8-2017

InSight2: An Interactive Web Based Platform for Modeling and Analysis of Large Scale Argus Network Flow Data

Hansaka Angel Dias Edirisinghe Kodituwakku
University of Tennessee, Knoxville, hkoditu1@vols.utk.edu

To the Graduate Council:

I am submitting herewith a thesis written by Hansaka Angel Dias Edirisinghe Kodituwakku entitled "InSight2: An Interactive Web Based Platform for Modeling and Analysis of Large Scale Argus Network Flow Data." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Engineering.

Jens Gregor, Major Professor

We have read this thesis and recommend its acceptance:

Mark E. Dean, Audris Mockus

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

InSight2: An Interactive Web Based Platform for Modeling and Analysis of Large Scale Argus Network Flow Data

A Thesis Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Hansaka Angel Dias Edirisinghe Kodituwakku
August 2017

In loving memory of Jovi

ACKNOWLEDGEMENTS

I would like to thank my advisor and mentor, Dr. Jens Gregor for his excellent guidance, infallible support and prospective wisdom throughout the course of my Master's program, without which I could not have completed this immense task within this time frame. I learned a great deal from his effectiveness in planning and optimizing. His timely decisions saved me a lot of time and enabled me to streamline my workflow to do more in less time. It was an honor and a joy to work with him and I am looking forward to continue working with him in the coming years.

I am thankful for Dr. Mark E. Dean and Dr. Audris Mockus for their time in serving in my thesis committee and their guidance during class work.

I would like to extend my sincere gratitude to Mr. Greg S. Cole for his unfailing faith in me and giving me the opportunity to be a part of GLORIAD family. He saw my potential more than me and encouraged me to achieve it even though at the time it seemed impossible to me. His visionary ideas inspired me to achieve more and made me realize that there is no limit for one's potential.

I would like to thank my parents for their incredible love, sacrifice and support they gave for me throughout my life. I would like to thank my mom who stood up for me when the tides were rough. I learned to always do the right thing and that hard work always pays in the end, from her. All my virtues and good characteristics are inherited from her outstanding parenting and shrewdness. I would like to thank my dad who worked hard to support our family and being my hero. His fearless behavior observed over the years primed my ability to face any circumstance head-on. I would like to thank my sister for her never-ending love and being there all the time for me. She is the best sibling anyone would hope for.

I would like to thank Fan Zhang for her support and never giving up on me. Her encouragement and appreciation enabled me to work harder and discover new potentials I never knew I had. I would like to thank Michele Norris for treating me as her own family and being there for me all the time. I would like to thank Dana Bryson for helping me out countless times with paperwork amidst her busy work schedule. I would like to thank Saeed, Seena and Mustafa for being my brothers which I never had in my childhood.

I would like to thank Jovi and Sasha for giving me their unconditional love. Jovi, you left your paw-prints of joy on our hearts and we will never forget you. You are a warrior and you fought through battles life unfairly bestowed upon you. Sasha you are the sweetest and one of a kind. No wonder why you steal everyone's hearts!

I would also like to thank The University of Tennessee for letting me be a part of the Big Orange to study the cutting edge technology from the best professors in the world via its Masters of Science program. I would like to thank Laurel Residence for being my home away from home half-way across the globe, for all the great memories and for all the amazing friends I made during my stay.

The work presented in this thesis was supported by the National Science Foundation under Grant No. IRNC-1450959. Any opinions, findings, and conclusions or recommendations expressed are those of the author and do not necessarily reflect the views of the National Science Foundation.

ABSTRACT

Monitoring systems are paramount to the proactive detection and mitigation of problems in computer networks related to performance and security. Degraded performance and compromised end-nodes can cost computer networks downtime, data loss and reputation. InSight2 is a platform that models, analyzes and visualizes large scale Argus network flow data using up-to-date geographical data, organizational information, and emerging threats. It is engineered to meet the needs of network administrators with flexibility and modularity in mind. Scalability is ensured by devising multi-core processing by implementing robust software architecture. Extendibility is achieved by enabling the end user to enrich flow records using additional user provided databases. Deployment is streamlined by providing an automated installation script. State-of-the-art visualizations are devised and presented in a secure, user friendly web interface giving greater insight about the network to the end user.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 Motivation: Need for Network Monitoring and Analytics

Computer networks need to be constantly monitored to ensure optimal service. As they grow to include multiple sub networks and end nodes with different bandwidths, managing them increases in complexity. Modern networks are also constantly being targeted by malware and malicious users for malignant intentions. One of the key challenges in managing large scale networks spanning across the globe is being able to proactively detect systemic problems related to performance and security in near real-time. Many solutions are tailor-made implementations; proprietary solutions aimed towards particular needs of a specific organization. There is no single platform that can apply to a wide variety of applications ranging from performance to security which also gives the end user the complete control over the visualizations, dashboard layout and the custom data to be enriched.

1.1.1 GLORIAD: A Research and Education Network

The Global Ring Network for Advanced Applications Development (GLORIAD) was an advanced internet network for science applications in research and education across the globe initiated by Greg Cole [1]. Figure 1.1 shows the GLORIAD logo.

Greg Cole is a graduate of The University Tennessee (UT) Knoxville. He received a Master's degree in Computer Science in 1987. He first started working in the UT Knoxville Office of Research Administration and became the Director of the Office of Research Services at The University of Tennessee. Then he became Director of the Center for International Networking Initiatives at the University of Tennessee in 1995.

GLORIAD
Connecting the World of Science and Education (www.gloriad.org)

Figure 1.1 GLORIAD Logo [2]

Greg Cole started a project called MIRNET in 1997 which connected scientific communities in the U.S. and Russia [3]. Over time, MIRNET evolved into GLORIAD starting with the single ring covering the northern hemisphere shown in Figure 1.2 and ending with the global network shown in Figure 1.3. GLORIAD connected over 15 million end-point addresses across the Research and Education internetwork during its active service from 1999 – April, 2016. It was supported by the National Science Foundation (NSF) as well as various institutions and organizations in the many member countries including national science ministries, national research and education networks, science institutions and universities. The network provided an advanced infrastructure supporting a broad range of joint science and education projects and included partners in Russia, China, Korea, Egypt, India, Singapore, Canada, Netherlands and the five Nordic countries. [4]

GLORIAD had a significant academic presence. The program not only actively worked on initiation, active development and improvement of computer networks across the globe but also contributed to the community through activities related to global networking and community building during its 16 years of existence through the "Net Challenges" series of initiatives.

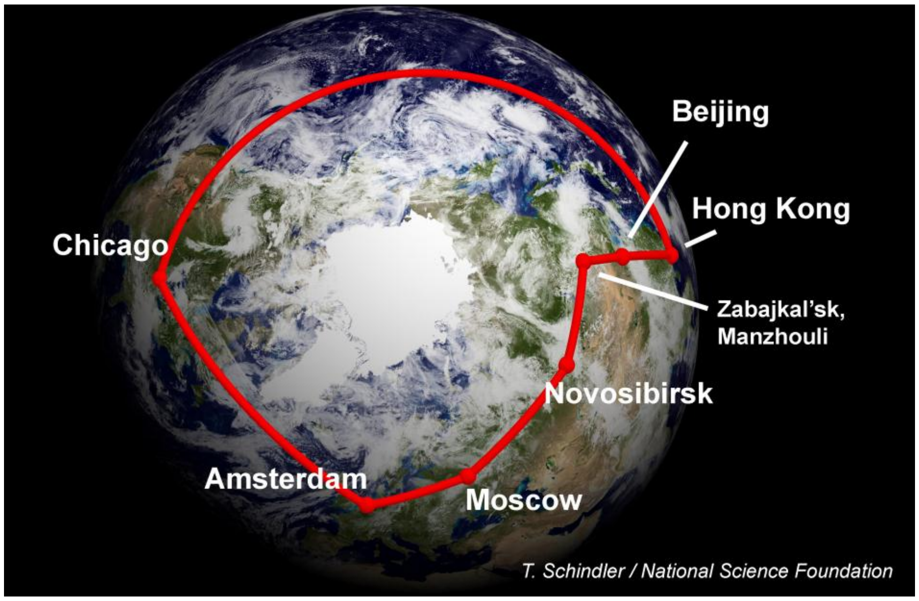


Figure 1.2 Little GLORIAD [5]

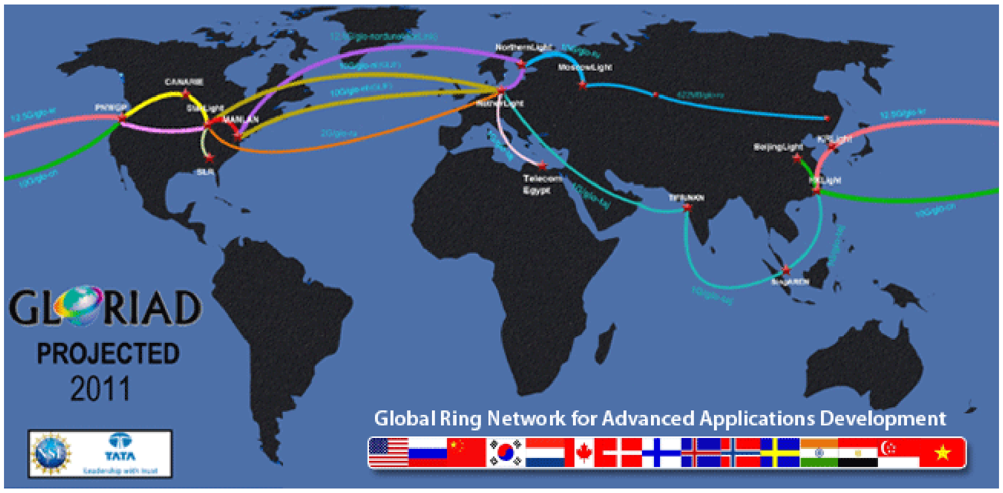


Figure 1.3 GLORIAD Coverage 2011 [6]

1.1.2 Argus Network Flow Data

A network flow is a record of metadata about one or more data transactions that share a predetermined set of parameters such as IP address, port and protocol. In other words, it summarizes a series of packets associated with a transaction. Network flow information of one type or another is routinely collected across large-scale networks to monitor their performance since compared to the packet size itself, the size of the flow data is much smaller and yet provides substantial insight into all the network traffic transactions allowing them to be archived for analysis.

Argus is open-source software developed by Carter Bullard and his company QoSient. Argus is a bidirectional network flow generator and aggregator [7]. The software consists of a server and a suite of clients. The server generates Argus records which can be read by clients. There are 25 Argus clients that specialize in different functions. They can be used in a pipeline to perform processing on flow records. It is a rich network flow data generation platform that is widely used by many universities, corporations and government entities including GLORIAD. Chapter 2 provides more information about Argus.

GLORIAD collected Argus flow data from its network from 2012 to 2015 amounting to 14.6TB of historical data [8]. Argus is capable of generating 127 features per network flow record. Appendix A gives a description of these features. For the work presented here, 37 Argus features were carefully selected to be extracted from these flow records; 27 of the features are common to both the source and the destination, while 10 of the features are unique to either the source or the destination.

We enrich the Argus flow data by adding 15 fields that convey information from the Global Science Registry (GSR), which is a database that relates IP addresses with research organizations and institutes. We add an additional 6 fields using information from the MaxMind GeoIP database which maps each individual IP address to a geographical location using latitude and longitude coordinates and assigns it a country, city, province, and zip code. Finally, we generate 11 dynamic features on-the-fly using ‘Scripted Fields’ within an Elasticsearch database. Elasticsearch is a noSQL database that has an integrated full-text search engine and is part of the Elastic Suite [9]. It is capable of storing data in arbitrary formats and provides an Application Programming Interface (API) to search for records. Appendix A gives a full list of these additional features. Geographical coordinates are used to show each flow record in a global map in the user interface of InSight2 which is described in-depth in Chapter 5.

1.1.3 GLORIAD InSight (2013-2014)

GLORIAD InSight is a network monitoring platform developed in 2013 in order to monitor the performance and security aspects of the GLORIAD network [10]. The main focus of it was to display network measurements such as packet loss, jitter, and load in order to detect systemic problems related to performance, routing and security that can contribute to reliability issues, and security vulnerabilities.

GLORIAD InSight consists of a ‘Farm of Animals’ where farm is a collective reference to a suite of Perl 5 scripts that have been given animal names such as ‘Elephant’, ‘Mouse’, ‘Rabbit’, and ‘Sheepdog’. Each animal carries out a specific task. Animal welfare is taken care by a script called the ‘vet’. Animals communicate with each other through a special message passing bus called ZeroMQ [11]. All components of the farm are maintained by Monit software [12] which monitors process numbers and ensures that all scripts are running. If a script terminates prematurely, be that an animal or even the vet, Monit re-invokes it. This was done to ensure continuous operation of the farm despite possible programmer or system errors that lead to malfunction of the software suite. A detailed description of the architecture and functionality of GLORIAD InSight is given in Chapter 4.

GLORIAD InSight sought to provide a complete suite of tools that can feed off of an Argus data source as its input, process the flow data using user-defined databases and render visualizations according to user requirements, rather than requiring user to adapt to predefined measurements and user interfaces. GLORIAD InSight was the most comprehensive and most flexible network monitoring software available at the time.

Using different so-called dashboards, GLORIAD InSight could display network traffic as illustrated in Figures 1.4 and 1.5. The data was categorized as follows:

- Region: Africa, Asia, Australia, Central America, Europe, Middle East, North America, Scandinavia, and South America.

- Country: GLORIAD partners, USA, Canada, Russia, China, South Korea, Egypt, Singapore, India, Malaysia, Netherlands, and Scandinavia.

- Organization: Government agency, research institute, corporation, and university.

- Application: All applications, other TCP, other UDP, file and data transfer, web, mail, remote access, audio and video, database, network, peer-to-peer, icmp, personar, and others.

- Discipline: All disciplines, atmospheric sciences, biological sciences, engineering, environment, genome, geophysical sciences, health sciences, mathematics, military science, nuclear sciences, oceanography, physics, space science, technology, university and other, and interdisciplinary.

- Security: Bad actors, scanners, spammers, suspicious DNS, ICMP events

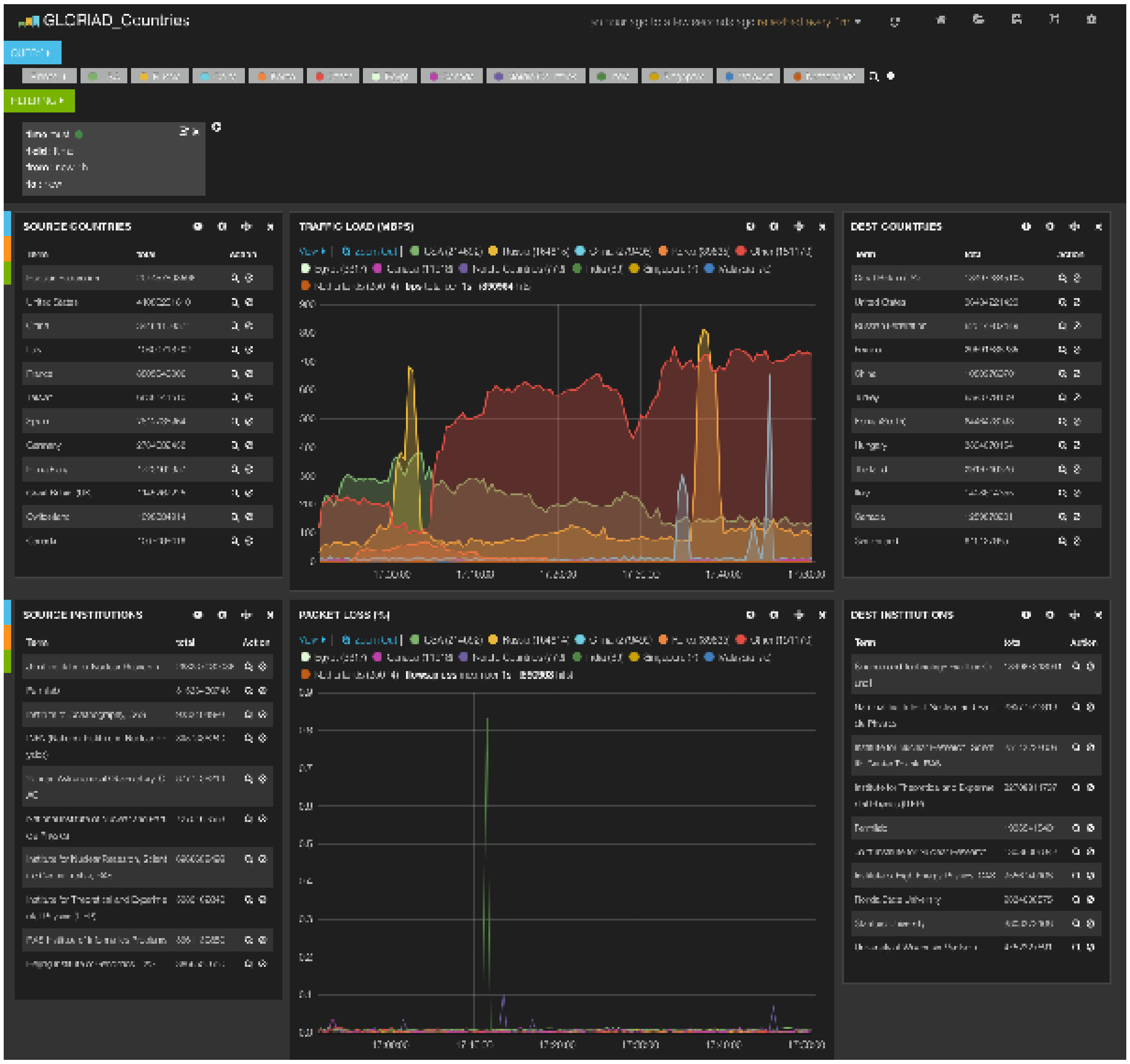


Figure 1.4 GLORIAD InSight Traffic Information.

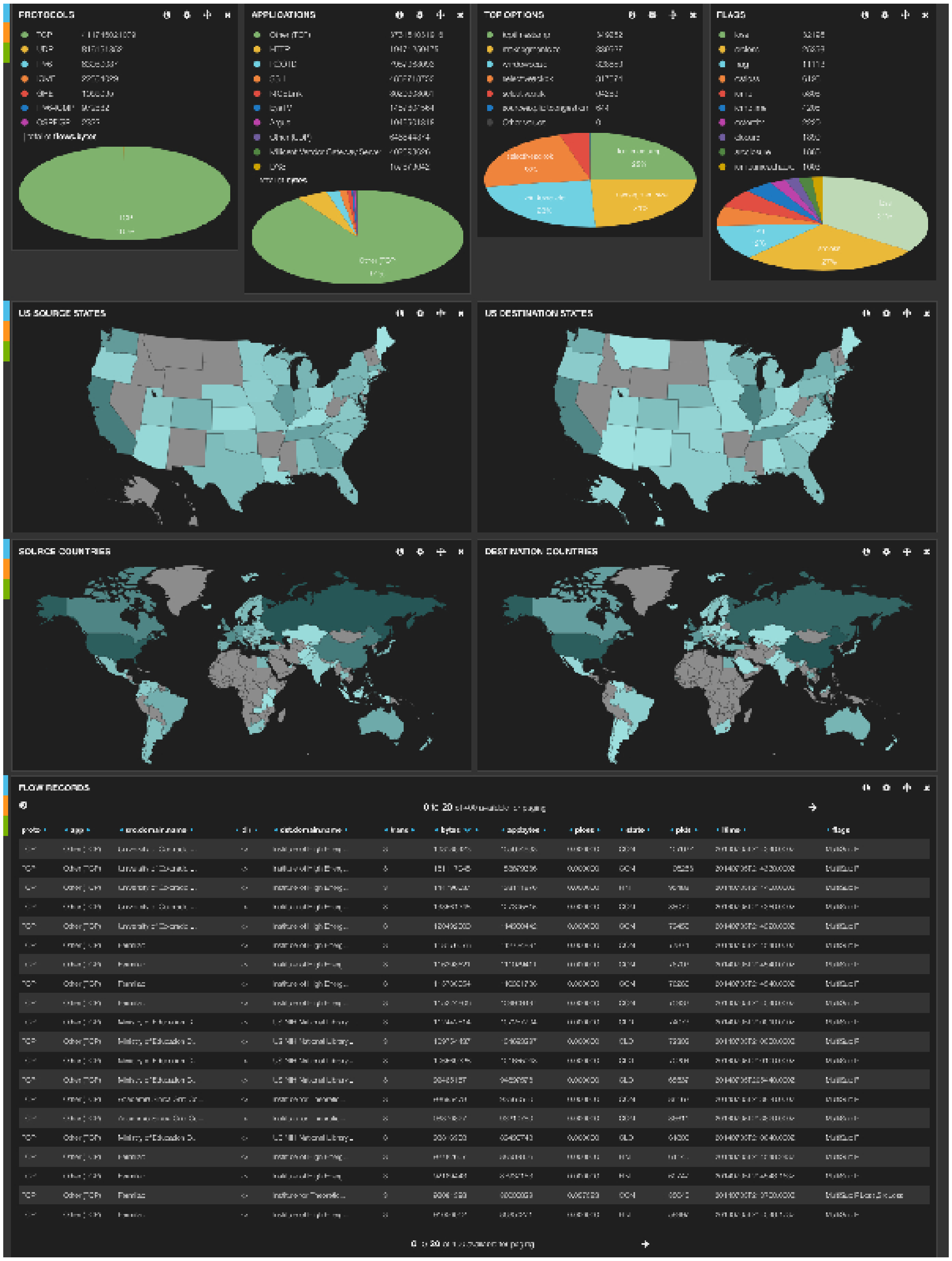


Figure 1.5 GLORIAD InSight Geographical Information

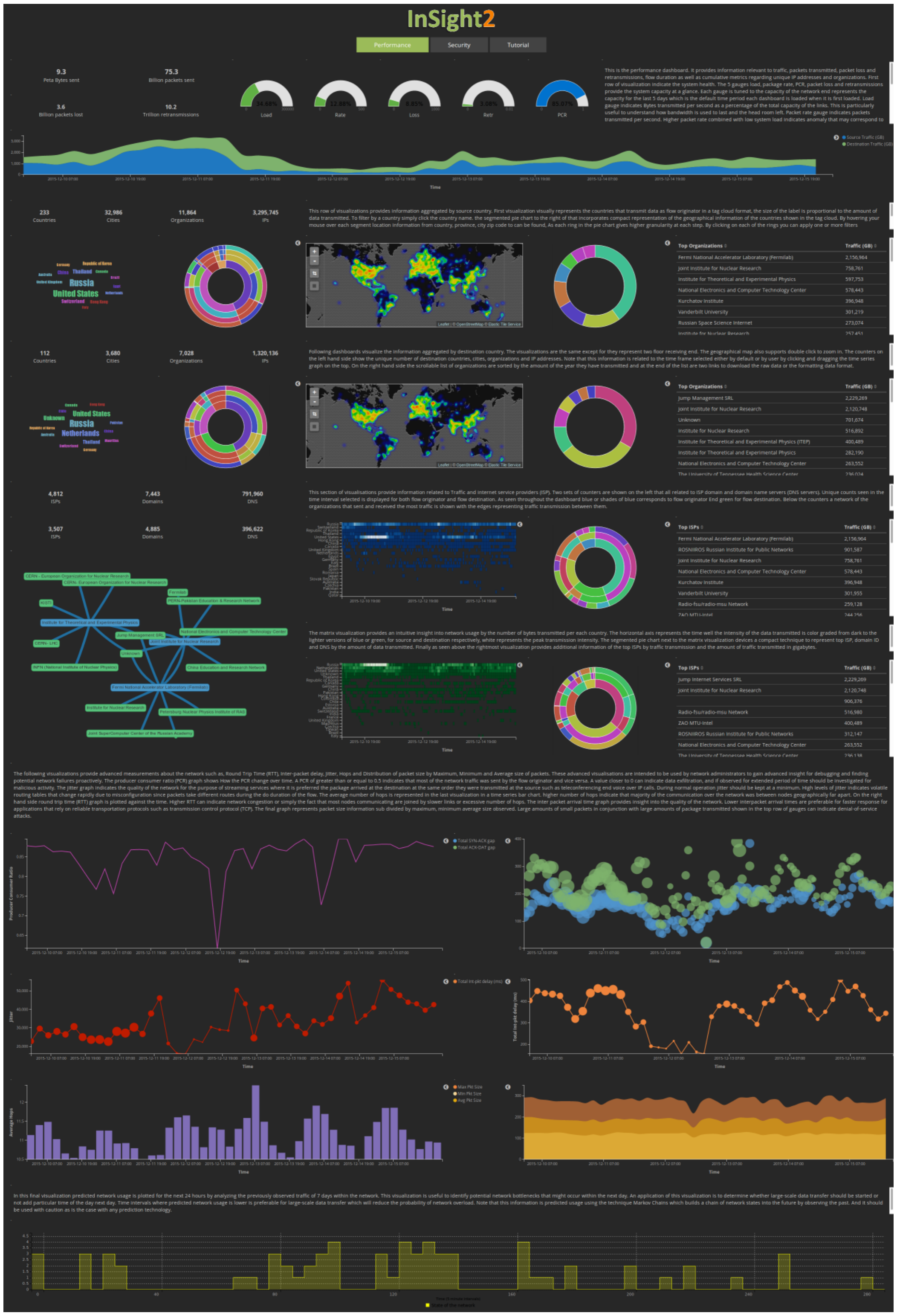
Argus network flow information is enriched into one of the above categories and stored in an Elasticsearch database. Kibana, a visualization rendering software that has tight integration with Elasticsearch and is a part of the Elastic Suite, is used to visualize the data in various graphs, charts, tables and geographical maps [9]. Topological information was displayed using regions and states as seen in Figure 1.5. Each of them is accompanied by textual information which showed top information sorted in descending order. Collections of visualizations, known as dashboards, are created to show different aspects of the network. GLORIAD InSight was a valuable and unique tool that offered an unprecedented amount of information about the activity inside a network in a visually coherent manner.

When the GLORIAD project came to an end, maintenance of the servers used to store the Argus flow data, enrichment databases and the web based user interface stopped which lead to catastrophic operating system and disk failures. Development and maintenance of the InSight platform also stopped. This caused the software to fall behind on several updates of Elasticsearch, Kibana, and Java which in turn meant that GLORIAD InSight became inoperable.

1.1.4 InSight2 (2017)

As part of the NSF project "The InSight Advanced Performance Measurement System" sponsored under the IRNC-AMI (International Research Network Connections - Advanced Measurement Network Infrastructure) program, a new InSight platform, henceforth referred to as InSight2, has been developed as detailed in Chapter 5.

The complex software architecture of GLORIAD InSight with the many moving parts decreased maintainability and introduced many points of failure. From the hardware layout to the software implementation, GLORIAD InSight had to be actively maintained. This involved performing regular system health checks, keeping up to date with the changes to the APIs and software tools, replacing tools that stop being actively developed etc. The more complex the hardware setup became and the more components the software architecture needed, the less robust the entire system became. InSight2 with its fresh code base has been focused on addressing these issues from the very start of the development. A new user interface was furthermore designed and implemented and is shown in the Figure 1.6.



9.3 Peta Bytes sent

75.3 Billion packets sent

3.6 Billion packets lost

10.2 Trillion retransmissions

This is the performance dashboard. It provides information relevant to traffic, packets transmitted, packet loss and retransmissions, flow duration as well as cumulative metrics regarding unique IP addresses and organizations. First row of visualization indicate the system health. The 5 gauges load, package rate, PCR, packet loss and retransmissions provide the system capacity at a glance. Each gauge is tuned to the capacity of the network end represents the capacity for the last 5 days which is the default time period each dashboard is loaded when it is first loaded. Load gauge indicates Bytes transmitted per second as a percentage of the total capacity of the links. This is particularly useful to understand how bandwidth is used to last and the head room left. Packet rate gauge indicates packets transmitted per second. Higher packet rate combined with low system load indicates anomaly that may correspond to

233 Countries

32,986 Cities

11,864 Organizations

3,295,745 IPs

This row of visualizations provides information aggregated by source country. First visualization visually represents the countries that transmit data as flow originator in a tag cloud format, the size of the label is proportional to the amount of data transmitted. To filter by a country simply click the country name. the segmented pie chart to the right of that incorporates compact representation of the geographical information of the countries shown in the tag cloud. By hovering your mouse over each segment location information from country, province, city zip code to can be found, As each ring in the pie chart gives higher granularity at each step. By clicking on each of the rings you can apply one or more filters

Top Organizations	Traffic (GB)
Fermi National Accelerator Laboratory (Fermilab)	2,156,964
Joint Institute for Nuclear Research	758,761
Institute for Theoretical and Experimental Physics	597,753
National Electronics and Computer Technology Center	578,443
Kurchatov Institute	396,948
Vanderbilt University	301,219
Russian Space Science Internet	273,074
Institute for Nuclear Research	257,451

112 Countries

3,680 Cities

7,028 Organizations

1,320,136 IPs

Following dashboards visualize the information aggregated by destination country. The visualizations are the same except for they represent two floor receiving end. The geographical map also supports double click to zoom in. The counters on the left hand side show the unique number of destination countries, cities, organizations and IP addresses. Note that this information is related to the time frame selected either by default or by user by clicking and dragging the time series graph on the top. On the right hand side the scrollable list of organizations are sorted by the amount of the year they have transmitted and at the end of the list are two links to download the raw data or the formatting data format.

Top Organizations	Traffic (GB)
Jump Management SRL	2,229,269
Joint Institute for Nuclear Research	2,120,748
Unknown	701,674
Institute for Nuclear Research	516,892
Institute for Theoretical and Experimental Physics (ITEP)	400,489
Institute for Theoretical and Experimental Physics	282,190
National Electronics and Computer Technology Center	263,552
The University of Tennessee Health Science Center	236,024

4,812 ISPs

7,443 Domains

791,960 DNS

3,507 ISPs

4,885 Domains

396,622 DNS

This section of visualisations provide information related to Traffic and internet service providers (ISP). Two sets of counters are shown on the left that all related to ISP domain and domain name servers (DNS servers). Unique counts seen in the time interval selected is displayed for both flow originator and flow destination. As seen throughout the dashboard blue or shades of blue corresponds to flow originator End green for flow destination. Below the counters a network of the organizations that sent and received the most traffic is shown with the edges representing traffic transmission between them.

Top ISPs	Traffic (GB)
Fermi National Accelerator Laboratory (Fermilab)	2,156,964
ROSNIIROS Russian Institute for Public Networks	901,587
Joint Institute for Nuclear Research	758,761
National Electronics and Computer Technology Center	578,443
Kurchatov Institute	396,948
Vanderbilt University	301,955
Radio-fsu/radio-msu Network	259,128
ZAO MTU-Intel	244,256

The matrix visualization provides an intuitive insight into network usage by the number of bytes transmitted per each country. The horizontal axis represents the time well the intensity of the data transmitted is color graded from dark to the lighter versions of blue or green, for source and destination respectively, white represents the peak transmission intensity. The segmented pie chart next to the matrix visualization devices a compact technique to represent top ISP, domain ID and DNS by the amount of data transmitted. Finally as seen above the rightmost visualization provides additional information of the top ISPs by traffic transmission and the amount of traffic transmitted in gigabytes.

Top ISPs	Traffic (GB)
Jump Internet Services SRL	2,229,269
Joint Institute for Nuclear Research	2,120,748
	906,376
Radio-fsu/radio-msu Network	516,980
ZAO MTU-Intel	400,489
ROSNIIROS Russian Institute for Public Networks	312,147
National Electronics and Computer Technology Center	263,552
The University of Tennessee Health Science Center	236,138

The following visualizations provide advanced measurements about the network such as, Round Trip Time (RTT), inter-packet delay, Jitter, Hops and Distribution of packet size by Maximum, Minimum and Average size of packets. These advanced visualisations are intended to be used by network administrators to gain advanced insight for debugging and finding potential network failures proactively. The producer consumer ratio (PCR) graph shows How the PCR change over time. A PCR of greater than or equal to 0.5 indicates that most of the network traffic was sent by the flow originator and vice versa. A value closer to 0 can indicate data exfiltration, and if observed for extended period of time should be investigated for malicious activity. The jitter graph indicates the quality of the network for the purpose of streaming services where it is preferred the package arrived at the destination at the same order they were transmitted at the source such as teleconferencing end voice over IP calls. During normal operation jitter should be kept at a minimum. High levels of jitter indicates volatile routing tables that change rapidly due to misconfiguration since packets take different routes during the do duration of the flow. The average number of hops is represented in the last visualization in a time series bar chart. higher number of hops indicate that majority of the communication over the network was between nodes geographically far apart. On the right hand side round trip time (RTT) graph is plotted against the time. Higher RTT can indicate network congestion or simply the fact that most nodes communicating are joined by slower links or excessive number of hops. The inter packet arrival time graph provides insight into the quality of the network. Lower interpacket arrival times are preferable for faster response for applications that rely on reliable transportation protocols such as transmission control protocol (TCP). The final graph represents packet size information sub divided by maximum, minimum average size observed. Large amounts of small packets in conjunction with large amounts of package transmitted shown in the top row of gauges can indicate denial-of-service attacks.

In this final visualization predicted network usage is plotted for the next 24 hours by analyzing the previously observed traffic of 7 days within the network. This visualization is useful to identify potential network bottlenecks that might occur within the next day. An application of this visualization is to determine whether large-scale data transfer should be started or not add particular time of the day next day. Time intervals where predicted network usage is lower is preferable for large-scale data transfer which will reduce the probability of network overload. Note that this information is predicted usage using the technique Markov Chains which builds a chain of network states into the future by observing the past. And it should be used with caution as is the case with any prediction technology.

Figure 1.6 InSight2 User Interface

InSight2 implements a new setup for the hardware servers which simplifies and improves efficiency, security, portability and ease of replication, a new software system architecture geared towards lean system design by eliminating multiple software components in favor of less moving parts, Python programming language and a custom web interface incorporating robust security features and state-of-the-art visualizations.

InSight2 migrates from Perl 5 scripts to Python which enables all the advantages of the latter more modern programming language. Python has native support for multi-threading and has a faster interpreter both of which contributes to increase the overall throughput of the platform. Due to its strict structure of coding a more maintainable and readable code base could be produced. Python also is advantageous in deployment since target host systems do not need to compile libraries since Linux-based operating systems either already come with Python pre-installed or is readily available from their respective repositories. GLORIAD InSight needed the Comprehensive Perl Archive Network (CPAN) repository to install its core components while native Python libraries are available from OS repositories. These libraries are furthermore constantly being updated allowing the installation package to easily execute a command to download and install relevant libraries directly.

One of InSight2's core development principles is the use of free and/or open source software to make it beneficial for everybody. All the related software and information sources used for the development and operation of InSight2 all readily and freely available. GLORIAD InSight used some fee-based databases for security tagging of the misbehaving IP addresses such as different types of botnets and bogons. InSight2 disposes of these databases in favor of only the free versions available online and are constantly being updated but at the same time keeping the option of adding paid information sources if provided by the end user. Databases used in InSight2 are described in depth in Chapter 5.

GLORIAD InSIght used separate scripts that processed data individually. They communicated with each other using a third-party tool called ZeroMQ. Monit was required to keep track of each script. Usage of sequential databases such as MySQL further increased the system complexity; when an asynchronous script needs access, the database gets locked and becomes unavailable to other scripts [13]. In order to mitigate this problem, the 'Elephant' kept a copy of the main MySQL database inside of a SQLite database. While this increased the data throughput, it also lead to increased disk usage which reduced their lifespan significantly [14]. Furthermore, sequential databases are not optimized for searching. The content is not indexed so any search query has to be performed sequentially. Any change to one of the many supporting software is prone to break the entire system which requires system-wide rewrite of the farm scripts.

InSight2 successfully addresses these drawbacks by changing the core architecture into a more robust and resilient one. The functionality of the farm is condensed into a core code named Enrichment Module (EM) eliminating processing and memory overhead incurred by invoking multiple instances of Perl interpreter for each animal.

GLORIAD InSight used an asynchronous event model and the implementation was single core based. The former lead to a complicated software architecture. The latter required some animals to schedule their processing at midnight since the blocking database transactions, decompression of the Argus archives and the processing would impact the performance of other critical animals. InSight2 has moved towards a synchronous multi-processing model by performing enrichment in a pipelined manner using multiple threads utilizing the modern multi-core CPU architecture. Information flows from different sources into the EM where the data is processed in parallel using all cores available on the host system and is subsequently sent to the Elasticsearch database without the need for intermediate message queues.

InSight2 disposes of the sequential MySQL and SQLite databases thereby eliminating blocking transactions that impact system performance by delaying read and write events. Instead, the superior Elasticsearch database is used for all storage needs. As an added benefit, this minimizes disk usage allowing the hardware to last its optimum lifespan, reducing the risk of drive failure in the long run.

Argus archives are compressed using ‘GZIP’ compression, and every decompression requires CPU time and memory, which are highly valuable in production environment. GLORIAD InSight required that this process take place every time an animal would access the archives. InSight2 eliminates the need for the archives to be decompressed more than once by decompressing the archives, reading them into memory, performing the enrichment and uploading into the database in a single pipeline.

InSight2 also aims to reduce the number of dashboards required to show the information since it requires the user and the browser to do extra work by having to load multiple web pages. This is achieved by designing the user interface to contain more information in a given area for a high information density. Improved and modern visualizations are chosen to combine information about various related aspects in a given dashboard. This not only results in high density dashboards but also improves the visual appeal. It is also optimized to be used in large monitors that constantly and frequently update and show real-time network information in dedicated areas ranging from network operations centers (NOC) to small-scale displays such as laptops and mobile interfaces.

InSight2 is engineered for speed and convenience by making sure that the platform is available from any compatible browser running on any platform regardless of the OS, CPU power and architecture, and memory capacity. InSight2 is optimized to download the dashboards only once. Subsequent updates that result in user applying different filters are handled dynamically using JavaScript. This enables InSight2 to update already downloaded visualizations with new data quickly and seamlessly. It also reduces the amount of data transmitted from web server for each filter applied. This is especially advantageous for mobile devices where they are limited by performance as well as network speed and bandwidth.

InSight2 has streamlined its deployment by providing one step installation package that handles checking of the target system for compatibility, installation of prerequisite software and setting up their configuration, installing database software and setting of the web interface. InSight2 deployment comes in two flavors, ‘Demo’ version and ‘Full’ version. ‘Demo’ version is for the purposes of demonstration which is geared towards quickly setting up the platform. It comes with 5 days of sample data, all the dashboards, and the web interface. The full version comes with compiled EM, GSR database in addition to the components in the ‘Demo’ version, which enables the end user to enrich and visualize their own Argus archives. Additional information sources are downloaded by InSight2 after being installed, such as Threats Database (TD), MaxMind GeoIP database. Site Specific Enrichment (SSE) database contains user specified information. It is loaded from the Elasticsearch database and used for enrichment.

Next steps include improving InSight2 by closely working with network administrators at Stanford University and Korea Institute of Science and

Technology Information (KISTI) to develop new features. This work is discussed in Chapter 7.

1.2 Thesis Outline

Chapter 2 discusses the overview of network monitoring software from network data capture to analysis. Different techniques for data capture including packet level and flow level is outlined. Existing traffic analysis solutions that utilize different data capture techniques are discussed. Chapter 3 outlines the recovery of the Argus data as well as the new setup of the servers and systems software. This chapter describes file system recovery, data extraction, data forensics and a design and implementation of computer server setup that is more manageable and resilient for failure in the future. Chapter 4 discusses the architecture of GLORIAD InSight and its drawbacks. A detailed description of the ‘Farm of Animals’ is provided along with a discussion of the Global Science Registry (GSR) database. Chapter 5 details the development of InSight2 including the new software architecture, database structure, data analysis and visualizations, user interface and security. It discusses the design and implementation of the Enrichment Module (EM), multi-core processing and the pipeline architecture. Chapter 6 discusses the deployment of InSight2 in the production environment. Chapter 7 provides results and conclusion. Chapter 9 outlines future work.

2. OVERVIEW OF NETWORK MONITORING SOFTWARE

In this chapter various techniques that are used to capture network data, data processing and data representation are explored. GLORIAD InSight and a few other related proprietary software that exist for network performance monitoring, are explored and discussed in comparison to InSight2.

2.1 Network Data Capture Techniques

In order to perform network data analysis either a passive or an active method has to be used for the data collection. Different data collection methodologies require different hardware and software setups that are applicable in different situations. We discuss strengths and weaknesses of each technique. There are two major network data collection techniques and they provide information at different levels of granularity. They are packet level and flow level data collection. In this section TCPDump, NetFlow and Argus are discussed. For the purpose of this research Argus has been selected as the data collection technique and the applicability and suitability of Argus is presented in depth compared to other existing techniques.

2.1.1 Packet Level Data Capture

At the packet level, either all the network data is directed through high performance capture device with two high bandwidth network ports acting as input and output respectively or mirrored using a network span port. A network span port is a specially configured port in a network device such as switch or router to output an exact copy of the traffic that is seen in all other ports. Packets are captured at the operating system (OS) level using TCPDump [15] and written to disk or internally piped to processing software for analysis purposes. This method yields the highest amount of information retention since the all the information in the packet is captured including user data. However, there are several significant disadvantages of this method.

- Since the entire network data is being directed through one capture device this capture device has to be of very high bandwidth and capable of processing the total amount of data transmitted per second, in real time without dropping packets. In other words, it is significantly expensive to device and maintain.

- In most cases, it is not feasible to route all the network traffic through one location due to geographic restrictions and topology.

- Increased latency is observed due to increased distance provisioned in order to accommodate to the new routing topology [16].

- Disk space and bandwidth requirements rise in linear relation to the incoming data rate. For growing systems, this translates to frequent upgrades of capture device components such as memory and network interfaces. This is not a solution that scales well.

- With the advancement of encryption techniques such as SSL and TLS packet level analysis of network data is impossible without SSL decryption or cryptanalysis because the content of the packet is encrypted, rendering the whole series of packets belonging to that particular connection between those source and destination to be unusable.

- In some cases, it is not legal to collect packet level information where sensitive information is transmitted and should not be stored, such as research and education networks where government agencies and research institutions share data with each other, which accounts for the majority of the users of GLORIAD.

As per the above reasons packet level data collection is not feasible for the purposes of this research. In order to mitigate these issues flow level data collection in the network is used instead.

2.1.2 Flow Level Data Capture

The basic principle of flow level data collection is to aggregate network packets into flows categorized by some given metrics. Below are five widely used metrics:

1. Source IP address
2. Destination IP address
3. Source port
4. Destination port
5. Protocol

This is called “Five Tuple” standard [17]. Most flow standards are based on this Five Tuple and in some cases adding more metrics. The most commonly used standards are Cisco Netflow, Juniper JFlow, Flow tools Inmon's sflow and Argus. We describe NetFlow and Argus in this section due to highest market adaptation of NetFlow and Argus being used in InSight2. Other flow capture standards are comparative to NetFlow.

Network data collection at the flow level yields much lower file size if archived to the disk and uses less memory to process if piped to processing software. This poses significant advantage over packet level data collection. Network flow capture devices can be smaller, generic/off-the-shelf and consumes less energy at the same time costing significantly less than packet capture devices. While packet capture devices require fast storage devices with higher write speeds such as SSDs flow level packet capture devices can utilize conventional hard disk drives (HDD) in most cases depending on the scale of the network.

Cisco NetFlow v9

Cisco defines a network flow standard named “Cisco NetFlow” [18]. It is the most widely used network flow standard. The latest version is NetFlow v9. As a uni-directional flow monitor, NetFlow reports state of each half of each conversation independently. It is proprietary and is implemented in many high-end Cisco devices such as switches and routers. NetFlow components and data flow is shown in the Figure 2.1. NetFlow standard categorizes flows using the seven metrics defined below:

- Source IP address
- Destination IP address
- Source port for UDP or TCP, 0 for other protocols
- Destination port for UDP or TCP, type and code for ICMP, or 0 for other protocols
- IP protocol
- IP Type of Service
- Ingress interface

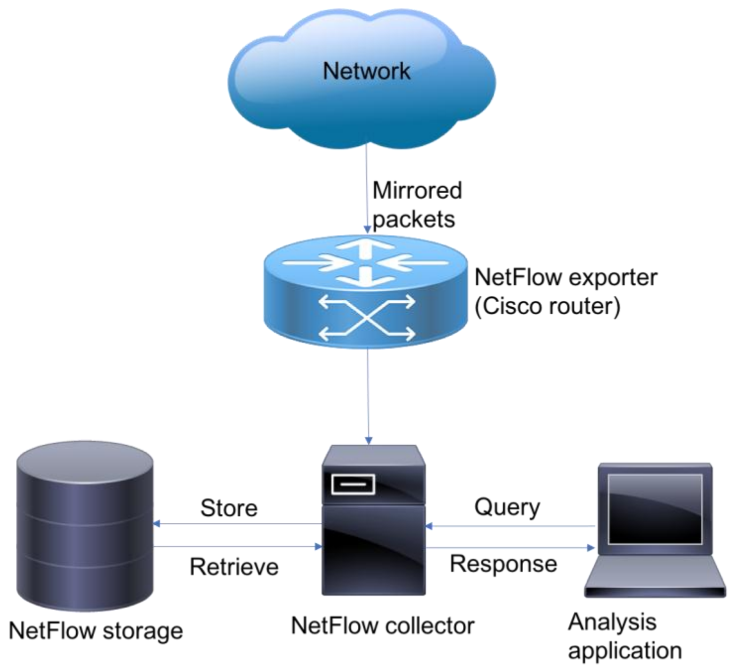


Figure 2.1 Cisco NetFlow Components

NetFlow standard defines three main components,

1. *Flow exporter*: Export flow information to one or more flow collectors.
2. *Flow collector*: Receive data from flow exporters, store and pre-process.
3. *Analysis application*: Used for analyzing the received data for various applications, such as NTOP.

QoSient Argus

Argus is an open-source network flow information generation and collection technique. InSight2 uses the Argus flow-monitoring system as its primary network activity data source. However, InSight2 is also capable of adapting to other protocols such as reading from Cisco NetFlow source. InSight2 has coupled Argus data generation and collection to its own data transport, processing and storage technology, using Elasticsearch, Kibana, and other technologies to provide an advanced network situational awareness capability.

Argus supports collection of advanced network flow measurements. It provides near-real-time comprehensive, multi-layer, bi-directional network data monitoring that is designed to support network operations, performance and security management. Argus provides structured data models and metrics for network entities such as Level 2 and Level 3 addresses, overlay identifiers, tunnel identifiers, service and application identifiers, as well as flow oriented utilization, transactional reachability, connectivity, availability, throughput, demand, load, loss and packet dynamics metrics, that can be used to describe complex application, system and path behaviors. Argus is capable of providing measurements from the flow information to a greater detail than NetFlow. Its core advantage is that it provides much more information from flow level network traces such that it almost negates the need for packet level data capture. Argus uses following six metrics to categorize packets into flows:

1. Source IP address
2. Destination IP address
3. Source port
4. Destination port
5. Protocol
6. Direction

Argus is a bi-directional flow aggregation protocol. This approach enables Argus to provide availability, connectivity, fault, performance and round trip measurements. These metrics are suitable for large-scale networks where network flows from different networks are aggregated into one place where directional information is more important than interface information as provided by Cisco NetFlow. Argus offers a superset of the functionalities offered by other flow monitoring protocols, thus it is compatible with competing technologies which enables it to read and convert Cisco Netflow, Juniper JFlow, Flow tools, and Inmon's sflow data without data loss since those technologies provide less information compared to Argus.

Argus also defines data exportation tools and collection nodes similar to Cisco NetFlow's Flow Exporter and Flow Collector. They are server and client components. The Argus server client architecture is shown in the Figure 2.2. Server is responsible for converting packet traces into Argus format while collection agent named Radium server collects Argus records sent by the Argus server. In addition, Argus provides supporting tools to read and filter records which are called Argus clients. Argus clients as a collective provide various filters and aggregation methods. We use these tools to read Argus filter and data.

These Argus-clients are in the form of binaries and scripts and are provided with the Argus-clients package. They can be installed using system repositories and is labeled as 'argus-clients' or can be compiled and installed individually. Argus clients are described in the Appendix A. As of the writing the Argus-server package was in its version 3.0.8.2 and the clients package 3.0.8.2. This suite of tools are being used by universities, corporations, and government entities to keep track of their network traffic that belongs to internal communication, incoming and outgoing. Argus clients are capable of reading from Argus files from the disk or network resources or directly from network sockets as a live stream. Its configuration is stored in 'raconf' file. The Argus-clients package consists of 8 core clients and 27 peripheral clients. In order to compile the Argus server and the client packages 'gcc', 'make', 'bison', 'libpcap', 'libpcap-dev', and 'flex' are prerequisite. These clients can be stacked together using Linux pipelines. Pipelines allow processing Argus data in a serial manner using the output of one client for the input of another client iteratively. This enables the user to take advantage of the different functionalities of the clients in order to achieve complex processing functions.

Argus requires very few system resources and has been proven to run even in router firmware such as OpenWRT, allowing it to achieve the functionality of Cisco NetFlow within a router. Figure 2.2 outlines the components of Argus and its dataflow.

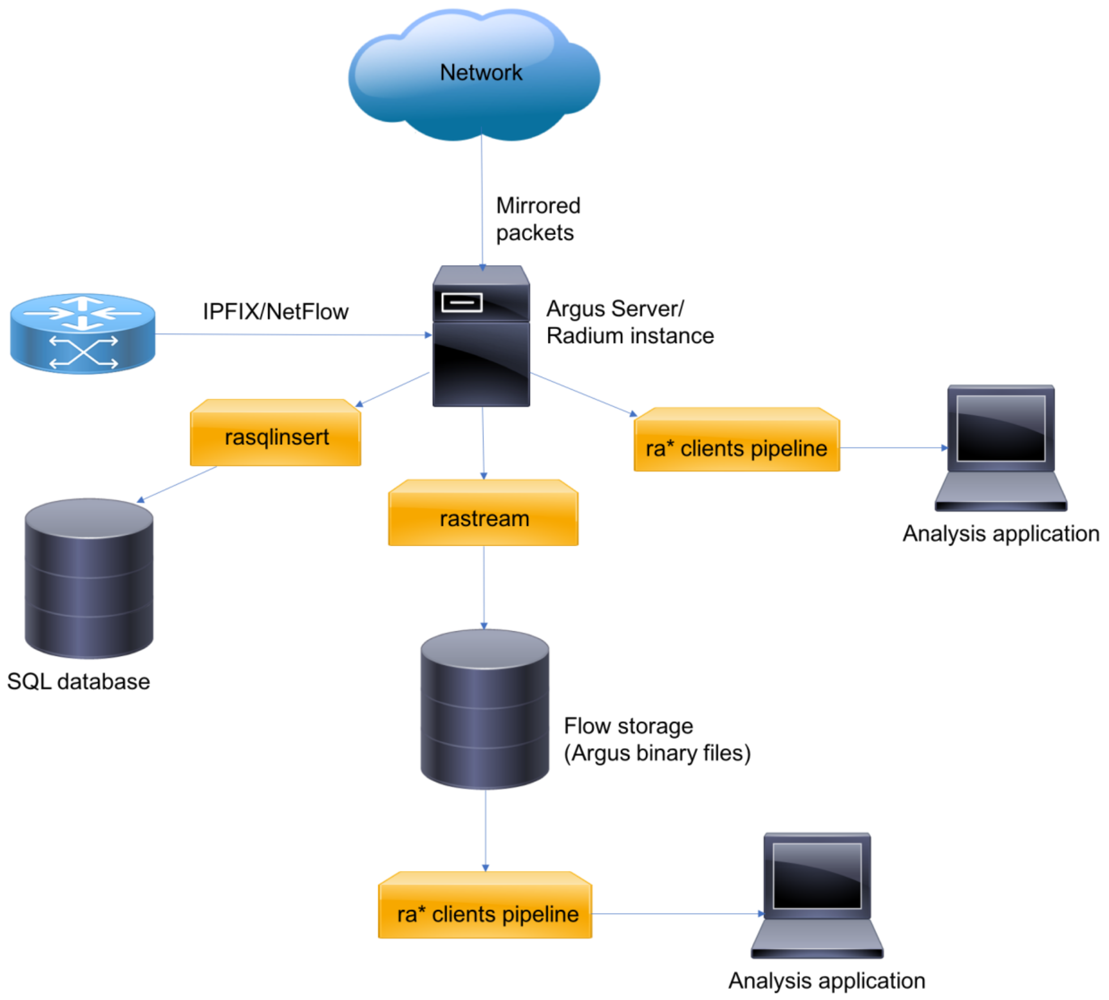


Figure 2.2 QoSient Argus Components

2.2 Existing Network Monitoring Solutions

There exist a few solutions for the purpose of network traffic analysis with different advantages and disadvantages of specific to each other. Some are focused on network performance analytics while others are security focused. Input data ranges between Cisco NetFlow and Argus.

2.2.1 Performance Monitoring

There exist a few software solutions that address performance monitoring needs of a network. Most of them take input from NetFlow while some take Argus input.

NTOP-NG is web-based traffic analysis and flow collection software that is aimed at monitoring network usage [19]. This software is cross platform and capable of sorting network traffic according to IP address, port, Layer 7 protocol, throughput and autonomous systems (AS). It is capable of producing reports on statistics on geo-location information on IP protocol usage sorted by protocol type. It is also capable of displaying Layer 2 information such as ARP statistics. It allows exploration of historical data that is stored in MySQL database. However it lacks customizability and only provides information provided by NetFlow standard. Its biggest advantage is that it is capable of directly reading from NetFlow output.

However, NTOP-NG lacks the ability to enrich data according to user maintained database. This limits the functionality of the software to only visualize essential information about the network defined by the software maintainers. No security information is generated and it is incapable of detecting evolving threats. That said, NTOP-NG may be an adequate solution overall for small-scale networks that only need the functionality of a performance monitor.

Spiceworks Network Monitor displays network and server related information in a dashboard including CPU disk and memory usage [20]. The extension into the server health includes I/O performance, individual OS processes and services as well as packet loss. The software is capable of restarting applications from within the software in a preprogrammed sequence. One of the unique features it offers is checking if connected servers are up and running using ping, HTTP, SIP, or SSH. The greatest advantage of Spiceworks Network Monitor is that it is completely free for use. However, it’s not open source, making it impossible for the end user to customize the software beyond its intended functionality such as tagging network flows by the user-defined databases. Having insight into the server components provides an extra layer of information which can compensate for the lack of security features, depending on the needs of the end user.

Logic Monitor is capable of automatically discovering network devices and interfaces without needing manual setup [21]. It can monitor CPU, memory, temperature, fan and other hardware metrics which gives it a unique edge among the software used for network and server health information. Network capabilities of this software include network throughput, packet rates, error rates and utilization, which includes Power over Ethernet (PoE) loads and wireless access point (AP) monitoring. Higher tiers of this fee-based software includes support for quality of service (QoS) policies IP service level agreement (IP-SLA) profiles, virtual private networks (VPN) and Voice over IP (VoIP) features.

This solution is better suited for applications where server health is needed to be monitored in conjunction with firewalls, routers, switches and wireless devices that are used to connect end users to these servers. It provides different aspect of server maintenance compared to NTOP-NG while providing insight into server hardware. One of the best features of this is the capability of integrating into the output of NetFlow, Jflow, and sFlow providing flexibility as to which network flow information source it can be plugged into. Lack of customizability in the form of user-defined information makes the system more suitable for specific applications where policy enforcement should be monitored and given priority compared to security features or customizability.

Pressler AG PRTG [22], focuses on the Windows environment. It supports data sources from SNMP, WMI Windows performance counters, secure shell (SSH), database, and packet sniffing. One of the differentiating factors of this software is capability of monitoring database held such as SQL databases. Compared to the other aforementioned solutions, apart from the database monitoring capabilities no significant features are offered. Strengths in common with the other solutions include CPU load, uptime, memory usage, disk usage, QoS measurements and cloud services health metrics.

2.2.2 Security Monitoring

Security monitoring software that utilize standard flow protocols that provide a standard set of tools applicable for wide variety of systems are less prevalent than performance monitoring tools. This can be explained by the fact that most security focused tools are developed “in-house” by each organization to suit to their requirements and environment.

Alienvault is a Network Vulnerability Assessment [23] tool which excels in intuitive and easy user interface. It is proprietary software that can also act as an intrusion detection system (IDS) on its own. It is capable of analyzing packets using its proprietary USM appliance. One of the biggest strengths of Alienvault is that it is capable of running from cloud-based deployments where the user is not required to install any software at the customer premises. However, as the complete system it lacks the ability to extend its functionality for performance metrics that NTOP-NG offers. The biggest drawback of Alienvault is that it is proprietary and subscription is needed for the operation.

3. SERVER CONFIGURATION

Before the development of the new platform InSight2 could be started, some preparation work was needed. During the time gap when GLORIAD stopped actively developing the InSight platform and the new InSight2 development started, the data server holding the four years of historical Argus flow data experienced a catastrophic file system failure causing the data contained within the server to be inaccessible. This chapter discusses the measures taken in order to recover these valuable archives and preventive measures taken to prevent such mishaps in the future. Limited resources were available to diagnose the cause of the failure since the OS debugging tools such as ‘dmesg’ and system files were out of reach. Unconventional diagnosis methods such as capturing the output of frame buffer based remote login client provided by Cisco Integrated Management Controller (CIMC) interface using high speed camera were used for data recovery of the GLORIAD Argus archives and they were successfully recovered. Finally, servers were formatted and their set-up was changed to a more robust, manageable, optimized scheme focused on speed and data protection.

3.1 Hardware Configuration

The GLORIAD project had 15 servers. None of them were being maintained at the time work on this thesis started. Appendix B provides a list of servers, the services they ran, their OS version, memory capacity, disk capacity, processor and disk arrangement. Some servers were offline and were not reachable and the assessment results are listed in the Table 3.1 The individual IP addresses were ping swept to check their status, whether there were online or not, their reach ability through secure shell (SSH), reachability of the Cisco management interface CIMC and additional comments are listed. Access to some of the servers could not be established since they were not documented. With the help of Joel Dickens, the system administrator at Cisco facility where the servers are located, access to the critical servers that held valuable data was established.

Table 3.1 Reachability Assessment of GLORIAD Servers

Server Name	CIMC reachable?	SSH reachable?	Comments
Bluemac	YES	YES	Fully functional
Anodos	NO	NO	CIMC HTTP port is open, login page does not load
Wingfold	NO	NO	CIMC HTTP port is open, login page does not load
Princess	YES	NO	SSH port closed
Goblin	YES	NO	SSH port closed
Curdie	YES	NO	SSH port closed
Mithril	UNKOWN	YES	CIMC IP is unknown
Bulika	UNKOWN	YES	CIMC IP is unknown
Lona	UNKOWN	YES	CIMC IP is unknown
Lilith	UNKOWN	YES	CIMC IP is unknown

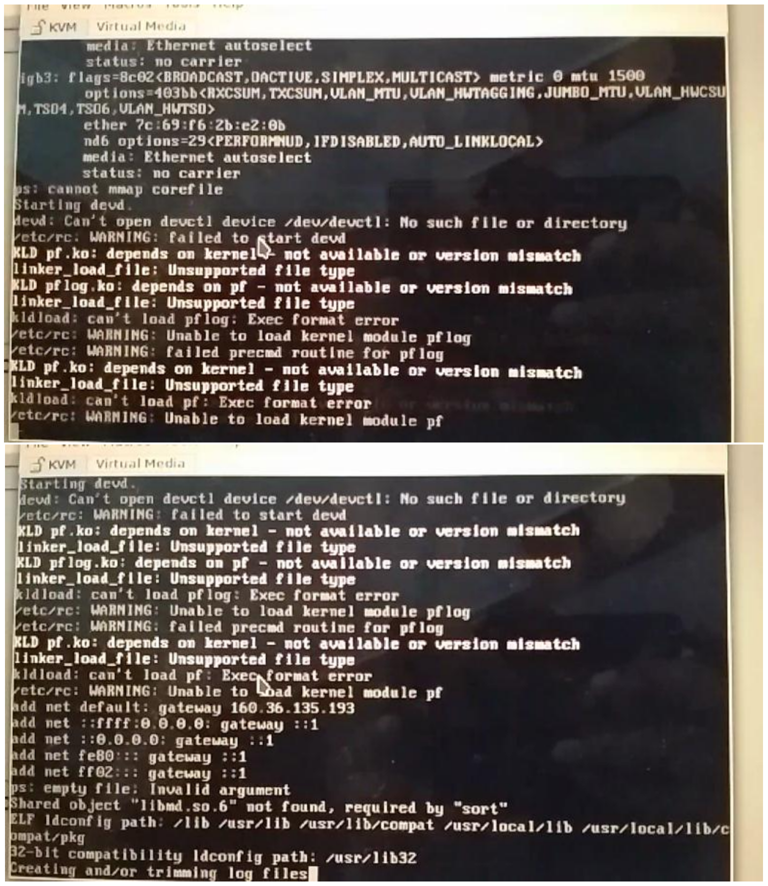
It was found that ‘Bluemac’ which held a copy of the InSight web interface and GSR database server had problems with its battery backup as well as degraded RAID configuration.

‘Mithril’, the data storage server used for archiving purposes of Argus data of the GLORIAD network contains 12 spinning hard drives with capacity of 3TB each amounting to 30 TB in total, arranged in a redundant array using ZFS RAIDz2 file system. It is a Cisco Blade server with 16 cores, 32 threads and 64 GB Random Access Memory (RAM). It was running on FreeBSD version 10.1 at the time. ZFS is a robust file system that can tolerate up to 2 drive failures and still continue to function thanks to its parity drive mechanism similar to standard Redundant Array of Independent Disks (RAID). It provides a strong error correction algorithm that scans drive errors in real time and corrects those errors while the server is operational and online, alleviating the necessity to bring the server offline to fix block level file-system errors. ZFS was first developed by FreeBSD development team and was later reported into other Linux operating systems. When the development of InSight2 started, this server had degraded due to lack of maintenance causing multiple OS level components to fail rendering the server into an unbootable state. This server held 14.6 TB of Argus flow archives from 2012 to 2015, inaccessible due to ZFS failure. Remote login into the OS was not possible and network stack failed to function, preventing data extraction. Root cause of the failure was investigated and it was found that multiple kernel modules were corrupted due to lack of maintenance.

Since all the attempts to boot the ‘Mithril’ server remotely using SSH, which was the only remote login protocol that was enabled at the time, were futile, it was booted from the CIMC interface. CIMC provides an additional interface using a Java applet that combines the frame-buffered video output, keyboard and mouse inputs, named ‘Keyboard Video Mouse’ (KVM) interface. It is also capable of remotely formatting to system, change the drive RAID configuration, uploading an ISO image so that new OS can be installed remotely, turning on and shutting down installed operating systems. It was seen that FreeBSD OS would hang midway during the boot process since further modules cannot be loaded without the corrupt prerequisite modules. It was not possible to find out which modules were corrupted since the output of the boot process is not recorded and the log files are not accessible. Each time when the server was rebooted it would quickly scroll the output of the boot process and hang at the same position.

High speed camera that is capable of shooting video at 240 frames per second was used to capture the output of the boot process and 2 relevant frames that show the file system corruption is shown in Figure 3.1. By reviewing the video frame by frame corrupted system modules were identified and listed since there was no way to record the text output to file or scroll up the output, as it was displayed frame by frame.

The following screenshots show the output of the high-speed video of the output of the boot information. It can be seen that several system modules are corrupted and prevents loading relevant kernel modules.



```
        media: Ethernet autoselect
        status: no carrier
igb3: flags=8c02<BROADCAST,OACTIVE,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=403bb<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,JUMBO_MTU,VLAN_HWCSU
M,TSO4,TSO6,VLAN_HWTSO>
        ether 7c:69:f6:2b:e2:0b
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
        media: Ethernet autoselect
        status: no carrier
ps: cannot mmap corefile
Starting devd.
devd: Can't open devctl device /dev/devctl: No such file or directory
/etc/rc: WARNING: failed to start devd
KLD pf.ko: depends on kernel - not available or version mismatch
linker_load_file: Unsupported file type
KLD pflog.ko: depends on pf - not available or version mismatch
linker_load_file: Unsupported file type
kldload: can't load pflog: Exec format error
/etc/rc: WARNING: Unable to load kernel module pflog
/etc/rc: WARNING: failed precmd routine for pflog
KLD pf.ko: depends on kernel - not available or version mismatch
linker_load_file: Unsupported file type
kldload: can't load pf: Exec format error
/etc/rc: WARNING: Unable to load kernel module pf
```

```
Starting devd.
devd: Can't open devctl device /dev/devctl: No such file or directory
/etc/rc: WARNING: failed to start devd
KLD pf.ko: depends on kernel - not available or version mismatch
linker_load_file: Unsupported file type
KLD pflog.ko: depends on pf - not available or version mismatch
linker_load_file: Unsupported file type
kldload: can't load pflog: Exec format error
/etc/rc: WARNING: Unable to load kernel module pflog
/etc/rc: WARNING: failed precmd routine for pflog
KLD pf.ko: depends on kernel - not available or version mismatch
linker_load_file: Unsupported file type
kldload: can't load pf: Exec format error
/etc/rc: WARNING: Unable to load kernel module pf
add net default: gateway 160.36.135.193
add net ::ffff:0.0.0.0: gateway ::1
add net ::0.0.0.0: gateway ::1
add net fe80::: gateway ::1
add net ff02::: gateway ::1
ps: empty file: Invalid argument
Shared object "libmd.so.6" not found, required by "sort"
ELF ldconfig path: /lib /usr/lib /usr/lib/compat /usr/local/lib /usr/local/lib/c
ompat/pkg
32-bit compatibility ldconfig path: /usr/lib32
Creating and/or trimming log files
```

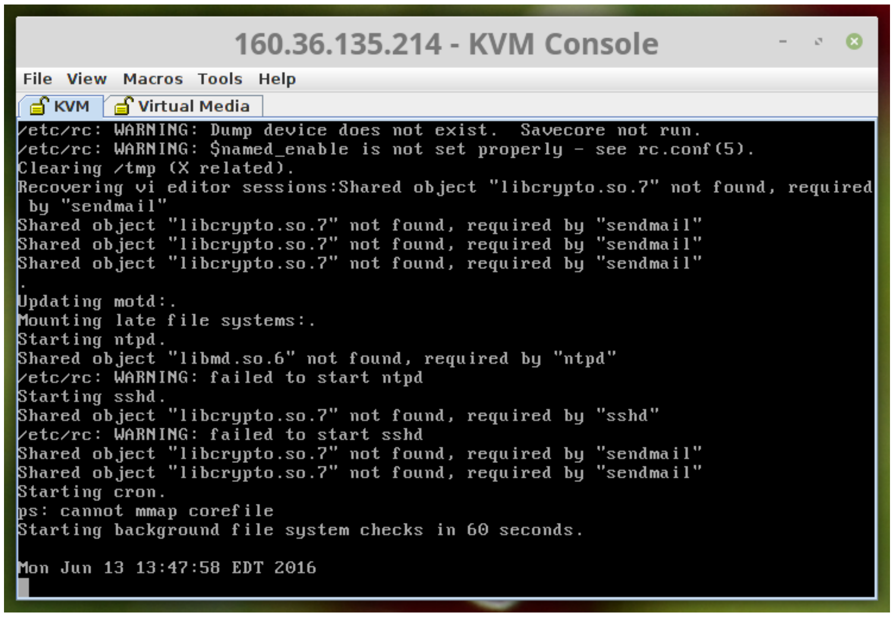
Figure 3.1 Output of the High Speed Camera Capture

CIMC's ability to load ISO images remotely was utilized to upload a standard Kali Linux Lite edition, Ubuntu 16.04 server and desktop live images. Most Linux operating systems including Ubuntu can be booted without actual installation into the hard disk drive. This is greatly beneficial for diagnostics and fixing unbootable systems. Kali Linux Lite edition, Ubuntu 16.04 server live images were not supported in this Cisco Blade servers and only Ubuntu 16.04 desktop edition booted up properly. This image was mounted using CIMC image mounting tool then the server was rebooted into the setup mode to configure the boot order to redirect the bootloader to load from the uploaded ISO image instead of the internal hard disk drive. The final reboot loaded the live OS. This allowed the access to the system modules. However, it was not possible to use the live OS to extract the data, since the ZFS file system was not assembled and was spread across multiple hard disk drives. The necessary configuration for the ZFS drive assembly was found in the root partition of 'Mithril' and without properly booting into the FreeBSD OS it was not possible to assemble the array of drives.

Corrupted system modules were replaced with fresh ones. The health of the individual disk drives was tested using the ‘fdisk’ tool found in the Ubuntu live OS. ‘Mithril’ was booted back into the setup mode to revert the changes to the boot order in the bootloader section and rebooted. This setting allowed it to boot back to the internal FreeBSD installation on its own hard disk drive.

Boot process went past the point where it hung last and was prompted with FreeBSD boot screen. This boot screen had options to select user mode ranging from single user to multi-user. First attempt to boot using the default boot mode, multi user mode, failed. During subsequent attempts to boot using the multi user mode server would hang again trying to load additional modules that are required for multi-user environment. Instead of trying to fix the modules preventing booting into this default mode server was booted into the single user mode by dropping into shell in ‘single user mode’ in the FreeBSD bootloader. ‘Single user mode’ is a mode used for emergency purposes for data recovery in Linux systems. Finally ‘Mithril’ showed a terminal which indicated the OS recovery was a success.

Figure 3.2 shows the successful recovery of the ‘Mithril’ server. It is indicated by the date prompt in the end. More related screenshots are provided in the Appendix B.



```
/etc/rc: WARNING: Dump device does not exist.  Savecore not run.
/etc/rc: WARNING: $named_enable is not set properly - see rc.conf(5).
Clearing /tmp (X related).
Recovering vi editor sessions:Shared object "libcrypto.so.7" not found, required
 by "sendmail"
Shared object "libcrypto.so.7" not found, required by "sendmail"
Shared object "libcrypto.so.7" not found, required by "sendmail"
Shared object "libcrypto.so.7" not found, required by "sendmail"
.
Updating motd:.
Mounting late file systems:.
Starting ntpd.
Shared object "libmd.so.6" not found, required by "ntpd"
/etc/rc: WARNING: failed to start ntpd
Starting sshd.
Shared object "libcrypto.so.7" not found, required by "sshd"
/etc/rc: WARNING: failed to start sshd
Shared object "libcrypto.so.7" not found, required by "sendmail"
Shared object "libcrypto.so.7" not found, required by "sendmail"
Starting cron.
ps: cannot mmap corefile
Starting background file system checks in 60 seconds.

Mon Jun 13 13:47:58 EDT 2016
```

Figure 3.2 Obtaining Access to Mithril

3.2 Argus Archives and Other Data Recovery

Within ‘Mithril’ server contained all the Argus flow archives from 2012 to 2015. This data is highly valuable since they represent real activity from the GLORIAD network. Recovery of this dataset was integral for this research and for the future study. With the rich feature set provided by Argus valuable insights and analytics can be derived from this dataset about the behavior of large-scale networks. First the corrupted ZFS assembly of disks array was reconstructed. Then the Argus and other data sets were extracted and backed up. This section illustrates this process and describes data contained within these datasets. ZFS manipulation tools ‘*zfs*’ and ‘*zfsutils*’ were installed into the Ubuntu live OS using the online Ubuntu repositories and the ZFS assembly was verified for any errors using the ‘*zpool’* tool. After verification passed ZFS pool was reconstructed.

3.2.1 Extracting Argus Archives

Network connectivity was checked with the replaced system modules but could not gain functionality to get Secure Shell (‘ssh’) working to securely transmit the archives even though system was recovered to a state that it completes boot procedure and provides a login prompt. As seen in the Figure 3.2 some of the replaced crypto modules did not work as expected which led to login and data transfer tools that use Secure Socket Layer (SSL) encryption unusable, which is a requirement for Secure Copy (‘scp’) to run.

Solution to this problem was found with the help of data processing servers, ‘Bulika’, ‘Lona’ and ‘Lilith’ which are high-end servers intended to be used for processing located at the same server room at the Cisco facility. These servers have 20 CPU cores, 384GB RAM and 12TB Solid State Drive (SSD) high-speed storage capacity. However the combined capacity of all 3 servers was adequate to store all the data from ‘Mithril’. Data copy was initiated by physically re-routing ‘Mithril’ to ‘Bulika’, ‘Lona’ and ‘Lilith’ using Ethernet cable physically one at a time in order to isolate the connection from Internet. This required physical presence at the Cisco facility, since it was not performed using remote login but using the physical keyboard and mouse connected to the server. Physical isolation was required since the standard secure tool for data copy using SSL encryption, ‘*scp*’, failed to function and ‘*rcp*’, which one of the tools in the minimalistic ‘*rcp’*, ‘*rlogin’* and ‘*rsh’* family of tools which offers no data confidentiality was used to copy the data to Bulika, Lona and Lilith. Even though end-to-end encryption was not used it was perfectly safe since connection was directly peer-to-peer and isolated from the internet. This procedure took over 2 weeks. This resulted in an exact copy of all Argus archives being copied into 3 servers. At this point the Argus archives were safely extracted from failing ‘Mithril’.

3.2.2 Data Forensics

Additional information was required to be backed up from the GLORIAD archives. Global Science Registry (GSR) which was an integral part of GLORIAD, paid versions of geo IP databases which contains IP address to location mapping and legacy archives of security information which were collectively known as ‘Bad Actors’ were backed up from the relevant servers. So the following databases were recovered and backed up:

1. GSR MySQL database
2. GeoIP database
3. ‘Bad Actors’ database

GSR database

GSR is a carefully curated repository of 14,000+ institutions and projects using global research and education networks. This database contains information that maps IP addresses to domains, domain name servers (DNS), names of institutions, disciplines, countries, provinces, cities, and zip codes etc. GLORIAD InSight used these archives to tag each network flow with its information. It was stored in a MySQL database and was recovered from the server ‘Bluemac’ which was in the format of MySQL dump of 8.6GB. ‘Bluemac’ was one of the servers that GLORIAD that used to host the InSight and store GSR. This was in the form of a large script which when executed through MySQL terminal Imports all the information contained in the dump to be added into the database creating necessary table definitions. It was useful in building InSight2 where this information was uploaded into Elasticsearch database for faster access since Elasticsearch supports indexing which is optimized for searching. This information was the most up-to-date information at the time when the GLORIAD project ended.

GeoIP database

Old archives of geographical location data provided by Maxmind geo IP database was also found in the ‘Bluemac’ server where the old ‘Farm of Animals’ used to be when it was operational. These provided up to date geo locations of the IP addresses at the time when GLORIAD InSight was operational. Currently InSight uses free version of the up to date Maxmind geographical location information. Collecting this information would enable us in the future to implement advanced metrics how IP addresses belonging to the GLORIAD network change overtime.

Bad Actors database

‘Bad Actors’ archives contained the following information:

- Emerging threats
- CYMRU Bogons
- IP addresses belonging to Zues botnet
- IP addresses belonging to Feodo botnet
- IP addresses belonging to Palevo botnet
- IP addresses belonging to Spyeye botnet

First dataset, a paid-for dataset containing emerging threats by Proofpoint [24], was subscription-based, that provided reputation intelligence about IP addresses which is built using its proprietary algorithm. This helps identify IP addresses domains that pose risk in order to prevent attacks. This database consists of separate list for IP addresses and domains, they confidence score about its reputation and constant updates. The bogon dataset provided by Team CYMRU [25] lists the IP addresses that belong to bogons. Internet routing table contains IP addresses that are allocated for public and private purposes. A bogon is a prefix of IP addresses that are not intended to be present in the internet routing table. IP addresses that belong to these bogon prefixes are malicious and should not be routed through the internet and they are commonly found during Distributed Denial-of-Service attacks (DDoS) as the source IP. The other datasets belong to different kinds of botnets. A botnet is a collection of computers belonging to individuals that are infected with malicious software and controlled as a group to perform the commands of the botnet master. These networks are used to launch denial-of-service attacks send spam messages, Bitcoin mining, distribution of spyware, and performing click fraud, which occurs when the infected computer visits websites without user’s knowledge to generate manufactured traffic to earn money. Some of these Bad Actors in the network are unaware of the fact that they are infected making them hard to detect by the owners. They are controlled by either centralized or decentralized command and control mechanism performed using Technologies such as Internet Relay Chat (IRC) and peer-to-peer protocol (P2P) respectively. Currently InSight uses a subsection of these datasets which are free in addition to the other free security data sets that were found during the development phase. These archived data sets provide up to date information about such infected computers at the time GLORIAD InSight was operational which will enable us in the future to determine how those bad actors evolved over time, whether they compromised IP addresses return to normal state or vice versa.

3.3 Preventive Measures Taken

3.3.1 Contingency Backups

GLORIAD used a backup storage device named ‘QNAP’ to back-up the essential information which contained a backup of Argus archives from 2012 to 2014 and the GSR. QNAP data storage device with the capacity of 16 hard disk drives is basically an ARM server optimized for data storage needs. It is essentially a network attached storage (NAS) server system that uses a proprietary operating system based on Linux, which offers advanced functionality built into the OS, such as Digital Living network Alliance (DLNA) multimedia streaming, data encryption, cloud backup, built-in dynamic DNS service which allows it to have a public dynamic IP address such as the IP addresses which are received from ISPs which change every time the uplink connection is refreshed and still be accessible over the internet by giving it a domain name that updates the IP address mapping automatically, the ability to manage the server using web browser and the ability to install third-party apps to extend functionality. However none of the advanced functionalities of this server were used and it was only used as a storage server for Argus archives. Login username and password for QNAP were not documented which prevented access. The only other way was to physically remove the hard drives, install them in another server, reassemble the RAID configuration, and copy the data, since the data was not encrypted in the hard disk drives. But due to the lack of such equipment it was impossible to access the data located in QNAP even though it contained Argus archives and GSR.

QNAP has a reset switch which pressed for more than 10 seconds will reset the configuration and delete all the files stored within. As a last resort to gain access into the system this method was used during which process all the data was destroyed. But since the purpose was to use the system to back up the already extracted data this was not an issue. After gaining access it was discovered that one of the hard disk drives was failed. This was replaced by a new one and OS was updated to apply the latest security patches. Screenshot in the Appendix B shows the QNAP drive information.

Redundant array of Independent disks (RAID) is the technology used to merge multiple drives into one virtual drive that provides higher degree of data protection compared to single drives. RAID has different versions suited for different levels of data redundancy, RAID 1 Is a mirror of all the data stored, RAID 5 uses Block Level striping with distributed parity which offers continued function after one drive failure. RAID 6 uses 2 parity drives to offer data redundancy for 2 drive failures. Both these methods require one or two drives respectively used for parity purposes which cannot be used for data storage purposes. QNAP's disk drives were arranged in an RAID 5 configuration which allowed the system to function even though there was a failed hard drive. This essentially reduces the number of usable hard drives by one which is used as a parity drive.

When data is stored in physical hard drives it consumes more than the total capacity of the data. This space is consumed by formatting block information and file header information. So even though total Argus archive capacity was 14.6TB, the capacity of QNAP of 16TB was not adequate to set up any RAID configuration. The RAID configuration was therefore removed in favor of 'linear drive merge' to accommodate all the of Argus archives into QNAP. Linear drive mode will combine all the hard drives of the system into one virtual hard drive which can be used to continuously store data without physical boundary between different hard disk drives. However, this mode does not offer any failure tolerance for any of the hard drives. In normal cases, it is highly advised against since it does not offer any protection by redundancy when a potential disk failure happens.

QNAP was geographically separated from ‘Mithril’ and was stored in Department of Electrical Engineering and Computer Science’s server room. Data was copied back to ‘Mithril’ from ‘Bulika’, ‘Lona’ and ‘Lilith’ after ‘Mithril’ was formatted of its configuration and OS was reinstalled which is described in the following section.

3.3.2 Hardware and Software Setup

The new processing servers, ‘Bulika’, ‘Lona’ and ‘Lilith’, were donated by Cisco each of which are capable of handling 40 parallel processing threads at once, hosts 384GB RAM, and has a total of 12TB SSD capacity.

GLORIAD had setup the 11 out of 12 drives in JBOD (Just a Bunch of Disks) configuration which bypassed the built-in hardware RAID controller. FreeBSD then merged them using redundancy scheme, the ZFS file system. After the data was recovered there was the option to completely change the configuration or keep the previous set up. This opportunity was used to change the drive setup from ZFS RAIDz2 (which is equivalent to standard RAID 5) to standard RAID 6 and format entire virtual drive the EXT4 and also to utilize the unused drive for data storage. At first the reason for not using the drive was assumed to be due to damage or failure but after applying new configuration to include this drive, it was found to be fully functional. More information about the previous drive configuration is shown in the Appendix B.

In this method two LSI MegaRaid SAS-9271-8i hardware RAID controllers that were present in each of these servers are used in the configuration shown in the Table 3.2.

Table 3.2 Disk Configuration

Controller Slot	Number of Drives	Usable Capacity	Number of actual usable drives	Purpose
1	4	2TB	2	OS
2	8	6TB	6	Data Storage

This new scheme converts 4 SSDs into parity drives which become unavailable for storage. But it offers the highest protection for the data by segregating the two hardware RAID controllers into OS and data storage separately and setting up all the drives in each controller in RAID 6 configuration which can tolerate up to 2 drive failures. The system can thus withstand 4 drive failures in total and still function properly. The second controller allocated for data storage was mounted at '/home' which transparently maps physically isolated disks and RAID controller into the OS seamlessly. Software and configuration is stored in the 'OS' set of drives (slot 1) and permanent storage is held at 'Data Storage' set of drives (slot 2). Furthermore, in the unlikely event of complete OS failure, simply replacing the drives related to the 'OS' RAID controller the system (slot 1) and installing the OS the system can be brought back up to full functionality since data storage is in a separate RAID controller which handles all the separate SSDs. This scheme also offers better software independence by making it easier to replicate the setup in wide variety of systems since it is transparent to the OS.

By utilizing the built-in RAID controllers the CPU and memory overhead to maintain ZFS redundancy and data protection is offloaded to the built-in RAID controller, freeing up processing capacity and memory for the function of the new InSight2.

Operating system was changed from FreeBSD to Ubuntu Server 14.04 Long-Term Support (LTS) which is based on Debian, a Linux distribution that is geared toward stability and performance. Ubuntu uses mainstream latest Linux kernel which integrates latest updates from Intel, the manufacturer of the Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz processor, which contains latest SSE instructions and stability improvements. The overall philosophy of Ubuntu is the stability so only the most stable version of the software is pushed to the end user systems during an upgrade. This is essential for systems that need to be kept up-to-date to make sure they have the latest security patches as they are facing the internet which can pose outside threats. Since upgrades are low-risk it encourages end user to upgrade the system without worrying about the system breaking by new versions of the software installed.

For the installation of GLORIAD many software had to be compiled increasing the complexity for the end user to install the system. All of InSight2's software is installed from Ubuntu repositories. They are also carefully curated in order to avoid conflicts with other software which makes upgrading them reliable and low risk. All of this software is readily available in the online Ubuntu repositories which makes it easy and reliable as well as secure to install them since they are managed by the maintainers of the OS distribution and are installed by downloading via built-in public key based encryption system mitigating the chance for man-in-the-middle (MITM) attacks during the installation procedure.

The following software environment was setup in '*Bulika'*, '*Lilith'* and '*Lona'*:

- Argus server and clients

  Argus server is used to generate network flow information while this information is displayed using Argus clients.

- Elasticsearch database and Kibana

  Elasticsearch is the full text search engine used to store and search enriched Argus data. It is highly capable of scaling up to the growing needs of large-scale network metadata storage. Elasticsearch is based on Apache Lucene and is known for its ability to handle large volumes of data in the scale of hundreds of millions of records per day. It is being used by a growing list of companies that handle "big data" such as Netflix, Facebook, Wikipedia, Atlassian and Github. It uses JavaScript Object Notation (JSON) formatted data documents, format that is known to be highly flexible that uses no scheme to define the records inside of a document which allows the arbitrary definition of documents and flexible typing of data contained within. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition. This format makes it easy for humans to read and write JSON encoded documents and it makes it trivial for machines to parse and generate them.

- Python modules

  python-elasticsearch: Python client for connecting to the Elasticsearch database and upload enriched records.

  python-mysqldb: Python client to connect to the MySQL database to extract the GSR into Elasticsearch database.

- Apache server, JavaScript, PHP

  Linux, Apache, MySQL, and PHP/Python/Perl are known as the LAMP stack. This LAMP stack is used for web-based software deployments. MySQL portion of this is disposed of in favor of Elasticsearch since it offers superior search engine and the rest of the Apache2, JavaScript and PHP is kept inclusive.

GLORIAD InSight was written using the Perl 5 programming language, which was superseded by Perl 6 in 2015. Perl stands for Practical Extraction and Reporting Language which was originally developed by Larry Wall in 1987 and was intended to be used as a general-purpose high-level scripting language to process reports. It has powerful text processing capabilities making it suitable for variety of applications ranging from system administration to network programming. Perl is a flexible programming language which comes with the slogan "There's more than one way to do it" (TMTOWTDI). For many cases this is an advantage since code can be written in variety of ways to achieve the same goal. Perl code can benefit from many of the third-party repositories that host many different libraries that extend the core functionality such as comprehensive Perl Archive Network (CPAN). However, when developing large-scale software platforms this makes it hard to maintain code, since there is no standard way of coding. Perl does not have native implementation of Object-Oriented Programming (OOP). However third-party libraries exist that offer OOP functionality if needed such as Perl Object Environment (POE) which was used by GLORIAD and is available through CPAN. POE achieves OOP functionality by having layered components on top of the ground layer POE::Kernel. Perl OOP is not a native implementation and introduces more moving parts and extra libraries to be installed to be functional in addition to the core language. Since the competition from other programming languages such as C, C++, Java, and Python for software development Perl programming language was mainly used in the niche usage of quickly writing scripts without considering long term readability or maintainability, to perform a few automation tasks regarding network or system administration, and not for coding multi-developer large-scale projects.

InSight2 is built from ground up using the Python programming language. First released in 1991, Python is a widely used language today, intended for general purpose programming at a high-level compared other languages such as C. A main focus of Python is to increase the readability of the code. It uses whitespace indentation instead of curly brackets or keywords and provides better syntax. Not only this allows code to be written with fever lines, it also comes with large amount of official libraries which supplements the core functionality. Compared to third party libraries and repositories, these modules are developed by the official maintainers of the Python programming language. They are also available from Linux OS official repositories making the installation streamlined and safe. Python is installed by default in all Linux distributions. The installation package that comes with InSight2 is greatly simplified by the use of Python libraries from the repositories. These are carefully curated in order to avoid conflict with other modules and installation is guaranteed to cover all the dependencies by OS package management system such as ‘apt’ found in Ubuntu.

Elasticsearch, Logstash and Kibana are referred to as the ELK stack. We use Elasticsearch and Kibana to store the data in the database, search necessary documents and visualization. Alongside the development of InSight2 Elasticsearch was upgraded from version 1.4 to 5.3 and Kibana from version 3 to 5.3 due to various issues associated with the old version. More details about the specific changes are discussed in depth in the Chapter 5.

4. DESCRIPTION AND ASSESSMENT OF GLORIAD INSIGHT

GLORIAD InSight architecture was a collection of Perl scripts that perform reading from either Argus archives or Radium server (a collection node for aggregating Argus data from multiple nodes), processing and inserting into Elasticsearch (ES) database to be visualized in a customized version Kibana webpage.

4.1 Software Architecture

It is important to understand the system architecture of GLORIAD InSight to improve upon that and make progress. Limited documentation was provided by the GLORIAD team about the purpose of each server such as whether they were used for hosting Elasticsearch, Kibana and other databases, what OS they run on, what Argus flow information are extracted from the flow records, and the general skeleton of an animal of the GLORIAD InSight platform. For in-depth understanding about each animal, the code base has been investigated to find out how each animal functions, what databases it accesses at what time, how intermediate data is stored, how the final results are posted and where, and how they communicate with each other.

GLORIAD InSight used FreeBSD on the ZFS RAIDz2 file system for the OS and its data storage purposes. While ZFS offers many advantages compared to traditional file systems such as being able to add new disks without breaking the disks array, perform compression, de-duplication, caching and quota management, it requires constant maintenance. During the investigation to find out the causes of failure of the file system it was found that ZFS has emailed the system administrator cautionary messages about the file system being used over the quota. ZFS requires significant amount of free space for its operations to perform as a scratch space and low disk space can lead to file system failure. InSight2 utilizes the standard RAID 6 redundancy enabled file system that allows the tolerance of failure of up to 2 drives. As opposed to the setup used by GLORIAD which involves bypassing the built-in hardware RAID controller to handing over the disk redundancy and error recovery two the software ZFS, the new server setup beginning with InSight2 involves using the built-in RAID controller to offload these functions to the hardware controller which frees up valuable system resources for data enrichment. While ZFS can transparently perform redundancy and error recovery while the file system is online disc write and read operations need to be performed through these extra file system layers. The standard implementation of RAID does not require expertise on ZFS file system to maintain it properly, and it is a truly set-up-and-forget-about-it solution.

Each function call was an event in the event-loop and was executed asynchronously with time-independence. In order for this event model to function properly the function calls needs to be non-blocking. For example, if a function gets blocked waiting on results from a database query then all the other events in the loop had to wait for this event to complete. Main database used by GLORIAD is stored in a MySQL database which is a sequential database. Since it only allows single read or write action at a time, database accesses every animal need to be optimized for database transactions to prevent bottlenecks. ‘Elephant’, the most significant animal of the farm needed to have access to this database more frequently than the others. It was invoked every 30 seconds and posted data into Elasticsearch database to be visualized. In order to make sure proper functionality of ‘Elephant’ it was paramount that ‘Main’ MySQL database was not blocked for read-write access. To achieve this, ‘Elephant’ kept a simpler and separate database in the form of a SQLite table called ‘ip_cache’. SQLite is a lightweight database which stores is database in a file. This makes it suitable as a secondary database for the use of ‘Elephant’ since updates to the database can be performed easily and without blocking other animals from accessing the ‘Main’ database.

This event model required a mechanism to pass messages between each script to be able to communicate with each other. It was achieved through ZeroMQ publisher subscriber style message passing queue. Animals were not restricted in the way they are allowed to access system resources. It enabled different animals to access any database, Argus archives, start new processing tasks and access memory at any given time. The architecture is outlined in the Figure 4.1.

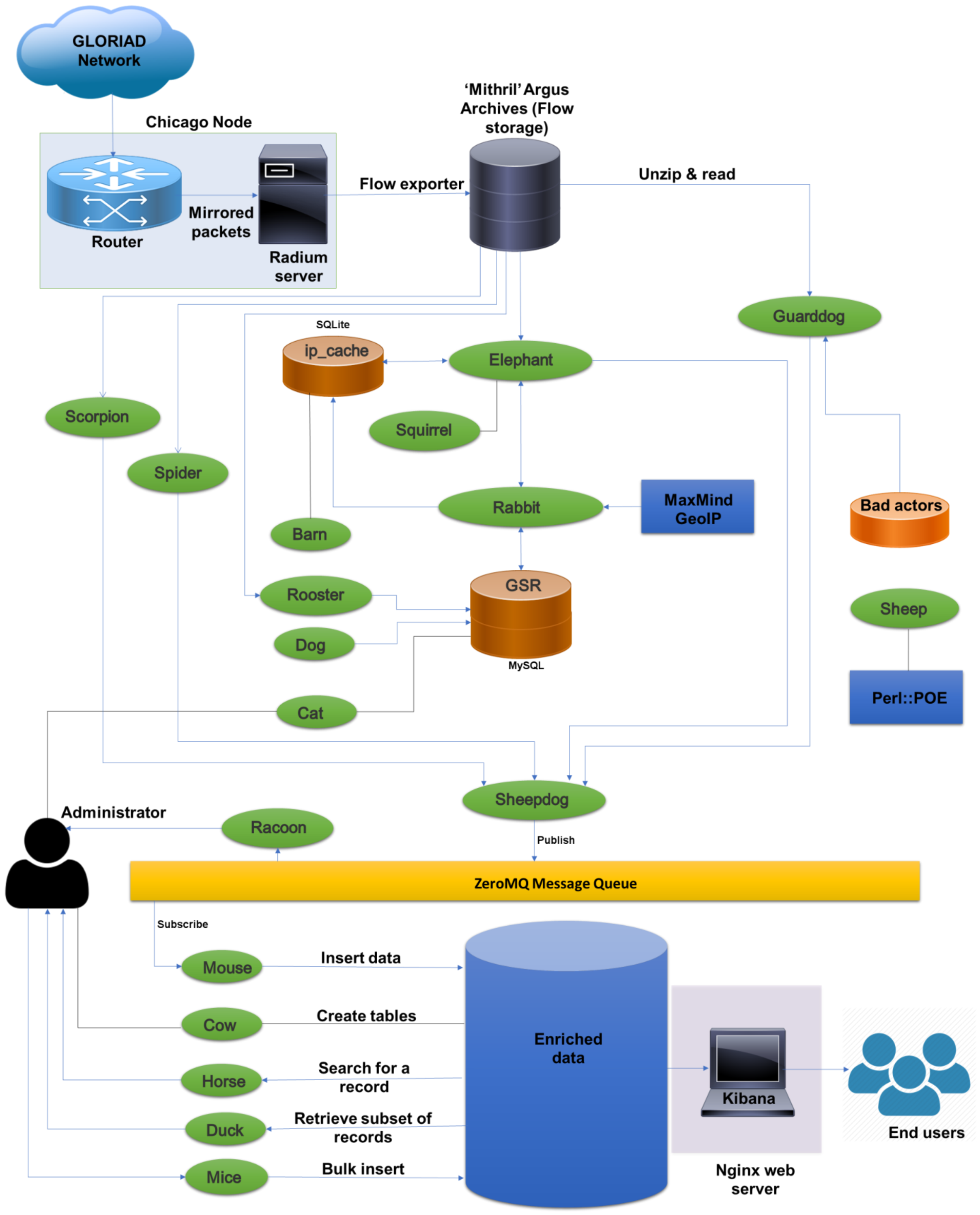


Figure 4.1 GLORIAD InSight System Architecture

Some animals may take longer time than others to process data, post the results and terminate. Because of this reason all animals need to be synchronized to pass and receive processed results correctly. Most animals are invoked using Linux system tool ‘crontab’ that runs jobs at certain programmed time intervals. ‘crontab’ is an automated tool that is used to perform system administration functions such as checking if a file content was changed, making sure certain file permissions are preserved, and periodical backups to a remote server. Timing has to be carefully benchmarked and determined beforehand due to the unpredictable nature of animals that occurs due to unpredictability of network performance such as spikes in the network utilization e.g. Denial of Service (DoS) attack.

Due to various unexpected events such as receiving unexpected data formatted out of scheme or programmer bugs scripts can exit before their tasks have been completed. Asynchronous event model requires separate daemon process which is capable of monitoring the said scripts from terminating prematurely. This program will keep track of the process numbers of the processes that it is programmed to monitor and in case such process exit for some reason it will re-invoke them and log it into its log file. These log files can be used to debug problems that might have caused the script to terminate. The asynchronous event model handles a number of moving parts. Continued functionality of the farm is guaranteed using monitoring tools, invoking scripts at the right time, and using message passing queues to enable communication between different independent animals.

The GLORIAD InSight has been written in Perl 5 programming language. The event-loop logic was taken care of Perl POE module [26] all animals run POE::IKC::Server. And all the animals are required to include this library in their code. Each IKC server runs sessions which is a collection of functions called as events or states. Generally each animal runs 3 sessions:

1. S_eat: This session reads input data and processes it.

2. S_clean: This session runs periodic events to do grooming jobs such as trimming the databases, cleaning up caches etc.

3. S_answer: This session has events to terminate the animal or keep it alive.

This posed limitations where behaviors of some animals require deviation from the standard sessions such as Elephant that has custom sessions to talk to other animals such as ‘Rabbit’ and ‘Mouse’.

The ‘Farm of Animals’ architecture allows users to add new animals to the system to get new functionality. For example to get the functionality to add new type of metric the end user or developer could develop a new animal that will publish the desired information to the pub-sub proxy, ‘Sheepdog’. Then by modifying the ‘Elephant’ that information can be included in the JSON document sent to the Elasticsearch database via ‘Mouse’. The intention was to allow of developers to write their own animals to extend the functionality of the system. The drawback of the system was that there were no guidelines or a structure that developers need to adhere to in order to ensure the proper maintenance and compatibility between animals, causing potential disarray if released as open source software.

Zero Message Queue is a communication bus queue and is used for passing messages between the animals. ZeroMQ is a free and open-source distributed messaging platform which allows multiple codes working on different aspects of a software system to communicate with each other. It supports message passing over inproc, IPC, TCP, TIPC, and multicast. Its core focus is asynchronicity. This is useful for any software platform that implements asynchronous architecture. It is being actively developed and available for many different programming languages. It is being used by companies such as AT&T, Cisco, Spotify, Samsung Electronics, and Microsoft. Unlike other message passing buses which only uses a broker system to delegate the messages as seen in architectures such as “hub-and-spoke” ZeroMQ Is capable of adopting different kinds of techniques for data transfer such as ‘broker’, ‘no broker’, ‘broker as a directory service’ and ‘distributed broker’. The messages are transmitted through the network and transmitter and receiver nodes are uniquely identified by the network socket address. GLORIAD InSight used the ‘broker’ functionality of ZeroMQ. Messages are passed into the network socket of the broker to be delivered to the intended recipient. Until the intended recipient polls the queue to check for new messages two messages are kept within the queue. In the case of InSight, processed and enriched data is pushed into this queue to be read by another animal which either uses that information to enrich more data or to upload into the Elasticsearch database.

ZeroMQ provides a reliable message brokering platform. It guarantees point-to-point reliability. The sender only needs to make sure that the message has been successfully passed into the broker. One disadvantage of this system is that it itself is a complex software and involves processing and memory overhead.

It is suitable for the architecture used by GLORIAD InSight since information is iterative enriched by different animals independently and asynchronously. Animals are complex in nature and differ by their function; some of them directly communicate with Elasticsearch to upload data while some passes data to another animal. Having this robust message passing platform makes it possible to unite the animals into one cohesive software platform.

Monit is an application used for monitoring processes. InSight used this to monitor each animal. If for some reason an animal gets killed Monit would re-invoke that particular animal. Monit would keep a log of the invoked animals which enabled easy debugging and investigation to find out why that animal died (no pun intended). While using a background service such as Monit eases the use of scripted animals, it is not an ideal solution, since the InSight software suite is not integrated tightly into the OS. This also introduces other software to be installed in the target system further contributing to the complexity as a whole.

4.2 Description of ‘Farm of Animals’

The Perl scripts were collectively known as the ‘Farm of Animals’. Animals are given names such as ‘Elephant’, ‘Mouse’ and ‘Guarddog’. However it should be noted that animal names in the real life does not necessarily reflect the function of the script. For example, ‘Elephant’ is responsible for filtering out traffic to select the large network flows and attach information ‘published’ by other animals such as ‘Scorpion’ another animal whose functionality is to detect network and port scanners within the network. These two animals in the real life do not reflect the functionality of the script itself. On the other hand, ‘Guarddog’ that publishes ‘Bad Actors’ found within the network into the publisher subscriber queue known as pub-sub queue, does provide semantic resonance to a real world ‘guard dog’ which watches out for threats. Publisher subscriber proxy, ‘Pub-sub proxy’ in short, is a message passing bus that enables different animals within the farm to communicate with each other. Publishers will publish data into the queue in the form of JSON format, such as ‘Scorpion’ which will publish the list of IP addresses that it detects as scanners, while the subscriber will collect this information from the queue and use for its own purpose, such as the ‘Elephant’ which will use the information published by the ‘Scorpion’ to tag those IP addresses before sending to an instance of ‘Sheepdog’ to be send to the Elasticsearch database using ‘Mouse’.

In its final version GLORIAD InSight was capable of incorporating different dashboards to show different aspects of the network ranging from network telemetry such as the number of bytes transmitted, number of packets transmitted, geographical information, institutional information such as organization name, discipline, and security information such as bad actors scanners and spammers. The user interface of GLORIAD InSight allows switching between different categories of dashboards that displays a range of information based on regions to operations. Each of these categories contains a suite of dashboards that offer relevant information. The dashboards are static, pre-configured collection of visualizations. When each dashboard is selected they are loaded allowing the user to visualize the network status during the given time period.

Each farm animal was designated one task, built on ‘asynchronous event based model’. Key animals are described next.

Elephant

This animal was the critical component of the farm. Due to the processing being performed using a single thread first *icmp, udp, ntp* traffic is filtered out and network flows that are less than 1500 Bytes are discarded. This made sure that all the incoming traffic is processed without packet drop. Argus flow information sent from these locations was collected at the Radium server in ‘Bluemac’. The architecture of the ‘Elephant’ is outlined in the Figure 4.2.

Elephant keeps copies of the tables it needs in a SQLite database named ‘ip_cache’. While this allows faster access compared to accessing the main MySQL database, it consumes more space since the data in the main database is duplicated. This also prevents the main table from locking up due to read access. This is a significant drawback of using sequential databases such as MySQL and SQLite, and asynchronous event model. In the asynchronous event model different animals are allowed to access the main MySQL database at any time, which can result in spikes in disk access incurring longer-than-usual wait times since sequential databases only allow one activity to be performed at the time. When the ‘Elephant’ process starts, it loads all the relevant data to local SQLite temporary tables and builds ‘ip_cache’ table as It reads all the IP addresses seen in Argus files in last 30s and attaches geo-location and domain info corresponding to each of those IP addresses into the ‘ip_cache’.

Records older than 2 hours are purged from the ‘ip_cache’ SQLite database. This ensures that this cached version of the main database is relatively up-to-date. However, if an IP address was updated in the main database within the last 2 hours it will not be reflected in ‘ip_cache’.

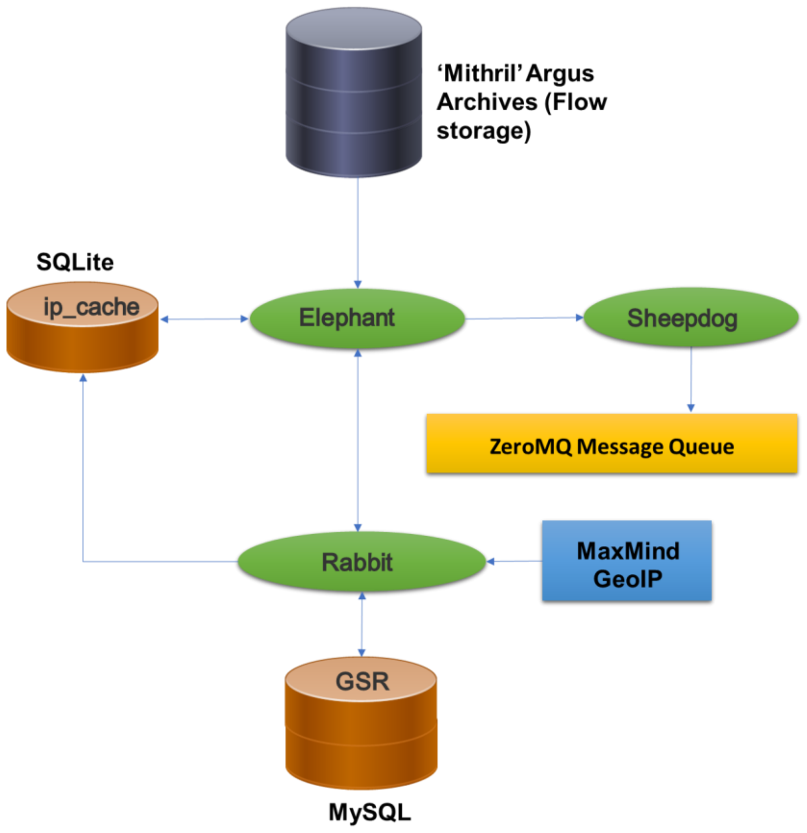


Figure 4.2 Architecture of the ‘Elephant’ animal

Core data structure of Elasticsearch is JSON documents, which are simple objects that contain keyword and value pairs in a nested format. Before a record can be uploaded to Elasticsearch database it is constructed locally until the record is complete. Elephant keeps these records inside this SQLite database. ‘Elephant’ reads from Argus archives every 30 seconds and all the flows in this 30 seconds are processed first by using the filters described above, then are tagged with the information sent by other animals and finally uploaded to Elasticsearch.

Mouse

Mouse accepts JSON encoded enriched documents and submits them to the Elasticsearch database. In this way, it abstracts animals from the details of interaction with Elasticsearch.

Rabbit

If ‘Elephant’ or any other animal in finds an IP which has no entry in the local ‘ip_cache’ cache table in SQLite database then it sends a message to ‘Rabbit’. ‘Rabbit’ tries to get the IP’s info such as city, country, latitude, longitude, Autonomous System (AS) numbers – ‘asnum’, GSR based institute information called ‘domain records’ and Domain Name Server (DNS) from the GSR tables in ‘Main’ MySQL database. If it is not in the database, it goes upstream and gets the info from Maxmind GeoIP database. It updates the ‘ip_cache’ database with the new information and sends a reply back to the animal which initiated the request.

Sheepdog

This is a ZeroMQ pub-sub proxy, which abstracts animals from details of implementing ZeroMQ related functions such as adding records to the queue or extracting from it to enable multiple clients to publish or subscribe to a ZeroMQ message queue. Animals such as ‘Guarddog’ and ‘Scorpion’ publishes IP labels to ‘Sheepdog’. ‘Elephant’ subscribes to those messages. Architecture of the ‘Scorpion is outlined in the Figure 4.3.

Guarddog

Guarddog is responsible for tagging network flows that belong to malicious IP addresses. It gathers these information from the 'bad actors' database. The Architecture of 'Guarddog' is outlined in the Figure 4.3.

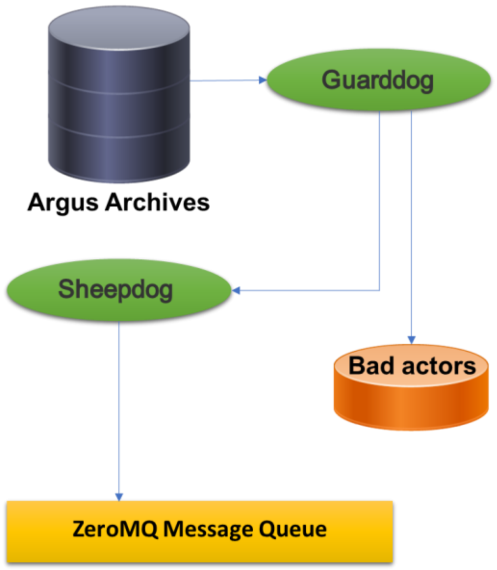


Figure 4.3 Architecture of the 'Guarddog' animal

Spider

According to Symantec Monthly report in October 2010 [27] percentage of spam messages sent via email in the world was 89.4% of all the messages sent in September and it was increased to 92.51% in August. This trend not only wastes the valuable bandwidth resource within the network that could otherwise be used for research and education purposes, but also poses risk to the recipients of these emails. Employees might open these messages thinking that they are from an official source only to be infected by malware. They can spread from one computer to another costing the organization time, money, and reputation. Research and education networks are not an exception to spam messages. During the development of spider several nodes were identified as spammers. And 'Spider' continue to identify more and more IP addresses that involve in spam activity. While it is not the practice nor the policy of GLORIAD to record or look at the contents of the spam messages they are very likely to be marketing materials where massive number of email addresses are targeted for spreading unsolicited advertisements, ‘phishing attacks’ where a large number of end users are tricked into doing something while pretending to be doing something else, and ‘spear phishing’ attacks where specific individual is targeted and messages are sent pretending to be from a bank or their employer tricking them to expose their credit card, Social Security Numbers (SSN) or other personally identifiable information (PII) in order to perform fraud. Spamming inside the network is a nuisance and a threat to its users and should be identified in order to take actions against these nodes such as blocking their access to the network or limiting their ability to send a large-scale email messages. Responsibility of ‘Spider’ stops here and later this information is visualized using the web interface. Figure 4.4 shows the data flow and architecture of ‘Spider’ animal.

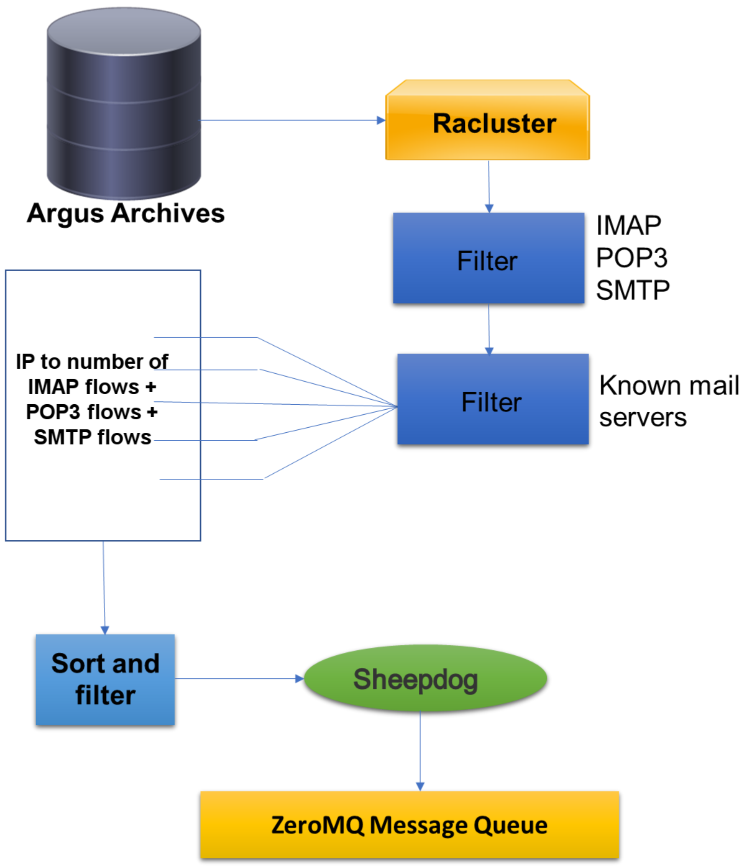


Figure 4.4 Architecture of the ‘Spider’ animal

Scorpion

Scanning is a network functionality which involves sending a series of packets in covert or in rapid succession in order to glean information about the target device. This information can vary from devices that are up and running, number of open ports, open port numbers, services running in these ports and their versions, of these devices.

Scanning is the first step of reconnaissance during a cyber-attack. This gives attackers enough information to find a suitable exploit to use against target service. For example, if a node is running a website, which is evidenced by the open port 80 to the outside world, that user should be running some version of a web server. First step is to identify which device is up in the network which is known as ‘ping sweep’ which sends a series of packets to a subnet of the network and see which devices reply. Unless the end user has taken measures to prevent replies to these probe messages the device will let its presence be known by sending a reply to the sender of this probe. Scanning allows to identify that the user is in fact have left the port 80 open which gives away that there is some service running in this port and it is accessible to the outside world. This is known as ‘port scanning’. Furthermore, using a ‘service scan’, which sends a series of predetermined and crafted TCP packets in a certain sequence and observing the sequence of the replies version of the web server can be guessed fairly accurately. By looking-up this information in vulnerability postings that are publicly available, such as Common Vulnerabilities and Exposures (CVE), attacker is capable of finding an attack that this specific version of web server is vulnerable to. This process of reconnaissance is generally identified as ‘network scanning’ and ‘port scanning’.

Identifying these misbehaving nodes early in their process of launching an attack gives a window of opportunity to act before the actual attack happens. ‘Scorpion’ reads Argus archives in order to detect which nodes perform scanning activity within the network. This is a time critical problem to detect scanners so scorpion is invoked every five minutes. Timing is essential since all animals in the farm are asynchronous in behavior and it is important to publish this information into the ZeroMQ message bus just before the invocation of the ‘Elephant’. When scanning IP addresses are identified ‘Scorpion’ will publish this information so that when elephant looks up in the queue it will extract this information using ‘Sheepdog’. IP addresses seen in the stream of flow information read within the last 5 minutes that matches the scanner IP addresses are tagged with the ‘scanner’ identifier. Then they are uploaded to Elasticsearch. Responsibility of ‘Scorpion’ stops here and later this information is visualized using the web interface. Figure 4.5 shows the data flow and architecture of ‘Scorpion’ animal.

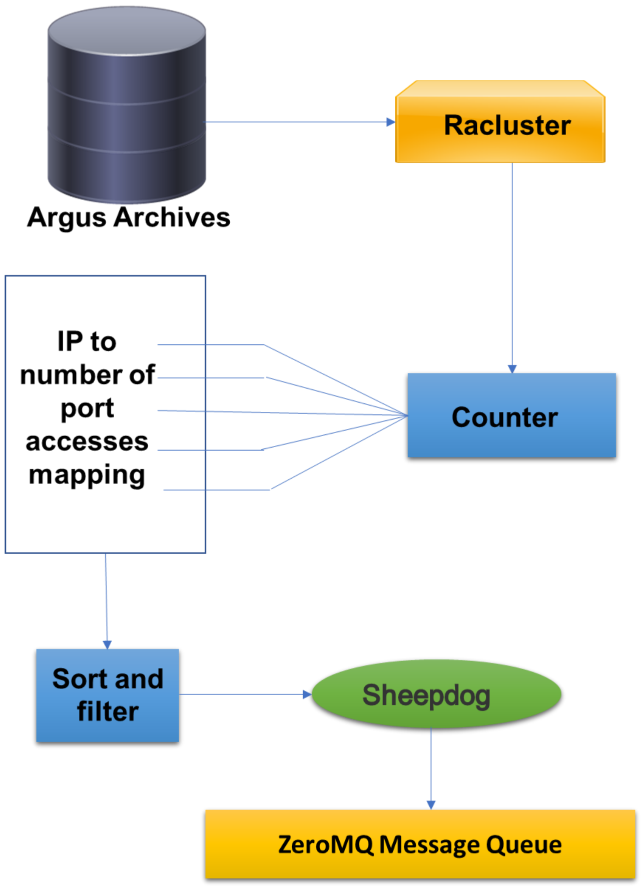


Figure 4.5 Architecture of the ‘Scorpion’ animal

Other animals

General functionality of these other animals was learned from their respective code and are categorized below. They perform minor tasks on behalf of the main animals.

Elasticsearch related

Cow	– Create new tables in Elasticsearch database
Horse	– Search for a given record in Elasticsearch database
Mice	– Bulk create Elasticsearch documents of an arbitrary length
Turkey	– Extract and hold records from Elasticsearch to be visualized
Duck	– Extract a given subset of data from database

MySQL, SQLite and database caches related

Dog	– Update information in the MySQL GSR table
Rooster	– Create temporary records of daily flow information
Cat	– Perform database and cleanup functions.
Barn	– Create, update and clean caches

Message passing related

Raccoon	– Dump data from a ZeroMQ message queue for debugging
Sheep	– Create Perl::POE session kernels

Administrative functions

Farmer	– Keep track of the farm of animals.
Squirrel	– Create more ‘Elephant’ sessions if needed

4.3 Global Science Registry

GLORIAD InSight uses sequential databases in addition to Elasticsearch in order to keep GSR and other flow related information such as services, information about equipment, VLANs, AS numbers etc. MySQL and SQLite databases are used to store permanent data and temporary data respectively. SQLite Database is primarily used by ‘Elephant’ to keep a cached copy of the information readily accessible that is separate from the main database using MySQL. Information is trickled down from the main MySQL database to the SQLite database as needed basis.

‘Main’ MySQL Database

The GLORIAD InSight ‘Main database’ is stored in a MySQL database which is a form of sequential database that allows data to be kept in a pre-configured manner. This format is in the form of tables and records, and each of these tables can contain either data or a foreign key that links a record to another record in another table. This pre-configured format is called the ‘database schema’. Insertions into the database should adhere to this schema and if any data differs from this schema it cannot be inserted without change to the schema which requires creation of new table and copying back of the old data along with the new fields added as null values or zeros. While the use of sequential databases is suited for data that does not change its fields over time, they are not suited for data that are dynamic in nature such as information about IP addresses; geo location coordinates country, city and zip code. This ‘Main database’ consists of information ranging from disciplines, organization class, application name, and region information. They are distributed across 29 tables, of which 12 tables contains the information relevant for data enrichment, such as ‘classes’, ‘domains’, ‘ipsdns’, ‘ip_unassigned’, ‘ccodes’, ‘disciplines’, ‘asnums’, ‘iplabels’, ‘govagencies’, ‘apps’, ‘ips’, and ‘ipstext’ which are explained below after the description of the SQLite temporary database.

‘ip_cache’ SQLite Database

Unlike MySQL which requires a server to hold the database and execute query functions, SQLite is self-contained, server-less, transactional sequential query database language engine. This makes SQLite very lightweight and easy to deploy for requirements that focus on portability. While MySQL can be disk input output (I/O) intensive SQLite is focused on keeping its data mainly in memory. the larger the database gets the more memory it needs to function efficiently. It is cross platform and open source. It has database access clients written in almost any programming language.

‘Elephant’ keeps a local SQLite database with copies of tables it needs from ‘Main’ MySQL database. This is to speed up the data reading and also to avoid table locks on the main database. When ‘Elephant’ process starts it loads all the relevant data into local SQLite table building up the ‘ip_cache’. It reads all the IP addresses seen in Argus files in last 30 seconds and loads geo-location and domain info corresponding to each of the IPs into the cache.

During the S_clean session it expires all IP addresses that haven’t been seen in last 2 hours from the cache.

Tables in the ‘Main’ MySQL Database

MySQL database dump was uploaded to a temporary MySQL instance to explore the contents in order to reconstruct the GSR by de-duplicating entries and converting the sequential tables into JSON documents for faster searching using Elasticsearch database.

Domains: ‘Domains’ table contains organization class, world class, government ID, discipline, country, city region, postal code, and latitude and longitude information. This table is an aggregation of ‘ipsdns’, ‘govagencies’, ‘Disciplines’, and ‘ips’ tables. ‘Elephant’ in GLORIAD InSight makes a copy of this table every time it is invoked into a SQLite database for faster access. This table is also the table that has largest number of fields, as well as the highest amount of information. Each time ‘Elephant’ encounters new IP address for which there is no information in the ‘domains’ table it will invoke a new instance of ‘Rabbit’ to fetch geo location related information, which ‘Rabbit’ will insert into the ‘Main database’. Subsequent searches for the same IP address will be taken directly from the SQLite database instead of consulting Maxmind GeoIP database. So over the time this database will expand to include all the IP addresses seen in the particular network.

Classes: IP addresses in the GLORIAD network are categorized into one of several classes. ‘Classes’ table contains information paired between Class ID and Class Name. Here Class ID accesses a foreign key that links record from other tables to the ‘Classes’ table. It also contains application ID, country code, direction and network information as well. Flow direction information is also present in flow records which essentially duplicate that information. Information such as country code network router and application ID information should be in their respective tables so that the most significant table ‘Domains’ table contains foreign keys which can be used to retrieve textual information. This allows having flatter architecture of the tables with less depth instead of hierarchical architecture of tables which result in higher complexity as well as information redundancy and longer-than-necessary look up times. InSight2 as described in the later chapters keeps all information in a separate index in the Elasticsearch database utilizing a simplified structure which results in faster access as well as ease of maintenance.

IPsDNS: This table is used to map key ID to IP address and DNS server. Other tables can use the key ID field to map records to records in this table. The DNS information stored in this table is used to determine which domain name lookup resulted in the information transaction. Create time and modify time information are also stored within the table which are redundant information since these records does not need to contain create time and modify time as they are static information.

Disciplines: This table uses ‘discid’ to map discipline ID into the discipline and name. However, it was noted that ’domains’ table duplicates this information in its table rendering this table redundant.

IPlabels: Key ID is used as the unique key for the records in this table which contains information such as IP address, IP name, and other labels. In practice this table is used to match IP address to the IP name. This table also contains create time and modify time fields which are redundant information.

GovAgencies: This table contains foreign key from ‘domains’ table, ‘govid’ and matches it with the agency name and country code. This table also contains ‘modify time’ field which is redundant information.

Apps: By looking at the port numbers, the application each flow belongs to can be determined. But these applications to port number matching have to be stored first. This table contains this mapping. ‘appid’ is matched with application name, service ID, and application category.

IPs: Records in this table give mapping for key ID, domain ID, autonomous system numbers and country codes. It also contains IP address in string format as well as hexadecimal encoding. Even though hex encoding can improve space efficiency keeping both string version and hex encoded version is an inefficient use of space.

IPsText: ‘keyid’ Field is matched with IP name, region code, city, postal code, latitude, longitude, ISP, organization name, country code, IP address, domain ID, autonomous system numbers (‘asnum’), source and destination traffic in Bytes, minimum traffic and maximum traffic by month.

4.4 Assessment

4.4.1 Software Architecture

Originally intended for sequential programming for text processing applications, Perl is probably not the best choice for a system such as GLORIAD InSight. Other programming languages are better suited for software development and maintenance. Low-level languages such as C++, and high-level languages such as Python and Java are better suited for large-scale software projects. They have native implementation for the creation of objects and their use. In order to use object orientation features in Perl, the third-party Perl::POE package must first be installed along with the community driven repository known as Comprehensive Perl Archive Network (CPAN). This increases system overhead with regard to disk usage as well as memory to maintain and update the additional software repositories apart from the system managed software repositories such as ‘apt’. Furthermore dependency requirements have the potential to change over time causing incompatibilities with previous versions, which in fact was the cause of the degradation of GLORIAD InSight. In these cases one option is to hold back onto older versions to maintain compatibility with the old code or upgrade the code to suit for the changes of the new modules. The former option is not recommended since overtime lagging behind too many software versions can increase the vulnerability of the whole system for cyber threats and these previous versions may contain bugs that are fixed in the newer versions which would not be applied if they are not updated. The other approach is to keep up with the changes of these modules and update the code appropriately. However these third-party maintained codes are more likely to be abandoned by their developers due to change of interest or management, compared to software modules maintained by the original development team of the programming language itself.

Perl Foundation released the new version of their language, Perl 6, which is a significant update to the older version, Perl 5. Perl 6 has been in the development for 29 years. It introduces a long list of changes that are aimed towards better maintenance of code, from the inclusion of static typing of variables to core changes to its syntax. Compared to Perl 5 where subroutines had to be defined without formal parameter list and calling the subroutine with arguments was using a list of elements. This deviation from most other programming languages required the programmer to change the way of thinking specific to Perl 5 and in Perl 6 this has been changed to formal parameters similar to other mainstream programming languages. Among the many core changes the ability to natively implement object orientation as well as multi-threading proves Perl 6’s suitability for large-scale software development over Perl 5. However these changes make the code written in Perl 5 incompatible with Perl 6 migrating to the newer programming language requiring a complete rewrite of the old code.

Perl 6 has been released as a specification instead of an implementation. Consequently, there are more than one implementation of the same language. As of this writing none of the Linux distributions ship with any Perl 6 implementation. The recommended implementation by the Perl 6 developers is ‘Rakudo Star’ which comes with support for the MoarVM backend. The latest version adheres to the implementation guidelines of Perl v6.c. While there are stable releases it is still in the development which can cause instabilities for code written in Perl 6.

Versions of software used by GLORIAD InSight were outdated by 2 years. During this course of time support for the older versions of the software had ended causing them to just display a message asking to update to the newer version. But the compatibility of the enriched data using the older version was removed in these new versions. Elasticsearch moved from JSON based scheme for uploading and accessing data stored within the database into a RESTful

Application Programming Interface (API). RESTful API is based on representational state transfer technology. This is a standard API for web related application development. It is also known as the language of the internet. RESTful API makes use of the HTTP methodologies Defined in the RFC 2616 protocol. It can abstract the functions of a web service into just 4 commands:

• GET	-	Retrieve a resource
• PUT	-	Update a resource or change its state
• POST	-	Create a resource
• DELETE	-	Delete a resource

This minimalistic protocol of communication is simple and effective as any function related to web can be fulfilled using this API. Elasticsearch's decision to migrate into this new standard protocol is beneficial to achieve uniformity across many different technologies since it is current and widely used. But it is not backwards compatible with JSON encoded documents. This not only broke the compatibility of communication with Elasticsearch but also rendered the dashboards created in the earlier version Incompatible. So the data that is already enriched cannot be easily imported without re-enrichment. As a workaround Elasticsearch introduced an experimental library that allows re-indexing databases that are already indexed using old versions into the new version. However this library was met with issues since not all the newer functionality could be replicated during indexing.

Apart from these core changes Java version from 1.7 was updated to 1.8 as a requirement for the latest Elasticsearch, version 5.3 Elastic suite of tools consisting of Elasticsearch and Kibana was used which was the latest at the time of development, and Nginx web server was replaced by Apache2 web server.

Requiring the use of other software such as script status monitoring software, such as Monit, for the proper functionality of the suite of Perl scripts incurs additional overhead with regard to processing power, memory, and disk space usage for logging purposes. Using the Linux system's built-in tool, 'crontab' to invoke scripts periodically is not a feature of a large-scale software suite. As a better alternative each animal maybe daemonized and run as a system service. Linux system services are the most robust way to run programs in the background. Systemd is a low level system and service manager which is compatible with SysV and LSB scripts that replaces older sysVinit, and is found in most modern Linux distributions today. It allows native parallelization capabilities, and uses standard socket and D-Bus activation to start services, which allows the user to start, restart, stop, and probe status using a single command, ‘systemctl’. Systemd Implements transactional dependency-based service control logic where are the processes are tracked using Linux ‘cgroups’. This is a powerful method to ensure each demon is running as intended and provides a standard way of investigating the status of each. Compared to other system-level demonizing technologies such as ‘Upstart’ found in the older versions of Ubuntu ‘Systemd’ has become the standard for all mainstream Linux variants including Red Hat Linux, Debian Linux , openSUSE, and Ubuntu and its derivatives which increase is the ease of deployment due to uniformity. This helps lower the complexity of the implementation by reusing tools readily available in the system that are intended to be used by user software. This also negates the need to log independently from system logs since ‘Systemd’ logs to the central logging system which reduces the disk usage. This would have increased the throughput of the hardware used by GLORIAD for InSight which mainly uses hard disk drives which utilize spinning disks and is I/O intensive in nature. Spinning hard drives are slower than solid state disks (SSD) and are better suited for sequential reading and writing. Constant logging environment increase is random write calls into the hardware platters which causes the movement of the platter head out of sequence causing ‘seek delay’ diminishing the overall performance of the storage system. Since InSight is heavily dependent on disk activity to store and search enriched Argus records, it is important to reduce the usage of the disks by other non-essential programs.

4.4.2 ‘Farm of Animals’

The asynchronous nature of the farm requires timing constraints when animals are passing messages using the ZeroMQ message queue. This poses several challenges when the software platform grows in scale especially since all the animals are run using a single thread of the CPU hence the unpredictability of the time each animal takes to ingest Argus archives, digest the results, and output the results back into ZeroMQ. If the window of opportunity to push the data into the next animal is missed this data would not be enriched until the next cycle of that particular animal. ZeroMQ was only used to pass data and not control signals. So even though animals are invoked and given the freedom to operate asynchronously they are not truly event-driven. If control signals were sent at the end of each animal then the next animal can be started as soon as the previous one ends, eliminating the chance of out of sync message passes.

Another disadvantage of asynchronous event model is that there's no regulation or outlined structure that animals needs to adhere to in order to use the system resources. This causes system instability and unpredictability. Argus archives are stored on the disk using ‘gzip’ compression method and every time and Argus client needs to access these archives they need to be decompressed into memory. When multiple animals access these archives simultaneously and independently the same archives may be decompressed by each of them. For example ‘Elephant’ is most likely the first animal to read a certain archive since it pulls Argus archives every 30 seconds followed by other animals such as ‘Spider’' and ‘Scorpion’ since the decompressed archives are only kept in memory until the animal that decompress the archive uses the data, after which the data is discarded. This incurs the system unnecessary disk read actions to read the archives, processing overhead for the decompression, increased memory consumption to temporarily store the decompressed archives repeatedly. For a system that requires all its power to enrich the network flow records as well as for other software such as Elasticsearch database to index and sort records, and web server to present InSight to each request by the users, needs a robust mechanism to distribute and allocate system resources efficiently without causing frequent bottlenecks in performance.

4.4.3 Global Science Registry (GSR)

GLORIAD kept GSR in a MySQL database, consisting of 29 tables. MySQL is a sequential database which requires ‘schema’ to be defined before data can be stored. Elasticsearch on the other hand is a NoSQL implementation or a non-sequential database which can contain data belonging to arbitrary structure. This is especially important to store data which is constantly evolving. For example currently the domain table consists of information about geo location, organization and government information. If at some point of time new information is to be added such as security information a new field cannot be created in one of these tables without changing the database schema hence the cluttered and inefficient structure of the ‘Main’ MySQL database of GLORIAD. It is almost impossible to predict the exact schema for the database and expect it not to change over time. This poses the challenge of either knowing the structure of the data beforehand or migrate all the data into a new table, both options are not feasible. A third option is to create new tables when new information is added which destroys the clean and well-designed structure of the database.

Elasticsearch by definition contains JSON encoded documents which can have arbitrary structure. It has built-in mechanisms to detect what type of data is being sent into the database and update the data types. Data types defined in Elasticsearch are loosely typed. This allows it to have dynamic mapping. Furthermore it allows the user to define their own data types known as an ‘Index Template’. Index template can either accept new fields and create new data types on-the-fly or completely reject documents that deviate from the original index template. Index template of InSight 2 is defined using partial strict mapping of the data types which will ignore the malformed data fields and accept the rest of the document in case corrupt data is transmitted along with good data. This produces balanced and maintainable index structure which is suitable for production environment.

This further simplifies the overall architecture by reducing the number of secondary databases needed such as MySQL and SQLite. It also helps to reduce the number of steps the installation program has to perform in order to install InSight 2 in a client system.

Another advantage of using the same Elasticsearch database is that every transaction is non-blocking. Blocking databases prevent further access or insert or delete into the database until the original transaction is complete. This impacts the performance of the system since the farm of animals are created to be asynchronous. Asynchronous event model has no protocol for accessing the database and all the animals are allowed to perform any operation into any of these sequential databases at any time. This increases the wait time and causes system instability since certain animals which have to wait until I/O lockup is released for the database access. Event based model further increases the unpredictability of the performance of the system since it makes it hard to model the wait time for database access at a given time.

4.4.4 Hardware Capabilities

GLORIAD used 6 servers for data visualization. ‘Bluemac’ is the main server at “insight.gloriad.org” that is responsible for hosting the web server, ‘nginx’. ‘nginx’ is a lightweight web server Which instead of relying on threads uses an event-driven architecture. This allows small scale web pages such as InSight to be served with a small memory footprint. All the other servers are used to store the Elasticsearch database.

GLORIAD Elasticsearch cluster had 11 nodes distributed across 6 physical servers. Following is the list of servers used by GLORIAD InSight:

• Bluemac	–	1 Elasticsearch data-less node
• Anodos	–	1 Elasticsearch node
• Wingfold	–	4 Elasticsearch nodes
• Goblin	–	2 Elasticsearch nodes
• Princess	–	2 Elasticsearch nodes
• Curdie	–	1 Elasticsearch S node

Out of these highest performing servers ‘Bulika’, ‘Lona’ and ‘Lilith’ are used for InSight2. Others are decommissioned or shutdown.

All the Elasticsearch nodes have the same configuration directory structure to make maintenance and updates easy. All the Elasticsearch related files were in “/data/es”. Network metadata was collected in a central server located at Chicago running an instance of Radium server. Argus nodes were located in ‘nprobe-ord’, ‘nprobe-sea’ and ‘argus-ord’. GLORIAD maintained 15 servers at the time the project ended. A complete list of servers and their functionality used by GLORIAD is provided in table 4.1

Table 4.1 GLORIAD servers and their functionality

Server Name	Services	OS	RAM	Disk Space	CPU(s)	ZFS
Bluemac	Mysql Workers, Farm, ES, Kibana (InSight)	FreeBSD 10	192 GB	10x1T(7200; SAS 5Gbps);	Intel Xenon R 8 core, 16 thread x 4	2 x mirror zroot; 8 x raidz2 (~6T) zdata
Anodos	Mysql, Gearman, ES(2 nodes)	FreeBSD 10.0	262 GB	12 x 2 T (7200 rpm; SAS 6Gbps)	Intel Xenon 8 core, 16 thread x 2	zdata raidz2 10 ~14T; zroot mirror ~1.8T
Wingfold	ES(4 nodes), Kibana(InSight)	FreeBSD 10.0	764 GB	12 x 1 T (7200rpm;SATA 6Gbps)	Intel Xenon 8 core, 16 thread x 4	2 x mirror zroot; 10 x raidz2 (~8T) zdata
Princess	ES (2)	FreeBSD 9.2	192 GB	16 x 1T (7200rpm; SATA 3Gbps)	Intel Xenon 6 core, 12 thread x 2	2 x raidz3 7 each (~12.5T)+ mirror (~1T)
Goblin	ES (2)	FreeBSD 9.2	192 GB	16 x 1T (7200rpm; SATA 3Gbps)	Intel Xenon 6 core, 12 thread x 2	2 x raidz3 7 each (~12.5T)+ mirror (~1T)
Curdie	ES (1)	FreeBSD 9.2	64 GB	16 x 1T (7200rpm; SATA 3Gbps)	Intel Xenon 6 core, 12 thread x 1	2 x raidz2 6 each (~11T) + mirror (~1T)
Mithril	Backup server	FreeBSD 9.2	64 GB	SAS 3TB x 12	Intel Xenon 8 core, 16 thread x 2	raidz3 11 disks (~24T)
Bulika	n/a	FreeBSD 10	384 GB	11TB	Intel Xenon E5 core, 20 thread x 2	raidz3 11 disks 8TB
Lona	n/a	FreeBSD 10	384 GB	11TB	Intel Xenon E5 core, 20 thread x 2	raidz3 11 disks 8TB
Lilith	n/a	FreeBSD 10	384 GB	11TB	Intel Xenon E5 core, 20 thread x 2	raidz3 11 disks 8TB

Table 4.1. Continued.

Server Name	Services	OS	RAM	Disk Space	CPU(s)	ZFS
nprobe-ord	Argus probe	Cisco	8 GB	n/a	n/a	n/a
nprobe-sea	Argus probe	Cisco	12 GB	n/a	n/a	n/a
argus-ord	Future argus probe reading from taps	FreeBSD 9.1	n/a	n/a	n/a	n/a
Shadowfax	n/a	FreeBSD	65 GB	4 x 2T	Intel Xenon 6 core, 12 thread x 2	n/a
Qnap-ord	NAS Remote backup sever	n/a	n/a	n/a	n/a	n/a

4.4.5 Recommendations

GLORIAD had 15 servers at their disposal, but only 6 of the older servers were setup to handle the InSight operation. Three new high performance servers that were donated by Cisco, ‘Bulika’, ‘Lilith’ and ‘Lona’ were not utilized to handle the farm, which is the most resource-intensive software component of GLORIAD. At the time development of InSight2 started ‘Bulika’ and ‘Lona’ both were running the older 10.1 version of FreeBSD OS and ‘Lilith’ was running CentOS which was installed only for testing purposes. Using older hardware for the data enrichment process, run the Elasticsearch database and to host the web interface of InSight comes with a performance penalty due to their limited processing, memory, and disk storage capacity. Elasticsearch was not installed in these servers to enable shard allocation. It is recommended that the servers are used instead of the old servers to increase the throughput of the system. During the development of InSight2 the servers were set up with the latest version of Ubuntu OS, version 16.04 and latest Elasticsearch 5.3 database was setup for distributed database functionality.

InSight2 web interface is a direct modification of Kibana. Since Kibana directly communicates with the Elasticsearch database to retrieve the records it opens up a possible vulnerability since Kibana is exposed to the outside world. Even though the web interface is proxied through ‘Nginx’ web server, techniques such as SQL and JavaScript injection maybe be used to gain access to the database by unauthorized parties. Due to the lack of protection to the dashboards anyone is allowed to create, modify and delete dashboards. General users should not be able to change the visualizations and dashboards which can result in unintended users deleting dashboards and legitimate users accidentally modifying them permanently. It is recommended that dashboards are made read-only so that they are not prone to vandalism easily. InSight2 dashboards are read only and when the browser refreshed they will revert to the original state.

GLORIAD InSight does not offer any user authentication. Users should be authenticated in order to limit visibility of the sensitive information about the network nodes to authorized users. Since most clients that use the GLORIAD network are directly related to research and education fields their IP addresses should not be exposed to the public internet. GLORIAD have taken measures to anonymize the IP addresses. This complete anonymization removes the ability to know the IP addresses even by network administrators who might need that information for debugging purposes. Instead of taking the all-or-nothing approach it is recommended that an authentication system put in place to offer different levels of visibility into the InSight platform to ensure all users’ diverse needs are met, for example keeping a database of users with different privileges and only show IP addresses to the users who have higher privileges. InSight2 achieves this feat by incorporating PHP enabled user authentication system that is separate from OS user database which will load different dashboards depending on the user logged in.

It is recommended to use separate custom web interface which will extract the dashboards from Kibana and display them in a separate HTML or PHP enabled web page which enable the use of SSL certificates to encrypt the connection between the web server and the client browser to prevent eavesdropping and man-in-the-middle attacks. GLORIAD InSight uses a modified version of the Kibana to host the web interface by adding JavaScript enabled buttons to switch between dashboards. This was only supported in the early version of Kibana, specifically version 3.0 and later. Changes introduced to the Elastic suite of tools broke the support for this HTML based web page as it moved to dynamic web pages generated on-the-fly by the Kibana application. This rendered the web interface of InSight to be obsolete. Using a custom web interface that will pull the dashboards from the Kibana application not only enables resiliency against changes to Kibana but also allows ways to incorporate logos and descriptions for each visualization to provide user guidance built into the dashboards.

‘Elephant’ animal filters out ICMP, UDP and NTP traffic as well as flows under 1500 Bytes. This amounts to approximately 86% of the total traffic being filtered out. This shows that majority of the data consist of small flows. Since the scope of the GLORIAD InSight was to visualize the raw data, selecting only the large network flows seemed to be a feasible solution. However these flows play a vital role in network diagnostics and security analytics and omitting those leads to an incomplete and inaccurate representation of the network.

The final version of the Kibana visualization software used by GLORIAD InSight was version 3 which lagged behind the latest release, namely, version 5.3. Updates to the application including new dashboards, intuitive visualizations and modern dashboards were not incorporated into GLORIAD InSight such as dynamic filtering that applies filters across all the dashboard components and heat map visualization which allows to show flows belonging to IP address in an exact location instead of region based or state based for the map of the U.S. It is recommended to keep up with the updates of the software used in order to obtain bug fixes, new features and security updates.

QNAP was used to hold backups of the important Argus archives and GSR database. QNAP was an expensive all-in-one solution that was intended for large-scale multimedia applications such as storage of movies in a central location which can be streamed over the network to be played by multiple clients. Since the server is used for the data backup purposes only, it would have been cost-efficient to use an off-the-shelf ARM computer with large capacity to hold multiple 3.5 inch hard disk drives. ARM computers offer low power computation. Since this server is used just for holding the backups it makes more sense to use an energy efficient architecture such as ARM. Since both systems use Linux variants to achieve the same functionality, using generic Linux variant on off-the-shelf hardware makes it easy to maintain and is cost-effective. It is also advised that backup devices are physically located in a substantially geographically-separated location to ensure maximum data protection from to natural disasters. After the Argus archives and other databases have been recovered they were stored in this QNAP. QNAP is now stored offline as an additional contingency backup.

The farm architecture does not impose regulations for the development of new animals by third parties. By allowing end users to create arbitrarily functioning animals the system can get further complicated and diminish in maintainability. This can be fixed by defining guidelines and best practices governing the development of new animals that will ensure that different developers do not change the core animals in a way that one version is too customized to the needs of the particular user such that it becomes no longer compatible with other versions developed by other developers. By defining and restricting the changes to the core animals of the system it is possible to ensure core functionality stays the same across many versions of the platform and different animals, which is the nature of the enrichment module found in the InSight2 architecture which supersedes the farm architecture. Taking a step further an application programming interface (API) can be defined instead of relying on the farm architecture to extend the functionality. An API will allow developers a better structured method to write new code to the platform.

The Argus features extracted by GLORIAD InSight contain a number of redundant features. Some of these fields can be simply calculated dynamically using Elasticsearch's scripted fields using 'painless' programming language without requiring space to store them such as 'pkts': sum of source and destination number of packets, 'byes': sum of source and destination bytes, 'appbytes': sum of source and destination application related bytes, 'ploss': sum of source and destination packet loss, 'load': sum of source and destination load in bits per second. They can be dynamically calculated using by summing of the source and destination components for e.g. pkts = spkts + dpkts. Some features are redundant and does not offer any value or reason to be stored. There are 2 such features, the first one being 'ltime' which is the record ending time that has no meaning in the context of visualization of flow records and not used to display any information in any dashboard. The other feature is 'srcid' which is the source ID of the Argus which is an internal parameter used by Argus for own management purposes and does not reflect any real world parameter.

One of the core goals of making software publicly available in an open-source manner is to allow the end users to make changes and improvements so that they can be incorporated into the main development branch using software collaboration tools such as GitHub. It is advised that some sort of application programming interface (API) is defined in order to allow third-party developers extend the functionality of InSight. This will ensure that extensions are modular and can be installed as plugins instead of changing the core system which can lead to platform fragmentation.

While using system scheduling tools such as ‘crontab’ to invoke animals during specific times of the day such as every 5 minutes or at midnight everyday can ease the development of the system in the short-term, it comes with high maintenance penalty in the long-term. It is suggested that use of built-in system functions such as using SysVinit found in Debian Linux systems and convert the farm into a system level service for tighter integration into OS.

5. DEVELOPMENT OF INSIGHT2

5.1 Software Architecture

InSight 2 is developed with simplicity, robustness and efficiency in mind. Servers are formatted and set up in standard RAID 6 configuration that is easy to maintain and replicate. Mainstream Ubuntu Linux OS was adopted in favor of FreeBSD OS to offer better support in terms of updates, stability and software repositories. New system architecture has been developed to eliminate many of the moving parts to make it compact and streamlined using the synchronous processing model which pipelines the entire enrichment process using parallelized multi-threaded architecture.

New web interface has been developed that displays dashboards using iFrames adding an extra layer of protection by making the communication with Kibana and Elasticsearch database one way and the dashboard visualizations read only. HTTPS for the web interface is enabled using signed SSL certificates Incorporating TLS 1.4 to offer the highest standard of encryption during transit. Server-side authentication is used to authorize users to ensure the maximum security.

Finally, a streamlined deployment package is developed that condenses the 152 commands required to install and configure GLORIAD InSight into a single step.

Development of InSight2 has focused on improving on GLORIAD InSight while adding features on top of the existing functionality. Attention has been paid to make the system more modular, efficient, faster, lean and user-friendly. New InSight2 Fully utilizes the system resources in a multi-core environment of the new hardware provided by Cisco, at the same time keeping the number of cores and thread allocation to each core, automatic to suit for wide variety of systems. Asynchronous event modeling has been replaced with synchronous processing which enables more efficient use of disk, processing and memory resources of the system producing more responsive user experience. Related software has been updated to the latest versions to account for the bug fixes as well as security patches. Issues arisen due to backward incompatibilities with the older versions of the software have been taken care of, by the new code implementation. The information sources have been updated where outdated Threats Databases are removed and new actively maintained databases are added. The overall architecture requires fewer dependencies which would increase system stability in the long term. Standard software development principles are followed to ensure high degree of maintainability for future development and the extension of features.

With the development of InSight2 GSR was moved from MySQL to Elasticsearch and stored in a condensed manner. First GSR tables, ‘Domains’, ‘Classes’, ‘IPsDNS’, ‘Disciplines’, ‘IPLablels’, ‘GovAgencies’, and ‘IPsText’ were read into a ‘dictionary’ which is a data type similar to hashmap found in other languages. Then duplicates were removed. This information was sent directly into the Elasticsearch database using the Python client under a new index named GSR. This was only performed once and GSR index would be reused during the course of operation of the InSight2. Using the Elasticsearch backup function this index is backed up to disk so that deployment of InSight2 to other servers only requires the restoration of this database rather than reconstructing it from the beginning.

The new InSight2 is inspired by GLORIAD InSight but builds upon a completely new unified architecture developed from scratch. A simplified architecture has been implemented to offer greater functionality offered compared to GLORIAD InSight. Moving from the Perl programming language to Python made it possible to achieve the object oriented programming (OOP) without having to import third party libraries such as Perl::POE and implement native multi-threading. InSight2 implements a new synchronous data processing model compared to the asynchronous event model used in the GLORIAD InSight. This allows it to read, enrich and store data in one continuous flow. In the GLORIAD architecture animals are invoked and killed for each event. The purpose of an animal is to eat (Read input data and process), clean (perform periodic events such as clearing cache and trimming databases), answer (send data to the ZMQ queue), and ultimately terminate. InSight2 implements a standard Unix style system level service. This demon is invoked at the OS start and continues to function until shut down either processing raw Argus flow records or waiting for new Argus data to arrive. Comparison of the components between the architectures of GLORIAD InSight and InSight2 is outlined in the Figure 5.1.

Argus archives are enriched using 40 parallel threads utilizing the total number of cores available in ‘Bulika’ server. First ‘racluster’ Argus client is used in order to extract Argus archives loading 5 minutes of Argus data iteratively until all the files belonging to the particular day is processed, at which point it will move to the next folder. Argus archives are separated into individual files by 5 minutes when they are created by the Argus server but EM is capable of processing Argus archives of arbitrary length in time. Feature, stime, trans, flgs, dur, proto, pkts, bytes, appbytes, pcr, load, loss, ploss, retrans, pretrans, rate, tcprtt, synack, ackdat, tcpopt, stos, dtos, shops, dhops, sintpkt, dintpkt, sjit, djit, svid, dvid, smeansz, dmeansz, smaxsz, dmaxsz, sminsz, dminsz, saddr, daddr, sport, dport, sbytes, dbytes, sload, dload, srate, and drate are extracted from each flow record. These are an optimized list of features and are different from the feature list extracted GLORIAD InSight. Detailed explanations of these fields are presented in Appendix A.

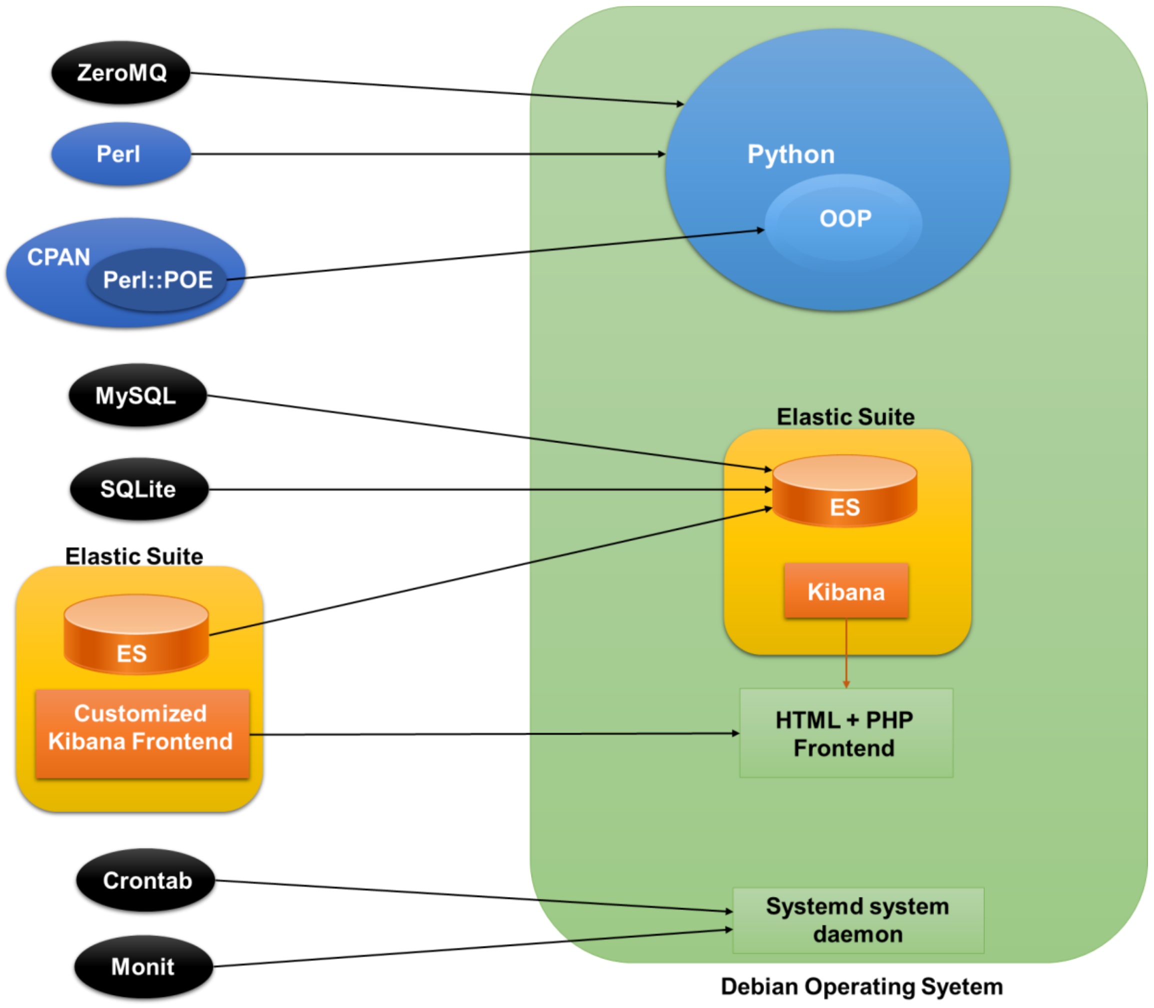


Figure 5.1 GLORIAD InSight vs. InSight2 Architecture

In comparison fields, proto, state, srcid, stime, ltime, dur, saddr, daddr, sport, dport, pkts, spkts, dpkts, bytes, sbytes, dbytes, appbytes, sappbytes, dappbytes, ploss, sploss, dploss, smac, dmac, svid, dvid, sjit, djit, flags, trans, tcpopt, sas, das, pcr, tcprtt, swin, dwin, load, sload, dload, inode, sco, dco are extracted by 'Elephant' for the enrichment by GLORIAD InSight farm. These features are explained in the Appendix A. There are some redundant features in this list of features which are identified in the recommendations section of chapter 4.

Each individual archive is compressed to using 'GZIP' compression technique saving disk space but requiring more steps to extract the archives. However, this step needs to be processed only one time since the processing is performed in a pipeline. The data processing pipeline of EM consists of pre-processing, data enrichment which involves tagging network flows seen in the incoming Argus records with the information taken from GSR, geographical locations of each IP address from Maxmind GeoIP database and Threats Database, and upload to the Elasticsearch database. The high-level data flow using the software used in the InSight2 is shown in the Figure 5.2.

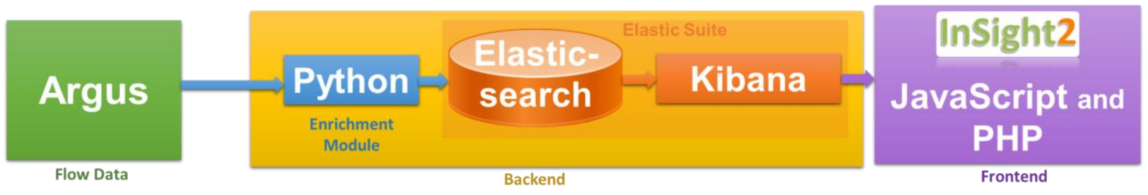


Figure 5.2 High-level Data Flow Architecture of InSight2

From obtaining flow information from Argus archives, organizational information from the GSR, geographical location information from the Maxmind GeoIP database, security information from various websites that maintain up-to-date threat related information, processing all this information to uploading to the Elasticsearch database is performed synchronously. Each flow record is sent through a pipeline of processes to ultimately be inserted into a database. This allows InSight2 to have a streamlined architecture with less moving parts. While this eliminates ZeroMQ, Monit and the farm architecture it is still capable of enriching data using user-provided databases by adding them to Elasticsearch. These custom databases can be stored and maintained in a separate index within Elasticsearch database like the GSR. The overview of the system architecture of the InSight2 platform is outlined in the Figure 5.3.

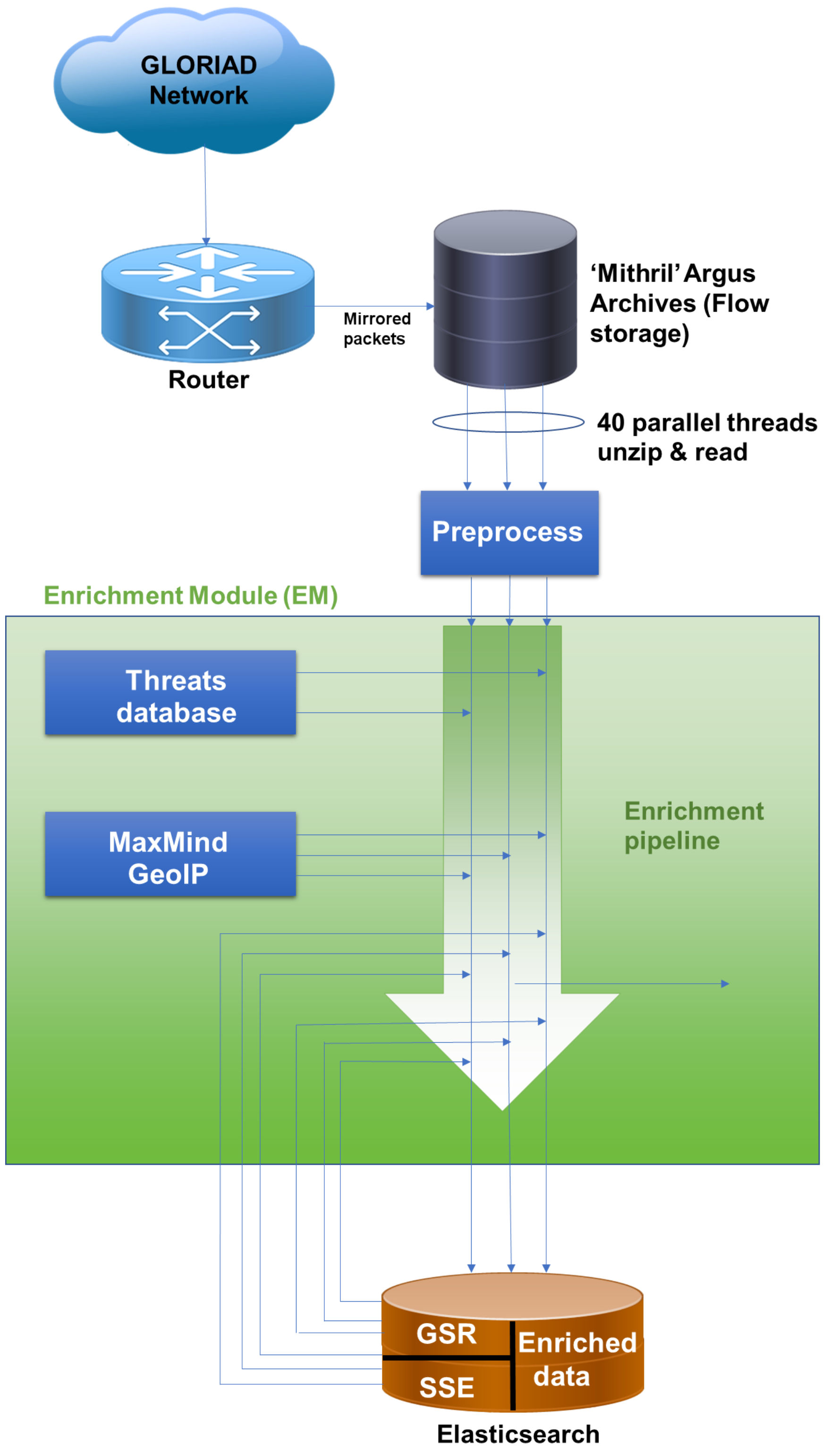


Figure 5.3 InSight2 System Architecture

5.2 Data Pre-Processing

Data pre-processing involves extracting the bi-directional flow information from the Argus archives and aggregating them by session. GLORIAD InSight’s ‘Elephant’ animal would drop network flows that are less than 1500 Bytes which accounts for 86% of the total number of flows. This is a significant number of flows unaccounted for. Even though this will reduce the number of transactions needed to process by the farm of animals and eventually reduce the number of records entered into the elastic search database, it fails to deliver a complete picture of the network. This small flows accounts for the majority of the network flows however larger natural flows that are bigger than 1500 Bytes accounts for the largest traffic sent through the network. Malicious events such as denial of service (DoS) attacks involves sending large amounts of small packets to the target server in order to overwhelm the server to diminish its function. When these Network flows are dropped network monitoring software cannot see the entire details of the network behavior, and can cause misdiagnosis and incorrect prediction.

InSight2 accounts for all the traffic. This is achieved by the efficient and high performance implementation of the EM which utilizes all the cores in the host system. During this pre-processing step EM aggregates all the network flows that belong to the same session. A session belongs to a collection of packets that define a single transaction. This transaction can vary from loading an image from a website to a sending of an email. By aggregating the flows that belong to a session duplicate flows that are counted separately for the session are combined into one session.

GLORIAD InSight utilizes the flows according to the ‘RMON’ specification which breaks the network flow into per direction flows. This negates the advantage of using network flow provided by Argus, since they are defined as bi-directional. InSight2 utilizes network flows by accepting the bi-directional of flow and combining it with the PCR value which defines the ratio of the sender and receiver. This PCR value is defined for the entire selection of the time frame as well as at the advanced metrics section of the visualizations. This allows the network administrator to investigate the general direction of the data transmission for a given time frame. This allows the identification of the network exfiltration activity. Network exfiltration is the process of stealthily gathering data from an organization, by using internal employees or Trojan software. This can be identified by the extreme skewness of the PCR graph.

5.3 Data Enrichment

The central enrichment code of InSight2 is named the 'Enrichment Module' (EM) and it handles all the functions from reading from the Argus archives and gathering information from GSR other Threats Databases (TD) to the insertion into Elasticsearch database. This solves the major drawback of the farm architecture duplication of the read actions from the Argus archives. Archives are read, only once throughout the entire procedure of enrichment. This streamlines the data flow and eliminates the additional overhead of decompressing to archives multiple times and the additional CPU and memory usage caused by different animals trying to access them simultaneously.

The EM is capable of utilizing all the cores found in the system. This multi-threaded environment is achieved using native implementation of 'multi-processing' library of the Python programming language. It does not require any additional software to be installed, since it is part of the core language implementation. 'Bulika' hosts the EM and its CPU is capable of handling 40 simultaneous threads. This decreases the time needed to enrich network flow records by almost 40 times, a process which is already streamlined by the synchronous model. Argus archives are read over the network from the main data storage server 'Mithril'. this does not incur any bottlenecks since 'Bulika' and 'Mithril' are connected via high-speed gigabit Ethernet links that spans over a short distance, since they are located in the same server room. This further reduces the overhead incurred by 'Bulika' since the archives are read through the network instead of from the local disk delegating the read overhead to 'Mithril's CPU and network controller. 384GB of RAM is efficiently used and read and write to the actual disk is avoided whenever possible. 100GB Of swap space was allocated for extreme cases in the unlikely event that the memory runs out.

Argus client 'racluster' was used to extract the data. Feature list to be extracted from the records is updated after rigorous experiments to see which fields are useful and provides meaningful information. Final list of features are described in the Appendix A. During this extraction no filter has been used to filter out flows as in the case of 'elephant'. Retaining the full amount of information portrays a complete and accurate representation of the network flows. It is also noted that Argus metadata is not sampled and represents all the flows transmitted per second throughout the existence of GLORIAD since 2012 to 2015. This has been achieved due to the multi-processing architecture of InSight2. When the Argus archives are received by the EM they are distributed across the threads and 40 threads are spawned in order to perform the enrichment. Each flow record is enriched with different kinds of information gleaned from different sources such as the GSR, geographical location information from GEOIP database and Threats Databases which requires looking up the IP address of the source and destination within these three locations. Out of these three locations GSR accounts for the largest database and it is kept in a separate index inside Elasticsearch database. Elasticsearch offers efficient and scalable full text search engine in addition to serving as a database. It is feasible to store the GSR in Elasticsearch since most of the entries in GSR are text based. All indices of Elasticsearch database are spread across the 3 servers ‘Bulika’, ‘Lona’ and ‘Lilith’. Elasticsearch is capable of offloading its workload to multiple threads found in system and this distributed nature allows it to search faster by utilizing 120 threads simultaneously.

Each thread will enrich one flow record at a time and partially completed network flow records are kept in memory until the enrichment is completed by each thread in a ‘dictionary’ data type. When all the threads have completed their enrichment result is added into JSON document that is kept in memory. Finally an instance of Elasticsearch client for python is instantiated and this JSON document is passed to it using JSON serializer. Elasticsearch database client support directly reading from JSON documents and this passage is performed internally as an object. The EM Does not require any additional modules to be installed in the Target system except for Elasticsearch database client producing minimal footprint in the target system.

Finally, the completely enriched JSON record is sent to the Elasticsearch database along with the index name as the date of the Argus record. A separate index is maintained within the Elasticsearch database that holds the predicted results in a timeline about the network utilization. This prediction is performed the using Markov Chains. Probability matrix is generated using data observed within the past week. Using this matrix a Markov Chain is generated that contains 10 levels of network utilization in 10% increments. More information is detailed in the performance dashboard where this information is visualized.

5.3.1 Information Sources

Argus data is read from Radium server or directly reading from archives. Argus fields used in InSight2 is described in the Table 5.1.

Table 5.1 Argus Fields Used in InSight2

Field Name	**Field property**	**Description**
stime	Per flow	Start time of the flow record
saddr, daddr	Per direction	Source or destination IP address
sport, dport	Per direction	Source or destination port
flgs	Per flow	Flags of the Flow State
proto	Per flow	Protocol
stos, dtos	Per direction	Type of service byte value of the source or destination
sttl, dttl	Per direction	Source time to live: source to destination or destination time to live: destination to source value
spkts, dpkts	Per direction	Packet account from source to destination or destination to source
sbytes, dbytes	Per direction	Transaction bytes from source to destination or destination to source
sappbytes, dappbytes	Per direction	Application bytes from source to destination or destination to source
sload, dload	Per direction	Load from Source or destination in bits per second

Table 5.1. Continued.

Field Name	Field property	Description
sloss, dloss	Per direction	Packets retransmitted or dropped from Source or destination
sintpkt, dintpkt	Per direction	Inter packet arrival time from Source or destination
sjit, djit	Per direction	Jitter observed at source or destination
tcprtt	Per flow	Round trip time of the connection
synack	Per flow	Time to set up connection between SYN and SYN_ACK
ackdat	Per flow	Time to set up connection between SYN_ACK and ACK
tcpopt	Per flow	Connection options observed or the lack of it during connection initiation
spktsz, dpktsz	Per direction	Histogram of the distribution of package sizes from Source or destination
dur	Per flow	Flow duration
rate, srate, drate	Per direction	Package rate in packets per second
trans	Per flow	Total record count of the incoming Argus stream

Table 5.1. Continued.

Field Name	**Field property**	**Description**
pkts	Per flow	Number of packets seen in the transaction
bytes	Per flow	Number of bytes seen in the transaction
appbytes	Per flow	Number of application bytes seen in the transaction
load	Per flow	Network load observed in the incoming Argus flow in bits per second
loss	Per flow	Number of packets retransmissions or dropped

Scripted fields allow the generation of fields on-the-fly by calculating the value of these fields during runtime. They reduce the disk usage by not storing them in the disk. Table 5.2 shows the scripted fields are used in the InSight2.

Table 5.2 Elasticsearch Scripted Fields Used in InSight2

Scripted Field Name	**Argus Field Used**	**Description**
srcGB	src bytes	Convert Bytes transmitted from the source into Terabytes.
sstGB	dst bytes	Convert Bytes transmitted from the destination into Terabytes.
GB	bytes	Convert total Bytes transmitted from the source and destination into Terabytes.
pretransGB	pretrans	Convert total Bytes re-transmitted from the source and destination into Gigabytes.
loadG	load	Convert the network load from bytes per second to Gigabytes per second
rateB	rate	Convert the network load from packets per second to billions of packets per second
retransM	retrans	Convert Total packet retransmissions from packets per second to millions of packets per second
lossM	loss	Convert Total packets lost from packets per second to millions of packets per second
synackAvg	synack	Calculate average of TCP connection setup round-trip time, sum of SYNACK and ACKDAT
intpktAvg	intpkt	Average of the total of source and destination interpacket arrival time in milliseconds
jitterAvg	jitter	Average of the source or destination active jitter in milliseconds

Script fields enable Elasticsearch to have virtual fields that are programed using ‘Painless’ scripting language. Painless is based on the syntax of Java to provide Groovy-style scripting language features. This enables Elasticsearch to extend its functionality. Painless language is several times faster than other alternate alternative languages [28]. And offer safe execution while offering object-oriented programming by offering fine-grained white list with method call and field granularity. The variables are optionally typed where necessary by using the ‘def’ keyword. It is designed specifically for Elasticsearch scripting. We have defined dynamically generated fields such as Gigabytes derived from Bytes field from the original data source, and load in megabytes per second in addition to bytes per second for the use of certain visualizations etc. A complete list of scripted fields can be found in the Appendix A.

InSight2 enriches its data using three sources of information; GSR, GeoIP and ‘Threats Database’ (TD). GSR was re-constructed using relevant tables in the ‘Main database’ stored within the MySQL database and uploaded to Elasticsearch. The relevant tables and their contents that were found in the original ‘Main database’ is described below:

• iplabels	-	IP names of the individual nodes
• ips	-	IP address to domain ID matching
• ipsdns	-	Domain name server information
• ipstext	-	Location information such as coordinates, country etc.
• asnums	-	autonomous system numbers
• disciplines	-	discipline; nuclear engineering, ocean science etc.
• domains	-	Domain ID the IP address belongs to
• govagencies	-	name of the government agency
• orgclass	-	classification of the organization

GSR is stored in the form of a JSON records within the Elasticsearch database index named ‘gsr’ with the following fields, and Appendix A shows index mapping for the ‘gsr’ index:

• ip	-	IP address
• ipname	-	IP name
• organization	-	Organization name

• domainid	- Domain ID
• dns	- DNS name
• isp	- ISP name
• labels	- Additional labels
• asnum	- AS number
• discipline	- Discipline
• agency	- Agency
• application	- Application

MaxMind GeoIP database was in the format of *.dat* files which contained different location related information described below:

• GeoIPASNum.dat	- Autonomous number systems for ipv4
• GeoIPASNumv6.dat	- Autonomous number systems for ipv6
• GeoIPCity.dat	- city name for ipv4
• GeoLiteCityv6.dat	- city name for IPv6
• GeoIPOrg.dat	- organization name
• GeoIPDomain.dat	- domain name
• GeoIPISP.dat	- internet service provider name
• GeoIP.dat	- ipv4 related information
• GeoIPv6.dat	- ipv6 related information

These are extracted from ‘Bluemac’ for future use for the time being. InSight2 uses up-to-date current MaxMind GeoIP *.mmdb* format which consists of just one file database containing geographical coordinates, country, city, region, zip code. This new mmdb format is faster, compact, easy to update and more portable than the previous .dat files. The following file is used for this purpose:

- GeoLite2-City.mmdb

The following databases are extracted from ‘Bluemac’ which were used as the ‘Bad Actors’ database by GLORIAD InSight:

- Emerging threats: https://rules.emergingthreatspro.com
- Bogon: http://www.team-cymru.org,
- Zues botnet: https://zeustracker.abuse.ch,
- Feodo: https://palevotracker.abuse.ch,
- Palevo: https://feodotracker.abuse.ch.

This is taken from the TD previously known as ‘Bad Actors’. The updated TD contains not just traditional malicious IP addresses but also IP addresses belonging to compromised nodes and ransomware servers. This database helps identify ransomware servers as seen in the cases of ‘WannaCry’, ‘Petya’ and ‘NotPetya’ ransomware in rapid succession June and July of 2017

Once EM categorizes a network flow as belonging to one of these lists it will tag them by its unique single letter tag which Elasticsearch will use for indexing for fast search and retrieval. The following updated databases are the TD.:

- Bogon: http://www.team-cymru.org/Services/Bogons/fullbogons-ipv4.txt
- Zues botnet: https://zeustracker.abuse.ch/blocklist.php?download=badip
- Compromised: https://rules.emergingthreatspro.com/blockrules/compromised-ips.txt
- Ransomware: https://ransomwaretracker.abuse.ch/downloads/RW_IPBL.txt
- Feodo: https://feodotracker.abuse.ch/blocklist/?download=ipblocklist
- Palevo: https://palevotracker.abuse.ch/blocklists.php?download=ipblocklist

5.3.2 Distributed Database Structure

InSight2 uses Elasticsearch to store different databases and as a search engine to search for records within that database. It is ideal since both functions are seamlessly integrated into one solution and the data stored within the database consists mainly of text information. Elasticsearch is based on Apache Lucene which is a low level high performance text search engine written in Java. It is capable of processing up to 150GB records per hour. Elasticsearch allows the database to be distributed across more than one server. This increases resiliency against failure and increases overall performance since indexing and retrieval of the data is performed in parallel across multiple servers using multiple processor cores in each server. A ‘shard’ is a Lucene [29] instance that is managed by Elasticsearch that acts as a low-level worker unit. Each of the ‘Bulika’, ‘Lona’, and ‘Lilith’ servers carry five shards each. Shard contains a portion of the database that can be located within same system or across different physical or virtual systems. This gives Elasticsearch the edge for faster access times that does not increase with the amount of data contained within the database, hence it is a highly scalable solution for big data systems. Indexing allows records to be searched in minimal delay. Unlike traditional databases such as MySQL and SQLite which requires sequential advancing through the table to find the relevant document, indexed databases use an index to find the relevant document In a fixed time. The time complexity of searches is in the scale of O(n). Figure 5.4 show the shard allocation of the 3 servers used by InSight2.

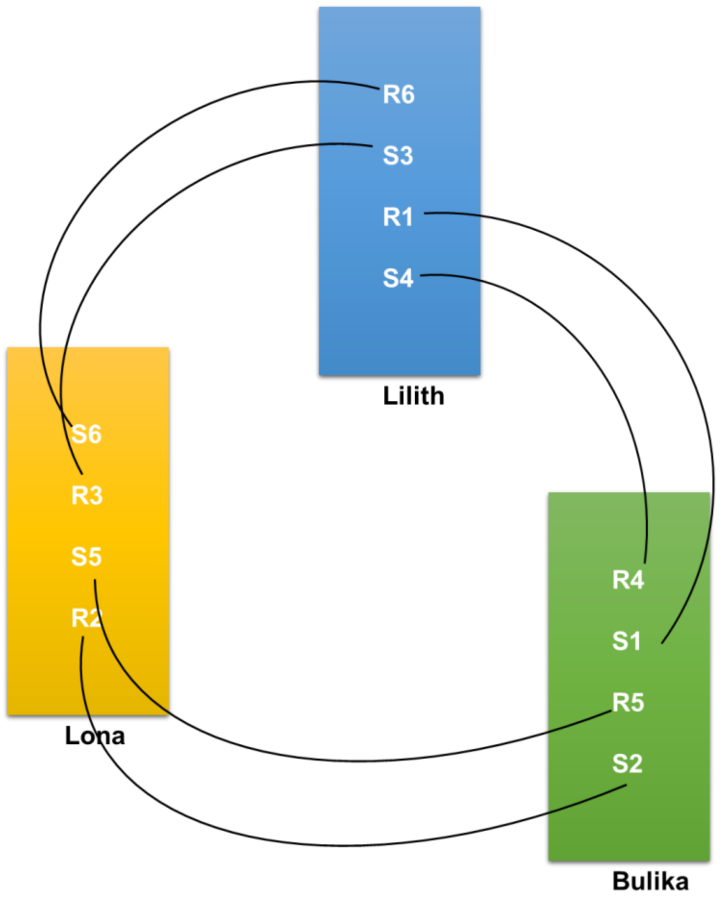


Figure 5.4 Elasticsearch Database Shard Allocation

InSight2 keeps daily indexes which enables the system to implement efficient rolling mechanism to store a fixed number of days within the database depending on the limitations on the storage capacity. By using a separate index to keep the data of a single day it makes it fast and efficient to purge outdated indexes since deleting an index is performed in an optimized manner compared to deleting record by record in Elasticsearch, and is performed using a single command.

The GSR is kept within the Elasticsearch in a separate index. This fully utilizes the capabilities of the full text search engine of Elasticsearch. When EM enriches incoming data it will look up the information in the GSR to tag the network flows. Since all the indexes stored within the database are optimized for searching and every read and write action is performed using all the available CPU cores within the host system it is the most feasible location to store GSR database.

Elasticsearch is no-SQL database, in other words it allows having scheme-less data structures which allows faster data access compared to SQL databases [30]. This allows the end user to define new data structures on the way without changing the initial data type mapping. InSight2 uses a semi-strict architecture when it comes to the definition of data types. The definitions imposed by the original mapping can be overridden by adding new data with new field definitions, at the same time it will discard any malformed data that belongs to an existing data type. This covers a middle ground compared to completely relaxed type definitions and completely strict type definitions. Relaxed definitions allow any new data type to be added to the index template and in case of an error for example if an existing data type is misspelled it will create a new data type. Completely strict definitions offer no flexibility to introduce new data types and this is not feasible as the user requirements grow it will require new data types.

5.3.3 High Maintainability

Server setup uses RAID 6 for redundancy of the disks in case of disk failure which is transparent from the underlying OS. OS upgrades are not affected by the RAID configuration unlike the ZFS configuration which needs to be set up every time the OS is reinstalled or system is replicated in a remote server.

Compared to the previous architecture where message passing software such as ZeroMQ (ZMQ) was required due to the limitation of the multiple component architecture new InSight2 architecture implements data retrieval, filtering, aggregation, enrichment and storage are performed using a single software module eliminating the need for multiple scripts and communication infrastructure.

One of the reasons that made GLORIAD InSight difficult to maintain was the fact that components of the system were tied down to specific versions of Java Elasticsearch and Kibana. GLORIAD InSight used libraries from Perl repository CPAN which added another layer of complexity since these components had to be maintained separately unlike Python libraries which can be installed and upgraded alongside a system upgrade.

Repositories of Linux system has been used to install tools needed instead of downloading and compiling specific versions of software, which may become unavailable at some point of time. When new versions of these software gets released a simple upgrade of the OS updates all of them. Since these modules are updated with backwards compatibility in mind by the maintainers of the software repository distribution updates will not cause system instability increasing the maintainability of the system as a whole.

GSR Stored within the Elasticsearch database has been backed up in order to be used in new servers without the need to extract and de-duplicate the records from the MySQL databases for every installation.

Well commented and structured code of InSight2 contributes to the high maintainability of the platform. InSight2 also offers detailed documentation about the design and implementation of the system. A video tutorial has been created to guide new users about the design philosophy of the web interface as well as its usage. An installation guide is provided with the one-step installation package which describes the minimum system requirements and the installation procedure.

5.3.4 Lean System Design

InSight2 Is built with lean system design in mind. The many software components that GLORIAD InSight used have been disposed of. Message passing is completely performed internally using object-oriented programming (OOP) which makes message passing ZeroMQ bus redundant. Synchronous processing is adopted in favor of asynchronous event model which also helps eliminate the need for publisher subscriber style message passing bus since a unified EM handles all the data enrichment duties.

A unified database Structure was introduced which contains both final enriched data as well as GSR database using Elasticsearch. It is feasible to use this scalable and fast full text search engine enabled database to store data in an arbitrary manner which is indexed to increases the searching efficiency. Traditional sequential databases are not capable of adapting to the growing needs of a scalable database as well as being able to accommodate schemeless data structures. This allows InSight2 to utilize and reuse already installed software which is part of the core package.

A programming language for large-scale project development with ease of maintainability and efficiency was selected in order to improve execution speed and reduce number of moving parts needed. The language of choice Python programming language has built-in OOP which negates the necessary to install third party package repositories and libraries.

InSight2 uses the package management software, ‘apt’, that is included in the OS by default to download and install other necessary software. It only requires one command to download and install a list of dependencies which are inherently minimalistic and limits to the essentials. If the software is compiled and installed Within the host device it requires additional programs for the compilation and this additional software will be left in the system with no function after the install.

EM Is implemented as a low-level system service. In GLORIAD InSight individual script is managed by centralized program, Monit, which keeps track of all the running programs and is responsible for responding any terminated scripts. By integrating EM as a system service management of the software is greatly simplified and uses less system resources since it uses existing Service layer.

5.4 Data Analytics

Prediction of the network utilization has been performed using Markov Chains. Markov Chains is a stochastic model describing a sequence of possible events. It describes states and their transition probabilities. Each state is defined as the state of network utilization in Bytes. The transition probability defines the likelihood of transitioning from one state to the next. States can be transitioned from one state to another or stay in their current state depending on their transition probability. This will decide in each time segment whether the system will stay in the same state or move to the next. Probability of the next state only depends on the current state the system is at and not the path taken to arrive at the current state and this is known as the Markov property [31]. Markov Chain can be visualized to using a graph as seen in the Figure 5.5. This is particularly useful to see the probability and also to plot the transition matrix.

Prediction is achieved by first observing the network activity for the time period of 5 days, generating the state transition Matrix of the Markov Chain, deriving the steady-state matrix, and calculating the predicted usage for the next 24 hours.

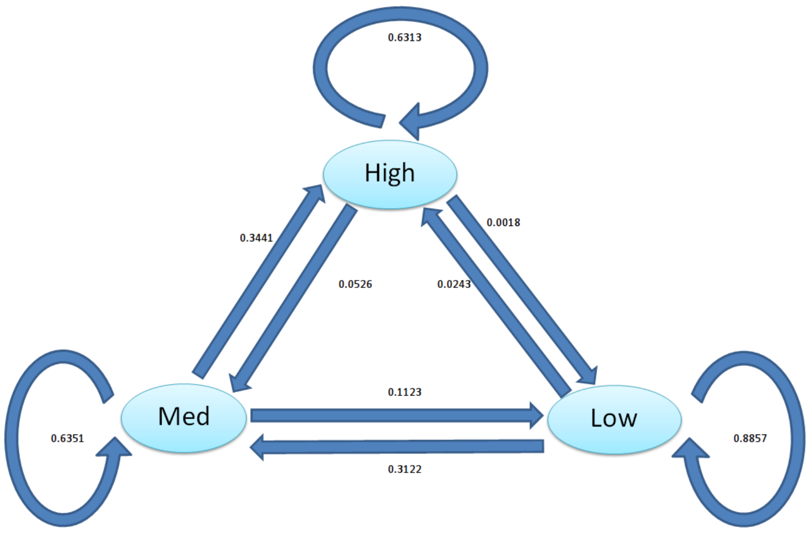
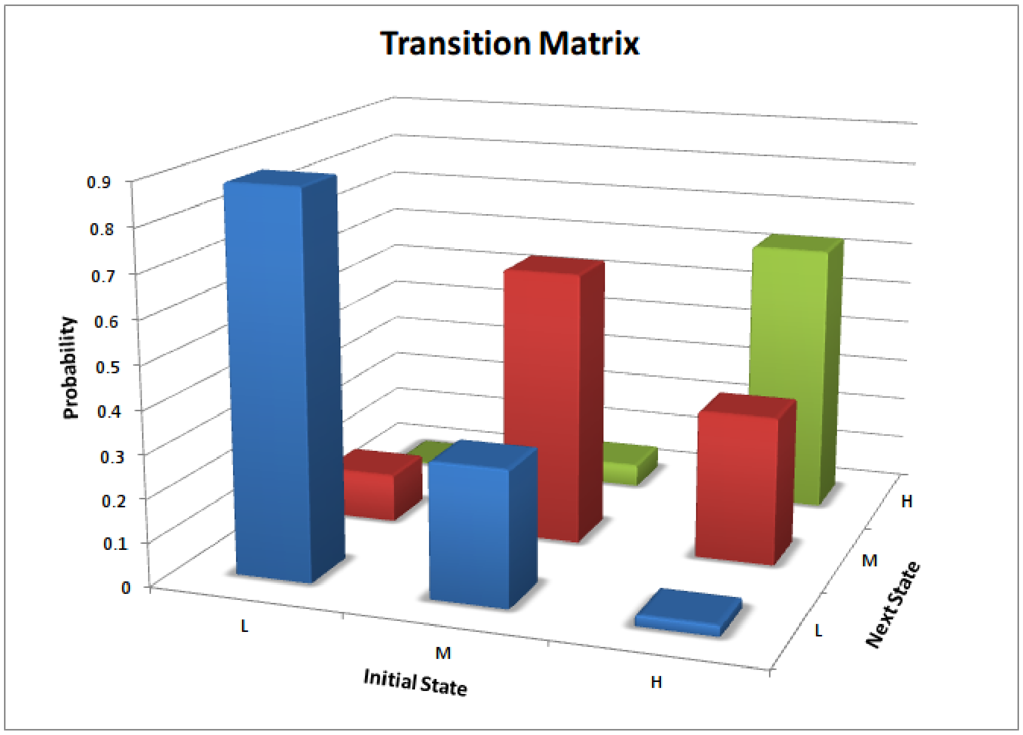


Figure 5.5 Markov Chains State Diagram

Figure 5.6 shows the state transition matrix in 3D bar chart. It is observed that states prefer to stay in their current state. This is reflected by the high probability for low to low transition, medium to medium transition, and high to high transition, compared to the other transition probabilities. This is explained by the network staying within certain utilization for certain amount of time. This is true especially for research and education networks where a large amount of data is being transmitted in bulk. When data transmissions are started they will occupy a large amount of network bandwidth and will last until the entire transmission is complete, such as sending large amount of research data from one institution to another. Transition to the adjacent states shows the next highest probability causing a smooth transition.



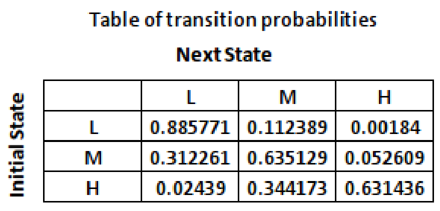


Table of transition probabilities

Next State

Initial State	L	M	H
L	0.885771	0.112389	0.00184
M	0.312261	0.635129	0.052609
H	0.02439	0.344173	0.631436

Figure 5.6 Markov Chains Transition Matrix

Steady-state matrix is derived from this transition matrix. Steady-state matrix shows the long-term probabilities of the Markov Chain. Figure 5 .7 shows the steady-state matrix of the Markov Chain. Using the generated Markov Chain a predicted proposition is generated into the next 24 hours for the network capacity utilization. This information is plotted in visualization in the interface of InSight2. Figure 5.8 illustrates the predicted network utilization in the near future generated by the Markov Chain. Mean, variance and their log versions are plotted in the Appendix C.

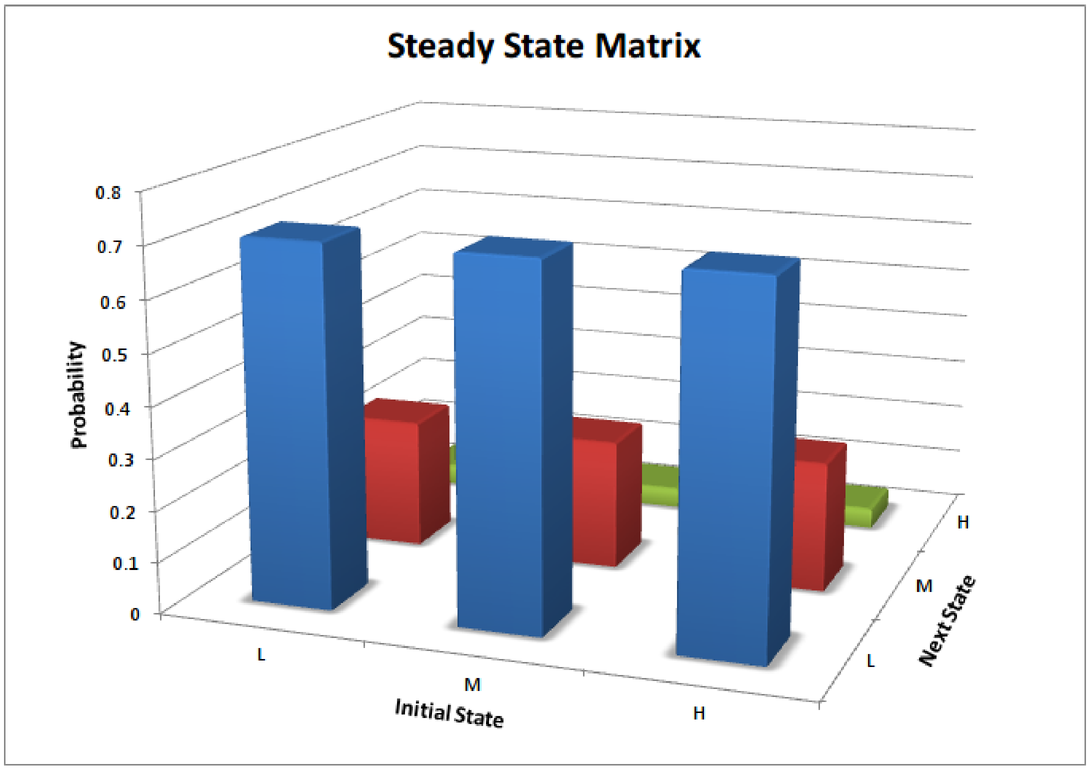


Table of Steady State transition probabilities

Next State

Initial State

	L	M	H
L	0.705234	0.254866	0.039901
M	0.705234	0.254866	0.039901
H	0.705234	0.254866	0.039901

Figure 5.7 Markov Chains Steady State Matrix

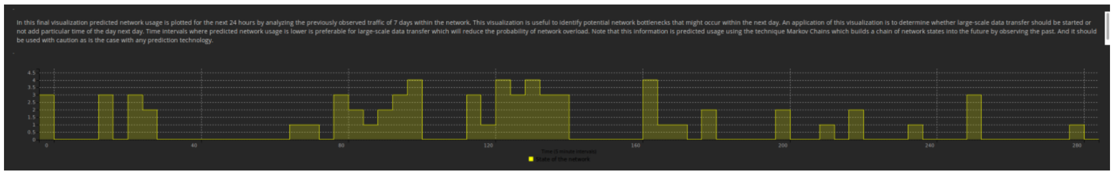

In this final visualization predicted network usage is plotted for the next 24 hours by analyzing the previously observed traffic of 7 days within the network. This visualization is useful to identify potential network bottlenecks that might occur within the next day. An application of this visualization is to determine whether large-scale data transfer should be started or not add particular time of the day next day. Time intervals where predicted network usage is lower is preferable for large-scale data transfer which will reduce the probability of network overload. Note that this information is predicted usage using the technique Markov Chains which builds a chain of network states into the future by observing the past. And it should be used with caution as is the case with any prediction technology.

Figure 5.8 Markov Chains Prediction for Next 24 Hours

5.5 Data Visualization

InSight has a completely redesigned interface that is inspired by GLORIAD InSight. InSight2 focuses on improving the usability for all end-users of different technical backgrounds. It groups information into different categories enabling the user to find necessary information quickly and easily. Steps have been taken to ease the learning curve of InSight2 by strategically placing annotations to describe functionality and usage and video tutorials to show how to perform complex filtering to obtain the information that the end user is looking for.

InSight2 merges many different dashboards that were separately implemented in the GLORIAD InSight. This allows the user to consume the whole breath of information that is relevant for one topic under one tab. They are optimized to render in different screen sizes and browsers. This enables the user to observe the network no matter where he/she is located and which device he/she is using.

Any browser that supports JavaScript is capable of handling InSight2. From Kibana 5 visualizations are rendered using graphical processing unit (GPU) of the host system decreasing the render-time compared to the previous versions, improving the overall responsiveness of InSight2 web-interface.

When InSight2 is loaded for the first time Kibana will send a HTML page with embedded JavaScript. InSight2 uses iFrames to extract the dashboards from Kibana. This method enables segregation of end users from the developers. Developers are allowed to modify the dashboards while the end-users are given access to only view the dashboards and to apply filters. Not only this improves the security of the system but also prevents unintentional changes to the dashboards.

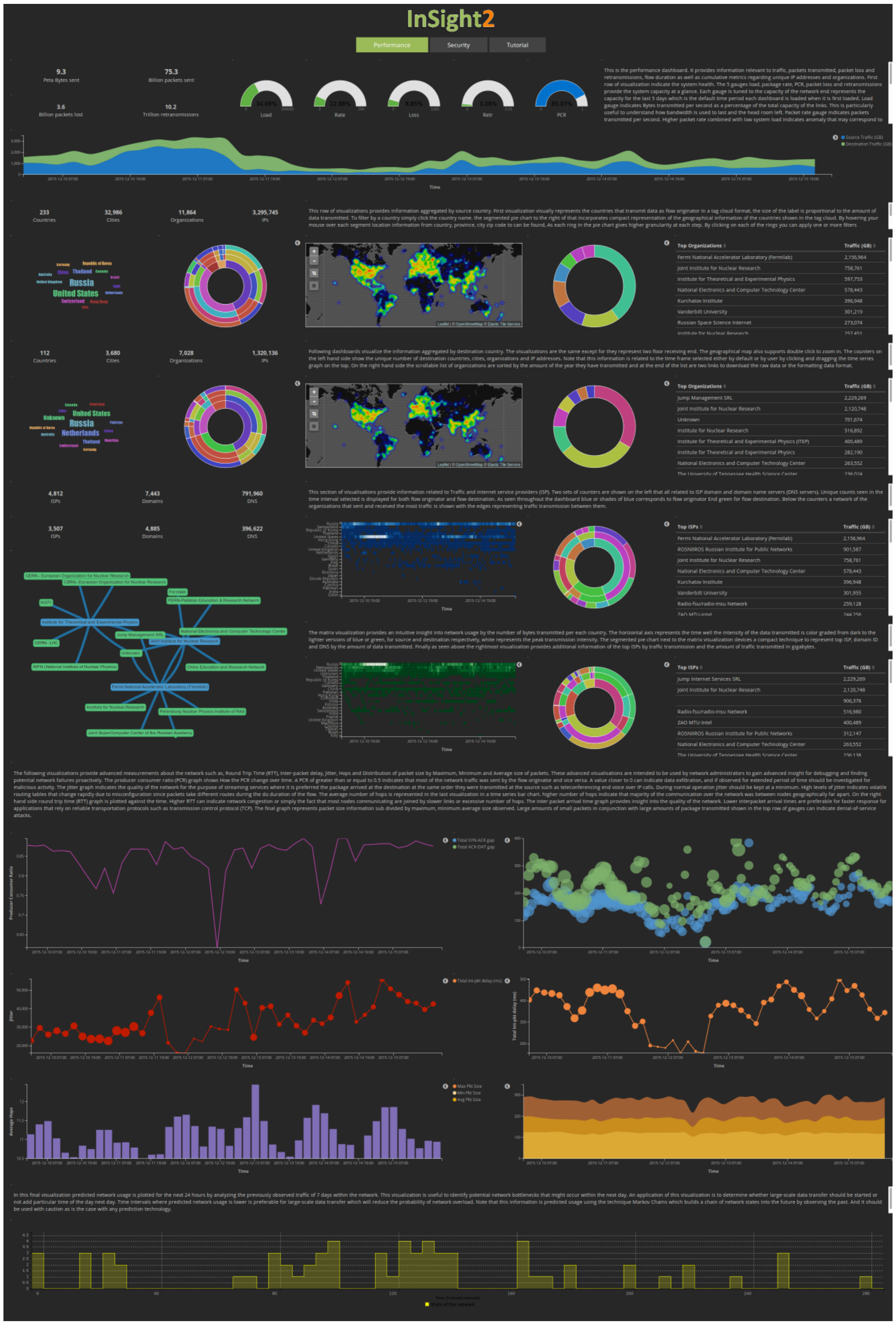
The InSight2 interface incorporates uniform material design to preserve the integrity and cohesiveness. It incorporates modern visualizations and delivers a consistent user experience throughout. Color scheme is carefully selected to emphasize visualizations by adopting a dark background. All visualizations that show source traffic are color coded by blue and destination in green. Visualizations that show intensity of the traffic use different shades of these colors. In cases where neither originator nor destination node is not represented different colors came is use such as red orange or purple etc.

5.5.1 Performance Dashboard

The performance dashboard provides information relevant to traffic, packets transmitted, packet loss and retransmissions, flow duration as well as cumulative metrics regarding unique IP addresses and organizations. Figure 5.9 shows the performance dashboard.

The first row of visualization indicates the system health. The five gauges, namely, load, package rate, PCR, packet loss and retransmissions, provide the system capacity at a glance. Load gauge indicates Bytes transmitted per second as a percentage of the total capacity of the links. This is particularly useful to understand how bandwidth is used to last and the head room left. Packet rate gauge indicates packets transmitted per second. Higher packet rate combined with low system load indicates anomaly that may correspond to very specific file transfer, a misconfigured server or denial-of-service attack. Packet loss and retransmissions gauges show how the system handles dropped packets. During normal operation all four load, packet rate, loss and retransmissions gauges should be as minimal as possible. The indicator will be green if the metric is within the acceptable range amber when It starts to go beyond this range, orange when attention is needed and ultimately red when system failure is imminent. Since all the network data is collected as flow information Producer Consumer Ratio (PCR) can be used to understand the direction of the packet transmissions. Higher the PCR more packets transmitted from the flow initiator to the flow receiver. This gauge reflects the unified material design of the dashboards using blue for source and green for destination. in most circumstances direction of the flow is from the flow originator to the flow receiver. However either blue or green gauge does not indicate a problem.

The second row of visualizations provides information aggregated by source country. First visualization visually represents the countries that transmit data as flow originator in a tag cloud format, the size of the label is proportional to the amount of data transmitted. To filter by a country simply click the country name. the segmented pie chart to the right of that incorporates compact representation of the geographical information of the countries shown in the tag cloud. By hovering your mouse over each segment location information from country, province, city zip code to can be found. Each ring in the pie chart gives higher granularity at each step. By clicking on each of the rings you can apply one or more filters depending on the depth of the ring. For example, by clicking on a segment in the zip code in the outermost ring 3 filters from country, province and city will be selected. By scrolling up to the top of the dashboard please filters can be verified. By clicking “Apply Now” button multiple filters can be applied in a single click.



9.3 Peta Bytes sent

75.3 Billion packets sent

3.6 Billion packets lost

10.2 Trillion retransmissions

This is the performance dashboard. It provides information relevant to traffic, packets transmitted, packet loss and retransmissions, flow duration as well as cumulative metrics regarding unique IP addresses and organizations. First row of visualization indicate the system health. The 5 gauges load, package rate, PCR, packet loss and retransmissions provide the system capacity at a glance. Each gauge is tuned to the capacity of the network end represents the capacity for the last 5 days which is the default time period each dashboard is loaded when it is first loaded. Load gauge indicates Bytes transmitted per second as a percentage of the total capacity of the links. This is particularly useful to understand how bandwidth is used to last and the head room left. Packet rate gauge indicates packets transmitted per second. Higher packet rate combined with low system load indicates anomaly that may correspond to

233 Countries | 32,986 Cities | 11,864 Organizations | 3,295,745 IPs

This row of visualizations provides information aggregated by source country. First visualization visually represents the countries that transmit data as flow originator in a tag cloud format, the size of the label is proportional to the amount of data transmitted. To filter by a country simply click the country name. the segmented pie chart to the right of that incorporates compact representation of the geographical information of the countries shown in the tag cloud. By hovering your mouse over each segment location information from country, province, city zip code to can be found, As each ring in the pie chart gives higher granularity at each step. By clicking on each of the rings you can apply one or more filters

Top Organizations	Traffic (GB)
Fermi National Accelerator Laboratory (Fermilab)	2,156,964
Joint Institute for Nuclear Research	758,761
Institute for Theoretical and Experimental Physics	597,753
National Electronics and Computer Technology Center	578,443
Kurchatov Institute	396,948
Vanderbilt University	301,219
Russian Space Science Internet	273,074
Institute for Nuclear Research	257,451

112 Countries | 3,680 Cities | 7,028 Organizations | 1,320,136 IPs

Following dashboards visualize the information aggregated by destination country. The visualizations are the same except for they represent two floor receiving end. The geographical map also supports double click to zoom in. The counters on the left hand side show the unique number of destination countries, cities, organizations and IP addresses. Note that this information is related to the time frame selected either by default or by user by clicking and dragging the time series graph on the top. On the right hand side the scrollable list of organizations are sorted by the amount of the year they have transmitted and at the end of the list are two links to download the raw data or the formatting data format.

Top Organizations	Traffic (GB)
Jump Management SRL	2,229,269
Joint Institute for Nuclear Research	2,120,748
Unknown	701,674
Institute for Nuclear Research	516,892
Institute for Theoretical and Experimental Physics (ITEP)	400,489
Institute for Theoretical and Experimental Physics	282,190
National Electronics and Computer Technology Center	263,552
The University of Tennessee Health Science Center	236,024

4,812 ISPs | 7,443 Domains | 791,960 DNS

3,507 ISPs | 4,885 Domains | 396,622 DNS

This section of visualisations provide information related to Traffic and internet service providers (ISP). Two sets of counters are shown on the left that all related to ISP domain and domain name servers (DNS servers). Unique counts seen in the time interval selected is displayed for both flow originator and flow destination. As seen throughout the dashboard blue or shades of blue corresponds to flow originator End green for flow destination. Below the counters a network of the organizations that sent and received the most traffic is shown with the edges representing traffic transmission between them.

Top ISPs	Traffic (GB)
Fermi National Accelerator Laboratory (Fermilab)	2,156,964
ROSNIIROS Russian Institute for Public Networks	901,587
Joint Institute for Nuclear Research	758,761
National Electronics and Computer Technology Center	578,443
Kurchatov Institute	396,948
Vanderbilt University	301,955
Radio-fsu/radio-msu Network	259,128
ZAO MTU-Intel	244,256

The matrix visualization provides an intuitive insight into network usage by the number of bytes transmitted per each country. The horizontal axis represents the time well the intensity of the data transmitted is color graded from dark to the lighter versions of blue or green, for source and destination respectively, white represents the peak transmission intensity. The segmented pie chart next to the matrix visualization devices a compact technique to represent top ISP, domain ID and DNS by the amount of data transmitted. Finally as seen above the rightmost visualization provides additional information of the top ISPs by traffic transmission and the amount of traffic transmitted in gigabytes.

Top ISPs	Traffic (GB)
Jump Internet Services SRL	2,229,269
Joint Institute for Nuclear Research	2,120,748
	906,376
Radio-fsu/radio-msu Network	516,980
ZAO MTU-Intel	400,489
ROSNIIROS Russian Institute for Public Networks	312,147
National Electronics and Computer Technology Center	263,552
The University of Tennessee Health Science Center	236,138

The following visualizations provide advanced measurements about the network such as, Round Trip Time (RTT), Inter-packet delay, Jitter, Hops and Distribution of packet size by Maximum, Minimum and Average size of packets. These advanced visualisations are intended to be used by network administrators to gain advanced insight for debugging and finding potential network failures proactively. The producer consumer ratio (PCR) graph shows How the PCR change over time. A PCR of greater than or equal to 0.5 indicates that most of the network traffic was sent by the flow originator and vice versa. A value closer to 0 can indicate data exfiltration, and if observed for extended period of time should be investigated for malicious activity. The jitter graph indicates the quality of the network for the purpose of streaming services where it is preferred the package arrived at the destination at the same order they were transmitted at the source such as teleconferencing end voice over IP calls. During normal operation jitter should be kept at a minimum. High levels of jitter indicates volatile routing tables that change rapidly due to misconfiguration since packets take different routes during the do duration of the flow. The average number of hops is represented in the last visualization in a time series bar chart. higher number of hops indicate that majority of the communication over the network was between nodes geographically far apart. On the right hand side round trip time (RTT) graph is plotted against the time. Higher RTT can indicate network congestion or simply the fact that most nodes communicating are joined by slower links or excessive number of hops. The inter packet arrival time graph provides insight into the quality of the network. Lower interpacket arrival times are preferable for faster response for applications that rely on reliable transportation protocols such as transmission control protocol (TCP). The final graph represents packet size information sub divided by maximum, minimum average size observed. Large amounts of small packets in conjunction with large amounts of package transmitted shown in the top row of gauges can indicate denial-of-service attacks.

In this final visualization predicted network usage is plotted for the next 24 hours by analyzing the previously observed traffic of 7 days within the network. This visualization is useful to identify potential network bottlenecks that might occur within the next day. An application of this visualization is to determine whether large-scale data transfer should be started or not add particular time of the day next day. Time intervals where predicted network usage is lower is preferable for large-scale data transfer which will reduce the probability of network overload. Note that this information is predicted usage using the technique Markov Chains which builds a chain of network states into the future by observing the past. And it should be used with caution as is the case with any prediction technology.

Figure 5.9 Performance Dashboard

Geographical information is represented in the heat map indicated to the right. The color changes from green to red indicating the regions of any color graded format. Controls on the map to the left allows to zoom the map to data bounds and apply filters by drawing rectangle on the geographical region of interest. The next visualizations provide the top organizations that sent data through the network, both by visual representation and by text. More information can be gathered by hovering mouse over each segment and filters can be applied by clicking segments in the pie chart or clicking the plus icon in the table to the right, which shows traffic counts in gigabytes per organization. Following dashboards visualize the information aggregated by destination country. The visualizations are the same except for they represent two floor receiving end. The geographical map also supports double click to zoom in. The counters on the left hand side show the unique number of destination countries, cities, organizations and IP addresses. Note that this information is related to the time frame selected either by default or by user by clicking and dragging the time series graph on the top. On the right hand side the scrollable list of organizations are sorted by the amount of the year they have transmitted and at the end of the list are two links to download the raw data or the formatting data format.

This section of visualizations provides information related to Traffic and internet service providers (ISP). Two sets of counters are shown on the left that all related to ISP domain and domain name servers (DNS servers). Unique counts seen in the time interval selected is displayed for both flow originator and flow destination. As seen throughout the dashboard blue or shades of blue corresponds to flow originator and green for flow destination. Below the counters a network of the organizations that sent and received the most traffic is shown with the edges representing traffic transmission between them.

The matrix visualization provides an intuitive insight into network usage by the number of bytes transmitted per each country. The horizontal axis represents the time well the intensity of the data transmitted is color graded from dark to the lighter versions of blue or green, for source and destination respectively, white represents the peak transmission intensity. The segmented pie chart next to the matrix visualization devices a compact technique to represent top ISP, domain ID and DNS by the amount of data transmitted. Finally, as seen above the rightmost visualization provides additional information of the top ISPs by traffic transmission and the amount of traffic transmitted in gigabytes.

The following visualizations provide advanced measurements about the network such as, Round Trip Time (RTT), Inter-packet delay, Jitter, Hops and Distribution of packet size by Maximum, Minimum and Average size of packets. These advanced visualizations are intended to be used by network administrators to gain advanced insight for debugging and finding potential network failures proactively. The producer consumer ratio (PCR) graph shows how the PCR changes over time. A PCR of greater than or equal to 0.5 indicates that most of the network traffic was sent by the flow originator and vice versa. A value closer to 0 can indicate data exfiltration, and if observed for extended period of time should be investigated for malicious activity. The jitter graph indicates the quality of the network for the purpose of streaming services where it is preferred the package arrived at the destination at the same order they were transmitted at the source such as teleconferencing end voice over IP calls [32]. During normal operation jitter should be kept at a minimum. High levels of jitter indicate volatile routing tables that change rapidly due to misconfiguration since packets take different routes during the duration of the flow. The average number of hops is represented in the last visualization in a time series bar chart. On the right hand side round trip time (RTT) graph is plotted against the time. Higher RTT can indicate network congestion or simply the fact that most nodes communicating are joined by slower links or excessive number of hops. The inter packet arrival time graph provides insight into the quality of the network. Lower inter-packet arrival times are preferable for faster response for applications that rely on reliable transportation protocols such as transmission control protocol (TCP). The final graph represents packet size information sub divided by maximum, minimum average size observed. Large amounts of small packets in conjunction with large amounts of package transmitted shown in the top row of gauges can indicate denial-of-service attacks.

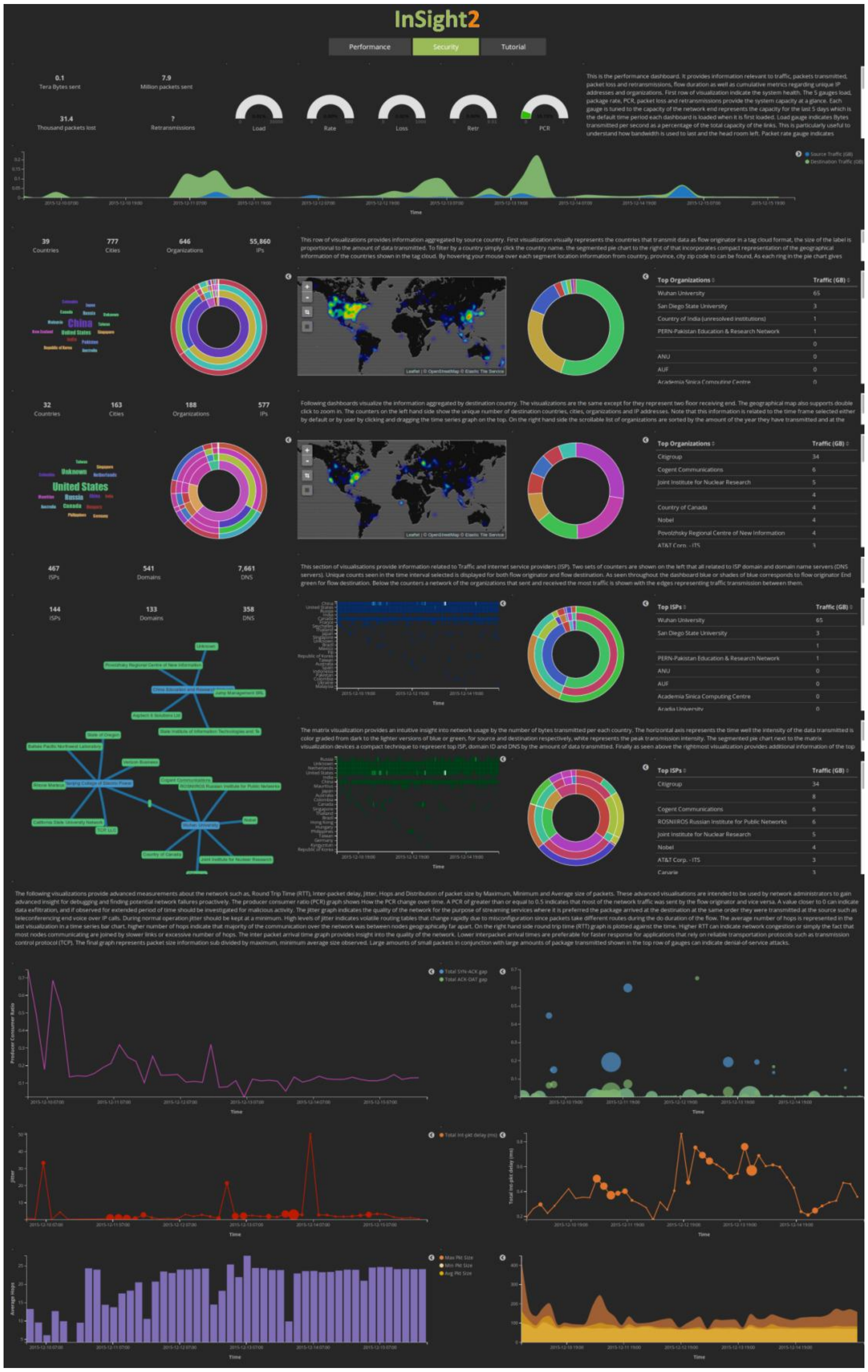
5.5.2 Security Dashboard

Security dashboard provides visualizations related to security aspects of the network. It shows the traffic patterns and network utilization of the malicious users and nodes. The information taken from the Security database is collected from different websites that keeping up-to-date lists of malicious IP addresses is integrated into the flow records. This dashboard pulls this information from the Elasticsearch database and displays it in relevant visualization components.

Layout of these individual components adheres to The Standard design philosophy incorporated in the performance dashboards.

Denial-of-service attacks (DoS) utilize half open connections to degrade or stop the functionality of a web service [33]. A half open connection is a connection that is not fully established. TCP requires 4 way handshake in order to set up a connection between two nodes. First the source will send a connection request to the destination using a SYN packet. When the destination receives this packet it will examine if it is a valid request. If it is then it will send a reply with SYN and ACK flags set. If the port the sender is trying to access is closed, the connection request packet is invalid or there is a firewall blocking connections the center will receive a RST packet.

In the case of DoS attacks SYN packets are sent by the sender, in the scale of millions of packets usually using spoofed IP addresses, and when the receiver sends back SYN and ACK (also known as SYNACK) the attacker will not respond causing the server to hang waiting for the completion of the handshake. Each connection consumes resources at the server-side. When such numerous connections are kept half-open the server will eventually run out of resources causing a DoS. Since metrics are collected about the round trip time which consists of SYN → ACK (the first half of the TCP connection) duration and ACK → DAT (the second half of the TCP connection) we can identify the unique patterns visually from the round trip time visualization. Figure 5.10 shows the security dashboard.



0.1 Tera Bytes sent

7.9 Million packets sent

31.4 Thousand packets lost

? Retransmissions

This is the performance dashboard. It provides information relevant to traffic, packets transmitted, packet loss and retransmissions, flow duration as well as cumulative metrics regarding unique IP addresses and organizations. First row of visualization indicate the system health. The 5 gauges load, package rate, PCR, packet loss and retransmissions provide the system capacity at a glance. Each gauge is tuned to the capacity of the network end represents the capacity for the last 5 days which is the default time period each dashboard is loaded when it is first loaded. Load gauge indicates Bytes transmitted per second as a percentage of the total capacity of the links. This is particularly useful to understand how bandwidth is used to last and the head room left. Packet rate gauge indicates

39 Countries | 777 Cities | 646 Organizations | 55,860 IPs

This row of visualizations provides information aggregated by source country. First visualization visually represents the countries that transmit data as flow originator in a tag cloud format, the size of the label is proportional to the amount of data transmitted. To filter by a country simply click the country name. the segmented pie chart to the right of that incorporates compact representation of the geographical information of the countries shown in the tag cloud. By hovering your mouse over each segment location information from country, province, city zip code to can be found. As each ring in the pie chart gives

Top Organizations	Traffic (GB)
Wuhan University	65
San Diego State University	3
Country of India (unresolved institutions)	1
PERN-Pakistan Education & Research Network	1
	0
ANU	0
AUF	0
Academia Sinica Computing Centre	0

32 Countries | 163 Cities | 188 Organizations | 577 IPs

Following dashboards visualize the information aggregated by destination country. The visualizations are the same except for they represent two floor receiving end. The geographical map also supports double click to zoom in. The counters on the left hand side show the unique number of destination countries, cities, organizations and IP addresses. Note that this information is related to the time frame selected either by default or by user by clicking and dragging the time series graph on the top. On the right hand side the scrollable list of organizations are sorted by the amount of the year they have transmitted and at the

Top Organizations	Traffic (GB)
Citigroup	34
Cogent Communications	6
Joint Institute for Nuclear Research	5
	4
Country of Canada	4
Nobel	4
Povolzhsky Regional Centre of New Information	4
AT&T Corp. - ITS	3

467 ISPs | 541 Domains | 7,661 DNS

This section of visualisations provide information related to Traffic and internet service providers (ISP). Two sets of counters are shown on the left that all related to ISP domain and domain name servers (DNS servers). Unique counts seen in the time interval selected is displayed for both flow originator and flow destination. As seen throughout the dashboard blue or shades of blue corresponds to flow originator End green for flow destination. Below the counters a network of the organizations that sent and received the most traffic is shown with the edges representing traffic transmission between them.

144 ISPs | 133 Domains | 358 DNS

Top ISPs	Traffic (GB)
Wuhan University	65
San Diego State University	3
	1
PERN-Pakistan Education & Research Network	1
ANU	0
AUF	0
Academia Sinica Computing Centre	0
Acadia University	0

The matrix visualization provides an intuitive insight into network usage by the number of bytes transmitted per each country. The horizontal axis represents the time well the intensity of the data transmitted is color graded from dark to the lighter versions of blue or green, for source and destination respectively, white represents the peak transmission intensity. The segmented pie chart next to the matrix visualization devices a compact technique to represent top ISP, domain ID and DNS by the amount of data transmitted. Finally as seen above the rightmost visualization provides additional information of the top

Top ISPs	Traffic (GB)
Citigroup	34
	8
Cogent Communications	6
ROSNIIROS Russian Institute for Public Networks	6
Joint Institute for Nuclear Research	5
Nobel	4
AT&T Corp. - ITS	3
Canarie	3

The following visualizations provide advanced measurements about the network such as, Round Trip Time (RTT), Inter-packet delay, Jitter, Hops and Distribution of packet size by Maximum, Minimum and Average size of packets. These advanced visualisations are intended to be used by network administrators to gain advanced insight for debugging and finding potential network failures proactively. The producer consumer ratio (PCR) graph shows How the PCR change over time. A PCR of greater than or equal to 0.5 indicates that most of the network traffic was sent by the flow orignator and vice versa. A value closer to 0 can indicate data exfiltration, and if observed for extended period of time should be investigated for malicious activity. The jitter graph indicates the quality of the network for the purpose of streaming services where it is preferred the package arrived at the destination at the same order they were transmitted at the source such as teleconferencing end voice over IP calls. During normal operation jitter should be kept at a minimum. High levels of jitter indicates volatile routing tables that change rapidly due to misconfiguration since packets take different routes during the do duration of the flow. The average number of hops is represented in the last visualization in a time series bar chart. higher number of hops indicate that majority of the communication over the network was between nodes geographically far apart. On the right hand side round trip time (RTT) graph is plotted against the time. Higher RTT can indicate network congestion or simply the fact that most nodes communicating are joined by slower links or excessive number of hops. The inter packet arrival time graph provides insight into the quality of the network. Lower interpacket arrival times are preferable for faster response for applications that rely on reliable transportation protocols such as transmission control protocol (TCP). The final graph represents packet size information sub divided by maximum, minimum average size observed. Large amounts of small packets in conjunction with large amounts of package transmitted shown in the top row of gauges can indicate denial-of-service attacks.

Figure 5.10 Security Dashboard

5.5.3 Dashboard Components

The following screenshots display the various components of the user interface of InSight2. The visualizations show source traffic in blue and destination traffic on green. First visualization shown in the Figure 5.11 shows the inbound traffic by green and outbound traffic by blue.

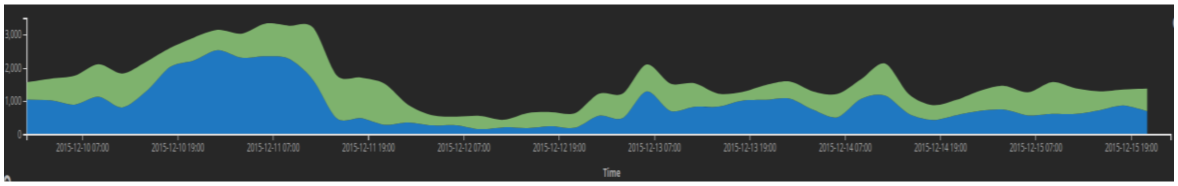


Figure 5.11 Network Usage by Data Transmitted

Figure 5.12 shows the gauges that represent the network status at a given time. These are useful to identify quickly at a glance useful and critical metrics about the network such as network load in bits per second, packet rate in packets per second, packet loss in packets lost per second, retransmissions of packets in packets retransmitted per second, and producer consumer ratio (PCR) which shows the percentage of data transmitted by the producer of the network flow to the receiver. PCR is particularly useful to understand how the average network flows behaves in the network by showing their bias of the amount of data transmitted from either the flow initiator or the flow receiver.

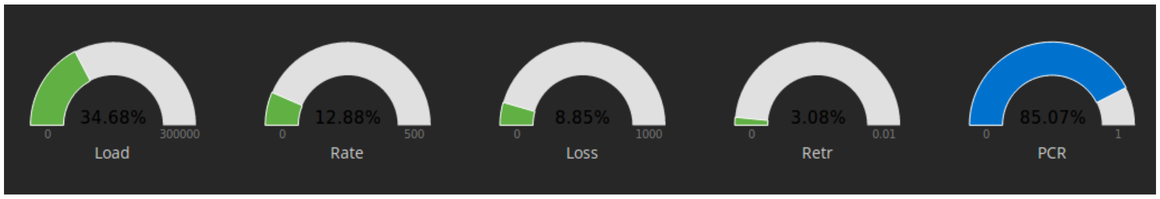
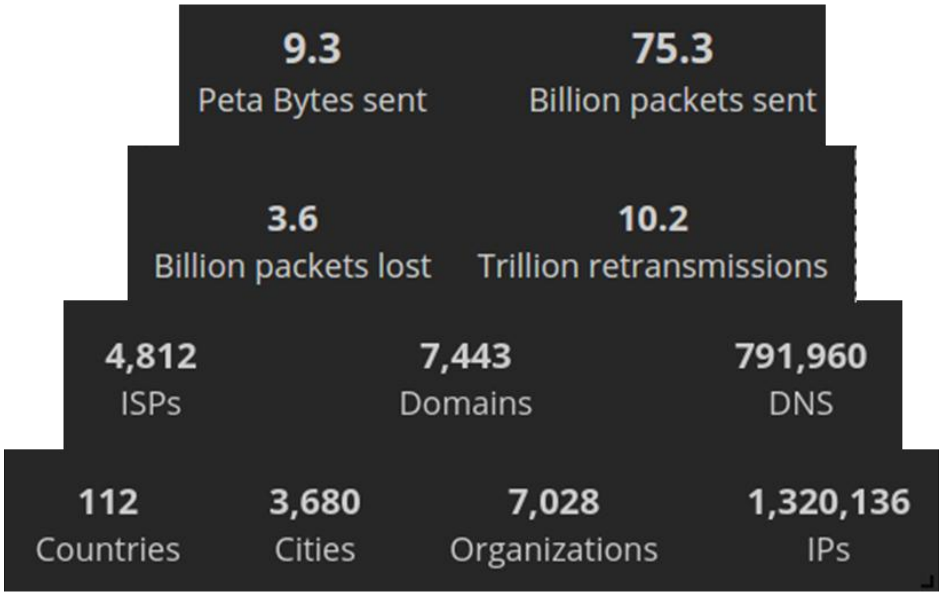


Figure 5.12 At-a-glance Gauges

The set of visualizations shown in the figure 5.13, provide aggregated counts of various metrics associated with the end nodes or the transmission of the packets. These metrics show Matrix using appropriate scales. For example instead of showing the raw number of bytes transmitted in the network it is configured to show petabytes, billions of packets and trillions of transmissions of flows. Unique accounts of different aspects of the end nodes such as unique number of internet service providers (ISP), domains, domain name servers (DNS), countries, city, organizations, and the IP addresses seen in the network.



9.3
Peta Bytes sent

75.3
Billion packets sent

3.6
Billion packets lost

10.2
Trillion retransmissions

4,812
ISPs

7,443
Domains

791,960
DNS

112
Countries

3,680
Cities

7,028
Organizations

1,320,136
IPs

Figure 5.13 Aggregated Unique Counts

Geographical information is represented in an interactive global map as shown in the Figures 5.14 to 5.16. Two maps are provided for source IP addresses and destination IP addresses that shows the intensity of the number of flows originated or ended using the geographical information provided by the MaxMind GeoIP. This information is accurate to the 50 kilometer radius. Compared to GLORIAD InSight, InSight2 is capable of showing the exact location of the IP address instead of assigning intensity colors to each of the regions of the US map or country-wise for global map. Controls are provided to zoom into or out of the map, fit the map to the data bounds, geographical region selectors to defined filters to isolate flows originating from or ending to specific geographical area.

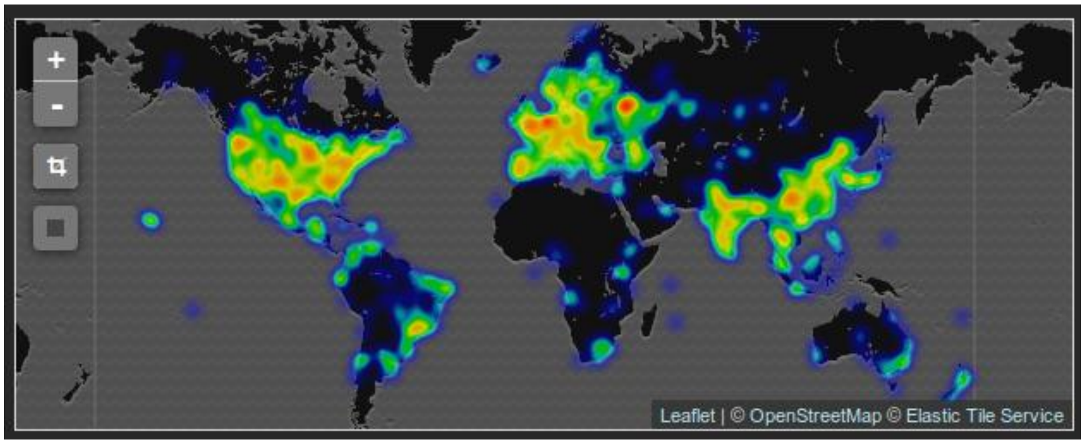


Figure 5.14 Geographical Information Globe

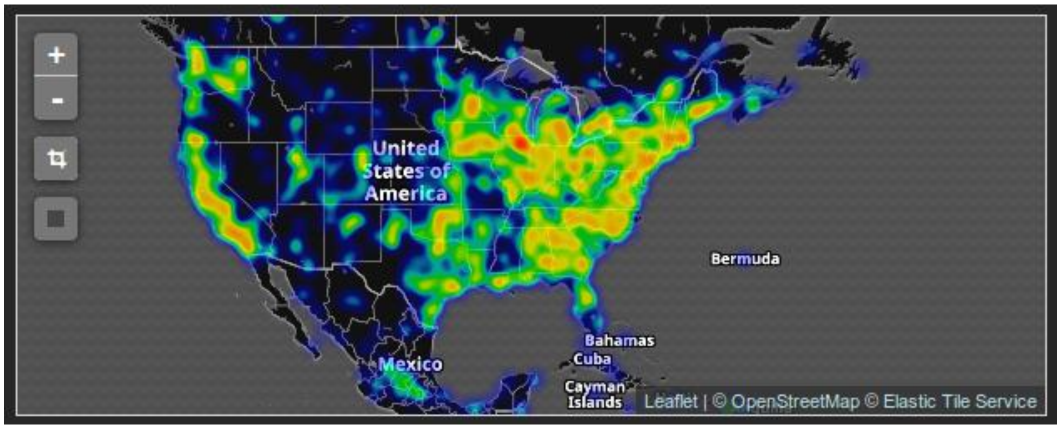


Figure 5.15 Geographical Information – Country

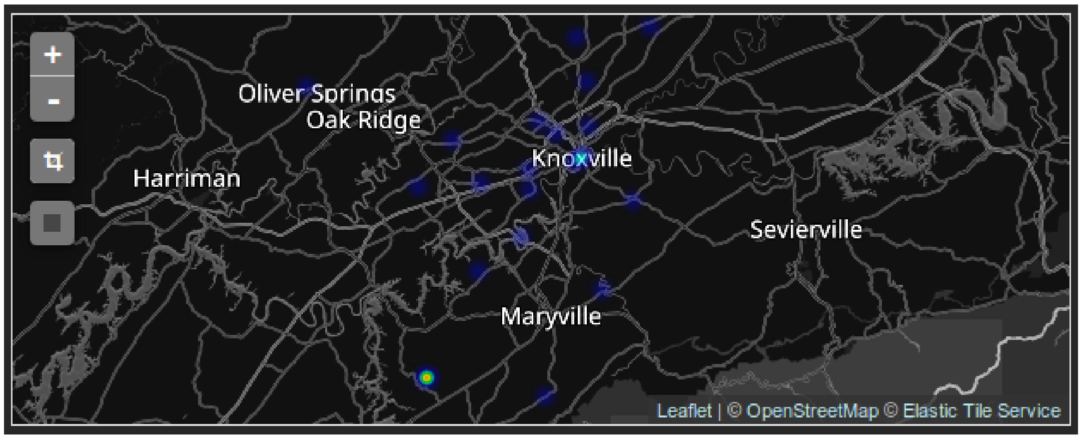


Figure 5.16 Geographical Information – City

Intuitive tag clouds shown in the represent the top countries that use the network shown in the Figure 5.17 to 5.18. Size of the font of these tag clouds represent the number of bites transmitted or received. Filters can be applied by clicking on one of the filters to filter the flow source or destination for a specific country. For example, by clicking United States as the source country and Germany as the destination country will define flows that originated from the United States and ended in Germany.

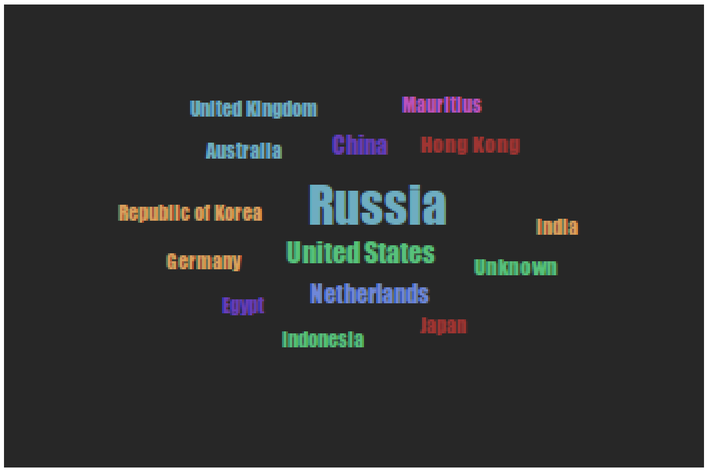
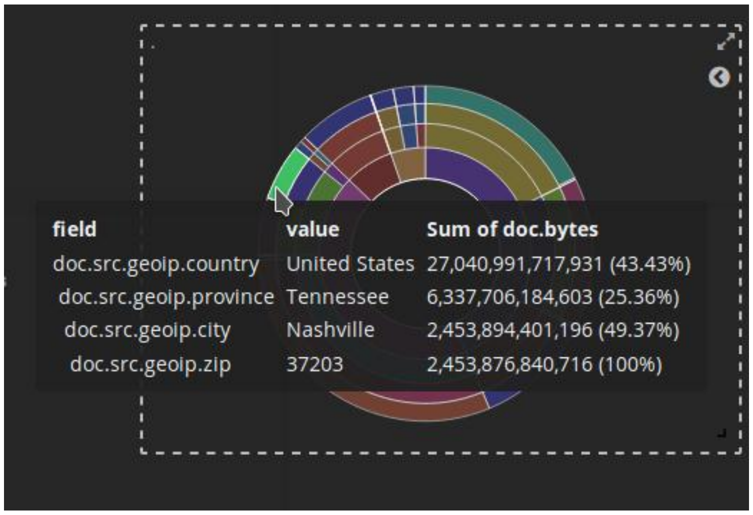


Figure 5.17 Tag Cloud of Top Users

Figure 5.18 Tag Cloud After Selection

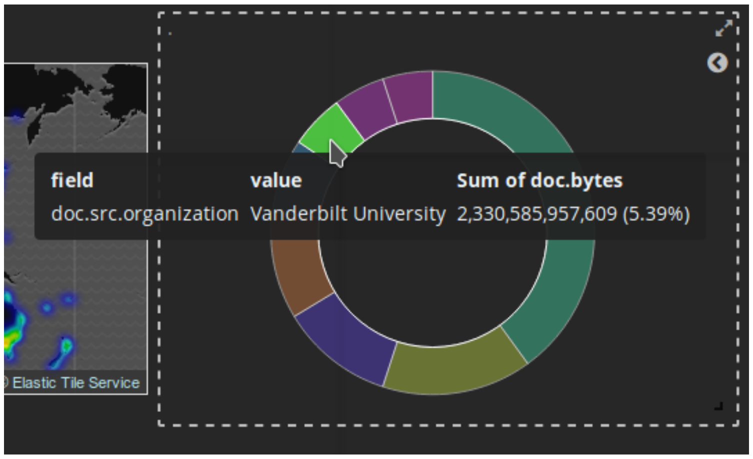
Geographical information such as country, province, city, and zip codes are provided in a segmented pie chart. Hovering the mouse over these charts reveals more information as seen in the Figure 5.19.



field	value	Sum of doc.bytes
doc.src.geoip.country	United States	27,040,991,717,931 (43.43%)
doc.src.geoip.province	Tennessee	6,337,706,184,603 (25.36%)
doc.src.geoip.city	Nashville	2,453,894,401,196 (49.37%)
doc.src.geoip.zip	37203	2,453,876,840,716 (100%)

Figure 5.19 Geographical Information of Top Users

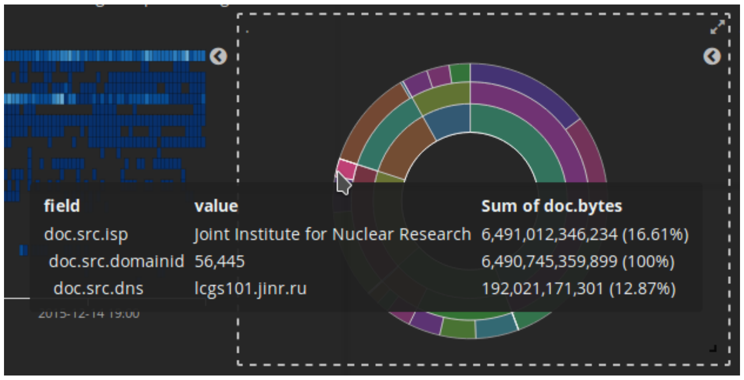
Organizational information is provided in the segmented pie chart shown in the Figure 5.20. Using the same technique as before hovering the mouse over each segment reveals institution or organization name number of bytes transmitted or received depending on the section the pie chart is located and the percentage of the total traffic that the specific institution or organization utilized in the network.



field	value	Sum of doc.bytes
doc.src.organization	Vanderbilt University	2,330,585,957,609 (5.39%)

Figure 5.20 Organizational Information

Domain information is provided in a similar manner to the geographical information. Figure 5.21 shows this domain information which consists of the ISP, domainID, and the domain name server that was used.



field	value	Sum of doc.bytes
doc.src.isp	Joint Institute for Nuclear Research	6,491,012,346,234 (16.61%)
doc.src.domainid	56,445	6,490,745,359,899 (100%)
doc.src.dns	lcgs101.jinr.ru	192,021,171,301 (12.87%)

Figure 5.21 Domain Information

Figure 5.22 and 5.23 show the usage of the network divided by each country at a given time interval. This time interval can be defined by the filters applied in the first visualization or by making a selection initiating from an empty area in these matrix. Interesting information can be revealed by the gradients of the color change observed within the row of the country. In the example shown source and destination matrix graphs are stacked together to show the corresponding traffic patterns. As to design philosophy utilized throughout the dashboard blue represents the source while green represents destination. The sum of all the traffic observed in the source matrix is equal to that of the destination matrix. When a country is selected this visualization changes its format to accommodate a single country as seen in the Figure 5.24.

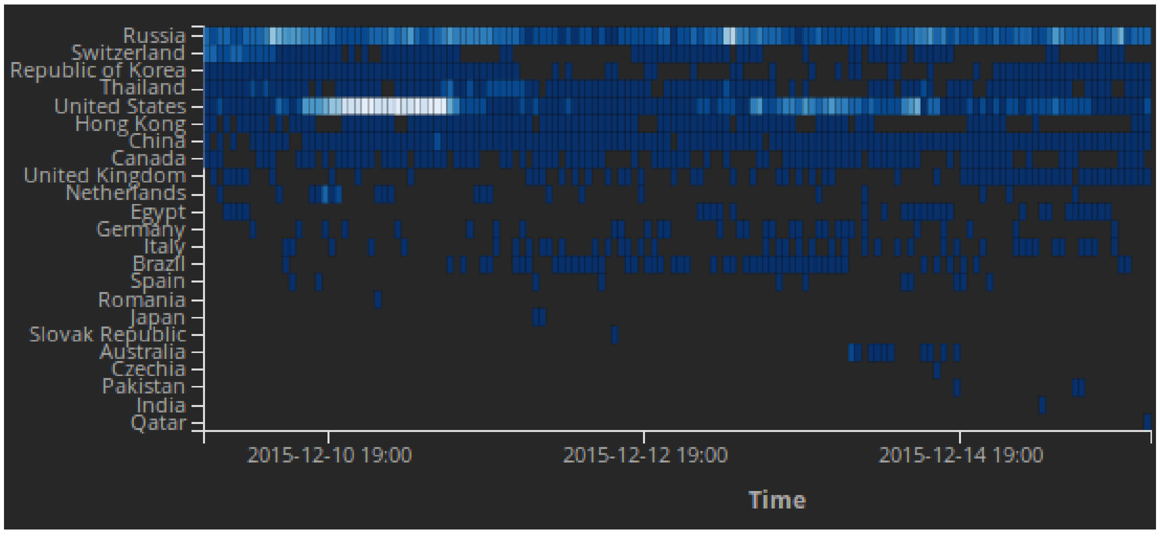


Figure 5.22 Usage by Source Country by Time

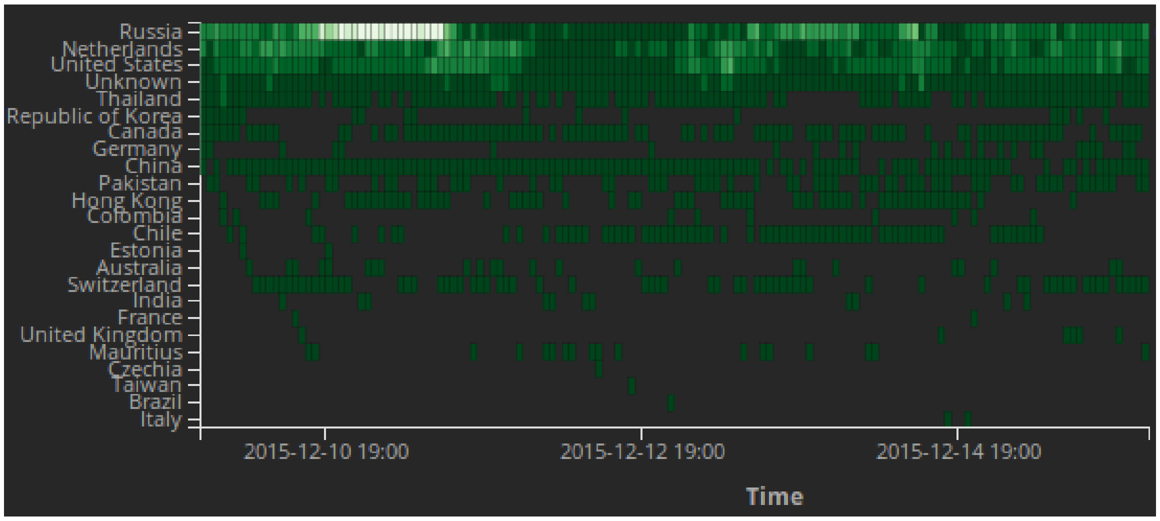


Figure 5.23 Usage by Destination Country by Time

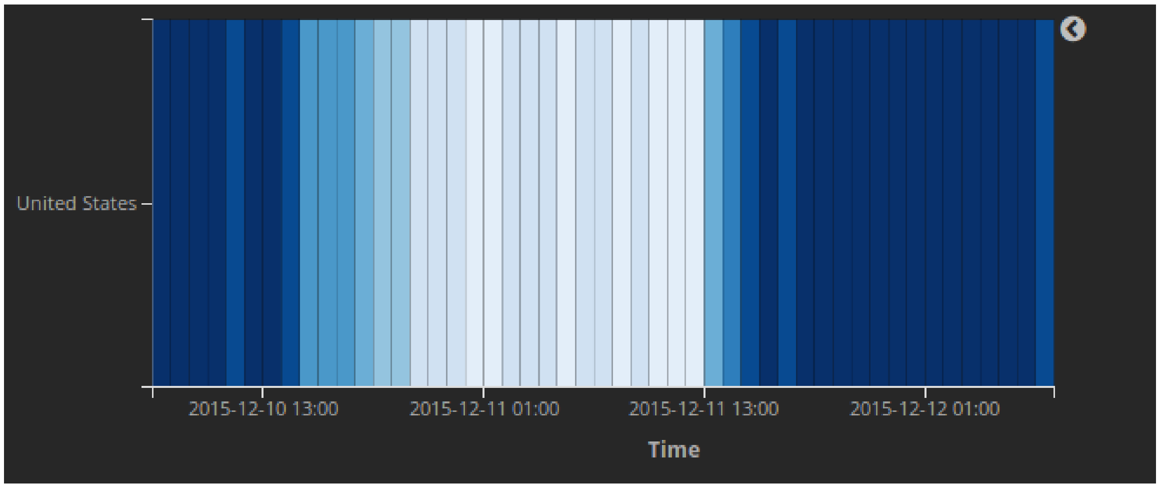


Figure 5.24 Usage by Country by Time After Selection

Network graph shown in the Figure 5.25 displays the connections between the organizations and follows the same common design principle with blue as source and green as the destination. Nodes are spread apart to prevent overlapping. Entities in the graph can be moved by dragging and dropping. Zooming with mouse wheel is allowed which helps to better see the connections and names of the organizations.

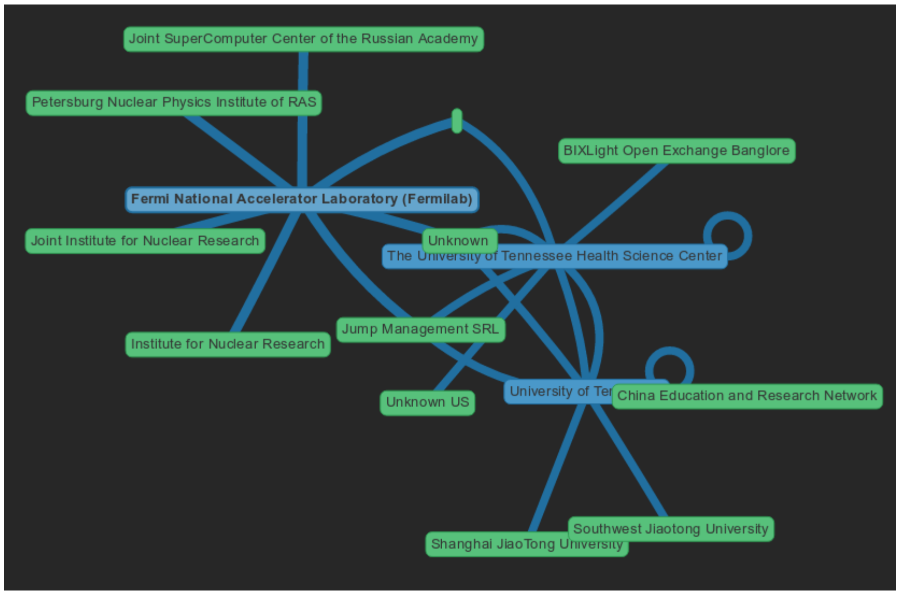

Joint SuperComputer Center of the Russian Academy
Petersburg Nuclear Physics Institute of RAS
BIXLight Open Exchange Banglore
Fermi National Accelerator Laboratory (Fermilab)
Joint Institute for Nuclear Research
Unknown
The University of Tennessee Health Science Center
Jump Management SRL
Institute for Nuclear Research
Unknown US
University of Te
China Education and Research Network
Southwest Jiaotong University
Shanghai JiaoTong University

Figure 5.25 Network Connection Graph of Organizations

Total inter packet delay is associated with a host of network quality issues. The average time between two adjacent packets increase with network congestion. Figure 5.26 shows the inter-packet delay by a line graph as well as total network bandwidth utilization by the size of each dot. This is particularly useful to identify the network speed as compared to the bandwidth. Network speed or the network latency is a measure of how fast a packet can be sent from one end of the network to another. Inter-packet delay and bandwidth utilization has some relation since at the upper balance of the bandwidth available congestion starts and inter-packet delay increases.

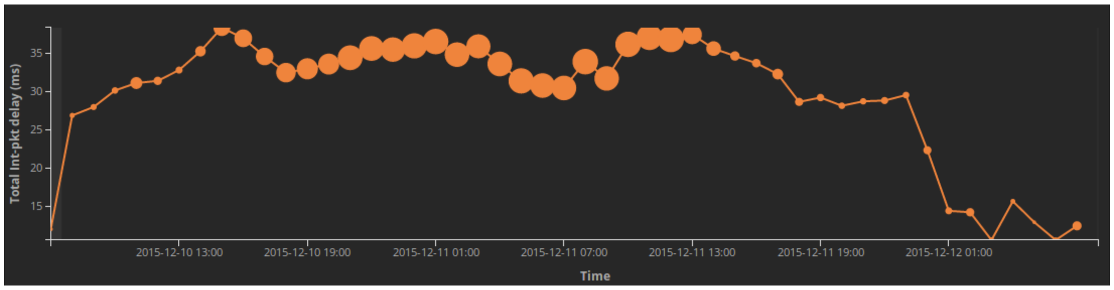


Figure 5.26 Total Packet Delay

Figure 5.27 shows connection set up time. TCP connection time is defined as a 4-way handshake. First half of the connection is center initiating connection with the receiver and the second half of the connection is the vice versa. Using the standard color scheme blue dots represents the sender setting up the connection which is derived by the time between the SYN and ACK packets. Second half of the connections depicted by the blue dots are calculated by time gap between ACK and SYNACK packets.

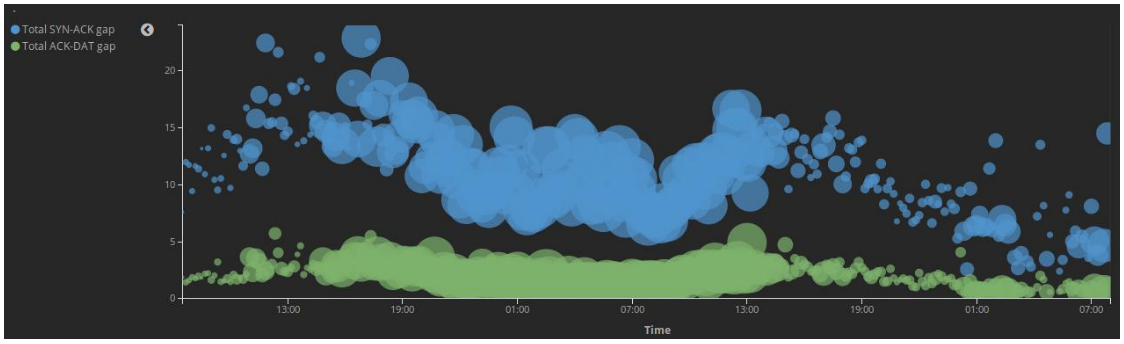


Figure 5.27 Connection Setup Time Gap

For real-time tasks such as video streaming and teleconferencing jitter plays a vital role. Jitter is the amount of package arriving out of order. TCP sequence numbers are used to ensure the TCP flows are create constructed at the receiver in the same order they were sent add the source. UDP does not have sequence numbers built-in however application-specific protocols may define their internal sequence numbers native to the application. Figure 5.28 visualizes jitter within the network. This is taken into account the number of packets arriving out of order in the TCP sequence.

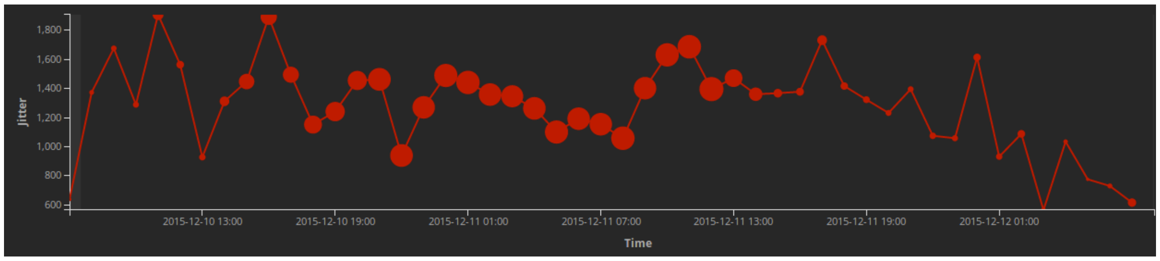


Figure 5.28 InSight2 Jitter

Producer consumer ratio (PCR) is also graphed over time as seen in the Figure 5.29 showing the variation of bias of the amount of data sent from source to destination and vice versa. This is especially useful for systems that use network flow information since Argus flows are bidirectional and PCR is needed to understand the percentage of data flow at each direction.

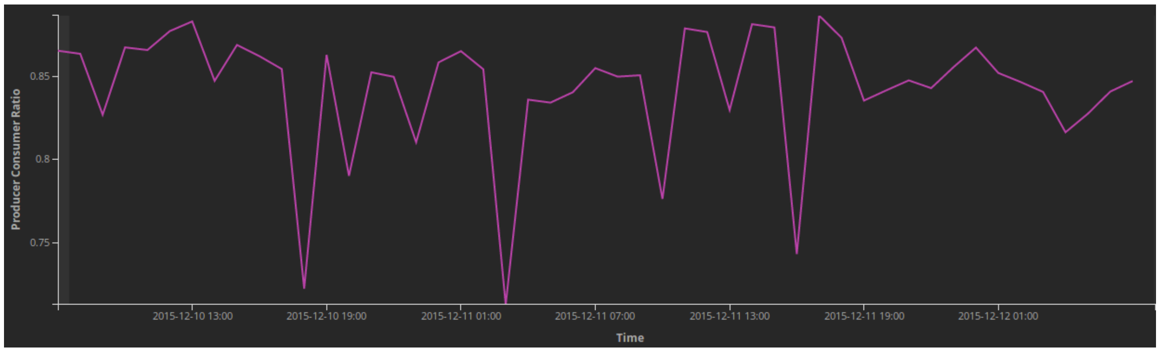


Figure 5.29 Producer Consumer Ratio

The average number of hops is a metric of the distance between two end nodes. High amount of hops can lead to network latency. The visualization provided in the figure 5.30 shows the average number of hops observed within each flow within that specific time window.

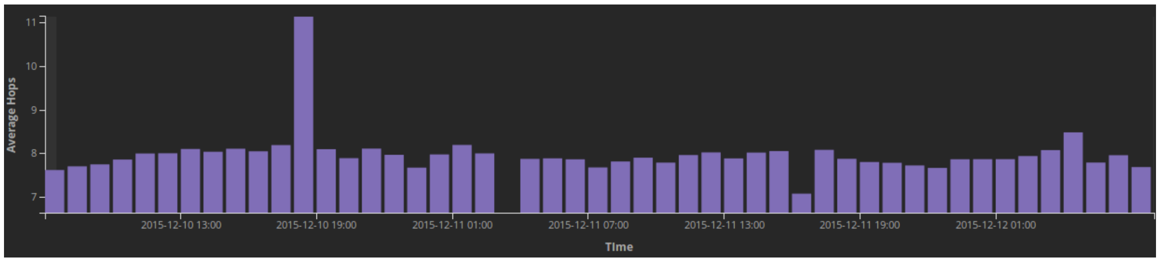
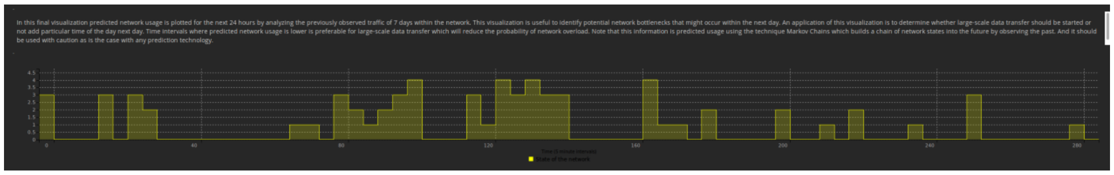


Figure 5.30 Average Number of Hops

Predicted network usage for the next 24 hours is calculated using Markov Chains discussed in the section 5.4 and shown in the Figure 5.31



In this final visualization predicted network usage is plotted for the next 24 hours by analyzing the previously observed traffic of 7 days within the network. This visualization is useful to identify potential network bottlenecks that might occur within the next day. An application of this visualization is to determine whether large-scale data transfer should be started or not add particular time of the day next day. Time intervals where predicted network usage is lower is preferable for large-scale data transfer which will reduce the probability of network overload. Note that this information is predicted usage using the technique Markov Chains which builds a chain of network states into the future by observing the past. And it should be used with caution as is the case with any prediction technology.

Figure 5.31 Network Load Prediction

5.6 InSight2 Security Features

Hardware servers used to store the databases, run the EM, and host the InSight2 user interface website are located in the Cisco facility which allows physical security from unauthorized access. Remote access is granted through encrypted SSH connections. Authentication is performed using public key authentication and username and password authentication has been disabled to prevent Brute Force attacks, which involves trying multiple username and password combinations until the correct one is found. Since these servers are assigned public IP addresses it is paramount to mitigate outside attacks as much as possible. While there is no perfect security, limiting the login to secure protocols and requiring the remote clients to be set up per client basis raises the bar of the ease for attack. Any new clients that require administrative access require its public key stored in the respective server. The public key authentication uses strict 2048 bit keys using RSA public-key cryptosystem to generate public and private key pair.

The new user authentication system that is built into InSight2 incorporates PHP back-end that performs username and password acquisition, transmission, and the granting of access on the server side. It offers the highest level of security compared to client side code such as JavScript [34]. The InSight2 user-password credentials are kept separate and isolated from OS user credentials providing an extra layer of protection for the data in case of a security breach. Regular InSight2 users are prevented from accessing low-level system functions that can impact the performance and security if misused. For these functions special user, 'root', is used by elevating privileges from the standard user. This allows having separate user accounts with different scopes of visibility and clearance. Administrative functions are carried out by 'sudo' command which gives standard user temporarily elevated privileges.

Configuration of Elasticsearch and Kibana is optimized to make sure that bare minimum access is granted for the proper function of the platform. Kibana accesses the Elasticsearch database using an internal network socket. Kibana and Elasticsearch are hosted within the same server to reduce activity across the network. Elasticsearch database is distributed across all 3 servers 'Bulika', 'Lona', and 'Lilith' and are connected using a secure LAN connection.

Access to the dashboards is granted using built-in authentication system that uses PHP to perform the authentication server side. This mitigates a wide variety of attacks that can be launched in the client-side such as injection attacks into the JavaScript authentication mechanism that was previously experimented. PHP keeps separate database of users and passwords that are completely decoupled from the OS user database.

Tailor-made dashboards are loaded depending on the username and different time periods in the timeline are visualized. User authentication system is also used to define relevant dashboards and visualizations depending on the user and their privileges. For example when users with low privileges are logged into the system they are presented with dashboards which do not display specific IP addresses for privacy purposes. This segregates access privilege through a single authentication system. This allows the separation of system administrator and end-user. InSight2 is not only intended for System administrators and network operators but also users of the network. Thus it is important to have different privileges of access and data that is displayed. Usage of the platform by network users is further simplified by the descriptions provided and video tutorials embedded in the tutorials tab.

InSight2 web-interface extracts the dashboards from Kibana and displays it using iFrames which enables one-way communication with Kibana and the web interface. Users are not allowed to modify dashboards which prevent tampering unless they are logged into main Kibana application which allows administrative functions from modifying and/or creating dashboards, changing the mapping template, creating new scripted fields, starting or shutting down of Elasticsearch or Kibana. Tabs are used to display the names of dashboards and perform as links to them allowing in the user to switch between different dashboards with a single click. Dashboards are loaded in a read-only manner so the end-user can neither modify nor create new dashboards. Another advantage of using this technique is that every time user reloads the web browser or clicks on one of the navigation tabs, the dashboards are reset to their original state acting as a simple mechanism to clear all the filters. Figure 5.32 show the InSight2 Security features.

InSight2 is encrypted during transmission using transport layer security (TLS 1.2) preventing man-in-the-middle (MITM) attacks [35]. This greatly elevates the security of the platform compared to the lack of security features of GLORIAD InSight. The encryption combined with logging of IP address and time provides a comprehensive security package. The TLS security certificates are provided by the ‘Let's Encrypt’ free security certificate provider and are updated automatically using it client. ‘Let's Encrypt’ is an open certificate Authority (CA) that runs on behalf of the public. This service is provided by Internet Security Research Group (ISRG) [36]. Its client seamlessly integrates into the Apache2 web server providing encryption for each incoming connection. Figure 5.33 shows the login page of the InSight2 web-interface.

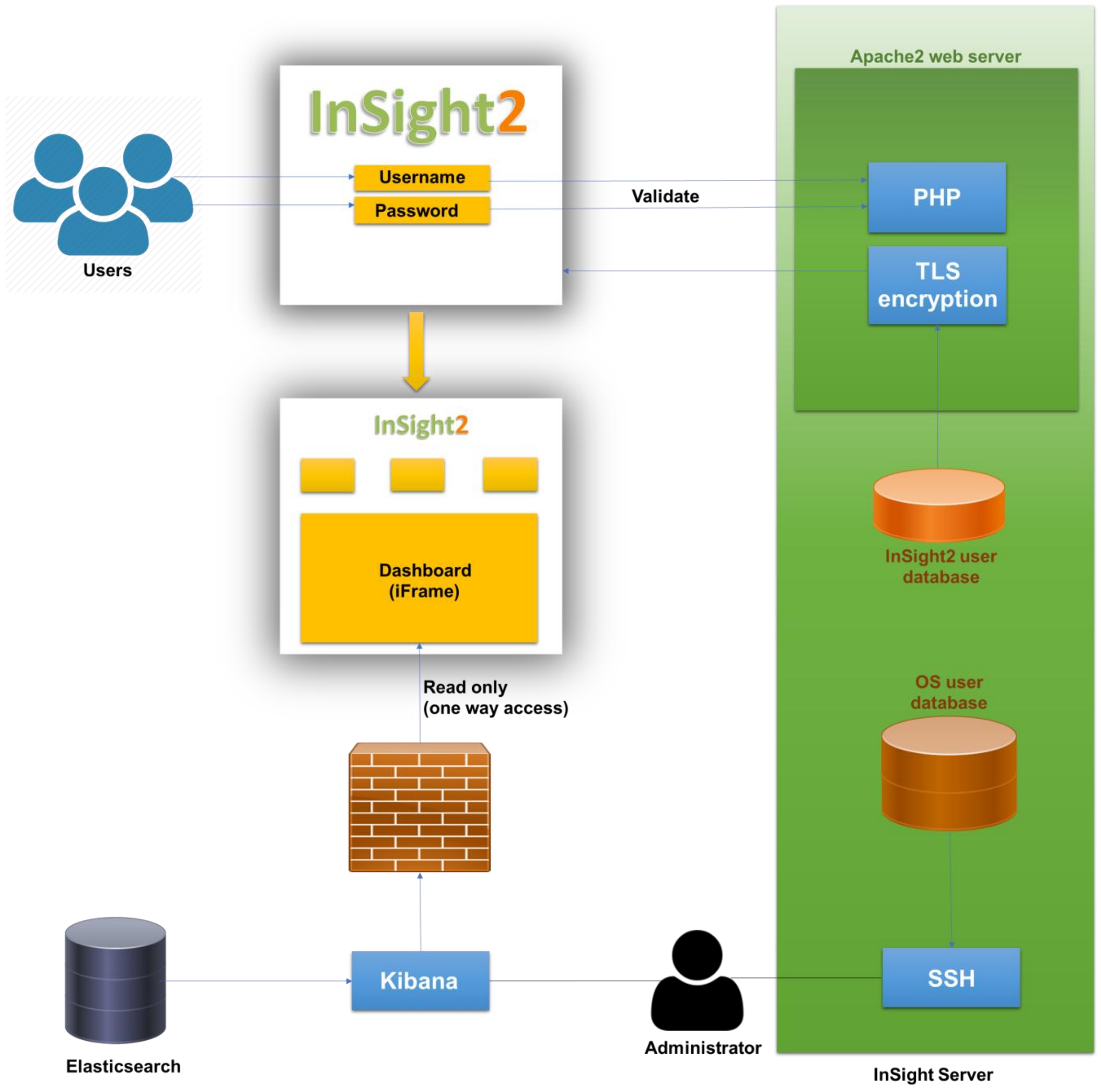
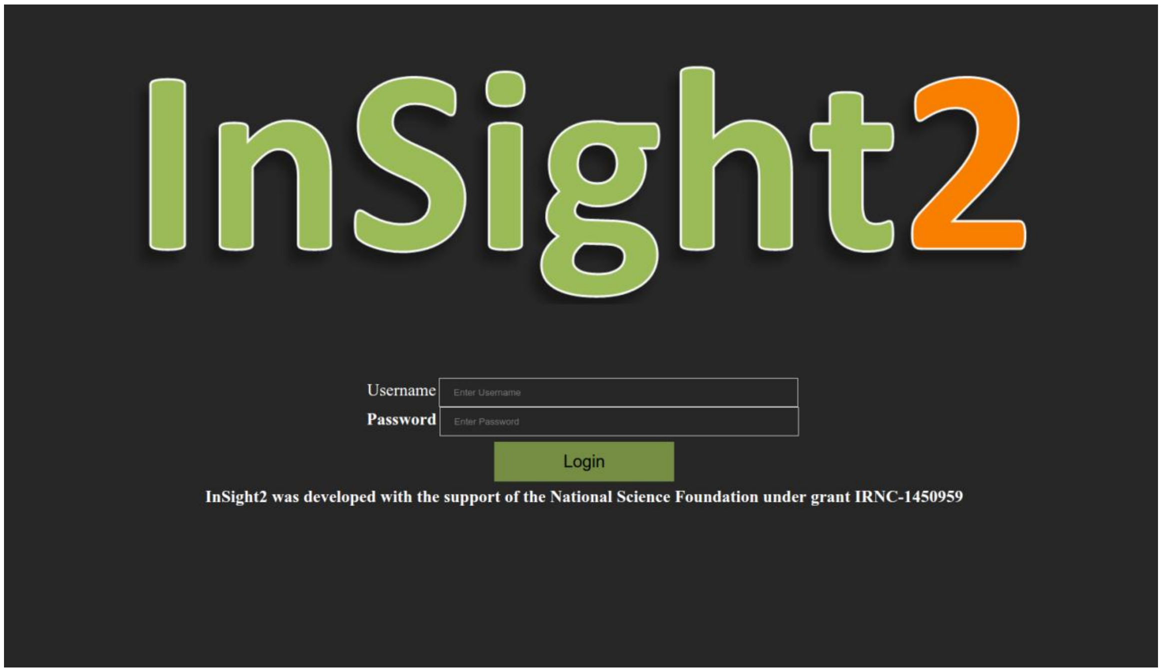


Figure 5.32 InSight2 Security Features



InSight2 was developed with the support of the National Science Foundation under grant IRNC-1450959

Figure 5.33 InSight2 Login Page

5.7 One-Step Installation Package

Ease of use of InSight2 starts from the installation itself. Unlike GLORIAD InSight, InSight2 comes with an installation package that includes all the software that is needed, a set of configuration files, a script that installs all the software and copies relevant configuration files to the right locations while setting the correct permissions and a sample of enriched data to get to use user up to speed with the platform and to test the installation. Essentially the InSight2 installation handles the complete installation of itself in a single step. GLORIAD InSight required 13 steps to install, with 152 user executed commands described in a 22 page installation guide. The new architecture also eliminates the need for ZFS file system on FreeBSD OS that required 27 steps to be set up. The installation script will probe the system for the following system requirements,

1. First the installation script checks for root privileges. This requirement is needed to install system-level modules copy the configuration files to system directories and set up permissions.

2. Internet connectivity for downloading and installing necessary software from the Linux repositories. This is an essential step and installation cannot proceed without this requirement.

3. Check the system for the following installed software. If any of them are not installed the installation script will install them automatically.

4. Install bundled versions of Elasticsearch and Kibana. This minimizes potential issues as the versions shipped with the installation package are tested rigorously for compatibility and optimal performance.

5. Set up the automatic start of the Linux demons at system level using “systemctl” command.

6. Copy the configuration files to necessarily locations and set up and verify correct permissions.

Parameters for the installation script are shown in the Table 5.3.

Table 5.3 Parameters for the InSight2 Installation Script

Name	TYPE	Default	Description
http.max_content_length	INTEGER	2000M	Maximum length of the bulk data transfer over http
path.data	STRING	/var/lib/ Elasticsearch	Path to the data storage by Elasticsearch
path.logs	STRING	/var/log/ Elasticsearch	Path to the logs storage by Elasticsearch
path.repo	STRING	["/tmp/gsr"]	Temporary extraction location of the GSR index
network.bind_host	IP	0.0.0.0	Binding host for the connection from Elasticsearch to Kibana
server.host	IP	0.0.0.0	Binding host for the connection from Kibana to Elasticsearch
jvm.options (max RAM)	INTEGER	-Xmx4g	Maximum memory allocation for Java virtual machine
jvm.options (min RAM)	INTEGER	-Xmx4g	Minimum memory allocation for Java virtual machine

6. RESULTS AND CONCLUSION

InSight2 is a powerful platform aimed at network performance measurements, security analytics and usage predictions. It presents user with in a modern and secure web-based user interface. It is capable of handling large volumes of Argus archives in near real-time enabling the network administrators of large-scale networks utilizing multi gigabit fiber optic links to be able to generate, collect, analyze, and visualize network related data in one convenient package. The multi-threaded feature enables InSight2 to be deployed in wide variety of platforms ranging from small scale off-the-shelf computers with limited processing and memory capacity to large-scale server farms where CPU cores and memory are abundant. It is fully capable of harnessing and utilizing all the CPU cores at its disposal to provide seamless user experience that is only rivaled by large scale paid software costing thousands of dollars.

Development of InSight2 was motivated by the lack of a platform geared towards an all-in-one network monitoring and analytics, built upon free and open source software. It resolves many of the drawbacks found in the network performance and security monitoring tools available today. It focuses on both network performance and security, provides time-critical predictions and recommendations into the future about network status with regard to bandwidth utilization, provides intuitive controls that give the end-user power of control to find the information quickly and intuitively in a couple of clicks rather than entering a series of commands, wait for them to be processed and displayed.

InSight2 achieves these feats through prudent selection of the network flow features from the Argus flow data, development of robust software architecture and the design and implementation of an intuitive and modern user interface. It provides rich information about the network that enables network administrators proactively determine network problems and security issues. Installation on a target system is automatic and a single step process.

InSight2 maximizes information per unit area in the user interface to include all the information found in the GLORIAD InSight and more by incorporating them into a fewer dashboards which negates the necessity for the user to switch into different dashboards to see different aspects of the network, eliminating the lag between page loads and the bandwidth usage to load the dashboards themselves. This makes InSight2 equally responsive under speedy WiFi as well as slower mobile data networks.

InSight2 user interface manages client resources efficiently. It reuses the already loaded dashboards to dynamically update according to the new filters reducing the strain on the client system, improving performance and lowering the general system requirements recommended to run the InSight2 user interface. InSight2 user interface does not need to be installed separately as it is completely web-based and can be accessed using any modern browser that has JavaScript enabled. Performance and security are divided into two dashboards giving relevant visualizations depending on the dashboard selected, since not all visualizations apply for both performance and security equally. This makes InSight2 run on desktops as well as mobile browsers.

The new architecture disposes of complex message passing between multitude of independently functioning scripts, to a more robust and lean architecture. InSight2 is a complete Python rewrite of the Perl implementation of GLORIAD InSight. Synchronous data processing as well as multi-threading is achieved regardless of the underlying platform since the Python interpreter is able to handle assignment of threads two different cores achieving true portability, as opposed to coding in low level programming languages such as C++.

Advanced network predictions provide the user to make informed decisions about status of the network in the near future. This allows the network administrator or the user to schedule large-scale data transfer to a time period where are the network utilization is predicted to be sustainably lower to achieve higher transmission speeds without congesting the network.

InSight2 is developed with modern design philosophy in mind adhering to one standard color scheme and layout to ease the steep learning curve of using this tool. The following software engineering best practices have been kept in mind:

Modularity: Argus server and its suite of clients are used for flow data generation, collection and filtering of the features. Data retrieval, enrichment and storage are performed using a single software module, EM. The Elastic Suite of tools is used to host the database, search for records using full text search engine and visualize enriched data using dashboards. Due to the modular nature of the architecture it is possible to swap any of these components with minimal change to achieve different functionality without breaking the entire system.

Fewer moving parts: InSight2 achieves more functionality than GLORIAD InSight with fewer dependencies. GLORIAD InSight components such as Perl POE, Monit and ZeroMQ are no longer necessary. The functionality of 'Farm of Animals' is now achieved using the EM, a multi-threaded, synchronous processing module that incorporates pipelined architecture written in Python.

Software and dependency management: Built-in software repository management system of the Linux OS has been used to install software tools and libraries instead of using programming language specific repositories such as Perl CPAN or compiling from source. When new versions of these software gets released a simple upgrade of the OS updates all the components these system level modules as well. Since these modules are updated with backwards compatibility in mind updates will not cause system instability. InSight2 servers use mainstream Debian derived latest Ubuntu Linux OS for greater compatibility.

Extendibility: For the scope of this research the InSight2 system is optimized for network flow data. The base architecture has been designed with generic inputs in mind so that not only network flow data but also other data can also be represented by the InSight2 system with minimal change.

Scalability: InSight2 can utilize the full potential of multicore processing and Elasticsearch provides search results that scale well with big data. InSight2 offloads the functionality to Elasticsearch instead of keeping all data in sequential databases.

Security: SSL encryption is used for encryption of the web interface during transit from the web server to client browser. Username and password authentication is implemented on the server-side to authorize users. Linux OS users and InSight2 users are segregated by keeping separate user/password database for InSight2 authentication system. Dashboards are extracted using iFrames preventing unauthorized users from modifying dashboards. Elasticsearch and Kibana configuration is optimized for best security practices. 'iptables' firewall was setup to harden the InSight2 server.

Ease of deployment: Convenient installation package for ease of deployment that includes all the prerequisites which will install InSight2 in just one step in systems that meet at least the minimum requirements.

Robustness: New server setup provides robust environment for InSight2, RESTful API enables standard communication. The new RESTful API introduced with Elastic suite of tools version 5.3 uses industry standard API allowing it to function well with other software. Robust installation script that checks system requirements before installation makes sure that target system meets the prerequisites, installs necessary packages automatically and ships versions of software that is tested to work well.

Stability: By utilizing Ubuntu Linux InSight2 inherits the stability compared to FreeBSD. In Ubuntu Linux software components are not always updated to the latest version and instead they are held back until they are considered to be stable. This greatly minimizes the possibility that a system upgrade will update system software to versions that are not compatible with each other.

Speed: GSR is kept in the Elasticsearch itself for faster search and InSight2 is optimized for multithreaded environments.

Documentation: An installation guide, a user guide, and a video tutorial are provided.

7. FUTURE WORK

Planned road map for the future development includes adapting the system to process live Argus data. This is proposed to be performed by either collecting data from Argus Radium server, reading directly from network span port using Argus server or Cisco NetFlow source as input to the Argus server to generate the flow data. With feedback from network operators at Stanford University and Korea Institute of Science and Technology Information (KISTI) new dashboards will be added to the system which will extend the functionality of the platform. The current Markov Chains prediction system will be expanded to other technologies such as machine learning and deep learning in order to perform advanced prediction on a variety of metrics. The Enrichment Module is planned to be extended to the other two servers in order to utilize their processing power as well, which will result in distributed multi-server, multi-core processing.

InSight2 can be incorporated into a Docker container further streamlining the installation procedure. Docker containers allows software to use the system resources without having to install software. This is greatly beneficial to move the platform from one server to another as well as for new deployments since the Docker container is a self-contained package.

REFERENCES

[1] Cole, Greg, and Natasha Bulashova. "GLORIAD: a ring around the Northern Hemisphere for science and education connecting North America, Russia, China, Korea and Netherlands with advanced network services", 2005.

[2] “Global Optical Ring Network for Advanced Applications Development – logo”. http://gloriad.org. Retrieved 1 July 2017

[3] Valerii A.Vasenin, Moscow State University. “High Performance Research and Education Networks in Russia.” http://www.meti.go.jp/report/downloadfiles/gokin13j.pdf. Retrieved 1 July 2017

[4] Greg Cole, Jun Li, Jerome Sobieski, Dongkyun Kim, Donald Riley. NSF Award: “IRNC: ProNet: GLORIAD”. https://www.nsf.gov/awardsearch/showAward?AWD_ID=0963058. Retrieved 1 July 2017

[5] “Little GLORIAD” https://www.nsf.gov/od/lpa/news/03/images/littlegloriadstill.jpg. Retrieved 1 July 2017

[6] “KISTI Super Computing Center Information”. http://www.nisn.re.kr/eng/action.do?menuId=50031. Retrieved 1 July 2017

[7] “QoSient Argus”. https://qosient.com/argus/. Retrieved 1 July 2017

[8] Greg Cole. “GLORIAD’s New Measurement and Monitoring System for Addressing Individual Customer-based Performance across a Global Network Fabric”. https://www.internet2.edu/presentations/tip2013/20130116-Cole-GLORIAD.pdf. Retrieved 1 July 2017

[9] “Elastic Software Suite”. https://www.elastic.co. Retrieved 1 July 2017

[10] “GLORIAD InSight”. http://insight.gloriad.org/. Retrieved 1 July 2017

[11] Hintjens, Pieter. *ZeroMQ: messaging for many applications*. "O'Reilly Media, Inc.", 2013.

[12] “Monit”. https://mmonit.com/monit/documentation/monit.html. Retrieved 1 July 2017

[13] Greenspan, Jay, and Brad Bulger. *MySQL/PHP database applications*. John Wiley & Sons, Inc., 2001.

[14] Junyan, Lv, Xu Shiguo, and Li Yijie. "Application research of embedded database SQLite." *Information Technology and Applications, 2009. IFITA'09. International Forum on*. Vol. 2. IEEE, 2009.

[15] Jacobson, Van, Craig Leres, and S. McCanne. "The tcpdump manual page." *Lawrence Berkeley Laboratory, Berkeley, CA* 143 (1989).

[16] Obraczka, Katia, and Fabio Silva. "Network latency metrics for server proximity." Global Telecommunications Conference, 2000. GLOBECOM'00. IEEE. Vol. 1. IEEE, 2000.

[17] Faezipour, Miad, and Mehrdad Nourani. "Wire-speed TCAM-based architectures for multimatch packet classification." *IEEE Transactions on Computers* 58.1 (2009): 5-17.

[18] Claise, Benoit. "Cisco systems netflow services export version 9." (2004).

[19] Scarlato, Michele. "Network Monitoring in Software Defined Networking."

[20] Sullivan, Francis. "System and method for hardware and software monitoring with integrated troubleshooting." U.S. Patent Application No. 12/358,424.

[21] “Logic Monitor”. https://3rxsdqm2iblvg8du1xff1z16-wpengine.netdna-ssl.com/wp-content/uploads/2017/05/LogicMonitor-Security-Whitepaper-v1.3.pdf

[22] “Pressler: PRTG”. https://download-cdn.paessler.com/download/prtgmanual.pdf. Retrieved 1 July 2017

[23] Lorenzo, JUAN MANUEL. "AlienVault Users Manual." *Version* 1 (2011): 2010-2011.

[24] Sandke, Steven Robert, and Bryan Burns. "Targeted attack protection using predictive sandboxing." U.S. Patent No. 9,596,264. 14 Mar. 2017.

[25] Dietrich, D. "Bogons and bogon filtering." 33rd meeting of the North American Network Operator’s Group (NANOG 33). 2005.

[26] Caputo, Rocco. "POE: The Perl object environment." published at http://www. perl. org/poedown/poe-whitepaper-a4. pdf (2003).

[27] “State of Spam and Phishing”. https://www.symantec.com/content/dam/symantec/docs/security-center/archives/spam-report-oct-10-en.pdf. Retrieved 1 July 2017

[28] “Elasticsearch Benchmark of Scripting Languages”. https://benchmarks.elastic.co/index.html#search_qps_scripts. Retrieved 1 July 2017

[29] Białecki, Andrzej, et al. "Apache lucene 4." SIGIR 2012 workshop on open source information retrieval. 2012.

[30] Abubakar, Yusuf, Thankgod Sani Adeyi, and Ibrahim Gambo Auta. "Performance evaluation of NoSQL systems using YCSB in a resource austere environment." Performance Evaluation 7.8 (2014).

[31] Kemeny, John G., and James Laurie Snell. Finite markov chains. Vol. 356. Princeton, NJ: van Nostrand, 1960.

[32] Zheng, Li, Liren Zhang, and Dong Xu. "Characteristics of network delay and delay jitter and its effect on voice over IP (VoIP)." Communications, 2001. ICC 2001. IEEE International Conference on. Vol. 1. IEEE, 2001.

[33] Moore, David, et al. "Inferring internet denial-of-service activity." ACM Transactions on Computer Systems (TOCS) 24.2 (2006): 115-139.

[34] Kirda, Engin, et al. "Noxes: a client-side solution for mitigating cross-site scripting attacks." Proceedings of the 2006 ACM symposium on applied computing. ACM, 2006.

[35] McGrew, David, and D. Bailey. AES-CCM Cipher Suites for Transport Layer Security (TLS). No. RFC 6655. 2012.

[36] Aertsen, Maarten, et al. "No domain left behind: is Let's Encrypt democratizing encryption?." arXiv preprint arXiv:1612.03005 (2016).

APPENDICES

Appendix A

Table A.1 Core Argus Clients

Client	Process-ing Type	Description
ra	live stream	This client Argus data stream from a file to network socket or piped from another clients output and outputs the processed result into a file, standard output or as a binary record in ASCII format.
rabins	buffered	Processes Argus records by time based bins. Argus Stream is Read from input and held for a Time period And the content is output as an Argus stream.
racluster	idle data	Clusters the input according to given parameters
racount	idle data	Outputs different counts in a given Argus file or stream that shows statistics about the input.
radium	live stream	This client is capable of Distributing Argus records and accepts inputs equivalent to ra* commands and functions in a similar manner.
aranon-ymize	idle data	Anonymous the Argus records
rasort	idle data	Based on the given perimeter list sorts the input
rasplit	buffered stream	Is capable of splitting Source records based on size, count, time, or flow event and outputs results in a similar manner to other ra* clients.

Table A.2 All Other Argus Clients

Client	Process-ing Type	Description
raconvert	idle data	Convert the ASCII format data into binary format
radark	idle data	Identifies IP addresses that belong to the dark address space, Which is the IP addresses that does not belong to active device.
radecode	live stream	Decodes user data using tshark program
radump	live stream	Prints the user data such as ‘suser’: Source user, ‘duser’: destination user.
raevent	live stream	Allows the use of creation and use of events for other ra* clients
rafilteraddr	live stream	this program filters record according to a given address list
ragraph	buffered stream	This client allows Argus data to be graphed using rabins and rrdtool programs
ragrep	live stream	Allows Argus data to be search using given search pattern
rahisto	idle data	Next day input data degenerate histogram
rahosts	live stream	Outputs the hosts seen in the input Argus data stream
ralabel	live stream	has the ability to label flow records in the incoming Argus stream based on IP address

Table A.2. Continued.

Client	Process-ing Type	Description
rapolicy	idle data	Uses Cisco Access Control list (ACL) and matches Argus records to these rules
raports	idle data	ra* based client that that uses host port for processing
rarpwatch	live stream	what ARP events are observed (supports IPv4 and IPv6)
raservices	live stream	Checks the bite pattern and identifies network Services seen in the input Argus stream provided by rauserdata
rasql	idle data	Can be used to read binary data stored Within SQL databases insert add using rasqlinsert
rasqlinsert	live stream	Used to insert Argus flow records into SQL database
rasqltimeindex	live stream	This client can be used to build time index that can be used to insert records into SQL database
rapath	idle data	Generate the path each communication took place by taking icmpmap Data into account
rastream	buffered stream	splits the incoming data stream into adjacent sections based on size, count, time or flow event
rastrip	live stream	This client can be used to remove specific fields from Argus records
ratemplate	live stream	Can be modified to support application specific requirements using a template

Table A.2. Continued.

Client	Process-ing Type	Description
ratimerange	live stream	output the time range of the input data stream
rauserdata	idle data	Produces an output from the input Argus data that can be used as a byte-pattern file for raservices.
ratop	live stream	Similar to Linux ‘top’ command this client shows top flows by the number of bytes transmitted

Table A.3 Regular Argus Fields

Field Name	Description
srcid	source identifier for Argus
stime	starting time of the record
ltime	ending time of the record
flgs	flags of the Flow State
seq	sequence number of Argus flow record
smac, dmac	MAC address Source or destination node
soui, doui	OUI part of the source or destination MAC address
saddr, daddr	IP address of the source or destination
proto	Protocol
sport, dport	Source or destination port number
stos, dtos	type of service byte value of the source or destination
sdsb, ddsb	diff serve light value of source or destination
sco, dco	country code of source or destination
sttl, dttl	Source time to live: source to destination or destination time to live: destination to source value
sipid, dipid	IP identifier of source or destination
smpls, dmpls	MPLS identifier of source or destination
spkts, dpkts	Packet account from source to destination or destination to source

Table A.3. Continued.

sbytes, dbytes	transaction bytes from source to destination or destination to source
sappbytes, dappbytes	application bytes from source to destination or destination to source
sload, dload	load from Source or destination in bits per second
sloss, dloss	packets retransmitted or dropped from Source or destination
sgap, dgap	bytes missing in the flow Stream from Source or destination
dir	transaction Direction
sintpkt, dintpkt	inter packet arrival time from Source or destination
sintdist, dintdist	Time-based distribution of arrival time between two packets by Source or destination
sintpktact, dintpktact	active arrival time between two packets from Source or destination
sintdistact, dintdistact	Time between two packets arriving from Source or destination
sintpktidl, dintpktidl	idle time between two packets from Source or destination
sintdistidl, dintdistidl	Distribution of the idle time of 2 packets from Source or destination
sjit, djit	Jitter observed at source or destination
sjitact, djitact	active jitter observed at source or destination
sjitidle, djitidle	Idle jitter observed at source or destination
state	state of the transaction
suser, duser	user data seen at the source or destination
swin, dwin	TCP window length advertised by Source or destination

Table A.3. Continued.

svlan, dvlan	virtual local area network (VLAN identification number at source or destination
svid, dvid	VLAN Identification number observed at source or destination
svpri, dvpri	Private VLAN Identification number observed at source or destination
srng, erng	Center time range by start or ending time
stcpb, dtcpb	base sequence number of TCP Source or destination
tcprtt	round trip time of the connection
synack	time to set up connection between SYN and SYN_ACK
ackdat	time to set up connection between SYN_ACK and ACK
tcpopt	Connection options observed or the lack of it during connection initiation
inode	intermediate node IP address of ICMP event
offset	offset reported in the TCP header
spktsz, dpktsz	histogram of the distribution of package sizes from Source or destination
smaxsz, dmaxsz	maximum package size of the packets Santa by Source or destination
sminsz, dminsz	minimum package size of the packets Santa by Source or destination

Table A.4 Calculated Argus Fields

Field Name	Description
dur	Flow duration
rate, srate, drate	package rate in packets per second
trans	total record count of the incoming Argus stream
runtime	total sum of duration of the records observed in the input
mean	mean of duration of the records observed in the input
stddev	standard deviationof duration of the records observed in the input
sum	sum of duration of the records observed in the input
min	minimum of duration of the records observed in the input
max	maximum of duration of the records observed in the input
pkts	number of packets seen in the transaction
bytes	number of bytes seen in the transaction
appbytes	number of application bytes seen in the transaction
load	network load observed in the incoming Argus flow in bits per second
loss	number of packet retransmissions or dropped packets
ploss	percentage of number of packet retransmissions or dropped packets
sploss, dploss	number of packet retransmissions or dropped packets by Source or destination
abr	ratio between source application bytes and destination application bytes

Table A.5 Fields Stored in Elasticsearch

Field Name	Type	FORMAT
doc.ackdat	Argus Data	FLOAT
doc.appbytes	Argus Data	LONG
doc.bytes	Argus Data	LONG
doc.count	Argus Data	INTEGER
doc.dst.application	GSR Data	KEYWORD
doc.dst.asnum	GSR Data	SHORT
doc.dst.bytes	Argus Data	DOUBLE
doc.dst.discipline	GSR Data	KEYWORD
doc.dst.dns	GSR Data	KEYWORD
doc.dst.domainid	GSR Data	INTEGER
doc.dst.geoip.city	GeoIP Data	KEYWORD
doc.dst.geoip.country	GeoIP Data	KEYWORD
doc.dst.geoip.location	GeoIP Data	GEO_POINT
doc.dst.geoip.province	GeoIP Data	KEYWORD
doc.dst.geoip.zip	GeoIP Data	KEYWORD
doc.dst.ip	Argus Data	IP
doc.dst.iplabels	GSR Data	KEYWORD
doc.dst.ipname	GSR Data	KEYWORD
doc.dst.isp	GSR Data	KEYWORD
doc.dst.legacy.location	Legacy Data	GEO_POINT
doc.dst.legacy.security	Legacy Data	KEYWORD
doc.dst.load	Argus Data	FLOAT
doc.dst.organization	GSR Data	KEYWORD
doc.dst.orgclass	GSR Data	KEYWORD
doc.dst.port	Argus Data	INTEGER
doc.dst.predicted.security	Predicted Data	KEYWORD
doc.dst.rate	Argus Data	FLOAT
doc.dst.region	GeoIP Data	KEYWORD
doc.dst.security	Security Data	KEYWORD
doc.duration	Argus Data	FLOAT
doc.flags	Argus Data	KEYWORD
doc.hops	Argus Data	SHORT
doc.intpkt	Argus Data	FLOAT
doc.jitter	Argus Data	FLOAT

Table A.5. Continued.

Field Name	Type	FORMAT
doc.load	Argus Data	FLOAT
doc.loss	Argus Data	INTEGER
doc.maxsize	Argus Data	INTEGER
doc.meansize	Argus Data	INTEGER
doc.minsize	Argus Data	INTEGER
doc.packets	Argus Data	INTEGER
doc.pcr	Argus Data	FLOAT
doc.ploss	Argus Data	FLOAT
doc.pretrans	Argus Data	FLOAT
doc.protocol	Argus Data	SHORT
doc.rate	Argus Data	FLOAT
doc.retrans	Argus Data	INTEGER
doc.rtt	Argus Data	FLOAT
doc.src.application	GSR Data	KEYWORD
doc.src.asnum	GSR Data	SHORT
doc.src.bytes	Argus Data	DOUBLE
doc.src.discipline	GSR Data	KEYWORD
doc.src.dns	GSR Data	KEYWORD
doc.src.domainid	GSR Data	INTEGER
doc.src.geoip.city	GeoIP Data	KEYWORD
doc.src.geoip.country	GeoIP Data	KEYWORD
doc.src.geoip.location	GeoIP Data	GEO_POINT
doc.src.geoip.province	GeoIP Data	KEYWORD
doc.src.geoip.zip	GeoIP Data	KEYWORD
doc.src.ip	Argus Data	IP
doc.src.iplabels	GSR Data	KEYWORD
doc.src.ipname	GSR Data	KEYWORD
doc.src.isp	GSR Data	KEYWORD
doc.src.legacy.location	Legacy Data	GEO_POINT
doc.src.legacy.security	Legacy Data	KEYWORD
doc.src.load	Argus Data	FLOAT
doc.src.organization	GSR Data	KEYWORD
doc.src.orgclass	GSR Data	KEYWORD
doc.src.port	Argus Data	INTEGER

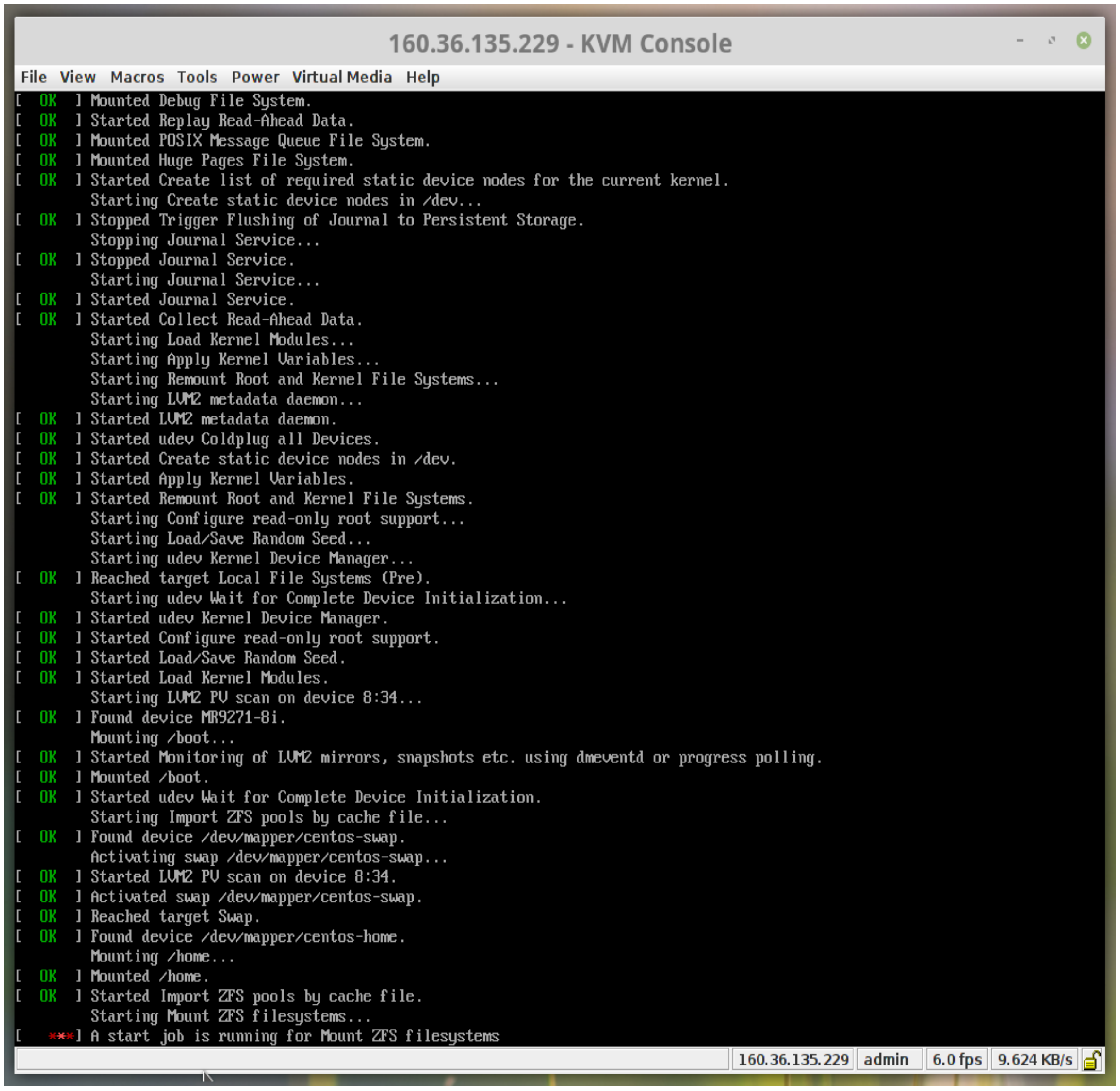
Table A.5. Continued.

Field Name	Type	FORMAT
doc.src.predicted.security	Predicted Data	KEYWORD
doc.src.rate	Argus Data	FLOAT
doc.src.region	GSR Data	KEYWORD
doc.src.security	Security Data	KEYWORD
doc.synack	Argus Data	FLOAT
doc.tcpopt	Argus Data	KEYWORD
doc.time	Argus Data	DATE
doc.tos	Argus Data	KEYWORD
doc.vlanid	GSR Data	INTEGER

Table A.6 GSR Information Stored in Elasticsearch

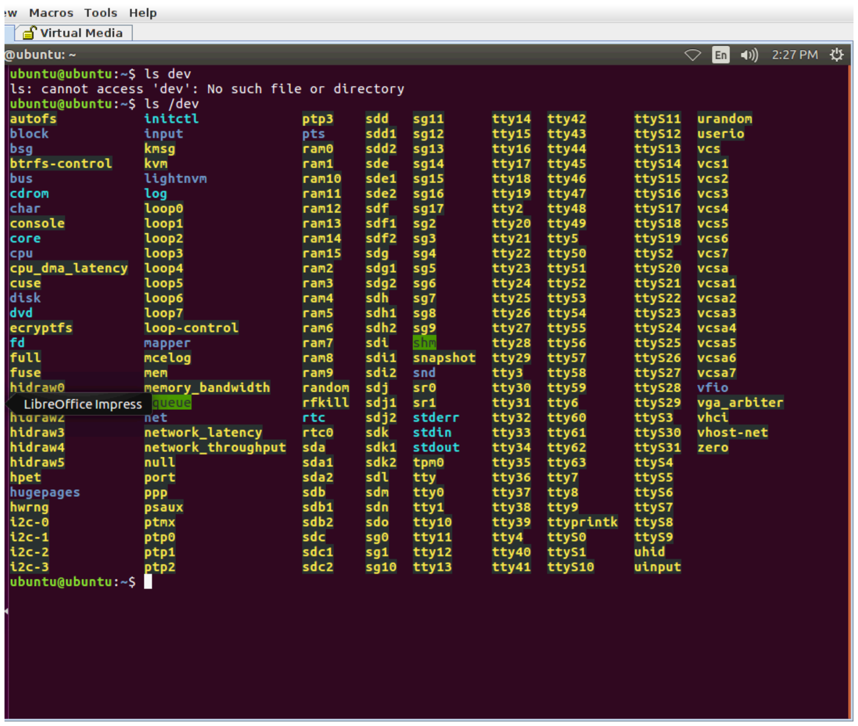
Field Name	Type	FORMAT
ip	IP Address	KEYWORD
organization	GSR Data	KEYWORD
domainid	GSR Data	INTEGER
dns	GSR Data	KEYWORD
ipname	GSR Data	KEYWORD
isp	GSR Data	KEYWORD
labels	GSR Data	KEYWORD
asnum	GSR Data	INTEGER
discipline	GSR Data	KEYWORD
agency	GSR Data	KEYWORD
application	GSR Data	KEYWORD

Appendix B


160.36.135.229 - KVM Console
File View Macros Tools Power Virtual Media Help
160.36.135.229 admin 6.0 fps 9.624 KB/s

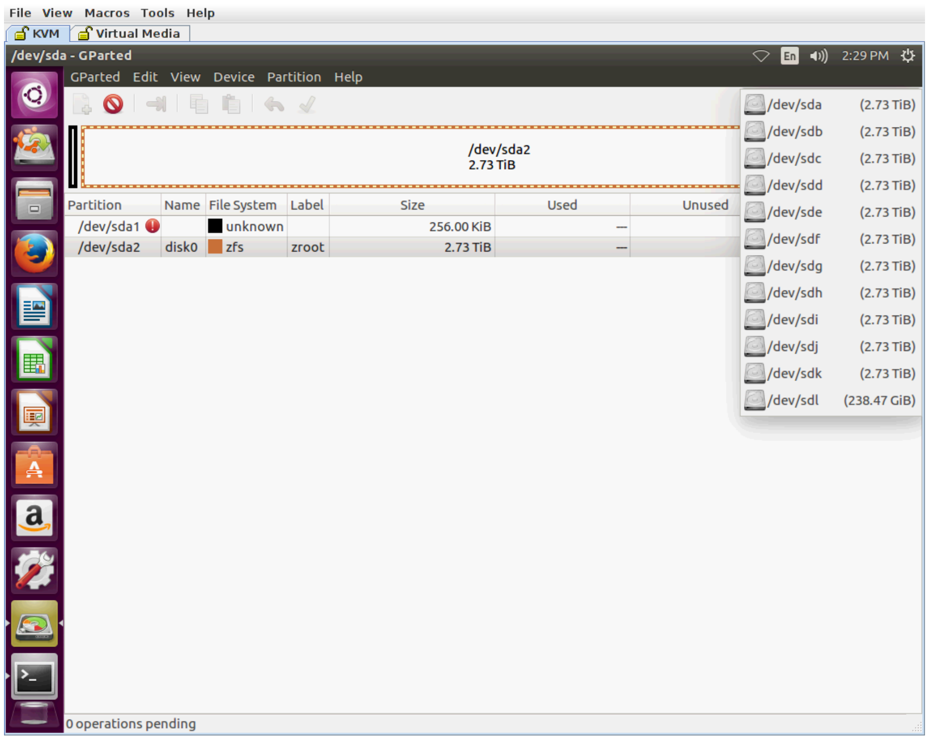
```
[  OK  ] Mounted Debug File System.
[  OK  ] Started Replay Read-Ahead Data.
[  OK  ] Mounted POSIX Message Queue File System.
[  OK  ] Mounted Huge Pages File System.
[  OK  ] Started Create list of required static device nodes for the current kernel.
         Starting Create static device nodes in /dev...
[  OK  ] Stopped Trigger Flushing of Journal to Persistent Storage.
         Stopping Journal Service...
[  OK  ] Stopped Journal Service.
         Starting Journal Service...
[  OK  ] Started Journal Service.
[  OK  ] Started Collect Read-Ahead Data.
         Starting Load Kernel Modules...
         Starting Apply Kernel Variables...
         Starting Remount Root and Kernel File Systems...
         Starting LVM2 metadata daemon...
[  OK  ] Started LVM2 metadata daemon.
[  OK  ] Started udev Coldplug all Devices.
[  OK  ] Started Create static device nodes in /dev.
[  OK  ] Started Apply Kernel Variables.
[  OK  ] Started Remount Root and Kernel File Systems.
         Starting Configure read-only root support...
         Starting Load/Save Random Seed...
         Starting udev Kernel Device Manager...
[  OK  ] Reached target Local File Systems (Pre).
         Starting udev Wait for Complete Device Initialization...
[  OK  ] Started udev Kernel Device Manager.
[  OK  ] Started Configure read-only root support.
[  OK  ] Started Load/Save Random Seed.
[  OK  ] Started Load Kernel Modules.
         Starting LVM2 PV scan on device 8:34...
[  OK  ] Found device MR9271-8i.
         Mounting /boot...
[  OK  ] Started Monitoring of LVM2 mirrors, snapshots etc. using dmeventd or progress polling.
[  OK  ] Mounted /boot.
[  OK  ] Started udev Wait for Complete Device Initialization.
         Starting Import ZFS pools by cache file...
[  OK  ] Found device /dev/mapper/centos-swap.
         Activating swap /dev/mapper/centos-swap...
[  OK  ] Started LVM2 PV scan on device 8:34.
[  OK  ] Activated swap /dev/mapper/centos-swap.
[  OK  ] Reached target Swap.
[  OK  ] Found device /dev/mapper/centos-home.
         Mounting /home...
[  OK  ] Mounted /home.
[  OK  ] Started Import ZFS pools by cache file.
         Starting Mount ZFS filesystems...
[   ***] A start job is running for Mount ZFS filesystems
```

Figure B.1. Mithril’s File System Failure



```
ubuntu@ubuntu:~$ ls dev
ls: cannot access 'dev': No such file or directory
ubuntu@ubuntu:~$ ls /dev
autofs           initctl             ptp3    sdd   sg11      tty14  tty42      ttyS11  urandom
block            input               pts     sdd1  sg12      tty15  tty43      ttyS12  userio
bsg              kmsg                ram0    sdd2  sg13      tty16  tty44      ttyS13  vcs
btrfs-control    kvm                 ram1    sde   sg14      tty17  tty45      ttyS14  vcs1
bus              lightnvm            ram10   sde1  sg15      tty18  tty46      ttyS15  vcs2
cdrom            log                 ram11   sde2  sg16      tty19  tty47      ttyS16  vcs3
char             loop0               ram12   sdf   sg17      tty2   tty48      ttyS17  vcs4
console          loop1               ram13   sdf1  sg2       tty20  tty49      ttyS18  vcs5
core             loop2               ram14   sdf2  sg3       tty21  tty5       ttyS19  vcs6
cpu              loop3               ram15   sdg   sg4       tty22  tty50      ttyS2   vcs7
cpu_dma_latency  loop4               ram2    sdg1  sg5       tty23  tty51      ttyS20  vcsa
cuse             loop5               ram3    sdg2  sg6       tty24  tty52      ttyS21  vcsa1
disk             loop6               ram4    sdh   sg7       tty25  tty53      ttyS22  vcsa2
dvd              loop7               ram5    sdh1  sg8       tty26  tty54      ttyS23  vcsa3
ecryptfs         loop-control        ram6    sdh2  sg9       tty27  tty55      ttyS24  vcsa4
fd               mapper              ram7    sdi   shm       tty28  tty56      ttyS25  vcsa5
full             mcelog              ram8    sdi1  snapshot  tty29  tty57      ttyS26  vcsa6
fuse             mem                 ram9    sdi2  snd       tty3   tty58      ttyS27  vcsa7
hidraw0          memory_bandwidth    random  sdj   sr0       tty30  tty59      ttyS28  vfio
[illegible]      queue               rfkill  sdj1  sr1       tty31  tty6       ttyS29  vga_arbiter
hidraw2          net                 rtc     sdj2  stderr    tty32  tty60      ttyS3   vhci
hidraw3          network_latency     rtc0    sdk   stdin     tty33  tty61      ttyS30  vhost-net
hidraw4          network_throughput  sda     sdk1  stdout    tty34  tty62      ttyS31  zero
hidraw5          null                sda1    sdk2  tpm0      tty35  tty63      ttyS4
hpet             port                sda2    sdl   tty       tty36  tty7       ttyS5
hugepages        ppp                 sdb     sdm   tty0      tty37  tty8       ttyS6
hwrng            psaux               sdb1    sdn   tty1      tty38  tty9       ttyS7
i2c-0            ptmx                sdb2    sdo   tty10     tty39  ttyprintk  ttyS8
i2c-1            ptp0                sdc     sg0   tty11     tty4   ttyS0      ttyS9
i2c-2            ptp1                sdc1    sg1   tty12     tty40  ttyS1      uhid
i2c-3            ptp2                sdc2    sg10  tty13     tty41  ttyS10     uinput
ubuntu@ubuntu:~$
```

Figure B.2. Checking Mithril’s Drive Configuration



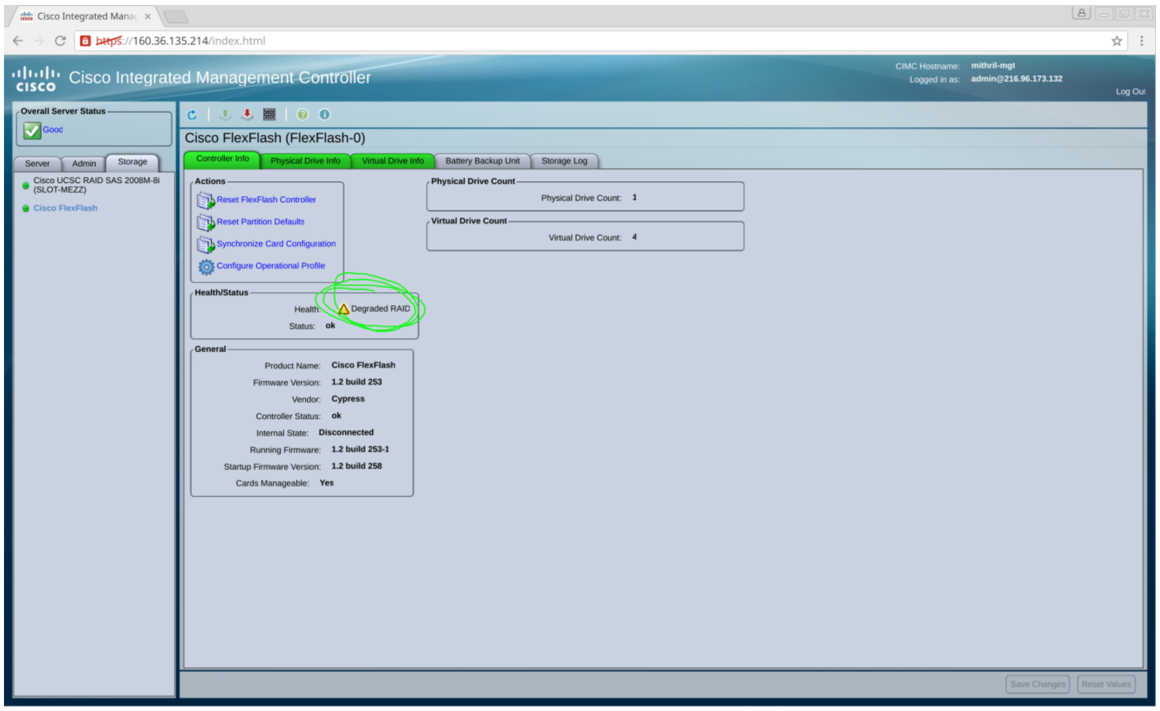
File View Macros Tools Help
KVM
Virtual Media
/dev/sda - GParted
En
2:29 PM
GParted Edit View Device Partition Help
/dev/sda2
2.73 TiB

Partition	Name	File System	Label	Size	Used	Unused
/dev/sda1		■ unknown		256.00 KiB	—	
/dev/sda2	disk0	■ zfs	zroot	2.73 TiB	—	

/dev/sda	(2.73 TiB)
/dev/sdb	(2.73 TiB)
/dev/sdc	(2.73 TiB)
/dev/sdd	(2.73 TiB)
/dev/sde	(2.73 TiB)
/dev/sdf	(2.73 TiB)
/dev/sdg	(2.73 TiB)
/dev/sdh	(2.73 TiB)
/dev/sdi	(2.73 TiB)
/dev/sdj	(2.73 TiB)
/dev/sdk	(2.73 TiB)
/dev/sdl	(238.47 GiB)

0 operations pending

Figure B.3. Checking Mithril’s Physical Disk Drives


Cisco Integrated Management Controller

https://160.36.135.214/index.html

CIMC Hostname: mithril-mgt
Logged in as: admin@216.96.173.132

Overall Server Status: Good

Server | Admin | Storage

- Cisco UCSC RAID SAS 2008M-8i (SLOT-MEZZ)
- Cisco FlexFlash

Cisco FlexFlash (FlexFlash-0)

Controller Info | Physical Drive Info | Virtual Drive Info | Battery Backup Unit | Storage Log

Actions
- Reset FlexFlash Controller
- Reset Partition Defaults
- Synchronize Card Configuration
- Configure Operational Profile

Physical Drive Count
Physical Drive Count: 1

Virtual Drive Count
Virtual Drive Count: 4

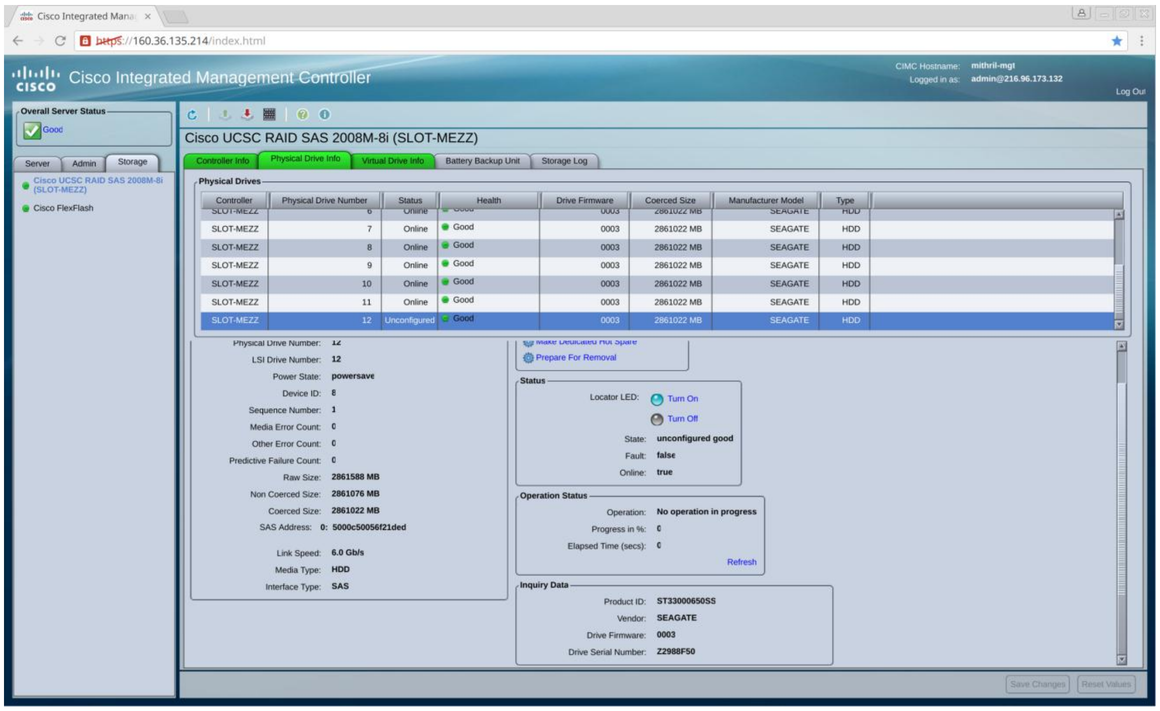
Health/Status

Health:	Degraded RAID
Status:	ok

General

Product Name:	Cisco FlexFlash
Firmware Version:	1.2 build 253
Vendor:	Cypress
Controller Status:	ok
Internal State:	Disconnected
Running Firmware:	1.2 build 253-1
Startup Firmware Version:	1.2 build 258
Cards Manageable:	Yes

Figure B.4. Degraded State of RAID Configuration


Cisco Integrated Management Controller

https://160.36.135.214/index.html

CIMC Hostname: mithril-mgt
Logged in as: admin@216.96.173.132

Overall Server Status: Good

Server | Admin | Storage

- Cisco UCSC RAID SAS 2008M-8i (SLOT-MEZZ)
- Cisco FlexFlash

Cisco UCSC RAID SAS 2008M-8i (SLOT-MEZZ)

Controller Info | Physical Drive Info | Virtual Drive Info | Battery Backup Unit | Storage Log

Physical Drives

Controller	Physical Drive Number	Status	Health	Drive Firmware	Coerced Size	Manufacturer Model	Type
SLOT-MEZZ	6	Online	Good	0003	2861022 MB	SEAGATE	HDD
SLOT-MEZZ	7	Online	Good	0003	2861022 MB	SEAGATE	HDD
SLOT-MEZZ	8	Online	Good	0003	2861022 MB	SEAGATE	HDD
SLOT-MEZZ	9	Online	Good	0003	2861022 MB	SEAGATE	HDD
SLOT-MEZZ	10	Online	Good	0003	2861022 MB	SEAGATE	HDD
SLOT-MEZZ	11	Online	Good	0003	2861022 MB	SEAGATE	HDD
SLOT-MEZZ	12	Unconfigured	Good	0003	2861022 MB	SEAGATE	HDD

Physical Drive Number:	12
LSI Drive Number:	12
Power State:	powersave
Device ID:	8
Sequence Number:	1
Media Error Count:	0
Other Error Count:	0
Predictive Failure Count:	0
Raw Size:	2861588 MB
Non Coerced Size:	2861076 MB
Coerced Size:	2861022 MB
SAS Address:	0: 5000c50056f21ded
Link Speed:	6.0 Gb/s
Media Type:	HDD
Interface Type:	SAS

- Make Dedicated Hot Spare
- Prepare For Removal

Status

Locator LED:	Turn On / Turn Off
State:	unconfigured good
Fault:	false
Online:	true

Operation Status

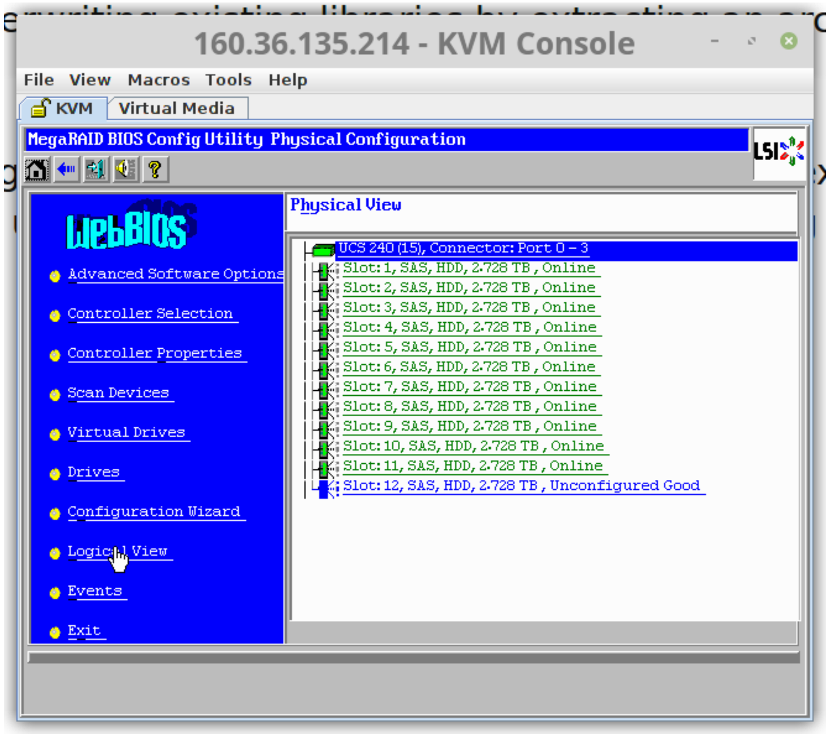
Operation:	No operation in progress
Progress in %:	0
Elapsed Time (secs):	0

Refresh

Inquiry Data

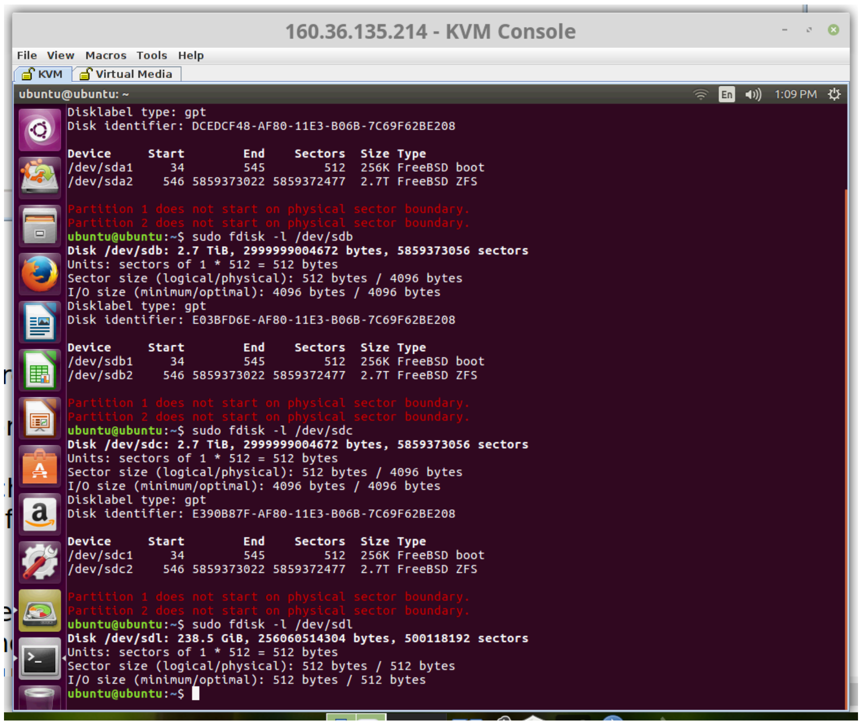
Product ID:	ST33000650SS
Vendor:	SEAGATE
Drive Firmware:	0003
Drive Serial Number:	Z2988F50

Figure B.5. Mithril’s Unconfigured Drive in CIMC



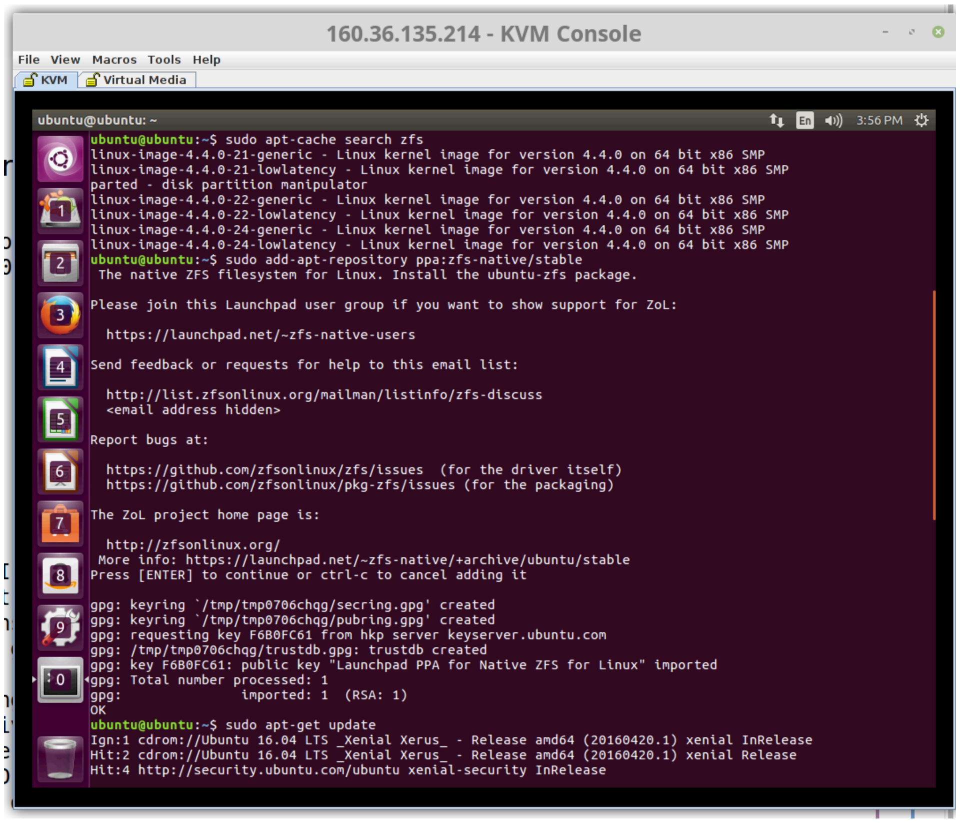
```
UCS 240 (15), Connector: Port 0 – 3
  Slot: 1, SAS, HDD, 2.728 TB , Online
  Slot: 2, SAS, HDD, 2.728 TB , Online
  Slot: 3, SAS, HDD, 2.728 TB , Online
  Slot: 4, SAS, HDD, 2.728 TB , Online
  Slot: 5, SAS, HDD, 2.728 TB , Online
  Slot: 6, SAS, HDD, 2.728 TB , Online
  Slot: 7, SAS, HDD, 2.728 TB , Online
  Slot: 8, SAS, HDD, 2.728 TB , Online
  Slot: 9, SAS, HDD, 2.728 TB , Online
  Slot: 10, SAS, HDD, 2.728 TB , Online
  Slot: 11, SAS, HDD, 2.728 TB , Online
  Slot: 12, SAS, HDD, 2.728 TB , Unconfigured Good
```

Figure B.6. Unconfigured Drive in WebBIOS



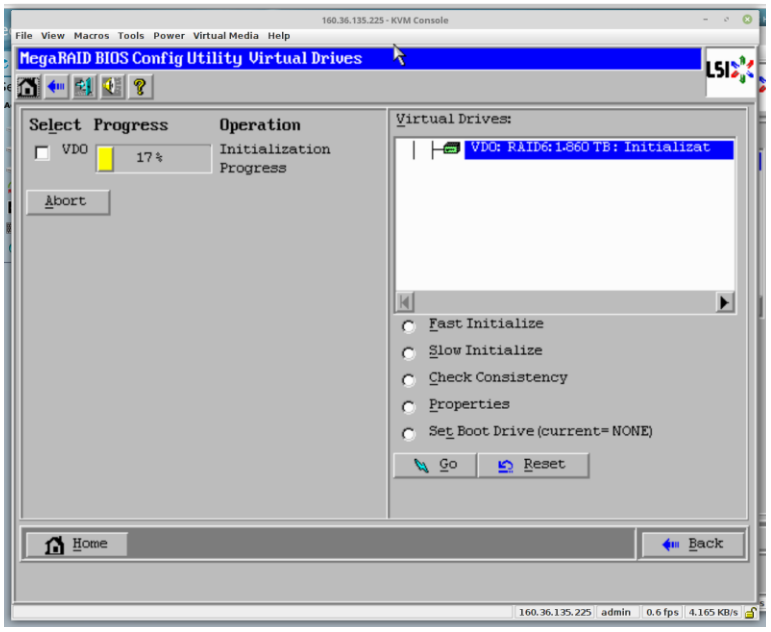
```
Disklabel type: gpt
Disk identifier: DCEDCF48-AF80-11E3-B06B-7C69F62BE208

Device      Start        End    Sectors  Size Type
/dev/sda1      34        545        512  256K FreeBSD boot
/dev/sda2     546 5859373022 5859372477  2.7T FreeBSD ZFS

Partition 1 does not start on physical sector boundary.
Partition 2 does not start on physical sector boundary.
ubuntu@ubuntu:~$ sudo fdisk -l /dev/sdb
Disk /dev/sdb: 2.7 TiB, 2999999004672 bytes, 5859373056 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: gpt
Disk identifier: E03BFD6E-AF80-11E3-B06B-7C69F62BE208

Device      Start        End    Sectors  Size Type
/dev/sdb1      34        545        512  256K FreeBSD boot
/dev/sdb2     546 5859373022 5859372477  2.7T FreeBSD ZFS

Partition 1 does not start on physical sector boundary.
Partition 2 does not start on physical sector boundary.
ubuntu@ubuntu:~$ sudo fdisk -l /dev/sdc
Disk /dev/sdc: 2.7 TiB, 2999999004672 bytes, 5859373056 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: gpt
Disk identifier: E390B87F-AF80-11E3-B06B-7C69F62BE208

Device      Start        End    Sectors  Size Type
/dev/sdc1      34        545        512  256K FreeBSD boot
/dev/sdc2     546 5859373022 5859372477  2.7T FreeBSD ZFS

Partition 1 does not start on physical sector boundary.
Partition 2 does not start on physical sector boundary.
ubuntu@ubuntu:~$ sudo fdisk -l /dev/sdl
Disk /dev/sdl: 238.5 GiB, 256060514304 bytes, 500118192 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
ubuntu@ubuntu:~$
```

Figure B.7. Checking Drive Health



```
ubuntu@ubuntu:~$ sudo apt-cache search zfs
linux-image-4.4.0-21-generic - Linux kernel image for version 4.4.0 on 64 bit x86 SMP
linux-image-4.4.0-21-lowlatency - Linux kernel image for version 4.4.0 on 64 bit x86 SMP
parted - disk partition manipulator
linux-image-4.4.0-22-generic - Linux kernel image for version 4.4.0 on 64 bit x86 SMP
linux-image-4.4.0-22-lowlatency - Linux kernel image for version 4.4.0 on 64 bit x86 SMP
linux-image-4.4.0-24-generic - Linux kernel image for version 4.4.0 on 64 bit x86 SMP
linux-image-4.4.0-24-lowlatency - Linux kernel image for version 4.4.0 on 64 bit x86 SMP
ubuntu@ubuntu:~$ sudo add-apt-repository ppa:zfs-native/stable
 The native ZFS filesystem for Linux. Install the ubuntu-zfs package.

Please join this Launchpad user group if you want to show support for ZoL:

  https://launchpad.net/~zfs-native-users

Send feedback or requests for help to this email list:

  http://list.zfsonlinux.org/mailman/listinfo/zfs-discuss
  <email address hidden>

Report bugs at:

  https://github.com/zfsonlinux/zfs/issues  (for the driver itself)
  https://github.com/zfsonlinux/pkg-zfs/issues (for the packaging)

The ZoL project home page is:

  http://zfsonlinux.org/
 More info: https://launchpad.net/~zfs-native/+archive/ubuntu/stable
Press [ENTER] to continue or ctrl-c to cancel adding it

gpg: keyring `/tmp/tmp0706chqg/secring.gpg' created
gpg: keyring `/tmp/tmp0706chqg/pubring.gpg' created
gpg: requesting key F6B0FC61 from hkp server keyserver.ubuntu.com
gpg: /tmp/tmp0706chqg/trustdb.gpg: trustdb created
gpg: key F6B0FC61: public key "Launchpad PPA for Native ZFS for Linux" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
OK
ubuntu@ubuntu:~$ sudo apt-get update
Ign:1 cdrom://Ubuntu 16.04 LTS _Xenial Xerus_ - Release amd64 (20160420.1) xenial InRelease
Hit:2 cdrom://Ubuntu 16.04 LTS _Xenial Xerus_ - Release amd64 (20160420.1) xenial Release
Hit:4 http://security.ubuntu.com/ubuntu xenial-security InRelease
```

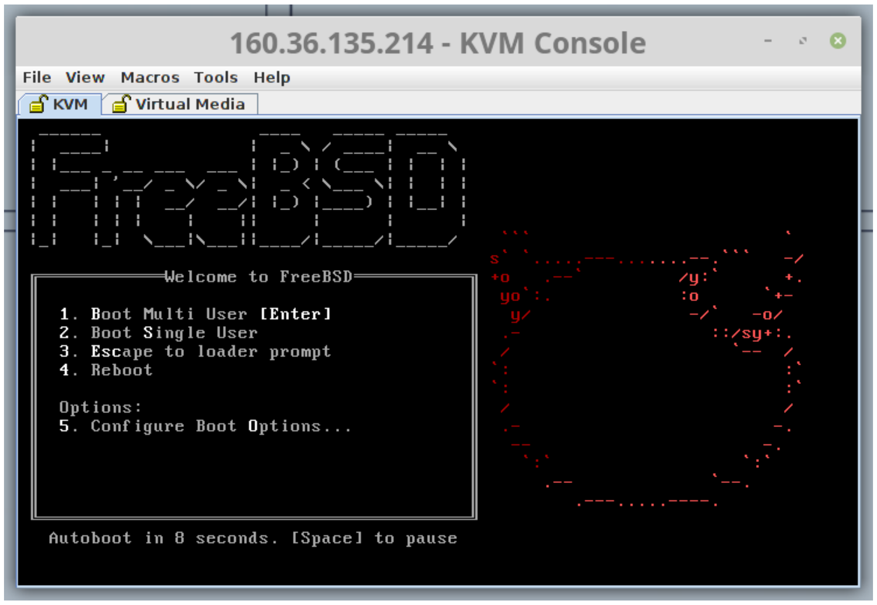
Figure B.8. ZFS Reassembly and Verification



```
Select  Progress     Operation
VD0     17%          Initialization
                     Progress
```

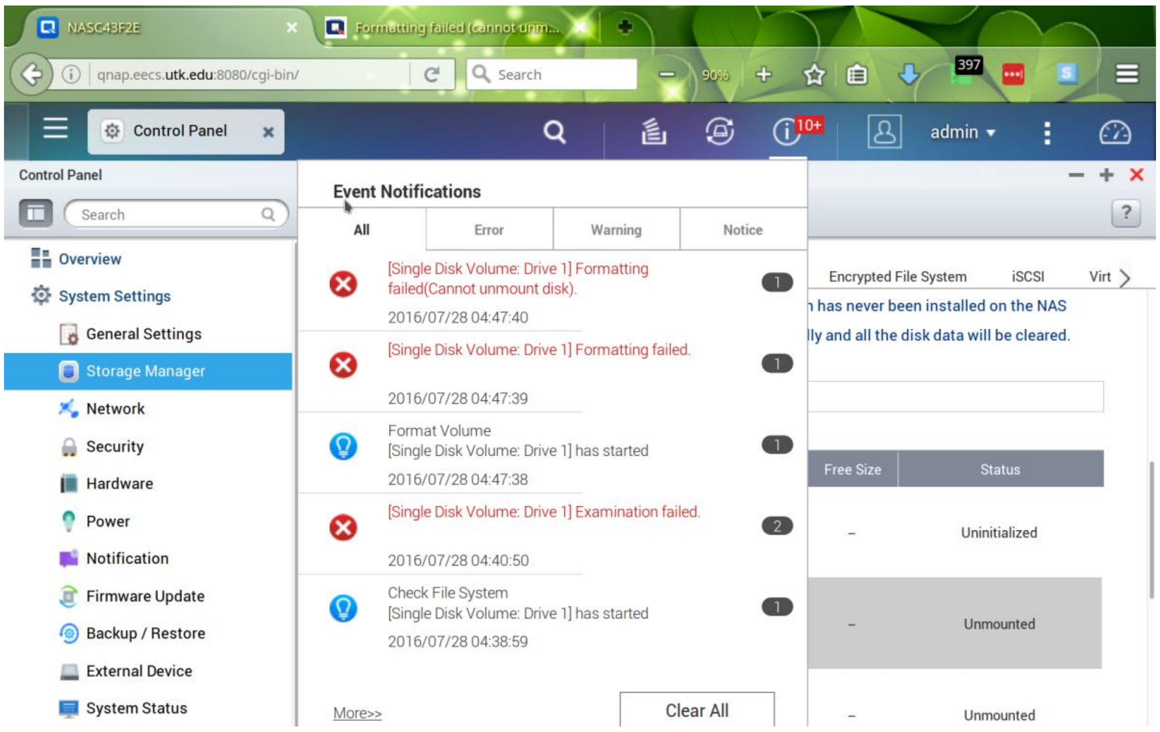
VD0: RAID6: 1.860 TB: Initializat

Figure B.9. Setting up RAID 6



```
Welcome to FreeBSD

 1. Boot Multi User [Enter]
 2. Boot Single User
 3. Escape to loader prompt
 4. Reboot

 Options:
 5. Configure Boot Options...

Autoboot in 8 seconds. [Space] to pause
```

Figure B.10. Booting Single User Mode



Event Notifications

All	Error	Warning	Notice

[Single Disk Volume: Drive 1] Formatting failed(Cannot unmount disk). 1
2016/07/28 04:47:40

[Single Disk Volume: Drive 1] Formatting failed. 1
2016/07/28 04:47:39

Format Volume
[Single Disk Volume: Drive 1] has started 1
2016/07/28 04:47:38

[Single Disk Volume: Drive 1] Examination failed. 2
2016/07/28 04:40:50

Check File System
[Single Disk Volume: Drive 1] has started 1
2016/07/28 04:38:59

h has never been installed on the NAS

lly and all the disk data will be cleared.

Free Size	Status
-	Uninitialized
-	Unmounted
-	Unmounted

Figure B.11. QNAP Disk Failure

Appendix C

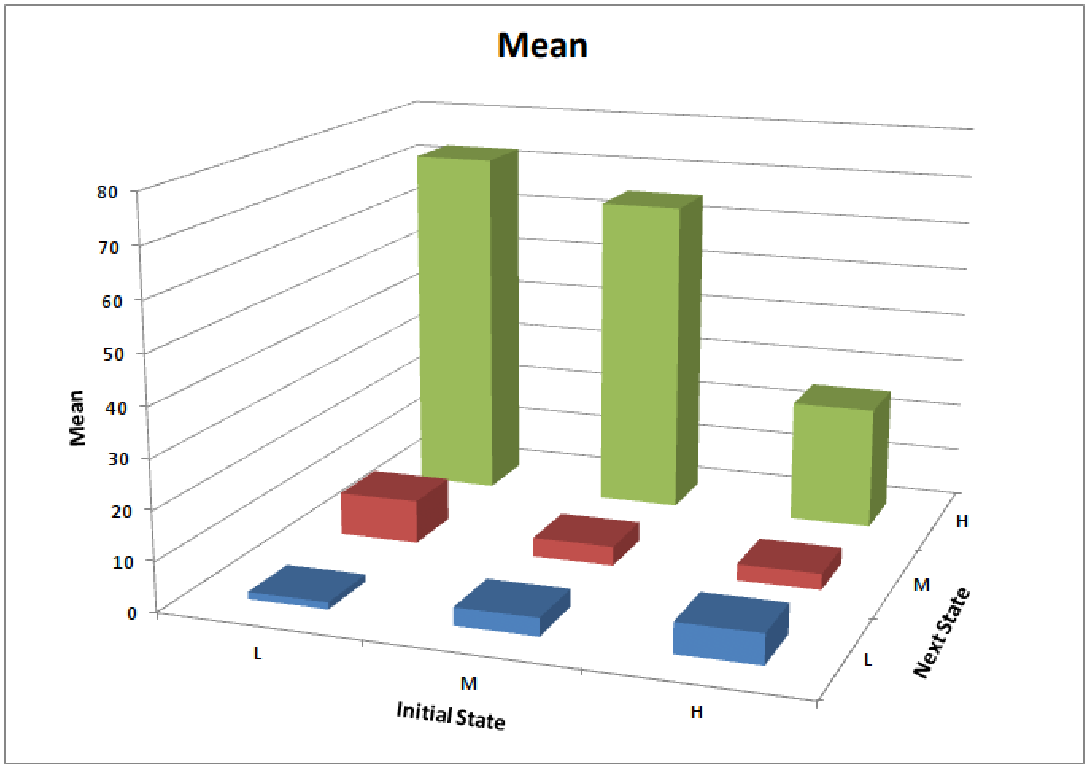


Table of mean values

Next State

Initial State

	L	M	H
L	1.41797	8.807453	72.48745
M	3.619215	3.923632	64.77647
H	6.092944	3.296081	25.06233

Figure C.1 Mean Value Distribution of the Markov Chain

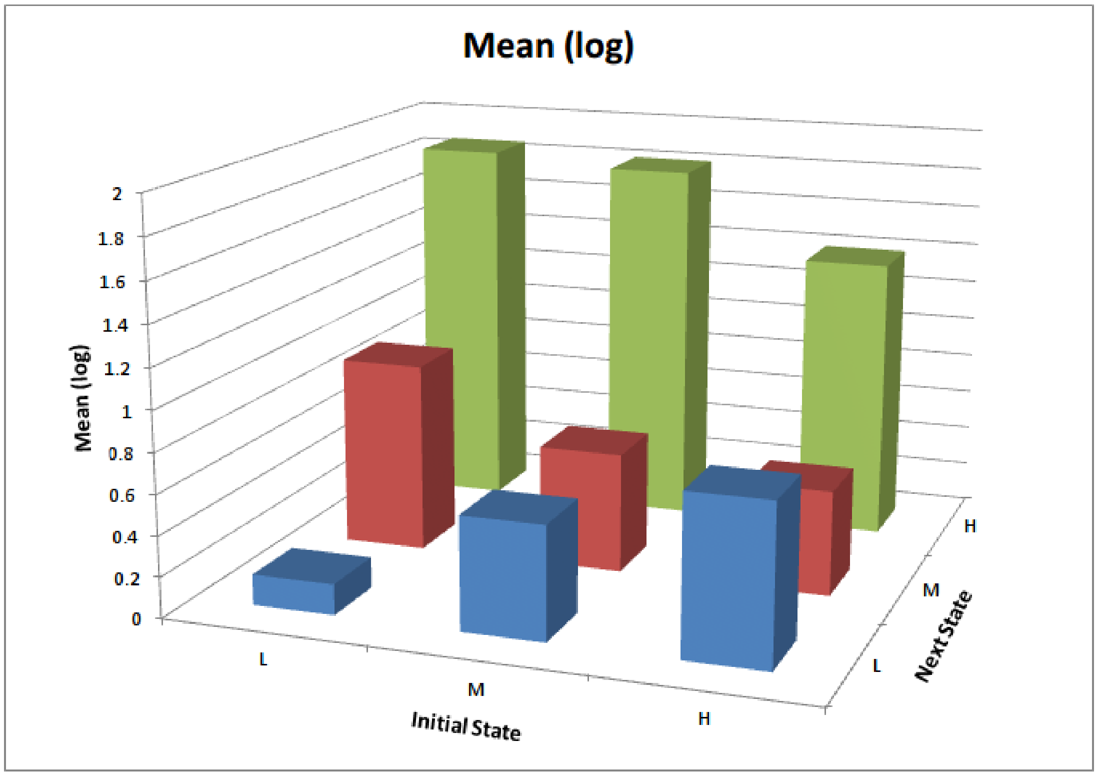


Table of log of mean values

Next State

Initial State	L	M	H
L	0.151667	0.94485	1.860263
M	0.558614	0.593688	1.811417
H	0.784827	0.517998	1.399021

Figure C.2 Log of Mean Value Distribution of the Markov Chain

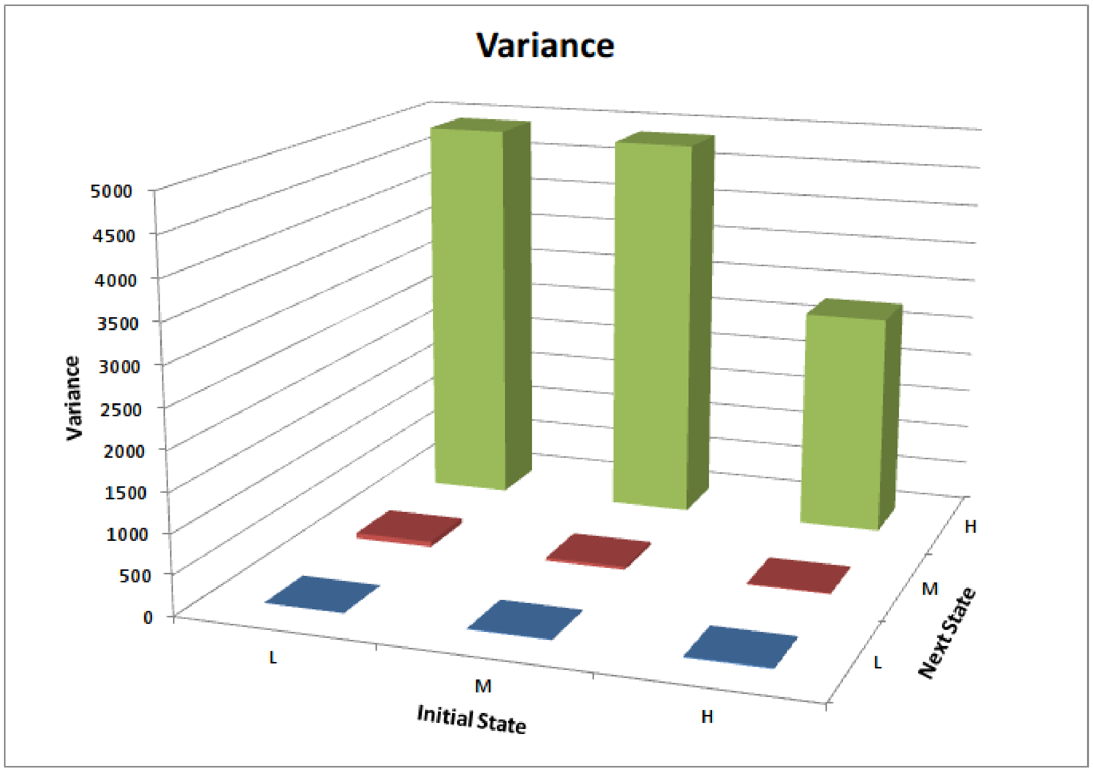


Table of variance values

Next State

Initial State	L	M	H
L	2.712694	68.28154	4918.691
M	11.71615	38.30258	4862.633
H	16.39869	13.96072	2786.867

Figure C.3 Variance Distribution of the Markov Chain

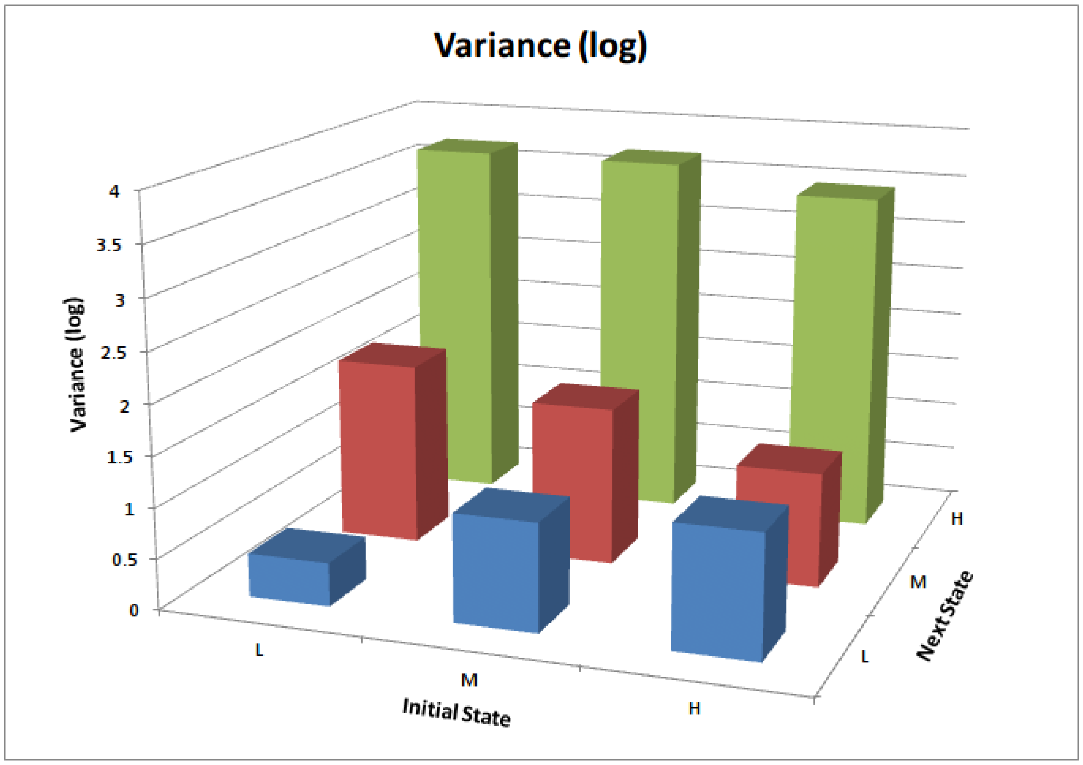


Table of log of variance values

Next State

Initial State

	L	M	H
L	0.433401	1.834303	3.69185
M	1.068785	1.583228	3.686871
H	1.214809	1.144908	3.445116

Figure C.4 Log of Variance Distribution of the Markov Chain

VITA

Hansaka Angel Dias Edirisinghe Kodituwakku was born in Colombo, Sri Lanka in 1990. During his bachelor’s degree in Electronic and Telecommunication Engineering at the University of Moratuwa, Sri Lanka, he did his internship at GLORIAD USA and contributed to the GLORIAD InSight platform by developing ‘Spider’ and ‘Scorpion’ modules. He graduated in 2014 and started Masters of Science in Computer Engineering at University of Tennessee in 2016. He developed InSight2 platform for his Master’s Thesis. He would graduate in August 2017.