Anonymization of Event Logs for Network Security Monitoring

Alis Rasic

A Thesis

In the Department

of

Concordia Institute

for

Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of

Master of Applied Science (Information Systems Security) at

Concordia University

Montréal, Québec, Canada

March 2020

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Alis Rasic**

Entitled: **Anonymization of Event Logs for Network Security Monitoring**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Information Systems Security)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Mohsen Ghafouri

_____ External Examiner
Dr. Dongyu Qiu

_____ Examiner
Dr. Walter Lucia

_____ Supervisor
Dr. Lingyu Wang

Approved by _____
Dr Mohammad Mannan, Graduate Program Director

February 26 2020

_____
Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Anonymization of Event Logs for Network Security Monitoring

Alis Rasic

A managed security service provider (MSSP) must collect security event logs from their customers' network for monitoring and cybersecurity protection. These logs need to be processed by the MSSP before displaying it to the security operation center (SOC) analysts. The employees generate event logs during their working hours at the customers' site. One challenge is that collected event logs consist of personally identifiable information (PII) data; visible in clear text to the SOC analysts or any user with access to the SIEM platform.

We explore how pseudonymization can be applied to security event logs to help protect individuals' identities from the SOC analysts while preserving data utility when possible. We compare the impact of using different pseudonymization functions on sensitive information or PII. Non-deterministic methods provide higher level of privacy but reduced utility of the data.

Our contribution in this thesis is threefold. First, we study available architectures with different threat models, including their strengths and weaknesses. Second, we study pseudonymization functions and their application to PII fields; we benchmark them individually, as well as in our experimental platform. Last, we obtain valuable feedbacks and lessons from SOC analysts based on their experience.

Existing works[42, 43, 47, 38] are generally restricting to the anonymization of the IP traces, which is only one part of the SOC analysts' investigation of PCAP files inspection. In one of the closest work[46], the authors provide useful, practical anonymization methods for the IP addresses, ports, and raw logs.

Acknowledgments

I am grateful to my wife for her patience, and my family, who were always there to support me. I am also thankful to my supervisor, Dr. Lingyu Wang, who introduced me to this exciting area of research and who helped me complete this thesis. I also appreciate Dr. Jeremy Clark's help, who enlighten me every time I needed clarity.

"Where fear is present, wisdom cannot be." - Lactantius

Contents

List of Figures viii

List of Tables x

1 Introduction 1

1.1 Background and Motivation . . . 1

1.2 Objectives and Contributions . . . 3

1.3 Thesis Organization . . . 4

2 Background 5

2.1 Intrusion detection system (IDS) . . . 5

2.1.1 Signature-based IDS . . . 5

2.1.2 Anomaly-based IDS . . . 6

2.1.3 Types of IDS methodologies . . . 6

2.1.4 Security Monitoring . . . 8

2.2 Managed Security . . . 10

2.2.1 Managed Security . . . 10

2.2.2 Data breaches . . . 11

2.3 Privacy . . . 13

2.3.1 Privacy and PII . . . 13

2.4 Anonymization, pseudonyimzation and searchable encryption . . . 16

2.4.1 Pseudonymization and anonymization . . . 16

2.4.2 Anonymization 18
2.4.3 Searchable encryption 19

3 Related works **25**

4 Threat model and architectures **30**

4.1 Objectives 32
4.2 Threat model description 33
4.2.1 Threat model 34
4.3 Presenting architectures 36
4.3.1 Architecture 1 - Classical architecture (without anonymization) 36
4.3.2 Architecture 2 - Anonymization on SIEM 37
4.3.3 Architecture 3 - Private key managed on the sensor 38
4.3.4 Architecture 4 - Private keys managed by the customer 38

5 Modelling use cases **40**

5.1 Real-world use case study of network security monitoring 40
5.2 The use case scenarios 41
5.2.1 Use case 1 - Data breach through malicious code execution (MC) 45
5.2.2 Use case 2 - External Vulnerability Scan - Heartbleed 47
5.2.3 Use case 3 - Insider attack 49
5.2.4 Use case 4 - Distributed denial of service attack 50
5.2.5 Use case 5 - Data breach through MC (data exfiltration) 52

6 Applying anonymization techniques **53**

6.1 Fields and anonymization techniques 53
6.1.1 The choices of encryption or randomization methods 55
6.1.2 Quasi-identifier-based attacks and defence 57
6.2 Benchmarks of anonymization techniques 59
6.2.1 Throughput (Op/s) comparison 60
6.2.2 Execution runtime 61

6.2.3 Memory usage 64
6.2.4 CPU usage 65
6.2.5 Conclusion on benchmarks 67

7 Integration into a SIEM platform **68**
7.1 The architecture of our platform with PII anonymization 68
7.1.1 Description of our architecture 69
7.1.2 Implementation 69
7.1.3 Lessons learned 74

8 Conclusion **75**

Bibliography **77**

Appendix A **86**

List of Figures

Figure 2.1 MSSP deploying its sensors at the strategic locations to monitor customers network . . . 10
Figure 4.1 General overview of our threat model . . . 34
Figure 4.2 Classic architecture - private keys managed by MSSP . . . 36
Figure 4.3 Classic architecture - private keys managed by MSSP . . . 37
Figure 4.4 Architecture 3 - private keys managed on the sensor (MSSP) . . . 38
Figure 4.5 Architecture 4 - private keys managed by the customer . . . 39
Figure 5.1 Activity diagram of the SOC analysts . . . 42
Figure 5.2 Data exfiltration by sending to the victim a forged email message with attached malicious PDF file (malware). . . . 45
Figure 5.3 Attacker performing horizontal vulnerability scans followed by execution of heartbleed vulnerability . . . 47
Figure 5.4 Insider attack - employee altering customer bank balance data without authorization . . . 49
Figure 5.5 DDoS attack on a bank1.com website. . . . 50
Figure 6.1 Grouped quasi-identifiers and direct identifiers among all identifiers fields from collected event logs . . . 57
Figure 6.2 Averaged execution runtime (in ms) with and without anonymization . . . 62
Figure 6.3 Benchmarks of anonymization overhead with and without anonymization in terms of memory usage . . . 65

Figure 6.4 Benchmarks of CPU usage with and without anonymization on a dataset of 1 million records 66
Figure 6.5 Benchmarks of CPU usage with and without anonymization on a dataset of 1 million records 67
Figure 7.1 Architecture of our SIEM platform 70
Figure 7.2 Apache Spark Net-flows 72
Figure 7.3 Net-flow view 73
Figure 7.4 Nessus vulnerability scan view 73

List of Tables

Table 5.1 List of action IDs with their descriptions 42
Table 5.2 Investigation types with descriptions 43
Table 5.3 Use case 1 - Data breach through malicious code execution (MC) 46
Table 5.4 Use case 2 - EVC (External Vulnerability Scan/Reconnaissance and exploitation) - Heartbleed 48
Table 5.5 Use case 3 - Insider attack 49
Table 5.6 Use case 4 - Distributed denial of service attack 51
Table 5.7 Use case 5 - Data breach through malicious code execution (MC) and Anomaly Network behavior (ANB) 52
Table 6.1 Fields and encryption or anonymization techniques 55
Table 6.2 Averages of anonymization benchmarks overhead in terms of operation per seconds 61
Table 6.3 Benchmarks of anonymization overhead in terms of “executor run time (in ms)” 63

Chapter 1

Introduction

1.1 Background and Motivation

The ever-decreasing computation and storage costs today enable more data to be collected and processed than ever before. Many technology companies are now relying on cloud services (e.g., AWS, GCP, Azure) and distributed big data tools to offer better services while improving the scalability of their solution and keeping costs low. On the other hand, the increased collection of data and easier access to data also lead to more frequent incidents of data breaches and privacy violations. The public may suffer from the risks of identity theft as a result of data breaches, and public or private enterprises are frequently affected by data breaches resulting in millions of personally identifiable and sensitive data being disclosed to the public incurring costs of millions of dollars damaging organizations' reputation [13, 14, 15, 17]. In a recent high profile case, when Cambridge Analytica collected personal data of millions of Facebook users to manipulate public opinion, they were able to influence the decisions of many peoples on political views during the 2016 elections in the US.

Therefore, both non-profit and commercial organizations have the responsibility to protect collected private or personal information. In particular, this thesis examines this issue within the specific context of Network security monitoring (NSM). Today's small to medium-sized companies

who cannot afford to hire dedicated security experts are increasingly benefiting from Managed Security Service (MSS) provided by third party Security Operation Centres (SOC) to tackle network-wide threats. The advantages include cost-effectiveness, skilled security experts, appropriate facilities, and up to date security awareness. In addition, the 24 hours continuous service encourages different companies to outsource their security services rather than having in-house security employees.

Network security monitoring (NSM), which is typically defined as "the collection, analysis, and escalation of indications and warnings to detect and respond to intrusions", was born as a different term to specify the new feature of MSS for continuous monitoring of networks by human experts rather than just installing security appliances. In order to provide NSM services, an MSS provider deploys various sensors in the client site, e.g., Intrusion Detection Systems (IDS), to gather various information such as suspicious alerts from each client's computer network and ship them back to the Security Operation Center (SOC). Then, SOC analysts correlate and analyze the alerts to confirm whether they are successful exploits. Once a security incident is detected and confirmed as True Positive (TP), the results of the analysis need to be reported to decision-makers in a process called escalation.

In order to support network security monitoring services, the provider must collect security event logs from their customers' network for monitoring and cybersecurity protection. These logs need to be processed by the provider before displaying it to the SOC analysts. The employees generate event logs during their working hours at the customers' site. The collected event logs usually consist of personally identifiable information (PII) data, which is visible in clear text to the SOC analysts or any user with access to the Security information and event management (SIEM) platform. This unavoidably leads to a challenge in protecting the customer's privacy, while still allowing the SOC analysts to perform their investigation.

There exist a rich literature on anonymization of network-traces and netflows works [42, 43, 47, 38, 49], which is every similar to the anonymization of the event logs. For instance, in [46], the authors provide practical anonymization methods for sensitive fields such as IP addresses, ports, and raw logs. Another more recent work [48] proposes the anonymization of events logs to protect employees with higher privileges become victims of attacks which sometimes result in data breach.

In [50], the authors are interested to anonymize event logs data before outsourcing to the cloud for data mining, and they suggest four approaches: 1) k-anonymity model, 2) randomization-based, 3) encryption-based, and differential privacy-based approaches.

1.2 Objectives and Contributions

In this thesis, we study how data anonymization can be applied to security event logs to help protect individuals' privacy from the SOC analysts while preserving data utility to facilitate the analysts' investigation tasks whenever possible. Specifically,

- We base our study upon real world practices in network security monitoring at a leading managed security service provider (MSSP). We hold frequent meetings and discussions with real-life SOC analysts in order to understand their daily tasks, and to solicit their feedback on the applicability of various solutions for data anonymization.
- We collect detailed information about typical investigation tasks performed by those SOC analysts, and we then formally model such tasks as several representative use cases, which will form the basis of further research.
- Based on the use cases, we compare the impact of using different pseudonymization functions on sensitive information or PII, e.g., non-deterministic methods provide a higher level of privacy but reduced data utility.
- We also propose three different architectures and compare the strengths and weaknesses of those architectures for applying cryptographic masking methods to protect PII while enabling the security operation center (SOC) analysts to perform security log monitoring services.
- Based on our study of the architectures and pseudonymization functions, we have implemented a SIEM platform to apply and integrate different anonymization techniques for network security monitoring, and we report lessons learned from the implementation.

Our contribution in this thesis is threefold. First, based on real-world use cases, we study available architectures with different threat models, including their strengths and weaknesses. Second,

we study pseudonymization functions and evaluate their application to PII fields. Last, we implement our SIEM platform and obtain valuable feedbacks and lessons from real-world analysts. Finally, those solutions and lessons we present in the context of cybersecurity monitoring can potentially also be used in other business sectors, such as finance, health, or marketing, which might provide additional benefits.

1.3 Thesis Organization

The rest of this thesis is organized as follows:

- **Background**: In this section, we provide some background information on cybersecurity monitoring tools and methods in the context of cybersecurity monitoring.
- **Related works**: We review related works in log auditing, GDPR and HIPPA compliances, personally identifiable information (PII), and privacy by design. We also describe in this section available cryptographic schemes for pseudonymization and influential searchable encryption schemes.
- **SOC analyst use cases and extraction of PII fields**: In this section, we study five use cases of real-life SOC analysts, and we extract the corresponding PII fields in each use case. For each identified sensitive PII field, we study the application of a cryptography-based masking function.
- **Threat model and proposed architectures**: In this section, we present our threat model with all entities that are involved. We propose three architectures and compare their strengths and weaknesses.
- **Secure SIEM platform**: In this section, we detail the implementation of our SIEM platform, which is designed to enable the anonymization of PIIs. We describe our architecture with used software and used libraries. We also conduct experiments to study the CPU and memory usage with and without pseudonymization.
- **Conclusion**: We share our lessons, thoughts, results and draw conclusion.

Chapter 2

Background

In this section, we provide necessary background information on cybersecurity monitoring, different types of IDS techniques and concepts related to pseudonymization. We describe the problems we try to solve, the importance of privacy, the parties that are involved, and whose data we are trying to protect, and from whom.

2.1 Intrusion detection system (IDS)

In 1980, Anderson [3] first discussed IDS. In his work, he provides solutions using IDS to detect attacks such as masquerading, accessing to systems outside normal working hours, and he raises challenges against attackers who hide their tracks after successfully intruding targeted unauthorized systems. At that time, this was more efficient than to store these audit log records in tapes and redistribute them for analysis. In 2000, S. Axelsson [4] published a survey and taxonomy for two known detection methodologies: anomaly-based IDs and signature-based IDS, which we review in the following.

2.1.1 Signature-based IDS

Signature-based IDS works by analyzing network traffic and relies on rules which are constructed by security researchers by extracting unique signatures that match a malicious activity

uniquely. For IDS to detect attacks rules must be updated frequently (i.e., every day). The advantage of this detection method is that it can achieve a low false-positive rate, but the weakness is that rules must be up to date to detect attacks, and 0-day vulnerabilities cannot be detected because there exist no rules for them until they are publicly available. Another limitation is that if monitored network traffic is encrypted, only analysis of packet headers can be performed as headers are un-encrypted while the payload cannot be analyzed because data is scrambled (encrypted using secret-key). It implies that the majority of signature-based rules become no longer useful in such a case. Known tools for signature-based IDS are Snort [1] and Suricata [2].

2.1.2 Anomaly-based IDS

Anomaly-based IDS are based on statistics and are designed to work in two steps. The first step implies creating a profile of regular activity based on various types of information such as system usage activity (i.e., CPU, memory, disk usage), type of system calls and rate, network activity or the number of emails that a user sends in average. The second step involves detecting activity that diverges from established profiles, generally when configurable threshold limits are reached. The advantage of using anomaly-based IDS is that it can detect 0-day exploits (i.e., a malicious program which exploits a weakness in a software or a system and for which there exists no patches or fixes for mitigation) if the malware generates activity outside of normal behavior. On the other hand, anomaly-based methods require more time to deploy and efforts to tune the profiles. It also can generate a higher rate of false-positive alerts, missing well-known attacks. Related work on anomaly-based IDS methods are reviewed in the survey [12].

2.1.3 Types of IDS methodologies

There exist many types of IDS. In [5], four types of IDS methods are described and compared: Network-based IDS (NIDS), Host-based IDS (HIDS), Wireless-based (WIDS) and network behavior analysis (NBA) IDS. Each of these types has its strengths and weaknesses and are used for

[1] Snort is an open-source, free and lightweight network intrusion detection system (NIDS) software for Linux and Windows to detect emerging threats: https://www.snort.org

[2] Suricata is a free and open source, mature, fast and robust network threat detection engine. The Suricata engine is capable of real-time intrusion detection (IDS): https://suricata-ids.org/

different types of problems. We describe them as follows:

- **Network-based IDS**: Network-based IDS is used in a context where we want to collect network flows (packet headers) and perform deep packet inspection (DPI) on packet payloads. The main advantage of NIDS is that it is easy to deploy. It can collect metadata about who communicates with whom, even if communication is encrypted. It can also be used to perform rule-based IDS or deep packet inspection (DPI) when signature rules are updated. DPI requires communication to be in an un-encrypted or readable state. If a network intrusion detection system (NIDS) is deployed in passive mode, it can be connected to a mirror port of the switch and is receiving broadcast communications from all ports implying that attacks cannot be blocked in real-time, but only detected. If it is deployed in "inline" mode or active mode, the two network interfaces are connected in bridge mode, and all the traffic that is going through allows the sensor to monitor network traffic actively and drop or block potential attacks in real-time.

- **Host-based IDS**: Host-based IDS is also known as agent-based IDS, which is a software installed on the end-point devices (e.g., desktop, laptop, servers). HIDS has more advantages over NIDS because more information can be collected about a host. For example, end-to-end communication can be analyzed in an unencrypted format, system calls can be extracted and more accurate information about the OS, installed updates, and list of installed software, including their version can be identified. On the other hand, HIDS requires more effort to deploy because they need to be installed on each device and consumes more resources from the host where it has been installed.

- **Wireless-based IDS**: A survey [7] by I.Butun, D. Morgera and R. Sankar describes security concerns in wireless networks and various types of attacks. Detection techniques are similar to network-based IDS, but attacks can be detected with dedicated sensors for WiFi protocols. There exist various types of attacks, such as evil twin access-point (AP) attacks [8], rogue AP, DoS attacks [10] or clone attacks [9].

- **Network Behavior Analysis (NBA)**: Network behavior analysis system analyzes network

traffic flows which can detect unusual network activity such as DoS attacks, a malware attempting to spread in the local network, a high rate of new incoming and outgoing connections originating from external networks, unusual network activity outside usual working hours or policy violations such as hosts trying to connect to critical servers. NBA can detect sophisticated attacks and stealthy network discovery scans and can be deployed across the whole network. Its weaknesses are that it can be slower to react when it needs to detect known attacks.

2.1.4 Security Monitoring

Collected raw logs can be parsed and analyzed on the sensor level, which is located at the customer's network or it can be transmitted as raw logs through a secure communication channel such as a virtual private network (VPN) from customers networks to the MSSP's data flow component for parsing and processing. There exist many tools for log parsing and normalization such as rsyslog [3], syslog-ng [4], fluentd [5] or liblognorm [6].

Raw logs are collected from sensors and aggregated to a centralized system for the processing, which involves parsing, normalizing, analyzing, and correlating. Identified attacks or potential threats are displayed to the SOC analysts as alerts. Other security controls can also be applied to improve protection (e.g., firewalls) and detection (e.g., performing vulnerability scans routinely, honey-pots).

Raised alerts are displayed on the user interface by the security information and event management (SIEM) centralized platform, which is used by the SOC analysts during their duty hours. In S. Kumar's work [6], four different activity states are described: intrusive and anomalous, not intrusive but anomalous, not intrusive, and not anomalous, intrusive and not anomalous:

- **True-positive (intrusive and anomalous)**: when a real attack is accurately identified by an intrusion detection system (IDS).

[3] RSYSLOG is the rocket-fast system for log processing: https://www.rsyslog.com/

[4] syslog-ng is an enhanced log daemon, supporting a wide range of input and output methods: syslog, unstructured text, message queues, databases (SQL and NoSQL alike), and more: https://github.com/balabit/syslog-ng/

[5] Fluentd: Open-Source Log Collector: https://github.com/fluent/fluentd

[6] Briefly described, liblognorm is a tool to normalize log data: https://www.liblognorm.com/

- **False-positive (not intrusive but anomalous)**: when a benign activity caused an alert to be raised.
- **True-negative (not intrusive and not anomalous)**: when a benign activity is successfully identified as benign where no alerts are raised.
- **False-negative (intrusive but not anomalous)**: when real attacks have occurred but were considered as benign or were not detected by the IDS. This state is usually considered to have the most serious consequences.

Analysts must investigate and analyze alerts and events to decide if there are enough facts to conclude the presence of successful attacks or threats.

An incident is created when sufficient facts are gathered pointing to an attack or threat and all related information (i.e., alerts, events, vulnerabilities) are linked to this incident.

Figure 2.1 shows an example where MSSP deploys its sensors on the customers' site for cybersecurity monitoring.

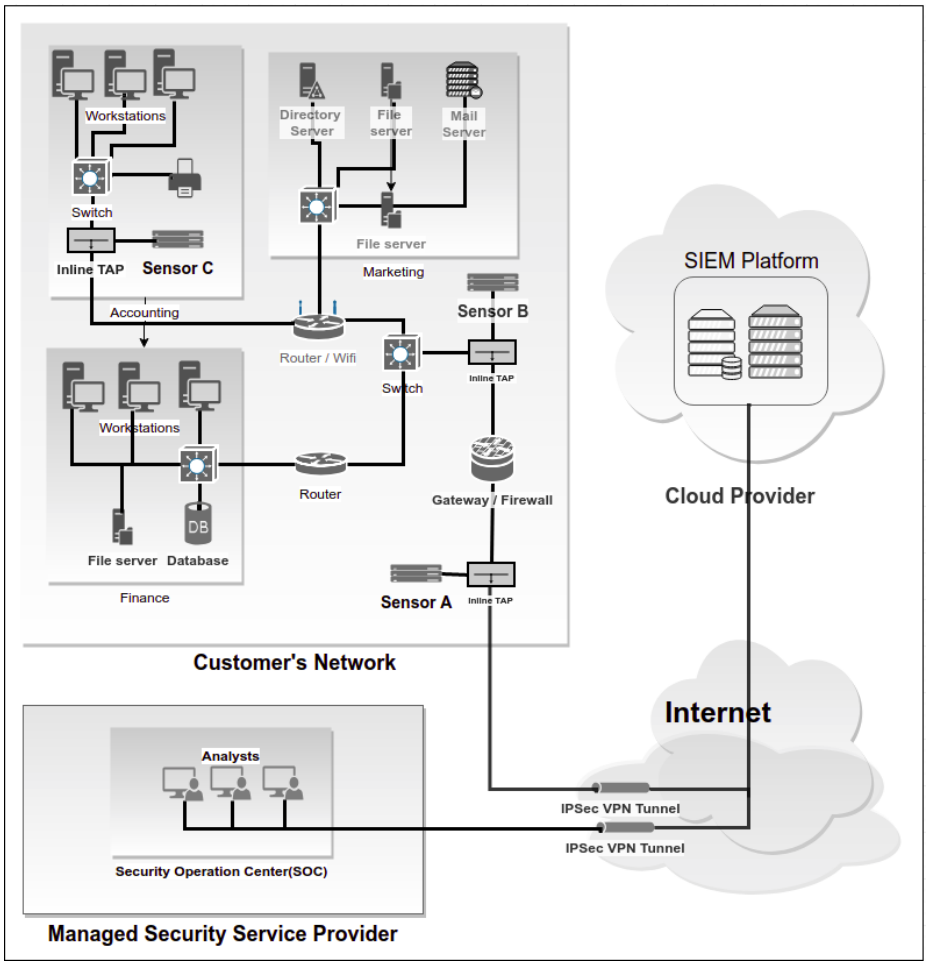

Figure 2.1: MSSP deploying its sensors at the strategic locations to monitor customers network

2.2 Managed Security

Managed Security Service Providers (MSSP) are organizations specialized in providing cybersecurity monitoring services. Since not all companies might have sufficient time, money or expertise to build its in-house cybersecurity team, small to medium companies can usually choose to outsource this responsibility to MSSP which is specialized in cybersecurity protection by offering its services 24/7 [1].

2.2.1 Managed Security

Cybersecurity firms offering managed security services consist of identifying critical assets that need to be protected and monitored, which is located on the customer's network infrastructure.

Data generated by these devices are then collected, analyzed, and escalated to respond to intrusions, which were defined by Richard Bejtlich as "Network Security Monitoring" (NSM) [2]. MSSPs offers NSM services by employing intrusion detection systems (IDS) tools to detect cybersecurity attacks and protect against unauthorized access to systems or networks.

Sensors that are analyzing the traffic in real-time are called Intrusion prevention system (IPS) and are characterized when two network interfaces are "bridged", where all traffic passes through the sensor for packet analysis. It enables the sensor to take action in real-time by dropping packets characterized as potential attacks. Sometimes, one network interface is used for management and one for events and alerts transmission. Sensors can also be used to forward logs generated by customer's monitored devices for analysis and correlation on MSSP's infrastructure.

2.2.2 Data breaches

Since security monitoring may involve accesses to sensitive information, it may lead to data breaches. Latest breaches have cost millions of dollars in damage and millions of private identifiable information (PII) leaked to the public. Yahoo data breach in 2013 has resulted in over three billions of accounts being compromised [13]. Equifax reported 143 millions of US customers being affected consisting of names, SSN, addresses, birth dates and in some cases, driving license [14]. In 2016, Uber reported that 57 million customers and drivers, which includes names, email addresses, mobile phone numbers and within that number, 600,000 names and driving license were affected [15]. In 2016, Linkedin discovered that additionally 100 million email addresses and hashed password (unsalted SHA-1) was disclosed [16], instead of 6.5 million as believed first, currently totalling 165 million [17]. Additionally, many organizations were selling sensitive personal information to other organizations who were willing to pay without customers consent. A website haveibeenpwned.com [7] offers users for free to verify if they have been victim of the data breach by checking if their email address exists in the breached data sets.

[7]Have I Been Pwned: Allows you to verify if your email address has been compromised in a data breach: https://haveibeenpwned.com/

We assume the parent company owns the Security Provider (SP). Any other user would have insufficient permission to read data which includes PII in clear-text format. Any individual or organization might be interested to steal sensitive collected data by the Security Provider systems, directly or indirectly. Here stealing directly could be by attacking the SP's platform or infrastructure, and stealing indirectly could be by attacking the SP through its customers' or any user having permissions into the systems.

2.3 Privacy

2.3.1 Privacy and PII

Ross Andreson[8], compares the difference between confidentiality and privacy as follows:

> "Confidentiality involves an obligation to protect some other persons or organizations secrets if you know them. Privacy is the ability and/or right to protect your personal secrets; it extends to the ability and/or right to prevent invasions of your personal space (the exact definition varies quite sharply from one country to another). Privacy can extend to families but not to legal persons such as corporations." [9]

NIST references United States Code (U.S.C.) to define confidentiality:

> "Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information."[10]

Personally identifiable information (PII)

Since our work focuses on protecting data with PII, we look at how PII is described in the following privacy laws and directives:

- The Personal Information Protection and Electronic Documents Act (PIPEDA) in Canada
- The Health Insurance Portability and Accountability Act (HIPAA) in United-States (US)
- General Data Protection Regulation (GDPR) in European Union (EU)
- National Institute of Standards and Technology (NIST), Guide to Protecting the Confidentiality of Personally Identifiable Information (PII), SP 800-122

PIPEDA is a Canadian law that governs how private businesses should collect, use and disclose personal information[11]. In PIPEDA, PII is defined as follows:

[8] R. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, Wiley, 2001
[9] Anderson, p. 10
[10] 44 United States Code, Section 3542(b)(1)(B), 2008
[11] Personal Information Protection and Electronic Documents Act, S.C. 2011, c. 5 3 (Can.). PIPEDA regulates the use

> "Personal information is data about an "identifiable individual". It is information that on its own or combined with other pieces of data, can identify you as an individual."[12]

Examples of PII are given as:

- race, national or ethnic origin, religion,
- age, marital status,
- medical, education or employment history,
- financial information,
- DNA,
- identifying numbers such as your social insurance number, or drivers licence,
- views or opinions about you as an employee.

Health Insurance Portability and Accountability Act (HIPAA) establishes policies for maintaining privacy and security of individually identifiable health information in U.S. It consists of HIPAA Privacy Rule and Security Rule. Under Privacy Rule, it regulates the use and disclosure of what is known as protected health information (PHI), similar to PII. Examples of PHI are: Individuals past, present, or future physical or mental health or condition Provision of health care to the individual Medical record, laboratory report, or hospital Names Addresses Birth dates, Social Security Numbers.

On the official website (HSS.gov - Methods for De-identification of PHI 2015), a guidance is given on how to proceed with de-identification of PHI in accordance of HIPAA. HIPAA Privacy Rule gives its citizens rights to inspect, review, and receive a copy of its medical and billing records which are held by health plans and health providers.

National Institute of Standards and Technology has published a guide "Guide to Protecting the Confidentiality of Personally Identifiable Information (PII) (SP 800-122)", and according to their guide, PII is defined as follows:

---

of personal information by federal organizations and data flows between Canadian provinces. Id. 23(1)23(3).

[12]Summary of privacy laws in Canada, Office of the Privacy Commissioner of Canada: https://www.priv.gc.

> *"PII is any information about an individual maintained by an agency, including (1) any information that can be used to distinguish or trace an individuals identity, such as name, social security number, date and place of birth, mothers maiden name, or biometric records; and (2) any other information that is linked or linkable to an individual, such as medical, educational, financial, and employment information."* [13]

NIST goes further by providing details on how to quantify PII in an organization because a breach of 25 records does not have the same impact as 25 million records. It also provides details about the importance as SSN is more sensitive than a phone number or a ZIP code. NISTs guide also provides a concrete example of network monitoring with its second example of "Intranet Activity Tracking", section 3.3.2. Here are some examples of PII fields that can be collected from intranet networks:

- The user's IP address
- The Uniform Resource Locator (URL) of the web site the user was viewing immediately before coming to this web site (i.e., referring URL)
- The date and time the user accessed the website
- The web pages or topics accessed within the organization's web site (e.g., organization's security policy).

General Data Protection Regulation (GDPR) is a regulation about data protection on the privacy of all individuals within European Union countries. It takes the place of previous known Data Protection Directive and is suppose to take effect starting from 25th May 2018. GDPR main goal is to protect and give control of personal data to EU citizens by encouraging companies to follow "privacy by design" approach during the software development life cycle (SDLC). This regulation affects companies outside of EU, such as Facebook or Google, because they are managing personal data from individuals within the EU.

In GDPR, it is suggested to use pseudonymization and encryption of personal data. Pseudonymization is defined under GDPR as:

ca/en/privacy-topics/privacy-laws-in-canada/02_05_d_15/#heading-0-0-1/

[13]This definition is the GAO expression of a amalgam of the definitions of PII from OMB Memorandums 07-16 and

pseudonymization means the processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information, provided that such additional information is kept separately and is subject to technical and organizational measures to ensure that the personal data are not attributed to an identified or identifiable natural person;

GDPR's definition of PII

GDPR defines personal data as follows:

"'personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, the physiological, genetic, mental, economic, cultural or social identity of that natural person;"[14]

Defense of Homeland Security (DHS) has published its Handbook for Safeguarding PII and it has grouped them in two categories: direct(stand-alone) PII and indirect(if paired with another identifier).

2.4 Anonymization, pseudonyimzation and searchable encryption

2.4.1 Pseudonymization and anonymization

There is a subtle difference between pseudonymization and anonymization. NIST defines anonymization as follows:

06-19. GAO Report 08-536, Privacy: Alternatives Exist for Enhancing Protection of Personally Identifiable Information: http://www.gao.gov/new.items/d08536.pdf.

[14]European Parliament and Council, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance), Chapter 1, Article 4," May 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX 32016R0679&from=EN#d1e1489-1-1

> "process that removes the association between the identifying dataset and the data subject." [15]

with a more detailed explanation in reference to ISO's ISO/TS 25237:2008(E).

> "NOTE Anonymization is another subcategory of de-identification. Unlike pseudonymization, it does not provide a means by which the information may be linked to the same person across multiple data records or information systems. Hence reidentification of anonymized data is not possible."

Definition of pseudonymization and comparison with anonymization is given by ISO [16] as follows:

> Pseudonymization (from pseudonym) allows for the removal of an association with a data subject. It differs from anonymization (anonymous) in that, it allows for data to be linked to the same person across multiple data records or information systems without revealing the identity of the person. The technique is recognized as an important method for privacy protection of personal health information. It can be performed with or without the possibility of re-identifying the subject of the data (reversible or irreversible pseudonymization).

While GDRP's definition of pseudonymization is as follows:

> 'pseudonymization' means the processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information provided that such additional information is kept separately and is subject to technical and organizational measures to ensure that the personal data are not attributed to an identified or identifiable natural person.

[15] ISO (International Organization for Standardization). Health informatics-Pseudonymization. Geneva, Switzerland: ISO; 2008. (ISO/TS 25237:2008).

[16] International Organization for Standardization - "Pseudonymization" new ISO specification supports privacy protection in health informatics: https://www.iso.org/news/2009/03/Ref1209.html

Definition of anonymization has stronger ties to privacy than pseudonymization because it supposes that the re-identification of anonymized data is impossible even if the attacker attempts to link data to the same person across multiple data records, as defined in NIST's guide which implies a reduction in data utility.

Pseudonymization, on the other hand, can be achieved by using encryption schemes such as AES encryption scheme adapted for deterministic encryption because we need to keep consistency between the individual's data.

We conclude that pseudonymization is weaker than anonymization because of two reasons. First, because it is deterministic, but anonymization should be non-deterministic for higher confidentiality. Second, pseudonymized information can be de-anonymized to its original value based on a secret key(i.e., two-way function encryption function) while we consider anonymization as a one-way function and once anonymized, cannot be de-identified or de-anonymized.

2.4.2 Anonymization

Anonymization is a process of de-identification of data about an individual on a dataset. It is generally applied before publishing data to the public. Anonymization of health-care data has been done very often in the past for research studies. Two known algorithms for anonymization are k-anonymity [26] and Differential Privacy(DP) [28].

k-Anonymity

A dataset is said to have k-anonymity of at least k-1 when each record is indistinguishable from the other k-1 records. This is achieved through suppressing, generalizing or aggregating fields for which a set of policies have been set. [26, 27] anonymization consists of the following policies:

- Suppression: fields are replaced with a constant or are completely dropped.
- Generalization: fields are replaced with more generic values such as a range of approximate general value

Differential Privacy

The authors [35] describe Differential Privacy(DP) as "Differential privacy is a formal mathematical framework for guaranteeing privacy protection". DP has been created to provide more robust protection against de-anonymization attacks such as adversaries with auxiliary databases or information, against linkage attacks. Linkage attack is defined in [28] as follows: "re-identification of one or more records in a de-identified dataset by uniquely linking a record in a de-identified dataset with identified records in a publicly available dataset." The intuition behind DP is to output data in a way that even if data about one individual is added or removed, anonymized results will not deviate too much from the original dataset. DP is a probabilistic anonymization technique and uses Laplace distribution for noise calibration.

2.4.3 Searchable encryption

Searchable encryption (SE) is another line of research. SE enables users to store data on a remote database server in an encrypted format while preserving searching capabilities. Data and indexed keywords are encrypted from the client-side and are sent to the server residing on the cloud. The secret key always resides on the client-side and is never shared with the server. The user can query the server by re-encrypting his/her keywords and sends them to the server to be searched on the encrypted index. Finally, matched results are returned to the client-side to be decrypted. Generally, a symmetric encryption (e.g., AES) type is used because it can achieve better encryption and decryption performances. Many works rely on deterministic encryption so that ciphertexts can be compared to each other inside the databases. Searchable encryption is useful in contexts where sensitive data is stored to a cloud provider, who is a trusted but curious party.

Order Preserving Encryption (OPE)

Order preserving encryption (OPE) was introduced in [56] in 2004. OPE algorithm is constructed based on "hypergeometric distribution", a discrete probability distribution used in statistics. Their scheme is limited because it is only applicable to numeric data and has not provided any

provable security definition. Boldyreva et al. [57] presented their OPE scheme with IND-OCPA (indistinguishability under ordered plaintext attack) and came with an ideal goal as: order-preserving encryption should not leak anything other than the order of clear text messages and ciphertexts. In other means, if message a $>$ b, then enc (a) $>$ enc (b) . This implies that OPE scheme is weaker than the deterministic encryption (DET) scheme because it leaks the order of plaintexts. In 2013, Boldyreva et al. [58] published an improved version of OPE scheme where they described that their previous scheme [57] leak not only the order between cleartexts and ciphertexts but also the distance and location between them. They proposed a new Modular OPE scheme which adds a secret modulo on the message space during message encryption hiding the location data points. Popa et al. [59] came with a mutable Order Preserving Encryption (mOPE) which essentially modifies some encrypted ciphertext for some message values, more specifically when B-Tree data structure used in mOPE need to be balanced during the merge or split operations of a node. P. Grubbs et al. [66] and N. Mohammad et al. [68] have published attacks and raised many further concerns in previously published OPE schemes.

Format Preserving Encryption (FPE)

Format-preserving encryption (FPE) is an encryption scheme that generates output (i.e., ciphertext) in the same format as the input in plaintext [61, 62]. The motivation behind FPS originates from problems where legacy systems require a specific format of inputs to be provided to operate, for example, the social security numbers or credit cards used in banking systems. Credit card number consists of 16 digit numbers (e.g., 1234567890123456) only whereas AES encryption scheme generates binaries which are encoded into base64 (e.g., MTIzNDU2Nzg5MDEyMzQ1Ng==) or hexadecimal value (e.g., 0x313233343536373839303132333435360a) which are alphanumeric and different lengths. Format-Preserving Encryption, they describe as FPS as follows:

> The goal is this: under the control of symmetric key K, deterministically encrypt a plaintext X into a ciphertext Y that has the same format as X

Brightwell and Smith have been the first to introduce formal FPE in 1997 [61]. Today, this area of research is also more active because the demand for FPE has increased dramatically in the last

years. Black and Rogaway have contributed by introducing three methods: “Prefix Cipher”, “Cycle-Walking Cipher” which is similar to prefix cipher and “Generalized-Feistel Cipher” based on Feistel construction and Data Encryption Standard (DES) as the underlying pseudo-random function (PRF). Improved version of FPE are constructed based on Feistel network AES instead of DES, which offers better security. There exist already three modes of FPE schemes submitted to National Institute of Standards and Technology (NIST). They are abbreviated as FF1, FF2 and FF3 which were submitted to NIST under the name FFX[Radix] , VAES3 and BPS respectively. FF1, FF2 and FF3 are Feistel-based encryption modes, but the National Security Agency (NSA) has advised NIST that FF2 did not provide the expected 128 bits of security strength which resulted in its removal and has been confirmed by NIST cryptographers later. On 12 April 2017, NIST concluded that FF3 is also no longer safe because of the vulnerability which has been identified by B. Durak and S.Vaudenay.

Hashing SHA-2

Hash function is a one-way (i.e., hash value cannot be decrypted to get back to the original value) function which generates a fixed-size hash value. We do not rely on hash function because in some use cases, customers (data owners) might request to be able to see data in a clear text format. This case causes conflicts with hash functions because once sensitive information is hashed and stored in a database, we cannot get back the original value if we do not store somewhere the original value. In our case, we might get original value from a raw log field. Keyed-hashing (secret information) and salts (public information) can be used as measures to reduce attacks on small domains, or guessing attacks using precomputed hash tables.

Deterministic encrytion

Deterministic (DET) encryption is a form of encryption where for the given two same inputs: message m and secret-key sk, will output identical ciphertext c. It is generally used in searchable encryption schemes so that encrypted indexed keywords can be matched when user submits a query (search)[74, 64]. Searching on encrypted indexes requires an encryption scheme to be deterministic and can be used in AES-CBC mode by fixing the initialization vector (IV) to the same consistent value, which makes it insecure a result. The reason is that it does not meet semantic security and

does not resist a chosen-plaintext attack (CPA) attack model of cryptanalysis, in which the attacker can choose his/her arbitrarily plaintext to encrypt and will receive its ciphertext. Deterministic encryption is vulnerable to statistical attacks, inferencing and when cross-referencing but has been used very often in searchable encryption [67] and searchable encrypted databases [64].

IP traces anonymizers

Greg Minshall proposed TCPDpriv in 1996 to anonymize IP traces which are collected by monitoring and listening on network interfaces. One way to collect network traces is by using Tcpdump, a command-line packet-analyzer available on Unix-like operating systems. TCPDpriv keeps track of a pair of original and anonymized IP addresses in a table. When a new IP address is encountered, it looks in its cache to check if it has ever been seen earlier. When a match occurs, the prefix of the IP address is anonymized using the same longest prefix of the existing anonymized prefix and the remaining parts of the IP addresses are generated randomly. This implies that anonymized IP addresses are generated non-deterministically [37].

Crypto-PAn [51] was later introduced as an improvement to TCPDpriv. Crypto-PAn is a cryptographic prefix-preserving pseudonymization tool. It relies on AES symmetric cryptography-based encryption scheme to anonymize IP addresses which implies that it uses a secret key to anonymize an IP address. Prefix-preserving property is necessary to preserve the utility of the IP traces by preserving the relationship of IP addresses to their network segments. Crypto-PAn generates outputs deterministically, which is a useful attribute if searching functionality is needed. Crypto-PAn has only a one-way function and this is why we do not consider it as a pseudonymization function, since we cannot get the original IP address value based on the anonymized IP address and its secret key.

CryptopANT [31] is an updated version of Crypto-PAn based algorithm which has been implemented by University of Southern California (USC)/ISI ANT project with many additional features. It offers a two-way function, implying that with a correct secret-key, we can anonymize an IP address or de-anonymize the already anonymized IP address back to its original state. CryptopANT

offers other functionalities such as pseudonymization of IPv6 and MAC addresses, caching for better performances and multiple encryption functions such as Blowfish, AES, SHA1SUM, MD5SUM. It is written in C language, and its source code is available from its official website [31] and GitHub [32]. The drawback is that there is no official document that describes its limitations, weaknesses or formal security analysis. We prefer CryptopANT over Crypto-PAn because CryptopANT is a two-way function allowing us to retrieve the original value from the pseudonymized one, whereas with Crypto-PAn we cannot.

Active attacks as demonstrated in [39] can defeat anonymizers such as Crypto-PAn or CryptopANT because of their deterministic attributes. We refer this attack to the chosen-plaintext attack (CPA) model in cryptanalysis.

The anonymization of IP traces can be useful in situations were signature based IDS are used to detect attacks and which will also record related packets. The analysts will then use any packet analyzer to do their analysis to learn more about the attack. However, as these packets consist of sensitive information, we also must not anonymize fields that might contain sensitive personal data. This refers to the action step 9 from the SOC analysts activity diagram (Section 5.1).

SCRUB-tcpdump [38] is an anonymization tool that extends multiple functionalities to Tcpdump. It offers different types of settings and granularity to which various fields can be anonymized and not only limited to the IP address field as in many other related works. For example, fields that can be anonymized are MAC addresses, network types, PCAP timestamps, IP addresses, IP transport protocol, ports, and various data types found in the payload such as IP address, hostname, URLs, filenames, email addresses, common people's names, DNS queries, DHCP queries, ARP requests, HTTP/HTTPS headers, SMTP headers RTP/RTCP information, FTP fields, IM information. For each of these fields, different anonymization options are available such as: left alone (no anonymization), black marker, random shifts (for timestamps), values set to 0 (for numeric values). They cite the work [40] by Coull et al. in which they describe that between 28% and 100% of targeted servers were de-anonymized by using 1) sensitive network topology and 2) monitored hosts behavior information such as DNS records, public information from search engines. These two information was

extracted using only anonymized IP addresses and ports and this is why SCRUB-tcpdump focuses on offering a wide range of fields that can be anonymized. In [42], the authors proposed a tool called "Framework for Log Anonymization and Information Management" (FLAIM) to anonymize collected network traces [43] to measure the utility and privacy tradeoffs. They concluded that it is mostly the question of which sensitive field was anonymized and not which anonymization method was applied that affects the utility of anonymized network traces. The most important fields were the IP addresses and ports.

In a more recent research work by Mohammady et al. [47], they present a multi-view based solution to solve Crypto-PAn's vulnerability to semantic security cryptanalysis. In their work, they shift the tradeoffs from privacy and utility to privacy and computational costs. The intuition behind their work is to create fake views (groups of packets) with fake collisions that share prefix-preserved IP addresses becoming indistinguishable to the attacker from real views. Their work is inspired by differentially-privacy (DP) methodology.

We believe that their approach is too limited because they focus solely on the IP address field to protect user's privacy, which is rarely the case in the real world use case scenarios. As described in [42, 68], other fields such as ports, payloads and other packet's signatures can tell a lot about monitored devices and relying only on IP address is not sufficient for privacy protection. It requires deeper analysis with a more holistic approach by taking other fields (i.e., potential indirect identifiers) into consideration.

There exist many de-anonymization works [44, 45] showing that anonymization is harder than thought because even when sensitive data is anonymized, other attributes, such as full name, phone number, address, institutional identifiers or any other field could still lead to personal identification. Researchers were still able to single out records and apply to cross-referencing on available auxiliary information.

Chapter 3

Related works

Anonymization on security logs

In [46], Zhang et al. present how collected security monitoring logs with sensitive PII can be anonymized. In their work, they present three methods for each of the following fields they consider as PII: IP address, timestamps and open ports.

Specifically, for the IP address field, they propose:

- Truncation: This technique is based by dropping the rightmost (least significant) bits of the IP address so that the network range is preserved, but the exact device cannot be identified.
- Random permutation: This method will generate a random IP address in a deterministically based on the seed. The weakness is that the prefix of the IP address is not preserved means that two IP addresses from the same network will not result in the anonymized IP address with a shared network range.
- Prefix-preserving pseudonymization: Prefix-preserving algorithms already exist such as Crypto-PAn [51], TCPdpriv [29] and CANINE [30]. They are used for different use cases, for example, TCPdpriv anonymizes private information from Tcpdumps whereas Crypto-PAn anonymizes IP address but is a one way function because it does not offer the possibility to retrieve an original IP address with the secret key.

For the field timestamp, they propose:

- Time unit annihilation: Timestamps and duration units such as year, month, day, hour, minute and seconds can be dropped. In the case of minute unit annihilation of a timestamp, related durations must be adjusted accordingly.

- Random time shifts: In random time shifts, all related timestamps events can also be shifted so that relative date and time are preserved in the relation of each event. Relative date and time are of greater value than knowing exactly when an event has occurred.

- Enumeration: This method consists of applying random time on the earliest event timestamp and shifts all following events timestamps equidistantly and retaining the same sequence order of the events. One weakness when applying this method is if collected event logs are not arriving in pre-sorted time.

Also for the field 'open ports', they propose three types of anonymization methods:

- Bilateral classification: If the open port number is smaller than the port 1024 (well-known ports), they are classified as 'known services' and greater than the port number 1024 is considered as 'ephemeral ports'. This method is analogous to IP address 'truncation' method.

- Black marker: In this method, open ports are replaced with a constant such as no port number information is revealed.

- Random permutation: Random permutation implies replacing any port number (0 to 65,535) to another mapped port number. The authors also warn that the weakness of this method is that the adversaries are able to determine the real port number by analyzing port patterns or by analyzing their frequency. For example, any frequent random port number can be mapped to open port 80 or 433, because these are well-known ports which are used when users are navigating the web.

Searchable encryption

The first practical SE scheme was introduced in [69] in 2000; the authors demonstrate how to construct a SE scheme in a single-writer and single-reader setting. It encrypts each word in the same order as they are encountered in the document without the use of any index, making it inefficient

for searching but more simple because no index needs to be updated. The size of the ciphertext also increases very quickly because it needs to be normalized to a fixed-size length (e.g., 32 bytes). The weaknesses are that the attacker can discover which document has which words after several queries due to statistical attacks and the access location of the encrypted words in the encrypted document are leaked, but the authors are aware of this issue and propose to randomize the ciphertexts location for each word.

An index-based keyword search was introduced in 2003 by Goh [70] and by Chang and Mitzenmacher [71] in 2005. This solves the problem of fixed-size ciphertext length. Their works greatly improved searching performance to constant time. Goh uses a secure z-index data structure for the index per document built on top of Bloom filters and pseudorandom functions PRF. Bloom filters enable searches in constant time and are practical for applications that need to search on a large amount of data. On the other, Bloom filters have only the function append (or add) and can return false positive results because it verifies if the searched value is: (1) possibly in the set or (2) certainly not in the set.

Chang and Mitzenmacher [71] on the other hand make use of masked index based on cryptographic primitives (e.g., Pseudorandom Function PRF and Pseudorandom Permutation PRP) in which for each document we can find a set of all words from all documents. Each document will have a pseudorandom string Gri (1) XOred to a binary 0 or 1 depending on the presence of n word in the document. If "1" is returned, then the server will return "E(k, D1)" as output. Because each document will have the same number of encrypted words for all documents, the attacker is not able to deduce the size of keywords in each document.

In 2006, Curtmola et al. [72] introduced an inverted index with key as unique words and value as sequence of document identifier (e.g., "word1":["identifier1"], "word2":["identifier1", "identifier2"]). Their cryptographic scheme offers search time in a sublinear time. They achieve a stronger security threat model with an adaptive adversarial model. Their data structure consists of an array of distinct keywords w(i) which contains a linked list of documents of node consisting of (documentId—key(i,j)—nextNodePtr) where documentId is the document's identifier, key(i,j) for i-th word

of j-th document and nextNodePtr, a pointer to the next node. This design and structure restricts parallel computation and is of great complexity for implementation.

In 2013, David Cash et al. [73] proposed a scalable SSE protocol which can scale to a very large data sets. It supports sub-linear searching capabilities with conjunctive and boolean queries. Their structure relies on the hardness of the decisional diffie-hellman (DDH) problem by precomputing the encrypted database index to achieve oblivious computation. Using this method, their client can achieve a single round of interaction with the server, thus reducing the leakage to the adversary on the server-side.

A survey by Bösch et al. [74] provides a general overview of SE with two main techniques: Searchable Symmetric Encryption (SSE) and Public Key Searchable Encryption (PKSE) whereas the recent survey Poh et al. [75] covers SSE more in-depth, which also provides excellent general high-level categorization.

Homomorphic Encryption

Homomorphic Encryption (HE) was first introduced in 1978 by Ronald L. Rivest et al. [76] at the MIT shortly after R. Rivest, A. Shamir, and L. Adleman have invented the asymmetric cryptographic algorithm, RSA R. L. Rivest et al. [77]. HE is a powerful form of encryption, enabling computation on encrypted data directly, which generates new encrypted results that when decrypted will match as if the evaluation has been executed on clear data. HE is one of the most desired kinds of encryption because it is the most general solution that can be applied in many different contexts. In 2009, Craig Gentry introduces a first viable solution [78] to Fully Homomorphic Encryption (FHE) problem for his phD dissertation. He describes FHE as follows: "At a high-level, the essence of fully homomorphic encryption is simple: given ciphertexts that encrypt $\pi 1, \pi 2, \ldots, \pi t$, fully homomorphic encryption should allow anyone (not just the key-holder) to output a ciphertext that encrypts $f(\pi 1, \pi 2, \ldots, \pi t)$ for any desired function f, as long as that function can be efficiently computed. No information about $\pi 1, \pi 2, \ldots, \pi t$ or $f(\pi 1, \pi 2, \ldots, \pi t)$, or any intermediate plaintext values, should leak; the inputs, output and intermediate values are always encrypted." [78]

In previous research works, lots of contributions were made on Partially Homomorphic Encryption (PHE) or Somewhat Homomorphic Encryption (SHE), a weaker set of HE. All previous works

allowed one type of operation: addition or multiplication, characterizing them as PHE, whereas SHE combines addition and multiplication but is limited in the number of operations that can be performed and this is why PHE and SHE are a weaker type of encryption. This is why Craig Gentry's breakthrough has been significant awakening the research community to explore more deeply the FHE line of research. HE has been used in searchable encryption problems but has been less attractive because HE imposes high computational costs. Consequently, interests are shifting to other cryptographic schemes (i.e., SSE, OPE/ORE, ORAM) which are less secure but more practical and efficient.

Chapter 4

Threat model and architectures

In this section, we first provide an overview, and then explain our threat model and present our architectures

A main challenge to the existing practice by MSSPs is that a large volume of event logs are collected and recorded or persisted by MSSPs without applying any de-identification mechanisms or privacy protections. A large number of collected event logs consist of personally identifiable information that is not sufficiently protected in case of data breaches (a few data breaches are mentioned in the section below).

It is a challenging problem because security operation center (SOC) analysts sometimes need to see sensitive information in clear while at the same time such information clearly requires to be protected against privacy violations, a problem implies tradeoffs between privacy and utility.

Collected logs are analyzed by the SOC analyst team to identify any potential attacks and mitigate them when necessary. SOC analysts are not required to see PII information in the clear at all the time during their duty. We will explore in detail how frequently the SOC analysts might need to know sensitive information about its customers and in which situations.

This problem is important since the disclosure of sensitive information may have serious legal and financial consequences. For instance, under General Data Protection Regulation (GPDR), companies can be fined of up to 2 million euros or %4 of annual global turnover, whichever is higher

if they do not comply. GDPR is more strict and general privacy regulation than "Health Insurance Portability and Accountability Act" (HIPAA) because it affects businesses across all sectors, while HIPPA affects only the health sector in the US. Organizations must be GDPR compliant if they are doing business in the EU and are collecting sensitive information about citizens living in the EU.

More specifically, in article 25 of GDPR [18], entitled "Data protection by design and by default" organizations that are processing sensitive data must implement appropriate technical measures to reduce any risks or likelihood affecting natural person rights and freedom. This requirement is similar to "privacy by design" [1] principles. These measures include data minimization, collecting only data about individuals that are needed for a specific purpose, privacy settings need to be activated as default settings and use encryption and pseudonymization to protect collected sensitive data. It also requires organizations to implement measures such that by default, sensitive data is protected and available only to a limited number of people that are required to see collected sensitive data for a specific amount of time.

[1] A. Cavoukian.: Privacy by design the 7 foundational principles. Technical report, Information and Privacy Commissioner of Ontario (January 2011) (revised version)

4.1 Objectives

Our main goal is to protect any personally identifiable information in logs while facilitating the security analysis. A large number of event logs and security systems logs collected by the security provider from its customers' network consist of PII. These PII can be found in network flows even if the communication is encrypted. PII can be found in software and application logs, system security logs and other middleware devices logs such as firewalls or VPNs.

In our context, we focus principally on event logs that are generated by individuals and employees during their work time (e.g., browsing the Internet, email communications, instant messaging communication, connecting to VPNs). PIIs that are collected by the security provider from its customers can also originate from cloud providers (e.g., Amazon Web Services, Google Cloud, Microsoft Azure) if the customer has critical, valuable assets that need cybersecurity monitoring protection.

Sensitive logs can contain information from customers' clients too. For example, a security provider's customer could be a vehicle insurance company or a telecommunication company that might be willing to share its logs relating its customers against insider threat attacks to be monitored. These might include PII from a car owner consisting of names, addresses, vehicle identification number (VIN), plate numbers.

Sensitive PII information such as names, IP addresses, URLs, GPS locations or browser signatures collected from various data sources are not always initiated by an individual's activity or cannot be related to an individual. Today as we have increasingly more automated software (programs) and IoT devices designed to exchange data and are triggered based on a recurring time schedule and based on a specific event from the external world detected by its sensors. We cover more in detail other identified fields based on the use cases in section 6.1.

4.2 Threat model description

As seen below, our threat model consists of four components. The first component, “Security Provider”, is considered as trusted (in green) and it collects sensitive personal data from multiple organizations, which are its customers.

The third-parties are providing REST API solutions for threat intelligence. This component interacts with Security Providers’ platform by querying the third-party services to verify if any sensitive personal information has been breached or has any relations with any malicious adversary. “Parent Company of SP” would require to have access to the security providers platform for the general overview and security posture of its customers and in order to be able to track the performance of its analysts.

In our threat model, we consider that analysts are honest but curious, and we explore the area if analysts are still going to be able to analyze collected security logs that are anonymized. Similarly, for users or managers from sister or parent company or third-party services’ are not trusted and should not have the permission to see data in clear, but only anonymized. Trusted users, in green colour, can read all data in a clear data format, including sensitive PII data. Trusted users need to have access to the platform to consult monthly reports, see collected events and raised alerts or consult incidents. Trusted users are generally in a limited number per team, with higher privileges and higher roles than, for example, SOC Analysts.

The platform simulates data ingestion from its sensors and events logs are transmitted through Apache Kafka. It processes them in Apache Spark and persists them in Cassandra database. We also built a user interface web application in Angular to retrieve data from the Apache Cassandra database using Java Spring Framework REST API. Our main goal is to examine the possibility of allowing analysts to analyze collected security event logs enabling them to continue to do their duty as SOC analysts they usually would.

4.2.1 Threat model

We assume that collected event logs in clear text, which orginate from customers' systems and networks, needs to be protected against the following users or systems:

- SOC Analysts
- Third-party services
- Parent or sibling organizations
- Any other malicious or unauthorized user

We also assume that the SOC analysts that are honest but curious in the sense that they might be motivated to look into sensitive data about other people which could lead to cause them harm (i.e., impersonation, blackmailing, disclosure of private information, manipulation of their opinion). This also applies to users with insufficient privileges from the customers' site.

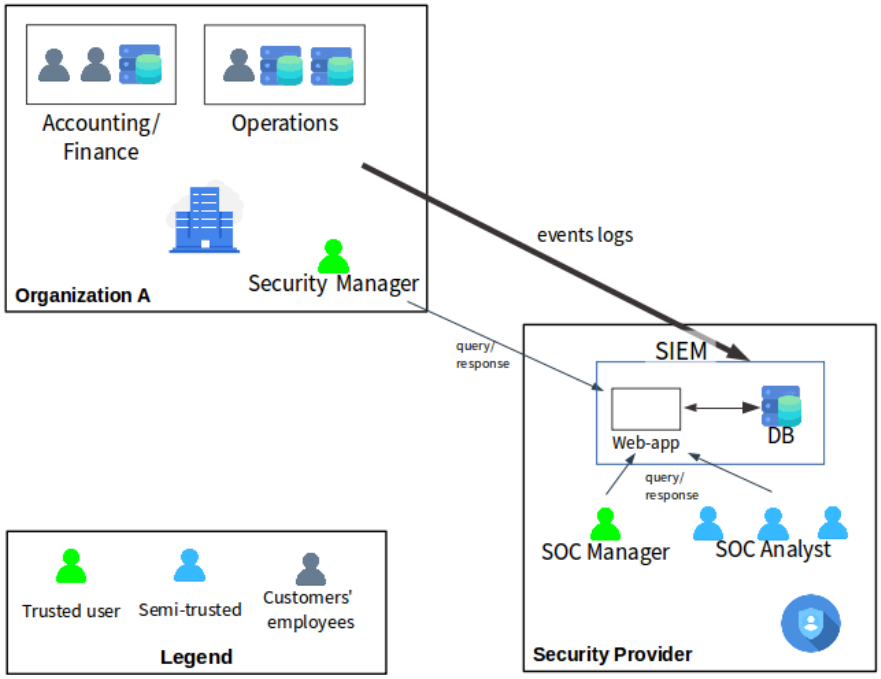

Figure 4.1: General overview of our threat model

In our threat model, we have two types of users.

- **Trusted user** are users with a role such as a SOC manager and are allowed to read any sensitive data in the clear-text from their console.
- **Semi-trusted user** with a role such as SOC analysts is not allowed to read sensitive data in the clear-text; they can only see anonymized or pseudonymized data.

4.3 Presenting architectures

4.3.1 Architecture 1 - Classical architecture (without anonymization)

As shown in Figure 4.2, in this architecture, raw logs are collected from the customer's site, and are forwarded by the sensor to the cloud providers infrastructure without any anonymization. The information in the logs is mapped into each field and are persisted. A web application collects the logs from DB to be displayed to the analysts.

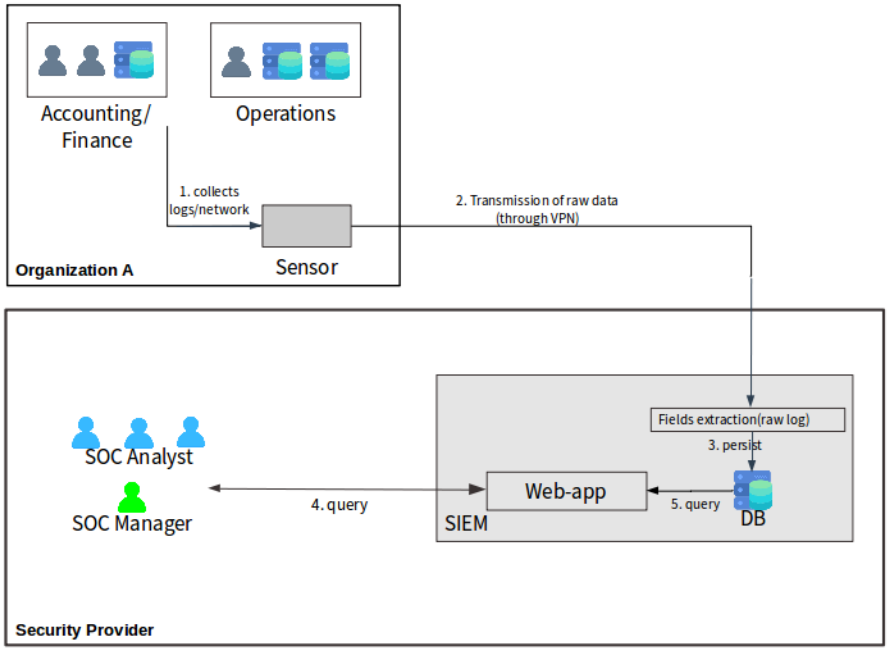

Figure 4.2: Classic architecture - private keys managed by MSSP

4.3.2 Architecture 2 - Anonymization on SIEM

As shown in Figure 4.3, in this architecture, raw logs are collected from the customer's site and are forwarded by the sensor to the cloud provider's infrastructure. After the raw logs are collected in the security providers infrastructure, each information is mapped into each field. Extracted PII fields can be identified to which anonymization functions are applied. After the PII fields have been anonymized, they are persisted. A web application collects the logs from DB to be displayed to the analysts.

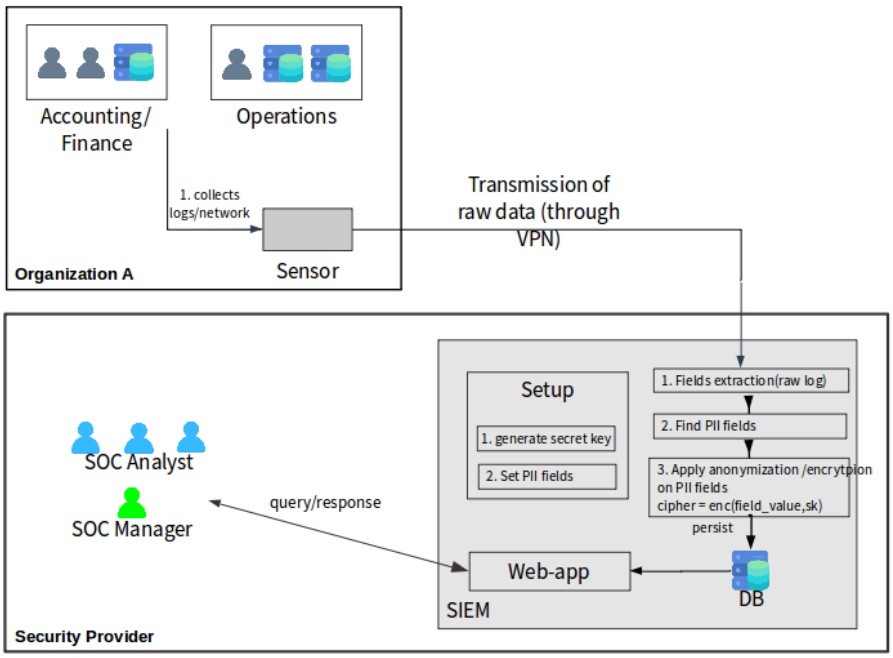

Figure 4.3: Classic architecture - private keys managed by MSSP

In this architecture, secret keys are stored and managed by the security provider's infrastructure, and as anonymization functions are applied on each field and are persisted in the database, they also need to be decrypted for the users who need to be able to see that data in the clear text format. Security Provider holds the secret key and customer's do not have a guarantee that collected monitoring data which consists of sensitive PII has not been accessed in clear.

An attacker that has breached in the Security Provider's system and is listening passively on the customer's infrastructure can also see data in clear before it is being encrypted or anonymized using a secret key.

4.3.3 Architecture 3 - Private key managed on the sensor

As shown in Figure 4.4, in some instances, customers might like data never to leave their network. When data collected by the sensor is persisted on the sensor's database, we call those "on-site" architecture because collected data is never persisted outside of the customer's network.

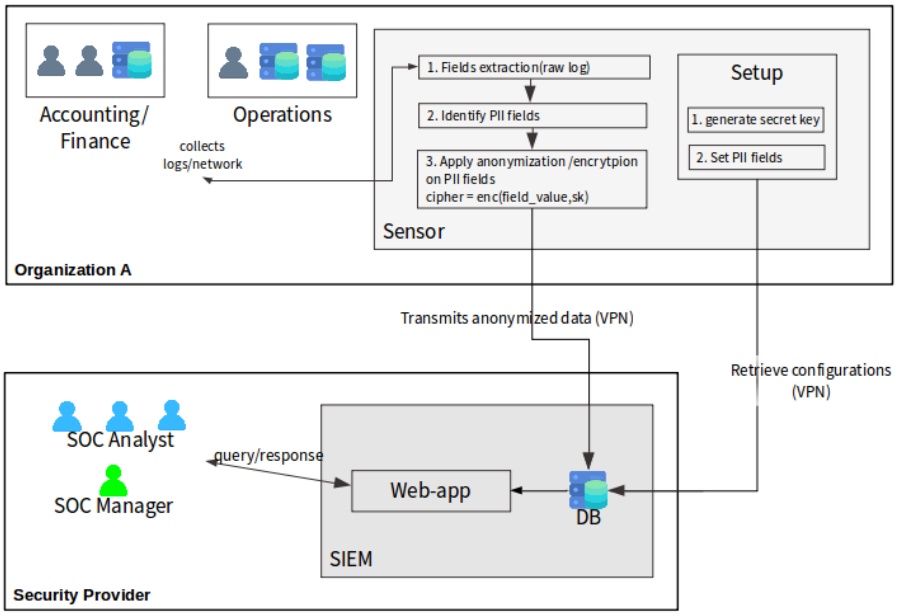

Figure 4.4: Architecture 3 - private keys managed on the sensor (MSSP)

4.3.4 Architecture 4 - Private keys managed by the customer

In this architecture, more computational effort is shifted to the customer. As shown in Figure 4.5, the software needs to run on the customer's site and the client must have full control of the software code, which receives event logs and extracts the fields. PII fields are then identified, and anonymized or encrypted using secret-key only known by the customer. Sensitive PII fields values are then encrypted on the client's site and are forwarded to the sensor, which redirects them to the security provider's infrastructure.

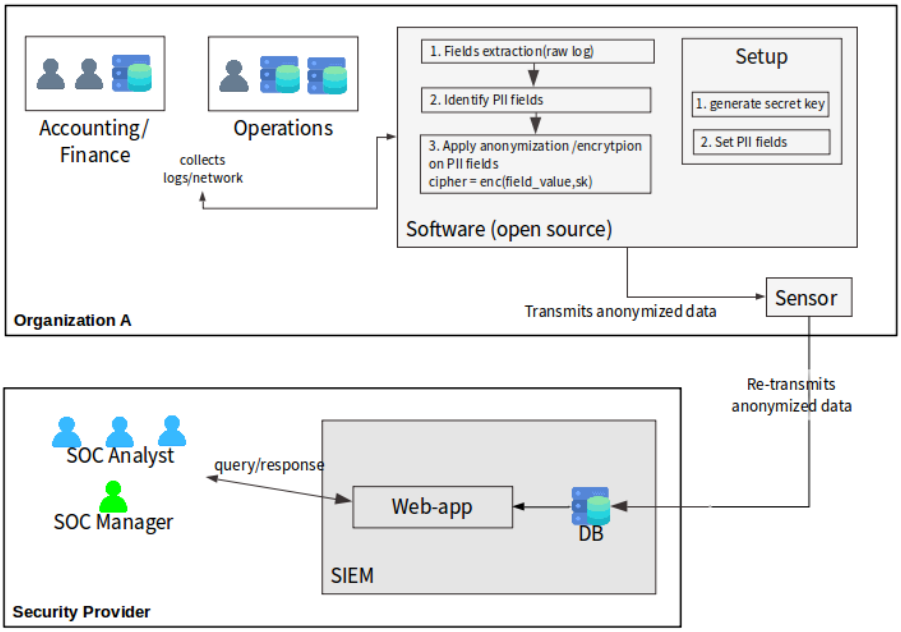

Figure 4.5: Architecture 4 - private keys managed by the customer

Secret keys that are used to encrypt sensitive PII are stored in the DB and are encrypted so that even the Security Provider does not have access to them. The secret key is then retrieved from SIEM's database and is decrypted to be used by the SOC Manager or specific SOC analysts who need to be able to decrypt sensitive PII data in order to access them in a clear text format.

Chapter 5

Modelling use cases

5.1 Real-world use case study of network security monitoring

Our research is based on the user case scenarios of a real-world Network Security Monitoring (NSM) provider. To understand and model such scenarios, we have obtained first-hand information and experiences from the analysts of the NSM provider through extensive meetings and discussions. We then analyze the obtained information to categorize them info different use case scenarios. Finally, we devise our models to more precisely depict such scenarios to facilitate further studies.

We present and describe available actions by the analysts from their platform/console. We then detail the investigation types by describing each of them with five use cases as follows:

- Use case 1: Data breach through malicious code execution (MC)
- Use case 2: External vulnerability scan (EVS) / Reconnaissance
- Use case 3: Insider attack
- Use case 4: Distributed Denial-of-Service (DDoS)
- Use case 5: Policy violations

In each use case, based on the analysis of the raised alerts, the analyst will be analyzing the alerts and looking for true positive which could lead to a potential threat undermining client's security.

5.2 The use case scenarios

In modeling the use case scenarios, our goal is to understand and determine if the analysts will be able to perform their analysis work based on pseudonymized or anonymized PII fields as if they were visible in cleartext and if not, we must understand why. We are interested in two types of privacy protection: pseudonymization and anonymization, as described in tables 5.1 and 5.2, and Figure 5.1.

Action ID	Description
A	Analyst visualizes General Overview with live alerts (alerts to be analyzed) for a client. Alerts are grouped/aggregated ordered by severity, time, rule name, src/dst IP or Port with other fields. They can be represented in different forms (listed, grouped, in charts or geographical maps)
B	Analyst selects grouped alerts or single alert and clicks on Acknowledge classifying the alert as false positive (FP).
C	Analyst clicks on Show more details for the selected grouped alerts to display detailed View of the alerts payload, raw logs including all fields available fields (e.g., timestamps, protocol, URL, username,) depending on the alert or event type.
D	Analyst selects a single alert, set of alerts or grouped alerts that he/she classifies as a true-positive attack (or even a potential threat) and clicks on Create New Incident or Add to existing Incident if an Incident already exists for the same investigation. This will inform the client of the situation.
E	An analyst can see alerts payload in their preferred packet analyzer client (e.g., Wireshark) in order to understand more deeply the communication that has caused the alert to raise.
F	Analyst requests more information from a client. For example, if the analyst does not have enough facts to confirm if the attack is a true-positive, he can ask a client for more information.
G	Search the web for more information about threat intelligence feeds [2] for specific IP address, domain, or even online analysis of unknown files. These features are today generally integrated part of the SIEM or NSM platform which enriches more useful information to the analyst.
H	The analyst performs an advanced custom search on archived alerts with specific conditions such as source type (e.g., NIDS/HIDS, firewall log, application logs, router logs), fields/attributes conditions (e.g: IP, Port, Protocol, Payload/Content, Message) and/or regex keyword search.

I	Requests to display all related information to the network device such as last activity (last time it was exchanging network data), open ports, services, OS, last scans and discovered vulnerabilities.

Table 5.1: List of action IDs with their descriptions

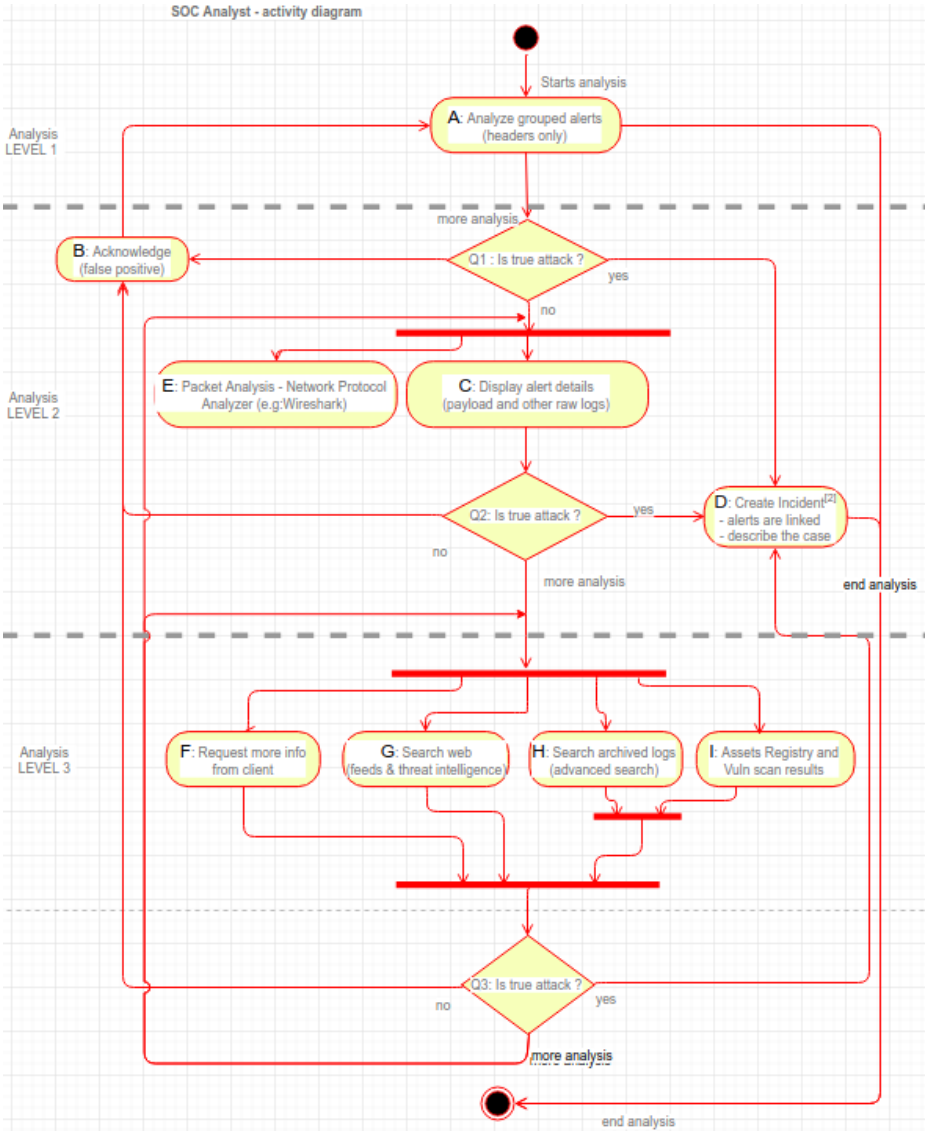

Figure 5.1: Activity diagram of the SOC analysts

Investigation Abbreviation Type Name	Investigation Investigation Type	Description
BFA	Brute Force Attack	Brute-force-attack consists of trying as many possible password combinations until a match is found, generally by trying all combinations. It is also considered as the least efficient way to crack a password.
DDoS	Distributed Denial of Service	DDoS is a type of attack in which the attacker overwhelms victims targeted system(s) resources (e.g., bandwidth, memory, CPU) resulting in downtime or degradation of its services.
EVS	External Vulnerability Scan	The attacker is scanning all "open ports" from its target in order to identify exploitable vulnerability so that get can gain access and control.
MC	Malicious Code	Attackers' malicious program (e.g., worm, malware, trojan-horse) that is successfully installed on a victims computer allowing the attacker to have control of it.
AA	Abnormal activities	Four categories of Snort alerts [25]; miscellaneous activity, miscellaneous attack, access to vulnerable web applications, and detection of TCP connection in the clients network are grouped together forming this category.
PV	Policy Violation	Alerts that are raised because adult contents were accessed or because peer-to-peer (P2P), social network websites were accessed.
OTH	Others	Alerts that are not part of any listed category (uncategorized).

Table 5.2: Investigation types with descriptions

In the next section, we detail five use cases with their step sequence identifier, action ID, which corresponds to the table 5.1 and involved fields. Involved fields will only be mentioned the first time and will not re-appear if they were mentioned in one previous section to reduce field repetitions.

For our use case scenarios, consider the following network ranges, where no anonymization is applied:

- Internal network or monitored IP range is: **1.0.0.0/8**
- Attacker IP's host range is: **9.9.9.0/24**
- Any other network range is considered as external network range, which is not monitored.

5.2.1 Use case 1 - Data breach through malicious code execution (MC)

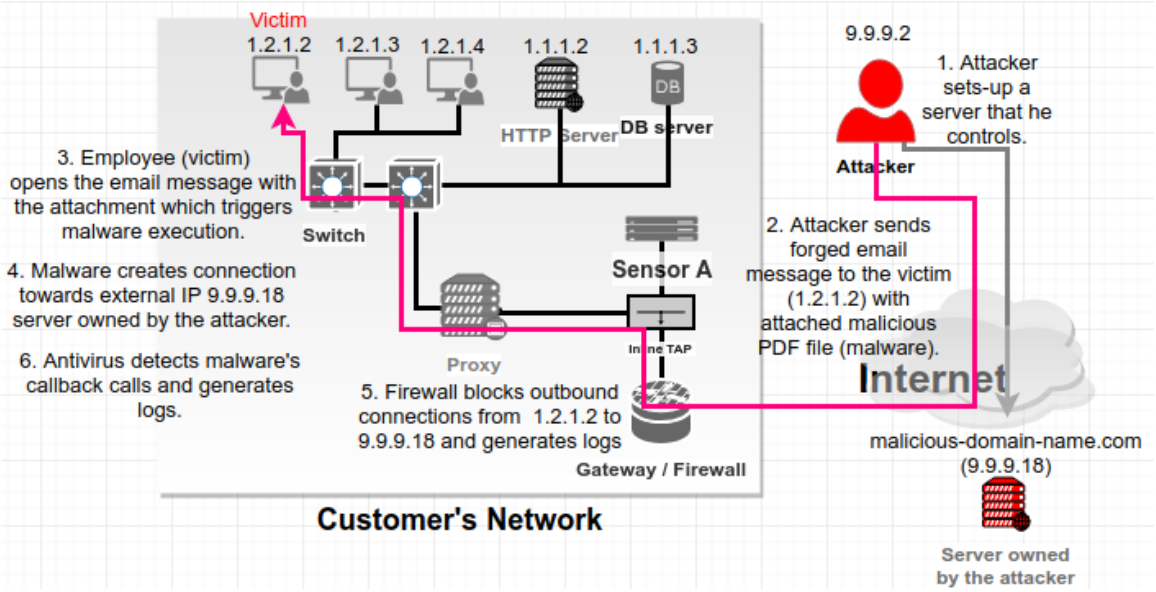

Figure 5.2: Data exfiltration by sending to the victim a forged email message with attached malicious PDF file (malware).

Step Seq	Action Id	Description	Involved Fields
1	A	Alerts with high severity are raised from source IP 1.2.1.2 (Internal IP) and remote black-listed domain name (malicious-domain-name.com) resolving to IP 9.9.9.18 (attacker's server).	domain-name, src/dst IP, Timestamp, protocol, service HTTP, URL, severity
2	C	Analyst inspects details/payloads of the alerts and confirms multiple HTTP POSTs on a blacklisted domain name in an attempt to transmit data from a local victim's computer.	file location, file size,
3	G	Analyst searches on the Internet feed (Threat Intelligence) to find more information relating the raised blacklisted domain name and IP and finds that there is a new malware that is spreading on the Internet linked with the attacker's IP and domain name.	antivirus name: version, last update (timestamp), malware family name, URL

4	H	The Analyst performs searching on all archived alerts (up now) and events with condition generated by the host 1.2.1.2. Some of the matching results include events indicating that the antivirus services have stopped running on the victim's local computer.	hostname, process name,
5	C	Analyst inspects alerts details related to Anti-Virus Application log data source shows: (1) scanned email attachments, (2) victim's antivirus services shuts-down less than 3 seconds after.	email address, process id, status id
6	D	Analyst selects related alerts and events and marks them to a new Incident. He specifies Incident classification type as Malicious Code and describes the case urging the client to shut-down the possibly infected computer and requesting investigation to confirm if it has effectively been infected and possibility of data exfiltration by the attacker.	Incident description, Incident classification, involved alerts Ids with incident, incident intervention, incident severity

Table 5.3: Use case 1 - Data breach through malicious code execution (MC)

Analyst 3 - based on the use case 1 (data breach through malicious code execution)

The analyst starts his/her analysis by exploring the alert raised by the Firewall - blocked outbound connection to external host - (**9.9.9.18**) originating from internal network **1.2.1.2**. Involved fields are: IP address, hostname, username. Analyst needs to perform a forensic search based on involved IP addressees (**9.9.9.18**,**1.2.1.2**) and domain-name (malicious-domain-name.com) to verify if any other devices have been compromised by trying to reach outside the network.

Conclusion on analysts' search/forensic fields

Our conclusion is if pseudonymization is performed deterministically, then searching could work as usual, but if non-deterministic pseudonymization function is used, then searching would not work because no matches could be found in the database.

5.2.2 Use case 2 - External Vulnerability Scan - Heartbleed

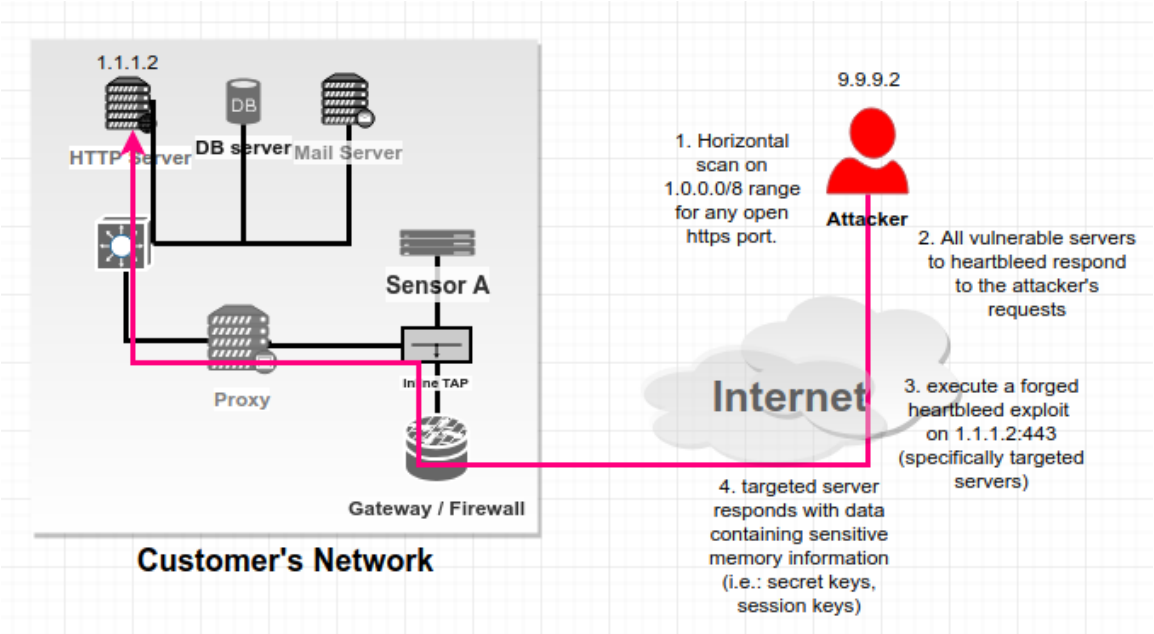

Figure 5.3: Attacker performing horizontal vulnerability scans followed by execution of heartbleed vulnerability

Step Seq	Action Id	Description	Involved Fields
1	A	Analyst views grouped alerts generated by Snort IDS "SERVER-OTHER TLSv1.1 large heartbeat response - possible ssl heartbleed attempt"A.14 targeting host with IP 1.1.1.2 (victim) originating from 9.9.9.2 (attacker IP).	snort Rule Name snort Rule Id (sid, rev) src/dst IP URL query parameters (key-value),
2	C	Analyst validates the payload (data) with rule's signature and confirms that it is a true positive.	HTTP status code, HTTP service
3	I	Looking at the targeted victim's server logs, analyst confirms vulnerability exploitation by the attacker, but needs more investigation on what information was extracted exactly from the vulnerable computer.	services/port, vulnerability Plugin name, Severity, CVE-ID, CVSS score.

4	H	The Analyst performs a search on all stored logs (from all data sources) with source IP 9.9.9.2 and destination IP 1.1.1.2 and confirms multiple occurrences of the attack.	username (victim)
5	D	Analyst creates an Incident classifying the case as External Vulnerability Scan and Sensitive data exfiltration including all related alerts by commenting possibility of exploitation[2].	

Table 5.4: Use case 2 - EVC (External Vulnerability Scan/Reconnaissance and exploitation) - Heartbleed

Further investigations (searching relating IP communications, amount of data exchanged between) are required if any other devices has been compromised and if the any sensitive information has been exfiltrated. Sample of a heartbleed snort rule:

Analyst 1 - based on the use case 2 (external vulnerability scan and exploitation)

The analyst focues on three sensitive fields: source IP and destination IP address, and payload. According to the SOC analyst, to validate if this attack is a true positive, he must look into the payload of the raised alert. If the payload is readable in cleartext and can confirm that the private keys or session keys have been exfiltrated from servers, he can then confirm that the attack is a true positive. In this context, the source IP (i.e., attacker's IP) and the destination (i.e., victim's IP which needs to be monitored) can both be anonymized (non-deterministically) or pseudonymized (deterministically).

Conlusion on non-deterministic anlaysis fields

For this use case, the analysts will need to rely on IP traces anonymizers because he will need to inspect the payload from PCAP files. If the IP address is anonymized non-deterministically, then the analyst will not be able to know which device has been targeted by the attacker because multiple alerts will always result in different IP addresses even if the same device is being been targeted. It will imply that the analyst will not be able to determine how many devices were affected.

5.2.3 Use case 3 - Insider attack

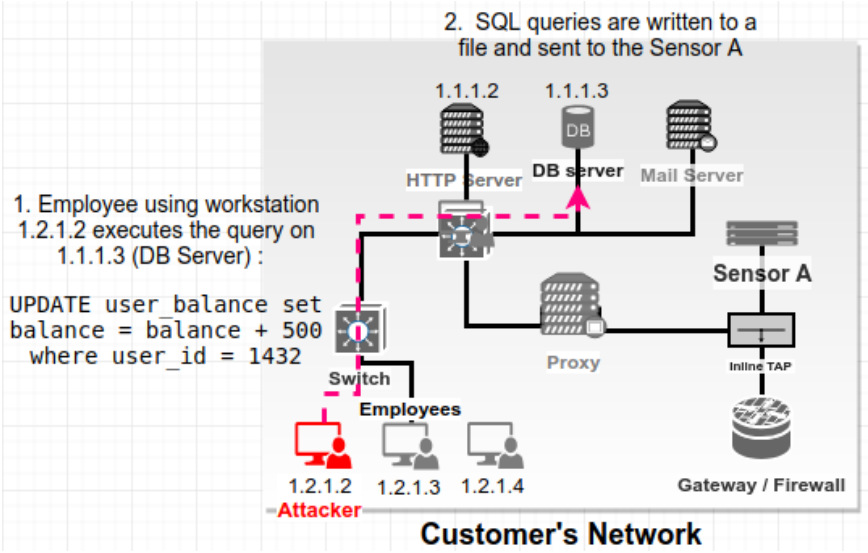

Figure 5.4: Insider attack - employee altering customer bank balance data without authorization

Step Seq	Action Id	Description	Involved Fields
1	A	Analyst analyze the alert with low severity originating from a database server (SQL) server (1.1.1.3) data source. Alert is raised for policy violations involving SQL query (Insert/Update/Delete) on behalf of a username 'bob'.	username, SQL query
2	C	Analyst displays grouped event payloads (raw logs, SQL queries) and can see suspicious SQL Query: "UPDATE user_balance set balance = balance + 500 where user_id = 1432".	SQL (user, roles, schema, database name, table name), Account Id
3	H	Analyst performs an advanced search on keywords containing 'UPDATE' and .	log source, site/domain
4	D	Analyst escalates the case informing the client of unauthorized data alteration.	

Table 5.5: Use case 3 - Insider attack

Analyst 2 - based on the use case 3 (insider attack)

The analyst makes the point that it is critical to understand which networks needs to be protected and which networks should be considered as non-trusted or external. This information is useful to determine if the attacker has already breached inside customer's internal networks.

Conclusion on prefix-preserving

If the anonymization is used in which prefix of the IP addresses is not preserved (i.e., the most significant bits-group of the IP addresses is lost), it would then be impossible to determine which device should be considered as trusted, and that would need to be protected. If it is prefix-preserving but non-deterministic (i.e., where the least siginificant bit changes all time), then the analyst would not be able to know how many devices are affected by the attacks because non-deterministic functions will always generate different IP addresses limiting searching capabilities during their investigation.

5.2.4 Use case 4 - Distributed denial of service attack

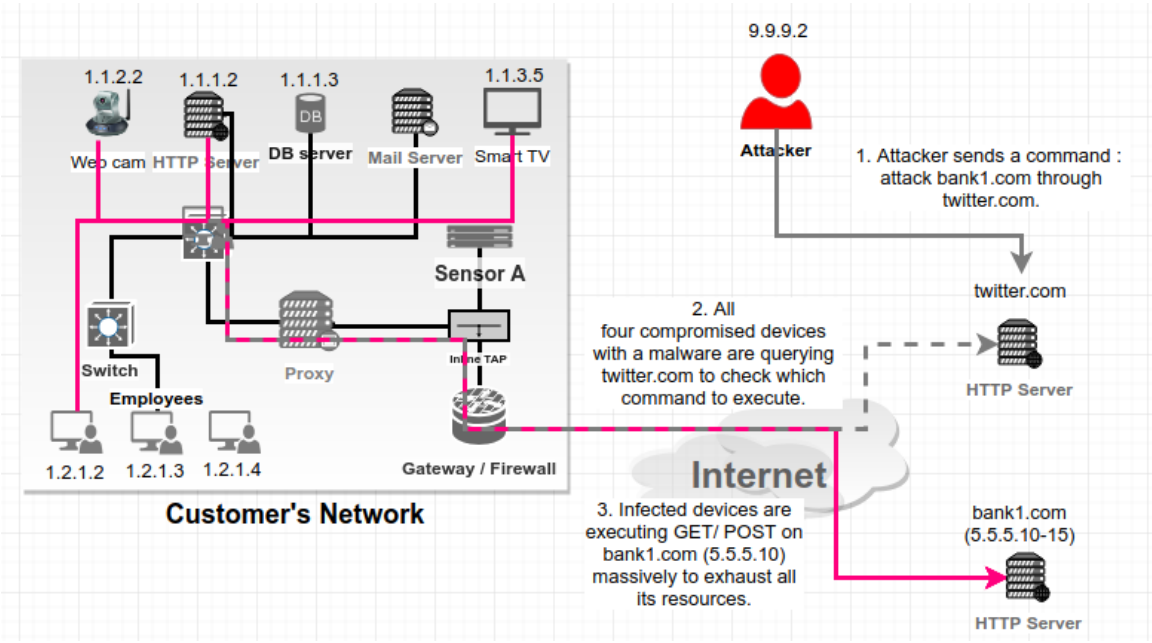

Figure 5.5: DDoS attack on a bank1.com website.

Step Seq	Action Id	Description	Involved Fields
1	A	The analyst is receiving alerts indicating unusual network usage (specifically amount of data transmitted).	Higher amount of data transmitting (kb/s)
2	C	In data payload, the analyst reads thousands of HTTP requests originating from monitored client's network (1.1.2.2, 1.1.1.2, 1.1.3.5, 1.2.1.2) targeting bank1.com (5.5.5.10). Some of the log records shows HTTP GET requests towards twitter.com's website. Analyst inspects full path's URL of twitter.com website containing a tweet message: "attack bank1.com" where Tweeter is being used as intermediate to communicate with its botnet (client's infected hosts) to attack bank1.com	GPS location - city- (source IPs - destination IP), traceroute/ping response, URL, HTML
3	E	Alerts indicating massive HTTP requests to the targeted bank1.com domain name (which resolves to IPs 5.5.5.10-15) using monitored client's network (1.1.0.0/16 and 1.2.1.0/24) resources and unreachability are being attached to a new Incident. Its classification is DDoS attack with explanations on how many requests and strength of the attack (requests per second, how many source IP are involved). The analyst is also trying to understand and communicate with the client if they have received any threats[3] from the attacker.	Firewall rules, Network routing, Netmask, Bandwidth, Throughput, Packet status, response time

Table 5.6: Use case 4 - Distributed denial of service attack

5.2.5 Use case 5 - Data breach through MC (data exfiltration)

Step Seq	Action Id	Description	Involved Fields
1	A	Alerts with an unusual amount of data transmitted from 1.1.1.2 (Internal IP - running file shared service) to a remote server 9.9.9.15 over FTP.	FTP data transmitted
2	C	Analyst analyzes the quantity of data that has been transmitted from 1.1.1.2 to 9.9.9.15 and find multiple hundreds of Megabytes (MB).	
3	H	Analyst searches network logs having IP 1.1.1.2 as SRC or DST and can see that machine with IP 1.1.1.2 has received a great amount of data from 1.1.1.15, 1.1.1.16 and 1.1.1.17 (all POS databases categorized as high criticality assets) over a file shared service in the last hours.	
4	H	Analyst investigates system logs from POS database server with IP (1.1.1.15) and finds abnormal processes running such as “POSWDS” (malware named as a normal process). The analyst also finds that a user with an unusual username has been connecting recently having admin privileges.	process name, last login time
5	H	Analyst investigates system logs from POS database server with IP (1.1.1.15) and finds that a user with an unusual username has been connecting recently having admin privileges.	username
6	D	Analyst selects related alerts and events and marks them to a new Incident. He requests further investigations to confirm the attacks because all symptoms seem to indicate a breach.	Incident title and description, Incident classification, involved alerts Ids with incident

Table 5.7: Use case 5 - Data breach through malicious code execution (MC) and Anomaly Network behavior (ANB)

Chapter 6

Applying anonymization techniques

6.1 Fields and anonymization techniques

Name of the field	Description	Encryption/anonymization techniques
Full name	Individual's last name and first name	AES-SIV encryption
Home address	Street address, City, Province/State, Postal code or zip-code	AES Synthetic Initialization Vector (SIV) encryption
Customer's name	Name of the customer or organization which is paying for its cyber-security monitoring services.	AES-SIV
SSN, CCN, VIN, Passport number	Social Security Number (SSN), Credit Card Number (CCN) or Vehicle Identification Number (VIN)	Format-Preserving Encryption (FPE), AES-SIV
Biometrics	Physical metrics identifying uniquely an individual E.g., fingerprint, face, iris, retina recognition, hand-geometry, voice and more	AES-SIV encryption
Date of Birth	Time and date of birth	Format-Preserving Encryption or by generalizing
Place of birth	City or/and country of birth	Format-Preserving Encryption

Phone Number	Consists of a country code, three-digit area code, three-digit central office and four-digit station code	Format-Preserving Encryption (FPE) or AES-SIV encryption
GPS location ($< 100meters$)	Global Positioning System (GPS) is a satellite navigation system used to determine an object's position on the ground Higher precision reduces privacy.	GeoHash or by generalizing (reducing precision)
GPS location ($> 100meters$)	Lower precision increases privacy	GeoHash or by generalizing (reducing precision)
IP address	A numerical label assigned to a device when it connects to a computer network. It consists of source and destination IP addresses	CryptoPAn, cryptopANT or Order Preserving Encryption (OPE)
Hostname (Device name)	Label (name) assigned to a device on a network	FPE or AES-SIV
E-mail address	Consists of two parts, first part, before @, the name of the mailbox which is generally the username and the second part is the domain name.	FPE or AES-SIV
MAC Address	Physical address of the network interface	FPE
Username, Account/User id	Identification used by a person to access computer/network	AES-SIV
Operating System	Device fingerprint can be used to identifying uniquely an device.	No anonymization or AES-SIV if needed
Browser signature	Browser fingerprint's can be used to identify uniquely a specific browser across different websites[1]. (i.e., User-agent, version, language, plugins, platform)	AES-SIV if needed
HTTP Cookie	HTTP cookie is a string generated by web server to user's browser to remember its browsing history or state.	AES-SIV
Timestamp (date and time)	Time and date when an event has occurred.	Time (backward or foreward) shifting or by generalizing (reducing precision)

Device Id (i.e., Serial Number, UUID)	Device identifier which can be unique.	FPE
Open Service/Port (http/80,443)	Running services or daemons on local machine must open a specific port to accept packets to provide any type of services $(i.e., FTP, ssh, http)$.	No anonymization or Order Preserving Encryption (OPE) if needed
Vulnerability name	Short description of a vulnerability.[2]	No anonymization
Correlation alert or rule name	Short name of the alert	No anonymization
Vulnerability CVE-ID	Common Vulnerabilities and Exposures (CVE) used to identify a specific vulnerability.	No anonymization
URL	Uniform Resource Locator is a web address references the location of a remote server's resource.	AES-SIV
Network activity rate	A network data transfer rate (rate kbps or pkt/s).	No anonymization

Table 6.1: Fields and encryption or anonymization techniques

6.1.1 The choices of encryption or randomization methods

The anonymization techniques we apply per field can be explained as follows. From our system perspective (e.g., database, applications) we believe that there are fields that must conform to specific formats or standards in order for them to support the analysis. For example, IP addresses, dates and times, various ID numbers (e.g,: passports, driving license, credit-card numbers, serial numbers), phone numbers and other group of fields, require to comply to specific formats and are less restrictive such as first name or last name and other names.

- IP address: There exists many anonymization functions such as CryptoPan[33] or cryptopANT[31] which are also prefix-preserving and is why they were chosen.

- MAC address: cryptopANT offers anonymization of MAC addresses. FPE could also be

[1] https://amiunique.org/

[2] https://cve.mitre.org/cve/list_rules_and_guidance/cve_assignment_information_format.html

drawn on this field by specifying policies.

- Hostname: As mentioned in RFC-1178[3], page two: "While computers aren't quite analogous to people, their names are.". We believe that AES encryption could be applied or FPE[61] as long as the output generates alpha-numeric value.

- GPS locations: The algorithm GeoHASH[34] generates approximative GPS location deterministically by generalizing GPS location.

- E-Mail address: A valid e-mail address must consists of two parts, the first part (i.e., before) is generally a username and the second part (i.e., after) is the domain name. AES encryption can be applied on siem platforms that supports values which do not contain '@', otherwise we propose FPE. More detailed specification is described under the RFC-5322[4].

- Social-security number (SSN), Credit-card number (CCN), Vehicle identification number, Passport number: We propose FPE because these fields will generate outputs with same format of the provided input identifier.

- First name, last name, customer name or username: We suggest AES encryption for these fields because software systems in general do not require any specific format.

[3]D. Libes, "Choosing a Name for Your Computer", Request for Comments RFC 1178, Internet Engineering Task Force, Aug. 1990.

[4]P. Resnick, Ed. "Internet Message Format", Request for Comments RFC 5322, Internet Engineering Task Force, Oct. 2008.

6.1.2 Quasi-identifier-based attacks and defence

Indirect-identifiers are also known as quasi-identifiers. Quasi-identifiers do not identify any individual by themselves, but when they are combined together can lead to singling out of the records which can point to a person's identity. Many attacks are going through quasi-identifiers that are not carefully protected as we have mentioned in the section data breaches 2.2.2.

Our assumption on the IP address fields is described as follows. The IP protocol header consists of source IP address and destination IP address. Our assumption is that if the source IP address of the packet is part of the internal network which needs to be monitored and protected by the SOC analyst and destination IP address is external (not monitored), then source IP address is direct PII that could be linked to an individual whereas destination IP address will be considered as indirect or quasi-identifier PII, similar to other fields which when combined, can lead to signling-out an targeted individual.

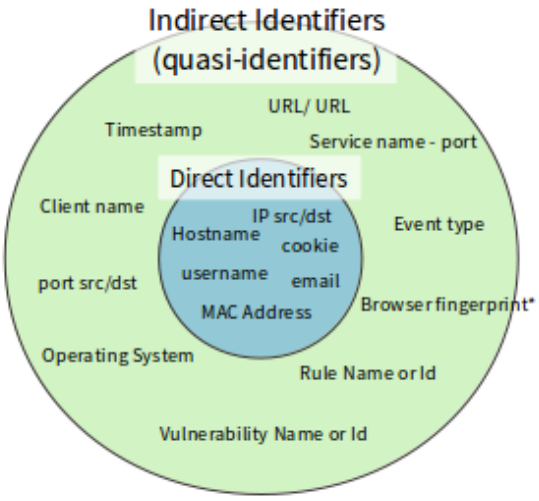

Figure 6.1: Grouped quasi-identifiers and direct identifiers among all identifiers fields from collected event logs

The relation between quasi-identifiers (indirect identifiers) and meta-data is almost alike. For example, in TCP/IP communication protocol, even if network traffic payload is encrypted, the packet headers might consist of meta-data which might tell a lot about a person that is browsing the web and which might consists of very useful information to the adversaries. In [22, 23], the authors

demonstrate how meta-data can be used to extract useful information about individual's. Network-flows consists of the following fields: timestamps, source and destination IP addresses and ports, amount of packets and data exchanged is considered as metadata. A similar analogy of network flow communications can be applied with recorded GPS locations or phone call history and call duration.

Examples of active attacks

The analyst could deduce the original values from anonymized IPs, URLs based on time or chronology. The analyst could infer related information about a device or user based on rule names, vulnerabilities, services (ports), leading to data inference attacks.

Proposed solutions to the active attacks

- **Apply timestamps shifting (forward or backward)** randomly during events ingestion to reduce the likelihood of time-based attacks by the adversary. It must be consistent across the platform in order to keep the order of the events chronologically. This technique has been mentioned in [46] where they propose shifting of the event timestamps. As they describe, the relative time of the events has greater important than knowing the exact date and time of when the events have occurred.

  Challenges: If there is a data breach, the sequence of the recorded events must be preserved. For example, if the adversary typically follows the pattern "cyber kill chain" methodology[41] and the sequences of events must be preserved.

- **Dropping unique events** that are returned after the query. We propose a method that drops uniqe events with probability of 50 precent to reduce odds of an adversary that tries to single out events. It could apply only for events that have never been linked to potential attacks or have very high certainty of false positive.

- **Create similar events** that are returned after query. We propose creation of similar events that can be duplicated with few random anonymized IP addresses in such a way that the impact is low (e.g,: low severity alerts), and the analyst cannot deduce which one has been truly

generated.

- **Add random noise** using middleware devices (servers), IoT and other devices which do not receive direct inputs from humans as decorators or shadows for targeted devices used by the employees to browse the Internet, write emails or communicate with someone. Queries that return singled out records with single anonymized direct identifiers are sensitive and can be targeted.

6.2 Benchmarks of anonymization techniques

In this section, we perform four types of benchmarking. The first type is on the throughput, which consists of measuring the number of calls that can be made per second. It is based on the Java library, Java Microbenchmark Harness (JMH)[79]. In this experiment, no field is anonymized. The hashing functions "MD5" and "SHA-256" were benchmarked only as a reference point for our anonymization methods.

The second type is on the execution time, where we measure the execution runtime (in ms) with datasets closer to real-world scenarios. The following six columns or fields were anonymized per record: source IP address, destination IP address, hostname, username, raw_log and email address. Each field value was randomized to reduce optimizations by caching. It consisted of five test cases with sample sizes from 100 records of up to 1 million records.

In the third and fourth types, we collected measures based on the memory usage and CPU usage, respectively. In each case, we took measures on memory and CPU usage every 30 seconds interval when 1 million records are anonymized. For each experiment, two series of results can be found, one with and one without application of anonymization functions so that they can be compared.

In the second, third and fourth methods, the following anonymization methods were used: AES encryption CTR mode, cryptoPAnt, CryptoPAn and FPE, which are based on the use cases we have discussed in section 6.1.

The hardware we used for benchmarks:

- CPU: Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz
- RAM Memory: 16 GB DDR3 - 1600MHz
- Hard-disk: Model ST500DM002-1BD142, 500GB SSD SATA 6Gb/s

The software we used:

- OS: 16.04.5 LTS (64 bits)
- Java: 1.8.0_201 (64 bits)
- Apache Spark version: 2.4.1

6.2.1 Throughput (Op/s) comparison

We used the library Java Microbenchmark Harness (JMH)[79] to perform our benchmarking on all of the anonymization functions. The anonymization function that we have used are all written in Java language, except cryptopANT, which is in C language and for which we wrote our custom JNI to interact. We used JMH's throughput mode, which measures how many times per second the function or the method can be executed (i.e., Op/s). Our goal in this experiment is to compare and rank them in terms of calls per second. Our benchmarks consisted of 100 iterations per second, preceded with ten iterations of warmup that were not considered in the metrics.

In the table 6.2 below, we present averaged results based on 100 iterations. They are sorted from most efficient on the top to the least on the bottom. The two first ranked hash functions were benchmarked for the reference point. The column "Decryption Ops/s" is empty for some masking methods are only one-way functions and decryption is not available.

"MD5" is outperforming all other functions with over 3.9M operations per second on average, followed by "SHA-256" with 2.2M Op/s, which are expected to perform better than the other encryption-based schemes. Surprisingly, the cryotopANT outperformed CryptoPAn and AES encryption schemes with our implementation of Java Native Interface (JNI), which enables calls from

Name	Encryption Ops/s	Decryption Ops/s
MD5	3942565	Not applicable
SHA-256	2234214	Not applicable
cryptopANT	1096619	94065
CryptoPAn	586451	Not applicable
AES-CTR	261082	249930
FPE	18572	18688

Table 6.2: Averages of anonymization benchmarks overhead in terms of operation per seconds

programs written in Java language to be made to cryptopANT program, written in C language. FPE could only do ten thousand operations per second, which is due to the poor performance in the function "com.idealista.fpe.component.functions. prf.DefaultPseudoRandomFunction.apply()" based on "JVisualVM" CPU profiler analysis. When we executed the encryption of FPE in debugging mode, we noticed that over 99% of the computation time was spent in the function "makeSessionKey" of com.sun.crypto.provider.AESCrypt, a low-level class for AES Symmetric encryption. The com.idealista.fpe.algorithm.ff1.FFA1Algorithm class, which performs the encryption, is currently configured to perform ten rounds in a for-loop, implying ten calls to AES encryption to encrypt only one single message. No anonymization has been yet applied to any of the fields. In all of the anonymization functions, we anonymized a string "helloworld" except for cryptoPAn and cryptopANT. For this benchmarks, we used a fixed IP address "10.2.1.23" as the value to anonymize.

6.2.2 Execution runtime

We present two types of benchmarks. The first benchmark is performed in a simple Java program without relying on any other framework or technology to reduce any extra overhead. In the second type of benchmarks, we use Apache Spark, a large scale data processing engine, which is what we have used in our SIEM platform.

In our experiment, we evaluate the performance of two encryption algorithms, FPE (FF1)[60] and AES symmetric encryption (see Appendix), which we have used on the fields: username, hostname and email. Our goal is to compare and evaluate the scalability of the two algorithms using input strings of similar lengths of those fields. The benchmarks consist of 100 rounds where each round has 10000 iterations of each encryption algorithms, preceded with warmups of 20 rounds to obtain

more consistent execution time measures. The execution time is measured in milliseconds, after every 10000 iterations of every round, and averaged over 100 rounds. After every 100 rounds, the input size length(X-Axis) is increased with ten random alphanumeric characters in lowercase.

The Figure 6.2 depicts the benchmark results of FPE and AES anonymization functions.

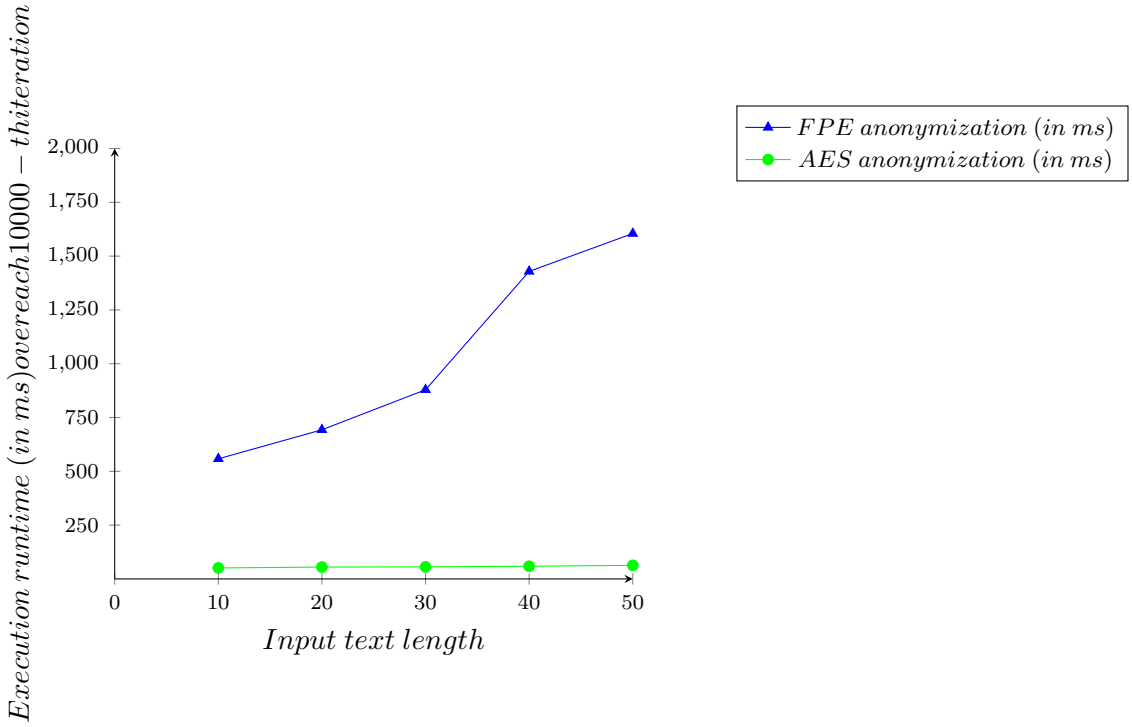

Figure 6.2: Averaged execution runtime (in ms) with and without anonymization

As we can see, the FPE algorithm's function performance results (Y-Axis) are closer to exponential increase rather than linear even with increments of input messages of ten random alphanumeric characters (X-axis). Therefore, we do not consider the FPE algorithm as scalable, but that's no surprise because it is not an implementation that originates from a trusted source. As we see, AES is constant in time and performs in constant time with input message strings that are not too large.

Our second type of benchmarking is performed on top of Apache Spark. We choose Apache Spark because this framework is very popular in the software industry, which has to deal with big data problems such as monitoring the cybersecurity log. Apache Spark has two types of operations[5]: transformation and action. New transformations (i.e., map, filter, join, groupByKey) are created from an existing one and are lazily evaluated for better performances, whereas actions (i.e., reduce, count, foreach) will trigger the launch of a Spark job. In our case, we trigger our action by calling a "foreach()" function, which will evaluate each record and apply pseudonymization functions on the dataset.

In Spark, statistical measures can be retrieved after each stage has completed. In our case, we have "executor run time" and "CPU Time" when anonymization functions are invoked and are collected by overriding the method "onStageCompleted()". Tests were performed over 150 iterations each, and a mean average was calculated. Five first iterations were excluded for warmup. The events per second (EPS) were calculated based on the sample size and the executor time (in ms) from which we solved the proportions. For example, if we know that it takes 11ms to process 100 records, we then deduce using proportional relationship that 12836 events are processed in one second. Our benchmark results are shown in the table 6.3 below.

Case	Sample (# log entries)	exec. time w/o anon.	exec. time w/ anon.	exec. diff. (in %)	EPS w/o anon.	EPS w/ anon.
1	100	11ms	22ms	+100%	12836	4636
2	1000	25ms	149ms	+496%	40558	6730
3	10000	201ms	1436ms	+601%	50342	6997
4	100000	2229ms	14481ms	+550%	45347	6930
5	1000000*	25403ms	168979ms	+565%	41152	6143

Table 6.3: Benchmarks of anonymization overhead in terms of "executor run time (in ms)"

We can see that the overhead when anonymization is applied is approximately between 500%

[5]RDD Programming Guide - Spark 2.4.4. Documentation: https://spark.apache.org/docs/latest/rdd-programming-guide.html#rdd-operations

and 600% increase, which is significant. The reason is that we are using FPE function to anonymize email addresses, which had poor performance. Another reason is that we are anonymizing all fields, as mentioned in section 6.2.2, with another extra field, "raw_log", which contains sensitive information that also needs to be anonymized. The most optimal sample size to anonymize is with a sample size of 10000 log entries, with an estimated 50342 EPS. Another remark is that the test case #5 has failed to reach 150 iterations per sample size because maximum (13548m) Java heap memory was reached. In both attempts, it reached 55 and 66 iterations at most.

6.2.3 Memory usage

Memory usage has been evaluated based on one million records. We made sure that these fields were randomly generated so that optimizations or caching would be minimized during the computation process. In the first case, no pseudonymization was applied, and statistics were collected for the JVM current heap memory and used heap memory (by the objects). In the second instance, we applied anonymization on the following columns: source/destination IP, username, hostname and email address and raw_data (raw log).

Statistics information has been collected using "jstat", a tool for Java virtual machine statistics monitoring[6]. These measures were collected with the help of a Python code that would find the Java process and pass it to 'jstat' program to generate statistics. These statistics would then be processed and stored in a file. Our results are visible in the Figure 6.3 below.

We can notice large memory fluctuation; this is because 1 million records are loaded in Spark's dataset on which anonymization functions are computed with our custom user-defined functions (UDFs). One of the explanation for poor performance is that the algorithm that has been used to anonymize data on CrytopANT is calling the original code written in C language from Java code using Java Native Interface (JNI). It enables Java code to create a bridge allowing it to call native applications written in C, C++ or assembly, but it creates some overheads, and we also have experienced issues when many parallel calls are performed to pseudonymize IP address from our

[6]jstat - Java Virtual Machine Statistics Monitoring Tool, https://docs.oracle.com/javase/7/docs/technotes/tools/share/jstat.html, accessed on 2019 Oct 14th.

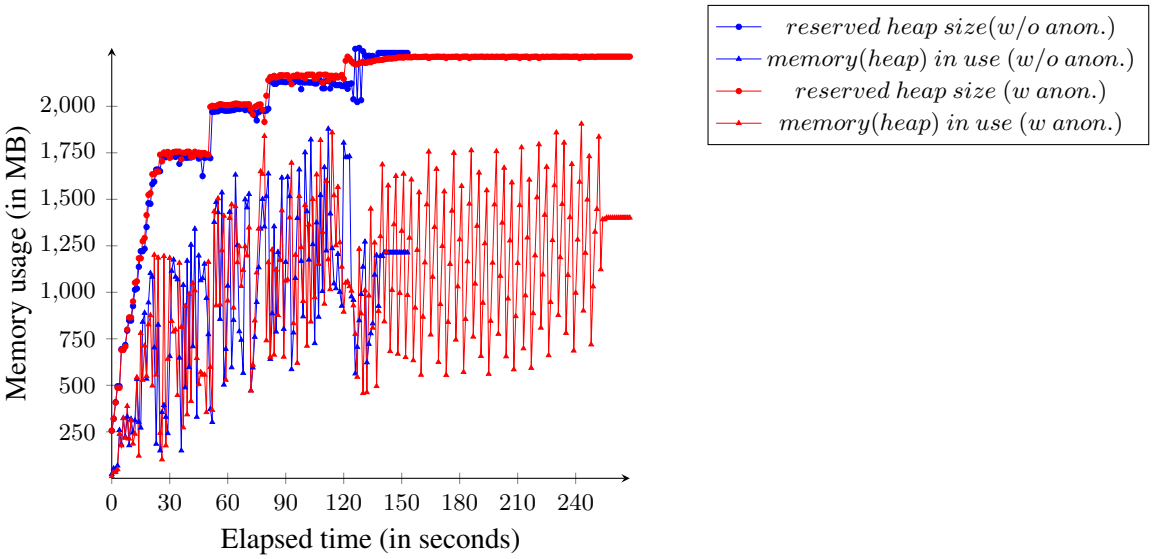

Figure 6.3: Benchmarks of anonymization overhead with and without anonymization in terms of memory usage

Java code.

6.2.4 CPU usage

We compare CPU usage similarly as in the previous experiments, with and without anonymization. This time, the sample dataset size of the benchmarks are of 1 million records. For CPU measure, we relied on JVMTOP[7], an open-source library, which relies on OperatingSystemMXBean[8].

The displayed CPU usage is in percentage and is collected every second after the process started to run. We can notice in both instances, with anonymization and without anonymization that until the first approximate 115 seconds, Apache Spark is loading 1 million records in a dataset. We can observe that the CPU usage is similar in both cases until the dataset sample creation is completed. In the case where no anonymization has been applied, the process finishes quickly, after approximately 25 seconds according to the table 6.3. In the second case, when anonymization is applied, it takes over 150 extra seconds to complete after a dataset sample is created, with CPU usage of

[7]Java monitoring for the command-line, profiler included: https://github.com/patric-r/jvmtop

[8]Monitoring and Management Interface for Java Platform: https://docs.oracle.com/javase/8/docs/technotes/guides/management/overview.html

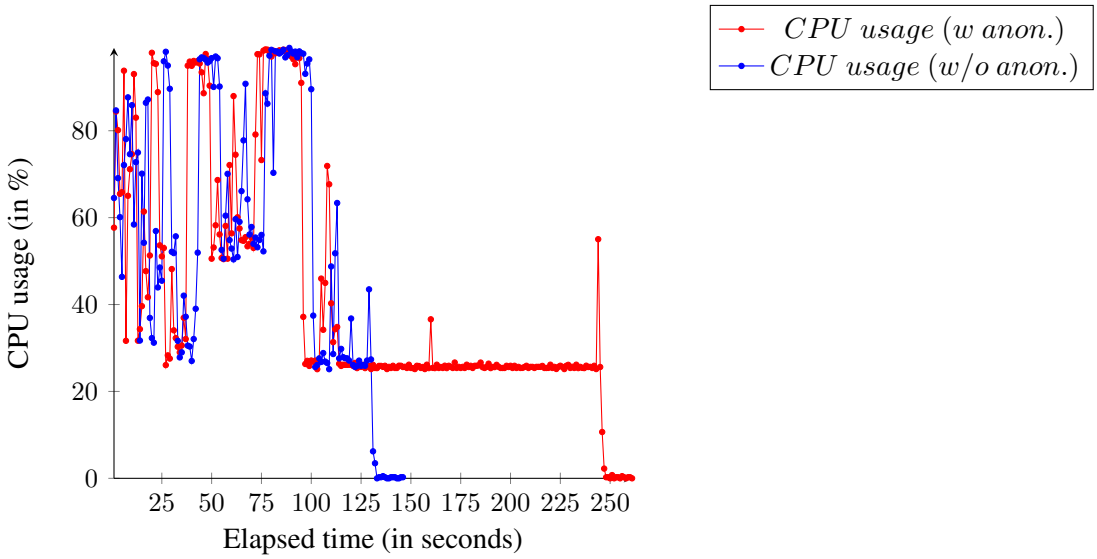

Figure 6.4: Benchmarks of CPU usage with and without anonymization on a dataset of 1 million records

approximately 26% because only one single core is used during the anonymization process.

We created another benchmark with CPU usage sample that can be seen in the Figure 6.5. The goal is to compare efficiency on CPU usage when FPE is disabled. We observe a significant improvement of approximately 90 seconds in time reduction, from 270 to 180 seconds, when anonymization using FPE on the email field is deactivated.

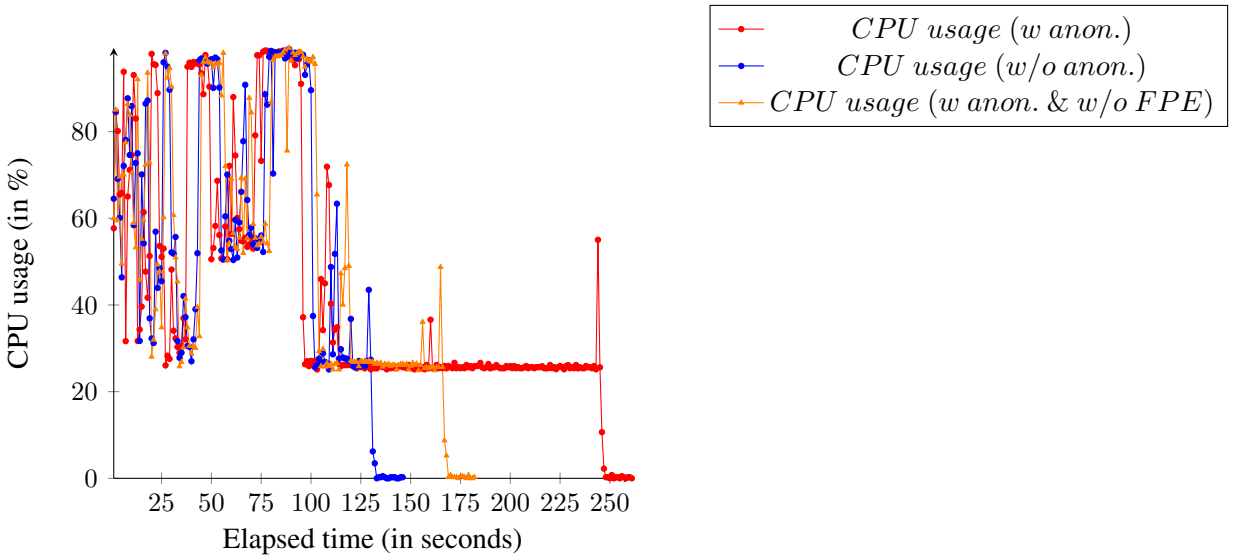

Figure 6.5: Benchmarks of CPU usage with and without anonymization on a dataset of 1 million records

6.2.5 Conclusion on benchmarks

As we have seen from various collected metrics from our benchmarks, the one anonymization function that affects the performance of our SIEM platform is FPE (FF1), as shown in Figures 6.2 and 6.5. Based on Figure 6.3, the estimated 6143 EPS is too low when anonymization is applied in six different fields using three anonymization algorithms: cryptopANT[31], AES symmetric encryption and format-preserving encryption[60] (FF1) on 1 million records. As we have noticed in Table 6.2, the anonymization function FPE is not scalable. In 6.3, when anonymization is applied, the performance is reduced dramatically. We believe that better results could be achieved if hardware resources were used at full capacity such as multi-threading and if cryptopANT algorithm would be used directly without having to rely on JNI implementation.

Chapter 7

Integration into a SIEM platform

In this section, we describe our implementation of a SIEM platform, the integration of the pseudonymization techniques into the platform, and lessons learned.

7.1 The architecture of our platform with PII pseudonymization

Our implementation relies on five components and technologies:

- Apache Kafka[1] is used to stream collected raw logs from the sensor.
- Apache Spark Stream[2] receives raw logs from Apache Kafka for processing, parsing, normalizing and anonymization.
- Apache Cassandra[3] database is used to stored processed data.
- Front-end implementation was done in Angular platform[4], which generates REST APIs queries from users (i.e., via web browsers).

[1] Apache Kafka: A Distributed Streaming Platform., https://kafka.apache.org/, accessed on 2020 Feb 11th.

[2] Apache Spark is a unified analytics engine for large-scale data processing., https://spark.apache.org/, accessed on 2020 Feb 11th.

[3] Apache Cassandra, Apache Cassandra is a NoSQL database for scalability, high availability, reliability and performance, http://cassandra.apache.org/, accessed on 2020 Feb 11th.

[4] Angular is a platform for building mobile and desktop web applications., https://angular.io/, accessed on 2020 Feb 11th.

- The backend web services are written in Java language with Spring Boot[5] framework. This service invokes queries to the Cassandra database and returns the results back to the user who has triggered the request initially.

We rely on Apache Spark, Kafka and Cassandra, because these are the latest technologies, which can enable us to perform large scale data processing in near real-time.

7.1.1 Description of our architecture

As shown in Figure 7.1, our SIEM platform works as follows.

- First, the event logs that are generated by the customers' devices are transmitted to our platform to be processed. These are raw logs collected in JSON format and forwarded to the Kafka data pipeline.
- Second, we process the logs using Apache Spark Stream, which ingests logs from Kafka in near real-time and applies the normalization part and anonymization.
- Third, these processed events are stored in Apache Cassandra database. Note that the logs are pseudonymized before they are being stored in the database.
- Fourth, the RESTful web service component queries Apache Cassandra database and returns the results to the fifth component, which is the front-end component implemented in Angular.

The Figure 7.1 shows the data flow from the Sensor to the SOC analysts. Also, a more detailed view of the Apache Spark component is visible in the left part of the figure.

7.1.2 Implementation

The implementation of the pseudonymization techniques is mainly based on the Apache Spark Stream. This is where the parsing, normalization and processing the pseudonymization processing happens. The code is mainly developed in Java programming language by relying on custom user-defined function (UDF) from Apache Spark.

[5]Spring Boot, Spring Boot is an open source Java-based framework used to create a micro Service.

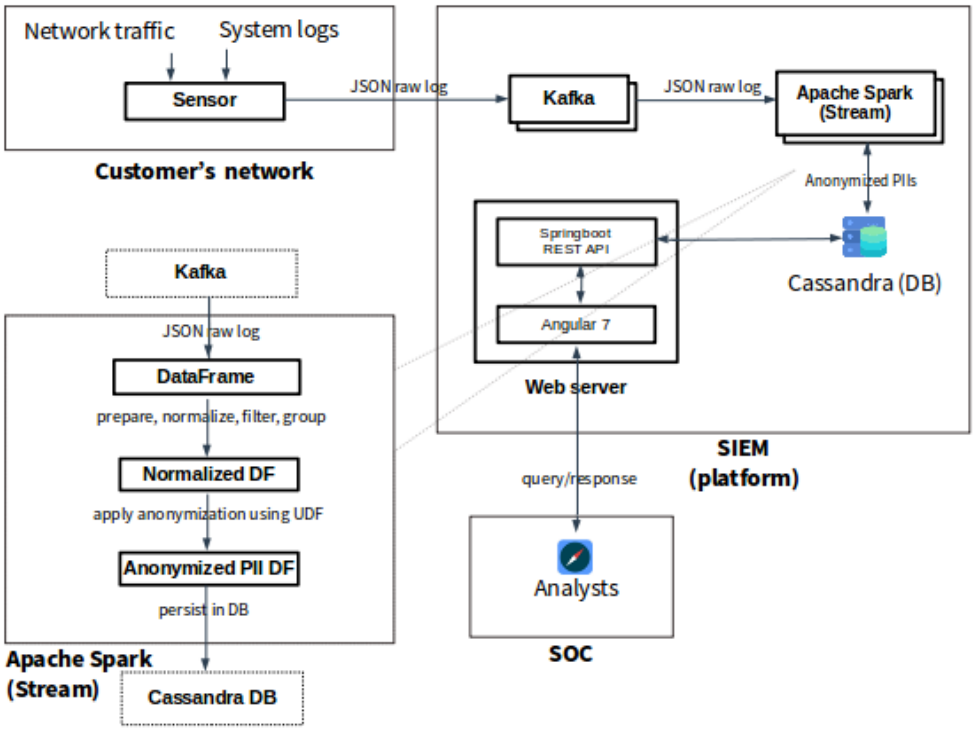

Figure 7.1: Architecture of our SIEM platform

The pseudonymization functions we have used are mostly symmetric key-based cryptographic encryption schemes, which allows our platform to achieve better performances than public key-based encryptions.

Log generation

To make our experiments more realistic, we generate various types of logs in JSON format. Specifically, five types of logs are generated, i.e., OSSEC[6], Snort[7], Argus' netflows [8], and Nessus vulnerabilities [9] and Nmap [10]. For our tests, we mainly follow OSSEC to create alerts such that they consisted of the most diverse data types or field (OSSEC is a host-based IDS that generates alerts based on correlation rules). These logs will be sent to the Kafka for real-time processing.

https://spring.io/projects/spring-boot, accessed on 2020 Feb 11th.

[6]OSSEC is a scalable, multi-platform, open source Host-based Intrusion Detection System (HIDS): https://www.ossec.net/

[7]Snort - Network Intrusion Detection & Prevention System: https://www.snort.org/

[8]ARGUS - Auditing Network Activity: https://qosient.com/argus/

[9]Nessus - Vulnerabiltiy Assessment: https://www.tenable.com

[10]Nmap - Nmap ("Network Mapper") is a free and open source (license) utility for network discovery and security auditing: https://nmap.org/

Based on the JSON formats of those five types of logs, our script can generate log entries consisted of randomly assigned IP addresses, emails, usernames. The log are sent to the Kafka topic for Apache Spark processing. Log generator source code in Python can be found in the Appendix Section A.12, and A.13. A sample of Snort logs in JSON format can be found in the Appendix, Section A.11

To generate a Snort type of logs, we relied on the rule2alert [11] open source project. This program allows us to generate alerts based on Snort rules. To convert these logs to JSON format, we used another script, u2json, which is part of idstools project[12].

The other approach we used to generate data is by redirecting events from Argus net-flows, OSSEC or Nessus from JSON format to Kafka. To redirect Argus net-flows, Kafka producer on topic test is launched by creating a netcat listening port 2552 as follows:

```
nc -vv -l 0.0.0.0 2552 | ./bin/kafka-console-producer.sh  \
--broker-list 192.168.2.4:9092 --topic test
```

Using ra, the argus client, logs are forwarded to netcat listening port 2552, which pipes data to Kafka producer as follows:

```
ra -L 0 -u -c ',' -S  192.168.2.14:561 | nc localhost 2552
```

Apache Spark Stream net-flow anonymization

In Figure 7.2, two datasets samples are displayed, each containing 20 records. We can also see collected net-flows using Argus client which is piped through Kafka to Apache Spark Stream for processing in near real-time, with window execution time set to every five seconds.

In the first dataset in Figure 7.2, a sample of 20 records is shown before any pseudonymization. In the second dataset, we can see that IP addresses are being anonymized by preserving the prefix (i.e., 192.168.2.0/24 $->$ 198.167.74/24) using CryptopANT masking function.

[11] Rule2Alert parses snort rules and generates packets on the fly that would alert the IDS. It can either write the packets to a pcap or send the packets directly to the IDS: https://github.com/pevma/rule2alert, accessed on 2019 Jun 2nd.

[12] u2json - A unified2 to JSON converter: https://idstools.readthedocs.io/en/latest/tools/u2json.html, accessed on 2019 Jun 2nd.

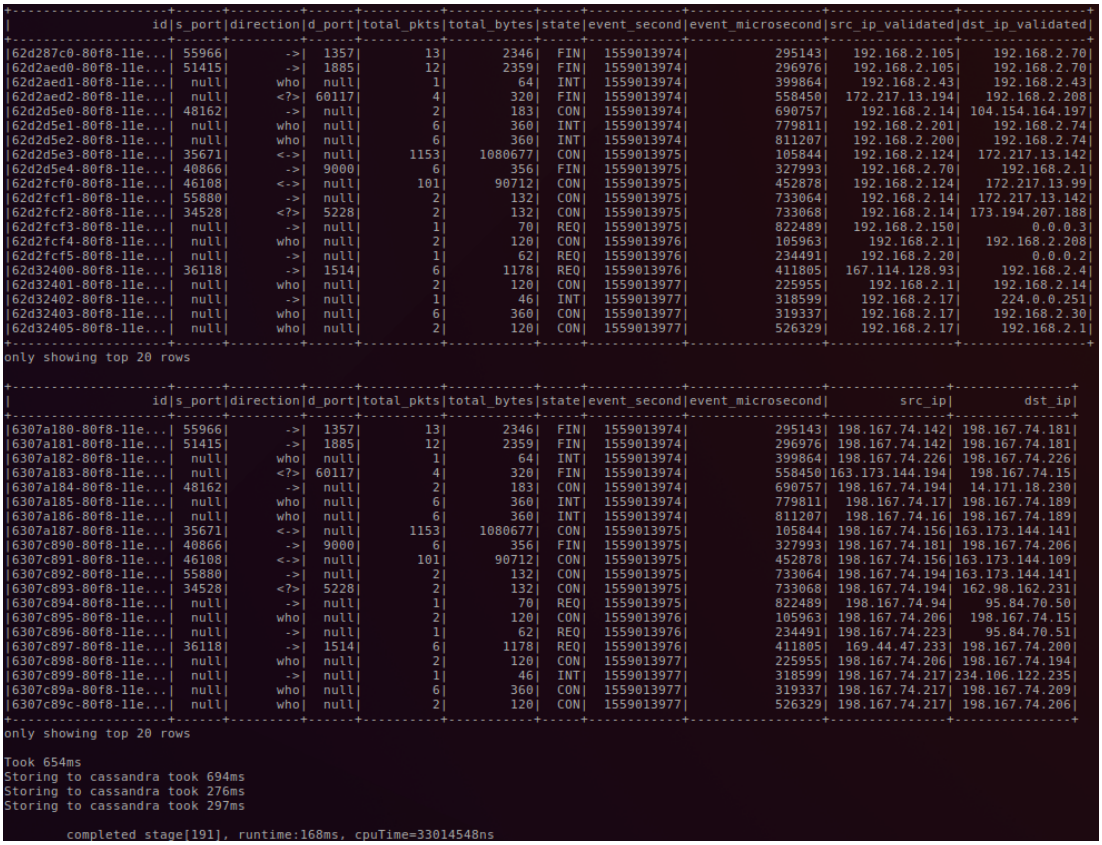

Figure 7.2: Apache Spark Net-flows

Presentation view

In this section, we show three different views of our SIEM platform to demonstrate its functionality, i.e., alerts in two screen captures, net-flows, and Nessus vulnerability scan views. The column hostname is encrypted under the AES symmetric encryption adapted such that generated ciphertexts remain deterministic, see listing A.1 of the Appendix. As the outputs of ciphertexts are in bytes, we are encoding the results using base64. The source code of symmetric encryption and decryption UDFs can be found in listings of A.4 and A.5

The source IP 159.187.99.79 is the anonymized IP address and is prefixed with a constant value "*src_ip*" to simplify the identification of the type of the value which has been pseudonymized. It is not always obvious to determine the nature of the pseudonymized value in other contexts with another type of raw logs. The raw data column is showing the anonymized values of the raw log. The column message is entirely encrypted using AES symmetric encryption before storing it in

Cassandra database, which is not indexed or searchable.

Only IP addresses are anonymized deterministically as in all other views, using the same secret key which is stored on the same server where Apache Spark is running. Ports are not anonymized to ease the utility of the data.

UUID.	Event second	Flags	Protocol	Src IP	Src Port	Direction	Dst IP	Dst Port	Total Pkts	Total bytes
ad2014f2-7fe5-11e9-958e-61dd62b6746c	1558894309	e		198.167.74.194	33162	->	163.173.144.202		3	214
aefd1251-7fe5-11e9-958e-61dd62b6746c	1558894691	e		198.167.74.156		->	234.106.122.235		4	1570
11c7fc92-7fe8-11e9-89c0-65b946f5a1cb	1558897018	e		198.167.74.89	8001	->	234.106.122.112	8001	6	1452
f7e87dd0-eecf-11e8-bed7-39dbe2d774e1	1542943754	e		8:bd:4ff:fe:3:73:c3:26		->	ff:ff:fff:fe:f:ff:ff:ff		4	240
b064dfb1-7fe5-11e9-958e-61dd62b6746c	1558894913	e		198.167.74.194	37432	<?>	198.167.74.209	8009	6	864
15b052a0-eed2-11e8-872a-d5ca4ecda4f4	1542944659	e		198.167.74.49	50377	->	198.167.74.156	8009	4	450

Figure 7.3: Net-flow view

The vulnerability scans view shown in Figure 7.4 also has only IP address pseudonymized. We can notice that the pseudonymized IP address “198.167.74.144” is also found in net-flow view allowing referencing between two different data types, net-flows and found vulnerabilities.

VULNERABILITY SCANS Filter

Scan date	Name	IP Address	Port	Protocol	Description	Plugin Id	CVE	CVSS	More
2019-05-26	ICMP Timestamp Request Remote Date Disclosure	198.167.74.144	0	icmp	The remote host answers to an ICMP timestamp request. This allows an attacker to know the date that is set on the targeted machine, which may assist an unauthenticated, remote attacker in defeating time-based authentication protocols. Timestamps returned from machines running Windows Vista / 7 / 2008 / 2008 R2 are deliberately incorrect, but usually within 1000 seconds of the actual system time.	10114	CVE-1999-0524		⋮
2019-05-26	ICMP Timestamp Request Remote Date Disclosure	198.167.74.149	0	icmp	The remote host answers to an ICMP timestamp request. This allows an attacker to know the date that is set on the targeted machine, which may assist an unauthenticated, remote attacker in defeating time-based authentication protocols. Timestamps returned from machines running Windows Vista / 7 / 2008 / 2008 R2 are deliberately incorrect, but usually within 1000 seconds of the actual system time.	10114	CVE-1999-0524		⋮
2019-05-26	IP Forwarding Enabled	198.167.74.206	0	tcp	The remote host has IP forwarding enabled. An attacker can exploit this to route packets through the host and potentially bypass some firewalls / routers / NAC filtering. Unless the remote host is a router, it is recommended that you disable IP forwarding.	50686	CVE-1999-0511	5.8	⋮

Figure 7.4: Nessus vulnerability scan view

7.1.3 Lessons learned

Based on our research, we have learned many practical lessons and received numerous feedbacks from real life SOC analysts. For instance, an important lesson we have learned is the following:

- The IDs of the items/records in front-end views can leak sensitive information about logs or devices (hosts) if these are not handled or managed in a careful way.

We also had many discussions with analysts about practical use cases based on their analysis work during their duty time, through which we have learned that:

- It is important to know which network range is external and which network range is internal, which needs to be monitored for protection. This step is crucial during their analysis work.
- Anonymizing (non-deterministically) IP addresses reduce the utility of event logs and alerts because they cannot be cross-referenced or linked (vulnerable IP addresses with the IP addresses of true attack alerts).
- For the SOC analysts, some functionalities may become limited in a SIEM platform where PIIs are anonymized. For example, searching on Google for more information on whether the IP address has been blacklisted would not be possible.

The reseach has demonstrated why privacy is important and about its origins. We have examined and compared multiple definitions and explanations about privacy and personally identifiable information (PII) from disciplines such as philosophy and laws. We have seen many different definitions and examples of PII privacy laws and directives, which helped us to better understand its relation with cybersecurity monitoring and network security monitoring (NSM).

Chapter 8

Conclusion

In this thesis, we have performed a case study on the application of anonymization techniques to real-world security monitoring practice. Specifically, we have created high-level use cases based on the SOC analyst's daily work, and we tried to understand which information is important in the security monitoring logs from the SOC analyst points of view. We then extracted fields from logs that are considered sensitive (PII) and applied a suitable anonymization technique per field and tried to validate if the analyst is still able to perform his work under different circumstances. Those techniques include AES in CTR mode, CryptopANT by relying on JNI to call the program in C language, CryptoPAN, and Format-Preserving-Encryption (FPE). We have compared different anonymization techniques and performend benchmarks to evaluate their performances in terms of CPU and memory usage, execution runtime, and throughput (operations per second). We also designed, and implemented a platform that includes front-end views for our experiments using scalable technologies (Kafka, Apache Spark Stream, Cassandra) by applying anonymization and pseudonymization techniques and comparing them. Finally, we shared our lessons learned.

We are aware that this approach does not provide sufficient security and privacy against active attackers, a stronger threat model. Nevertheless, our examination of existing pseudonymization methods can provide insights when applying anonymization techniques in large scale data ingestion applications. One limitation lies in the FPE algorithm which was only achieving approximately 10 000 encryptions per second. However, we believe that this is due to inefficient or un-optimized Java implementation we used for our benchmarks, which will be improved in our future work. As

another future direction, it is necessary to explore differential privacy (DP) using synthetic data [36] because it could improve privacy in security monitoring logs for stronger threat models. We believe that this would be a promising direction because analysts need to analyze individual events and not grouped or aggregated events as it is the case with k-anonymity and DP.

Bibliography

[1] J. Allen, D. Gabbard, C. May, E. Hayes, and C. Sledge, Outsourcing managed security services, tech. rep., DTIC Document, 2003

[2] Bejtlich, The Tao of network security monitoring: beyond intrusion detection. Pearson Education, 2004.

[3] Anderson, J.P., 1980. Computer security threat monitoring and surveillance, Tech. rep., Technical report, James P. Anderson Company, Fort Washington, Pennsylvania

[4] S. Axelsson. Intrusion detection systems: A survey and taxonomy. Technical Report 99-15, Department of Comp

[5] Liao H-J, Lin C-HR, Lin Y-C, Tung K-Y. Intrusion detection system: a comprehensive review. J Netw Comput Appl 2013;36(1):1624.

[6] Sandeep Kumar and Eugene H. Spafford. A pattern matching model for misuse intrusion detection. In Proceedingsings of the National Computer Security Conference, pages 1121, Baltimore, MD, 1994.

[7] I. Butun, S. D. Morgera, R. Sankar, A survey of intrusion detection systems in wireless sensor networks, IEEE Communications Surveys Tutorials 16 (1) (2014) 266282. doi:10.1109/SURV.2013.050113.00191

[8] Mustafa, H., & Xu, W. (2014, October). CETAD: Detecting evil twin access point attacks in wireless hotspots. In Communications and Network Security (CNS), 2014 IEEE Conference on (pp. 238-246). IEEE.

[9] H. Choi, S. Zhu, and T. F. La Porta. SET: Detecting Node Clones in Sensor Networks. In SecureComm 07, pages 341350, 2007.

[10] Pelechrinis K, Iliofotou M, Krishnamurthy SV. Denial of service attacks in wireless networks: the case of jammers. IEEE Communications Surveys and Tutorials 2011;13:24557.

[11] International Organization for Standardization - "Pseudonymization" new ISO specification supports privacy protection in health informatics, https://www.iso.org/news/2009/03/Ref1209.html

[12] V. Chandola, A. Banerjee, and V. Kumar, Anomaly detection: A survey,ACM Comput. Surv., vol. 41, no. 3, pp. 158, 2009.

[13] Staff, W., 2017. Yahoos 2013 Email Hack Actually Compromised Three Billion Accounts. WIRED. Available at: https://www.wired.com/story/yahoo-breach-three-billion-accounts/ [Accessed May 22, 2018].

[14] Haselton, T., 2017. Credit reporting firm Equifax says data breach could potentially affect 143 million US consumers. CNBC. Available at: https://www.cnbc.com/2017/09/07/credit-reporting-firm-equifax-says-cybersecurity-incident-could-potentially-affect-143-million-us-consumers.html [Accessed May 22, 2018].

[15] Lee, D., 2017. Uber concealed huge data breach. BBC News. Available at: http://www.bbc.com/news/technology-42075306 [Accessed May 6, 2018].

[16] Scott, C., Protecting Our Members. Available at: https://blog.linkedin.com/2016/05/18/protecting-our-members [Accessed May 22, 2018].

[17] Hackett, R., 2016. Linkedin lost 167 million account credentials in data breach. Fortune.com, Available at: http://fortune.com/2016/05/18/linkedin-data-breach-email-password/.

[18] European Parliament and Council, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance).

Chapter 4, Article 25," May 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX 32016R0679&from=EN#d1e3063-1-1

[19] European Parliament and Council, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance), Chapter 1, Article 4," May 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX 32016R0679&from=EN#d1e1489-1-1 Memorandums 07-16 and 06-19. GAO Report 08-536, Privacy: Alternatives Exist for Enhancing Protection of Personally Identifiable Information, May 2008, http://www.gao.gov/new.items/d08536.pdf.

[20] Solove, D.J., 2006. A Taxonomy of Privacy. University of Pennsylvania law review, 154(3), p.477. Available at: http://dx.doi.org/10.2307/40041279.

[21] Thomson, J.J., The right to privacy. In Philosophical Dimensions of Privacy. pp. 272289. Available at: http://dx.doi.org/10.1017/cbo9780511625138.012.

[22] J. Mayer, P. Mutchler, and J. C. Mitchell. Evaluating the privacy properties of telephone metadata. Proceedings of the National Academy of Sciences of the United States of America (PNAS), 113(20), May 2016

[23] Y.-A. de Montjoye, L. Radaelli, V. K. Singh, and A. Pentland, 'Unique in the shopping mall: On the reidentifiability of credit card metadata," Science, vol. 347, no. 6221, pp. 536539, 2015.

[24] Neil Richards, The Dangers of Surveillance, 126 HARV. L. REV. 1934, 196465 (2013).

[25] Lyon, D. (2007). Surveillance studies: An overview. Cambridge: Polity.

[26] Sweeney, L.: k-anonymity: A Model for Protecting Privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10(5), 557570 (2002)

[27] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. Int. J. Uncertain. Fuzz., 10(6):571588, 2002

[28] Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating Noise to Sensitivity in Private Data Analysis. In: Proceedings of the 3rd Theory of Cryptography Conference, pp. 265284 (2006)

[29] Greg Minshall. TCPdPriv: Program for eliminating confidential information from traces, 2005. http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html

[30] Y. Li, A. Slagell, K. Luo, and W. Yurcik. CANINE: A Combined Conversion and Anonymization Tool for Processing NetFlows for Security. In Proceedings of Tenth International Conference on Telecommunication Systems, 2005.

[31] cryptopANT IP Address Anonymization Library, https://ant.isi.edu/software/cryptopANT/index.html, accessed on 19 May, 2019.

[32] cryptopANT IP Address Anonymization Library, https://github.com/DNS-OARC/cryptopANT, accessed on 19 May, 2019.

[33] J. Xu, J. Fan, and M. H. Ammar. Prefix-Preserving IP Address Anonymization: Measurement-based Security Evaluation and a New Cryptography-based Scheme. In IEEE ICNP, 2002.

[34] Labix Blog, geohash.org is public!, https://blog.labix.org/2008/02/26/geohashorg-is-public, accessed on 2019 October 6th.

[35] Nissim K, Steinke T, Wood A, Altman M, Bembenek A, Bun M, Gaboardi M, OBrien D, Vadhan S. In press. Differential privacy: a primer for a non-technical audience. Vand. J. Ent.Technol. Law

[36] Abay, N.C., Zhou, Y., Kantarcioglu, M., Thuraisingham, B., Sweeney, L.: Privacy preserving synthetic data release using deep learning. The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (PKDD 2018)

[37] T. Brekne and A. rnes. Circumventing IP-Address Pseudonymization. In Proceedings of the 3^{rd} IASTED International Conference on Communications and Computer Networks, October 2005.

[38] W. Yurcik and etc. Scrub-tcpdump: A multi-level packet anonymizer demonstrating privacy/analysis tradeoffs. In Proceedings of the 3rd SECOVAL Workshop, September 2007.

[39] T. Brekne, A. Ames, and A. Oslebo, Anonymization of IP Traffic Monitoring Data Attacks on Two Prefix-Preserving Anonymization Schemes and Some Proposed Remedies, Workshop on Privacy Enhancing Technologies (PET), 2005.

[40] S. E. Coull, C. V. Wright, F. Monrose, M. P. Collins, and M. K. Reiter, Inferring Sensitive Information from Anonymized Network Traces, Network and Distributed Systems Security (NDSS), 2007.

[41] Martin L. Cyber kill chain. URL: $https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/Gaining_the_Advantage_Cyber_Kill_Chain.pdf$. 2015.

[42] A. F. Karr, C. N. Kohnen, A. Oganian, J. P. Reiter, and A. P. Sanil. A Framework for Evaluating the Utility of Data Altered to Protect Confidentiality. The American Statistician, 60(3):224232, 2006

[43] Slagell, A., K. Lakkaraju, and K. Luo, FLAIM: A Multilevel Anonymization Framework for Computer and Network Logs, LISA06, Washington DC, Dec 3-8, 2006.

[44] Ohm P (2010) Broken promises of privacy: responding to the surprising failure of anonymization. UCLA Law Review 57: 1701.

[45] A Narayanan, V. Shmatikov How To Break Anonymity of the Netflix Prize Dataset. arxiv cs/0610105, Oct. 2006.

[46] J. Zhang, N. Borisov, and W. Yurcik. Outsourcing Security Analysis with Anonymized Logs. Proceedings of the Workshop on the Value of Security through Collaboration, 2006.

[47] Mohammady, M., Wang, L., Hong, Y., Louafi, H., Pourzandi, M. and Debbabi, M., 2018,

October. Preserving Both Privacy and Utility in Network Trace Anonymization. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (pp. 459-474). ACM.

[48] M Meinig, P Trger, C Meinel, Finding Classification Zone Violations with Anonymized Message Flow Analysis. Proceedings of the 5th International Conference on Information Systems Security and Privacy, ICISSP 2019, Prague, Czech Republic, February 23-25, 2019.

[49] M. Burkhart, D. Brauckhoff, M. May, and E. Boschi. The Risk-Utility Tradeoff for IP Address Truncation. In ACM Workshop on Network Data Anonymization (NDA), 2008.

[50] Marrella, A., Monreale, A., Kloepper, B., and Krueger, M. W, 2016. Privacy-Preserving Outsourcing of Pattern Mining of Event-Log Data - A Use-Case from Process Industry. 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom).

[51] Slagell, A., Wang, J.& Yurcik, W., 2004. Network log anonymization: Application of crypto-pan to cisco netflows. In Proceedings of the Workshop on Secure Knowledge Management 2004. pdfs.semanticscholar.org. Available at: https://pdfs.semanticscholar.org/accc/29066bb4eb5d0f4dc71e75ef4a585015055a.pdf.

[52] Niksefat, S., Sadeghiyan, B., Mohassel, P., Sadeghian, S.: ZIDS: A privacypreserving intrusion detection system using secure two-party computation protocols. Comput. J. 57, 494509 (2014) Process. 79 (2015).

[53] R. Ostrovsky and W. E. Skeith, III, A survey of single-database private information retrieval: Techniques and applications, in Public Key Cryptography, Lecture Notes in Comput. Sci. 4450, T. Okamoto and X. Wang, eds., Springer, Berlin, 2007, pp. 393411.

[54] Yabo Xu, Benyu Zhang, Zheng Chen, and Ke Wang. 2007. Privacy-Enhancing Personalized Web Search. In International World Wide Web Conference. Ban, Canada.

[55] W. Gasarch. A survey on private information retrieval. The bulletin of the European Association for Theoretical Computer Science (EATCS), 82:72–107, 2004

[56] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In ACM SIGMOD, 2004.

[57] A. Boldyreva, N. Chenette, Y. Lee, and A. ONeill. Order preserving symmetric encryption. In EUROCRYPT, 2009.

[58] A. Boldyreva, N. Chenette, and A. ONeill. Order preserving encryption revisited: Improved security analysis and alternative solutions. In CRYPTO, 2011

[59] R. A. Popa, F. H. Li, and N. Zeldovich. An ideal-security protocol for order-preserving encoding. In IEEE Security and Privacy, 2013.

[60] Juan D. Vega (jdvega@idealista.com), fpe - Format Preserving Encryption Implementation in Java, GitHub repository, https://github.com/idealista/format-preserving-encryption-java, accessed on 2019 Oct 14th.

[61] M. Brightwell and H. Smith. Using datatype-preserving encryption to enhance data warehouse security. 20th National Information Systems Security Conference Proceedings (NISSC), pp. 141149, 1997. csrc.nist.gov/nissc/1997/proceedings/141.pdf.

[62] M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers. Format-preserving encryption. Selected Areas in Cryptography (SAC 2009), LNCS 5867, Springer, 2009. Also ePrint report 2009/251

[63] Y. Zhang, J. Katz, and C. Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In USENIX Security, 2016.

[64] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. Leakage-abuse attacks against searchable encryption. In CCS, 2015

[65] Paul Grubbs, Thomas Ristenpart, and Vitaly Shmatikov. Why your encrypted database is not secure, 2017.

[66] Paul Grubbs, Kevin Sekniqi, Vincent Bindschaedler, Muhammad Naveed, and Thomas Ristenpart. Leakage-abuse attacks against order-revealing encryption. In 2017 IEEE Symposium on Security and Privacy, pages 655672. IEEE Computer Society Press, May 2017

[67] Islam, M. S., Kuzu, M., and Kantarcioglu, M. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In 19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012 (2012).

[68] Muhammad Naveed, Seny Kamara, and Charles V. Wright. Inference attacks on property-preserving encrypted databases. In CCS, 2015.

[69] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In 2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000, pages 4455. IEEE Computer Society, 2000.

[70] E-J. Goh. Secure indexes. Technical Report 2003/216, IACR ePrint Cryptography Archive, 2003. See http://eprint.iacr.org/2003/216.

[71] Y. C. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In Applied Cryptography and Network Security Conference, 2005.

[72] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In A. Juels, R. Wright, and S. De Capitani di Vimercati, editors, ACM Conference on Computer and Communications Security (CCS 06), pages 7988. ACM, 2006

[73] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In CRYPTO, 2013

[74] C. Bösch, P. Hartel, W. Jonker, and A. Peter, A survey of provably secure searchable encryption, ACM Comput. Surv., vol. 47, no. 2, pp. 18:118:51, August 2014. [Online]. Available: http://doi.acm.org/10.1145/2636328

[75] G.S. Poh et al., Searchable Symmetric Encryption: Designs and Challenges, ACM Computing Surveys, vol. 50, no. 3, 2017

[76] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In Foundations of Secure Computation, pp. 169180, 1978.

[77] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. In Comm. of the ACM, 21:2, pages 120126, 1978.

[78] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC 2009, pp. 169178. ACM, New York (2009)

[79] Aleksey Shipilev. Java Microbenchmarks Harness (the lesser of two evils). http://shipilev.net/talks/ devoxx-Nov2013-benchmarking.pdf, 2013.

Appendix A

Listing A.1: Definition of SYMMETRIC_ENCRYPTION UDF

```
@Override
public String encrypt(final String message) {

   try {

      byte[] r = CryptoPrimitives.generateCmac(key, 0 + message);
      byte[] cipherText = CryptoPrimitives.encryptAES_CTR_String(key, r, message);
      return Base64.getEncoder().encodeToString(cipherText);
   } catch (UnsupportedEncodingException ex) {
      Logger.getLogger(AESEncryption.class.getName()).log(Level.SEVERE, null, ex);
   } catch (Exception ex) {
      Logger.getLogger(AESEncryption.class.getName()).log(Level.SEVERE, null, ex);
   }
   return "encryption failed..";
}

/* This source code sample originates from Clusion github projec:
https://github.com/encryptedsystems/Clusion/blob/master/src/main/java/org/crypto/sse/CryptoPrimitives.java#L232 */
public static byte[] encryptAES_CTR_String(byte[] keyBytes, byte[] ivBytes, String message)
      throws InvalidKeyException, InvalidAlgorithmParameterException, NoSuchAlgorithmException,
      NoSuchProviderException, NoSuchPaddingException, IOException, IllegalBlockSizeException, BadPaddingException {

   IvParameterSpec ivSpec = new IvParameterSpec(ivBytes);
   SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
   Cipher cipher = Cipher.getInstance("AES/CTR/NoPadding", "BC");
   cipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
   ByteArrayInputStream bIn = new ByteArrayInputStream(message.getBytes());
   CipherInputStream cIn = new CipherInputStream(bIn, cipher);
   ByteArrayOutputStream bOut = new ByteArrayOutputStream();

   int ch;
   while ((ch = cIn.read()) >= 0) {
      bOut.write(ch);
   }
   byte[] cipherText = concat(ivBytes, bOut.toByteArray());

   return cipherText;
}
```

Listing A.2: Definition of SYMMETRIC_DECRYPTION UDF

```
@Override
public String decrypt(final String cipherText) {

   try {
      byte[] r = CryptoPrimitives.generateCmac(key, 0 + cipherText);
      byte[] decrypt = CryptoPrimitives.decryptAES_CTR_String(key, r, Base64.getDecoder().decode(cipherText));
      return new String(decrypt);
   } catch (UnsupportedEncodingException ex) {
      Logger.getLogger(AESEncryption.class.getName()).log(Level.SEVERE, null, ex);
   } catch (Exception ex) {
      Logger.getLogger(AESEncryption.class.getName()).log(Level.SEVERE, null, ex);
   }
   return "decryption failed..";
}
/* This source code sample originates from Clusion github projec:
https://github.com/encryptedsystems/Clusion/blob/master/src/main/java/org/crypto/sse/CryptoPrimitives.java#L342 */
public static byte[] decryptAES_CTR_String(byte[] key, byte[] ivBytes, byte[] encryptedMessage)
      throws InvalidKeyException, InvalidAlgorithmParameterException, NoSuchAlgorithmException,
      NoSuchProviderException, NoSuchPaddingException, IOException {


   byte[] cipherText = new byte[encryptedMessage.length - 16];

   System.arraycopy(encryptedMessage, 0, ivBytes, 0, ivBytes.length);
   System.arraycopy(encryptedMessage, ivBytes.length, cipherText, 0, cipherText.length);

   IvParameterSpec ivSpec = new IvParameterSpec(ivBytes);
   SecretKeySpec secretKeySpec = new SecretKeySpec(key, "AES");
   Cipher cipher = Cipher.getInstance("AES/CTR/NoPadding", "BC");

   // Initalization of the Cipher
   cipher.init(Cipher.DECRYPT_MODE, secretKeySpec, ivSpec);

   ByteArrayOutputStream bOut = new ByteArrayOutputStream();
   CipherOutputStream cOut = new CipherOutputStream(bOut, cipher);

   cOut.write(cipherText);
   cOut.close();

   return bOut.toByteArray();
}
```

Listing A.3: Definition of Pseudo Random Function (PRF) used with AES encryption and decryption

```
/**
 * This method acts as Pseudo Random Function (PRF) which essentially
 * generates deterministically a ciphertext from two inputs:key and message.
 * source code origin:
      https://github.com/encryptedsystems/Clusion/blob/master/src/main/java/org/crypto/sse/CryptoPrimitives.java#L81
 * @param key secret key
 * @param msg message in clear
 * @return ciphertext
 * @throws UnsupportedEncodingException
 */
public static byte[] generateCmac(byte[] key, String msg) throws UnsupportedEncodingException {
   CMac cmac = new CMac(new AESFastEngine());
   byte[] data = msg.getBytes("UTF-8");
   byte[] output = new byte[cmac.getMacSize()];

   cmac.init(new KeyParameter(key));
   cmac.reset();
   cmac.update(data, 0, data.length);
   cmac.doFinal(output, 0);
   return output;
}
```

Listing A.4: Definition of SYMMETRIC_ENCRYPTION UDF

```
public static final UDF1 AES_CTR_ENCRYPTION = new UDF1<String, String>() {
   public String call(final String string) throws Exception {
      if(!ANONYMIZE){
         return string;
      }
      // automatically generates a nonce
      //System.out.println("encrypting string = "+ string);
      String encrypt = aesEncryption.encrypt(string);

      STATS_MAP.put(Constants.SYMMETRIC_ENCRYPTION, STATS_MAP.get(Constants.SYMMETRIC_ENCRYPTION)+1);
      //System.out.println("base64 = "+ encrypt);
      return encrypt;

   }
};
```

Listing A.5: Definition of SYMMETRIC_DECRYPTION UDF

```
public static final UDF1 AES_CTR_DECRYPTION = new UDF1<String, String>() {
   public String call(final String string) throws Exception {

      final byte[] plaintext = string.getBytes(StandardCharsets.UTF_8);

      String msg = aesEncryption.decrypt(new String(plaintext));

      return msg;

   }
};
```

Listing A.6: Definition of ANONYMIZE_IP UDF

```
public static final UDF1 ANONYMIZE_IP = new UDF1<String, String>() {

   public String call(String str) throws Exception {
      if(str == null) {
         return null;
      }

      if(!InetAddresses.isInetAddress(str)){
         if(VERBOSE)
            System.out.println("not a valid IP address = " + str);
         return null;
      }

      if(VERBOSE)
         System.out.println("anonymizeIP = " + str);

      if(!ANONYMIZE || !DO_ANONYMIZE_IP){
         return str;
      }

      String outputs = cryptopANT.anonymize(str);
      if(outputs != null && outputs.equalsIgnoreCase(str)){
         outputs = null;
      }

      // used for statistics
      STATS_MAP.put(Constants.ANONYMIZE_IP, STATS_MAP.get(Constants.ANONYMIZE_IP)+1);
      return outputs;
   }
};
```

Listing A.7: Applying ANONYMIZE_IP UDF on src and dst IP

```
   net_flows = net_flows.select(
         col("*"),
         callUDF(Constants.ANONYMIZE_IP, col("src_ip_validated")).as("src_ip"),
         callUDF(Constants.ANONYMIZE_IP, col("dst_ip_validated")).as("dst_ip")).
         drop("src_ip_validated").
         drop("dst_ip_validated");
```

Listing A.8: CryptopANTJNI Java JNI class that uses CryptopANT code in C

```
public class CryptopANTJNI {

   private boolean textBased = true;
   private String keyFile = "/home/alis/gitlab/cryptopANT/cryptopANT-1.0.3_modified/secret_key";
   private String pass4 = "--pass4=5";

   private CryptopANTJNI cryptopANT;

   private native void setup(String msg);
   private native String encrypt(String msg);
   private native String decrypt(String msg);

   static {
      System.loadLibrary("cryptopant");
   }

   public CryptopANTJNI(){
   }

   public CryptopANTJNI(String keyFileLocation){
      cryptopANT = new CryptopANTJNI();
   }

   public CryptopANTJNI build() {
      CryptopANTJNI cryptopANT = new CryptopANTJNI();
      String textBasedArg = (textBased) ? " -t " : "";
      String args = "./program " + pass4 + textBasedArg + keyFile;
      cryptopANT.setup(args);
      return cryptopANT;
   }

   public String anonymize(String ipAddress) {
      return encrypt(ipAddress);
   }

   public String denonymize(String ipAddress) {
      return decrypt(ipAddress);
   }
```

Listing A.9: Implementation of setup method declared in scramble_ips.c which calls main() function

```
JNIEXPORT void JNICALL Java_maskingtechniques_CryptoANTJNI_setup(JNIEnv *env, jobject thisObj, jstring inJNIStr) {
   char *inCStr = (*env)->GetStringUTFChars(env, inJNIStr, NULL);

   char array[100][100];
   int i = 0;
   int s = 0;
   char *pArgs = strtok(inCStr," ");
   while((pArgs = strtok(NULL, " ")) != NULL) {
      int k;
      for(k=0; k<strlen(pArgs);k++){
            array[s][k] = *(pArgs + k);
```

```
      }
      array[s][k] = '\0';
      s++;
   }

   int argc = s;
   char* argv[argc + 1];
   int position;
   for(position=0;position<s;position++){
      argv[position] = array[position];
   }

   int status = main(argc, argv);
}
```

Listing A.10: Implementation of native encryption and decryption methods declared in scramble_ips.c file

```
JNIEXPORT jstring JNICALL Java_maskingtechniques_CryptoANTJNI_encrypt(JNIEnv *env, jobject thisObj, jstring inJNIStr) {

      // Step 1: Convert the JNI String (jstring) into C-String (char*)
      char *inCStr = (*env)->GetStringUTFChars(env, inJNIStr, NULL);

      ipIn = inCStr;

      char cbuf[BUFSIZE];
      char cbuf2[BUFSIZE];

      strncpy(cbuf, ipIn, strlen(ipIn));
      cbuf[strlen(ipIn)] = '\0';
      anon_ip4_txt(cbuf, cbuf2);
      return (*env)->NewStringUTF(env, cbuf2);
}

JNIEXPORT jstring JNICALL Java_maskingtechniques_CryptoANTJNI_decrypt(JNIEnv *env, jobject thisObj, jstring inJNIStr) {
      // Step 1: Convert the JNI String (jstring) into C-String (char*)
      char *inCStr = (*env)->GetStringUTFChars(env, inJNIStr, NULL);

      ipIn = inCStr;
      char cbuf[BUFSIZE];
      char cbuf2[BUFSIZE];

      strncpy(cbuf, ipIn, strlen(ipIn));
      cbuf[strlen(ipIn)] = '\0';
      reverse_mode = 1;
      anon_ip4_txt(cbuf, cbuf2);
      reverse_mode = 0;
      return (*env)->NewStringUTF(env, cbuf2);
}
```

Listing A.11: Log sample of Snort alerts - snort.unified2.1537860154.json


```
1  {"type": "event", "event": {"impact": 0,"generator-id": 1,"protocol": 1,"dport-icode": 0,"signature-revision": 7,
       "classification-id": 14,"signature-id": 2100368,"sensor-id": 0,"impact-flag": 0,"sport-itype": 8,"priority": 3,
       "event-second": 1537860157,"pad2": null, "destination-ip": "192.168.2.1", "event-id": 1,"mpls-label": null,
       "vlan-id": null, "source-ip": "192.168.2.119", "event-microsecond": 95888,"blocked": 0}}
2  {"type": "packet", "packet": {"packet-second": 1537860157,"linktype": 1,"sensor-id": 0,"packet-microsecond": 95888,
       "event-second": 1537860157,"length": 98,"data":
       "ROndUGqkpHczpCD2CABFAABUAABAAEABtODAqAJ3wKgCAQgApvEPRAABPeKpW2+IAAAICQoLDA0ODxAREhMUFRYXGBkaGxwdHh8gISIjJCUmJygpKissLS4vMDEyMzQ1Njc=",
       "event-id": 1}}
3  {"type": "event", "event": {"impact": 0,"generator-id": 1,"protocol": 1,"dport-icode": 0,"signature-revision": 8,
       "classification-id": 14,"signature-id": 2100366,"sensor-id": 0,"impact-flag": 0,"sport-itype": 8,"priority": 3,
       "event-second": 1537860157,"pad2": null, "destination-ip": "192.168.2.1", "event-id": 2,"mpls-label": null,
       "vlan-id": null, "source-ip": "192.168.2.119", "event-microsecond": 95888,"blocked": 0}}
4  {"type": "packet", "packet": {"packet-second": 1537860157,"linktype": 1,"sensor-id": 0,"packet-microsecond": 95888,
       "event-second": 1537860157,"length": 98,"data":
       "ROndUGqkpHczpCD2CABFAABUAABAAEABtODAqAJ3wKgCAQgApvEPRAABPeKpW2+IAAAICQoLDA0ODxAREhMUFRYXGBkaGxwdHh8gISIjJCUmJygpKissLS4vMDEyMzQ1Njc=",
       "event-id": 2}}
5  {"type": "event", "event": {"impact": 0,"generator-id": 1,"protocol": 1,"dport-icode": 0,"signature-revision": 7,
       "classification-id": 14,"signature-id": 2100368,"sensor-id": 0,"impact-flag": 0,"sport-itype": 8,"priority": 3,
       "event-second": 1537860158,"pad2": null, "destination-ip": "192.168.2.1", "event-id": 3,"mpls-label": null,
       "vlan-id": null, "source-ip": "192.168.2.119", "event-microsecond": 108994,"blocked": 0}}
6  {"type": "packet", "packet": {"packet-second": 1537860158,"linktype": 1,"sensor-id": 0,"packet-microsecond": 108994,
       "event-second": 1537860158,"length": 98,"data":
       "ROndUGqkpHczpCD2CABFAABUAABAAEABtODAqAJ3wKgCAQgAx78PRAACPuKpW025AAAICQoLDA0ODxAREhMUFRYXGBkaGxwdHh8gISIjJCUmJygpKissLS4vMDEyMzQ1Njc=",
       "event-id": 3}}
7  {"type": "event", "event": {"impact": 0,"generator-id": 1,"protocol": 1,"dport-icode": 0,"signature-revision": 8,
       "classification-id": 14,"signature-id": 2100366,"sensor-id": 0,"impact-flag": 0,"sport-itype": 8,"priority": 3,
       "event-second": 1537860158,"pad2": null, "destination-ip": "192.168.2.1", "event-id": 4,"mpls-label": null,
       "vlan-id": null, "source-ip": "192.168.2.119", "event-microsecond": 108994,"blocked": 0}}
8  {"type": "packet", "packet": {"packet-second": 1537860158,"linktype": 1,"sensor-id": 0,"packet-microsecond": 108994,
       "event-second": 1537860158,"length": 98,"data":
       "ROndUGqkpHczpCD2CABFAABUAABAAEABtODAqAJ3wKgCAQgAx78PRAACPuKpW025AAAICQoLDA0ODxAREhMUFRYXGBkaGxwdHh8gISIjJCUmJygpKissLS4vMDEyMzQ1Njc=",
       "event-id": 4}}
9  {"type": "event", "event": {"impact": 0,"generator-id": 1,"protocol": 6,"dport-icode": 22,"signature-revision": 2,
       "classification-id": 16,"signature-id": 2019876,"sensor-id": 0,"impact-flag": 0,"sport-itype": 58271,"priority": 3
       ,"event-second": 1537860172,"pad2": null, "destination-ip": "192.168.2.4", "event-id": 5,"mpls-label": null,
       "vlan-id": null, "source-ip": "116.31.116.42", "event-microsecond": 134142,"blocked": 0}}
10 {"type": "packet", "packet": {"packet-second": 1537860172,"linktype": 1,"sensor-id": 0,"packet-microsecond": 134142,
       "event-second": 1537860172,"length": 81,"data":
       "xDRrUaaKROndUGqkCABFAABDFelAADAGidZ0H3QqwKgCBOOfABZkhzT5VptU/4AYAOU/xAAAAQEIChoOIV8Brh+IU1NILTIuMC1QVVRUWQ0K",
       "event-id": 5}}
11 {"type": "event", "event": {"impact": 0,"generator-id": 1,"protocol": 1,"dport-icode": 0,"signature-revision": 7,
       "classification-id": 14,"signature-id": 2100368,"sensor-id": 0,"impact-flag": 0,"sport-itype": 8,"priority": 3,
       "event-second": 1537860187,"pad2": null, "destination-ip": "192.168.2.1", "event-id": 6,"mpls-label": null,
       "vlan-id": null, "source-ip": "192.168.2.119", "event-microsecond": 507704,"blocked": 0}}
12 {"type": "packet", "packet": {"packet-second": 1537860187,"linktype": 1,"sensor-id": 0,"packet-microsecond": 507704,
       "event-second": 1537860187,"length": 98,"data":
       "ROndUGqkpHczpCD2CABFAABUAABAAEABtODAqAJ3wKgCAQgAWCYPxwABW+KpW5nQBgAICQoLDA0ODxAREhMUFRYXGBkaGxwdHh8gISIjJCUmJygpKissLS4vMDEyMzQ1Njc=",
       "event-id": 6}}
```

Listing A.12: Log generator (main)

```
#!/usr/bin/env python
import sys
import configurations as cnf
from configurations import logger
from generate_snort_logs import GenerateSnortLogs
from generate_ossec_logs import GenerateOSSECLogs
from generate_nessus_logs import GenerateNessusLogs
from generate_nmap_logs import GenerateNmapLogs
from generate_incident_logs import GenerateIncidentLogs

class LogGenerator:
   '''General Log generator class'''

   path = "../logs/"
   log_types = {
      "types":[
         {"type":"ossec", "path": path+"ossec/"},
         {"type":"correlation", "path": path+"correlation/"},
         {"type":"incident", "path": path+"incidents/"},
         {"type":"nessus", "path": path+"nessus/"},
         {"type":"snort", "path": path+"snort/"},
         {"type":"nmap", "path": path+"nmap/"}
      ]
   }

   def __init__(self):
      print("Which type of logs do you wish to generate ?\n")
      options = {}
      for key,value in self.log_types.items():
         for index, record in enumerate(self.log_types[key], start=1):
            options[index] = record
            print(str(index)+")"+record["type"])
      while True:
         selected_option = input("Enter the number of the selected option: ")

         if type(selected_option) == int and selected_option in options:
            print("genearting logs of type "+options[selected_option]["type"] + ", path
                  ="+options[selected_option]["path"])
            break
         else:
            print("entered invalid option! please enter a valid option.")

      while True:
         quantity = input("Enter a number of records: ")

         if type(quantity) == int:
            print("generating " + str(quantity) + " records.")
            break
         else:
            print("entered invalid option! please enter a valid option..")

      if options[selected_option]["type"] == "snort":
         GenerateSnortLogs(options[selected_option]["path"], quantity)
      if options[selected_option]["type"] == "ossec":
```

```
            GenerateOSSECLogs(options[selected_option]["path"], quantity)
        if options[selected_option]["type"] == "nessus":
            GenerateNessusLogs(options[selected_option]["path"], quantity)
        if options[selected_option]["type"] == "incident":
            GenerateIncidentLogs(options[selected_option]["path"], quantity)
        if options[selected_option]["type"] == "nmap":
            GenerateNmapLogs(options[selected_option]["path"], quantity)
        if options[selected_option]["type"] == "correlation":
            GenerateCorrelationLogs(options[selected_option]["path"], quantity)

def main():
    LogGenerator()

if __name__ == "__main__":
    print("python version: "+str(sys.version))
    main()
```

Listing A.13: Log generator (snort type or alerts)

```
#!/usr/bin/env python
from configurations import *
from kafka import KafkaProducer, KafkaConsumer

class GenerateSnortLogs:
    ''' https://github.com/dpkp/kafka-python/blob/master/example.py'''
    def __init__(self, path, quantity):
        self.producer = KafkaProducer(bootstrap_servers=KAFKA_ADDRESS+":"+KAFKA_PORT)
        self.path = path
        if (quantity%2) == 0:
            print("quantity is even")
        else:
            print("quantity is odd, adding 1")
            quantity+=1
        self.quantity = quantity
        files = os.listdir(self.path)
        self.total_lines = 0
        print("generating " + str(quantity) + " of snort logs.")
        for file_name in files:
            self.parse_file(file_name)
        logger.info("parsed "+str(self.total_lines))

    def parse_file(self, file_name):
        if file_name.endswith(".json"):
            logger.info("parsing : " + self.path + file_name)
            for line in self.read_lines(self.path +file_name):
                json_line = load_line_in_json(line)
                if not json_line:
                    break
                json_line["data_type"] = "snort"
                #randomnizing ips
                if "event" in json_line:
                    json_line["event"]["source-ip"] = str("10.2.") + str('.'.join([str(randint(0,255)) for x in range(2)]))
                    json_line["event"]["destination-ip"] = str("10.2.") + str('.'.join([str(randint(0,255)) for x in
                            range(2)]))
                line_str = json.dumps(json_line)
```

```
            if(self.total_lines >= self.quantity):
               break
            line_str = json.dumps(json_line)
            if(self.total_lines >= self.quantity):
               break
            self.send_to_kafka(line_str)
            self.total_lines +=1

   def read_lines(self, file):
      with open(file, 'r') as f:
         for line in f.readlines():
            yield line

   def send_to_kafka(self, line):
      print "using topic " + KAFKA_TOPIC
      self.producer.send(KAFKA_TOPIC, line)
      self.producer.flush()
```

Listing A.14: Snort rule signature for heartbleed attack

```
alert tcp $HOME_NET 443 -> $EXTERNAL_NET any (msg:"SERVER-OTHER
TLSv1.1 large heartbeat response - possible ssl heartbleed attempt";
flow:to_client,established; content:"|18 03 02|"; depth:3;
byte_test:2,>,128,0,relative; detection_filter:track by_dst, count 5,
seconds 60; metadata:policy balanced-ips drop, policy security-ips
drop, service ssl; reference:cve,2014-0160; classtype:attempted-recon;
sid:30516; rev:3;)
```