CS-GY-9963

Implement network security monitoring & incident response

Advanced Project

Yuvraj Singh
Guidance, Professor Thomas Reddington

NEW YORK UNIVERSITY

New York University, Tandon School of Engineering
Department of Computer Science
Department of Cyber Security

11-09-2019

Contents

1. Abstract

This project is an advanced project done under the curriculum of NYU Tandon School of engineering. The purpose of this advanced project was to analyze the effort and resources required to setup a minimalist Network Security Monitoring Model and Incident response. The effectiveness of various tools available in the open-source domain are attempted to be analyzed on the basis of the resource consumption, effectiveness and usefulness. Once the network was reasonably hardened, efforts to understand it's response to some classical and improvised network attacks were also tested out.

This paper attempts to serve as a guideline to the implementation done, technological prerequisites and knowledge required and challenges faced to any one who wishes to acquire more knowledge in this area. The paper concludes with scope of future improvement on the objectives and incorporating more kinds of attacks into the picture, or deal with a different threat model or network design all together.

2. Understanding NSM and deciding Implementation

2.1 NSM: Abstract

The definition of network security monitoring as stated in many online sources is *Network security monitoring is a service that monitors your network for security threats, vulnerabilities, and suspicious behavior. ... As it is automated and continuous, network security monitoring is a key tool for quickly detecting and responding to threats.*

Before a decision on the granular level can be made it is important to identify and consider several factors. The network size , no of subnets , no of assets, and firewalls, resource at dispense both financial and computing power etc. NSM is something which is done based on the context it is being done for, the assets and the threat actors .

For this implementation which would be a simulation, a network was setup on a network simulation open-source software called GNS3. For any kind of educational networking endeavor, GNS3 is an indispensable open-source tool with lot of features and flexibility supported by a large open-source community , well-written documentation and tutorial videos available at gns3 academy.

2.2 Implementation Architecture

For this implementation GNS3 was setup in a remote Ubuntu Machine. This remote version of GNS3 is also called the remote server version wherein GNS3 runs as a service directly on Ubuntu. It is worth noting that GNS3 is by no means a hypervisor, it is a mere simulation of starting qemu-imgs and simulating a network for this (virtualization being done by qemu). GNS3 documentation can be referred to for learning more on GNS3.

The GNS3 used here runs on an **Ubuntu18.03 distro** with **250 Gigs of RAM** which are more than sufficient if we are simulating **50 device network**. The routers, hubs and switches run on very low memory. For some basic device testing, Alpine Linux may be used which run on very little ROM and RAM. For larger machine qemu's such as Kali or Security Onion there would be additional RAM requirement.

The network inside GNS3 can be accessed by using a GNS3 software client provided by them. A typical GNS3 network can then be interacted with a GUI. This GUI is the means of interacting with the GNS3 simulated network. When the client is started for the first time, it must be configured to connect to the remote GNS3 client.

The remote GNS3 server (if setup) will ask for authentication credentials, but in a trusted private network or local versions may be avoided. The GNS3 client software (GUI) sees the remote server and machines running on it. Multiple clients may connect to this remote server, but it can run only 1 project at a time, and all connected clients make changes to this network which the other clients see reflect. The screenshot below shows a typical setup of a GNS3 network with some of the appliances.

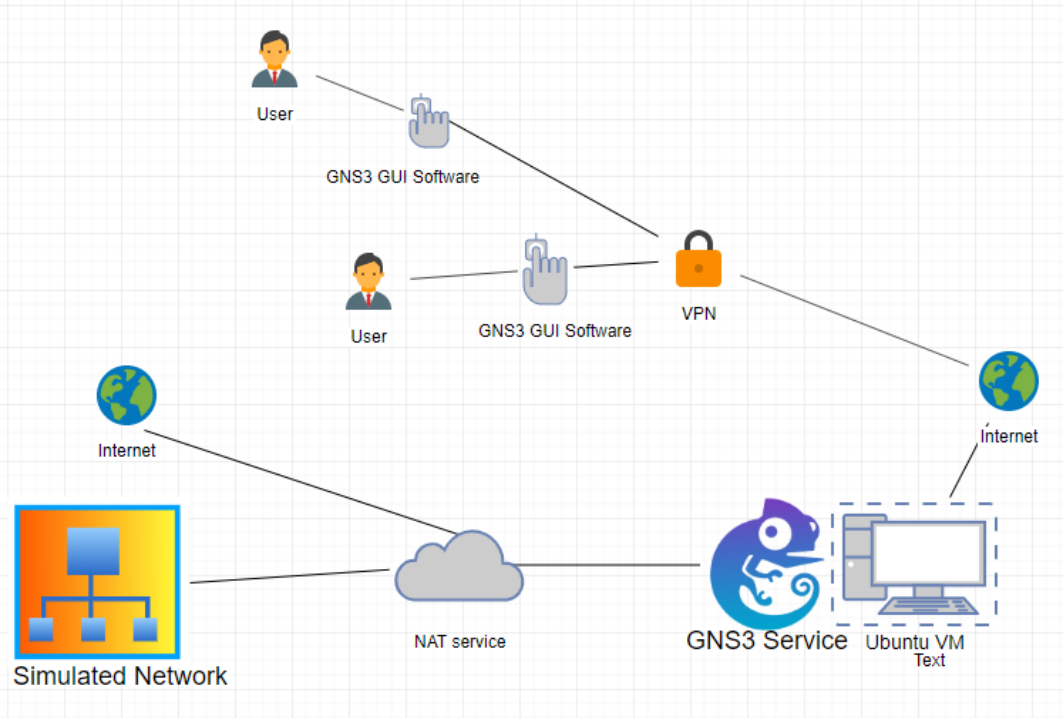

Figure 2.1: Simulation overview

2.3 Understanding NSM principles

Network security monitoring is based around some foundation steps, and these do not change with the implementation design or strategy. There are several baseline skills required , and network security design begins when the network is setup. Network security monitoring can also be done at various levels of granularity depending upon the expertise of the network security analyst. It can be intelligence driven backed by machine learning algorithms or rely more on public data for alert and use signature based detection.
The 3 pillars of network security monitoring are as follows.

2.3.1 3 major steps:

- **Collection:** Broadly speaking,this is the phase where the network data is collected , stored, categorized , maintained , rotated and discarded after they have served their purpose. The collection involves sensor placement as well , deciding on factors which can affect visibility in data, and a couple of other factors weigh in such as available computing power, average traffic on the wire, storage available, and assets. Some assets may be valued more than others, and not all packet data is equally insightful, and no. of sensors on the network cannot be near as many as no of devices present.
- **Detection:** In this phase, there are events which are parsed or extracted out of live packet capture or from the logs in retrospect. These events may be done differently depending on what the design principles are trying to shield against. The detection may be based on reputation , anomalous behavior or a signature match . Once an event is detected, it must be mapped to an action item which states what should be the action taken against this event.

- **Analysis:** Once the event is detected, it requires analysis before it can be tied to an action item. Much of detection is automated though it can prove to be a challenge. Analysis may also be intervention, but more often than not it requires some or other form of human intervention or decision making.

  These 3 steps mentioned do not follow a linear model, but a cyclic loop which never stops just makes the network security monitoring model more mature as time passes.

These steps are not sequential but a continuous cycle in the process of Network Security Monitoring. As threats keep on evolving , detection must follow analysis to continously harden the network against evolved threats.

2.4 Principles of Data Collection and Retention

The network is flood with vast amounts of data. Each device on the network is an asset but not equally valuable, and there is no such thing as infinite storage or infinite resouces. The whole of NSM would be a different ballgame, if there was no difficulty in computation and detection speed matching the wire traffic speed.
In a traditional network the data can be categorized into 3 major buckets.

- **Full Packet Capture (FPC) Data:**This fully detailed packet data stored in the traditional Pcap file format or taken just off the wire with all the information present. The advantage here is that this data is extremely granular so every decision of detection and analysis can be fully informed, and the signature matching can be done on complex attacks as well. The challenge here is that Full Packet Capture Data is extremely bulky to deal with and store. In a large network this can grow in size tremendously, and in the event of a DDoS attack this size can explode.
- **Session Data:** Also known as "Flow" data this format of data stores the summary of conversation between two devices in flows. It has information present in the packet header, but does not store anything about the packet payload. This is much smaller in size than FPC data, but the no of events that can be detected based solely on flow data is limited.
- **Packet String Data:** Packet String Data is a step in-between flow data and FPC data. It stores some payload information but not all. There are numerous ways of collecting Packet String Data (PTSR) depending on protocol and purpose of collection.

2.5 Key design decisions,Constraints Considerations:

Before moving, ahead with the sensor placement and zeroing in on what are the assets to protect, and which are crucial ingress and egress points of network data collection. We must plan our network. As this experiment was performed on a simulated network, there would be no dealing with **public ip** addresses and hence any scans or detection strategies based on reputation lists of (blacklisted IP's) will not play a part here. This also makes us change our **adversary** to be present already in a reachable private network (assumption being he has compromised a machine).
The basic network design around which most of the design decisions will be based is shown below. The central machine (Ubuntu) is running GNS3 and has control over every machine. In a non-virtualized environment, this can be replaced by using a tool like **Ansible**.

2.5.1 Sensor Sniffing Mode:

The sensors in any practical network must sniff traffic through *dormant* interfaces. This means that the sole purpose of these interfaces is to sniff traffic and not respond or participate in any sort of traditional interactions in the network nor respond to ARP or tcp queries. The sensor's aim should be to remain passive and sniff the traffic in the most efficient way possible. In traditional non-virtualized networks sensors usually utilize **smart switches** or **L3 switches** which have port mirroring capabilities. The two main ways sensors sniff traffic are :

- **Span Ports**: These are ports on which the switch mirrors the traffic from any other port. It is present in many advanced switches produced by cisco. In GNS3, to use open source and free software this was not an available option.
- **Network Tap**: A hardware device that takes traffic flowing through it and sends it out on a port connected to the sensor directly.

In this implementation, due to unavailability of the above technologies, the mirroring was done by placing the sensor in the same collision domain of the subnet (by using a Hub). This would obviously flood the network with more traffic than necessary, but in a small scale the difference would not be too high.

2.5.2 Acting remotely:

The sniffer is present on a different machine than the active next hop on the network layer which is the router. In this design model the alerts needed to be converted into actions of the remote host. This poses and additional design calculation that must be taken and will manifest itself later into the implementation. Since this focuses on a network based intrusion detection and prevention system, the hosts must be remotely protected at a network level. This does not supresss the importance of host based intrusion detection systems which must actively push the logs to SIEM for analysis at a regular frequency for corelation

3. Threat Modelling and simulation hierarchy

The textbook definition of threat modelling is this *"Threat modeling is a procedure for optimizing network security by identifying objectives and vulnerabilities, and then defining countermeasures to prevent, or mitigate the effects of, threats to the system."*.

Before the network design prototype is generated the major step is to identify and define the threats in the network. The threats in the network stem from the definition of asset definition in the network. This is important. The process of identifying threats is the first thing we do while designing the Application collection framework. The purpose of this endeavor was to closely examine the involved and thorough process of network security monitoring which can outline the expertise and knowledge required, the resources both storage and computative required to secure a decent network. With that goal in mind the model chosen here was network security in **Zero trust** or **Partial trust networks**.

3.1 Importance of threat modelling

Network security monitoring is a complex dynamic task. There will be blind spots in the networks, and there will be compromises made. Ease of Management, Visibility or efficiency cannot all be satisfied to the same decree. Threat modelling is defining and outlining these threats, their attack trees, their targets and categorize these threats according to severity, ease of compromise, depth of impact. This is albeit a small simulation of Network security monitoring, but the threat modelling is still an important aspect to keep in mind while beginning NSM and SIEM. There are some of the questions that need answered before we model our entire network security model based off of the threat model foundation.

Questions we ask during threat modelling :

- **Product/System being built?:** What are we designing? A web application? A network ? A mobile app? Once we know that, we need to chart out the architecture , the dataflow , the transactions and point of injections and data transitions.
- **What can not work as expected?**: Asking what can go wrong? where can it go wrong, what will it affect, how probable is the event, what is the impact and what can be done to mitigate the risk.
- **Prevention is better than cure:** What are the measures that need to be taken to prevent the event from happening or make it less probable from happening. In an exploit scenario, we are talking about making the exploit difficult to be executed.
- **Cure if prevention fails**: Analyzing the event after it occurred, analyze why it happened and work on improvements in posture.

The attacker here is assumed to be an agent who wants to do the following.

- **Compromise systems**

- **Exfiltrate data**
- **Stay hidden**

This brings to light the importance of threat modelling which is

1. Helps arrive at a more secure design
2. Efficient distribution of resources, prioritize security and development, and other tasks
3. Identify and segregate threats and evaluate the risk involved.
4. Define,design and implement required preventive measures and controls.
5. Strike a balance between risks, controls, and usability
6. Identify where risks are acceptable, what are the limitations and blindspots
7. Document threats and mitigation effectively. Learn from the pattern of threats being discovered in the networks over many years.
8. Most importantly ensure that business are adequately protected in the face of a malicious actor, accidents, or other causes of impact and almost never disrupted.
9. Identify security scenarios for testing out the model designed in place for the security requirements.

This is NSM done on a small scale, so the resource utilizations and other aspects need to be scaled accordingly. The simple way of explaining this would be that this is a network which lies somewhere where a part of the private network or subnet is the trust zone. The routers lie in the DMZ's of this subnet and are trusted. Another implication of this assumption is that the IP addresses involved in this threat model will be private IP addresses:- further implying that reputation based or public blacklisted IP's will not play a part here. Summarizing the above points, the attacker could also be assumed to be in the second phase of the attack where he has broken into a public network or Wifi and now looking to daisy chain his way into the network.

3.1.1 Defining Adversary

The adversary here is assumed to be having proficient technical knowledge and skills to carry out this attack with access to substantial amount of computing power. The attacker has time, and for this purpose has moderate amount of risk tolerance towards being prosecuted. This would also imply that the attacker's main focus is to sabotage systems and exfiltrate data whilst trying best to avoid detection. In short, the attacker is now trying to pivot deeper into the network using the concept of daisy chaining and exfiltrate data or damage system.

3.2 Describing the hierarchy

The network being examined or simulated in GNS3 represents a small protected subnet which connect to a router handling the traffic of other subbnets. The router may be an egress router, an area border router or other router. For this simulation it is important to note that, the router cannot be accessed over the internet and sits behind a NAT. The router is hence reachable only via a private address.
The sensor sits passively sniffing all the traffic that enters the subnet (Hub used here), in a real life network scenario a SPAN port or TAP interface will be used. The Sensor is passive and does not respond to any ICMP probes or ARP request and response. The sensor is connected to the **"Central Analysis Machine"** which is the SIEM instance. In the design implementation, this SIEM machine is the host Ubuntu running the GNS3 service.

The SIEM is the place where all the logs are aggregated and kept for analysis and further retrospection. The important thing to note here is that sensors are always decoupled from the SIEM center. The sensors are tasked with sniffing on network traffic, and usually have resource extensive operations and hence they should be minimized with tasks. Anything which is not a part of data sniffing or conversion should not be run on the sensors directly.
It is also important to note that sensors are not active devices on the network. They are by definition passive and hence they cannot directly "prevent" an attack by setting up firewalls on themselves, they need to convert these action items to alerts and instruct the routers to do this. The image below shows a logical representation of the network.

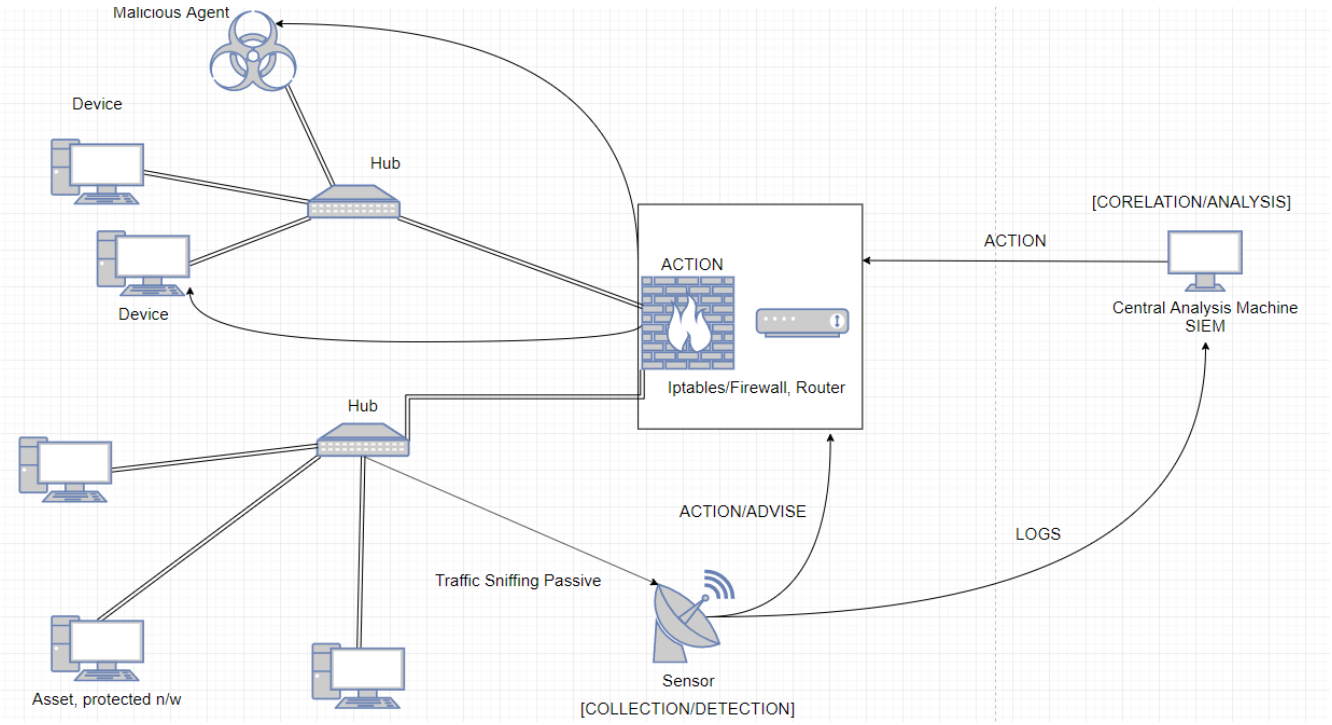

Figure 3.1: Logical network view

3.2.1 Segregating logical planes

From the section above, the network security monitoring can be viewed as 3 decoupled planes which interact with each other. The following diagram explains the abstraction of the planes and what constitutes them. Once these planes are clearly defined, their area of operations and interactions can be implemented. That is shown below. In this NSM model, the sensors are Half cycle. A half cycle sensor majorly performs collection and detection. There is immediate analysis performed (events which do not require human intervention) using IDS sitting on the sensors, but the retrospective analysis is done on the SIEM machine which is centralized. The logical grouping of the events look like this.

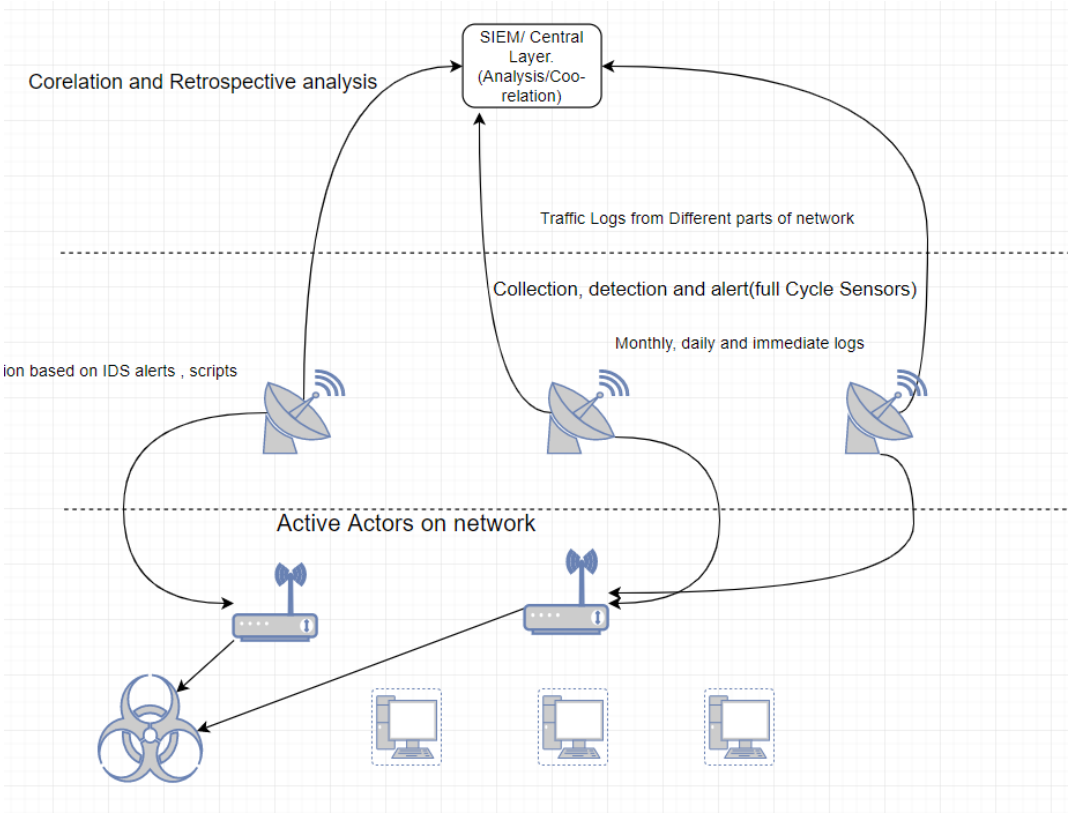

Figure 3.2: Logical Planes in the network (NSM perspective)

3.3 Defining Assets

The victim or the target here would be the close real life simulation of a startup or an organization which shares its private network space with other companies. This is true in many cases, and drawing from personal experience true for small companies that share large workspaces managed by third parties. For this kind of organizations , though it is most common to outsource the network or infrastructure security.

For this implementation, as shown in Fig 3.1 , the organization possesses one or more of the subnets in the private network with a trusted router. The organization is connected to a **shared or public** private space which has been compromised by an attacker. This implementation focuses more on network based intrusion

detection and prevention systems but will touch upon **HIDS** in ending or concluding notes.
The assets of this organization are as expected, with internal instances of web, mail, ftp servers running and also a handful of Linux machines. Since this endeavor does not focus on hardening host based IDS /IPS , the operating systems or vulnerabilities related to them are not of prime focus.
The assets or types can be enumerated as

1. Web servers
2. Routers
3. Personal Linux Machines(prone to exploits)
4. FTP Servers

Since this is a focus on the network based intrusion detection and prevention, the focus will be more towards the analysis of network protocol traffic and analysis. The different assets described are to enumerate on what protocols are expected and to be protected on network and determine protocol abuse or identifying protocols which are considered anomalous on certain networks.
The web servers would usually sit behind application firewalls which are more suited to detect attacks on the application layer such as SQL injection or CSRF or XSS. Network traffic however may be suitably used for detecting scans or bruteforcing easily.
When there is encryption involved however, a passive sensor would not have access to the **decrypted payload** and hence will be forced to base the judgement on heuristic analysis. There are several techniques based on flow size to identify malicious activity even with encrypted payloads based on flow size with a great degree of accuracy. These are discussed in greater depth in the analysis section of this report.

4. Phase-1:Data Collection

The biggest and most challenging aspect of NSM is to perform Data Collection. The entirety of analysis and detection and response sits on the foundation of data collection. When it comes to dealing with network data we are faced with the traditional problem of *Visibility vs Size*. The quality of the network security monitoring model and incident response hinders on the foundation of the quality of data being collected and analyzed.
There are several forms in which data may be collected and stored. There are 3 major categories in which this may be organized. These are :

- **FPC Data**: Full packet data
- **Flow Data:** This is a summary of conversation or data. The format depends on the flow generation tools. These may be of the format of IPFix or the Cisco netflow.
- **Packet String Data (PTSR)**: Size and detail wise these sit right between FPC and Flow Data (also called Session Data). They are like application styled and have some granularity but not like FPC data.

The size and detail metric of the data type is thus:
$\boldsymbol{FPC > PTSR > FlowData}$
The 3 data kinds have been elaborated on a bit more depth in the upcoming sections and also how this data is being collected.

4.1 Application Collection Framework

The process of collecting data in the context of Network security monitoring is governed by the Application Collection Framework. This defines the stages involved in the process of data collection and these have been enumerated below.

1. **Define and Identify threats**: This was discussed under threat modelling. The threats we want to should be identified along with the ways of compromise and the assets they will be likely to target.
2. **Quantify Risks**: Once the threats identified, the risks they pose to the network are identified and graded, this in security is based off of two factors, the impact or severity of the risk and the probability of the risk being realized.
3. **Identify Data Feeds:** The next crucial thing is to identify data feeds. In this context we are looking to identify the source of network data for the sensors. There also needs to be several design decisions that needed to be made based on Sensor design, placement and hardware capacity. Many placement decisions for the sensor can compromise the access to data and hinder zeroing in on actual threat.
4. **Narrow Focus:** Once all the above has been realized, it is important to narrow the conentration based on priorioties and the focus needs to be narrowed into solving these existing , identified and most crucial problems.

4.2 Session Data

Session Data or flow data is summary of network streams. The network flows are generated from packet streams and presented in Netflow or IPFix format, these are different ways of representing 5 tuple or 7 tuple data from the network constituted by source and destination IP address, transport layer port numbers, IP Protocol, Type of Service (ToS), and the input interface port. The flows are usually terminated when a tcp connection or stream ends or if there is a max cap to the flow size limit.

4.2.1 Utility

Flow data is summary of network data, so lot of information is hidden. But there are things which can be uncovered with flow data even with the summary information. The advantage of flow data is that this can be stored for a longer period because of it is size. The kind of attacks or threats that can be identified with flow data is

- **Port Scans:** Any form of port scanning results in a abnormal amount of short flows. These are due to half open syn scans which result in shorter flows (size in bytes) but have a longer time, as these are never closed. Though port scans may be detected as occurring in network, the half connect scans are difficult to zero in to a device as IP addresses can be spoofed.
- **DDoS**: Detecting DDos attacks are usually measured by comparing against a size of traffic metric compared against what is expected and what is not. Flow Data can thus be used to detect DDoS attacks as well. Some of DDoS attack variants are also done by sending a lot of TCP syn traffic and hence may be detected but the defaulting IP is still difficult to be narrowed on.
- **Bruteforcing attempts:** Since the wrong authentication generates a different message from the successful authentication, using flow size and packets estimates can be made whether the flow represents a successful authentication or a failure. Using this multiple attempts over short time may be tagged by just using flow data.
- **ICMP based**: There are ICMP based attacks which can be detected using flow data. These attacks are Ping of Death, Teardrop as these have different payload size than normal ICMP packets and makes identifying them easier from packet size.

As flow data is of summary nature not all attacks can be detected with it. The above attacks can be reasonably well detected using flow data. The flow data is useful for getting information about network traffic size without having to store the entire packet data in detail.

4.3 FPC Data

The most granular form of data and this gives the advantage of getting every detail of the packet. However, sniffing and logging every packet becomes a humongous challenge even for specialized hardware when done over high speed links. The size of FPC data is huge and hence most sniffers and IDS/IPS choose to sniff every 5th or every 3rd packet. FPC data needs to be stored and log rotated properly.
Usually packet data is not logged into files but fed into an intrusion detection system using port mirroring and the logs which the IDS generates after looking at the logs is what defines the data extracted from the FPC data.

4.3.1 Utility

Since we can see everything on paclkets with visible non -encrypted payloads, the signature based detection schemes used by certain Intrusion detection systems works well with such granular data and separate out the malware if regular expression or behavior matches.
The full capture data can also be for checking the integrity of the downloaded file. Any downloaded file over clear text protocol can be extracted out from the packet capture file and then looked at the files separately or analyzed by an anti-virus system. With enough processing and computing power, full capture traffic may be used to identify a wide range of network attacks depending upon the kind of approach being used (Signature or Anomaly based). This implementation works around Signature based detection only.

4.4 PTSR Data

Packet string data tools usually exist on a per application or per protocol basis. There is no hard and fast rule governing what should be contained in PTSR or packet string data. Packet String data is good for seeing the interesting bits of a stream or packet flow without delving into extreme granularity of FPC , hence PTSR tools are engineered towards the purpose of why the sniffing is being done for. URLSniffer a PTSR tools emphasises the HTTP domains and URLS' being accessed to generate the logs from packet data over wire or packet capture file. Another tool Justniffer is a TCP based tool that outputs customizable logs. Most IDS output or logs about packets can be considered a form of packet string data.

4.4.1 Utility

The packet string data can serve a number of utilities , one of them being detecting protocol misuse or bruteforce attempt or access to blacklisted URL's or IP addresses. Packet string data is the most useful of all data type if they are designed to contain the right and useful header and payload information balancing both efficiency and minimizing storage space.

4.5 Data segregation and push

Each kind of data requires a collection , storage and shipping methodology. Since each kind of data grows at a different rate with respect to sniffed network traffic, they are setup differently for this implementation.

They however share the common design of shipping the logs to the SIEM for retrospective analysis. H

4.5.1 Flow Data Collection

Flow data is collected using flow generators and flow collectors. Flow generators are tools that generate packet data sniffed off the wire into flow format. Collectors are tools that store the flow records into compressed binary format for later querying or conversion to logs.

4.5.1.1 Implementation

- This implementation used the SiLK set of open source tools by CertNetsa Cert Netsa For this implementation, **YAF** or *Yet Another Flowmeter* was chosen for flow generation. YAF converts the packets to flows in IPFIX format. There are multiple options for outputting the flow-data.

- The flow-data was sent to an IPFIX listener on the sensor machine which is **rwflowpack** (from Cert Netsa)which stores flows in a binary format that they can be queried or filtered.
- Once these tools are installed and configured, the next thing to set up a log rollback mechanism for maintaining the log size.
- Also, there must be a watcher that periodically takes the flows and converts them into log files which can be shipped.
- Secondly, a watcher or cronjob should be configured which will restart the YAF and rwflowpack service in case they exit or quit unexpectedly.In this implementation, the flow data
- Later, these data must be stored in the SIEM machine with higher processing power to spare for a longer duration.

The flow data collection logically can be pictured like this.

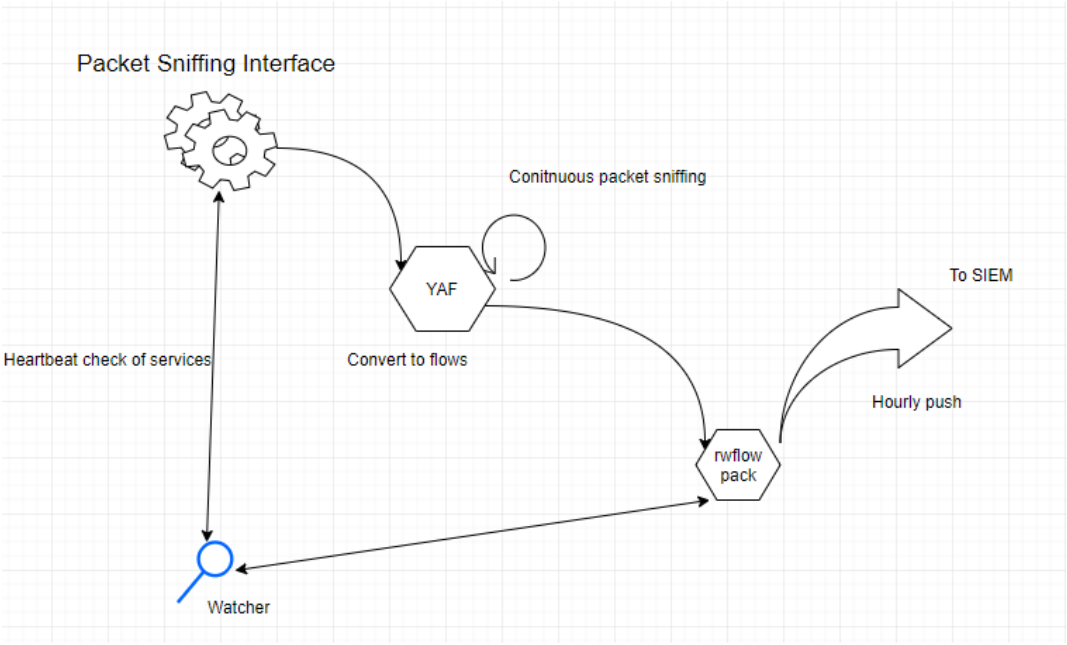

Figure 4.1: Flow Collection

4.5.2 Packet String Data Collection

In this implementation the open source utility **Justniffer** has been used to collect packet string data. Justniffer is an open source utility that sniffs TCP traffic and can print custom output to files in a highly customizable format to logs. Packet string data is useful for detecting application level attacks especially HTTP and SSH. Justniffer can be downloaded or learn about from here. Justniffer directly writes to log files which are managed by a custom bash script written as a cron job for the implementation.

4.5.3 Full Packet Data Collection

The fullpacket data is the easiest to collect but the hardest to store and maintain. In this implementation the full packet data is directly sniffed off the wire and fed to **Zeek** which converts them into zeek style logs. These zeek logs are of PTSR type and gives a lot of useful information. The logs show the interesting and critical field information to the security analyst and zeek is highly customizable with respect to performance and logging. There is an advanced setup where zeek can be run as a **clustered architecture**, but for this project a standalone design was chosen.. The zeek scripts are customizable and generate logs in PTSR format

4.6 Data Collection and Analysis: Requirements Architecture

With the local data collection strategies set in place, the next challenge in network security monitoring is to create an architecture for data collection and analysis which will provide visibility on network data as well as provide means to act on the data. The first step that needs to be taken in this regard is to estimate the size of the network data and collection capability.

The Analysis part is discussed in a later section but needs to be kept in mind initially from the architectural standpoint as well.There are other requirements which are not covered in depth here, one of them primarily being **Critical Baseline skills**. Each organization must devote resources either by recruiting or outsourcing for the continuously adapting process of Network security monitoring.

4.6.1 Storing the data offshore

In any real life and practical network security monitoring scenario, the resource utilization is extreme. Most of the sensor's resource utilization happens to keep track of network data and parse network traffic into flows. Hence the data size is too large to be kept in the sensor for longer periods.
The other important factor is **corelation**. For analyzing relations between similar activities in logs from other sensors, it is of paramount importance to ship the data to an off shore SIEM unit.

4.6.2 Understanding available resources and hardware

The important constraints when it comes to implementing data collection are as follows. The data constraints kept in mind while designing the network are as follows.

- **Sensor Choice:** The sensor needs to run on some hardware or could be entirely software based (cheaper) and have an operating system run such as Security Onion. Cisco uses hardware based sensors which generate Netflows in the hardware level as well.As mentioned earlier, the sensor could be 1 of the 3 major types: **Collection Only**, **Collection and Detection: Half-cycle** or Collection Detection and Analysis..
  For a network which needs the analysis to be centralized (fewer SIEM units than sensors), the sensor needs to be half cycle. In this implementation, the sensor is a half cycle sensor which performs collection and detection, but not analysis.

- **Sensor Placement:** Often overlooked during the process of Network Security Monitoring is the concept of Sensor Placement. The sensor placement is extremely crucial to the network design. The sensor placement is dictated by several factors. Some of them are :

  – **Proximity to critical assets:** The Sensor should have sufficient proximity to network access so that the access and action on events if needed can be channelled through the sensor.

  - **Network Visibility:** The sensor should visibility should not be hidden or compromised. If the assets are behind a Natted firewall, the sensor could lose visibility to which device is being compromised. Such factors like NAT , routing different zones such as DMZ must always be kept in mind while placing the sensor. It is beneficial to the overall process if sensor visibility diagrams are created to facilitate this process.

- **Sensor Operating System:** Another critical choice,as certain tools can run only on certain operating systems. The popular choice running on non-customized hardware is Linux, for this the implementation Security Onion was chosen.

- **Network Ingress and Egress Points:** It is of paramount importance to identify all ingress and egress points in a network. These are the areas of crucial sensor placement for critical traffic visibility.

- **Sensor Protection:** The better network choice is always to keep the sensor passive and only accessible over tty shell or a protected IPMI interface, and all updates made to it must be strictly monitored. It must be physically protected as well.

- **Traffic Data Size and Rate Estimate:** Using a sample analysis during the network peak hours , an estimate can be made on the traffic that will flow through the sensor over the day or week. A sample of several hours during peak hours can be a good metric for traffic estimation.Traffic Rate is another metric which is important to keep in mind , as this is something which will be restricted by hardware choice heavily. Some implementations choose to sniff every nth packet as a work around and still provide substantial visibility.

The network design chosen is shown below, with minor modifications being made to host devices based on the attacks being tested.
The **green** area represents the monitored network by the sensor.
The center zone, marked by a dotted circle represents action devices which are accessible via the sensor. The devices in the centre zone, including the sensor must be hardened and not have any service running on them. Theses devices excluding the sensor are routers and should have access list configured to block every access except those from **sensor and central SIEM device**.
The red and yellow zones represent networks outside the trust domain or autnomous system. All traffic flowing through these into the green domain is **ingress** traffic. All traffic flowing from the green zone to these domains are **egress** traffic. Due to lack of access to proprietary cisco switches with **TAP and SPAN** ability, mirroring of traffic was achieved in a **suboptimally** by using a HUB (this in actuality could simulate a Wireless network).

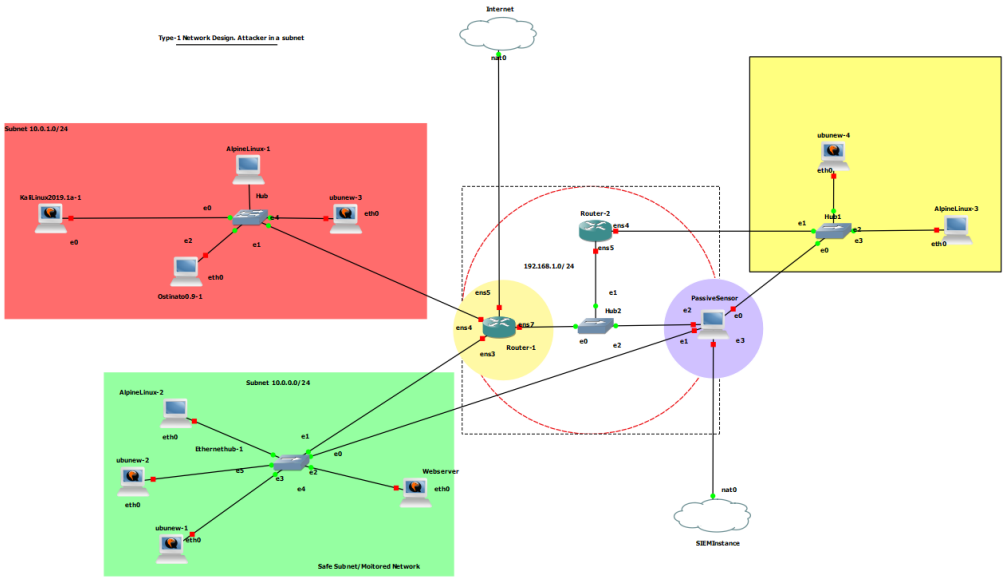

Figure 4.2: GNS3 Network Implementation

4.6.3 Implementing the Collection architecture:

The different aspects of collection, sensor placement and analysis must be implemented. Ther are several aspects of this which may be achieved by selecting from a wide array of tools both from enterprise and open-source free solutions.The collection goals and tools used to accomplish them have been enlisted below. The entire network is already setup on GNS3 and looks

- **Sensor choice:** Security Onion OS. This operating system is based on ubuntu and already configured with data collection and analysis tools such as Suricata, Sguil etc. The security onion is highly customizable and configuring interfaces in either management or sniffing mode is also easier. This however is not a binding restriction, and it can be very well be replaced by any traditonal Ubuntu Distribution. The sensor was an emulated qemu image running on GNS3. For a network monitored having 15 devices, 6 GB of RAM for the sensor was found to be more than sufficient with 25GB of disk space for minor storage and applications.
- **Flow Generation Collection:** This was done using SiLK suite tools by Cert Netsa (Link). This was done on the same sensor OS (ran locally)
- **Packet String Data Collection:**. Locally done using an utility called justniffer (Link).Most of the analysis PTSR data was generated by **Zeek Logs**. Zeek is an open-source network traffic analyzer (Link). Both justniffer and Zeek were generating the logs locally
- **Full Packet Data**: Zeek was configured to sniff traffic directly off the wire. Zeek was running in **standalone** mode and not in cluster mode. One monitor per sensor is being monitored by the running zeek instance. On security onion, it is important to disable the older version called **BRO** as it can lead to a port conflict.
- **Log Rotation**: This is being done by simple bash scripts that rotate logs over the time based on size. Flow data is kept on the sensor for over 3 months, while the PTSR data lasts for 1 month , and FPC

data is refreshed every day. This is a resource utilization choice and depends on available resources and also on requirements. For some flow logs like ones generated by **YAF**, these are already stored in **binary compressed format**, so these were converted into ascii logs by bash scripts periodically as dictated by cron.

- **Log Shippers: Sensors**> For shipping and segregating logs to a central SIEM instance, the decentralized shipper **Filebeat** is used.The SIEM is a central instance which uses a queing mechanism to handle these inputs. Filebeat has inline caching which helps when the sensor services go offline unexpectedly. Filebeat can also be extended to be used on every host in form of **Winlogbeat** or **PacketBeat** to ship host logs and metrics for HIDS approach.
- **Watchers:**These are services that overlook the management of services that are responsible for data collection and segregation. These services also ensure that the filebeat only accesses the intended logs and converts logs into standard csv format if need be.
- **Logstash:SIEM** This is a central instance running remotely on the SIEM machine and parses the logs and ships them into the desired "**Elastic Index**". Elastic Indexes are noSQL elasticsearch datastores that store data on the SIEM, however the architecture design is flexible to have them run remotely. Elasticsearch adds resiliency , redundancy, and distributed and parallel processing to the design. In this implementation, the elasticsearch runs locally on the SIEM (Same machine as logstash).
- **Elasticsearch:** A distributed noSQL full-text based noSQL data storage area which can be run offshore from SIEM. For this implementation it runs locally on the SIEM machine. The main purpose of SIEM is to support the analysis phase of the NSM which will be aided by Kibana. The Kibana instance will be pointed towards this elastic index to get insights into the log data.

These services are very easy to setup and supported by extensive documentation. The SIEM and the sensor machine should have an interface with communication and internet connection on that interface. The sensor would have this interface configured in managed mode (not sniffer mode). The following shows the logical structure of the data collection implementation.

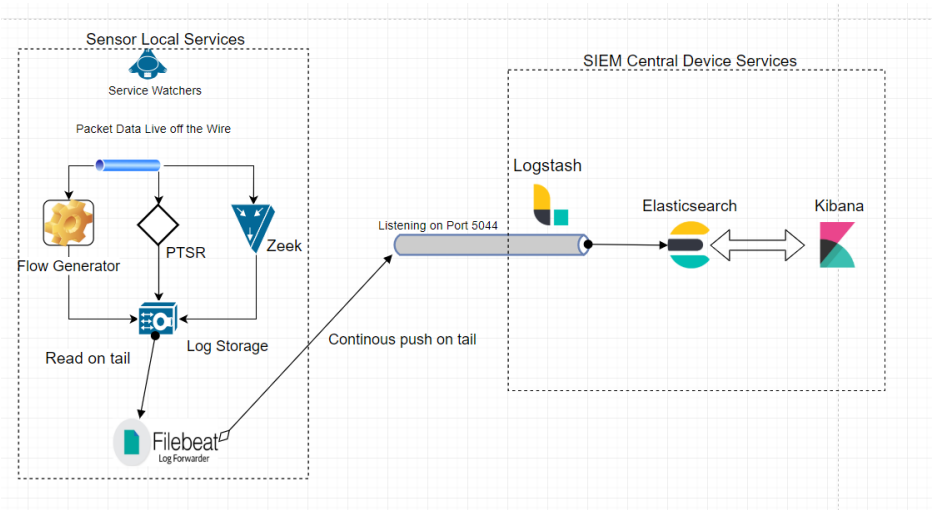

Figure 4.3: Data Collection Architecture

5. Phase 2:Detection

5.1 Abstract

The final phase of Network Security Monitoring and the most complex one is Analysis. This is the phase where there is human or automated interpretation of data to detect meaningful events that have occurred in the network, and segregate them according to normal or abnormal. This is an important decision and the foundation of this lies on **Indicators of Compromise**. These combined with threat modelling define what is normal in the network and what is not. It ultimately boils down to looking at the available data and logs, and determining where and how to look for activity by threat actors. This task is simple to state, but incredibly complex. Threat actors change, zero-day attacks bypass reputation based detection, data is huge and it is a challenging task to parse these fast to respond to threats in real time. A good model of NSM is one which is continuously adapting from the data and events observed, and this is propelled by the Analysis stage of the NSM.

5.2 Detection Mechanisms

After the data has been collected, there are several detection mechanisms by which different events may be detected. These depend on the kind of threat actor we are trying to detect or the kind of network attack we are trying to prevent.
There is 1 major classification for these detection mechanisms.

1. **Host Based Detection Mechanisms:** These are typically mechanisms which focus on analyzing the host logs and behavior to distinguish host based attacks. These are mechanisms which are used to design and build most anti-virus programs. These are not the focus of the implementation done here, but there are several softwares and tools which can aid in host based detection Mechanisms.
2. **Network Based Detection Mechanisms:** These are all the packet sniffers and sensors that parse network traffic to parse for suspicious activity or interesting events.

5.3 Indicators of Compromise

All of detection mechanisms boils down to determining what is normal and what is not. This disticntion is ruled or determined by the indicators that something is amiss or something has gone wrong. These indicators are data feeds or sources which can reveal information about an intrusion or abnormal activity. Some of the well known network based indicators of compromise have been highlighted below.

- **Ipv4/Ipv6 addresses**: This is an useful source of information and usually the first looking point whenever an intrusion is detected.

- **.x509 Certficate Hash**: These are useful in detecting if some malicious or "weird" domain was accessed by any actor in the network. These hashes often are looked against blacklisted domains and identifying botnet activity or tunnels being used as protocoln abuse to CC centers of threat actors.
- **Flows:**Flow data and short flows can be used to identify network scans and attacks as we will see later.
- **URLS:**Malicious URLS are often revealed in the network data, and useful to look at in NSM models.
- **Traffic Peaks**: Unusual peak in DNS traffic can indicate a potential DDoS attack, any abnormal peak in network traffic data is a very strong indicator of something amiss.

Some of these indicators are identifiable by using Flow data, other are identified by PTSR data. The intrusion detection systems come loaded with the common indicators of compromise. Zeek is highly customizable in this regard, and can be used to define custom compromise detection methodologies.
The indicator and signature used to detect compromise must go a continuous and dynamic process of evolution and update. Attacks and threats change, and signatures become irrelevant with time. An attack that work against BSD systems in the 1990's on a network with no such devices will not be a priority. The signatures when introduced into NSM models will be immature first,and later over time they become mature after reviews or detecting events before they are retired.

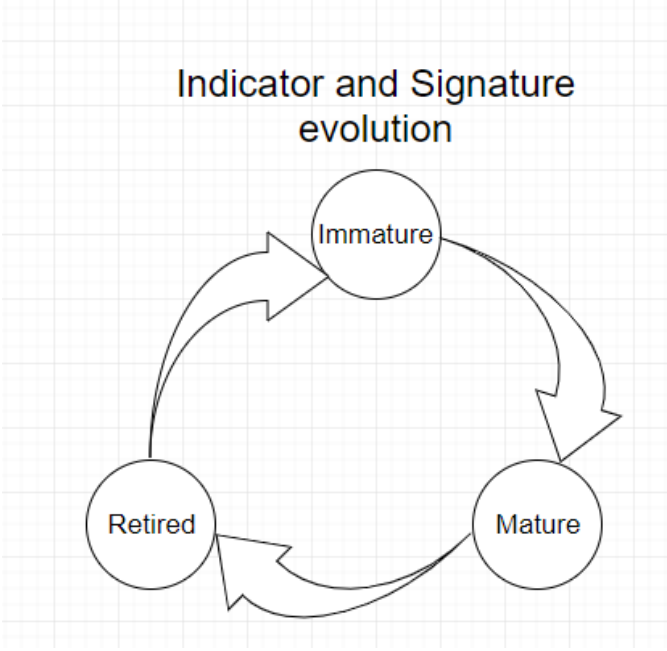

Figure 5.1: Signature and Indicator evolution

5.4 Signature based detection

The indicators are further divided into static and dynamic indicators. Static indicators can be further subdivided into **Atomic, Computed and Behavorial**. The static indicators are predetermined events or outcomes that are detected based on defined rules. These are used to form static indicators of compromise.

While easy to react to or detect, the static indicators can be often fooled by obfuscation. While defining detection based on signatures it is important to build them to be as generic as possible. There are metrics in any detection mode which determine the efficiency of that signature. Signatures need to be tuned to make them effective and balance these metrics to be favourable:

1. **False Negatives:** Detecting an anomalous event as normal.
2. **False Positives:** Detecting a normal event as anomalous.
3. **True Positives:**Detecting an anomalous event as anomalous
4. **True Negatives:**Detecting a normal event as normal

These are metrics which are used to measure the effectiveness of real-world firewalls, from the above 4 metrics the factor that determines the effectiveness is called **Base Rate Fallacy**. This is a conclusion or false alert which arises when the fall in accuracy in detecting an event that occurs seldom can lead to a huge bias in the outcome. Here the keys are:

- **P(I)** Probability that there is an intrusion.
- **P(A)**: Probability that there is an alert.
- **P(A|I)** True Positive rate (Alert given there is an intrusion)
- **P(A|!I)** False Positive rate (Alert given there is no intrusion)
- **P(I|A)** Given there is an alert , probability that there is an intrusion (crux metric)

An NSM model that generates too many false positives and floods the network administrators over nothing is as ineffective as the one which generates no alerts and lets events go unnoticed. Hence a good NSM model has signatures that have high **P(I|A)** or in simple terms when it generates an alert, it has a high chance of being correct.

$$P(I|A) = \frac{P(I)*P(A|I)}{P(I)*P(A|I) + P(-I)*P(A|-I)}$$

Figure 5.2: Base Rate Fallacy

5.4.1 Attacks Covered

To set up a signature based detection model , it is important to determine the kind of attacks and protocol abuse needs to be detected. Each attack has a different kind of signature or footprinting, and each kind of attack requires a different level of granularity of information in the packet data. The attacks which were targeted for detection were as follows.

- **Network Port Scans:** Kind of attacks which involves ping sweep or automated port scanning of machines. These include full connect scans, syn scans, and any other network scans or ping probes.
- **Protocol Bruteforcing:** Any application layer protocol which was being used for bruteforcing can be blocked, FTP password bruteforce attempts or SSH password bruteforce attempts.

- **Website Scanning:Bruteforcing:** Website scanning or crawling including those used by automated tools over HTTP protocol are blocked. This method can be used to protect HTTPS services by using methods similiar to those mentioned in Protocol bruteforcing but have not been included in this implementation.
- **Protocol Abuse:** Side channel attacks which are termed as protocol abuse by Zeek are marked. These include any protocols which may be used to transmit information in an out of channel manner.

The signatures are generic and can be extended to cover similar attacks which are likely to display a similar flow pattern.

5.4.2 Detection Signatures and Sources

In the collection phase we collected data from 3 major sources namely **Flow**, **Packet String** and **Full Packet Capture.**. Using inferencing based on these data sources we can prevent a number of attacks described above. The inferencing would give out output based on malicious IP addressses which will be sent over to the action module. The signatures may be drawn from flow based or PTSR based data. These are detection signatures and pattern that can be detected with automation and do not require further analysis or "**Human eyes**".
In the analysis phase, there will be certain feedback received from the network administrators that will either be added as a new static signature or manually prevented by the action Module. The signatures are detected for the various attacks as well.

5.4.2.1 Flow Based Detection

These attacks can be detected by analysis of Flow data with reasonable accuracy. These are mostly network scan attacks. Using a technique of Connection Failure ratio and a strategy of flow length analysis malicious activity may be analyzed. The strategies are described at length below.

- **Connection Failure Ratio:** A legitimate device will connect successfully to more devices than it fails beyond a certain threshold of attempted connections. Eg: A legitimate device will try to connect to services which are more often than not available on the destination machine. Based on this a metric can be obtained called connection failure ratio, which is nothing but the proportions of failed connections of a device compared to the successful ones. A device on the network performing scanning would be an outlier in this regard as it would have a very high connection failure ration.

$$ConnectionRatio(x) = No\_of\_failed\_connection\_flows/No\_of\_total\_connection\_flows \quad (5.1)$$

  With PTSR data, these findings may be further verified.

- **Shorter Flows:** Another feature that may be used suitably to determine a network scan activity is flow size. A scan produces shorter flows as it does not maintain a connection, and from empirical data analysis this is anywhere between **40 - 50 bytes**, and typical legitimate flow size is much larger (hundreds of bytes). A large number of short flows indicate a malicious scan activity. These results were visualized on **Kibana dashboard** and shown below. The IP address with the green column was performing the network scans and clearly the outlier.
- **Malicious User agents:** Agents which show up in the logs having definite nmap strings are not uncommon. Nmap scans always shows up in PTSR as logs in http and different application logs making them easy to identify and blacklist. More versed adversaries however will be able to mask this using custom scanning scripts with forged agent names.

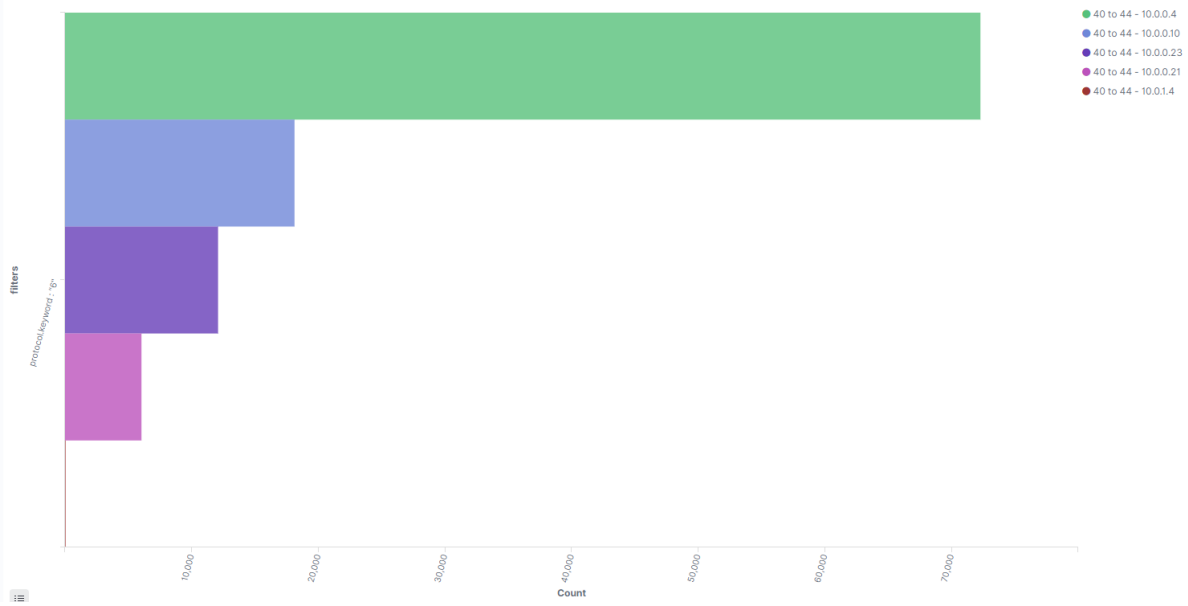

Figure 5.3: Network scan flow size count top 5 (size 40-45 bytes)

- **Unsuccessful Authentication Flow Size:**Encrypted protocols such as SSH have their payloads hidden from the sensor, but the size can reveal the authentication attempt to be successful or not. The successful SSH login usually is followed by a banner which is much larger in size, and the failure results in a short string failure message. Using a tool like justniffer or zeek, it was identified that an unsuccessful login to Ubuntu Machine produces a flow entry of about **(50-70 bytes)**, whereas a successful produces in upwards of **300 bytes.** While a single failed attempt may not be suspicious, multiple attempts in a short succession can point to an automated login bruteforce attack.

5.4.3 PTSR Detection: Harnessing the power of Zeek

While the flow data is useful in detecting some attacks, it falls short in detecting many more as majority of details of the network conversation is lost in the summarization. The PTSR data produced by zeek is the right balance in between which can not only be customized per use but provides just the right amount of information for detecting interesting events. Here some of the logs are described and how they can be used to identify malicious activity in the network.
Zeek logs are useful as they were designed with a security mindset exposing the headers and information about the network activity that can be of use by anomaly detectors.

- **conn.log:** These are more detailed version of flow data, and can be used to identify network scans or ping sweeps like flow data
- **notice.log:**: Based on heuristics and custom zeek scripts, notices are events that are a notch above usual logs, usually containing only the information that needs attention. Zeek is able to detect ssh bruteforce based on heuristic size analysis as described in the section above. Below is an snippet from the actual incident logs.
- **http.log:**These are http logs with response code data with simialr granularity as released by justniffer. The response codes are key here. Similar to the notion of connection failure ration in the previous section,

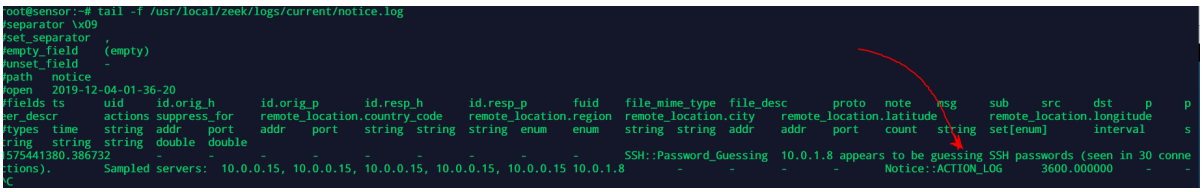

Figure 5.4: SSH bruteforce raises a notice right away(notice.log zeek)

we can detect webscans using the same analogy. A legitimate device will have the response code of 20X and 30X much more than occurrences of 40X (not found) or 50X (failed authentication). Using this we can determine which are malicious actors. Just a single minute of **dirbuster scan pushes the weight heavily in favor of 40X codes with 98% of response codes being 404**. These are used to zero in on actors performing webscans. The visual analysis explains the findings. Here the **green** represents a

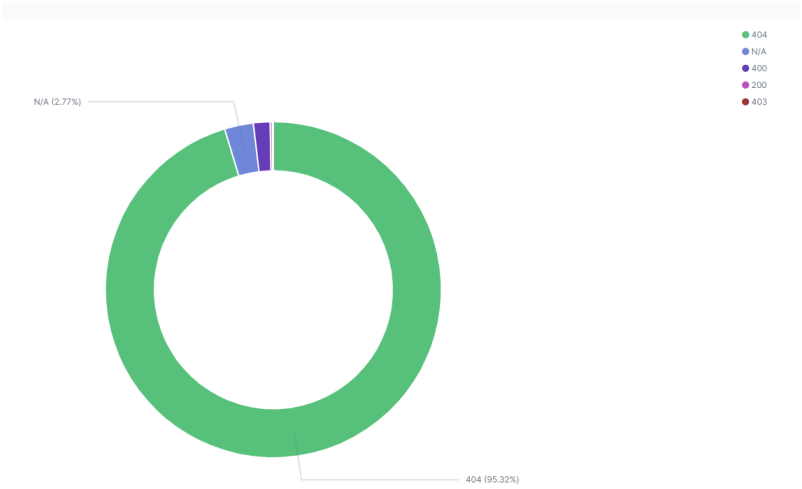

Figure 5.5: Web-scan actors have heavily anomalous ratio

404 response code ratio which is way too high to be coming from a legitimate user.

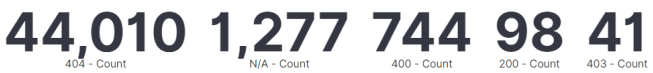

Figure 5.6: Web-scan actors have heavily anomalous ratio (404 codes are >44k while 200 codes are < 100

- **weird.log:**These get filled whenever there is a inappropriate protocol transaction which could be weird unexpected data in payload, unexpected reuse of connections (used during network scanning) and IP addresses here are more often than not threat actors.

- **ssh.log**: SSH bruteforcing as mentioned in notice.log can be detected when the attempts are increasingly bruteforced. But what if they are suppressed? These can be determined again by similar crests of ssh attempts by a device signalling that the attempts are automated. For eg: A device A makes 45 ssh attempts towards B every 3 minutes, is something that won't be exhibited by legitimate users very often and would need closer inspection.

Figure 5.7: Periodic SSH patterns indicate automated attempts even under suppression

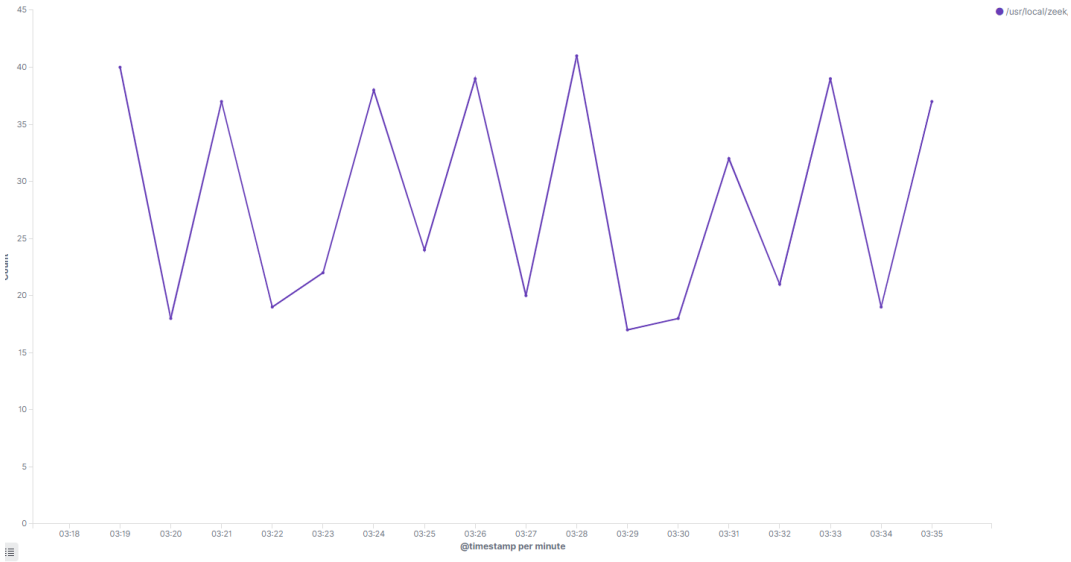

Figure 5.8: Periodic SSH patterns indicate automated attempts even under suppression

- **dns.log**: The dns traffic in a network of usual activity is usually on the lower side as the dns caching is

in effect. A rise in DNS traffic above the average could indicate a DDOS attack initiation on the source IP addresses. A screenshot taken from the **Kibana** dashboard shows the anomaly .

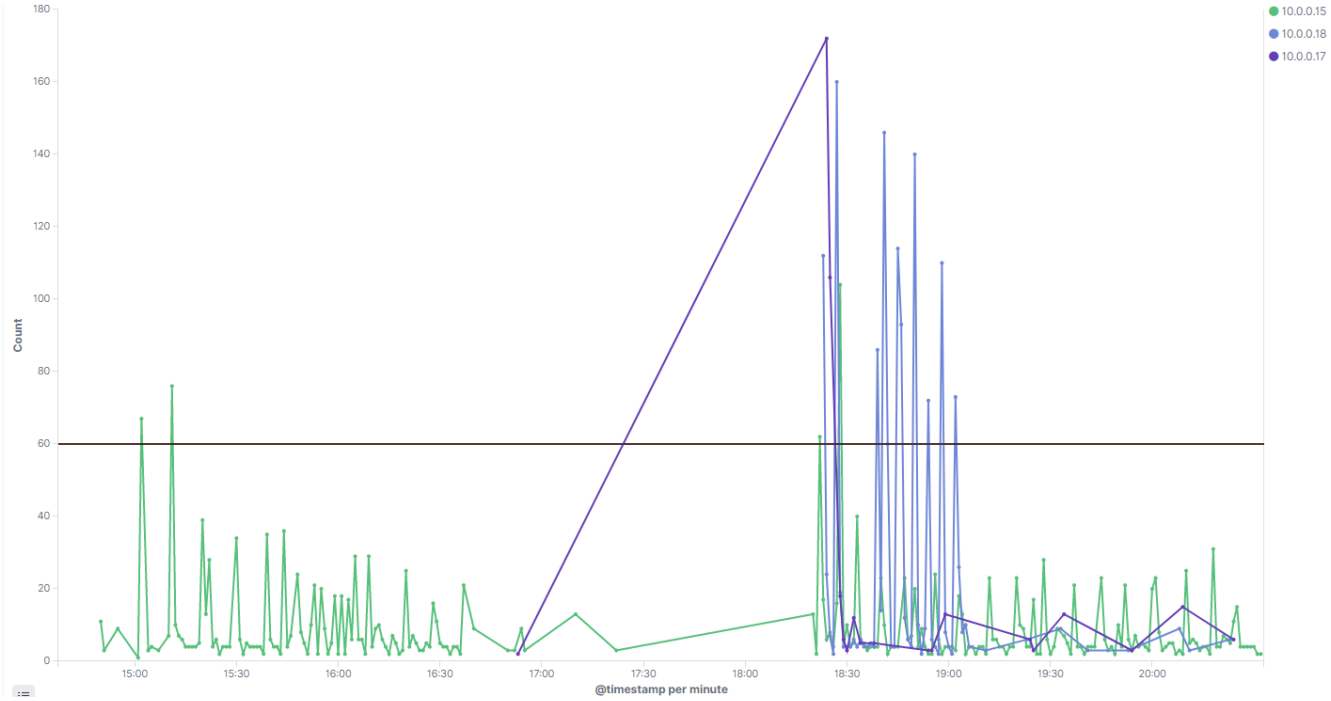

Figure 5.9: Surge in DNS could indicate DDoS attack on SrcIp's

There are other anomalies which emphasize the above findings. Any IP in the**weird.log** should be monitored and acted.

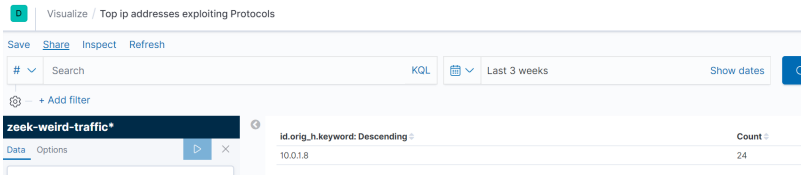

Figure 5.10: Weird IP , logged here is a malicious actor performing network scans

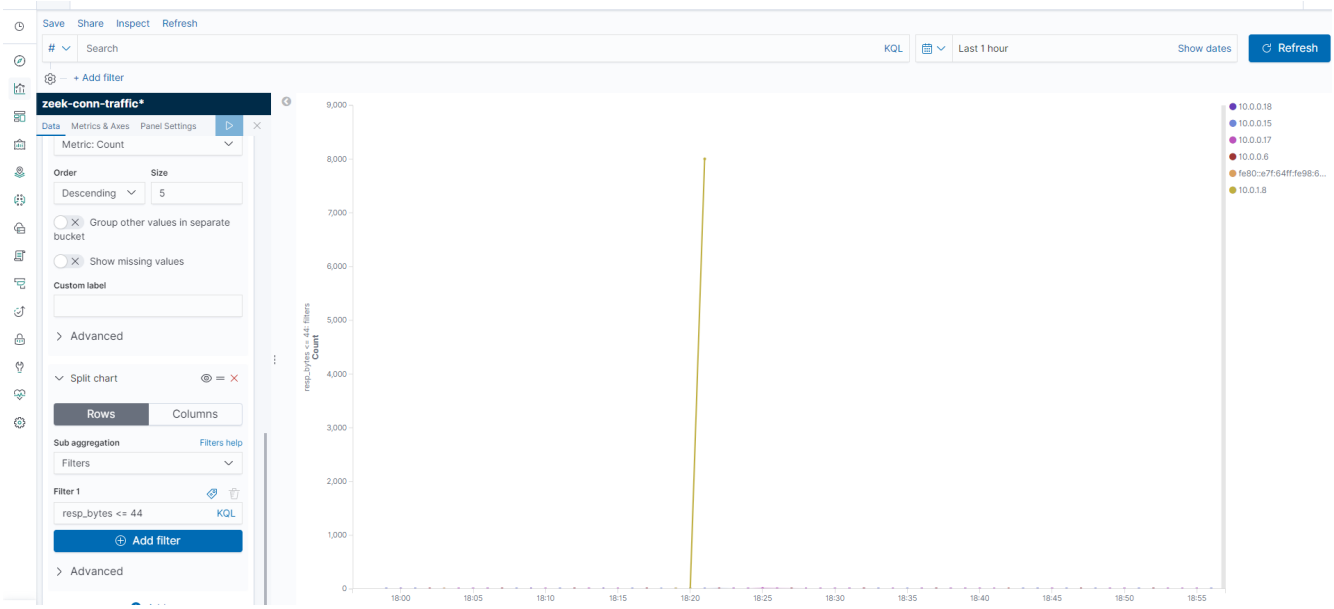

Figure 5.11: Network scan causes the malicous actor to have extremely high no of short flows(kibana)

5.4.4 Performance and Resource usage

The resource usage during detection has been elaborated in this section. The zeek uses around 300-700 mb of RAM on a network of around 10 devices at peak with traffic flowing upto **300 Mb** per minute. This is reasonable resource consumption and working on a hw with max ram **6gb** , it does not stress the system at all. The logs are generated instantaneously by zeek, but there is latency between the time the logs are generated and filebeat pushes it to elasticsearch via Logstash. The entire pipeline of **filebeat**, **logstash** and **Elasticsearch** takes around **5-6s**. This is a reasonable amount of time to detect attacks and stop them. However, the pipeline is not a factor on automated action, only the analysis part which requires human intervention. The action modules sit on the sensor and are able to prevent attacks within seconds of successful detection. Since zeek generates PTSR logs fast, most of the bottleneck in latency is squashed.

5.5 Reputation/Anomaly based detection

SInce most of the implementation is based on a scenario of internal network, there is not a significant role , reputation based detection plays here. However, this functionality may be easily extended by using custom zeek scripts for reputation based IP blacklisting or looking up malware in the network by comparing hashes in the **file.log** with the online malware hash list such as **virustotal**. A combined list of file hashes, certificates in x509.log and source Ip addresses can be verified against online list by zeek automatically or custom modules can be written in zeek to achieve the same.

6. Phase 3:Analysis

This is the phase where the detected events are scrutinized further and categorized. The phase usually refers to the phase where humans look at the detected events and decide the priority of the event or whether it classifies as a false positive or a false negative. This is also the phase where the signatures are improved or retired based on the accuracy of the results. The results or logs which are not real time but viewed in retrospection on a monthly or a weekly basis. This is also the phase where it is decided how the NSM processes events or in simple terms , *What to do with the events detected?*.

6.1 Automated Analysis:

Based on static signatures defined in the previous section, automated programs check the respective data source continuously for a match. Based on network size, there is some amount of fine tuning required to correctly identify threat actors, especially those signatures which check against an allowed threshold of traffic for identifying what is normal and what is not. These signatures could be split into programs which continuously scan for malicious actors in the real time and retrospective logs and send them to a feed which will be fed into the **action** module (Intrusion Prevention). For isolating these signature watchers , it is better to run them as separate processes for efficiency and a more scalable design.
The choice of programming language for this implementation was **Python 3.x** due to its powerful scripting abilities. Each python worker is a thread in the background that runs and acts on the identified actors by itself. The action module is orchestrated with the automated workers to block threats in real time. The architecture for this in the implementation has been elaborated in the next section.

6.1.1 Automated Analysis and Prevention: Implementation

The choice of technology for implementing Prevention is follows.

- **Python:Monitors** These are worker threads that run in background locally on the sensor, but can run anywhere else as long as they have access to the log feed and the boundary router. These implement the signature detection discussed in the last modules and read the log files for changes independent of each other. Each of these workers runs a different kind of signature detection algorithm.
- **Ansible:Implementors** Once the signature determines an IP address to be a threat actor, the next step is to contact the boundary router of the victim network and make changes to the firewall rules. While a script could be written easily to SSH , ansible is an open source tool that automates remote management of Linux devices with ease, having various modules configurable by using commandline or a simple YAML playbook. The ansible module accesses the firewall rules of the remote routers and DROPS all packets in all the policy chains which may be **INPUT**, **OUTPUT** and **FORWARDING**. The ansible script may also be used to look up the DHCP table for the mac address associated with the IP address to further identify the threat actors, but mac addresses are easily spoofable and do not hold much value in the context of intrusion prevention, and hence have been left out.

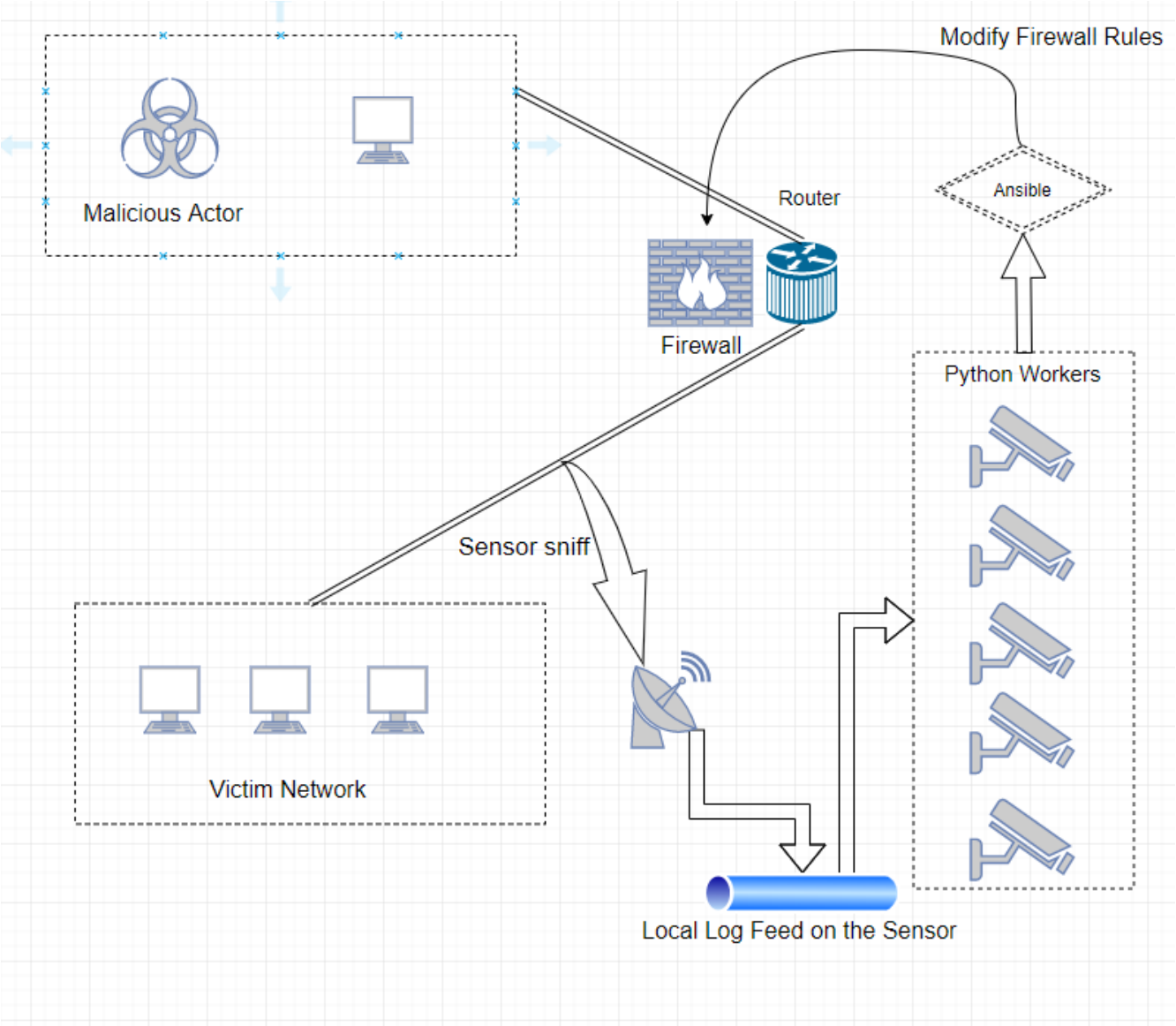

Figure 6.1: Action architecture overview

6.2 Human Analysis: NRT Dashboard

There are some signatures which needed to be developed over time , and created after human analysis. These may be observed only after detailed analysis of monthly data and difficult to detect by static signatures in the first go. These may be produced after running Machine Learning algorithms over the data feeds or by a network analyst who notices a malicious pattern or suspicious activity which is not being caught by the automated python actor.
The differentiating factors would be the NRT dashboard would need to look at logs over a larger timeframe or in retrospection unlike the **tail/live mode** used by the python workers. The larger timeframe allows for

deeper analysis and hence a lot of visualization and data analysis tools come into the picture here. There are several possible choices that can be used here but for this implementation the open-source **ELK Stack** with the visualization tool Kibana is used.
Kibana provides a plethora of analysis tools and visualization which can reveal a lot of insights into network data. Kibana even provides custom support for SIEM utilities, Zeek log plugins and in built Machine Learning algorithms (Enterprise edition) for anomaly detection, Timeseries analysis and organizing the different results on a dashboard. Some of the previous sections have these visualizations present, and some other have been presented below.

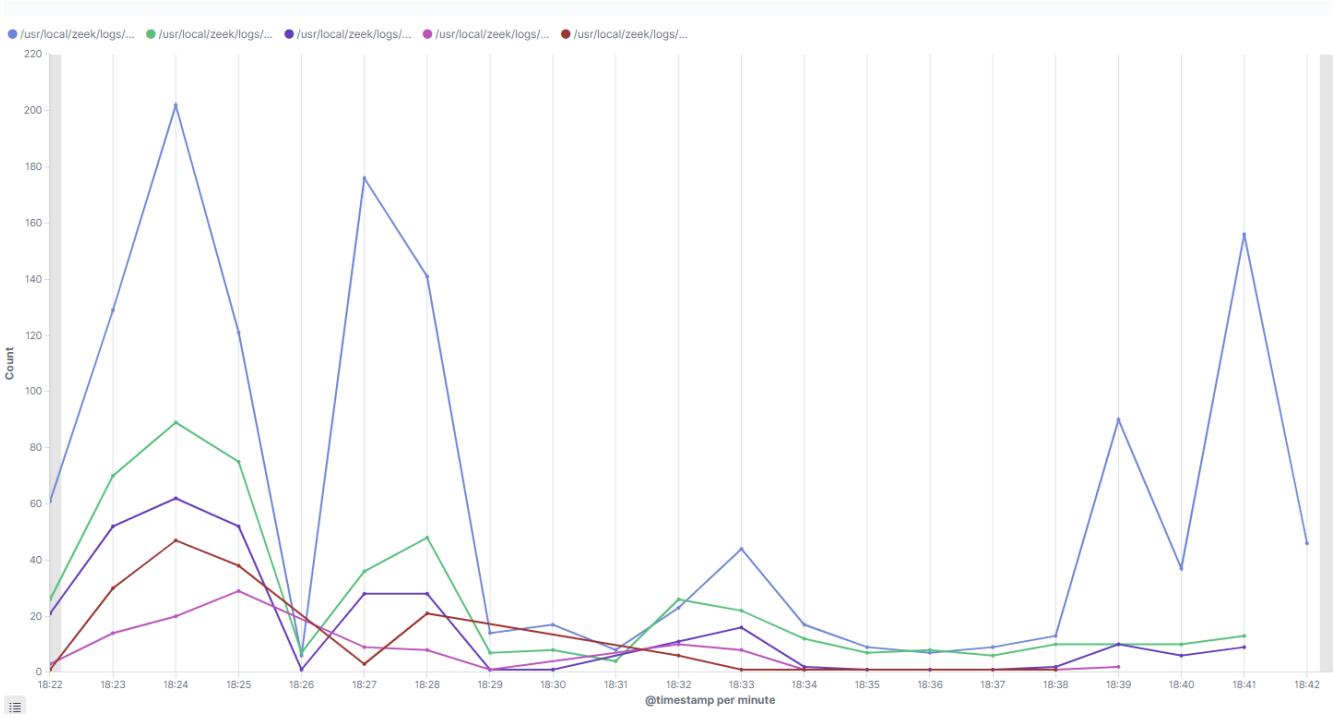

Figure 6.2: Traffic split by type

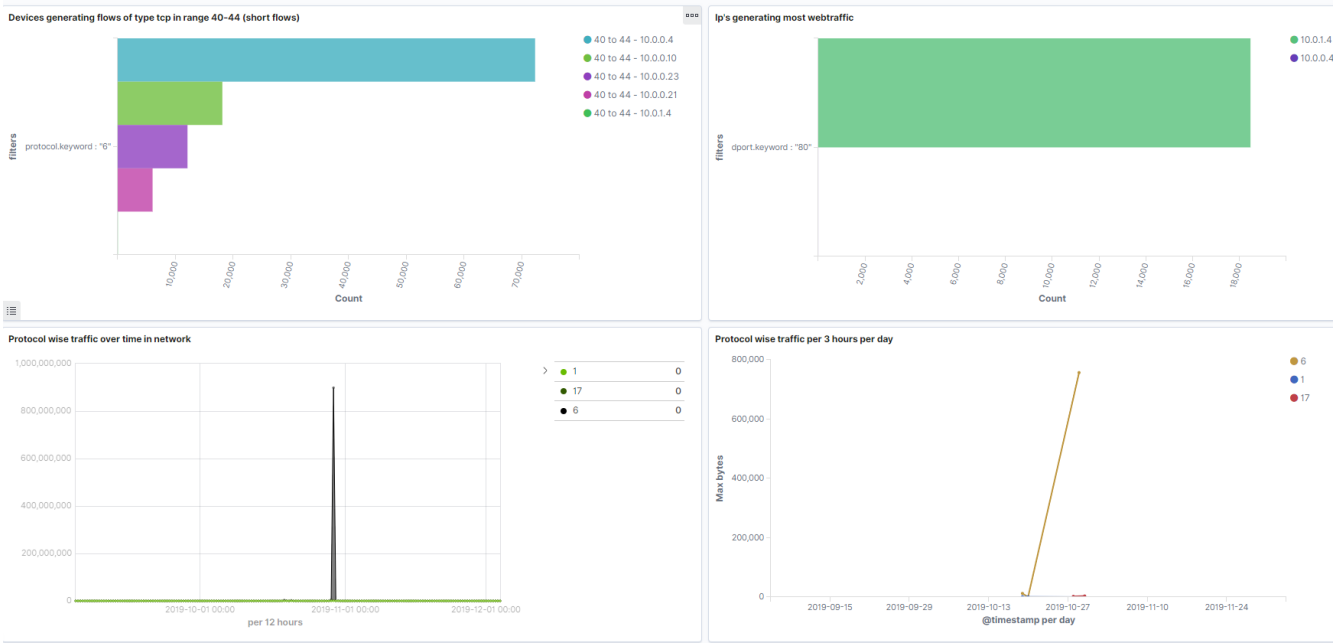

Figure 6.3: Dashboard example

6.3 Efficiency of Implementation

This section attempts to identify the efficiency of implementation based on several metrics. These are enumerated and elaborated on in the subsections below. The implementation put in place was simulated by placing an attacker with Kali Linux or Debian Distro with the latest enumeration,exploitation and reconnaisance tools at their disposal. The attacker would then attack the systems with different kinds of attacks and switch between aggressive attacks and stealthy suprressed spaced out attacks or scans to check the aspects of implementations. The attacks that were carried out are described in the previous section 5.4.1

6.3.1 Base Rate Fallacy

The implementation put in is tested against aggressive and stealthy variants of the attacker and the results observed by the implementation are presented below. The heuristic based approach used by zeek in detecting ssh bruteforce **is often evaded by suprressed or spaced out attempts**, but is caught in the crest detection fueled by the analysis in the visual analysis dashboard. However, when it does produce an alert it is accurate (good base rate fallacy)

Attack Type	BFR	False-Positives	False-negatives
SSH Bruteforce Zeek (Aggressive)	99%	0%	1%
SSH Bruteforce Zeek (Passive)	92%	0%	5%
Web Scan (Aggressive)	100%	0%	0%
Web Scan (Passive)	100%	0%	0%
Network Port Scan(Aggressive)	99%	0%	2%
Network Port Scan (Passive)	91%	3%	2%
Protocol Abuse	100%	0%	0%

The heuristics based approach is used where encryption is in place and the flow size is where the decision is based off of takes a hit when suppressed techniques are used.

6.3.2 Response time

The python workers have varying response time based on the kind of attack being prevented and responded to. Since the network is small the response time is fast and may be a poor indicator of how large networks and high speed links may behave. The delay in response time of scans is mostly due to a minimum no of flow threshold (set to 50 for this network) before it being considered for judgement or action. The response time where **t=0 was when the attack initiates** and **t1 is the time where threshold connection limit is passed**. The times are measured using linux time module.

Attack Type	Time of min flows (ms)	Time of prevention (ms)
SSH Bruteforce Zeek (Aggressive)	200	300
SSH Bruteforce Zeek (Passive)	1200	1300
Web Scan (Aggressive)	100	120
Web Scan (Passive)	1400	1490
Network Port Scan(Aggressive)	1000	1200
Network Port Scan (Passive)	2000	2200
Protocol Abuse	4020	4220

The response time is affected mostly due to the threshold factors which are taken to reduce the number of false positives. However these should be drawn according to the size and scale of network and the pattern of normal traffic.

7. Future Work/ Design Improvements

Since this was undertaken in a span of 2 months of a graduate course, there are several areas which could not be covered with substantial depth in this undertaking. The critical baseline skills required and the tools for flow collection, parsing, storing and shipping etc also had to be devoted a large slice of the resource. This section highlights the future work and improvements in this hands on version of "*Getting Started in Network Security Monitoring*". There are several scalable design changes that need to be made to make the implementation scale out better in practical larger networks. These improvements and intended future work are touched upon in this section of this report.

7.1 Analysis Feedback Mechanism: Honeypots

Due to the time constraints, this implementation leaves out a key and central aspect to network security monitoring and these are Honeypots. The backbone of any adaptive and dynamic NSM model is to continuously learn from recent attacker activity and feed it back to the analysis engine of the implementation. The honeypots are designed to do exactly that, a honeypot is usually designed for a particular service and to lure in attackers whilst not revealing their true purpose. The honeypots lure in threat actors on the network by appearing as the lowest hanging fruits, and once the attacker bites in, they are designed to log all attacker activity.
Honeypots are an indispensable way of learning from attacker activity by monitoring strategies , patterns and then using it to train the detection and analysis models , further hardening the NSM implementation.

7.2 Scalable Zeek Architecture

The zeek mode used in this implementation is the standalone mode which is the easiest to configure and get started. There is a cluster configuration mode for zeek which is used in large networks and sensors running and referring to a single zeek instance. This is a huge improvement in centralizing logs and leading to faster and improved co-relation without any additional external effort. The clustered mode of zeek has workers assigned to different interfaces and IP's which fetch logs to the zeek managed instance. The diagram below is an overview of the clustered architecture of zeek.

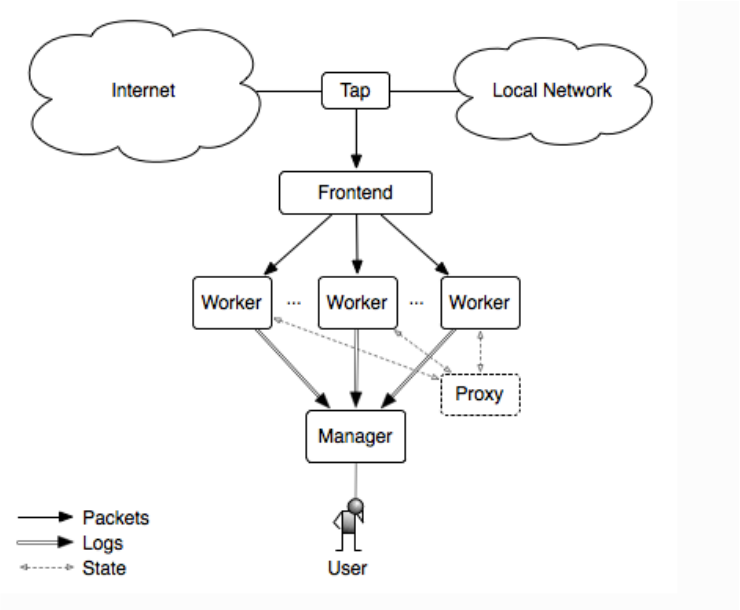

Figure 7.1: Zeek cluster configuration

7.3 Reputation Based Detection Integration

Reputation based integration is another aspect that can be extended for the NSM implementation. This is partially done in the implementation by the sensor and zeek taking frequent updated to check and update the virus definitions, and implement IP level blacklisting . The reputation based detection could be fed the results taken from the setup honeypots across the network and further strengthen the analysis results.
It would also be beneficial to pull these data from major sources and running them by the network using a packet replay method to further test the efficiency of implementation.

7.4 Testing and Simulating enterprise-level attacks and tools

The attacker in this scenario was manually simulated, the further extension would be to simulate multiple attackers on a simulated busy network using enterprise grade stealth techniques or atleast those used by pen-testing professionals to attack the network and compare the results obtained. There are tools such as **Infection Monkey** which test network resiliency could be used to check the network security being implemented, but before that is done other types of attacks need to be covered as well

7.5 More Attacks

There are several attacks which the future work on this will cover, which were not mentioned here and these are enumerated in this section.

- **Routing Attacks:** Attacks on Protocols such as OSPF and RIPv1,v2 are not covered in this implementation.

- **SMB/Netbios/SNMP** Attacks based on exploiting SMB shares , not covered explicitly but can be extended by testing for brute-forcing , and enumeration can be checked by looking for specific strings in signatures.
- **Layer 2 Attacks:** Layer 2 attacks on bridge protocols, switch authentication protocols and cam flooding were not covered in this scenario.
- **Host Based attacks:** Since this focused only on the network based attacks, host based attacks were not covered in this implementation.
- **Malicious Insider**: The case where the attacker is in the same subnet as the victim is not covered. Though nothing would change on the detection phase, as the sensor can detect the attacker just the same as the one acting from outside the subnet, prevention strategy would definitely change here. For such an adversary, the hosts in the network must be instructed to block using their own host based firewalls but again that would be infeasible on a large network. This is an area where further work needs to be planned for this implementation.

8. Conclusion

Network security monitoring is the central cog of network security. It is a complex process involving a lot of different micro components that need to be fine tuned to orchestrate a smooth, fast , adaptive and scalable implementation. For an organization to have a hardened NSM model, it must go through thorough and rigourous designs, redesigns and reviews before arriving at an optimal implementation. The dynamic nature of the threat that a network faces adds to the challenge of the task at hand, and an organization must either outsource or have a dedicated team of professionals in this area who are continously trained to maintain the NSM and modify it when the need be. The Application Collection Framework and Collective Intelligence frameworks are some which provide good reference points in setting up NSM for a given network. From deciding sensor hardware and placement, to deciding algorithms that detect threats faster, to building a scalable architecture that can adapt to changes well- A network security analyst is expected to know a vast array of skills proficiently.

As a guided project it was a thorough learning experience for me under Prof. Thomas Reddington who provided invaluable guidance along the way, and warned me of the pitfalls. I feel that this is a first stepping stone to a steep ladder of a network security professional , and I am motivated to continue my study and work to learn more about the same. With the dearth of cyber security professionals in the industry , the skill of a proficient NSM analyst is indispensable to an organization looking to secure and monitor its network.

Bibliography