1

Introduction

The high penetration of data-centric services has markedly increased the risk of exposing sensitive customer and corporate data. In particular, the sectors of critical infrastructures (CIs), information technology (IT), and mobile computing are those which are constantly targeted by sophisticated adversaries, insiders, or bribed workers who launch an attack using advanced malware and hacking tools [1]. The main purpose of these attackers is to gain long-term access to a system and steal critical sensitive data from the enterprise. This causes data breach or data leakage, also known as *data exfiltration*, which poses huge losses every year to a wide range of industries including many technology giants such as Google, Facebook, and Tesla. Google alone stores an enormous volume of sensitive data derived from sources worldwide [2]. Despite the Covid-19 pandemic, in 2020, of the reported 32,002 incidents, 3950 involved data breaches [3], of which 86% were financially motivated. Recent outbreaks of ransomware are good examples of new data exfiltration-based attacks for the purpose of financial gain [4]. Not only are attack methods becoming increasingly sophisticated, but most of the advanced hacking is conducted by state-sponsored hackers [5]. Moreover, the consequences of data leakage could pose a critical threat to security and the privacy of users, particularly those who work for the government or the military. For instance, the recent leakage of the subcontractor database of the Australian Defence Force had severe implications for national security as the design of military combat aircraft was compromised [6].

Intruders take the opportunity to exploit the unknown vulnerabilities of security systems in order to penetrate an organization. Stuxnet worm [7] was one of the most well-known attack tools created by the intelligence agencies of the United States and Israel and was intended to thwart the Iranian nuclear development program. Indeed, this worm virus could be classified as a military-grade weapon considering its origin and sophistication. Stuxnet was used to exploit several undiscovered bugs in the Windows operating system to wirelessly spread itself and install a rootkit on the Siemens' programmable logic controllers (PLCs),

Data Exfiltration Threats and Prevention Techniques: Machine Learning and Memory-Based Data Security, First Edition. Zahir Tari, Nasrin Sohrabi, Yasaman Samadi, and Jakapan Suaboot.

which were manipulated and used to destroy the delicate equipment at the targeted nuclear power plant. Another example of the widespread vulnerability of security systems is the OpenSSL Heartbleed [8]. There was no clear evidence indicating whether the bug was discovered and used by hackers before 2014, and for how long. Despite having a mathematically robust design, the implementation bug was accidentally embedded in the widely used OpenSSL cryptography library for decades. Since OpenSSL is the core of Transport Layer Security (TLS) encryption, vital services like secure web pages, email, or the secure shell protocol had been affected. Unfortunately, the security patching process took several months or more to fix the widespread vulnerability in several million devices worldwide. This resulted in several data breach incidents, such as the data breaches involving the Community Health Systems (CHS), which was the giant private hospital chain in the United States, and the Canada Revenue Agency (CRA) incident that impacted millions of Canadian taxpayers.

Since attack prevention measures might not be adequate, the detection-based approach plays a crucial role in minimizing damages resulting from data breaches. Generally, a malicious program (known as malware) is the primary tool employed by adversaries to help them gain access to a system or even automatically exfiltrate sensitive information. In spite of numerous malware detection approaches proposed in the literature, few works focus on data exfiltration. On the other hand, the studies that propose solutions for data breach detection focus only on a single data leakage channel: network monitoring, unsafe data exportation, or user authorization, to name a few. Such atomistic solutions give attackers opportunities to exfiltrate sensitive data via alternative channels. This suggests the need for a holistic data exfiltration detection approach. Hence, this research investigates several methods for detecting data exfiltration, which is crucial for various corporate sectors as the modern world is evolving toward the adoption of data-centric services.

Generally, off-the-shelf antivirus scanners primarily use known signatures to detect malicious programs. The signature-based solution has a very low false alarm rate, as it uses a hash of the program or signature strings contained in the malware binary to match with the virus signature. However, advanced hackers could develop a new unseen malware for the well-protected target or even use the benign program to steal the data. Furthermore, some sophisticated malware does not need to be installed on the disk storage at all (e.g. Code Red worm [9]). Instead, this malware could run in the memory to perform the malicious activity for the entire time. This is where the real-time behavior detection method plays an important role. However, detecting a nefarious purpose from a series of the program's actions is challenging, as there are too many possibilities that the program's actions infer the malicious transaction, especially in a real-time context. Hence, behavior-based data exfiltration is one issue that needs a careful and systematic investigation.

Apart from using malware to steal sensitive data, some attackers might simply exploit the benign program to exfiltrate the sensitive information. In some cases, the monitoring system cannot detect the unforeseen malware. Hence, the malicious behavior-monitoring approach alone might not be sufficient. Therefore, by taking different perspectives, we may be able to obtain a list of processes used to access sensitive data, which could cause data leakage. To do so, one will need to search for sensitive data in the memory space of those processes. This will allow sensitive files to be read, and other inputs such as keystrokes containing sensitive keywords, to be detected. Ultimately, all data in the computer system needs to be loaded into the main memory before it can be processed; hence, the adversary cannot avoid having the sensitive data in the main memory. Moreover, the data can remain in the memory even if the process has already been terminated [10]. Hence, if those processes can be listed, it will be easier to narrow down and identify the root cause of the data breach regardless of whether the program is classified as malicious or benign. However, to the best of the authors' knowledge, this idea has not been closely examined by previous studies, and therefore this book will elucidate new ways of addressing such issues.

A sophisticated adversary could obfuscate the data exfiltration even more by using a temporal attack [11] to evade the detection system. Here, the data-stealing activity is delayed so as to trick the monitoring system into "thinking" that it is just a false alarm (false positive). In other words, the hacker could minimize the chance of being discovered by the monitoring system by stealing a minuscule amount of sensitive data at a time. Over a period of time, the attacker could reassemble those small amounts to form the original sensitive file. Even though these time-delay attacks have been reported for over a decade, few researchers have attempted to address the issue (e.g. [11, 12]). If this method is used to penetrate the critical systems of government departments or the military, the consequences will be devastating. Hence, this book looks at the data exfiltration detection issue by **holistically** approaching the problem from several perspectives, namely by: (i) examining the program's behavior, (ii) monitoring sensitive data program is accessing, and (iii) monitoring the collective activities of the process related to fractions of sensitive information being collected over a period of time.

1.1 Data Exfiltration Methods

The prevention of data exfiltration is a broad and complex issue. To examine this issue more closely, current methods can be categorized into four areas: (i) state-of-the-art survey, (ii) behavior-based data exfiltration solution, (iii) memory-based data exfiltration solution, and (iv) temporal-based data exfiltration prevention. The shortcomings of each of these methods are discussed here.

1.1.1 State-of-the-Art Surveys

To begin with, this book surveyed technologies that have a high potential to be used as a fundamental building block for data leakage prevention (DLP) methods, and data exfiltration prevention solutions shared similar core technologies that are used for intrusion detection systems (IDSs). While an IDS is a standard measure to protect computer systems from outsider and insider attacks, DLP is a more specialized and advanced security solution that can provide a better protection against security breaches. DLP aims to detect abnormal access to sensitive data, and this is based on the use of either machine learning (ML) or temporal reasoning algorithms.

Industrial control systems (ICSs), especially supervisory control and data acquisition (SCADA) systems, are a constant target by cyberattacks as their inability to function can have serious impact on people's daily lives. This obviously creates a nightmare to the security community to find the right solutions that could protect ICSs' essential services, so these can provide the required functionalities. IDSs, when tailored to deal with specific requirements (e.g. real-time detection and reliability), can provide an appropriate protection to ICSs.

Despite several attempts to address SCADA security, only a few of the studies reported in the literature have focused on the development of appropriate ML solutions for SCADA systems. Many of the researchers examined, from different perspectives, the design of specific IDS for CIs. For example, the survey conducted in [13] focused on the design of IDS architecture for SCADA systems. In the other hand, the work in [14] looked at classifying IDS solutions based on detection methods and audit materials. In [15], a review is conducted to gather information about the testbeds used for ICSs. Therefore, it is crucial to conduct a survey based on the supervised ML methods, focusing on CI which has the potential to contribute to the research on efficient methods and technologies for the prevention of data leakage.

1.1.2 Malicious Behavior Detection Methods

Malware has become one of the most effective tools for searching and stealing users' sensitive information. Over several decades, numerous malware detection methods have been proposed in the literature. However, only a few current malware detection solutions focus directly on the detection of data exfiltration; instead, they are used to detect different types of virus/malicious programs. Intuitively, if a Trojan horse software accesses sensitive data, it is highly likely to cause data leakage. Hence, observing the behavior of all running processes (to identify potential malware) is the key to detecting data breaches.

Nevertheless, in most cases [16], data is stolen by malware but remains undetected for two reasons [17]: the signature of the malware is new to antivirus

programs, and the anomaly-based systems cannot detect variations in the malware behavior and therefore cannot differentiate between legitimate and exfiltration activities. Based on a sequence of application programming interface (API) calls invoked by a process, hidden Markov model (HMM) has been used extensively to discriminate between malicious and benign behavior (e.g. [18–20]). However, the dynamic behavior analysis of a malware can generate a very long sequence of API calls. For example, in the experiments carried out by our team, Keylogger malware (MD5 hash value `d4259130a53dae8ddce89ebc8e80b560`) generated more than 300,000 series of API calls in less than two minutes. Also, HMM seems to perform poorly with such long sequences of API calls. This allows the exfiltration-based malware to pose as a benign software for most of the time and perform malicious activities only for a very short period of time to avoid detection. Hence, the effect of the sequence length on the detection accuracy which has not been fully studied in the context of malware detection will be examined in more detail in this book.

1.1.3 RAM-Based Data Exfiltration Detection Methods

The detection of suspicious activities is one approach used to prevent data exfiltration. However, the behavior-monitoring approach has two significant limitations: (i) it covers only certain types of malicious activity (e.g. searches for user identity/credentials, installation of a keystroke-capturing service, and attempts to contact or export the user's documents to the malicious server). However, the adversary could pass on the data through several legitimate services by, for instance, attaching the sensitive file with email or putting in the email content, inserting the sensitive data into the untrusted database server, storing the sensitive data in files which will be exported later. (ii) The behavior-monitoring system cannot totally guarantee the detection of all suspicious incidents. Therefore, one needs a detection method that monitors the system from different angles. Specifically, the idea of monitoring the random-access memory (RAM) for the sensitive data has great potential as a solution that protects the system from attacks coming from nearly all channels. Intuitively, any computer program needs to load the stored sensitive data into the memory before any processing activity can be conducted. In other words, RAM is a sole gateway for the processing of computer data, including the sensitive data that needs protection.

The monitoring of sensitive data in the memory is however a challenging task, since the size of the sensitive data to be monitored (i.e. patterns) could be large, in some cases even larger than the size of the main memory itself. Existing text-searching methods are designed for using a small-size patterns to search against a larger-size database, e.g. Smith–Waterman [21], Aho–Corasick [22], and regular expressions [23]. Unfortunately, existing string-matching methods

are not designed for the monitoring of a large set of sensitive documents. In the context of this book, the searching patterns could be bigger than the size of the memory. Therefore, a new search approach will be provided here to allow a large database of searching patterns to be summarized/compressed into a smaller representation. This will enable more sensitive files to be monitored in real time, particularly in the memory.

There are several challenges to be addressed to build such a system. Firstly, sensitive data could include various information and therefore not only limited to personal information. It can also include organizational information such as hospital patient medical records, government CI sites detail, corporate intellectual property, and organization internal information. Sensitive data can be also classified into various formats, for example, a corpus of text documents (e.g. ASCII, Unicode UTF-8/16/32) and database files (e.g. comma separated values, spreadsheet, and JavaScript Object Notation). Dealing with different format of sensitive data, indeed, requires different monitoring methods. As the type of data search is directly related to the design of a monitoring algorithm, the discussion here will first focus on sensitive data that is a corpus of text documents, giving that the text is pre-decoded (i.e. using standard Unicode) from the raw memory data/snapshot.

1.1.4 Temporal Data Exfiltration Methods

Much research work focused on monitoring sensitive data using various methods, such as creating and tracking data signatures of sensitive documents [24, 25], tagging sensitive files with additional meta-data [26, 27], or classifying/detecting sensitive documents using advanced ML methods [12, 28, 29]). However, sophisticated espionage might use highly advanced methods to hide malicious software from being detected while still be able to spy on a victim.

Temporal-based data exfiltration is one of the hardest attack detection problems and remains one of the most challenging threats [11], because a spy software keeps collecting small fragments of sensitive information over an interval of time (hence, a temporal attack). Consequently, a malicious process could avoid being detected by the security system. Only a few solutions have dealt with temporal data exfiltration attacks by, for example, monitoring the network traffic over a period of time [12]. Interestingly, the hierarchical temporal memory (HTM) technology, which mimics the behavior of neocortex cells in the human brain [30], has been able to recognize a pattern in temporal input data. This technology has not yet been investigated fully to determine whether it can be applied to memory-based data leakage patterns to detect temporal-based data exfiltration. At this stage, related work attempting to detect temporal data exfiltration attacks cannot be found, especially by monitoring the RAM's sensitive data, which is also the main focus of this book.

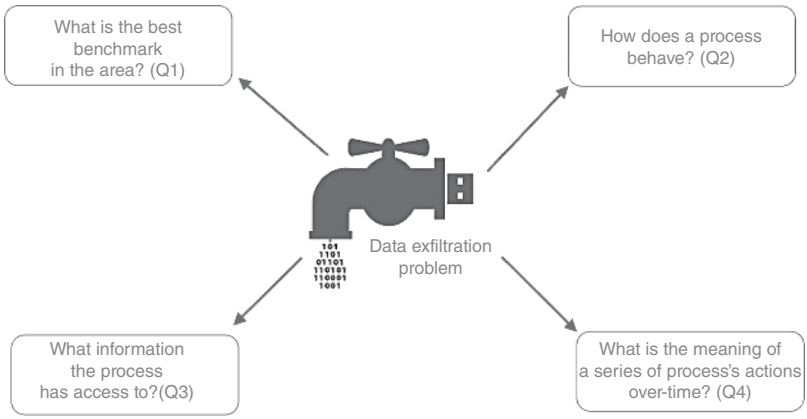

Figure 1.1 Overview of data exfiltration problems.

1.2 Important Questions

This book comprehensively addresses the data exfiltration issues by answering the following four important questions, as depicted in Figure 1.1.

- What is a survey and benchmark of existing methods?
- How does a process behave or how to detect the malicious behavior of the process?
- What information the process has access to or how to efficiently detect if any process has access to the sensitive data?
- What is the meaning of series of process's actions over time or how to efficiently detect the temporal data exfiltration?

A detailed description of each data exfiltration problem is given in the following section.

(Q1) What are the limitations of existing supervised learning methods for data leakage prevention (DLP) and/or intrusion detection system (IDS) for industrial control systems (ICSs)?
To address the first problem, an extensive literature survey is conducted to find and investigate the state-of-the-art technologies that have been used in data leakage and/or IDSs. Over the past decade, various ML methods have been used to train artificial intelligence to detect abnormal behaviors, including data exfiltration. The supervised-based machine learning (supervised ML) methods are more common in the anomaly detection system than the unsupervised methods (known as clustering). This is because the patterns of the attack vector are known. To contribute to the current knowledge regarding supervised ML for DLP/intrusion detection, a survey is conducted to examine existing supervised

ML technologies that were proposed for anomaly detection systems, including data exfiltration and other cyberattacks. Specifically, one will need to: (i) categorize existing Supervised-ML based on detection methods and auditing material; (ii) identify system-specific requirements of supervised ML; and (iii) benchmark supervised ML methods from a holistic perspective.

(Q2) How to effectively detect data-stealing behaviors from the sequence of API instructions of a process?
Detecting the malicious activities of a specific running process is a challenging task, particularly when we need to differentiate between patterns of malicious and legitimate processes using a sequence of API calls. Unlike text or image processing where all features are observed and extracted by examining the whole picture, a running process can act differently depending on the underlying stages or task it is doing. Therefore, the program behavior, which changes expeditiously, could easily hide fractions of malicious instructions over a brief period of execution time. Hence, it is very challenging for the malware detection technology to detect the malicious behavior that is carried out within seconds or milliseconds. Therefore, the methodology tailored for detecting data exfiltration behavior will require a new method and algorithm to address this problem. This will particularly look at the feature extraction problem, in particular, for the long sequence of API calls that indicate a stream of ongoing activities currently being executing by the program.

(Q3) How to efficiently inspect the sensitive information on the random-access memory space to which a process has access?
This problem is related to the monitoring of data in the memory space of a process in order to detect the presence of sensitive information. In a computer system, data needs to be loaded into the RAM before it can be processed or transferred; hence, a malicious software will not be able to hide from the detection system if it is accessing sensitive information. Thing et al. [10] suggest that attackers have even tried to evade the memory forensic tool itself to prevent their malicious software from being detected. Even though data in the RAM is changing all the time, the unused data will stay in the RAM until the address space is recycled by the operating system. For instance, they capture up to 75–100% of conversation on instant messages by analyzing the mobile phone memory. On the other hand, Wang et al. [31] identified up to 98.6% of the kernel rootkit by using memory images. The monitoring of sensitive data on the RAM will be a more robust way of mitigating various data exfiltration incidents. However, monitoring memory-based data is challenging in terms of scalability issue, especially when there is a large number of sensitive documents. The robustness of the method is also an important issue, particularly when the monitoring system directly accesses data from the program's memory.

(Q4) How to efficiently mitigate with a sophisticated temporal-based data exfiltration?
The last problem relates to a particularly sophisticated attack, namely temporal-based data exfiltration. This threat might occur when the target information is highly protected, such as military or business databases. Despite the tight surveillance, the hacking bot steals sensitive documents from the compromised machine by splitting them into chunks. The bot then exports pieces of the sensitive data one by one through single or multiple data leakage channels. Even though the small fraction of sensitive data can still be detected, the matching probability could be lower than the detection threshold and considered as a false negative (i.e. not detected). This enables attackers to evade the detection system by collecting several pieces of sensitive data over a period. The hackers then reconstruct the pieces of data to obtain the complete original sensitive file. Furthermore, the delay of data exfiltration action could also return a false positive (i.e. false alarm) when the small part of sensitive data is detected repeatedly. For instance, if a fraction from the sensitive text file is found twice, it could be the same instance that has been detected previously or the new data that has just been loaded into the memory. Unfortunately, the current DLP technology is not adequate in preventing temporal data exfiltration.

1.3 Book Scope

The main purpose of this book is to address the problems stated in Section 1.2: exhaustive survey (Q1) and specialized data exfiltration detection methods (Q2–Q4). For (Q1), state-of-the-art supervised ML methods have been surveyed here and describe the intrusion detection methods/systems proposed in the literature for over the past decade. Existing supervised-ML-based methods are categorized and evaluated on specific requirements, namely the requirements of CI systems (i.e. SCADA), as such systems are those that are used in our daily life (i.e. critical) and therefore need to be fully, strongly, and properly protected. Ultimately, such an assessment and evaluation of (critical) systems will help the reader understand the key research challenges and ideas about the use of supervised ML methods in IDSs, in particular, for critical systems. On the other hand, Q2–Q4 relate to data exfiltration detection method from three different perspectives, namely behavior-based (Q2), memory-based (Q3), and temporal-based (Q4). Since they are all technical methods that share similarities with existing data leakage/intrusion/malware detection methods, the scope of the Q2–Q4 questions is summarized by comparing them with the mentioned methods based on the following categorization criteria: (i) the detection source, (ii) detection method, and (iii) input characteristics. Figure 1.2 depicts such a categorization.

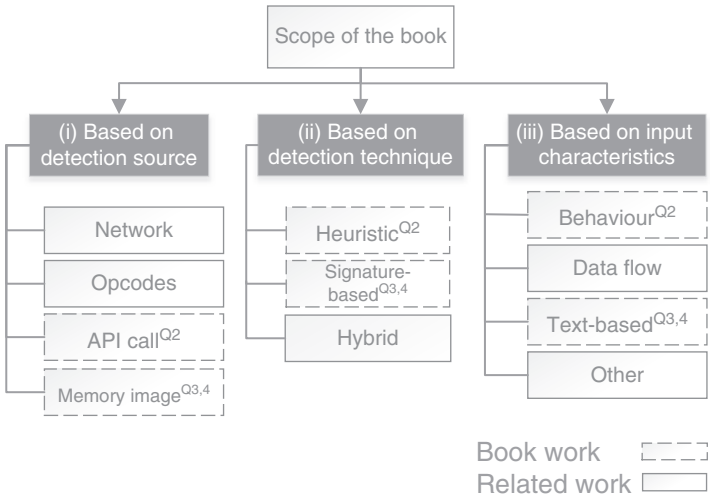

Figure 1.2 Scope of the work for Q2–Q4 compared to related work based on three different listed aspects.

To address RQ2, the goal is to detect the program's suspicious behavior, in particular, the data scraping and stealing actions. The sequence of API calls is chosen as the detection source because it indicates the ongoing activities of a program. This is known as a heuristic approach and is very flexible compared to the static method. In the behavior-based model, the number of command sequences issued by a program to steal sensitive data could be very large, making it impossible to find all possible signatures of the malicious command sequences. Also, the signature-based approach suffers from zero-day attack, or variant of polymorphic malware where their signatures are unknown. Indeed, the delay in generating the new signature consumes a lot of time and budget. Therefore, the heuristic-based approach is being developed, facilitated by artificial intelligence and data mining technology.

RQ3 and RQ4, on the other hand, refer to signature-based methods. The explanation is as follows: to detect a sophisticated malware or a corrupted employee exporting sensitive data using ordinary software, the sensitive data will be monitored regardless of whether or not the program is classified as malicious. The main advantage of the signature-based method is that it has a very high true positive rate compared to the heuristic-based method. Here, the static signature or fingerprint is used instead of the generic pattern used by the heuristic approach. However, because sensitive data is text-based (i.e. unformatted text), the signature-based method is not scalable. For instance, when the size of the signature database is large, the memory footprint and runtime could be affected.

Moreover, the signature-based method becomes less effective when the input data contains too much noise. In this case, RQ3 and RQ4, which relate to the monitoring of data leakage by examining the physical memory, will definitely need to deal with noise from the RAM's data extracted (Chapter 7). These scalability and robustness issues are primary targets of the research addressing RQ3 and RQ4.

1.4 Book Summary

By addressing the four problems, this book describes new methods that address various limitations of current state-of-the-art methods that prevent them from detecting data exfiltration incidents from the four different aspects mentioned earlier.

1.4.1 Data Security Threats

Data security refers to preserving data against unwanted access, corruption, or theft across all stages of its life cycle. This also refers to covering every aspect of security, including the logical security of applications, administrative and access controls, and the physical security of hardware and storage devices. Specific policies and procedures are used to guarantee data security. However, data can be still vulnerable to various attacks and threats. Data security threats refer to activities that have the potential to compromise the confidentiality, integrity, and availability of the data and have therefore considerable damage and harm to organizations. *Confidentiality* refers to ensuring that the data is kept secret or private, and accessing it must be controlled to prevent any malicious or accidental unauthorized sharing of data. *Integrity* refers to ensuring that the data is reliable and immutable. *Availability* means that the data should be available for the users in the entire life cycle of the data. Chapter 3 aims to cover data security threats in detail, and more specifically it will cover the most known cyberattacks, e.g. Malware, denial of service (DoS), SQL Injection, Emotet (malspam), Man in the Middle (MITM), Password Attacks, and Social Engineering & Phishing. This will provide readers with a good understanding of existing cybersecurity threats.

1.4.2 Use Cases

Studying data security can be seen as "boring" (and thus not convincing) when the focus is only on the theoretical aspects and concepts cybersecurity. This study will hopefully convince readers about the importance of properly understanding as well as dealing with cyberattacks through the description of some real-world

use cases. Thus, to make this book more interesting to readers, Chapter 4 will study several use cases of data leakage attacks that occurred over the last three years across all the continents. More specifically, it first introduces the cyberattack types and categorized them based on the objectives and attackers intend to accomplish. Some of these cyberattacks (e.g. ransomware, server access, business email compromise (BEC), data theft, credential harvesting, remote administration Trojan (RAT), misconfiguration, and malicious insider) are all grouped based on attackers ultimate goals. Ransomware is considered as the most common attack in the recent three years. After ransomware, the server access attacks are ranked the most frequent common ones. They occur when an attacker gains unauthorized access to a server. The third most frequently used attack is BEC. After discussing the cyberattack types in detail, the chapter explains the initial infection vectors, i.e. the method through which a network is breached and compromised. It is important to note that the cyberattack types and initial infection vectors are two different concepts and not be confused. Phishing, stealing credentials, and exploiting vulnerabilities are considered initial infections vectors. Reports show that in 2021, phishing alone achieved the highest record of 222,127. The effect of this attack vector can be reduced by monitoring suspicious connections.

1.4.3 Survey

After covering all the necessary background in the early chapters of the book, a survey of various supervised ML methods is thoroughly conducted to provide information how such methods are used in intrusion detection and DLP systems. This review focuses on specific applications, namely CIs (i.e. ICSs such as electrical/power/water systems), as these need to be protected from major disruptions. It addresses Q1, and over a 100 of peer-reviewed articles were reviewed, which gave an insight into current research trends regarding supervised ML methods to implement anomaly detection systems (i.e. DLP and IDS). This study illustrates the development of such systems from industry perspectives and provides a comprehensive study of supervised ML methods for SCADA-based IDS systems using specific criteria and properties. A framework is described to categorize various supervised ML methods and made qualitative and quantitative comparisons of various state-of-the-art research methods to identify the directions of research that target different data auditing sources and attacking methods. Additional issues and challenges are discussed for industrial-based IDS systems using supervised learning methods and illustrated the trends in developing such systems to identify the future directions for the development of new algorithms and to guide the selection of algorithms for industrial IDS systems. The survey not only provides a framework enabling administrators to choose the right ML method for a prevention system, but it also provides a comparison of various supervised ML algorithms.

1.4.4 Sub-Curve HMM

The work related to Q2 has led to the design of a method that accurately detects data exfiltration behaviors that occur at any time a process is executing. This method is called Sub-Curve HMM, and it is an innovative feature extraction method based on HMM, which makes use of API call sequences by extracting the subcontained behavior from a long API call sequence. Sub-Curve HMM is probably one of the early attempts to extract the subcontained pattern from a long API call sequence to detect data exfiltration malware. This enables small pieces of malicious activities contained in the long API call sequence to be detected. The limitations of current detection solutions are also considered here, especially in terms of a long API call sequence. Compared to existing methods, Sub-Curve HMM outperforms baseline methods across several datasets with various average API sequence lengths. The experimental results confirm the ability of Sub-Curve HMM to match interesting behaviors from subsequences of all observed activities. Unlike the previous works that focused on using all information gathered from executable binaries, Sub-Curve HMM requires only parts of detection activities. Hence, this method can be applied in a real-time context where it is not possible to gather all information from the executable binary. In addition, to prevent data exfiltration incident, Sub-Curve HMM can be used to monitor only those programs that are accessing sensitive information.

1.4.5 Fast Lookup Bag-of-Words (FBoW)

To address Q3, an efficient way to monitor sensitive data in real time on a RAM is described. This is probably the first attempt to challenge existing data leakage detection methods by monitoring sensitive data in the RAM. The method described here, called Fast lookup Bag-of-Words (FBoW), is an approximate multi-pattern matching method for text documents. FBoW addresses several aspects in matching the RAM's textual sensitive data, such as scalability and noise. When there is a large number of sensitive documents in a database, the problem is that existing pattern-matching methods, e.g. Aho–Corasick [22] and Smith–Waterman [21], cannot be used, and regular expression methods, e.g. [23], do not meet the scalability requirements of RAM data files. The second issue is that the text extracted from memory contains noise. For instance, the noise is produced by decoding the non-textual elements in the memory to extra characters or re-ordering the content according to memory paging [32]. This issue poses a serious practical impediment to use exact matching algorithms, e.g. Knuth–Morris–Pratt [33], Boyer–Moore [34], Aho–Corasick [22], or Commentz–Walter [35]. Although approximate matching is more robust in this context, approximate matching methods are either runtime-inefficient (e.g. Smith–Waterman [21]) or unable to accurately identify free-form text (e.g. using

regular expression [23]). By addressing Q3, this book will provide the followings: (i) an innovative pattern-matching algorithm for multiple corpora or a large corpus, that is, memory- and runtime-efficient; (ii) a customizable approximate search algorithm that allows a user to fine-tune a trade-off between scalability (i.e. memory footprint and processing time) and the detection accuracy; and (iii) a series of experimental evaluations that benchmark single and multiple pattern-matching algorithms (e.g. inference of regular expressions [RegEx] [23], Smith Waterman [21], and Aho–Corasick [22]) for both exact and approximate solutions in the context of matching sensitive data in three different formats: keywords only, whole text files, and sensitive data in the physical RAM.

FBoW is efficient in the way it monitors RAM sensitive data in real time. It addresses issues in matching the RAM's textual sensitive data as follows. The first aspect is the scalability problem when the database of sensitive data contains many documents, which prevents existing pattern-matching methods (i.e. Aho–Corasick [22], Smith–Waterman [21], and using regular expression [23]) from being scale enough to match the RAM content with the files in the database. The second issue is that the text content extracted from memory contains noise, for instance, the noise from decoding the non-textual elements in the memory to extra characters or re-ordering the content as per memory paging [32]. This issue raises a serious practical impediment to use the exact matching algorithms, e.g. Knuth–Morris–Pratt [33], Boyer–Moore [34], Aho–Corasick [22], or Commentz–Walter [35]. Although approximate matching is more robust in this context, the approximate matching methods are either runtime-inefficient (e.g. Smith–Waterman [21]) or unable to accurately identify free-form text (e.g. using regular expression [23]). RQ3's details can be summarized as follows: (i) an innovative pattern-matching algorithm for multiple long text corpus that is memory and runtime-efficient; (ii) a customizable approximate search algorithm that allows a user to finetune a trade-off between scalability (i.e. memory footprint and processing time) and the detection accuracy; and (iii) a series of experimental evaluations that benchmark single and multiple pattern-matching algorithms (e.g. inference of RegEx [23], Smith–Waterman [21], and Aho–Corasick [22]) for both exact and approximate solutions in the context of matching sensitive data in three different formats: only keywords, whole text files, and sensitive data in the physical RAM.

1.4.6 Temporary Memory Bag-of-Words (TMBoW)

Q4's will deal with one of the long discovered sophisticated data exfiltration threats [11]: temporal data exfiltration; yet not many research works have proposed solution to this problem. This book will describe TMBoW – Temporary

Memory Bag-of Words – to capture time-delay data exfiltration activities. The design of TMBoW is based on bag-of-words (BoW) and sparse distributed representation (SDR) to represent textual sensitive documents. TMBoW allows a large number of sensitive documents to be added to the detection model. This is because the SDR and BoW shrink unique words that appear in the sensitive database to only one-bit-per-word frequency. The sensitive data-matching process is performed via multiple bitwise operations. Its design enables parallel programming to be done on a modern system with a multicore processor. As a result, TMBoW can more quickly match sensitive data that appear in the memory with of a multicore CPU system. Hence, both memory and runtime efficiency can be achieved simultaneously. Put simply, the feature of temporary memory and the multiple time-step detection framework enable temporal or time-delay data exfiltration to be detected, even though the operation has been extended over a long period of time. Furthermore, the new data structure that contains the sensitive database with BoW and SDR representation allows noisy data from the RAM to be monitored efficiently with a very low probability of returning a false positive (i.e. false alarm). To summarize, by addressing Q4, this book will describe the followings: (i) a method to detect the leakage of the sensitive data in the memory using time-delay channel, (ii) the scalability issue of the multi-pattern-matching algorithm for text-based sensitive data will be discussed, and (iii) the issue of noises associated with extracting strings from the memory snapshot will be addressed.

1.5 Book Structure

Figure 1.3 depicts the structure of this book, which consists of nine chapters. The arrow depicts the flow of the research steps. The chapters with main contributions are highlighted in the dotted trapezoid. Chapter motivates the issues as well as the methods described in the book. Chapter 2 provides background knowledge on various cybersecurity threats associated with the data leakage issues. Details of the data security threats are provided in Chapters 3 and 4 reports on recent real-world data leakage and top cybersecurity attacks from 2020 to 2021 in all five continents. A comprehensive literature survey of existing supervised machine learning technologies is presented in Chapter 5. Afterward, three data exfiltration detection approaches are presented. Chapter 6 discusses the approach based on software behavior. Chapter 7 discusses a novel approach based on monitoring the physical memory. Finally, a solution to the advanced temporal data-stealing threat is described in Chapter 8. Chapter 9 concludes the book.

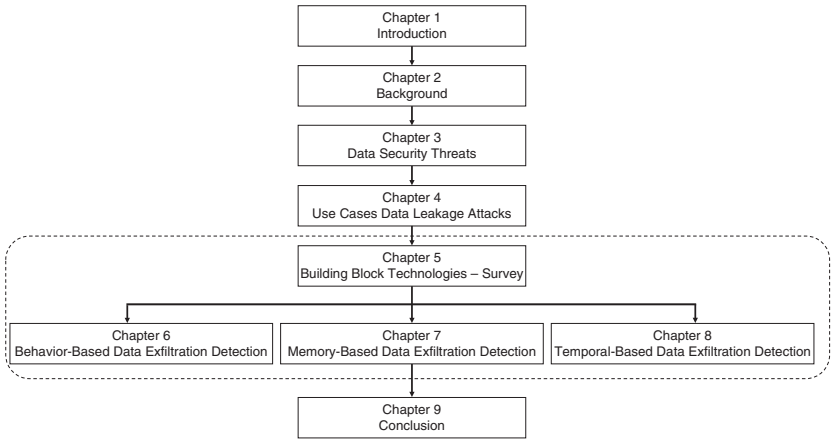

Figure 1.3 Book structure.

References

1 Faheem Ullah, Matthew Edwards, Rajiv Ramdhany, Ruzanna Chitchyan, M. Ali Babar, and Awais Rashid. Data exfiltration: a review of external attack vectors and countermeasures. *Elsevier Journal of Network and Computer Applications*, 101:18–54, 2018.

2 Scott Wilson, Susan Maphis, Helga George, Rebecca Turley, Steven Carver, and Hannah Coffman. Top 5 Most Technically Advanced Hacking Attacks of All Time. https://www.cybersecurityeducationguides.org/2017/11/top-5-most-technically-advanced-hacking-attacks-of-all-time/, 2017.

3 Verizon. 2020 Data Breach Investigations Report. https://enterprise.verizon.com/resources/reports/2020-data-breach-investigations-report.pdf, 2020.

4 Kevin Savage, Peter Coogan, and Hon Lau. The evolution of ransomware. Technical report, Symantec Corp., Mountain View, CA, USA. https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-evolution-of-ransomware.pdf, 2015.

5 National Cyber Security Centre. Russian state-sponsored cyber actors targeting network infrastructure devices. https://www.ncsc.gov.uk/pdfs/blog-post/malicious-russian-cyber-activity-what-does-it-mean-small-organisations.pdf, 2018.

6 Dan Conifer. Defence contractor's computer system hacked, files stolen, cyber security report reveals. https://www.abc.net.au/news/2017-10-10/defence-contractors-files-stolen-in-hacking:-security-report/9032290, October 2017.

7 Nicolas Falliere, Liam O. Murchu, and Eric Chien. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response*, 5(6):29, 2011.

8 Zakir Durumeric, Frank Li, James Kasten, Johanna Amann, Jethro Beekman, Mathias Payer, Nicolas Weaver, David Adrian, Vern Paxson, Michael Bailey,

et al. The matter of heartbleed. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 475–488, 2014.

9 Hal Berghel. The code red worm. *Communications of the ACM (CACM)*, 44(12):15–19, 2001.

10 Vrizlynn L. L. Thing, Kian-Yong Ng, and Ee-Chien Chang. Live memory forensics of mobile phones. *Digital Investigation*, 7:S74–S82, 2010.

11 Annarita Giani, Vincent H. Berk, and George V. Cybenko. Data exfiltration and covert channels. In *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense V*, volume 6201, page 620103. International Society for Optics and Photonics, 2006.

12 Brian A. Powell. Malicious Overtones: hunting data theft in the frequency domain with one-class learning. *ACM Transactions on Privacy and Security (TOPS)*, 22(4):1–34, 2019.

13 Bonnie Zhu and Shankar S. Sastry. SCADA-specific intrusion detection/prevention systems: a survey and taxonomy. In *Proceedings of the 1st Workshop on Secure Control Systems (SCS)*, volume 11. Berkeley University of California, 2010.

14 Robert Mitchell and Ing-Ray Chen. A survey of intrusion detection techniques for cyber-physical systems. *ACM Computing Surveys*, 46(4):55, 2014.

15 Hannes Holm, Martin Karresand, Arne Vidström, and Erik Westring. A survey of industrial control system testbeds. In *Proceedings of the 20th Nordic Conference on Secure IT Systems (NordSec 2015)*, pages 11–26. Springer International Publishing, October 2015.

16 Check Point Research. H2 2017 Global Threat Intelligence Trends Report. Technical report, Check Point Technologies Ltd. https://research.checkpoint.com/h2-2017-global-threat-intelligence-trends-report, 2018.

17 Symantec Corporation. Symantec internet security threat report 2017. Technical report, Symantec Corp. https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf, 2017.

18 Chinmayee Annachhatre, Thomas H. Austin, and Mark Stamp. Hidden Markov models for malware classification. *Springer Journal of Computer Virology and Hacking Techniques*, 11(2):59–73, 2015.

19 Gerardo Canfora, Francesco Mercaldo, and Corrado Aaron Visaggio. An HMM and structural entropy based detector for android malware: an empirical study. *Elsevier Journal on Computers & Security*, 61:1–18, 2016.

20 Anusha Damodaran, Fabio Di Troia, Corrado Aaron Visaggio, Thomas H. Austin, and Mark Stamp. A comparison of static, dynamic, and hybrid analysis for malware detection. *Springer Journal of Computer Virology and Hacking Techniques*, 13(1):1–12, 2017.

21 William R. Pearson. Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith–Waterman and FASTA algorithms. *Elsevier Journal of Genomics*, 11(3):635–650, 1991.

22 Alfred V. Aho and Margaret J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM (CACM)*, 18(6):333–340, 1975.

23 Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. Inference of regular expressions for text extraction from examples. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 28(5):1217–1230, 2016.

24 Yuri Shapira, Bracha Shapira, and Asaf Shabtai. Content-based data leakage detection using extended fingerprinting. *arXiv preprint arXiv:1302.2028*, 2013.

25 Hao Zhuang, Rameez Rahman, Pan Hui, and Karl Aberer. Optimizing information leakage in multicloud storage services. *IEEE Transactions on Cloud Computing*, 8(4):975–988, 2018.

26 Hao Li, Zewu Peng, Xinyao Feng, and Hongxia Ma. Leakage prevention method for unstructured data based on classification. In *Proceedings of the Springer International Conference on Applications and Techniques in Information Security*, pages 337–343, 2015.

27 Chunwang Zhang, Ee-Chien Chang, and Roland H. C. Yap. Tagged-MapReduce: A general framework for secure computing with mixed-sensitivity data on hybrid clouds. In *Proceedings of the 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 31–40, 2014.

28 Sultan Alneyadi, Elankayer Sithirasenan, and Vallipuram Muthukkumarasamy. Detecting data semantic: a data leakage prevention approach. In *Proceedings of the IEEE Trustcom/BigDataSE/ISPA International Conference*, volume 1, pages 910–917, 2015.

29 Shuaiji Li, Tao Huang, Zhiwei Qin, Fanfang Zhang, and Yinhong Chang. Domain Generation Algorithms detection through deep neural network and ensemble. In *Proceedings of the World Wide Web Conference (WWW)*, pages 189–196, 2019.

30 J. Hawkings and S. Blakeslee. On intelligence: how a new understanding of the brain will lead to the creation of truly intelligent machines. *An Owl Book*, Henry Holt and Company, New York, 2004.

31 Xiao Wang, Jianbiao Zhang, Ai Zhang, and Jinchang Ren. TKRD: Trusted kernel rootkit detection for cybersecurity of VMs based on machine learning and memory forensic analysis. *Mathematical Biosciences and Engineering*, 16(4):2650–2667, 2019.

32 Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. *Operating System Concepts*. John Wiley & Sons, 2006.

33 Donald E. Knuth, James H. Morris, Jr., and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350, 1977.

34 Robert S. Boyer and J. Strother Moore. A fast string searching algorithm. *Communications of the ACM (CACM)*, 20(10):762–772, 1977.

35 Beate Commentz-Walter. A string matching algorithm fast on the average. In *Springer International Colloquium on Automata, Languages, and Programming*, pages 118–132, 1979.