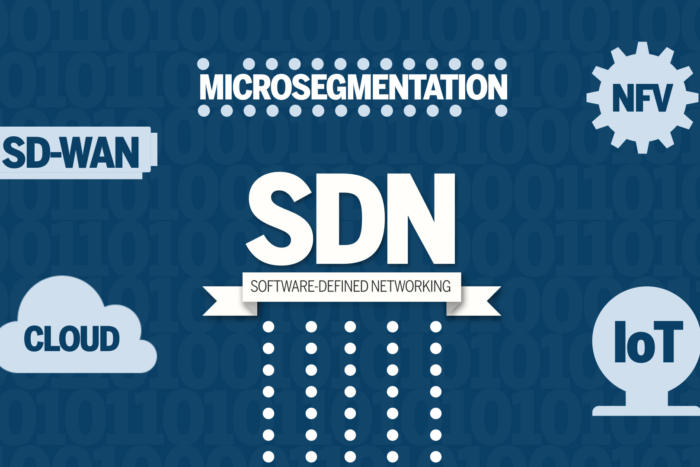
**Implementation of Real Time Intrusion Detection using ONOS**



**Project Team**

Stephen Blystone (smb032100)

Taniya Riar (txr170430)

Juhi Bhandari (jxb161830)

Ishwank Singh (ixs170930)

Table of Contents

[Introduction 3](#_Toc512116867)

[Problem Statement 3](#_Toc512116868)

[Architecture 4](#_Toc512116869)

[Tools Used 5](#_Toc512116870)

[Implementation 6](#_Toc512116871)

[Traffic and Network Generation 6](#_Toc512116872)

[Big Data Framework 8](#_Toc512116873)

[Big Data Pipeline Architecture 8](#_Toc512116874)

[Elasticsearch Index Creation 9](#_Toc512116875)

# Introduction

The project focuses on implementing Intrusion Detection using ONOS controller in real time. It is divided into following parts:

* Creation of a miniatured network to simulate traffic which is controlled by the ONOS Controller.
* Extraction and loading the real time traffic log using NetFlow on a Big Data Architecture for further visualization and processing.
* Implementing statistical models on the processed data to detect anomalies in non-signature attacks.
* Sending control to ONOS to block traffic from the anomaly device

## Problem Statement

There are many commercial and open source software and tools available for intrusion detection which focuses mainly on signature attacks. With the increasing popularity and implementation of Software Defined Networks and powerful controllers, we hosted a customized application on them to perform intrusion detection for non-signature attacks. The real-time implementation will detect the attack immediately and hence blocking the traffic from the anomaly device via ONOS Controller.

For the anomaly detection, the project focuses on processing the NetFlow data from the network. The traffic log data generated by the network in real time is in terabytes. This invited the need of storing the data in some data storage and retrieval application. Since, the data is semi-structured and Big Data, we had to include the Big Data architecture to process and visualize it.

The network data used to implement our project is simulated in a miniatured network setup which includes Database server, application server, Web server, nine hosts and ten switches. We have five switches NetFlow enabled.

The network log data is then pipelined to the Big Data architecture for processing and implementation of statistical model. The idea is to train a Machine learning model to learn for the anomaly pattern in the log data and take a “blocking” action when such attack is recognized. The motive is to lay a foundation for a such kind of automation system.

## Architecture

The project includes three important parts:

1. ONOS Controller – It is hosted on a Virtual Machine and communicates with the miniatured network and the Big Data Infrastructure.
2. Miniatured Network – It consists of hosts, switches and the servers. Few of the switches are NetFlow enables and thus connecting to the Big Data Infrastructure.
3. Big Data Infrastructure – It consists of Stream Sets which connects the NetFlow switches via a UDP port to Apache Kafka. Further, it connects to Data Feature and Machine Learning module consisting of Apache Spark and Elasticsearch Data Storage. Kibana is then connected via Elasticsearch for the visualization of the real-time data.

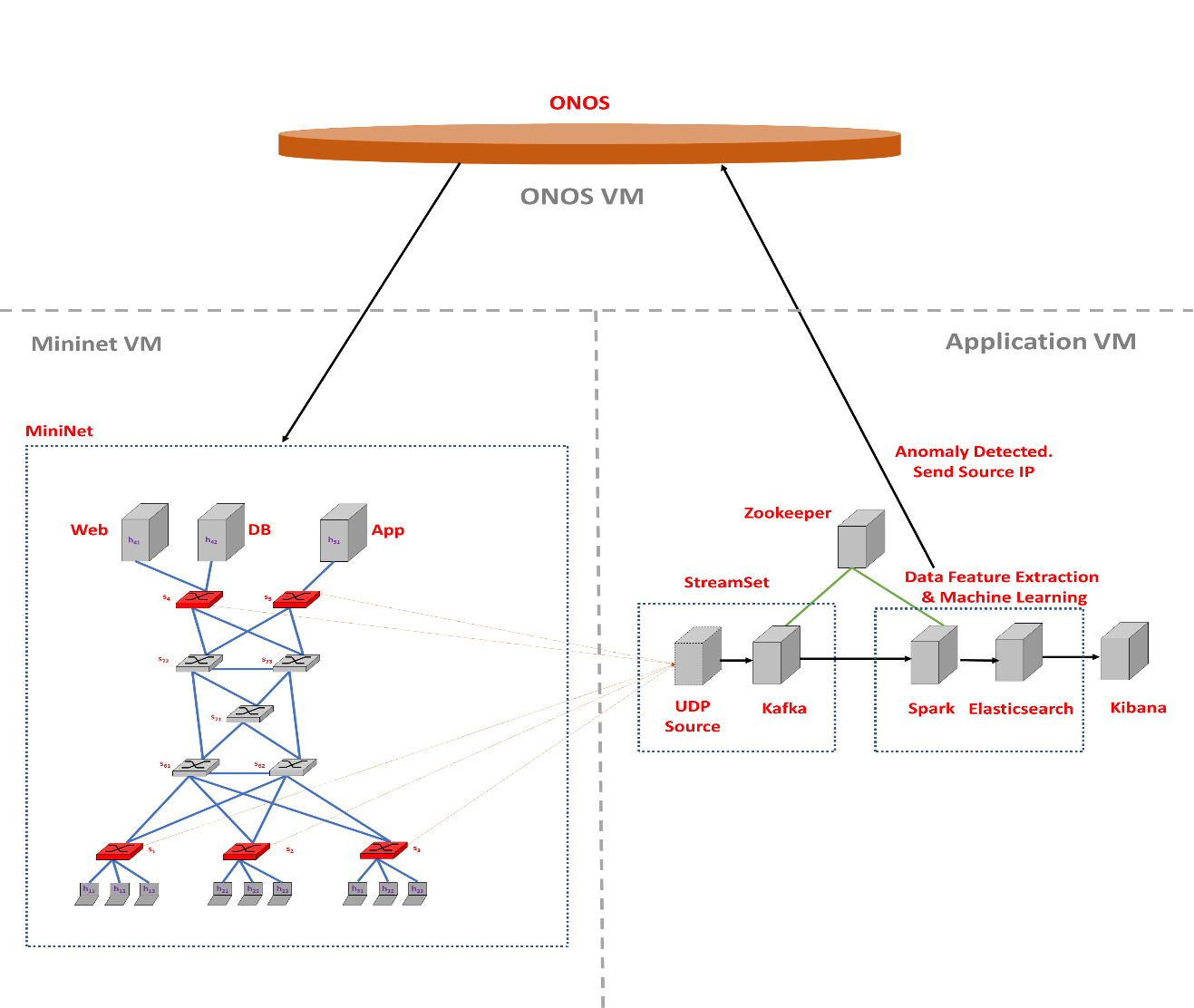


Figure 1 The project setup architecture.

## Tools Used

* ONOS Controller – ONOS stands for Open Network Operating System. ONOS provides the control plane for a software-defined network (SDN), managing network components, such as switches and links, and running software programs or modules to provide communication services to end hosts and neighboring networks.
* MiniNet – MiniNet is a software emulator for prototyping a large network on a single machine. It can be used to quickly create a realistic virtual network running actual kernel, switch and software application code on a personal computer. It allows the user to quickly create, interact with, customize and share a software-defined network (SDN) prototype to simulate a network topology that uses Openﬂow switches.
* NetFlow Collector – NetFlow is a network protocol developed by Cisco for collecting IP traffic information and monitoring network traffic. By analyzing flow data, a picture of network traffic flow and volume can be built. Using a NetFlow collector and analyzer, you can see where network traffic is coming from and going to and how much traffic is being generated. Routers and switches that support NetFlow can collect IP traffic statistics on all interfaces where NetFlow is enabled, and later export those statistics as NetFlow records toward at least one NetFlow collector - typically a server that does the actual traffic analysis.
* Stream Set – Stream Sets is a cloud native collection of products to control data drift; the problem of changes in data, data sources, data infrastructure and data processing. The company calls its applications a data operations platform. Included features are a living data map, performance management indices and smart pipelines providing a similar level of control to common business operations systems.
  + Apache Kafka – Apache Kafka s a distributed streaming platform. It is horizontally scalable, fault-tolerant, wicked fast, and runs in production in thousands of companies. Kafka is generally used for two broad classes of applications:
    - Building real-time streaming data pipelines that reliably get data between systems or applications.
    - Building real-time streaming applications that transform or react to the streams of data.
* Apache Spark – Apache Spark is an open-source cluster-computing framework. Apache Spark is a unified analytics engine for big data processing, with built-in modules for streaming, SQL, machine learning and graph processing. It is a fast, in-memory data processing engine with development APIs to allow data workers to execute streaming, machine learning or SQL.
* Elasticsearch – Elasticsearch is a search engine based on Lucene. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. Elasticsearch is developed in Java and is released as open source under the terms of the Apache License. Elasticsearch is developed alongside a data-collection and log-parsing engine called Logstash, and an analytics and visualization platform called Kibana. The three products are designed for use as an integrated solution, referred to as the "Elastic Stack" (formerly the "ELK stack").
* Kibana – Kibana is an open source data visualization plugin for Elasticsearch. It provides visualization capabilities on top of the content indexed on an Elasticsearch cluster. Users can create bar, line and scatter plots, or pie charts and maps on top of large volumes of data.
* Logstash – The combination of Elasticsearch, Logstash, and Kibana, referred to as the "Elastic Stack" (formerly the "ELK stack"), is available as a product or service. Logstash provides an input stream to Elastic for storage and search, and Kibana accesses the data for visualizations such as dashboards.

# Implementation

## Traffic and Network Generation

The miniatured network topology used, is very close to a realistic scenario which includes servers and clients connected through switches. The traffic is being simulated using three python scripts. The setup\_topo.py creates the network layout. The client.py sets up the hosts and assigns IPs to the servers and the clients. The host can talk to other host on HOST\_PORT, Web Server on WEB\_PORT and App Server on APP\_PORT. The hosts do not talk to DB Server.

The randomness of the data is maintained by making the hosts talk at random time interval between 1 and MAX\_SLEEP\_INTERVAL seconds. The following rules are applied to all the hosts:

* + Choose Host Task at random by sending data (between 1 and 1024 bytes) to another host.
  + Choose host to send to at random (other than self) by pinging another IP.
  + Choose host to send to at random (other than self) by sending HTTP request to Web Server and by sending data (between 1 and 1024 bytes) to App Server.

The servers are created and managed using Server.py script. It manages the client server communications and is selected randomly to maintain the randomness in the traffic generation. We are using simplified network protocols like TCP and ICMP.

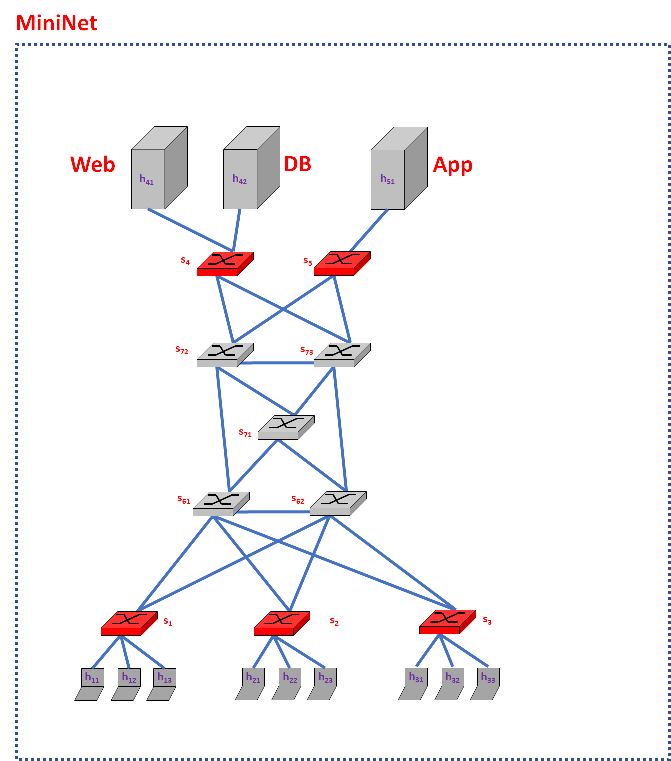
****

Figure 2 The miniatured network architecture

## Big Data Framework

The real- time data streaming form the network is a Big Data and we have setup a big data infrastructure to handle and process it. The data is being collected by the NetFlow enabled Open vSwitches using Stream Set and pipelined through Apache Kafka on the Apache Spark platform for processing. The data is stored in Elasticsearch data storage in index form for fast retrieval and visualized using Kibana.

### Big Data Pipeline Architecture

Below is a basic Big Data architecture for real-time processing from the Microsoft website. We are taking inspiration from there.

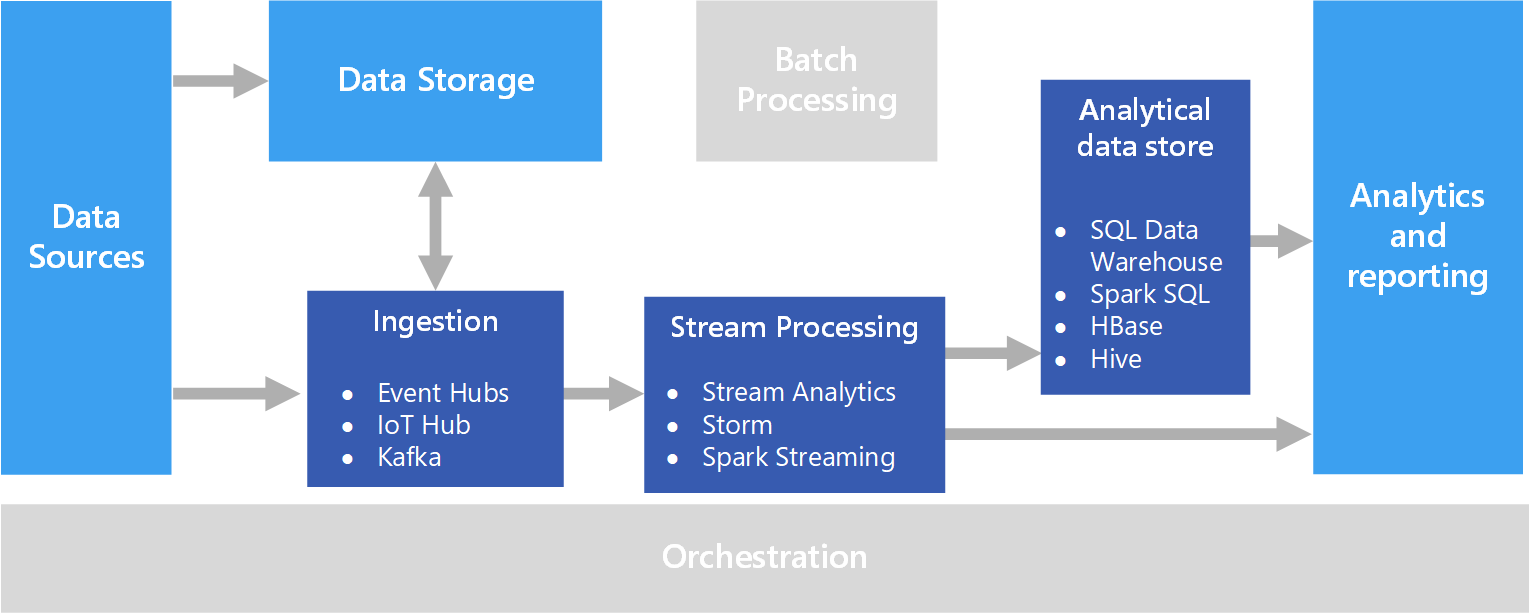


Figure 3 Big Data Pipeline Architecture from Microsoft

We will have several components in our architecture:

1. Data Sources - These are the Open vSwitch nodes sending NetFlow data.
2. Real-time Message Ingestion
   * This component essentially performs Stream Buffering, providing the “Stream Processing” block enough time to process the stream data without being overrun with data. We will use Apache Kafka.
3. Stream Processing
   * This component filters, aggregates, and prepares the data for analysis. We will use Apache Spark Streaming.
4. Analytical Data Store
   * This is where the data is stored after being processed. We will use Elasticsearch. The Machine learning part is done using PySpark.

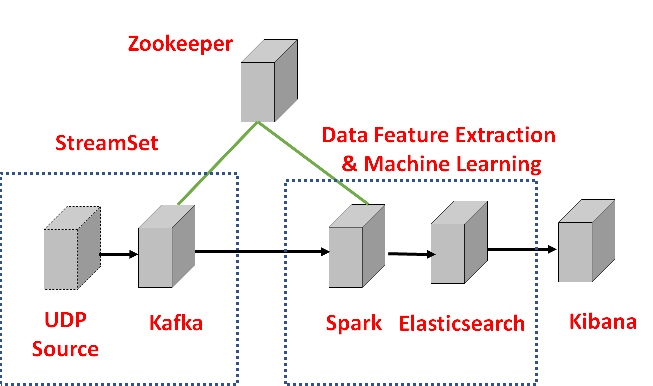


Figure 4 the Big Data infrastructure and various sub parts

We are using Apache Zookeeper to maintain configuration information, naming, providing distributed synchronization and providing group services. It is predominantly used for distributed systems.

### Elasticsearch Index Creation

Elasticsearch is a distributed document store. It can store and retrieve complex data structures—serialized as JSON documents—in real time. In other words, as soon as a document has been stored in Elasticsearch, it can be retrieved from any node in the cluster.

The following is the Index created for storage structure of the data in the Elasticsearch.

# the index created in Elasticsearch using Kibana Dev Tool

PUT /netflowrepo/entry/\_mapping

{

"entry" : {

"properties" : {

"sumOfFlows" : { "type" : "long" },

"sumOfBytes" : { "type" : "long" },

"uniqDstIPs" : { "type" : "integer" },

"uniqDstPorts" : { "type" : "integer" }

}

}

}