# Modelling BGP Updates for Anomaly Detection using Machine Learning

Daniel Robertson

***Abstract*—This report will outline the problem of detecting and visualising Border Gateway Protocol (BGP) update anomalies in real-time. Detecting and visualising these anomalies in real-time matters because it can help prevent impacts such as denial of service, service slowdown, or loss of revenue that results from accidental or malicious BGP updates. The output of this project is a system that can extract BGP update features and detect and visualise BGP anomalies in real-time, allowing network teams to minimise the effect of such anomalies.**

***Index Terms*—BGP, Border Gateway Protocol, Anomaly Detection, Machine Learning, BGP Anomaly**

## I. Introduction

BORDER Gateway Protocol (BGP) is an essential protocol that allows the internet to function. Autonomous System (AS) routers are constantly sending and receiving BGP updates, changing the structure of internet routing, and therefore the flow of traffic for the billions of devices using the internet. Anomalous BGP updates can have catastrophic effects on the function of the internet, and network teams need a way to identify these anomalies in order to mitigate damaging outcomes. This report proposes a program that can visualise and detect anomalous BGP updates in real-time, allowing network teams to minimise the effect of such anomalies.

On August 30th 2020, CenturyLink, a large American Internet Service Provider, endured a significant outage that resulted in a 3.5% decrease in internet traffic over the outage period of five hours [1]. The outage was caused by a misconfiguration. A customer of CenturyLink requested for a specific IP address to be blocked, however, wildcards were mistakenly used, leading to a larger range of IP addresses being blocked, which then had flow-on effects that caused the outage [2]. This error ultimately highlights the importance of being able to quickly detect BGP anomalies in real-time. The motivation of this project is therefore to mitigate BGP anomaly impacts by detecting anomalies in real-time, to reduce the time and impact of such outages.

### A. Project Goals

The goals of this project were to create a system that can:

1) Take live and historical BGP data as input and process the data.
2) Extract and store important feature values from BGP data.
3) Detect anomalous BGP activity using a machine learning model.
4) Display extracted BGP feature values in a graph that highlights anomalies.

This project was supervised by Winston Seah (primary) and Alvin Valera.

### B. Performance Requirements

To ensure the project goals are measurable the following performance measurements were defined:

1) The program must be able to continually process live BGP data.
2) The program must be able to produce a graph of extracted BGP feature values and mark anomalies.
3) The program must be able to identify two true-positive BGP anomaly events using a machine learning model.

### C. Product and Evaluation

The final solution of this project has the following key components:

1) Historical Data Processor - Processes historical BGP data from [3]
2) Live Data Processor - Processes live BGP data from [4]
3) Feature Extraction and Storage - Extracts features from historical and live data for use in machine learning algorithm and graphing. Stores extracted features in a CSV file.
4) Machine Learning Model - Isolation Forest.
5) Graphing - Time-series graph that displays extracted features in periods of one minute, and highlights anomalies.
6) Web Application - User interface with the system. Allows the user to start and stop the program and select the route collector [3].

The solution's accuracy and performance traits were evaluated. Accuracy was tested using historical BGP anomaly incidents and was able to accurately identify two different incidents, the 2003 Slammer worm anomaly [5], and the 2008 Mediterranean undersea cable cut [6]. Performance was tested to ensure the program could process live data for a period of 24 hours from three different BGP data sources, which it was able to.

### D. Sustainability and Environmental Impacts

The key sustainability and environmental impact of this project is power consumption. The solution of this project, a program, would need to be run constantly to provide the best detection capability. This would have a moderate power consumption, however, this can be compared to running

a server, of which there are millions running currently. This project may have positive environmental impacts. BGP anomalies cause network traffic to go to an incorrect destination, therefore the packets are a waste of power consumption. By using the program to quickly detect, and then correct, BGP anomalies, there is less power consumption waste.

This project contributes to goal 9 of the United Nations Sustainable Development Goals [7], which aims to build resilient infrastructure. This project contributes by providing a system that can allow for harmful BGP anomalies to be detected and fixed quicker, therefore improving the availability of the internet. Additionally, it contributes to goal 12 [8], which aims to ensure sustainable consumption, and goal 13 [9], which aims to combat climate change. By using this project's system, BGP anomalies can be mitigated faster, therefore reducing the increased power consumption caused by wrong traffic routing and an increase in BGP updates, which are a general impact of an anomaly. Reducing power consumption by the infrastructure of the internet improves sustainable consumption (goal 12), and helps to decrease the impacts of climate change (goal 13), of which electricity consumption contributes. The remaining 15 Sustainable Development Goal [10] are not directly supported or impacted by this project.

### E. Tools

*1) Python:* Python [11] is a programming language that was chosen for this project due to it's third-party machine learning libraries and compatibility with other project tools. There are many options to implement machine learning in a Python project, including PyTorch, scikit-learn, and TensorFlow, which allows versatility in the design. Additionally, Python is compatible with all the other project tools. An alternative language, Java, was considered for this project. However, previous work by Huang [12] was done in Python, and Java machine learning libraries are more complex to implement.

*2) Django:* To implement the web application component of the project, Django [13] was used. Django is a Python web framework for developing web applications. Flask was considered for this project, however, Django was chosen as it allows better extensibility of the project, and has more features as a full-stack framework.

*3) bgpdump:* bgpdump [14] is a tool developed by Réseaux IP Européens Network Coordination Centre (RIPE NCC) for parsing raw BGP dumps in MRT or Zebra/Quagga formats. This tool was chosen as the data from the RIPE Routing Information Service (RIS) collectors is in MRT format, and the tool was determined to be easily integratible with the system design.

*4) scikit-learn:* There were three machine learning libraries considered for this project, scikit-learn, PyTorch, and TensorFlow. Ultimitely, scikit-learn [15] was chosen for it simplicity and because it is lightweight. PyTorch and TensorFlow are powerful libraries with a focus on deep learning models, which do not align with the goals of the project. scikit-learn algorithms are easy to implement and work effectively with the smaller dataset sizes of the project.

*5) PyCharm:* PyCharm [16] is a Python Integrated Development Environment (IDE). No other IDEs were considered as PyCharm had all the features needed for the project, including scientific mode for displaying matlibplot graphs, and GitLab integration. PyCharm was also chosen due to it's familiarity.

*6) GitLab:* Using GitLab [17] was a requirement of the project, as a repository had been setup specifically for this project. This repository stored the codebase of the project.

## II. Background Research

### A. BGP

BGP was first defined in RFC 1105 in 1989 [18]. The most recent version is BGP-4, which is defined in RFC 4271 [19]. BGP-4 is an inter-AS routing protocol, allowing ASes to share network reachability information. This allows internet traffic to take the quickest path from source to destination. Each AS uses an Interior Gateway Protocol (IGP) to route packets within the AS, and an Exterior Gateway Protocol (EGP) to route packets to other ASes.

BGP uses four message types to communicate information. These are outlined below.

1) OPEN - An OPEN message is sent by sender and receiver once a TCP connection is established.
2) UPDATE - An UPDATE message advertises routes and/or withdraws routes.
3) KEEPALIVE - A KEEPALIVE message is used to keep a BGP connection alive, in the absence of UPDATE or NOTIFICATION messages.
4) NOTIFICATION - A NOTIFICATION message closes a TCP connection when an error is detected.

The UPDATE message provides the core functionality of BGP. Each UPDATE message contains "Path Attributes" that outline information about the route that is being advertised or withdrawn. These attributes are outlined below. Attributes marked as mandatory must be included in an UPDATE message.

- ORIGIN (mandatory) - Describes if the message originated internally (intra-AS), externally (inter-AS), or from an other mean.
- AS_PATH (mandatory) - Describes the ASes that the message has passed through.
- NEXT_HOP (mandatory) - Describes the IP address that should be used as the next hop for the IP address prefixes in the Network Layer Reachability Information field of the message.

- MULTI_EXIT_DISC - Used by an external router to determine the entry or exit point to/from a neighbour AS.
- LOCAL_PREF - This attribute is mandatory for intra-AS messages and states the preferred external BGP route.
- ATOMIC_AGGREGATE - Included when the message contains aggregated routes.
- AGGREGATOR - Describes the AS number and IP address of the router that performed route aggregation.

RFC 4271 declares that any BGP implementation must support RFC 2385 [20]. This document outlines an extension to the Transmission Control Protocol (TCP) for the purpose of providing enhanced security to BGP messages. The extension involves applying the MD5 algorithm to parts of the BGP messages and adding the hash to the message header. The receiver of the message would then calculate the hash of the received message and check if the hash matched the one in the header. This method prevents tampering of the message in transmission. However, malicious or accidental BGP anomalies still occur through owned or controlled ASes.

### *B. BGP Anomalies*

A BGP anomaly occurs when either an incorrect route is annouced and propogated, or there is a failure of one or more routers. As defined in [21], BGP anomalies can be placed into four categories. The first two are direct intended anomaly and direct unintended anomaly, and cover anomalies caused by human interaction with the BGP system of an AS. A direct intended anomaly is caused by BGP hijacking, such as when an attacker owned or controlled AS announces an IP address prefix they do not own for malicious purposes. A direct unintended anomaly is caused by a legitimate AS creating a misconfiguration in it's BGP system. The third and fourth anomaly types are indirect anomaly and link failure. An indirect anomaly occurs when malicious activity targeting internet components (e.g. web servers) affects BGP routing. A link failure anomaly occurs when a link between two paired ASes fails, preventing data transmission. This project focuses on the first two types of BGP anomalies, as these are the anomalies that occur most frequently, and cause the most damage.

## III. Related Work

### *A. ARTEMIS*

ARTEMIS [22] is a tool developed by Foundation for Research and Technology (FORTH) [23] and Center for Applied Internet Data Analysis (CAIDA) [24] to detect and mitigate BGP hijacking events in real-time. The tool has three core services it provides.

1) BGP update monitoring in real-time.
2) Detection of BGP prefix hijacking.
3) Mitigation of BGP prefix hijacking.

The advantages of ARTEMIS are it's near (within a few seconds) real-time anomaly detection, it's automatic mitigation, and it's ability to monitor certain prefixes. It's automatic mitigation calls a script to run when a user clicks the "Mitigate" button when viewing a BGP hijack event. It allows a user to only monitor specific IP address prefixes, such as those the user's organisation controls, to detect hijacks affecting their network. Additionally, ARTEMIS is open-source [25], which improves the security of the codebase, and free, so available to users belonging to organisations of all sizes.

The disadvantage of ARTEMIS is it's absence of machine learning. ARTEMIS uses a local configuration file, setup by the network operator, to detect hijacking. This file contains information on what BGP announcements for owned prefixes should look like, and if there are any discrepencies it will be detected. This method is limited to a narrow view of the worldwide BGP network, and relies on the operator to update the configuration file when any changes to prefixes are made. A machine learning method takes into account a wide view of the worldwide BGP network, and doesn't rely on an up-to-date configuration file to detect anomalies.

The differences between ARTEMIS and the solution being developed in this project are the solution's ability to detect anomalies in a network consisting of many ASes, and therefore it is able to detect anomalies that may affect the user's network ahead of time. This ability will allow a user to implement mitigation's for an anomaly that is affecting ASes, and is propagating to other ASes.

### *B. BGPalerter*

BGPalterer is a tool with a wide range of capabilities related to monitoring BGP activity. It is open-source [26] and monitors in real-time. It's full list of features can be found in the GitHub README [27]. Some of it's main features are as follows:

- Detect if prefixes are hijacked.
- Detect if an AS is announcing incorrect Route Origin Authorization (ROA) or Resource Public Key Infrastructure (RPKI) information.
- Detect if an AS is announcing a not announced before prefix (indicates direct intended/unintended anomaly).

The advantages of BGPalerter are it's varied range of capabilities and alerting system. The tool has twelve key features relating to BGP hijacking, ROA and RPKI. It's alerting system can notify the user via fourteen methods, such as file (logs), email, Microsoft Teams, and Slack. These capabilities allows a user the option to be notified anywhere for a range of unwanted changes to their BGP network.

BGP alerter suffers from the same disadvantage as ARTEMIS, where it's absence of machine learning means it cannot detect a large anomaly that would affect the users network ahead of time. Additionally, the solution being developed in this project specifically focuses on BGP anomalies, compared to BGPalerter's ROA and RPKI capabilities.

### C. BGPGuard

BGPGuard [28] is an BGP anomaly detection tool that can evaluate live and historical data. Data is sourced from RIPE and RouteViews and is evaluated either in real-time using pre-trained models, or evaluated for a historical period based on a start and end time. The tool can be used as a web application or terminal-based for Linux operating systems.

The advantages of this tool are it's real-time and offline classification, pre-trained machine learning models, and pre-processed anomalous data. It's offline classification is useful for evaluating historical BGP anomalies for use in future work. The tool contains processed data from five popular BGP anomaly incidents which saves the user time if they choose to investigate these commonly used anomalies.

The disadvantages of this program are it's lack of prefix-hijacking detection when running the tool in real-time, and it's lack of documentation. Prefix-hijacking detection is simple to monitor for and only requires a specified AS and list of IP prefixes, however, it is clear this is not an aim of the tool and alternative programs are available for this purpose. The tool has a lack of documentation and besides basic information on the tool and how to install it, there is no troubleshooting or user guide, which is important for a tool with many features and user configuration.

Outlined in table I is a comparison of the three discussed BGP anomaly detection programs and the project solution.

## IV. Design

### A. Possible Approaches

The first approach of this project was to utilize previous work done by [12] and re-design and re-implement the code-base to achieve the project goals and requirements. However, this approach was unsuccessful for multiple reasons. Firstly, there was little to no documentation for a significant proportion of the system, as well as parts of the code being commented out with no explanation as to why. Additionally, the system, when running, took more than fifteen minutes to process the data and produce a result, which is unsuitable for the project requirement of a system that can perform real-time anomaly detection. It was therefore decided to abandon this approach and use an alternative design.

Built network graph of ASes - Harder to graph, resource intensive.

Extract important features - Low resources needed, works well with ML.

### B. System Architecture

The design of the project solution can be broken down into a range of components as follows:

- BGP data (external).
- BGP data processing.
- Feature extraction.

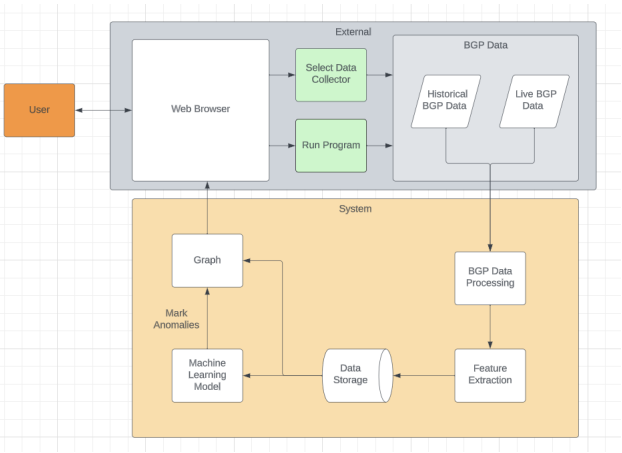


Fig. 1: Design architecture.

- Machine learning.
- Data storage.
- Graphing.
- Web browser (external).

The diagram in Fig. 1 shows a flow diagram detailing each component and how it interacts with other components.

*1) BGP Data:* The system requires the ability to access and collect historical and live BGP data from an external system. This data will be used by the BGP data processing component to parse the collected data into a format that can then be used to extract important features from the data. This component will be impacted by user interaction with the system via their web browser. The user will be able run the program by interaction with the website.

This component is required by the system as BGP data is needed to achieve the goals of the project. There are no alternative components that could replace this component.

*2) BGP Data Processing:* BGP data will need to be processed to ensure it is compatible with the feature extraction component. This component will involve taking raw BGP data as input from the BGP data component and producing an output of data that can then be input into the feature extraction component. It is a requirement that no important data is lost when processing to ensure the data gathered from the BGP data component and the data used in the system is consistent, otherwise, false positive or false negative anomalies may be more likely to occur.

*3) Feature Extraction:* This component will extract features from the processed BGP data for storage and ultimately use in the machine learning and graphing components. Feature selection is an important part of creating a machine learning model, and the following ten features have been selected for use in the system. All features will be stored as an integer.

- Number of ASes observed.
- Number of prefixes announced.
- Number of prefixes withdrawn.

| Capability | ARTEMIS | BGPalerter | BGPGuard | Project Solution |
|---|---|---|---|---|
| Prefix Hijacking | Yes | Yes | No | No |
| Time-Series Graphing | No | No | No | Yes |
| ROA Detection | No | Yes | No | No |
| RPKI Detection | No | Yes | No | No |
| Machine Learning Anomaly Detection | No | No | Yes | Yes |
| Data Selection | No | No | Yes | Yes |

TABLE I: Comparison of related work and project solution.

- Number of updates.
- Number of announcements.
- Number of withdrawals.
- Number of withdrawals after announcement.
- Number of updates with an origin of IGP.
- Number of updates with an origin of EGP.
- Number of updates with an origin of Unknown.

These features have been chosen due to their effectiveness as indicators of an anomaly in related work [29]–[31]. Once extracted, the features will be stored in the data storage component. During the feature extraction process, the collection of features will be grouped in one minute intervals based on the timestamp of the BGP message packet. One minute intervals have been chosen because of their prevalence in related work [32] and because they are fitting for a time-series graph as used by the graphing component.

*4) Data Storage:* Once the features have been extracted, they must be stored for use by the machine learning and graph components of the system. The data received from the feature extraction component will be collections of features grouped by one minute intervals. This data will then be stored and passed to the machine learning and graph components.

*5) Machine Learning:* A machine learning model will be used in the system to detect anomalies. One consideration of the model it must be able to detect anomalies from BGP data sourced from different collectors. Therefore, a concern with using a supervised model arises. Different collectors have different data flows, for example, spikes in BGP activity at certain times each day, which means training a model on one collector would likely be unreliable at detecting anomalies when testing another collector's live data. Therefore, it was decided to use an unsupervised machine learning model, which additionally removes the process of having to label data for training and training the model itself.

The machine learning component will take BGP data as input from the data storage component, and will mark anomalies in the data. It will then communicate with the graph component the anomalies so they can be marked on the graph.

*6) Graphing:* To visualise the extracted features and any anomalies that have been detected, a graphing component will be used. This component will take input from the data storage component to display extracted features in a time-series graph with an x-axis of one minute intervals and a y-axis of the total count of the features. Each feature will have it's own line in the graph. The machine learning component will communicate which intervals, if any, are anomalous. The graph will be provided to the user's web browser.

*7) Web Browser:* The web browser component is responsible for displaying the graph to the user, and allowing them to start the program, and select which collector to use.

### C. Sustainability

*1) Environmental:* The design of the solution is intended to be lightweight and can be run on any computer system with reasonable performance. This results in the energy usage of the system being significantly less than a system that needs to be run on an individual server or multiple servers in a data centre. Additionally, the design does not require large historical datasets to train the machine learning model, where training a machine learning model with large datasets is resource intensive and can use a significant amount of energy.

Furthermore, as the design of the solution functions in real-time, it can be used to detect, and then mitigate or resolve, BGP anomalies quickly which would otherwise result in an increase in the energy consumption of the internet infrastructure due to incorrectly routed internet traffic.

*2) Social:* BGP anomalies can have a social impact due to the inability to communicate with affected destinations (i.e. websites) during an anomaly. By detecting anomalies in real time, they can be resolved faster, therefore reducing the downtime of affected websites. Possibly the most notorious BGP anomaly incident with a social impact was in 2008, when Pakistan Telecom, at the bequest of the government, redirected requests to youtube.com to one of their IP addresses and accidentally published the route worldwide [33]. This resulted in YouTube being inaccessible worldwide for around two hours. An incident like this explains the need for anomalies to be detected as quickly as possible.

*3) Economic:* Many financial and economic institutions worldwide rely on the internet to function. It is therefore critical that internet infrastructure be resilient to failure. However, when the infrastructure does fail, it is important it can be fixed as soon as possible. By detecting BGP anomalies in real-time, the monetary impact of anomalies, such as a disruption to banking transactions, can be limited.

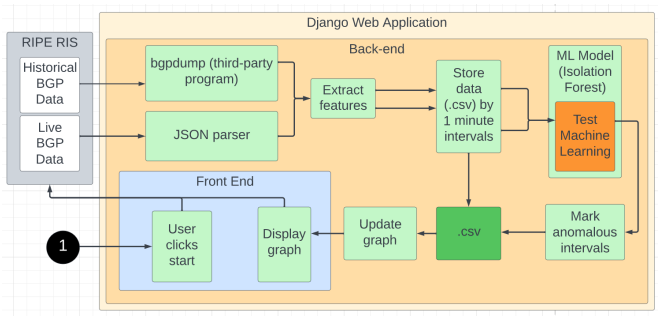


Fig. 2: Implementation system flow

*4) Technical:* Technologies used in the project solution are able to support the functional and maintenance requirements of the project for at least two years. These technologies include Python, Django, bgpdump and scikit-learn which are regularly maintained by their respective professional groups. It is almost certain the groups will continue maintaining their technologies for the next two years.

## V. Implementation

### A. System Flow

The system is run as a Django web application on 127.0.0.1:8000. When the user accesses the web-page, they are shown one input box, two buttons, and a graph. The input box allows the user to select which RIS RIPE collector to use, with validation to check the input is between 0 and 26, and a default value of 0 if the user runs the program without having input a number. There is a button to begin the program, and a button to terminate the program. Once started, the program will fetch the previous 30 minutes of BGP update data from the chosen collector (e.g. collector 0 [34]), process it using bgpdump, and extract the features in one minute intervals. The data is then stored in a CSV file, with each row representing one minute. Next, the data is used by the scikit-learn isolation forest method to determine if there are any anomalous intervals, and used by matlibplot to plot the features in a time-series graph. Once this initial stage is complete, the processing of live data begins, where it is processed from RIS Live using a WebSocket. The live data follows the same process as the historical data, and every minute, the features extracted from the live data are saved, and the graph and isolation forest model are updated, before repeating the process again.

### B. BGP Data

The project design required access to historical and real-time BGP data. This was implemented by using the RIPE RIS route collectors [3] for historical data, and the RIPE RIS Live service [4] for real-time data.

Historical data is sourced from 23 active route collectors and 3 inactive collectors, with each collector containing years of historical data in MRT format. Two types of data are saved, updates and dumps, with updates being saved every 5 minutes, and dumps being saved every 8 hours. Updates contain all changes to the routing system for the 5 minute period, and dumps contain the state of the routing system at the captured time. Data can be downloaded in the gzip format, where it must be unzipped into the intended MRT format. In the system, these gzip files are downloaded and stored before being processed by bgpdump.

Live data is sourced using the RIS Live WebSocket interface. In the system, when the user runs the program, a WebSocket is created with parameters that specify the route collector to use, and that raw (all) data should be sent. The WebSocket is then connected to the RIS Live URL and a message is sent subscribing to the service, telling the service to begin sending data. Data is sent by the service in JSON format and is continuously processed by the system.

RIPE RIS was chosen because it is free, reliable, and provides access to both historical and real-time BGP data. As historical and real-time data is sourced from the same organisation, it is consistent, this is because the real-time data from RIS Live ends up as part of a historical data dump. RouteViews [35], An alternative source of historical and real-time BGP data was considered for use in the project. The differences between RIPE RIS and RouteViews historical data services are their collectors, the data format (RouteViews uses FRR and Cisco formats), and the periods of saved data (2 hours for RIPE RIS equivalent of dumps, and 15 minutes for RIPE RIS equivalent of updates). RIPE RIS was chosen over RouteViews because RouteViews real-time data stream is much more complex to setup compared to RIS Live.

### C. Data Processing

To process the historical data, stored in a gzip file in MRT format, bgpdump is used. bgpdump is a tool developed by RIPE NCC that is written in C and can be used to parse BGP dumps in MRT format. In the system, the tool is used by executing the command ”bgpdump -O [output_file_name].txt [input_file_name].gz” through an ”os.system” Python method call, where the input file name is the gzip file to be parsed. In the standard flow of the program, once the 30 minutes of historical data (six 5 minute update gzip files) are downloaded, they are then parsed one at a time by bgpdump. The output of this process is six text files containing the raw human-readable BGP message data from a five minute period. An example of a BGP message from one of these text files is shown in Fig. 3.

To process the real-time data, it is parsed from JSON format into a custom update object class, which stores the information contained in the BGP update message that is used in feature extraction. In the feature extraction process, only BGP update messages are relevant, therefore, all other BGP message types (Keepalive, Open, Notification) are all skipped. Shown in Fig. 4 is the update class and it’s fields that are relevant for feature extraction.

```
TIME: 10/16/23 01:35:10
TYPE: BGP4MP/MESSAGE/Update
FROM: 198.32.160.39 AS9304
TO: 198.32.160.99 AS12654
ORIGIN: IGP
ASPATH: 9304 1239 6762 16735 25933 28169 53066
NEXT_HOP: 198.32.160.39
ANNOUNCE
  187.86.154.0/23
  179.189.156.0/23
  179.189.158.0/23
```

Fig. 3: bgpdump output, example of an update message.

```python
class UPDATE:
    Daniel Robertson
    def __init__(self, timestamp, peer, peer_asn, message_id, path, community, origin, announcements, withdrawals):
        self.timestamp = timestamp
        self.peer = peer
        self.peer_asn = peer_asn
        self.message_id = message_id
        self.path = path
        self.community = community
        self.origin = origin
        self.announcements = announcements
        self.withdrawals = withdrawals
```

Fig. 4: Update class

### D. Feature Extraction

Once historical BGP data has been processed it is stored in a text file, with each message containing information on multiple lines, as shown in Fig. 3, and a line break in-between each message. The features are extracted by, for each message, iterating down the lines and incrementing a feature's count when certain information is found. If the message type is not an update message, it is skipped, as only update messages are relevant to the features.

Once live BGP data has been processed it is stored in an update message object, as shown in Fig. 4. Feature values are incremented based on the information stored in each field of an update message object, for example, if the origin field of an object is "EGP" the "Number of updates with an origin of EGP" will be incremented by one.

When extracting feature values from historical and live BGP data, the feature values are stored in a TimePeriod class object (shown in Fig. 5. The non-integer fields are used to store information that is used to calculate integer feature values. For both historical and real-time data, a TimePeriod object stores a one minute period of data, which is then used by the data storage component to store the data.

### E. Data Storage

The data that requires storage is the collection of feature values in one minute periods. This information is stored in a CSV file, with each row representing a one minute period, and headers specifying the information in a column, such

```python
class TimePeriod:
    Daniel Robertson +1
    def __init__(self):
        self.ASes_observed = set()
        self.number_of_ASes = 0
        self.prefixes_observed = set()
        self.number_of_prefixes_announced = 0
        self.number_of_prefixes_withdrawn = 0

        self.number_of_updates = 0
        self.number_of_announcements = 0
        self.number_of_withdrawals = 0

        self.number_of_updates_AS = dict()
        self.number_of_announcements_AS = dict()
        self.number_of_withdrawals_AS = dict()

        self.number_of_updates_prefix = dict()
        self.number_of_announcements_prefix = dict()
        self.number_of_withdrawals_prefix = dict()

        self.number_of_withdrawals_after_announce = 0
        self.number_of_IGP = 0
        self.number_of_EGP = 0
        self.number_of_unknown = 0
```

Fig. 5: TimePeriod class

| Second | Scores | Anomaly | Number of ASes Observed | Number of Prefixes Announced |
|---|---|---|---|---|
| 0 | 0.171720488752911 | 1 | 544 | 934 |
| 60 | 0.163197248369767 | 1 | 581 | 1227 |
| 120 | 0.184105309060684 | 1 | 399 | 642 |
| 180 | 0.179103524965346 | 1 | 488 | 665 |
| 240 | 0.170714758608022 | 1 | 449 | 615 |
| 300 | 0.139443233892904 | 1 | 912 | 2731 |
| 360 | 0.150879372971043 | 1 | 726 | 2006 |
| 420 | 0.104190007697531 | 1 | 887 | 3082 |
| 480 | 0.0578742000461185 | 1 | 888 | 6429 |
| 540 | 0.171019694565877 | 1 | 398 | 464 |
| 600 | 0.129242081352059 | 1 | 510 | 1131 |
| 660 | 0.147089407284482 | 1 | 436 | 1093 |
| 720 | 0.0722550097854492 | 1 | 1116 | 5994 |
| 780 | 0.160253410480174 | 1 | 428 | 796 |
| 840 | 0.171820181303161 | 1 | 427 | 829 |
| 900 | 0.152507275176256 | 1 | 393 | 751 |
| 960 | -0.0262998503645197 | -1 | 723 | 1781 |
| 1020 | 0.156781526291804 | 1 | 385 | 829 |
| 1080 | 0.159735572043632 | 1 | 369 | 530 |
| 1140 | -0.0722972432721561 | -1 | 1026 | 4168 |
| 1200 | 0.182829500591928 | 1 | 404 | 472 |
| 1260 | 0.15600908358519 | 1 | 382 | 592 |
| 1320 | 0.127617457192371 | 1 | 417 | 725 |
| 1380 | 0.0783203184411065 | 1 | 411 | 958 |
| 1440 | 0.164592823288483 | 1 | 388 | 739 |
| 1500 | 0.165279003033495 | 1 | 474 | 856 |
| 1560 | 0.0321442615566352 | 1 | 736 | 2677 |
| 1620 | 0.13650188088654 | 1 | 473 | 2084 |
| 1680 | 0.178388640358594 | 1 | 411 | 531 |
| 1740 | 0.17453325082066 | 1 | 380 | 583 |
| 1800 | 0 | 0 | 589 | 1565 |

Fig. 6: Example of 30 minutes (1800 seconds) of stored data.

as the feature name. The columns can be split into three categories: time, machine learning, and features. Time is a single column containing the second of an interval, which acts as a label, and the x-axis in the graph. The machine learning contains two columns, one for the anomaly score, and one for the anomaly label, where 1 is normal and -1 is anomalous. The remaining columns are features.

To limit the data storage, the maximum number of rows (i.e. minutes) that can be stored in the CSV file is 1440, equating to one day of data. If this limit is reached, new data that is added will overwrite old data, giving a "sliding

window" of 24 hours for use by the machine learning and graph components.

### F. Machine Learning

The machine learning method implemented in the program using scikit-learn is the isolation forest algorithm. This algorithm is an unsupervised anomaly detection algorithm, and takes the feature values stored in the data storage CSV file as input. The model uses 100 estimators, and a contamination value of 0.03. The model calculates the anomaly score and prediction for each row, which is then output to the CSV as the "Scores" and "Anomaly" columns, where an "Anomaly" value of -1 is anomalous.

Isolation forest was chosen as the machine learning algorithm for this project based on the project goals and requirements. A unsupervised method was needed as the model must be able to be applied to one of 23 different route collectors, using a supervised method would be infeasible as the amount of data that would need to be labelled to train the model is too large for the project scope. The algorithm is also quicker to fit, and less sensitive to the scale of variables than other unsupervised methods such as local outlier factor and support vector machines [36].

### G. Graphing

To visualise the extracted feature values and anomalies, a time-series graph is created using matlibplot. The graph uses the CSV data storage file, which contains the feature values and anomaly label for each interval. This data is plotted on the graph with the x-axis being the "Second" value, and the y-axis being the integer count of the feature values. Each of the ten features are plotted on separate lines, resulting in ten lines being displayed on the graph. If an interval is labelled as anomalous, a vertical red line is plotted on the graph to mark this. allowing to user to see an anomaly.

### H. Web Application

To visualise the program output and allow user interaction, a Django web application is used. The web application has three functions: allow the user to start and stop the program, select the route collector, and display the graph.

## VI. Evaluation

To evaluate the solution, two of the system's qualities were tested, accuracy and performance. To test accuracy, the project's solution was evaluated against two real-world BGP anomalies to determine if it were correctly able to identify the anomalies. These anomalies were the 2003 Slammer Worm and 2008 Mediterranean undersea cable cut incidents. To test performance, the system was assessed to check it could run indefinitely using any route collector.

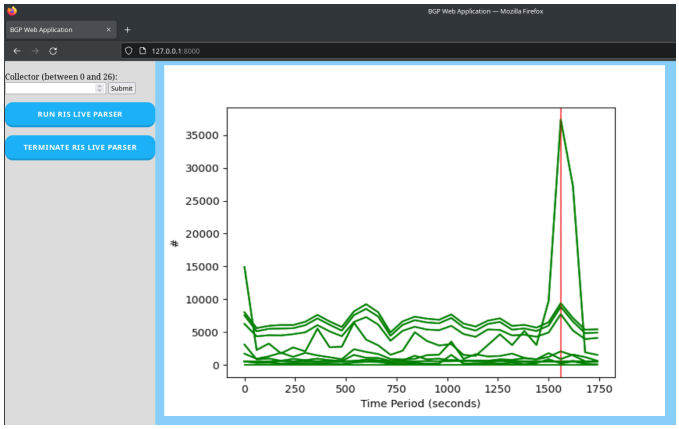


Fig. 7: Web application.

The solution was not evaluated for it's accuracy in detecting an anomaly in real-time due to the significant challenge in predicting or waiting for an anomaly to occur. However, the way the solution was evaluated, using historical anomalies, is of negligible difference to how the system runs in real-time.

### A. 2003 Slammer Worm

On January 25th 2003 just before 05:30UTC Slammer worm, the fastest spreading computer worm to this day, began infecting machines [5]. The total number of machines infected doubled every 8.5 seconds from its release and it quickly caused networks to slowdown or shutdown entirely. As a result of this, BGP traffic during the incident's period was affected, and many more BGP messages were sent and received.

To evaluate the project solution using this anomaly, historical data from two RIPE RIS route collectors was used. These collectors were RRC00, based in Amsterdam, Netherlands, and RRC01, based in London, United Kingdom. These collectors were chosen due to their data availability, as some other collectors did not have historical data going back to 2003. The time period chosen for the evaluation was between 04:30UTC and 07:00UTC, beginning roughly one hour before the worm began spreading, and ending roughly one and a half hours after.

Shown in Fig. 8 & 9 are the graphs from both collectors in the time period. As can be seen, the solution has detected the anomaly (marked by vertical red lines) as it begins, just before 4000 on the x-axis, which is around 05:30UTC. Additionally marked on the Amsterdam graph are two major, and one minor, fluctuations during the anomaly period.

### B. 2008 Mediterranean Undersea Cable Cut

On December 19th 2008, three undersea fibre optic cables were cut at 07:28UTC, 07:33UTC, and 08:06UTC in the Mediterranean sea [6]. The cables run between Italy and

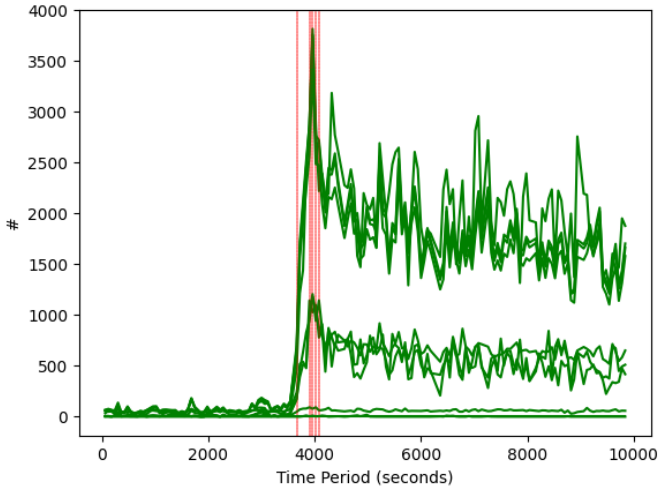


Fig. 8: RRC00, London collector.

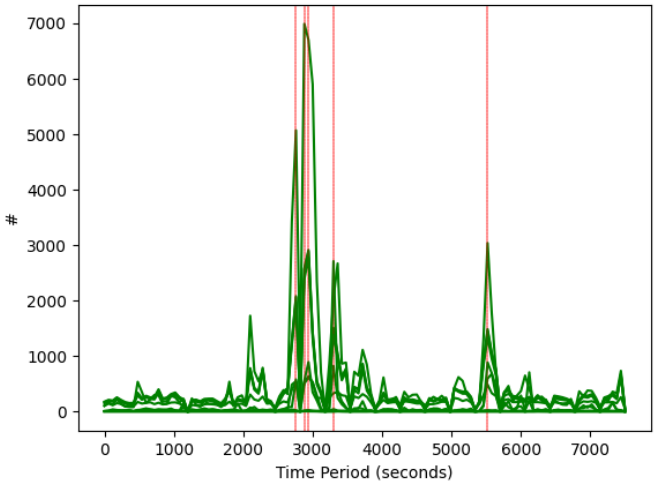


Fig. 10: RRC10, Milan collector.

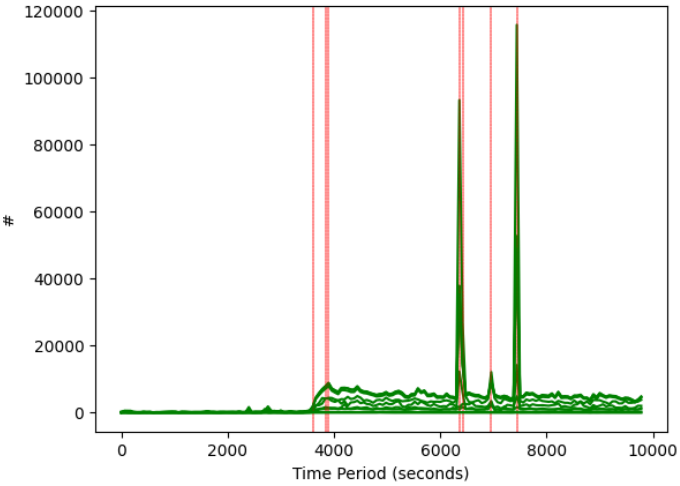


Fig. 9: RRC01, Amsterdam collector.

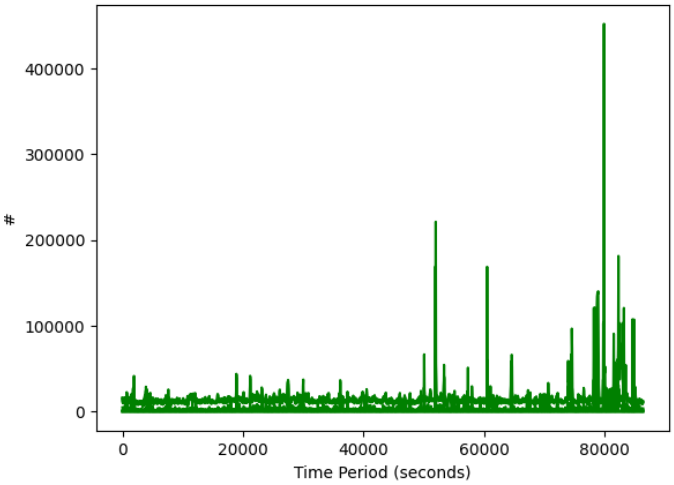


Fig. 11: 24 hours worth of continuous live data from RRC11

Egypt and allow for the transfer of around 90% of data between the Middle East and Europe. As the cables were rendered inoperable during this incident, traffic was re-routed, causing disruptions, and a noticeable increase in BGP traffic for a period.

To evaluate the project solution using this anomaly, historical data from one RIPE RIS route collector was used. This was the collector RRC10, based in Milan, Italy. It was chosen due to it's proximity to the incident. The time period chosen for the evaluation was between 06:45UTC and 08:30UTC, beginning roughly 45 minutes before the first cable was cut, and ending roughly one hour after.

As observed in the graph shown in Fig. 10, there are spikes, and marked anomalies at around 3000 seconds and 5500 seconds, which are at around 07:35UTC and 08:15UTC respectively, and just after the cables were cut.

### C. Live Performance

As one of the requirements of the system is that in runs in real-time, it must be validated that the system can handle the BGP data flow from any of the 23 active RIPE RIS collectors. To evaluate this, the system was run using three high-traffic collectors, RRC00, RRC11, and RRC24 for a period of one day each. All three trials were able to continuously process data for the period. Shown in Fig. 11 is the feature data from RRC11 when graphed.

## VII. Future Work

To extend and improve on the functionality of the project solution, the following ideas are proposed as future work.

- Implement more features - Currently the program extracts ten features.
- Implement alternative machine learning algorithm(s).
- Redesign the user interface for better user experience.

## VIII. Conclusion

This paper has outlined the problem of detecting BGP anomalies in real-time. Detecting anomalies in real-time can reduce the time taken to mitigate or fix the cause, and can limit the damages caused, such as service slowdown or outage, or loss of revenue. The project goals and performance measurements were defined, and background research on BGP was explained. Related work, including the ARTEMIS, BGPalerter, and BGPGuard tools were analysed and compared to the project solution. The project solution design was outlined in depth, followed by detail on how the design was implemented to achieve the project's goals. To evaluate the solution, accuracy and performance were examined, and the solution was determined to be able to function in real-time, and accurately detect real-world BGP anomaly events.

## References


[1] M. Prince, "August 30th 2020: Analysis of centurylink/level(3) outage." https://blog.cloudflare.com/analysis-of-todays-centurylink-level-3-outage/, Aug 2020.

[2] A. Kesavan and A. Medina, "Centurylink / level 3 outage analysis — thousandeyes." https://www.thousandeyes.com/blog/centurylink-level-3-outage-analysis, 2020.

[3] RIPE Network Coordination Centre, "Route collectors." https://ris.ripe.net/docs/10_routecollectors.html, 2023.

[4] RIPE Network Coordination Centre, "Ris live." https://ris-live.ripe.net/, 2023.

[5] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "The spread of the sapphire/slammer worm." https://www.caida.org/catalog/papers/2003_sapphire/, Jan 2003.

[6] "Undersea cable cuts in the mediterranean affected 14 countries - submarine networks." https://www.submarinenetworks.com/news/cable-cuts-affected-14-countries, Mar 2011.

[7] U. Nations, "Goal 9 — department of economic and social affairs." https://sdgs.un.org/goals/goal9, 2022.

[8] U. Nations, "Goal 12 — ensure sustainable consumption and production patterns." https://sdgs.un.org/goals/goal12, 2022.

[9] U. Nations, "Goal 13 — take urgent action to combat climate change and its impacts." https://sdgs.un.org/goals/goal13, 2022.

[10] U. Nations, "The 17 sustainable development goals." https://sdgs.un.org/goals, 2023.

[11] Python Software Foundation, "Python." https://www.python.org/, 2023.

[12] J. Y. L. Huang, "Modelling bgp updates for anomaly detection using machine learning," 2021.

[13] Django Software Foundation, "Django." https://www.djangoproject.com/, 2023.

[14] RIPE NCC, "bgpdump." https://github.com/RIPE-NCC/bgpdump.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[16] JetBrains, "Pycharm." https://www.jetbrains.com/pycharm/.

[17] GitLab Inc, "Gitlab." https://about.gitlab.com/.

[18] Y. R. K. Lougheed, "A border gateway protocol (bgp)." https://www.rfc-editor.org/rfc/rfc1105, 1989.

[19] S. H. Y. Rekhter, T. Li, "A border gateway protocol 4 (bgp-4)." https://www.rfc-editor.org/rfc/rfc4271, 2006.

[20] A. Heffernan, "Protection of bgp sessions via the tcp md5 signature option." https://www.rfc-editor.org/rfc/rfc2385, 1998.

[21] G. A. B. Al-Musawi, P. Branch, "Bgp anomaly detection techniques: A survey." https://ieeexplore.ieee.org/document/7723902, 2017.

[22] FORTH, CAIDA, Code BGP, "Artemis." https://bgpartemis.org/, 2023.

[23] FORTH Institute of Computer Science, "Forth institute of computer science." https://www.ics.forth.gr/, 2023.

[24] Center for Applied Internet Data Analysis, "Caida." https://www.caida.org/, 2023.

[25] INSPIRE Group FORTH-ICS, "Artemis." https://github.com/FORTH-ICS-INSPIRE/artemis, 2022.

[26] NTT Global IP Network, "Bgpalerter." https://github.com/nttgin/BGPalerter, 2023.

[27] NTT Global IP Network, "Bgpalerter readme.md." https://github.com/nttgin/BGPalerter#readme, 2023.

[28] Z. Li, "Bgpguard." https://bgpguard.com/, 2023.

[29] J. Li, D. Dou, Z. Wu, S. Kim, and V. Agarwal, "An internet routing forensics framework for discovering rules of abnormal bgp events," *ACM SIGCOMM Computer Communication Review*, vol. 35, p. 55, Oct 2005.

[30] N. M. Al-Rousan and L. Trajković, "Machine learning models for classification of bgp anomalies," in *2012 IEEE 13th International Conference on High Performance Switching and Routing*, pp. 103–108, 2012.

[31] I. O. de Urbina Cazenave, E. Köşlük, and M. C. Ganiz, "An anomaly detection framework for bgp," in *2011 International Symposium on Innovations in Intelligent Systems and Applications*, pp. 107–111, 2011.

[32] B. Al-Musawi, P. Branch, and G. Armitage, "Bgp anomaly detection techniques: A survey," *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, p. 389, 2017.

[33] D. McCullagh, "How pakistan knocked youtube offline (and how to make sure it never happens again)." https://www.cnet.com/culture/how-pakistan-knocked-youtube-offline-and-how-to-make-sure-it-never-happens-again/, Feb 2008.

[34] RIPE NCC, "Rrc00." https://data.ris.ripe.net/rrc00/.

[35] University of Oregon, "University of oregon route views project." https://www.routeviews.org/routeviews/, 2023.

[36] D. Cortes, "An introduction to isolation forests." https://cran.r-project.org/web/packages/isotree/vignettes/An_Introduction_to_Isolation_Forests.html.