3-19-2015

Unknown Threat Detection With Honeypot Ensemble Analsyis Using Big Datasecurity Architecture

Michael Eugene Sanders
Illinois State University, mesand3@ilstu.edu

UNKNOWN THREAT DETECTION WITH HONEYPOT ENSEMBLE ANALYSIS USING BIG DATA SECURITY ARCHITECTURE

Michael E. Sanders

77 Pages May 2015

The amount of data that is being generated continues to rapidly grow in size and complexity. Frameworks such as Apache Hadoop and Apache Spark are evolving at a rapid rate as organizations are building data driven applications to gain competitive advantages. Data analytics frameworks decompose problems with a large volume of data into smaller more manageable data sets to build applications that are more than just inference and can help make predictions as well as prescriptions to problems in real time instead of batch processes. For example, organizations like Netflix and Amazon use these frameworks to analyze their users.

Information Security is becoming more important to organizations as the Internet and cloud technologies become more integrated with their internal processes. The number of attacks and attack vectors has been increasing steadily over the years. Border defense measures (e.g. Intrusion Detection Systems) are no longer enough to identify and stop attackers. Data driven information security is not a new approach to solving information security; however there is an increased emphasis on combining heterogeneous sources to

gain a broader view of the problem instead of isolated systems. Stitching together multiple alerts into a cohesive system can increase the number of true positives.

With the increased concern of unknown insider threats and zero-day attacks, identifying unknown attack vectors becomes more difficult. Previous research has shown that with as little as 10 commands, which deviate from a user's normal behavior, it is possible to identify a masquerade attack against a user's profile.

This thesis is going to propose a data driven information security architecture that relies on both behavioral analysis of SSH profiles and bad actor data collected from an SSH honeypot to identify bad actor attack vectors. Previous studies have used behavioral analysis in an attempt to identify masquerade attacks. Multiple studies on POSIX Linux masquerade detection have used the Schonlau data set. Simple changes in user behavior can create high false positives.

Honeypot systems should not be used by normal users within an organization's environment. Only bad actors are going to attempt to use the system. This experiment is going to collect bad actor behavior from a honeypot ensemble with normal user behavior to increase the true positive rate of bad actor detection. Using Apache Spark and Apache Hadoop, we can create a real time data driven architecture that can collect and analyze new bad actor behaviors from honeypot data and monitor legitimate user accounts to create predictive and prescriptive models. Previously unidentified attack vectors can be cataloged for review.

UNKNOWN THREAT DETECTION WITH HONEYPOT ENSEMBLE ANALYSIS USING BIG DATA SECURITY ARCHITECTURE

MICHAEL E. SANDERS

A Thesis Submitted in Partial
Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

School of Information Technology

ILLINOIS STATE UNIVERSITY

2015

UNKNOWN THREAT DETECTION WITH HONEYPOT ENSEMBLE ANALYSIS USING BIG DATA SECURITY ARCHITECTURE

MICHAEL E. SANDERS

COMMITTEE MEMBERS:

Yongning Tang

Douglas Twitchell

James Wolf

ACKNOWLEDGMENTS

I would like to thank and dedicate this thesis to my wife, Melissa and my son, Oliver. They were with me during the long nights from start to finish. My wife kept me both motivated and positive through the tough process as I wracked my brain trying to complete this work. I would also like to thank my friends and colleagues as I was able to leverage areas in the field in which I needed extra guidance.

Also, I would like to express my gratitude to Dr. Tang for working with me on big data technologies and giving me the opportunity to build a Hadoop cluster for research. The staff and faculty at Illinois State University have been invaluable in ensuring that I had the knowledge and expertise in order to complete this thesis. Multiple friends, including Elliott, Robert and Jared gave a lot of real world advice in the Information Security field and on Honeypot architectures.

M.E.S

CONTENT

FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

APT	Advanced Persistent Threat
ETL	Extraction Transformation Loading
HACE	Heterogeneous, Autonomous, Complex, Evolving
HDFS	Hadoop File System
IDS	Intrusions Detection System
HMM	Hidden Markov Models
REPL	Read Eval Print Loop
RDD	Resilient Distributed Dataset
SSH	Secure Shell
SANS	SysAdmin Audit Networking Security
SYSLOG	System Logging
TCP/IP	Transmission Control Protocol/Internet Protocol
VM	Virtual Machine
VPN	Virtual Private Network
YARN	Yet Another Resource Negotiator

CHAPTER I

INTRODUCTION

New or unknown attack vectors are difficult for traditional border systems to detect. There are two main types of detection systems: knowledge based and behavioral based. Knowledge based detection systems focus on previously known attacks [5]. Behavioral systems are anomaly based detection systems that take a user's or accounts usage pattern and alerts when the behavior pattern deviates [53]. Due to the amount of data and how quick new attack vectors are discovered, they both have limitations causing high false positive rates [34].

Cyber-attacks are continuously on the rise, and the complexity is constantly evolving. Throughout the news, people are hearing about how organizations are being breached and consumer data is constantly compromised by these attacks. Two security breaches in 2014 include Chase bank [50] and a payment data breach at Home Depot [49]. Both attacks are analyzed in Appendix C, and are classified as advanced persistent threats (APT). According to Virvilis, et al. there are other major attacks and malware that have enormous complexity and are highly sophisticated; including Stuxnet, Flame and Red October [62] were high profile malware [34]. Each of these programs was skillfully crafted and required a background in multiple technology domains.

Detection of a network intrusion usually goes unnoticed for months. Some organizations believe that 71% of intrusions go undetected and some of the 29% of

detections don't happen until on average of 10 months after the detections happen [8]. The actual numbers could be much higher, because most organizations rotate security logs after so many days, let alone being able to analyze months of network logs. Intrusion software will alert on events based on a preset configuration. Complex and skillfully crafted Advanced Persistent Threats can go completely undetected in an organization because the software needs to be configured to detect the signature of the attack.

Near Real Time Data analytics systems, like Apache Spark, built on top of Hadoop can provide valuable insight into what kind of traffic is happening on a network. This paper will recommend an implementation with a sample architecture and analysis of real network logs. The paper will review literature related to both Advanced Persistent Threat detection and big data Analytics.

Organizations increasingly leverage their Information Systems to glean as much information about their processes, customers, competitors and products. Traditionally organizations have kept only the data they think will bring value to their stakeholders. As computing power increases and storage space becomes cheaper, companies have increased the variety and volume of data. Another trend is that systems are becoming more interconnected. The data would stay and become consumed in one system. With new tools and frameworks data is more flexible and the way that we use Extraction, Transformation, and Load (ETL) tools and middleware technology to move data around between multiple systems. Cloud computing and server virtualization have enabled Platform as a Service (PaaS) providers to enable smaller organizations and give larger organizations flexibility in these various spaces. All these trends have pushed for companies to keep more and more data, which has caused an explosion in the world.

Advances in all these technologies have created the age of big data. Open source infrastructure and analytical frameworks are being utilized to wrangle and make sense of the large amounts of data. Some companies have become strong leaders by gaining a competitive advantage in their industries by using their vast amounts of data in innovative ways. These companies are sharing anywhere from partial abstractions of their systems to access of the source code making lowering the barrier of entry for new organizations or companies with an established reputation to enter into the big data analytics arena fairly quickly.

Motivation

Border penetration detection and prevention is not enough to defend organizations information assets. A majority of the attacks that have been in the media from 2011-2014 were Advanced Persistent Threats that bypassed border security. Two of the more notable attacks in 2014 were against Home Depot [49] and Chase Bank [50]. Analysis of both attacks is in greater detail in Appendix C and Appendix D.

An unknown/unintentional insider threat (UIT) and zero day attacks are raising concerns for organization's information security [51]. An UIT is a classification of insider threat where an insider unknowingly compromises an information system. Research in the information security community focused on this sub classification of insider threat and potential behavioral indicators which can identify UIT is present within an organization [19]. UIT are major concerns for organizations and security professionals that have been focusing on border security because traditional border security does not detect or prevent these advanced persistent threats (i.e., UIT) [45].

Zero day attacks occur between the times that vulnerability has been announced and before a patch has been applied. Hackers are constantly scanning the information on the Internet for vulnerabilities and quickly launch attacks against known compromised hosts (e.g. Apache server that is on an older version with known vulnerabilities). Attackers have been known to penetrate an organization's border defenses, but they may wait to commit any malicious attacks until a zero-day vulnerability becomes known on the target's system [5].

In both attacks on Chase and Home Depot (See Appendix C), the attackers were able to penetrate border security and masquerade with legitimate credentials. Home Depot's attack originated from a third party vendor that Home Depot was partnered with. Attacks that originate from a third party are becoming more common as organizations are embracing cloud computing and buying software over to build software policies.

The number of security breaches for organizations are on the rise. In a 2013 survey conducted by SANS with 647 responses over 65% of the respondents responded that APT was the reason for expanding their Information Security Systems to collect logs and respond to more than just boarder attacks. Multiple APT attacks in 2014 were caused by UIT, see Appendix C. The reasons for not being able to discover attacks were: "lack of system awareness/vulnerability awareness, lack of relevant event context to observe normal behavior, lack of skills/training, difficulties with normalization, and cost of storage and analysis [32].

Machine learning and statistical analysis can be used to help identify attacks. Over the last twenty years data analytics techniques have been used heavily within information security; however, they have primarily been geared towards border controls.

Intrusion detection/prevention systems employ a variety of anomaly based behavior detection using Markov Chains and behavioral analysis [53].

Most of these techniques are employed against network traffic and not application data. In the past sensor (e.g. IDS, IPS or Honeypot) data alone has been very challenging to analyze accurately due to the speed that network traffic generates and analytics systems haven't been able to respond quickly [2]. Also, the amount of traffic causes some issues as hackers have adapted their techniques so that they are attack at a slower rate so that the attack data becomes hidden [8].

With the decreased cost of storage and computational power as well as frameworks such as Hadoop, HBase (which is a variant of Google's BigTable [9]), and Apache Spark some of the traditional challenges in analyzing large volumes of network and security data are overcome [1]. There have been numerous models that have been theorized between 2013 and 2015, which can now be leveraged with the right data analytics tools, but those techniques will bring their own challenges to information security [42].

The literature in 2014 on information security analytics has increased heavily and there were a multitude of books published in 2014-2015 on this subject: Data-Driven Security: Analysis, Visualization and Dashboards; Network security with NetFlow and IPFIX, Information Security Analytics: Finding Security Insights, Patterns and Anomalies in big data are just to name a few [32]. The books mentioned; however, don't really have an answer on solutions to problems using data; they are geared towards an intersection on data analytics and information security. Specifically, the books are geared

towards getting information security professionals “up to speed” on data analytics techniques using the more popular tools.

Even with these books there is still a gap on what to do with the tools and information security data. Due to the vast amount of content released with very little to offer in solutions, it is a good area to try new techniques and think about solving information security threats from a different angle. These tools make it a lot easier to disparate data sources and tying different events to a probable threat using ensemble methods [44].

This thesis is going to use multiple different sources of data to help detect probable unknown insider threats using heterogeneous data sets and data analytics tools that scale – Hadoop and Spark. Detecting UIT has been attempted on a small scale and with simulated data [45]. This experiment is with a Linux Honeypot that is used to collect attacker behavior data and Linux Shell data, and ties the sources together to make attacker predictions. Using a Honeypot can be advantageous because data collected could be a new or unknown attack vector. Doing the analysis alone isn’t enough. The reasons these technologies are used in this experiment are because it’s possible to create an intelligent information system that can act upon the analysis.

Thesis Organization

The following chapters will include information on big data technologies, terminology and topics.

- Chapter 1: Introduction – Motivation and Introduction.
- Chapter 2: Review of Related Literature –The literature is going to review both big data analytics and advanced persistence threats.

- Chapter 3: Hadoop Ecosystem – Hadoop has many components that are constantly being added to and each have their own use cases.
- Chapter 4: Experiment Architecture – The environment uses the Hadoop Ecosystem, and a Honeypot to collect logs.
- Chapter 5: Analysis of the Experiment – Review the results of the Experiment.
- Chapter 6: Conclusion – Conclusion based on the Experiment and recommendations for future studies.
- Appendix – Will contain custom development scripts.

CHAPTER II

LITERATURE REVIEW

The experiment presented within this thesis utilizes big data frameworks on an information security problem. Therefore the literature analyzed for this paper includes both big data Analytics as well as Advanced Persistent Threats in Information Security. There is a lot of information on both topics; however, there are very few papers that combine the two. There are many approaches to security where using big data Analytics is unnecessary. Doing analytics on a large volume of network traffic or even Security Alert Events can glean more information than be yielded by one individual monitoring tool. Implementation of big data Analytics frameworks is starting to be integrated into existing Security Information and Event Management systems, for example at the end of 2013 Zions Bancorporation released attended implementation of Hadoop cluster with intelligence tools to parse through data quicker.

The related literature will focus on both Advanced Persistent Threats and big data Analytics. Both fields are very active and are constantly evolving. First, this thesis will discuss the impact of information security breaches will be analyzed to show that this is a big problem in the real world. Continuing it will also discuss Advanced Persistent Threats and current ways they are handled in research as well as in the industry. Next, this chapter will look at how network security is a big data issue. Big data will be defined and an overview of the various big data analytics tools will be provided. Lastly, the

chapter will highlight on how some of the Security tools have integrated big data appliances into their current frameworks.

Impact of Information Security Breaches

Coverage over large scale Security breaches in the media has been rising significantly in the last few years. Giant retail chains have become the target of numerous security attacks. Another security breach that was highlighted by the media was when former government contractor Edward J. Snowden released NSA architecture used to spy on United States citizens. Cyber surveillance has caused much interest in the general public as many businesses, governments and other institutions have been the leak of consumer information. What makes information security even more of an interesting topic is "it is estimated that as many as 71% of compromises go undetected [8]."

Larger companies tend to find intrusion events compared to smaller firms. Companies with less than $100 million in revenue detected on average of 1,091 incidents, Medium size companies ($100 million - $1 billion) detected over 4,227 incidents, and large companies detected on average 13,138 incidents in 2014 [8]. Cyber threats to smaller and medium sized businesses are a threat to the large companies though, because oftentimes the smaller companies are interconnected with the larger companies that they are trading partners.

Not only are the number of security incidents and threats to organizations on the rise, but the amount it is costing organizations is increasing as well. As more and more security incidents become publicized, organizations are increasing their attention on managing, mitigating and preventing cyber security threats and losses. It's estimated that

"the average financial loss attributed to cybersecurity incidents in 2014 was $2.7 million [8]."

Cyber security threats are not just from external hackers, foreign nation-states, competitors, information brokers, organized crime, etc. Insiders, e.g. current employees, former employees, trading partners and customers are also a threat to an organization's information assets. The percentage of incidents attributed to current and former consultants/contractors was around 18% [8]. With incidents involving Snowden and Manning in the United States military leaking government documents those organizations are trying to understand how to handle insider threats. Organizations have not been monitoring insider threat as much as they have been focusing on outsider's breaking into their networks.

When taking into account insider threats coupled with external threats, information security professionals need to consider more than just the technological approaches to keep threats external to the network contained to the perimeter of their networks. Organizations and security professionals tend to focus on the technological aspects of Information Security [34]. There is research in InfoSec field that takes into account socio-organizational concerns instead of the technical aspect. "Behavioral InfoSec research is a subfield of the broader InfoSec field that focuses on the behaviors of individuals which relate to protecting information and information assets, which includes computer hardware, networking infrastructure and organizational information [13]."

The behavioral information security approach considers separating insider normal behavior from abnormal behavior and understanding the behaviors of external attacks.

With insiders still being a significant threat to information assets, there are two different types of behaviors that are identified "deviant –such as sabotage, stealing, and industrial or political espionage" or "misbehavior – selecting simple passwords, visiting non-work related websites, inadvertently posting confidential data on unsecured servers, or carelessly clicking phishing links on emails and websites[13]." Understanding these different behaviors can improve an organization's understanding on how their employees use internal resources.

Advanced Persistent Threats

An Advanced Persistent Threat (APT) is a targeted attack against physical systems; usually the systems that are targeted have a high monetary value to an organization. Most APT attackers operate in a "Low-and-Slow" mode of operation. "Low" means that they hide themselves within normal network traffic, thus making the traffic appear to blend in with normal traffic. "Slow" means that the attack has a long execution time. These attacks hide themselves and are in for the long-run to break into a system. This is vastly different from the mass-spreading worms, viruses, Trojans and bot-nets. Most of the time, organizations are oblivious to any intrusion done by APTs. Most of the attacks involve stolen user credentials or zero-day exploits to avoid any alerts that the Intrusion Detection Software would transmit [1].

APTs are some of the more sophisticated and serious security threats to an organization today. APTs are becoming even more sophisticated and use a variety of methods and technologies, including Social-Engineering to use an organization's own employees to penetrate the various systems within the organization's firewalls. The major challenge in detecting an APT attack is that there are massive amounts of network data

that are stored in various systems within organizations. Some of the systems include: firewall logs, intrusion detection systems, network switches and routers, application logs, domain controllers and VPN logs. Most of the time logs are analyzed in isolation and not used for correlation purposes.

Zero-day attacks are one of the biggest threats to an organization, due to the fact that most intrusion detection systems need to be pre-configured to detect a signature of a specific attack. Zero-day exploits, by definition are exploits that haven't been identified and the day that they are identified publically is known as "day zero." Criminal organizations look at unpatched software such as Microsoft Office to target vulnerabilities. Two recent Zero-Day attacks that had a lot of media attention were: Heartbleed, and Shellshock. Both affected a majority of Internet Linux systems, because the first uses a bug in OpenSSL, which is used in most open source software and the second is found in Bash, which has been a popular shell for Linux for over twenty years.

There have been a lot of research done recently on the study of zero-day attacks, specifically; Bilge and Dumitras from Symantec have released a study on "Zero-Day Attacks in the Real World." Their research introduced a technique using the Worldwide Intelligence Network Environment (WINE), to identify and detect zero-day attacks [5].

The lifecycle of an attack is that first a vulnerability is introduced into the software, usually it is an overlooked area of the software. If someone outside the organization discovers the vulnerability, it is then exploited in the wild. Once the exploit or vulnerability is discovered by the vendor they normally create a patch for it. When the vendor releases a patch it's normally disclosed publically. After the patch and disclosure happen, anti-virus software and intrusion detection software creates a signature to detect

the exploit being used against an organizations systems [5]. Detecting attacks against an organizations system is hard to do with traditional methods, especially if organizations are relying on their antivirus and intrusion detection systems to detect software only after the signature has been released by the vendors.

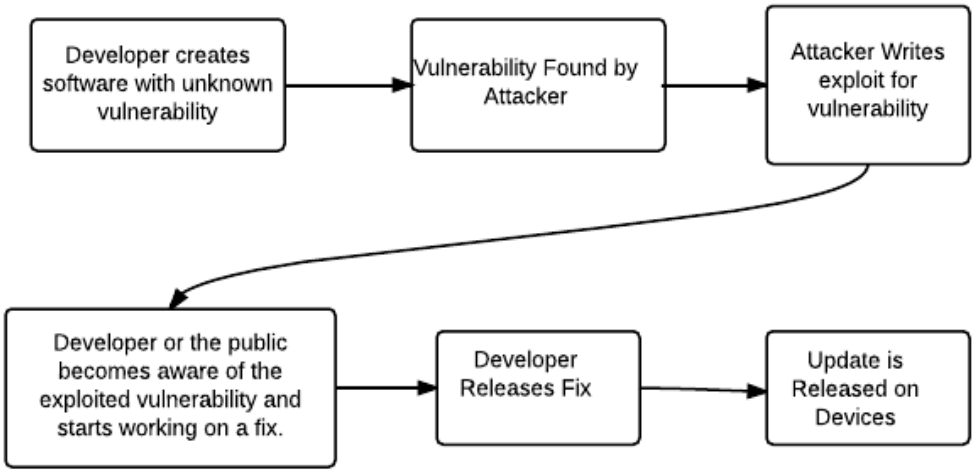


Figure 1. Typical Path of Zero-Day Attack

Honeypots

Honeypots can be very useful when trying to detect any type of malicious activity between firewall domains as well as exposed to the Internet. They allow security professionals to gather data from external sources. They are being used heavily in research in the Information Security Industry as well as academia. Honeypots alone aren't enough to identify and deter attacks, but should be used ensemble as another technology to aid in detection and prevention [54].

Honeypots have been used for two primary reasons: research and production. Research honeypots have been used by the information security and academia in order to understand cyber threats. Production honeypots are what organizations are using to help mitigate risks [26]. These systems can take on many forms from a clone web site to an

interactive Linux shell. There is one organization “The Honeypot Project” that is an international volunteer organization that shares tools and techniques to aid in the research of information security [56]. The tools range from the various types of honeypots to "tcpdump” analysis tools. Overall most of the systems are fairly simple and focus on collecting very specific data sets, and can be used to identify new tools and tactics. One of the challenges is that it is difficult to identify new tactics with honeypots alone.

They can be very useful to identify network sensor data and trends. Honeypot data can be combined within an organization or combining the data with honeypots that exists within other organization as there have been research using honeypots to identify insider threats [54]. Attack trends can be identified by employing a Honeypot within a network, for example - an attack that is originating from an IP shows up on honeypots that are distributed across the Internet. A majority of the research on honeypots are still used to detect external threats.

There are numerous issues and challenges when deploying honeypots, especially at scale. Some of the high level issues with managing and utilizing the data from honeypots are: deploying new honeypots, setting up data feeds into a centralized location, storing and indexing the data in a usable format, correlating the data with heterogeneous data sources and real-time visualization [26]. Some of the advantages are: they use minimal resources, gather only the data that is necessary to analyze a specific problem, they can be simple (Although heterogeneous network of sensors can quickly become complex), and they can help discover new attack vectors.

Detecting insider threats is a combination of information security policy and information security technology. Fine detail user access controls alone aren’t enough to

deter an attacker, nor does it make a system impenetrable. Solving the problem of unknown insider threat takes multiple techniques. Using a honeypot internal and external to model potential attacks can aid in the technological detection of unintended insider threats [55]. Combined with data analytics honeypot behavioral data using ensemble methods for accuracy of detection can aid in identifying attacks [13].

Information Security as a Big Data Problem

The definition of big data will be defined in greater detail; however, there is the question on whether or not network intrusion traffic is classified as big data. There are a multitude of vendors that are starting to provide an extension to connect the data to a big data Analytics framework, most commonly used is Hadoop. These types of frameworks are causing companies to re-think how they store and use data within physical products as well as digital products [25]. Hadoop has been the solution to the issues in order to analyze data. Solving problems that were once not solvable due to the volume, velocity and variety of data is where the real value of these frameworks like Hadoop makes their mark [27].

Some data analytics will not gain any benefit from using a big data Analytics solution and there are techniques that are more than efficient for data analytics on smaller data sets. Based on the definition of big data discussed below, it is important whether or not network intrusion is classified as a big data problem, before trying to apply a big data Analytics solution. Using HDFS along with Public Cloud storage models can be used to store large amounts of data that can be used to analyze network data [20]. The following topology is a recommendation for storing and analyzing network data [35].

Figure 2.2 has three major components: data ingest; user interface and the analytics processing engine. Different data pipelines are heterogeneous data sources across the enterprise. The data is ingested into HDFS sorted by the type of data. The user interface allows for ad hoc analytics or for viewing the variety of data sources. Analysis system can be a mix between real time and batch processing against the various data sources [35].

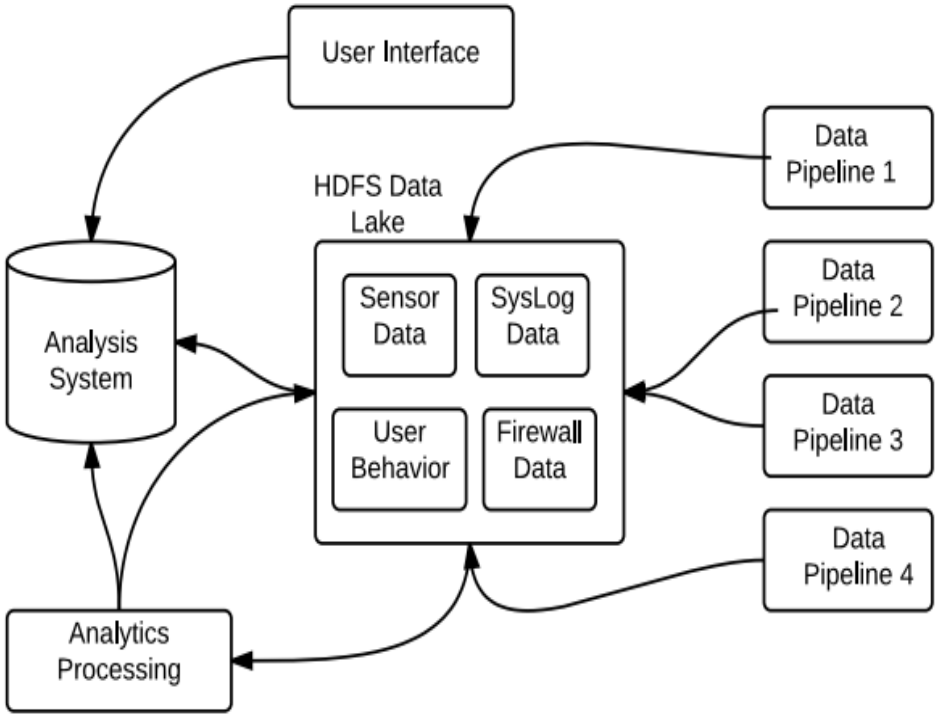


Figure 2. Big Data Framework for Networks

Networking challenges is trying to minimize the communication cost, while storing, processing and retrieving data that will be used in security analytics. Figure 2 that is abstracted to one system; however, in reality the logs come from application logs, operating system logs, firewall logs and intrusion detection logs. The variety of the raw

data that will be analyzed is what separates big data Analytics for security from traditional intrusion detection systems which will store "Events" that are triggered from a set of criterion. Analyzing the data from the various heterogeneous sources can be solved using big data analytics frameworks such as the Hadoop Ecosystem.

Organizations are increasingly collecting more and more security information. The majority of the security logs and event data are collected for various reasons; however the top reasons are: "Detecting and tracking suspicious behavior, supporting forensic analysis and correlation, preventing incidents, and achieving compliance with regulatory requirements [33]." Traditional security platforms are now trying to leverage the power of big data analytics platforms, creating more than known attack vector prevention; however, these systems are still in their infancy. Many of these platforms are currently being used for exploratory data analysis. These systems require security analysts with a background in both security and analytics manually configuring the system. Due to the manual nature of these systems numerous security events go unnoticed.

What is Big Data and How Big is Big Data

Increasing computational power enables organizations to produce so much data that it is causing an explosion in the amount of data that is stored on servers. Organizations are starting to keep data that wouldn't even be considered for archiving. New areas of data creation such as social networking, embedded devices, and smart phones are also contributing to the amount of data that is out there. This overwhelming amount of data created is creating a paradigm shift and what some would consider "the era of big data" [30].

Big data is a buzzword that many people use, but don't really understand the meaning of it. This is a newer term that is used to reference datasets that are so large that they do not fit well with traditional databases, primarily Relational Database Management Systems (RDBMS). Since technology is constantly evolving it's impossible to define "big data" by a specific amount or size of data, but it can be defined where the size of the data cannot reasonably be analyzed on machine, but needs to be distributed across several machines.

A 2010 McKinsey Quarterly has a report, many of the technologies and trends are related to the growth in big data. The trends that are leading to the growth in big data are: Distributed cocreation moves into the mainstream, making the network the organization, the growing of the internet of things, imagining anything as a service, the age of the multisided business model, and Experimentation and big data [7]. Many of these technologies are a byproduct of Cloud technologies. Enterprises are leveraging more and more systems external to the organization's systems.

Big data is discussed in the middle of the article; however, over half of the technologies mentioned in the article feed the growth of big data. Internet companies, such as Google and Yahoo, are the pioneers of big data research and experimentation; however, "Companies selling physical products are also using big data for rigorous experimentation [7]." Organizations are experimenting with social networks, their internal networks, the Internet of Things, and correlated data to gain competitive advantages against in their various industries. As of 2014 these trends are only growing as more organizations are leveraging these various dispersed heterogeneous sources of new data.

Big data is still in its infancy. It is estimated that digital data growth between 2009 and 2020 will grow 44-fold to 35ZB per year [37]. The systems that are processing big data today don't revolve around a single technology, platform, architecture, or use. Although, companies like Google, Facebook, and Twitter seem to be the companies that have pioneered a lot of the technology revolving around big data, other industries such as healthcare and manufacturing are trying to gain a competitive advantage by leveraging big data analytics frameworks, such as Hadoop [39].

Cloud companies in particular have been taking advantage of big data platforms, by optimizing custom environments so that their customers can start using those frameworks quickly. VMWare, for example, optimized their product for big data analytics frameworks. Then VMWare ran a performance study as they want organizations to leverage their systems without the reservation that those frameworks don't run as optimal on cloud based infrastructure instead of specific hardware [6].

There are three attributes of big data, commonly known as the "Three Vs of big data [16]," are Volume, Variety, and Velocity. Figure 2.1 – 3Vs of big data show a quick visual on how the different Vs relate to big data. Gartner analyzed that these are the three areas when qualifying and analyzing big data. Although, the 3V model isn't the only way to view the classification of big data, it is a useful framework that is recognized by many Data Scientists within the field. Information Security data is definitely classified by all three [1].

Often users will cite all three areas when trying to understand the data that is being analyzed by the big data analytics frameworks. The growth in each: Volume, Variety and Velocity of data are the focus behind the MapReduce and Distributed File

System paradigm that many big data analytics frameworks attempt to solve, which will be explored further in this thesis.

Volume is the easiest of the three to understand and quantify. Traditionally data sets have been in the megabytes and gigabyte range. Traditional enterprise volume has come from transactional data being stored into relational database management systems. Newer volumes are coming from non-traditional areas, primarily due to Internet technologies. Log files that used to be archived off or deleted are now being stored in permanent data stores. Before current data analytics platforms too much volume has been an issue for analytics departments. According to a study by Booz & Company, structured data volumes are continue to rising; however unstructured data makes up 80% of the world's data [28].

The second V is Velocity. Velocity refers to how fast a data is being produced as well as how fast the data must be analyzed to meet demand. When the data is being produced faster than it can be analyzed this is when new technologies and algorithms need to be used in order to process the data. Legacy systems have primarily processed data in batch. Recent trends using Internet technologies have begun to process data while it aggregates into the system, e.g. streaming data analytics. Streaming data analytics provides challenges for even the more popular data analytics infrastructure frameworks.

The final and most complex portion of the 3V model is Variety, which refers to the type of data that is being stored. There are three categories on the Variety of data. Structured data means that the data is modelled and will be in the expected format. RDBMS uses data normalization techniques in order to reduce redundancy and dependency; however, this makes the data very rigorous and structured. RDBMS data

may have some complexities when data scientists need to run analysis on the data; but there is a standard query language, SQL, that enables data scientists to be able to mine data from a RDBMS, due to the structured nature of the data [16].

The second category of Variety, semi-structured data, would include data formats such as XML, JSON, and video data. There are other "standard" document protocols including EDI and Rosetta-net, that would also be considered semi-structured. Data in this category has a structure to it, but can be very flexible on how the data is structured from one instance to the next. Semi-structured data can still be mined, but it takes a lot more work to run analysis against the data than against an SQL database. Due to the ubiquity of JSON, XML and EDI there are many commercial and open source parsers that make it easier for data scientists to sift through the semi-structured nature of the data.

Unstructured is the third category of Variety, and is the most difficult to do data analysis against. Companies harnessing the power of big data analytics frameworks have been trying to build systems that make analysis on unstructured data easier. This type of data can include: text files, logs, social media posts, telematics information, excel documents, pdf documents, and photos. Unstructured data have been mostly ignored until recent advances in technologies. Organizations are now trying to figure out how to gain a competitive advantage on the unstructured data that is being thrown away and could potentially provide a competitive advantage [28].

Organizations are not gaining a competitive advantage with just one type of data. They are now trying to see how the various sources of unstructured data relate to their structured and semi-structured data. Unstructured data alone will not provide value to

organizations, but will provide value in parallel with their current data warehouse and data analytics systems [11].

Gartner's three V model for defining big data Analytics is just one way to define and model big data analytics. Another model is the HACE Theorem. "HACE big data starts with large-volume, heterogeneous [H], autonomous [A], sources with distributed and decentralized control, and seeks to explore complex [C] and evolving [E] relationships among data [43]." According to Wu, these characteristics help determine the challenges for making useful information from big data. There are some overlap between the HACE theorem and the 3V model, which will be examined in the next section.

Heterogeneous data relates to Variety of data in the 3V model. The different types of data arrive because each information system has its own schemas and protocols. Due to the variety of data, it makes it difficult to for analytics systems to make sense of the data. One of the major challenges of big data analytics frameworks is to overcome the heterogeneity and diverse dimensionality issues when aggregating the data into a meaningful output.

Data in modern information systems are generated from distributed and decentralized areas. The Internet has proven organizations do not need to rely solely upon the data that resides within their infrastructure. Each source needs to generate and collect data without any command center. Autonomous sources are necessary in order to solve many of the technological challenges of distributed systems and scaling.

Evolving relationships underneath the data become even more difficult as the volume and variety of data increases. Finding and connecting relationships becomes one

of the major challenges in Data Analytics systems. This challenge is amplified as organizations are consuming more and more data that is created and modelled external to their organization. It is important to take these key characteristics of the complex and evolving data into consideration when trying to gain insight into diverse data sources without clear relationships.

Even though the HACE theorem has some overlap with the 3V model, it is very useful when analyzing the characteristics of big data. Using the HACE Theorem, when mining big data it's important to take in to consideration three separate properties of the big data systems: data accessing and computing, data privacy and domain knowledge, and big data mining algorithms [43]. These three areas will be discussed in the Data Mining and Algorithms section of this thesis.

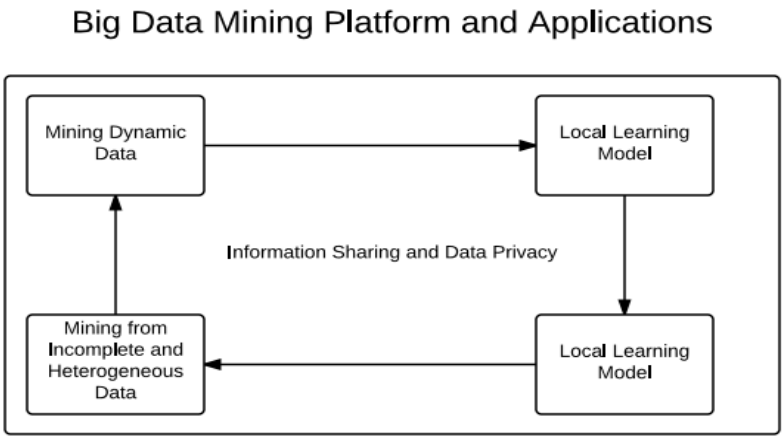


Figure 3. Big Data Processing Frameworks

Data Mining and Algorithms

Business Intelligence and Analytics have gone through three major generations based on how the data is approached. Chen, Chiang and Storey wrote a paper for MIS Quarterly – December 2012: "Business Intelligence and Analytics [11]: From Big Data to

Big Impact," which discussions the evolution of Business Intelligence and Enterprise analytics. At the end of the article the authors include a table of emerging research in the field. Although research in artificial intelligence and data analytics has been around since the 1950's, Business Intelligence didn't become a popular term until the 90's. More recently big data and big analytics have been used popularly due to the volume, velocity and variety of data that are being stored within enterprises.

BI&A 1.0 has been around for as long as database reporting. Data in this category are mostly structured, collected through legacy systems. Data is often stored on the file system or a traditional Relational Database Management System (RDBMS). Many of the statistical techniques in this category have been around since the early 1970s and have been heavily used up through the 1990's. Data-marts and data-warehouses have been designed to aggregate and analyze the data. Extraction, transformation and Load (ETL) tools have been used to convert and integrate the data into a specified format.

BI&A 2.0: Web technologies during 98 – 2003 caused an explosion in the amount of data generated by organizations. Internet technologies presented a unique way that data was being generated, stored and analyzed. Companies such as Google, Yahoo, Amazon and eBay, began taking advantage of data that fell outside the traditional way of storing and analyzing the data. Web Intelligence, web analytics, and user-generated content through Internet Applications centered on unstructured text and created challenges and opportunities for the analytics community. The biggest difference from BI&A 1.0 and BI&A 2.0 are the amounts of unstructured data that cannot be easily stored and analyzed within a RDBMS. Web 2.0 applications that generate the data include: social media, forums, web blogs, social networking, socio-political comments, images,

video, and voice. Customers are creating content that can now be searched, where previous analytics systems analyzed data created by the enterprises.

BI&A 3.0: Identified around 2011 due to the number of mobile devices including phones and tablets surpassing the number of Laptops and PCs. The number of people always connected to the Internet ushered a momentum in the amount of data created. Software is in everything we do at this point. There are complete software packages for multiple domain areas, from mobile games to enterprise application software to social media. Radio Frequency Identification (RFID) barcodes and radio tags have helped creates a trend known as "the Internet of Things (IoT)." These devices are in the cars people drive and can even be inside of wearable devices that are used to track a users' progress in their fitness routine. The underlying BI&A in this era will stem around mobile analytics with location and context aware sensors. Analysis will be able to visualize and bring together data from multiple sources [11].

The third column shows the emerging research, which is current day. According to Chen, et al. the major areas of research are: big data Analytics, Text Analytics, Web Analytics, Network Analytics, and Mobile Analytics. Software defined networking and newer network technologies have been embracing big data frameworks to analyze data that would have once been considered too large to analyze as a whole [32]. A majority of these emerging research areas stem from Internet Technologies. These areas aren't just being researched at Universities or just for experimentation, but are being commercialized by industry and government. Some utility companies are starting to collect sensor data that enable predictive and prescriptive analytics for consumers [12].

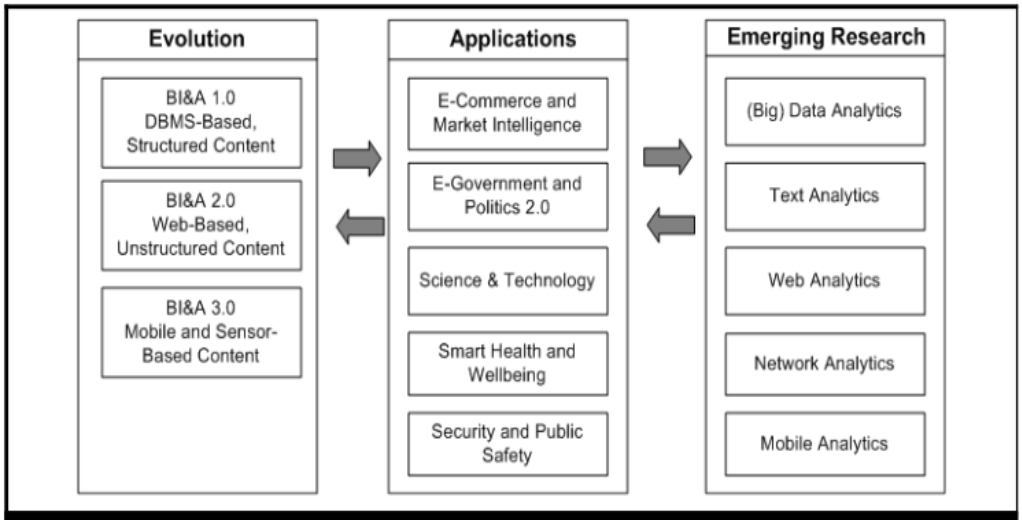


Figure 4. BI&A Overview: Evolution, Applications, and Emerging Research [11]

Figure 4 represents the Foundational Technologies and research in those various areas. Chen, et al. provided multiple areas based on the opportunities of emerging and exciting applications that have shared excitement in the various research communities. They classify the research opportunities into five critical technical areas.

Big data analytics include running algorithms such as: cover classification, clustering, regression, association analysis and network analysis over data sets so large that big data analytics frame works such as MapReduce are required in order to analyze the datasets [23]. Text analytics decimate context out of user-generated content based on computational linguistics and statistical natural language processing.

Web analytics are based on HTTP and HTML typed information as well as web page updating, log search analysis, and newer areas based on web services and their Application Programming Interfaces (API). Network analytics have evolved to include link networks and community networks. Network analysts have been trying to depict various social network theories and topologies based on connections [36]. Finally, mobile

analytics are based around analyzing the various components of the Internet of Things (IoT). Each of these areas are not just limited to BI&A 3.0, because as time goes on new techniques and algorithms have been crafted to take advantages of the various systems.

	(Big) Data Analytics	Text Analytics	Web Analytics	Network Analytics	Mobile Analytics
Foundational Technologies	• RDBMS • data warehousing • ETL • OLAP • BPM • data mining • clustering • regression • classification • association analysis • anomaly detection • neural networks • genetic algorithms • multivariate statistical analysis • optimization • heuristic search	• information retrieval • document representation • query processing • relevance feedback • user models • search engines • enterprise search systems	• information retrieval • computational linguistics • search engines • web crawling • web site ranking • search log analysis • recommender systems • web services • mashups	• bibliometric analysis • citation network • coauthorship network • social network theories • network metrics and topology • mathematical network models • network visualization	• web services • smartphone platforms
Emerging Research	• statistical machine learning • sequential and temporal mining • spatial mining • mining high-speed data streams and sensor data • process mining • privacy-preserving data mining • network mining • web mining • column-based DBMS • in-memory DBMS • parallel DBMS • cloud computing • Hadoop • MapReduce	• statistical NLP • information extraction • topic models • question-answering systems • opinion mining • sentiment/affect analysis • web stylometric analysis • multilingual analysis • text visualization • multimedia IR • mobile IR • Hadoop • MapReduce	• cloud services • cloud computing • social search and mining • reputation systems • social media analytics • web visualization • web-based auctions • internet monetization • social marketing • web privacy/ security	• link mining • community detection • dynamic network modeling • agent-based modeling • social influence and information diffusion models • ERGMs • virtual communities • criminal/dark networks • social/political analysis • trust and reputation	• mobile web services • mobile pervasive apps • mobile sensing apps • mobile social innovation • mobile social networking • mobile visualiza-tion/HCI • personalization and behavioral modeling • gamification • mobile advertising and marketing

Figure 5. BI&A Research Framework: Foundational Technologies & Research [11]

Algorithms Used for Security Data Analytics

Information Security has long used various data analysis techniques to combat cyber security threats. Intrusion detection systems have long used machine learning techniques. The security community is now regularly employing more and more machine learning techniques into the various suites. However much of the book is geared towards introducing information security professionals how to use Analytic languages, such as R.

Intrusion Detection Systems have employed various supervised and unsupervised learning techniques. Signature based techniques have been commonly been employed

and rely on human accounts or application accounts to deploy signatures. Anomaly detection intrusion detections use machine learning techniques that can learn from those behaviors and can give unseen decisions [53].

The various machine learning techniques include, but are not limited to Neural Networks, Support Vector Machine, Decision Trees, Genetic Algorithm, Fuzzy Logic and Bayesian Networks [24]. Decision trees are used to classify a set of data based on previous signatures, but can be limited in detecting new attacks. Genetic algorithms are used to find approximate solutions to a task, and have been used in IDS based on past behavior [53]. Fuzzy logic can be used as a classification system when detecting intrusions that rely upon predefined rules; however it has known to have issues on large datasets [11]. Bayesian Networks are models that predict probably outcomes using prior knowledge; however does not handle new features well. Bayes doesn't work best by itself, but as an ensemble methodology alongside other predictive analytics [44].

Machine learning for intrusion detection has received a lot of attention because of the high promise to learn from the data [10]. As industries become more focused on cyber security the security community is going to focus more and more on machine learning techniques to help identify new attacks. One of the problems with the techniques that have been employed is that they are learning-based systems. Many of the algorithms mentioned above rely on known attack vectors. Attackers can use that knowledge and obfuscate their attack methods.

One well known method of machine learning is behavioral analysis. The main algorithms are supervised learning and classification algorithms. The two main approaches are Naïve Bayes and Hidden Markov Models [57]. There are well known

datasets that have been used for detecting masquerade attacks on UNIX shell commands. One of the more popular datasets is Schonlau masquerade data set consisting of over 50 users [58]. This data set has been used widely in the information security community, but has limitations since it only keeps track of the command and not the arguments [59]. It is unclear as to whether the dataset contains commands that are "piped" or call external commands. Another limitation of the data set is that it is missing enriched data. According to Maxion, "the detection of attacks increases 18% to a 70% detection rate due to unusual flags used by attackers.

Big Data Security Analytics – Bringing It Together

Current tools focus on intrusion prevention. It is good practice to do so, but it does not stop intruders. There are security tools, but each has its own command center and its own separate analytics systems. Current applications are focused on securing the perimeter; including firewalls, IPS, and gateways. There is little to zero coverage for advanced malware based upon zero-day vulnerabilities. This is not alluding current security environments are not protective, which can prevent known and common attack vectors; however, enterprise security following only these guidelines are growing more and more effective. Presently there are a large amount of false positives, and there are too many tools and manual processes for incident detection.

Big data security analytics will change the way we look at security incidents and response. With Big Data analytics can do automated intelligence thinking and can correlate the data from all these heterogeneous sources. Big Data Analytics can help centralize security data monitoring and visualization and storing the data in one place, and in its raw format. Using Big Data Security Analytics, organizations can use dynamic

self-learning methods needed to detect sophisticated attacks with quicker response times than are currently available in traditional security detection [38]. The detection goals for a big data security analytics for advanced persistent threats are to detect the existence of the attack in the past, understand the goal of the attack and identify the source.

Big Data analytics can take advantage of web server logs of all incoming, outgoing traffic, IP address, times, and session durations. Domain controllers can offer user IDs accessing specific IP addresses, times, durations. Content servers can offer a detailed account of customer data and access control lists.

Organizations are already connecting large amounts of data, but they are being removed after an allotted amount of time. The data could be collected into one repository before being discarded. Advanced analytics can be applied in various areas in security: Detect that organization was attacked, find out what the goal was, and then understand where the attack came from. There is no single detection algorithm to cover all areas of big data security analytics. The issue with APTs is that most information security analysts feel vulnerable to the multitude of vendor software their organizations leverage [39]. Using analytics on the large volume of network and application data, information security analysts will be more prepared to predict and respond to potential attacks.

There are multiple areas where security can leverage big data analytics to increase security responsiveness within an organization. The use cases can include: "External malware-based threats, Advanced Persistent Threats, Compliance Monitoring, Insider Threats, Risks and compliance management, Reporting on real-time data, Reporting on historical data, correlation rules/patterns, fraud detection, and improved queries and reports [34]." Since the application of big data security is a new field there are a

multitude of research opportunities in each of these areas. Fraud detection has been used in the financial space; however, there is much promise in all these use cases depending on the organization's needs.

CHAPTER III

HADOOP ECOSYSTEM

Apache Hadoop was created by Doug Cutting. Originally the work was on a project called Apache Nutch, which is an open source web crawler. The challenge was to be able to store the massive amount of data and have the capability to process the data. After reading the 2003 - Google File System and 2004 – MapReduce papers, Cutting was inspired to combine the two systems into one combined system that could be handled as a data processing and storage framework. Hadoop's core, known as Hadoop Common, has the HDFS, Hadoop Distributed File System, and the MapReduce framework [41].

Google File System

Google's Research and Development community is constantly releasing new research papers in various areas. Their goal is to collaborate with the broader scientific community and make a contribution to various fields. Their R&D team consists of multiple researchers that hold a Ph.D. with various backgrounds in Computer Science, ranging from Algorithms to Distributed Computing. These teams are used to develop new products from a research to implementation on Google Infrastructure. In 2003, 2004 and 2007 Sanjay Ghemawat, et al. and Jeffry Dean released papers that changed the way computer architects designed big data applications, which will be reviewed below.

A paper titled "The Google File System" [18] was released by Google R&D department in 2003. The file system, GFS or GoogleGFS, is a distributed file system that

is designed to provide reliable and efficient access to data using a large number of commodity hardware. The file system is implemented at Google in a proprietary way, however since the release of the paper there have been several attempts to engineer a similar system based on the qualities that were released in the paper.

GoogleGFS was implemented to increase the efficiency and reliability of Google's Search as their processing needs were rapidly growing at the time. Scalability, Reliability and Availability running on commodity hardware of their distributed file system are the core tenants of their design. The design assumes that it is built from many inexpensive commodity hardware components, stores a large number of large files, low writes, high reads, and have an easy to use user API [18]. There are key ideas from their paper that have translated well to big data analytics frameworks.

MapReduce Distributed Computing

Google released another research paper to the public at the end of 2004. This time it was based on a programming model used to process large datasets on a large cluster of commodity hardware. At MapReduce's core it is based on functional programming style paradigm where the processing doesn't deal with state variables, but only on the inputs and outputs associated with the functions used to process the data. Programs written this way are automatically parallelized, distributed, and executed across a large cluster of servers. The paper didn't just propose MapReduce as a theoretical programming paradigm, but stated that "thousands of MapReduce jobs were submitted through Google each day" [14] [15].

This system was designed due to the growing complexity in the amount of data being stored, and processing was done through very specialized systems for each task.

The goal of MapReduce was to abstract out the common processes and hide many of the complex programming details such as: parallelization, fault-tolerance, distribution, and executing and compiling code across multiple systems. The core of the system is based upon the _map_ and _reduce_ pieces of code inherent in the Lisp Programming language.

Google's implementation of both the GFS and MapReduce Frameworks are written in C/C++. Hadoop is written in Java and provides an API to all major components through their Java API. There are some inherent performance debates due to the Hadoop being written in a JIT language compared to C++ which is compiled to machine code.

HDFS is primarily designed to run on large amounts of commodity hardware; however, there is multiple cloud organizations that are offering Hadoop as an online service, where users can pay for their processing by the amount of CPU and memory used. The file system is very similar to existing distributed file systems; however, it is more fault-tolerant. It is also designed with similarities using the file system syntax that POSIX file systems use and modifies a few of the interfaces.

The File System is built upon the same Master/Slave architecture as the Google File System paper presented. The cluster consists of a single NameNode, the master server, which manages the file system and access to files by clients. It is possible to create a backup NameNode in case the primary one fails. Files are stored into blocks across multiple DataNodes. Data is replicated to multiple DataNodes creating redundancy and improving fault tolerance, the default replication value is three separate data nodes. The metadata for each file is stored in the NameNode in memory, but also a log file is created for the NameNode for when the system reboots, it can load the metadata back into memory.

MapReduce is the second core component of the Hadoop common. Both the HDFS and MapReduce run on the same systems, that way the processing is loaded from disk on the hardware directly into memory, which increases performance of data processing. There is Master/Slave architecture to the MapReduce framework. There is one primary Resource Manager and each node will have its own Node Manager.

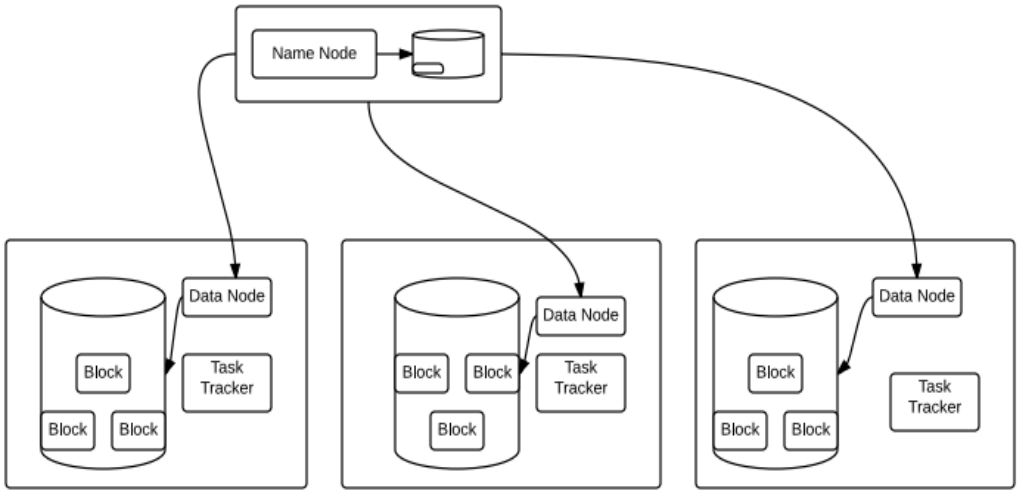


Figure 6. Overview of Hadoop Distributed File System

The client will submit a job, the Resource Manager will talk to the NameNode to get the location of the data that will be processed then submit the job to the Node Manager to pull the data from HDFS. Next the Node Manager will launch a Java container that will start the MapReduce tasks.

The first phase of MapReduce is mapping. Each line of input will be processed individually to a function called the Mapper, which will perform some a transformation or calculation upon the data and then output that data element. The Map phase takes an input pair, and produces a set of key/value pairs. Each output will be sent to a Reducer. Map takes something from the record that is meaningful to the client and shuffles and sorts the result, then hands it off to the reducer.

The reduce function receives the key/value pair and merges the values with the key and merges them together. Output from the reduce phase can be sent to a file on the HDFS, or it can be sent through another iteration of the MapReduce phase to further analyze the data. It is possible to send the data through MapReduce several times before producing the desired output, whether to do some massive calculation or transform the data. Reduce aggregates, summarizes, filters, and transforms the data. The data then can be sent to another MapReduce job, a file or the output screen.

Both Map and Reduce are done in parallel and are controlled by the Master node. Each line of input is split into *X* map tasks into chunks of a default size, set in the configuration. Reduce phase partitions *Y* reduce tasks. Tasks are designated to workers dynamically. Each worker sends a heartbeat job back to the Master, and if for some reason a worker is knocked offline and the master node doesn't receive a heartbeat back from the worker, then the master will designate a new worker for that previous task. The master node can consider the location of the data so that the worker can read from disk on the same node that the data resides. Since the Master Resource manager monitors the workers tasks, the MapReduce portion of Hadoop has built in reliability in case one of the worker nodes fails for whatever reason.

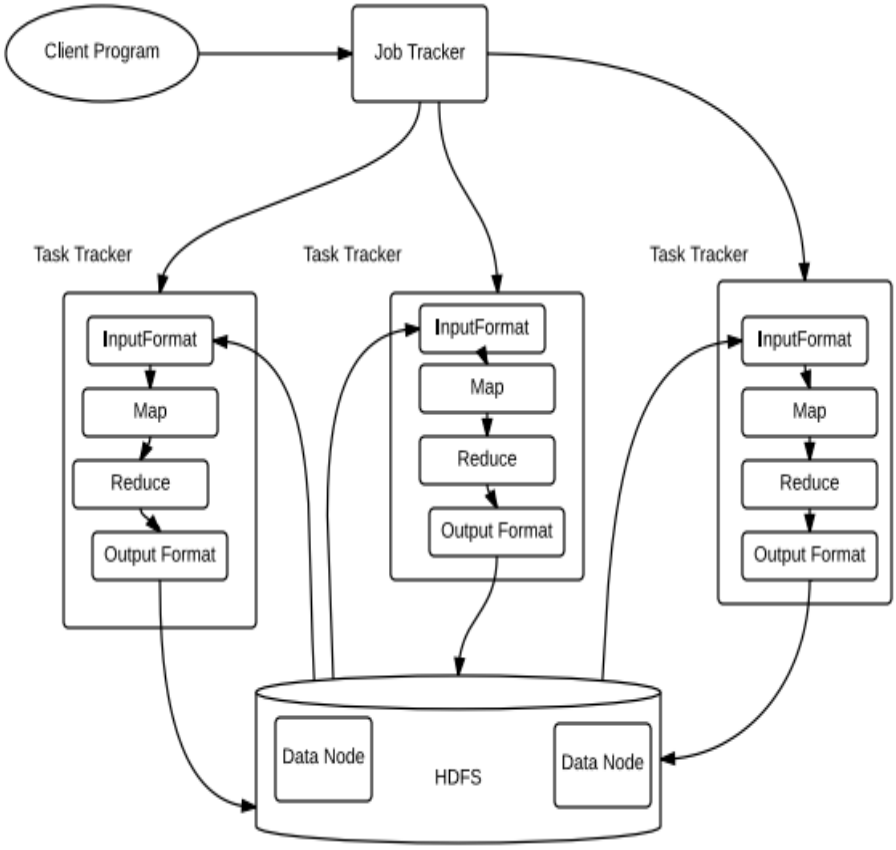


Figure 7. Overview of Hadoop MapReduce

MapReduce jobs are powerful, and are the heart of Hadoop; however, they are very difficult to write. Hadoop is referenced as an ecosystem is due to the number of external products that have been developed as add-ons to Hadoop. Yahoo, Facebook, Twitter, and other early adapters have contributed to the ecosystem of Hadoop in some way. Pig, Mahout, R Connectors, Hive, and Oozie are perhaps the most popular out of the different components of the Hadoop ecosystem. Pig and Hive allow for abstractions built on top of MapReduce which allow for a non-programmer friendly environment [17]. There are many smaller products, or specialized products, such as Storm, which is Twitters real time analytics engine. Hadoop is still going through a growth stage and there will most likely have a number of new products in the near future.

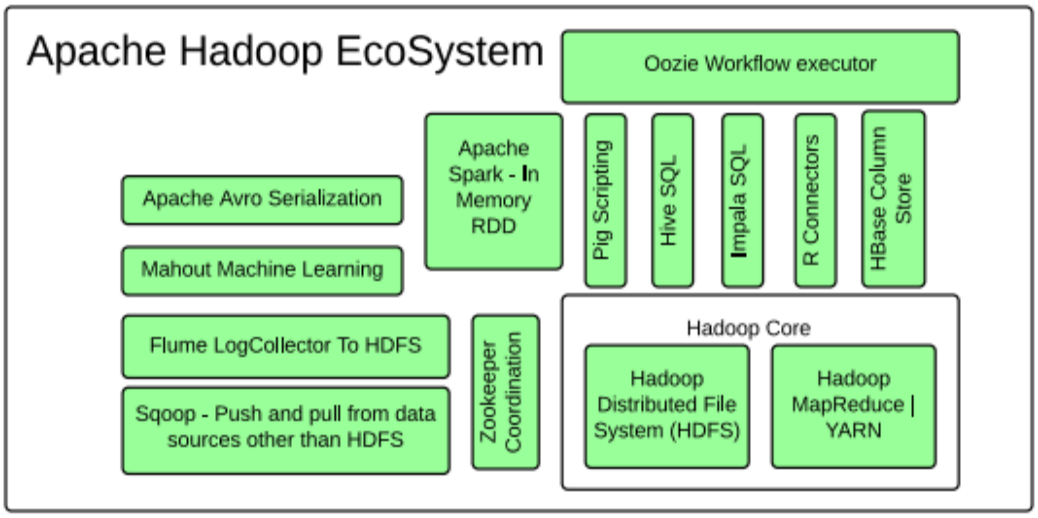


Figure 8. Apache Hadoop Ecosystem

Apache Spark

Apache Spark is a high level project that utilizes the power of the Hadoop Framework. There are some limitations that are inherent in MapReduce [3] that Spark was built to overcome. Spark does not leverage the MapReduce paradigm, due to some I/O performance gaps between each MapReduce Job. It uses the Hadoop file system API for persistent storage and retrieval; however, Spark has been designed with abstraction in mind and has ensured that the framework is not tightly coupled with Hadoop and can potentially be connected to future distributed file systems. At this time the current projects that are included in the Spark community are: Spark, Spark SQL, Spark Streaming, MLib (Machine Learning), and GraphX (Graph based system). Spark can help solve some of the issues that are apparent when applying Machine learning techniques to Information Security data [42].

The internals of Spark are based on a distributed data abstraction introduced at Berkeley, known as Resilient Distributed Datasets (RDD). RDDs are fault-tolerant abstraction for In-Memory Cluster Computing. They are immutable, created through

data from storage or from another RDD that has been transformed in some way. When an RDD is created from another RDD it is through in-memory processes and not stored on disks like MapReduce. This has an advantage over the MapReduce processing that is inherent in Hadoop. The immutable nature of the distributed dataset is that the transformation and actions performed on them must be applied to every element in the dataset. The disadvantage of this abstraction is that working on a datasets that require mutable state will not be fully realized in this type of system [46].

Spark's advantage is the speed in which it can process data. There have been numerous benchmarks applied to compare Spark to Hadoop's MapReduce, that have shown if the algorithm is immutable and functional in nature that the speeds can be up to 20-40x faster. Spark's computational model allows for quicker real time data analytics [40]. The use cases that have been applied with Spark: Conviva Inc., was able to do large In-Memory Analytics took 30 minutes versus 20 hours in Hadoop; Mobile Millennium was able to model traffic with GPS sensors; and Twitter was able to do Spam Classification algorithm, which was able to identify links as spam.

Discretized Streams (D-Streams) is another model that came out of Berkeley; which allows big data applications to process data arriving near real time, which overcomes a major challenge in the data analytics community for real time analysis on big data [4]. D-Streams allow for fault-tolerant and consistent streaming programming model that helps overcome many of the challenges when analyzing data in real time. D-Streams build on top of RDDs that partition a series of deterministic batch computations on small time deliverables [47]. Spark implements both D-Streams and RDDs to allow for scalable real-time analytics processing systems. Currently, D-Streams should handle

time intervals greater than half a second. If the requirements are to handle the data in sub second intervals, then another technique should be applied.

CHAPTER IV

ARCHITECTURE

Current methods of detecting and following up with security threats are not enough in with today's security threats [1] [10] [62]. Gone are the days where our perimeter defenses are enough to deter attacks. Hackers are using a combination of Social Engineering and Security know-how to break into organizations systems. Attacks are now targeting specific systems, software and hardware. There are various techniques that can be used to analyze security threats, that haven't been a possibility in the past due to the large volume and variety of logs.

The following is a suggested architecture for future big data Security Analytics framework. With the rise and research poured into big data Analytics, it is now possible to leverage algorithms and strategies that were not available before. This framework is not intended in replacing traditional security tools, but will add another layer of detection. Advanced Persistent Threats tend to happen within the security perimeter and therefore; this technique will compliment traditional perimeter security defense.

Experiment Setup

Behavioral analysis and honeypots can be used on a variety of systems (e.g. website data, fraud detection, windows users, Linux ssh session). This experiment will be focused on using an SSH Honeypot to collect attacker behavior. User data is going to be collected using "pacct" and "bash_history."

There are some limited public datasets available; however, it is difficult to find SSH honeypot data. Due to the limitation of open public honeypot data, the experiment will include a Honeypot to collect data from users over a one month period. Bash profile data is widely available. A popular data set used for detecting masqueraded attacks has been used to analyze behavioral data [58]. A disadvantage of this data is it only shows the commands without the arguments. Hackers target vulnerabilities within a system and the arguments of a command can help determine if a user is doing something malicious.

Once the datasets have been collected they will be placed into HDFS in their raw format. All the analysis will be done by cleaning and transforming the raw data into the expected data using regular expressions. Every command to transform the data will be in the Appendix and all data will be stored within HDFS so that the experiment is reproducible using the same datasets.

Using Naïve Bayes, the datasets can be broken into two classifications: bad actor and good actor. The first 5000 commands of each of the user accounts are good actor behavior and honeypot data is bad actor behavior. One of the limitations with these data sets is that it only has the first command and doesn't have any enhanced fields, such as flags or arguments passed into the commands [59].

Raspberry PI Honey Pot

Raspberry PI is a miniscule computer that was originally targeted for a young audience. This project started back in 2006 at the University of Cambridge. Originally the idea was to provide a small affordable computer, twenty-five to thirty-five dollars, for kids to learn the principles of computer science before attending a University. The Raspberry PI project has spawned off its own community of developers sharing ideas on

how utilize the computational power of the tiny credit card sized unit. There have been many projects that can show the different types of projects for Information Security.

The Honeypot that was used in this experiment ran on a Raspberry PI model B+ with the CanaKit running a Linux based OS. The CPU was 700 MHz with 512 MB of memory with a 4 GB hard drive. Raspberry PI as a honeypot was a good option due to how inexpensive it was to implement and that it has a low power usage footprint. With this set up it would be easy to install multiple sensors fairly cheaply to collect data from multiple points.

Tweaking the system (/etc/proc) files allowed the system to look like it was more powerful system using 48 cores and 128 GB of ram. The system was exposed to the internet through port forwarding SSH traffic from a Cisco router that is connected to the Internet. The default port for SSH is 22; however the system was configured to accept SSH traffic through port 4576. Kippo's documentation suggests listening to port 2222; however, it is not a recommended setting due to the popularity of using this port for Kippo and other SSH Honeypots.

There are multiple open source honeypots that are readily available. This project the SSH Kippo was chosen. Kippo is a simple Python script that emulates a shell. It has been commonly used for monitoring brute force attacks and also emulates "most" shell commands. A primary reason this system was chosen is due to its ability to mimic the UNIX file system that allows for fake file contents to commonly used files (e.g./etc/passwd and/etc/proc). Even though Kippo mimics POSIX file systems fairly well, it is worth doing some customization in order to lure attackers to stay longer.

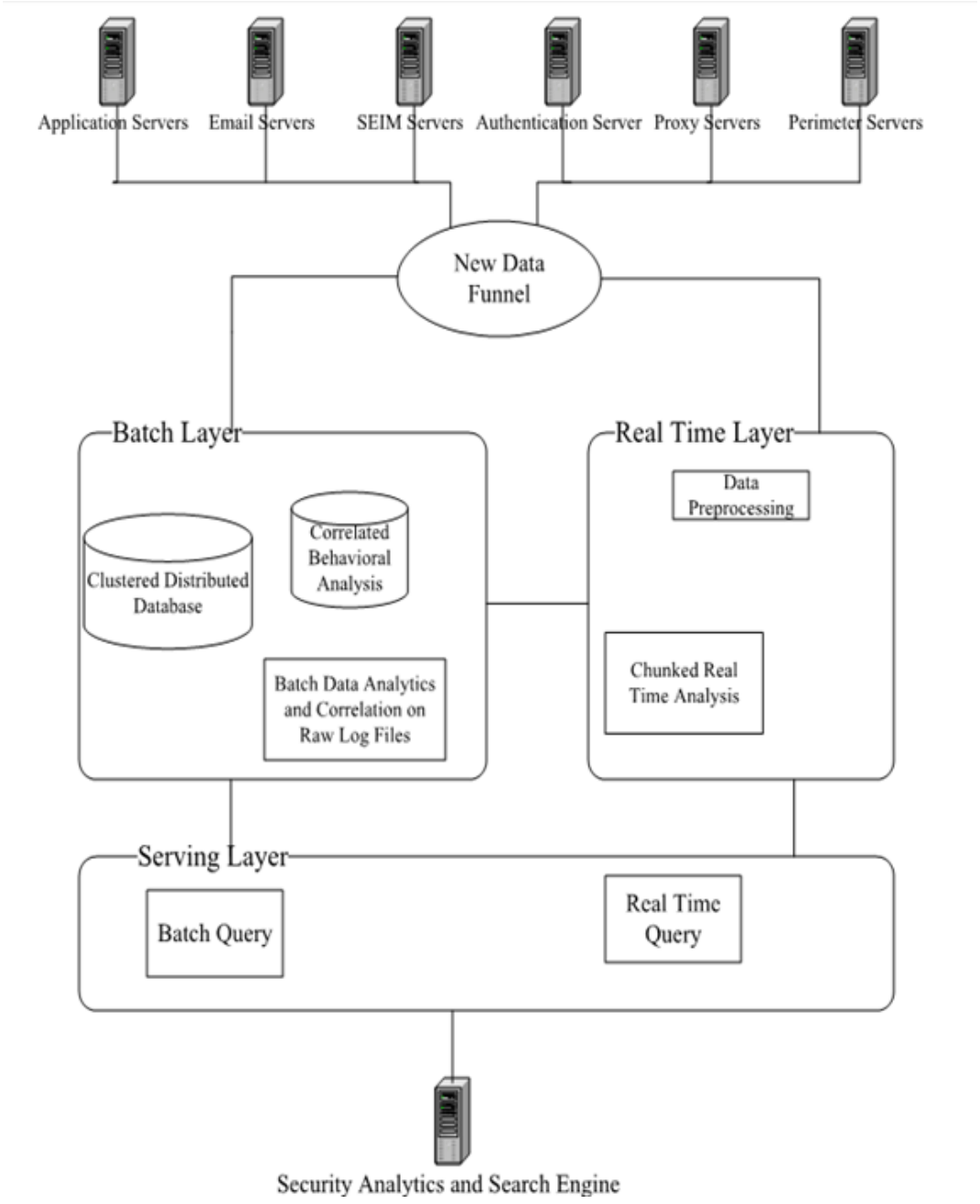


Figure 9. Big Data Security Analytics Architecture

Big Data Architecture for the Experiment

Figure 4.2 is a model that can be abstracted out for various big data Analytics systems. However the diagram depicts the various sensors that may be present within a network system that provide traffic logs of various systems and users within an organization. The goal isn't to collect every log available, but to actually bring value from the potential insight provided by those log data [22]. This diagram could easily include the organizations firewall boundaries so that we can have different areas within a network being logged.

This experiment used 4 Virtual Machine (VM) servers. Each server had 4 CPUs with 2 cores per socket. A total of 16 GB for the Name Node/Master server and 8 GB RAM for each of the slave nodes were allocated to the VMs. The master server needs more RAM than the slave servers because that server keeps track of where all the files are stored within HDFS and is the master node for any Spark jobs which can be memory intensive processes. Each of the nodes only had about 128 GB of Hard drive space each, this is primarily due to testing purposes. This allowed for over 500 GB of total disk space. This amount is not really enough to be considered big data, it is enough to enable proof of concept and development as both MapReduce and Spark scale really well [48].

The servers are running Ubuntu 14.04 with several dependencies. The "/etc/host" files on each server need to have a row inserted for each server in the cluster. Whenever a server is added to the cluster the "/etc/host" file on each server should be modified. It would be worth containing one master file that can be distributed across the cluster so that new servers can be added with ease. Each node has the same configuration for

Hadoop and Spark the configuration files are updated on the master server, but are distributed to the other nodes within the cluster.

There are several dependencies that each server has installed. Apache Spark is built on top of Scala 2.10+, Python 2.6+ and Java 6+. It's possible to program in any of the three languages, but was written primarily in Scala with a functional programming paradigm. Python is a popular choice due to the number of advanced analytic libraries like Numpy and h5py. HBase is another high level apache project that was installed; it can be used as a columnar data store. The primary packages are described in further detail in Appendix B.

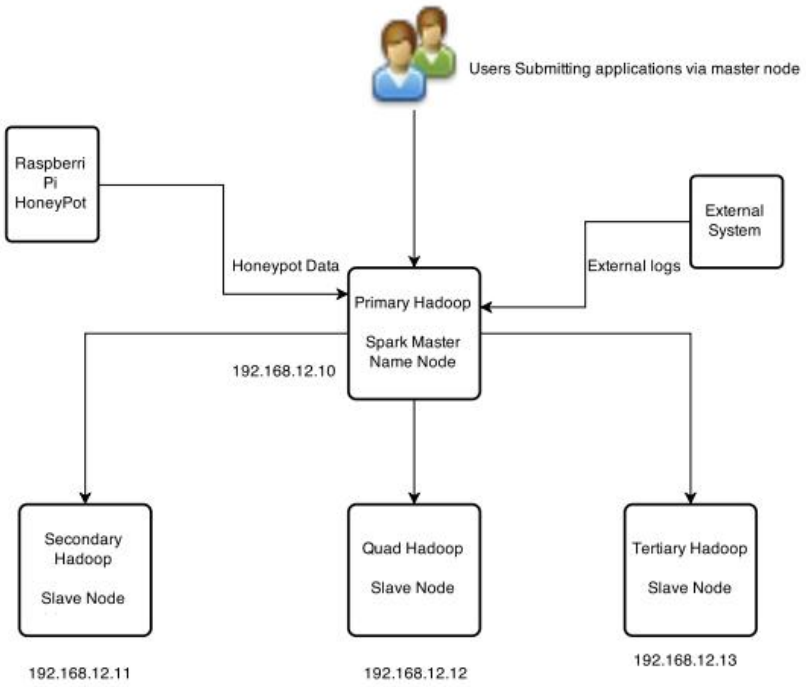


Figure 10. Hadoop / Spark Experiment Architecture

Figure 10 is the infrastructure setup for the cluster set up for the experiment described in chapter 5. The overall high level view of the architecture on how the different systems are connected. Every system uses port 22 to move data around. The internal Hadoop/Spark system uses password less SSH to communicate with one another.

Figure 11 below is the overall analytics architecture. Honeypot data is continually monitored and updates the Honeypot Behavior model. User historical command data are continually updated within HDFS. If the data isn't considered bad the user profile model needs to be updated with the new data. The method checks user model behavioral analysis against both its own previous data and known bad actor behavior that is generated by honeypot data.

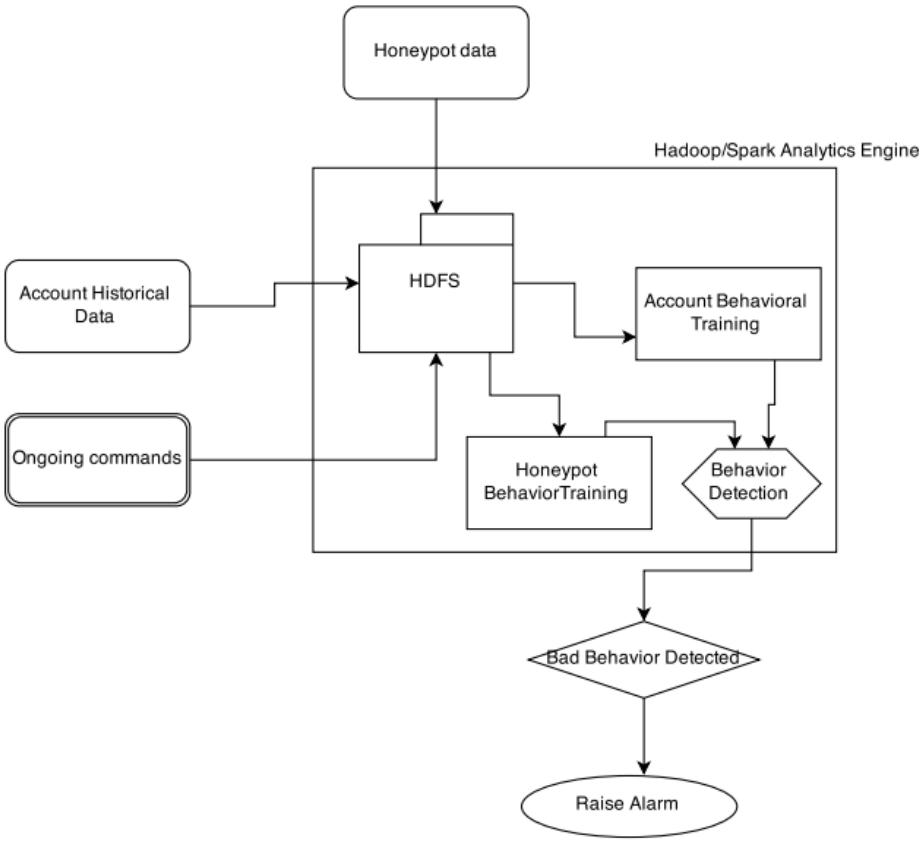


Figure 11. Honeypot Behavior Analytics Engine

Naïve Bayes Algorithm for the Experiment

There are several classification algorithms that can be used for this type of experiment; however for simplicity a variant of Naïve Bayes was used. Naïve Bayes is easy, robust and can be fairly powerful in classification. The algorithm used in this experiment is very successful in spam detection; and therefore was chosen to identify bad behavior from a Honeypot and Good Behavior from trained user behavior. Using Bayes theorem it we can calculate the probability that a bad actor event Y, given the observed training data from a Honeypot, given as X [44].

$$P(X|Y) = \frac{P(Y|X) * P(X)}{P(Y)}$$

This formula is a proven method and has been used in a variety of practical applications in multiple fields. This theorem is the cornerstone of Bayesian Statistics, which can calculate the probability of an event that is happening based on earlier probability known estimates based on historical data. Using this type of formula to detect behavior data has been used in masquerade detection on bash profile. The formula often has been overlooked due to the constant changes in user behavior.

One approach to estimate the probability that a command is issued by a bad actor is to take the probability of that command being used by the bad actor by the total times (i.e., N) that enriched command has been used by both good actors and bad actors. This will give estimation that the theoretical probability that the command is executed by a bad actor increases. The formula used for this common Bayes Implementation is as follows: The formula below is the number of occurrences of a bad actor giving command x over the total of bad actors and good actors giving the command x.

$$P_{MLE}\,(x_i) = \frac{|1x_i|}{N}$$

One drawback of using the Naïve Bayes is that the formula assumes that every command is given equal weight for both bad actors and good actors [44]. There are more sophisticated implementations of Naïve Bayes that use additional context. Building in additional context requires a more sophisticated model, which would take time and analysis.

Honeypots can provide robust data concerning malicious behavior due to the data having a very high positive rate, since only a bad actor would attempt to do anything on the system. With a high level of confidence that the data collected from a Production Honeypot it is possible to predict malicious behavior. Masquerade attacks have shown that attackers will use enriched commands differently that can deviate from good actor or a specific actors behavior [59].

These are two of the simpler versions of Bayes Theorem; however, they are still powerful enough to test the architecture. Some of the more advanced theorems require an iterative approach. The data analytics framework in this experiment would pair well with the iterative algorithms where traditional systems have issues providing a timely response. With frameworks like Apache Spark it may be possible to get near real time results. Similar architecture using big data frameworks have proven to scale for other domains [46].

CHAPTER V

ANALYSIS OF THE EXPERIMENT

Kippo was primarily used as a way to create groups of behaviors that could be seen as bad. Originally the intention was to get real data instead of faking all of the data used for detection. However this turned out to be an exciting part of the experiment as explained below due to the interesting findings found.

Hadoop Distributed File System (HDFS) proved to have several technical challenges. Setting the system in distributed mode was fairly difficult; however the most frustrating challenge was when the NameNode was lost and some of the core data was lost in a binary format that only the lost NameNode meta-store could read.

Apache Spark was the analytics engine chosen to carry out the analysis since it can operate on Hadoops HDFS and through the cluster as a replacement for MapReduce. PySpark is the version of Spark used since it has an interactive shell, which makes for a faster development cycle after (painfully) trying to write MapReduce code in Java. Another advantage of using Spark is that it does equally well with smaller data sets as it does with large datasets. The properties of Apache Spark scale well when a solution is going to be analyzing several streams of data, and if switching the dataset to real time.

The data used in this experiment were saved in HDFS, and in raw form. Traditional databases require that the data is cleaned and formatted into a schema first design. With HDFS and MapReduce/Spark it is easy to keep the data in is raw format and

apply a schema at runtime. This schema-second capability is what separates big data architectures like Hadoop and HDFS from traditional RDMS.

Interesting Kippo Findings

There were many interesting insights that were gleaned from using a SSH honeypot that were unrelated to the analysis that was the primary focus of this thesis. The information that can be mined from just the default configuration with a Kippo server can be useful for understanding attack vector trends.

Often a quick Google search would reveal the hidden meaning of some of the attacks, especially some of the obscure software names. Many of the hacks were trying to install an IRC Bot. One of the IRC hacks that were common was to install a system called psyBNC. This program turned out to be a popular IRC Bouncer that allows users to hide their IP addresses; it can also be used for malicious attacks as well as a way to redirect attacks from the host machine, among other things. One of the areas worth exploring for other research is the various ways these common packages are installed, because not every attempt goes about the same method.

Most of the hacks used Perl and would give up after some sort of Perl segmentation fault. Kippo tries to fake package installation as best as it can; however, as soon as a hacker encounters some sort of segmentation fault it immediately gives up. There were numerous attempts at trying to install some sort of Bitcoin miner to steal CPU cycles from the host. Worth noting are numerous hackers that try to masquerade as common processes such as httpd or even MySQL daemon accounts. There are a lot of clever hacks and tricks that try to avoid and blend in to the system background.

One of the limitations using Kippo is that it has minimal interactivity. Kippo tries hard to mimic a real Linux shell; however, due to the low connectivity time against the server, it appears that attackers abandon the sessions fairly quickly. After the experiment was over and the logs were collected and analyzed it was clear that using a common usernames and passwords would scare away any serious attackers. Any future experiments it is recommended to use stronger credentials to better mimic a real system. It would also be beneficial to use a system that can better emulate a POSIX Unix server to better entice attackers to stay and execute more commands against the server.

The figure below is a chart of the top 20 commands. The total number of attempts against this particular honey pot was 1,210,028. The two usernames excluded from the graph were admin and root because they comprised of over half the total username attempts. It's worth noting that "pi" was roughly 3% of the attempted passwords.

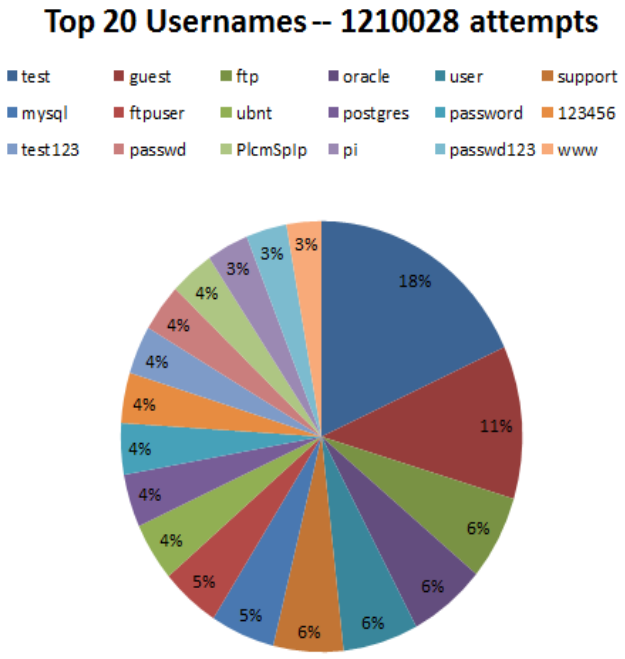


Figure 12. Top 20 Kippo Usernames

The top commands were much more varied. Below is an image of the top first commands. The most used commands were ls, and wget. It's interesting that the first thing an attacker attempts to do is connect to a web server and download software. The server is usually by IP address and not by name. Command wget is used to download software off the internet. Most traditional masquerading detection systems use only the command name and not the arguments [58]. Using the argument of the command the system can record the name of the software being downloaded which can give even more information about the deviant behavior.

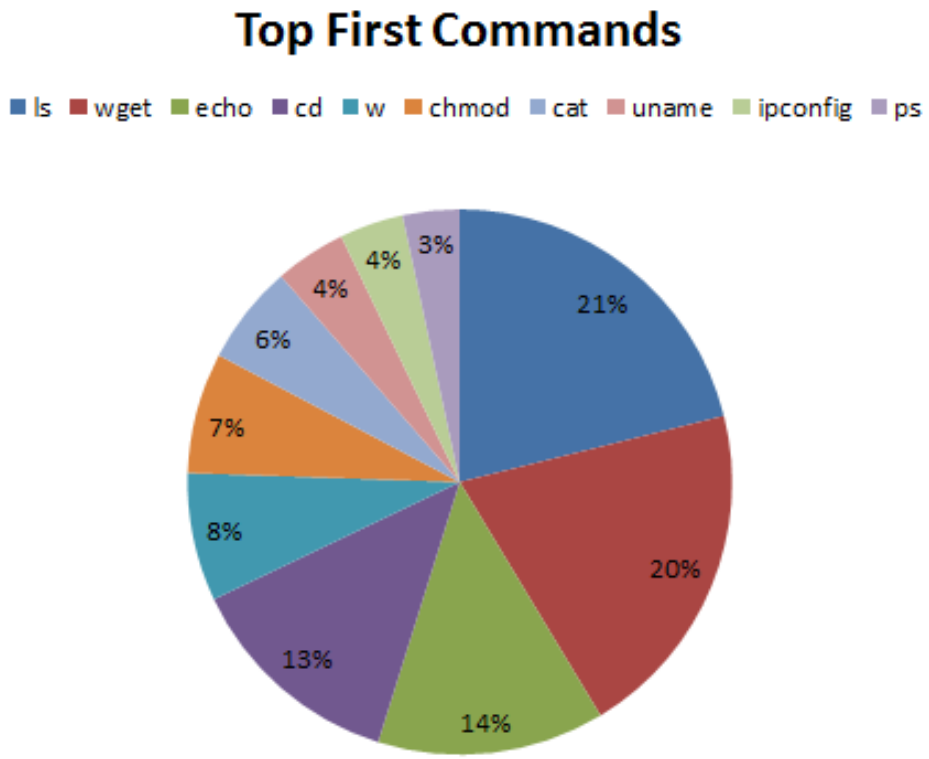


Figure 13. Top First Kippo Commands

Sanitizing Honeypot Data

After collecting data it was apparent that using an easy username and password combination for the honeypot deterred attackers. Some of the users would leave after committing one or two commands. However, despite users not staying long, the number of commands was more than enough to analyze.

Kippo Honeypot data is semi-structured data. There are projects that store data into MySQL; however there are performance hits and limitations when a honeypot is under heavy use. Spark works very efficiently with unstructured and semi-structured data. Spark-shell and PySpark are interactive REPL that allow for quick data exploration on distributed datasets. Regular expressions are used heavily to get the data into a usable format. "wget" and "curl" are common commands that are used by attackers within the honeypot; therefore, any commands that download software form the Internet will remove the IP address and only take the name of the software being downloaded.

The main limitation of the overall experimental data is due to the type of honeypot used. The main reason Kippo was chosen was because it's a highly used SSH honeypot with good documentation and tutorials. Production honeypots are becoming more popular by organizations. It is important to disguise the honeypot as a real system.

To make sure that the commands had enriched detail (e.g. flags and the first argument) were kept as a whole. Appendix G has the commands in Scala used to sanitize the commands into a standard format. Sanitizing them into a standard is important to standardizing a way in which we apply the Naïve Bayes formula as discussed in Chapter IV. The implementation used in this experiment is very narrow due to a small set of data that was collected through the Honeypot.

Conclusion from the Analysis

Eighty percent of the data was used for training and the remaining was employed for testing purposes. Using the results with enriched commands; the accuracy presented around 83%. Issuing the first command the rate was far worse at around 33% accuracy. This was in part due to the use of common commands. Adding in the command flags adds a lot more variety in how the users used the system. Certain flags are almost perfect in interactive usability testing.

Applying a four node Apache Spark cluster the analysis on five hundred megabyte datasets completed in less than one second. Spark has the ability to scale; therefore can easily give responses that are acceptable for quick response time when analyzing attacker data. The solution creates real time results from new honeypot data and updated behavioral data.

Combining the analysis of the probability of a bad actor using a SSH honeypot along with modelling normal user behavior with new user behavior creates an ensemble method. The system will keep a history of all commands that a user has made along with a timestamp. When a user uses a command or a flag that is not in their history, the probability that the command is executed by a bad actor combined with data analysis against normal behavior versus honeypot behavior creates an ensemble method to create a more accurate probability.

Ensemble analysis is not limited to just one formula within the architecture described in this thesis. All network data and application data that can provide useful analysis that can be correlated in the same timeframe can create a more accurate picture of an attack. For example, if there is unusual increased network traffic that exceeds a

specific threshold while at the same time there are multiple potential bad actors that were defined through comparing honeypot analysis against behavioral analysis the probability there is a bad actor is significantly increased with higher confidence.

The real time analysis is only one portion of the overall solution. While the analysis is going against historical data, new attack vectors are constantly updated. The raw data is stored within HDFS unaltered. The data required is transformed programmatically into the desired format for analysis. As new attack vectors are stored, it's possible to rerun historical analysis to see if there were any probabilities that an attack has already happened that was before identified. Historical attacks can alert analysis to investigate and correlate previous potential attacks.

Example Scenario of Honeypot Ensemble Analysis

Figure 5.3 is a visual representation of an attack and analysis scenario. The following scenario is based on APT scenarios using UIT attack patterns [38] [51] [59]. This scenario assumes a zero day attack vulnerability being released according to Figure 2.1. The honeypot ensemble analysis architecture in figure 5.3 is based on the architecture presented in Figure 4.3.

This scenario assumes that a new vulnerability has been identified on a Bash system. Bad actors will try to exploit that vulnerability on massive servers throughout IP addresses on the Internet [54]. Since the honeypot is not a production based system that is used by normal users; therefore, only bad actors will attempt connections [26]. New bash shell command signatures will be collected from the honeypot and sent to the honeypot ensemble analysis architecture.

During this time, an UIT that has penetrated the organization's intranet uses the same commands to exploit an internal facing server against the known vulnerability. The UIT is successfully giving the UIT privileged access to the system. Commands executed on the internal facing server are collected into the honeypot ensemble analysis architecture. The commands are parsed and analyzed using the command parser in Appendix G. Probability that the command is a bad actor command is calculated by checking whether or not that command with enriched flags has been executed by that user before, if that command has been used by only bad actors on the honeypot, and how recent that attack vector has been recorded through honeypot analysis.

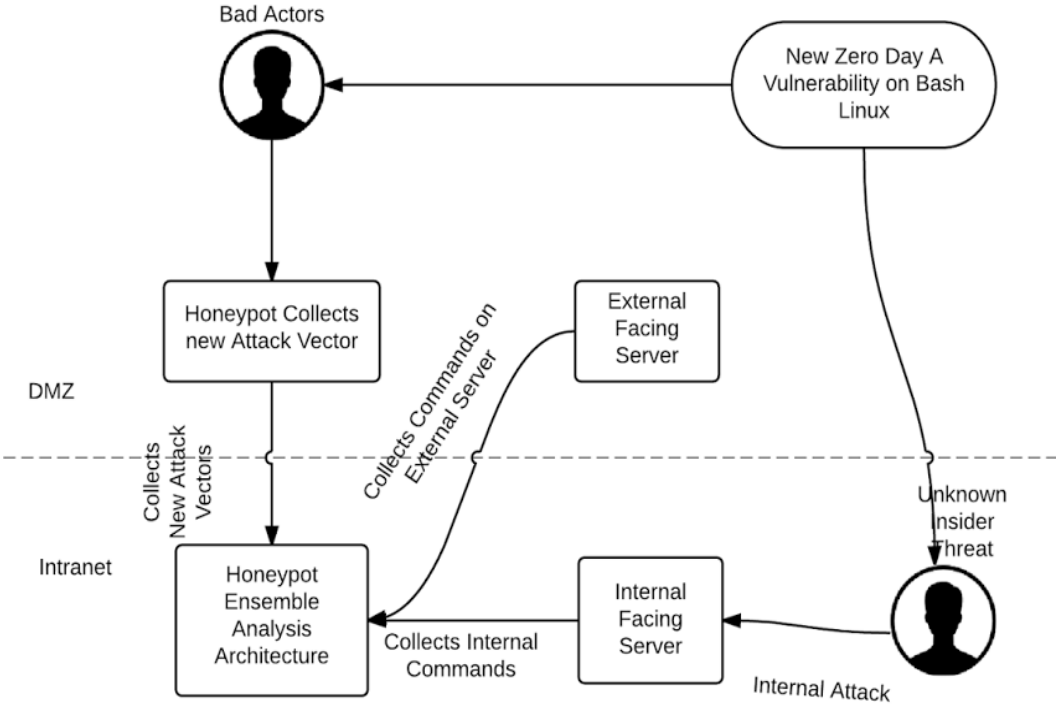


Figure 14. Example Honeypot Analysis Scenario

Multiple outcomes of the automated analysis can occur. One – the probability and the weight of the analysis can send in automated alerts for analysts to check the account on the internal facing server. Two – the account can be monitored for a set number of

alerts in a time period or within a number of executed commands. Three – some kind of prescriptive event could happen (e.g. the account's password could be reset so that the legitimate user contacts support to reset the password). Each of these outcomes has tradeoffs, but they are all possible outcomes. The Naïve Bayes implementation described in this thesis is not as context rich as other algorithms (e.g. support vector machines [11]); therefore, it may not be robust enough to trigger external events, such as locking out the user account.

How This Experiment Differs From Previous Research

This experiment was inspired by unrelated journal articles. One of the primary inspirations for this experiment was based on a 2013 paper, *Future directions for behavioral information security research*. Three future information security themes for behavioral analysis are: "Differentiate insider deviant behavior from insider misbehavior," "Improve methods for collecting and measuring security related data," "Capture actual behavior [13]."

Honeypots: Catching the Insider Threat, was an article that inspired the use of a honeypot to collect bad actor data. Their study was limited to deploying a honeypot within a firewall in order to detect insider threat since connections external to their network couldn't connect to the system [54]. Combining the ideas from both papers gave the idea to differentiate bad actor behavior from normal behavior through capturing only bad actor attempts against a fake system.

Hadoop and machine learning have been used in recent studies for automated detection. One of the studies focused on automating analysis against a honeypot to identify bots that have been deployed within an organizations network [62]. Another

study was for automatic analysis of malware using machine learning techniques [61]. Both of these studies influenced the use of Hadoop ecosystem for behavioral analysis.

Behavioral analysis has been well studied within information security. Intrusion detection system utilizes anomaly detection for network access patterns [53]. One drawback of this approach is that the IDS focus on the access patterns of network based traffic and not what users are doing on an application or system. Application profiles offer more robust data sets; however, there is no standard format for how applications collect user interactions.

Within Linux systems there are standard logging for shell interaction. Numerous behavioral analysis studies have used bash profiles for anomaly detection within Linux. A variety of algorithms have been used for masquerade detection against these profiles (e.g. Naïve Bayes and Hidden Markov Models) [59]. There are some drawbacks to some of the studies. Data points need to be collected on a user for an initial in order to reliably identify whether a new action is a deviation from normal behavior [60].

This experiment is different from previous approaches in that it collects bad actor behavior on a system that only bad actors would access [54][56]. The bad actor data can be used to compile a known bad actor profile. If a user commits a series of actions that deviate from their normal behavior and those actions correspond with known bad actor behavior, then the probability that the action is done by a bad actor is increased.

CHAPTER VI

LIMITATIONS, FUTURE RESEARCH, AND CONCLUSION

Information security can benefit from the new data analytics architectures analyzing heterogeneous sets of data to identify potential security breaches. Attacks are becoming more complex and sophisticated; therefore detection and analysis need to be in place to accurately identify and prevent attacks that can compromise an organizations network.

Limitations of the Experiment

There were several limitations of this experiment so that it wasn't trying to test too many variables. It was limited in only advanced logging shell sessions and data gathered from a honeypot. The more heterogeneous data points that can be collected and aggregated to show similar data at the time of an attack can greatly improve the probability that an attack has occurred. The experiment would benefit from a larger more complete dataset. Even though there were over 2 million events with this honeypot, only a fraction of the events were commands. The commands did provide insight as the patterns used on the honeypot were much different than normal user behavior.

The honeypot that was used is an open source product that has powerful capabilities for a SSH Honeypot. The configuration allows administrators to add capabilities for advanced session information to mimic a Linux shell. Future security

architecture research can dynamically route false positives masquerade attack occurring on a Linux server to a honeypot server. This could be used to help automatically determine if a potential attack is a false positive or a false negative. The experiment could have benefitted from both more honeypot data as well as user data.

Maximum likelihood Naïve Bayes

Future Research

Behavioral analysis on Honeypots to build a data driven security system are excellent candidates for future research in the information security field as well as academia. SSH command behavior isn't the only type of behavior. Exploring HTTP Honeypot user behavior would be a good research area to explore with honeypots to detect unknown attack vectors. With attacks such as "Heartbleed," a honeypot could help identify the attack vector before it is announced into public attention.

Data driven security is rising in popularity due to frameworks such as Hadoop and Spark. The rise of Data Science as an occupation makes it appealing for information security organizations to focus on data driven products using machine learning for predictive and prescriptive capabilities. This research used a data analytics framework with honeypot data to create a predictive model that can help detect unknown attack vectors using behavioral analysis on SSH systems.

Conclusion

The experiment in this thesis was to use big data architecture within a security context to analyze potential bad actors within a network. Behavioral ensemble analysis using an SSH honeypot proved to be a successful solution for detecting unknown threats within real time analysis. There are two benefits of using honeypot behavioral analysis in

combination with normal user behavioral analysis. First, it allows security analysts to identify when a user has bad actor behaviors. Second, the solution is robust enough to identify changes in bad actor behavior and potential new attack vectors. Using enriched command analysis, this experiment was able to take into account changes in how bad actors use different command flags in new ways.

REFERENCES

[1] Analytics, Big Data. "Big Data Analytics for Security." (2013).

[2] Andersen, David G., and Nick Feamster. "Challenges and opportunities in Internet data mining." *Parallel Data Laboratory, Carnegie Mellon University, Research Report CMU-PDL-06-102* (2006).

[3] Bansal, G., Gupta, A., Pyne, U., Singhal, M., & Banerjee, S. A Framework for Performance Analysis and Tuning in Hadoop Based Clusters. In *Smarter Planet and Big Data Analytics Workshop (SPBDA 2014), held in conjunction with International Conference on Distributed Computing and Networking (ICDCN 2014), Coimbatore, INDIA.*

[4] Bifet, Albert (2012). Mining Big Data in Real Time. *Yahoo Research*.Web. http://www.informatica.si/PDF/37-1/03_Bifet%20-%20Mining%20Big%20Data%20in%20Real%20Time.pdf

[5] Bilge, et al. "Before we knew it: an empirical study of zero-day attacks in the real world." *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012.

[6] Buell, Jeff. "A Benchmarking Case Study of Virtualized Hadoop Performance on VMware vSphere 5." *technical white paper. VMware, Inc.* (2011).

[7] Bughin, J., Chui, M., & Manyika, J. (2010). Clouds, big data, and smart assets: Ten tech-enabled business trends to watch. *McKinsey Quarterly*, *56*(1), 75-86.

[8] Burg, David, et al. (2014). Managing Cyber Risks in an Interconnected World. Key Findings from the Global State of Information Security® Survey 2015. http://www.pwc.com/gsiss2015.

[9] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., & Gruber, R. E. (2008). Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, *26*(2), 4.

[10] Chaudhuri, S., Dayal, U., & Narasayya, V. (2011). An overview of business intelligence technology. *Communications of the ACM*, *54*(8), 88-98.

[11] Chen, H., Chiang, R. H., & Storey, V. C. (2012). Business Intelligence and Analytics: From Big Data to Big Impact. *MIS Quarterly*, *36*(4).

[12] Courtney, M. (2014). Data on Demand. *Engineering & Technology*, *9*(1), 64-67.

[13] Crossler, Robert E., et al. "Future directions for behavioral information security research." *computers & security* 32 (2013): 90-101.

[14] Dean, J., & Ghemawat, S. (2004). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107-113.

[15] Dean, J., & Ghemawat, S. (2010). MapReduce: a flexible data processing tool. *Communications of the ACM*, *53*(1), 72-77.

[16] Gartner Group. Pattern-Based Strategy: Getting Value from Big Data. (July 2011 press release); http://www.gartner.com/it/page.jsp?id=1731916

[17] Gates, Alan F., et al. "Building a high-level dataflow system on top of Map-Reduce: the Pig experience." *Proceedings of the VLDB Endowment* 2.2 (2009): 1414-1425.

[18] Ghemawat, S., Gobioff, H., & Leung, S. T. (2003, October). The Google file system. In *ACM SIGOPS Operating Systems Review* (Vol. 37, No. 5, pp. 29-43). ACM.

[19] Greitzer, Frank L., et al. "Analysis of Unintentional Insider Threats Deriving from Social Engineering Exploits." *Security and Privacy Workshops (SPW), 2014 IEEE*. IEEE, 2014.

[20] Hashem, Ibrahim Abaker Targio, et al. "The rise of "big data" on cloud computing: Review and open research issues." *Information Systems* 47 (2015): 98-115.

[21] Jagadesh, H.V., et al. "Big Data and Its Technical Challenges." *Communications of the ACM* 57.7 (2014): 86-94. *Computers & Applied Sciences Complete*. Web. 30 Aug. 2014.

[22] LaValle, Steve, et al. "Big data, analytics and the path from insights to value." *MIT Sloan Management Review* 21 (2013).

[23] Lee, Y., & Lee, Y. (2013). Toward scalable internet traffic measurement and analysis with Hadoop. *ACM SIGCOMM Computer Communication Review*, *43*(1), 5-13.

[24] Louridas, P., & Ebert, C. (2013). Embedded Analytics and Statistics for Big Data. *IEEE Software*, *30*(6), 33-39.

[25] Manyika, James, et al. Big data: The next frontier for innovation, competition, and productivity. May 2011

[26] Mokube, Iyatiti, and Michele Adams. "Honeypots: concepts, approaches, and challenges." *Proceedings of the 45th annual southeast regional conference*. ACM, 2007.

[27] Mone, G. (2013). Beyond Hadoop. *Communications of the ACM*, *56*(1), 22-24.

[28] Nair, Ramesh, & Narayanan, Andy. (2012). Strategy and Benefitting from Big Data: Leveraging Unstructured Data Capabilities for Competitive Advantage. Booz & Company Inc. Web (August 2014). http://www.strategyand.pwc.com/media/file/Strategyand_Benefitting-from-Big-Data.pdf

[29] Qin, H. F., & Li, Z. H. (2013). Research on the Method of Big Data Analysis. *Information Technology Journal*, *12*(10).

[30] Sakr, S., et al. (2013). The Family of MapReduce and Large-Scale Data Processing Systems. *ACM Computing Surveys*, *46*(1), 11-11:44.

[31] Sallam, Rita L., et al. "Magic quadrant for business intelligence platforms." *Gartner Group, Stamford, CT* (2011).

[32] Simoncelli, D., Dusi, M., Gringoli, F., & Niccolini, S. (2013). Stream-monitoring with BlockMon: convergence of network measurements and data analytics platforms. *ACM SIGCOMM Computer Communication Review*, *43*(1), 29-36.

[33] Shackleford, Dave. (2012) Security Intelligence in Action: A review of LogRhythm's SIEM 2.0 Big Data Security Analytics Platform. December 2012 A SANS Whitepaper. Web. http://www.sans.org/reading-room/whitepapers/analyst/security-intelligence-action-review-logrhythm-039-s-siem-20-big-data-security-analytics-35155

[34] Shackeford, Dave. (2013) SANS Security Analytics Survey. September 2013. Web. http://www.sans.org/reading-room/whitepapers/analyst/security-analytics-survey-34980

[35] Suthaharan, Shan. "Big data classification: Problems and challenges in network intrusion prediction with machine learning." *ACM SIGMETRICS Performance Evaluation Review* 41.4 (2014): 70-73.

[36] Taherimonfared, Aryan, Tomasz Wiktor Wlodarczyk, and Chunming Rong. "Real-Time Handling of Network Monitoring Data Using a Data-Intensive Framework." *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*. Vol. 1. IEEE, 2013.

[37] Villars, R. L., Olofson, C. W., & Eastwood, M. (2011). Big data: What it is and why you should care. *White Paper, IDC*

[38] Virvilis N., Gritzalis D., “The Big Four - What we did wrong in Advanced Persistent Threat detection?”, in Proc. of the 8th International Conference on Availability, Reliability and Security (ARES-2013), pp. 248-254, IEEE, Germany, September 2013.

[39] Virvilis N., Gritzalis D., “Trusted Computing vs. Advanced Persistent Threats: Can a defender win this game?”, in Proc. of 10th IEEE International Conference on Autonomic and Trusted Computing (ATC-2013), pp. 396-403, IEEE Press, Italy, December 2013

[40] Wang, Chengwei, Infantdani Abel Rayan, and Karsten Schwan. "Faster, larger, easier: reining real-time big data processing in cloud." *Proceedings of the Posters and Demo Track*. ACM, 2012.

[41] White, T. (2009). Hadoop: The Definitive Guide: The Definitive Guide. O'Reilly Media.

[42] Whitworth, Jeff, and Shan Suthaharan. "Security problems and challenges in a machine learning-based hybrid big data processing network systems." *ACM SIGMETRICS Performance Evaluation Review* 41.4 (2014): 82-85.

[43] Wu, Xindong, et al. "Data mining with big data." *Knowledge and Data Engineering, IEEE Transactions on* 26.1 (2014): 97-107.

[44] Yang, Zhen, et al. "An approach to spam detection by naive Bayes ensemble based on decision induction." *Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on*. Vol. 2. IEEE, 2006.

[45] Young, Williams T., et al. “Detecting Unknown Insider Threat Scenarios.” Security and Privacy Workshops (SPW), 2014 *IEEE*. IEEE, 2014.

[46] Zaharia, Matei, et al. "Spark: cluster computing with working sets." *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*. 2010.

[47] Zaharia, Matei, et al. "Discretized streams: Fault-tolerant streaming computation at scale." *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. ACM, 2013.

[48] N. Marz and J. Warren. Big Data: Principles and best practices of scalable real time data systems. Manning Publications, 2013.

[49] Home Depot reports findings in Payment Data Breach Investigation. (2014). [Online]. https://corporate.homedepot.com/MediaCenter/Documents/Press%20Release.pdf

[50] Scharr, Jill. Chase Bank Security Breach May Not Be That Bad. (2014). [Online] http://www.tomsguide.com/us/chase-bank-breach-update,news-19545.html

[51] CERT Insider Threat Team. (2013) Unintentional insider threats: A foundational study. [Online] Available: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=58744

[52] Clearswift. (2013) The enemy within. [Online]. Available: http://www.clearswift.com/sites/default/files/images/blog/enemy-within.pdf

[53] Liao, Hung-Jen, et al. "Intrusion detection system: A comprehensive review." *Journal of Network and Computer Applications* 36.1 (2013): 16-24.

[54] Spitzner, Lance. "Honeypots: Catching the insider threat." *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*. IEEE, 2003.

[55] Hunker, Jeffrey, and Christian W. Probst. "Insiders and insider threats—an overview of definitions and mitigation techniques." *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* 2.1 (2011): 4-27.

[56] Watson, David, and Jamie Riden. "The honeynet project: Data collection tools, infrastructure, archives and analysis." *WOMBAT Workshop on Information Security Threats Data Collection and Sharing*. 2008.

[57] Stamp, Mark. "A Revealing Introduction to Hidden Markov Models." *Department of Computer Science San Jose State University* (2012).

[58] Schonlau, Matthias, et al. "Computer intrusion: Detecting masquerades." *Statistical science* (2001): 58-74.

[59] Maxion, Roy A. "Masquerade detection using enriched command lines." *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE Computer Society, 2

[60] Rieck, Konrad, et al. "Automatic analysis of malware behavior using machine learning." *Journal of Computer Security* 19.4 (2011): 639-668.

[61] Haltas, Fatih, et al. "An automated bot detection system through honeypots for large-scale." *Cyber Conflict (CyCon 2014), 2014 6th International Conference On*. IEEE, 2014.

[62] Virvilis, Nikos, Dimitris Gritzalis, and Theodoros Apostolopoulos. "Trusted Computing vs. Advanced Persistent Threats: Can a defender win this game?." *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*. IEEE, 2013.

APPENDIX A

KIPPO INSTALLATION AND CONFIGURATION

Kippo is an Open Source project. Google code used to host the project; however it has since moved to GitHub. Since the project is open source it is easy to configure the server to either collect additional data or customize the system to serve the end user's needs. Kippo is written in python and requires python and python-twisted installed on the system in order to run.

Kippo Open Source Honeypot Location - https://github.com/desaster/kippo

INSTALLING KIPPO – Installation on Debian

Install Python:	`sudo apt-get install python`
Twisted Python:	`sudo apt-get install python-twisted`
Install SSHD	`sudo apt-get install openssh-server`
/etc/ssh/sshd_config	`Port 5576`
Restart SSHD	`sudo service ssh restart`
Add kippo user	`sudo service ssh restart`
Download Kippo	`wget http://kippo.googlecode.com/files/kippo-0.5.tar.gz`

INSTALLING KIPPO – Installation on Debian (Continued):

Untar Kippo	Tar –xzf kippo-0.5.tar.gz –C /opt sudo iptables -t nat -A PREROUTING -p
Redirect SSH	sudo iptables -t nat -A PREROUTING -p tcp --dport 22 j REDIRECT --to-port 5576
Configure FS	/opt/kipp-0.5/utils/creatfs.py > fs.pickle
	Echo “Red Hat Enterprise Linux Server release 6.5 (Santiago) \n \l > honeyfs/etc/issue
authBind setting	Apt-get install authbind
create file	Touch /etc/authbind/byport/22
Configure start.sh	authbind Twistd –y kippo.tac –l log/kippo.log –pidfill kippo.pid
Start Kippo	./start.sh

Once Kippo is started users can attempt to connect to Kippo. All the logs are stored at kippo.log within the Kippo directory. The logs aren’t in a great format to parse, which is why Kippo/Honeypot was used in this experiment. Attacker information is captured by any connections made to the Honeypot SSH server including the attackers: IP information, Port, Username/password, and RSA Id.

It is possible to configure Kippo further. The kippo.tac file contains the various username and passwords that users can connect to. By default this is root and 123456; which can actually scare hackers away. It is best if that configuration is changed. Kippo is written in python; and is pretty easy to add in custom logging and configuration; documentation on Twisted API can help configure the system further.

There are plenty of resources on Kippo’s home page. A proficient python programmer will want to look through the source code. Advanced functionality can be easily added to Kippo by extending Python functionality.

APPENDIX B

INSTALLING AND CONFIGURING HADOOP

Currently there are two separate versions of Hadoop. Hadoop 1's processing model is oriented towards MapReduce and batch oriented jobs. The Hadoop's architecture is for the most part the same. The biggest differences are between how HDFS is used in Hadoop 1 and YARN is used in Hadoop 2.

Hadoop 1 is a single Namenode that keeps metadata about each file and directory within Hadoop. Hadoop 2 uses HDFS federation which allows for multiple Namenodes to manage multiple namespaces and allow for greater scalability. Most of the differences are in the administration of the cluster and performance, and doesn't affect the way developers and analysts use the cluster (for the most part).YARN allows for multiple applications to share the resources of the Hadoop Ecosystem.

Installing Hadoop:

Download Hadoop	wget http://mirrors.gigenet.com/apache/hadoop/common/hadoop-2.6.0/hadoop-2.6.0.tar.gz
Extract hadoop	tar –xzf hadoop-2.6.0.tar.gz /C /opt/
Create hadoopuser	useradd –s /bin/bash –m –d /home/hadoopuser hadoopuser
Generate SSHkeys	ssh-keygen [Press enter : enter again : No passphrase]
Create sshkeys	Cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
Configure core-site	Name = edgenode || edgenode:9000

Edit \etc\hosts (e.g. 4 Nodes)

```
192.168.12.10   PrimaryHadoop
192.168.12.11   SecondHadoop
192.168.12.12   TertiaryHadoop
192.168.12.13   QuadHadoop
```

Copy Hadoop and SSH Keys to Other Environments

```
ssh secondhadoop "/sbin/cat >> ~/.ssh/authorized_keys" <~/.ssh/rsa_id.pub
```

Configure all the hadoop servers the same. Verify that the primary server/edge node can ssh to each of the slave nodes without password. This is important because Hadoop starts and communicates to each of the slave nodes through SSH. The /opt/hadoop/conf/slaves file on the serer needs to contain each of the host addresses for communications.

Format HDFS:

```
/opt/hadoop/bin/hadoop primaryhadoop -format
```

Start HDFS:

```
/opt/hadoop/bin/start-dfs.sh
```

Start Mapred:

```
/opt/hadoop/bin/start-mapred.sh
```

Verify Working

```
/opt/hadoop/bin/hdfs dfs –ls /
```

Installing and configuring Spark:

Download Spark:

```
wget http://www.apache.org/dyn/closer.cgi/spark/spark-
```

Untar Spark

```
tar –xzf spark-1.2.1.tgz /C /opt/
```

Configuring spark, the administrator will need to edit the file /opt/spark/conf/spark-env.sh file. The environment variables need to be set: SPARK_MASTER_IP : PrimaryHadoop. Configure the /opt/spark/conf/slaves file to contain all the other addresses that were set in the "/etc/hosts" file.

Copy to each Environment

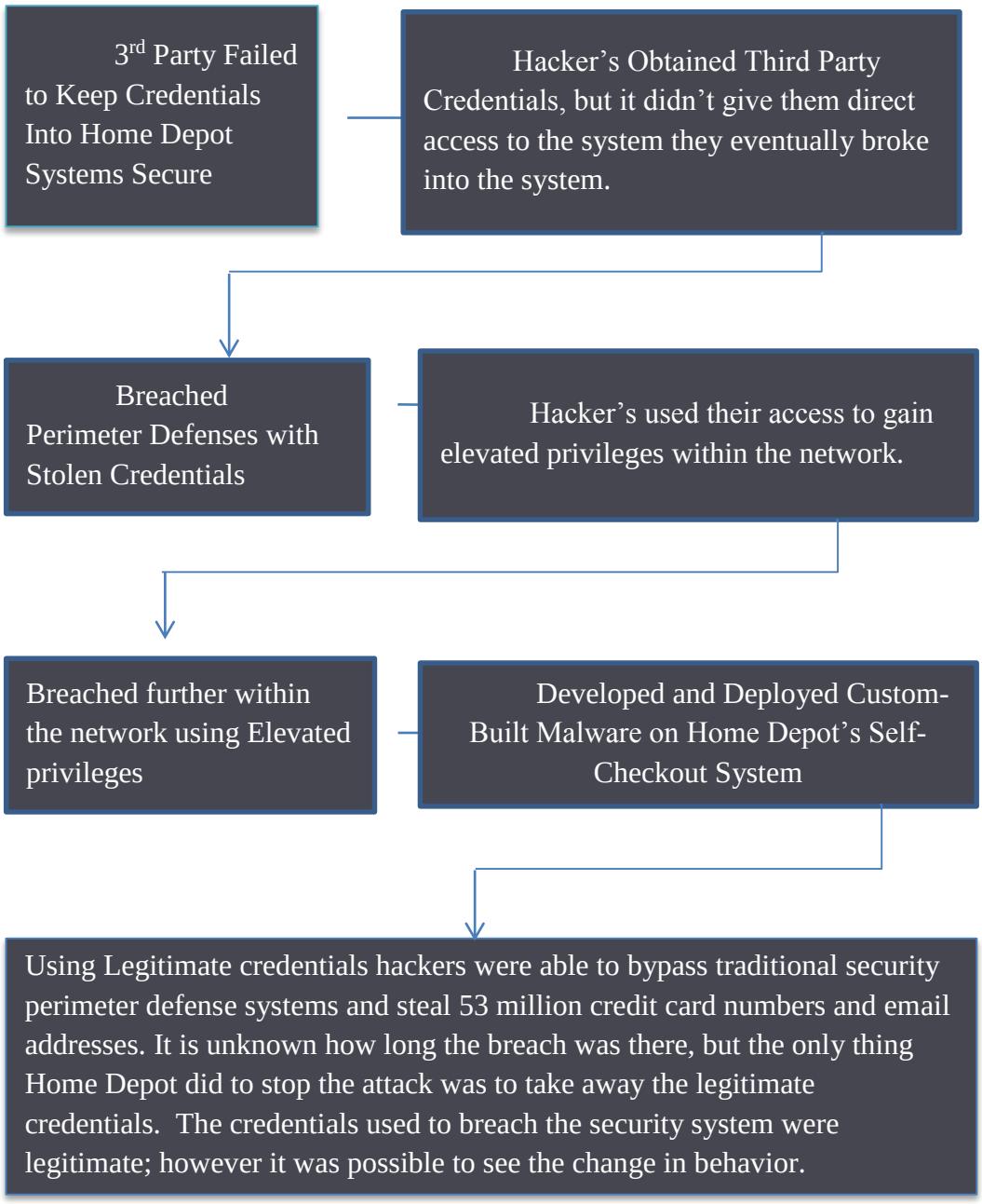
```
scp /opt/spark/* secondhadoop:/opt/spark
```

Start Spark

```
/opt/spark/sbin/start-all.sh
```

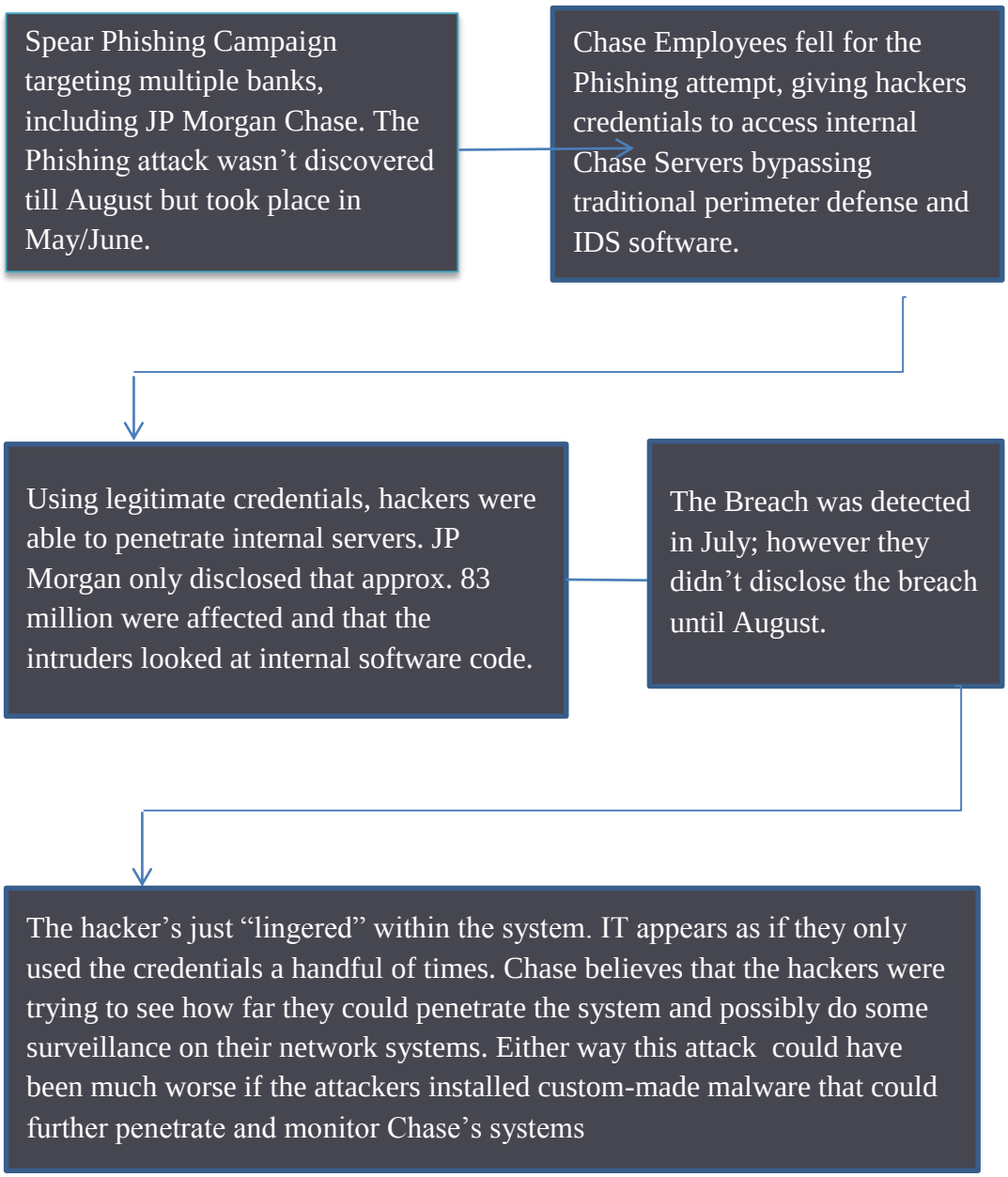
APPENDIX C

ANALYSIS OF HOME DEPOT ATTACK



APPENDIX D

ANALYSIS OF CHASE MULTI-STAGE ATTACK



APPENDIX E

ADVANCED BASH HISTORY SETUP

The following command will add some parameters to the bash_history. By Default the bash_history is just the command and parameters. It's possible to do further changes to bash_history, but for the purpose of this project was to only make minor modifications for repeatability.

```
#!/bin/bash
# This script should be run on every user's account on creation.

USER_HOME=$(echo ~)
BASHRCL=$USER_HOME/.bashrc

# Change the output of the bash history format so that it shows timestamp
echo "export HISTTIMEFORMAT = ' %F %T ' " >> $BASHRCL

# Change the .bash_history so that the bash history file size is larger than
normal
echo "export HISTSIZE=20000" >> $BASHRCL
```

APPENDIX F

SCALA COMMANDS TO ANALYZE HONEYPOT DATA

The following commands were used to do analysis on the honeypot data within Hadoop. These are basic map and filter commands that do string manipulation on the logs to get the relevant data out.

```
/*============== SCRIPT LOADING AND PARSING DATA ============*/
val honeypotlogs = sc.textFile("hdfs://primaryhadoop:10001/honeypotlogs/HoneyPotlogs.log")
val logins = honeypotlogs.filter(_.contains("Login"))
val regexlist = logins
    .map(line => """\[[^/|^\]]+/[^\]]+\]""".r.findAllIn(line).toList)
val allusers = regexlist.filter(!_.isEmpty)
    .map(_.mkString).filter(_.size > 0).filter(_.contains("/"))
val usernamepasswords = allusers.map(line => line.replace("[","").replace("]",""))
val tupleusernamepasswords = usernamepaswords.filter(_.split("/").size > 1)
    .map(line => (line.split("/")(0), line.split("/")(1)))
val usernamecounts = tupleusernamepasswords
    .map(tuples => (tuples._1, 1)).reduceByKey((a, b) => a + b)

// TOP Ten username attempts
tupleusernamepasswords.map(tuples => (tuples._1, 1))
   .reduceByKey((a, b) => a+ b).map(_swap).sortByKey(false).take(10).foreach(println)

tupleusernamepasswords.filter(e=> e._1 != e._2).filter(!_._1.contains("admin")).filter(!_._1.contains(
    "root")).map(tuples => (tuples._1 + "_" + tuples._2, 1)).reduceByKey((a, b) => a + b).map(_.swap).
    sortByKey(false).take(150).foreach(println)

// Grab every line that isn't a log on then split them
val non_log_ons = honeypotlogs.filter(!_.contains("Login"))
    .filter(_.contains(" "))
    .map(line => (line.split(" ")(0, line.substring(line.indexOf(" "))))

// Grab every command and the arguments used for that command
val allCommands = honeypotlogs.filter(_toUpperCase.contains("COMMAND"))
    .map(line => """Command \[[^\]]+\]""".r.findAllIn(line).toList)
    .map(line => line.mkString)

// List of each individual command:
val individualcommands = allCommands.filter(!_.isEmpty).map(line => """\[[^ ]+""".r.findFirstIn(line).
    toList).filter(!_.isEmpty).map(_.mkString.replace("[","").replace("]","")).collect.foreach(
    parseEnrichedCommands(_))
```

APPENDIX G

SCALA PARSER FOR ENRICHED COMMANDS

The following functions are used to transform bash commands into enriched commands so that we can look at any unusual flags or software downloaded.

```
object CommandParser {
    def parseHttpURL(input:String) : String = {
        val index = input.lastIndexOf("/")
        if (index >= 0) input.substring(index + 1) else input
    }
    /* Takes any thing in quotes and transforms it into a single command */
    def quotesToSingleCommand(input:String) : String  = {
        val regexlist = """[^"]+""".r
        val otherCommands = regexlist.findAllIn(input).toList.map(line => line.replace("\"", ""))
        var returnlist = List[String]()
        for(i <- 0 to otherCommands.size - 1) {
            returnlist = returnlist :+ transformCommand(otherCommands(i))
        }
        returnlist.mkString("||")
    }
    /* Removes anything between two doublequotes from the string */
    def removeQuotesFromCommand(input:String) : String = {
        val regexlist = new Regex("\"[^\"]+\"")
        regexlist.replaceAllIn(input, "").mkString
    }
    /* Transform Commands so that we have enrichment that can further be used to identify an attacker */
    def transformCommand(input:String) : String = {
        var returnValue = ""
        var firstArgExist = false
        if(!input.isEmpty) {
            val inputlist = input.split(" ")
            returnValue = inputlist(0)
            for (i <- 1 to inputlist.size - 1) {
                var value = inputlist(i)
                if (value.size > 0 && (value(0) == '-' || !firstArgExist)) {
                    if (value(0) != '-') { firstArgExist = true }
                    if (inputlist(0).contains("wget") || inputlist(0).contains("curl")) {
                        value = parseHttpURL(value)
                    }
                    returnValue = returnValue + " " + value
                }
            }
        }
        returnValue
    }
    def parseEnrichedCommands(input:String) : String = {
        var returnValue = ""
        if (!input.isEmpty) {
            val otherCommand = quotesToSingleCommand(input)
            val removedCommands = removeQuotesFromCommand(input)
            val removedCommandsTransformed = transformCommand(removedCommands)
            if (!otherCommand.isEmpty) {
                returnValue = otherCommand + "||" + removedCommandsTransformed
            } else {
                returnValue = removedCommandsTransformed
            }
        }
        returnValue
    }
}
```

APPENDIX H

SCALA NAÏVE BAYES TRAINING COMMANDS

The following functions are used to train Naïve Bayes for the honeypot data against behavior commands.

```
/* Training Naive Bayes */
def main(args:Array[String]) {
    val badactorCommands = Source.fromFile("allcommands.txt").getLines.toList
    val goodactorCommands = Source.fromFile("trainingGoodActor.txt").getLines.toList

    // Training Bad actor Data
    println("Training Bad Actors: ")
    for (i <- 0 to badactorCommands.size - 1) {
        val enrichedCommands = parseEnrichedCommands(badactorCommands(i)).split("\\|\\|")
        for (x <- 0 to enrichedCommands.size -1) {
            val probability = calculateProbabilityBadActor(badactorCommands, goodactorCommands,
                enrichedCommands(x))
            println(enrichedCommands(x) + " , "  + probability)
        }
    }

    println("Training Good Actors: ")
    // Training Good Actor Data
    for (i <- 0 to goodactorCommands.size - 1) {
        val enrichedCommands = parseEnrichedCommands(goodactorCommands(i)).split("\\|\\|")
        for (x <- 0 to enrichedCommands.size - 1) {
            val probability = calculateProbabilityBadActor(badactorCommands, goodactorCommands,
                enrichedCommands(x))
            println(enrichedCommands(x) + " , " + probability)
        }
    }
}

def calculateProbabilityBadActor(goodActorList:List[String], badActorList:List[String], command:
    String) : Double = {
    val countinBad = goodActorList.map(parseEnrichedCommands(_))
        .count(_.contains(command))
    val countinGood = badActorList.map(parseEnrichedCommands(_))
        .count(_.contains(command))
    calculateBayesTheorem(countinBad, countinGood)
}


def calculateBayesTheorem(first:Int, second:Int):Double =  {
    (first * 1.0) / (first + second * 1.0)
}
```