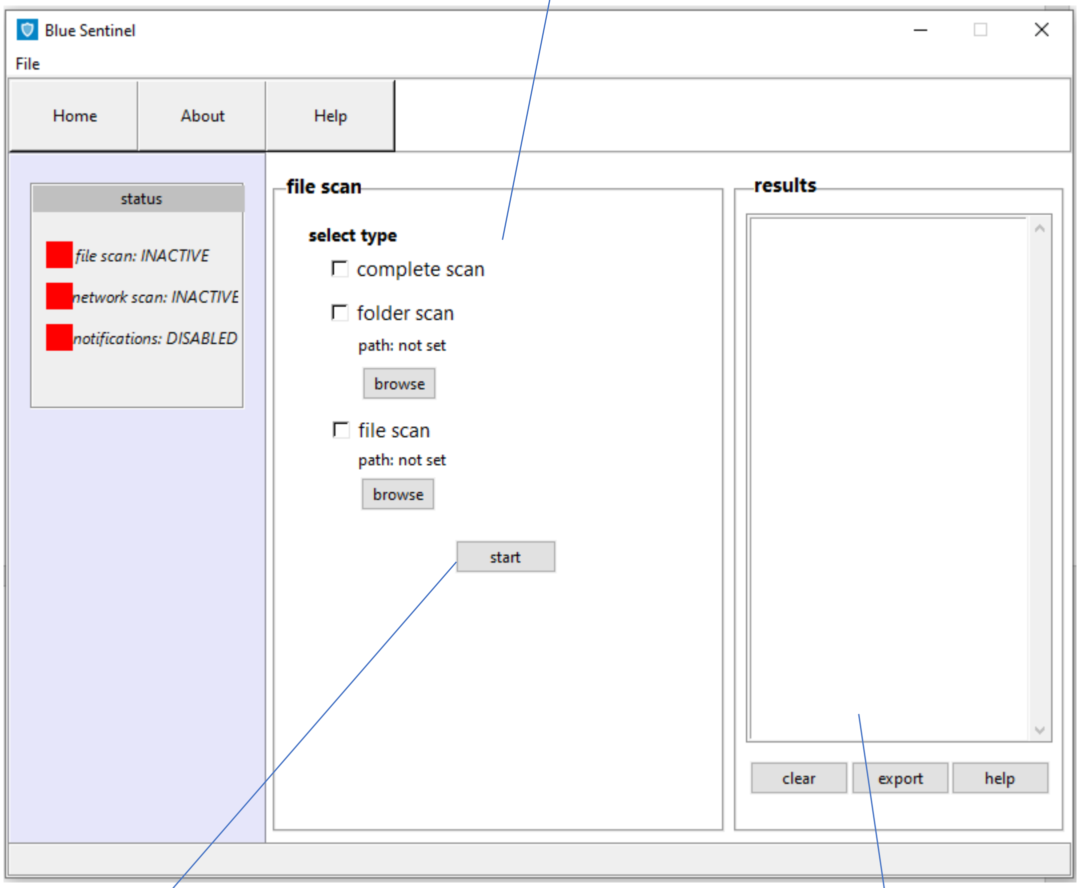
- I will be using Python's built in Tkinter library to build the GUI as not only does it give me more control but is cross-platform and provided by default with Python installation.

### Home



## File Scan

Scan config using check buttons



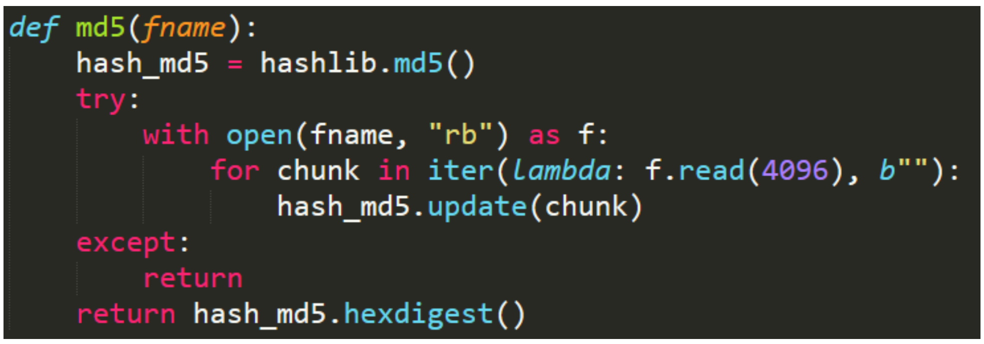
Button which starts the file scan but also does validation e.g.

- Is path empty
- Is an option selected (only 1 check button can be activated at a time)

Listbox which is populated with scan results

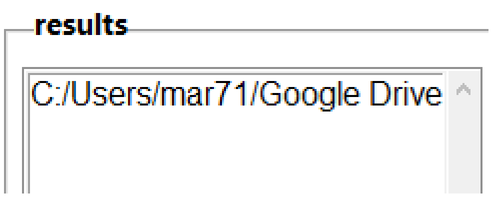
### File scan functions overview

- Calculate md5 hash



```python
def md5(fname):
    hash_md5 = hashlib.md5()
    try:
        with open(fname, "rb") as f:
            for chunk in iter(lambda: f.read(4096), b""):
                hash_md5.update(chunk)
    except:
        return
    return hash_md5.hexdigest()
```

Reads files in binary mode and in chucks of 4096 bytes incase file size is too large.



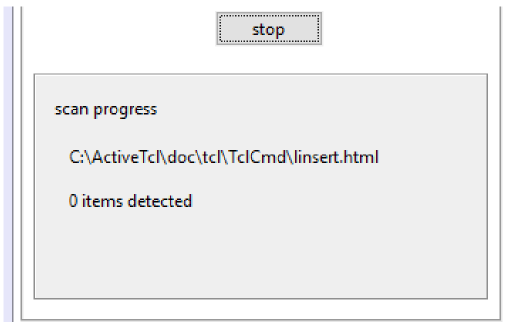
C:/Users/mar71/Google Drive

- Threading

I make use of threading by executing the file scan code in a thread to prevent the app from hanging allowing the user to continue using the app whilst the scanning is ongoing.

```
t1 = threading.Thread(target=self.scan).start()
```

- Scan progress



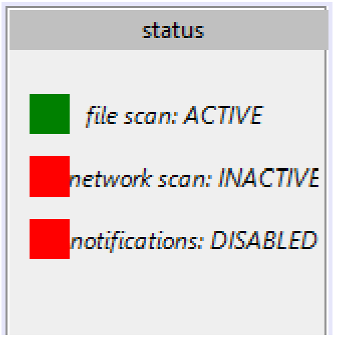
scan progress

C:\ActiveTcl\doc\tcl\TclCmd\linsert.html

0 items detected

Dynamic display which shows scan progress

- Updates status bar when scan is active/inactive

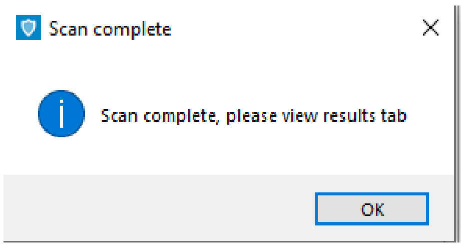


🟩 *file scan: ACTIVE*

🟥 *network scan: INACTIVE*

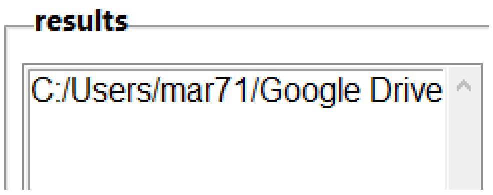
🟥 *notifications: DISABLED*

- Alerts

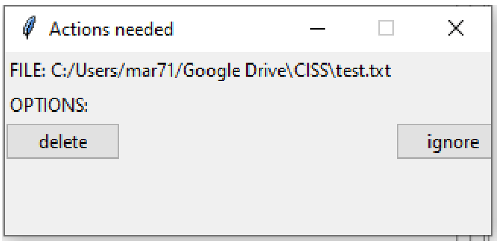


Scan complete, please view results tab

- Results



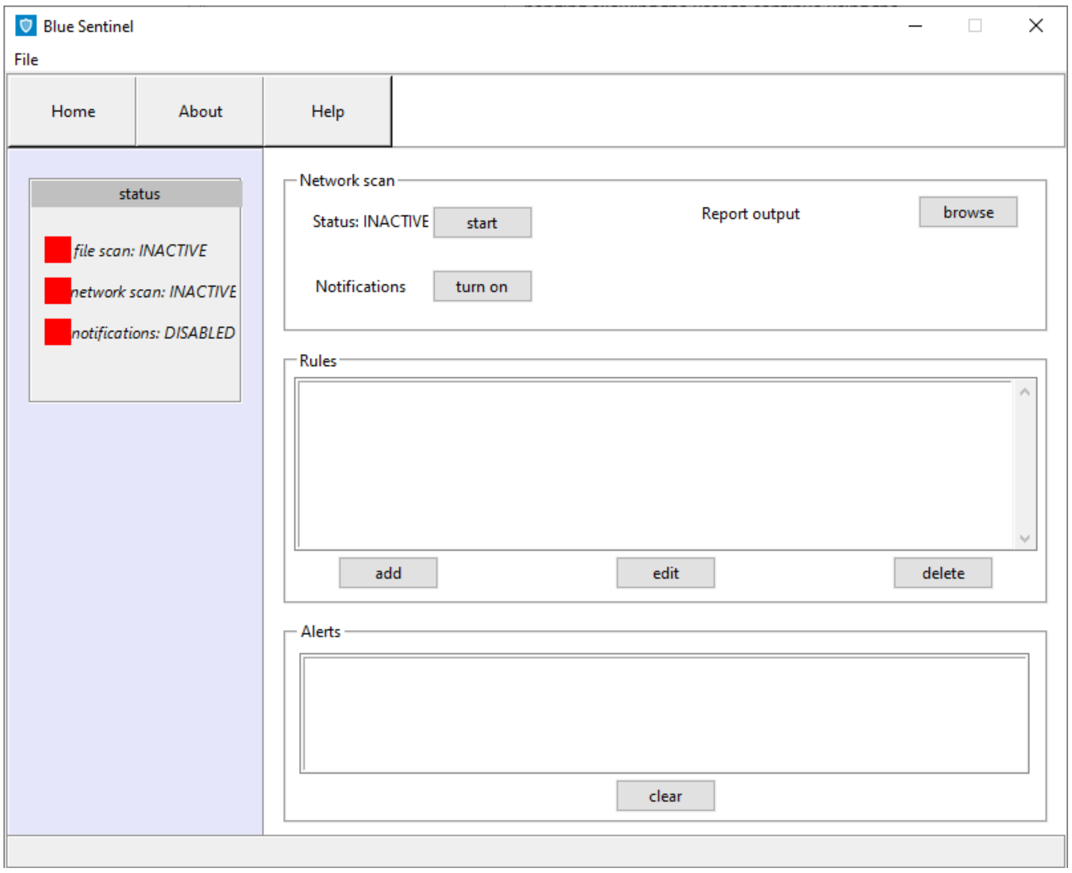
C:/Users/mar71/Google Drive



FILE: C:/Users/mar71/Google Drive\CISS\test.txt

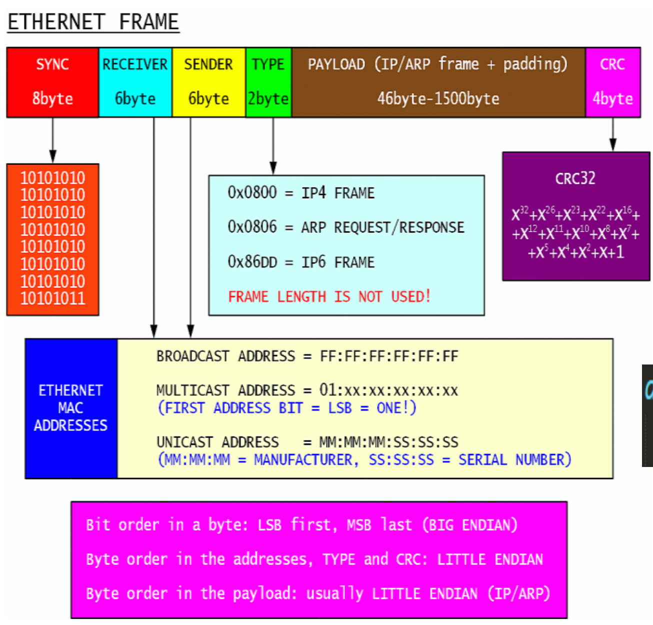
OPTIONS:

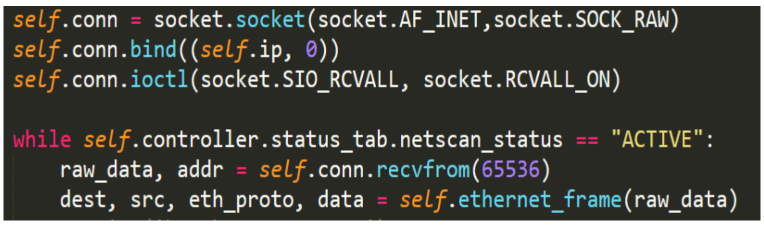
## Net scan

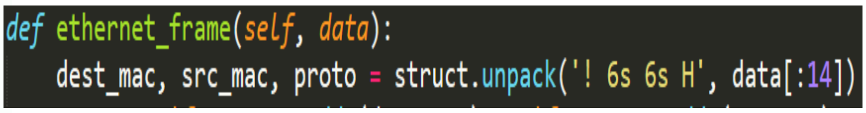


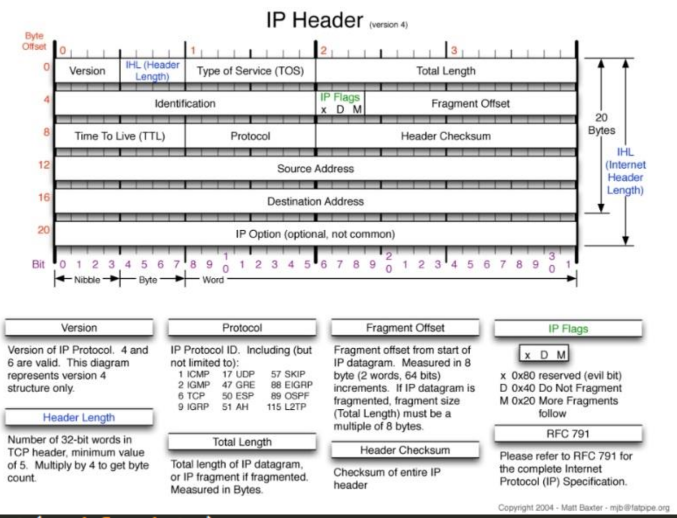
### Net scan functions overview

- Decoding and formatting packets and protocols (Ethernet frame, IPv4, TCP, UDP, ICMP)


ETHERNET FRAME
SYNC 8byte
RECEIVER 6byte
SENDER 6byte
TYPE 2byte
PAYLOAD (IP/ARP frame + padding) 46byte-1500byte
CRC 4byte
10101010
10101010
10101010
10101010
10101010
10101010
10101010
10101011
0x0800 = IP4 FRAME
0x0806 = ARP REQUEST/RESPONSE
0x86DD = IP6 FRAME
FRAME LENGTH IS NOT USED!
CRC32
X^32+X^26+X^23+X^22+X^16+X^12+X^11+X^10+X^8+X^7+X^5+X^4+X^2+X+1
ETHERNET MAC ADDRESSES
BROADCAST ADDRESS = FF:FF:FF:FF:FF:FF
MULTICAST ADDRESS = 01:xx:xx:xx:xx:xx
(FIRST ADDRESS BIT = LSB = ONE!)
UNICAST ADDRESS = MM:MM:MM:SS:SS:SS
(MM:MM:MM = MANUFACTURER, SS:SS:SS = SERIAL NUMBER)
Bit order in a byte: LSB first, MSB last (BIG ENDIAN)
Byte order in the addresses, TYPE and CRC: LITTLE ENDIAN
Byte order in the payload: usually LITTLE ENDIAN (IP/ARP)


```python
self.conn = socket.socket(socket.AF_INET,socket.SOCK_RAW)
self.conn.bind((self.ip, 0))
self.conn.ioctl(socket.SIO_RCVALL, socket.RCVALL_ON)

while self.controller.status_tab.netscan_status == "ACTIVE":
    raw_data, addr = self.conn.recvfrom(65536)
    dest, src, eth_proto, data = self.ethernet_frame(raw_data)
```


```python
def ethernet_frame(self, data):
    dest_mac, src_mac, proto = struct.unpack('! 6s 6s H', data[:14])
```



Version

Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.

Header Length

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

Protocol

IP Protocol ID. Including (but not limited to):

| | | |
|---|---|---|
| 1 ICMP | 17 UDP | 57 SKIP |
| 2 IGMP | 47 GRE | 88 EIGRP |
| 6 TCP | 50 ESP | 89 OSPF |
| 9 IGRP | 51 AH | 115 L2TP |

Total Length

Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.

Fragment Offset

Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.

Header Checksum

Checksum of entire IP header

IP Flags

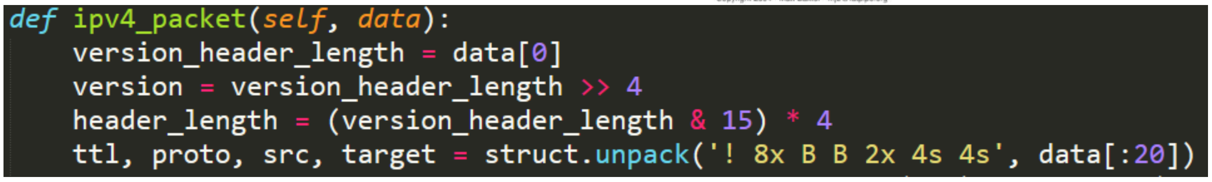
x D M

x 0x80 reserved (evil bit)
D 0x40 Do Not Fragment
M 0x20 More Fragments follow

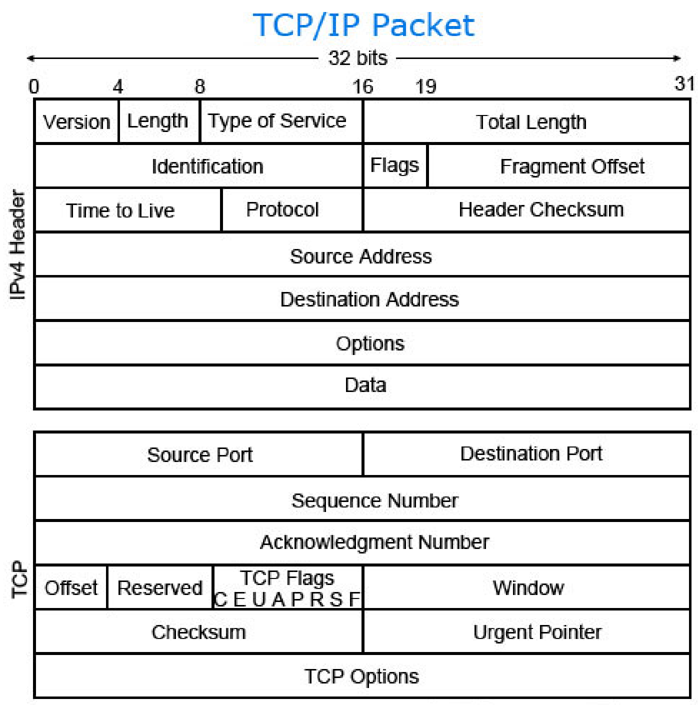
RFC 791

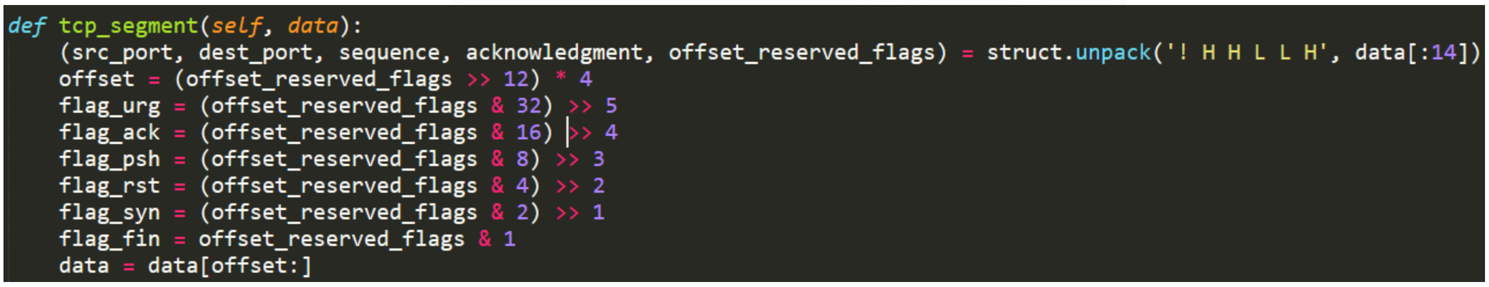
Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.

Copyright 2004 - Matt Baxter - mjb@fatpipe.org



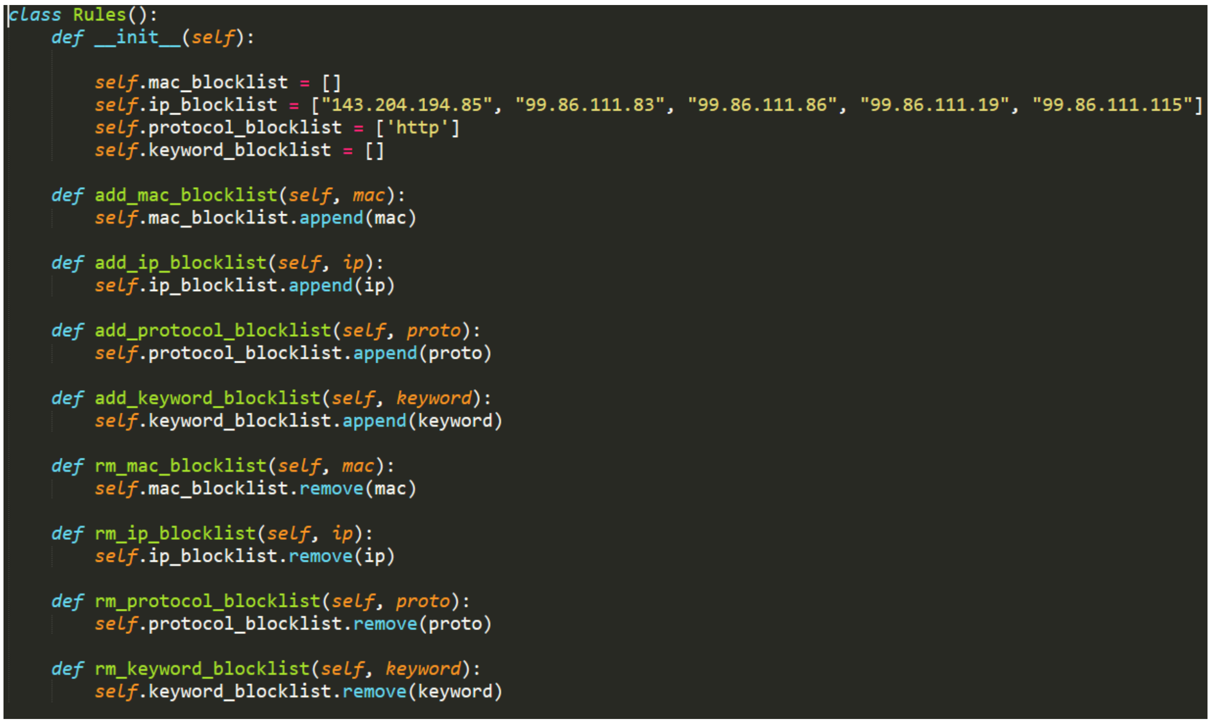
```
def ipv4_packet(self, data):
    version_header_length = data[0]
    version = version_header_length >> 4
    header_length = (version_header_length & 15) * 4
    ttl, proto, src, target = struct.unpack('! 8x B B 2x 4s 4s', data[:20])
```





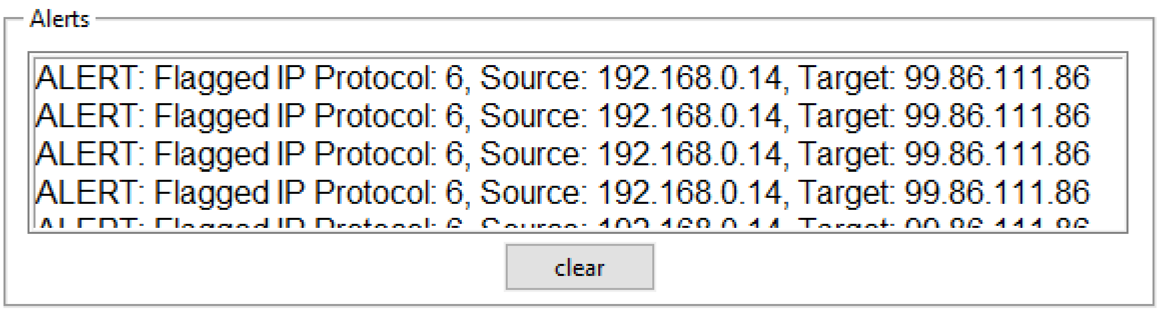
```
def tcp_segment(self, data):
    (src_port, dest_port, sequence, acknowledgment, offset_reserved_flags) = struct.unpack('! H H L L H', data[:14])
    offset = (offset_reserved_flags >> 12) * 4
    flag_urg = (offset_reserved_flags & 32) >> 5
    flag_ack = (offset_reserved_flags & 16) >> 4
    flag_psh = (offset_reserved_flags & 8) >> 3
    flag_rst = (offset_reserved_flags & 4) >> 2
    flag_syn = (offset_reserved_flags & 2) >> 1
    flag_fin = offset_reserved_flags & 1
    data = data[offset:]
```

- Rule checking



```python
class Rules():
    def __init__(self):

        self.mac_blocklist = []
        self.ip_blocklist = ["143.204.194.85", "99.86.111.83", "99.86.111.86", "99.86.111.19", "99.86.111.115"]
        self.protocol_blocklist = ['http']
        self.keyword_blocklist = []

    def add_mac_blocklist(self, mac):
        self.mac_blocklist.append(mac)

    def add_ip_blocklist(self, ip):
        self.ip_blocklist.append(ip)

    def add_protocol_blocklist(self, proto):
        self.protocol_blocklist.append(proto)

    def add_keyword_blocklist(self, keyword):
        self.keyword_blocklist.append(keyword)

    def rm_mac_blocklist(self, mac):
        self.mac_blocklist.remove(mac)

    def rm_ip_blocklist(self, ip):
        self.ip_blocklist.remove(ip)

    def rm_protocol_blocklist(self, proto):
        self.protocol_blocklist.remove(proto)

    def rm_keyword_blocklist(self, keyword):
        self.keyword_blocklist.remove(keyword)
```

- Alerts



```
ALERT: Flagged IP Protocol: 6, Source: 192.168.0.14, Target: 99.86.111.86
ALERT: Flagged IP Protocol: 6, Source: 192.168.0.14, Target: 99.86.111.86
ALERT: Flagged IP Protocol: 6, Source: 192.168.0.14, Target: 99.86.111.86
ALERT: Flagged IP Protocol: 6, Source: 192.168.0.14, Target: 99.86.111.86
```

## Project structure

I have followed the recommended/good practice way of structuring a large scale Python application for deployment as documented in detail on https://docs.python-guide.org/writing/structure/.

## Git

The development of this project was done on a private GIT Repo which I am happy to share upon request.

## APP USAGE

# Stretch Project

DIY Intrusion Detection System (HIDS + NIDS)

USAGE:

- Clone repo
- Install project directory as a package (run in direcotry of project folder)

```
python -m pip install -e .
```

- Install required packages from requirements.txt (Although all are default with Python installation)

```
pip install -r requirements.txt
```

- Change IP in net_scan.py to your Host IP
- Run main.py

```
python main.py
```