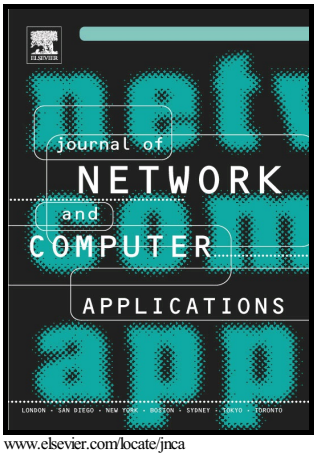
Data Exfiltration: A Review of External Attack Vectors and Countermeasures

Faheem Ullah, Matthew Edwards, Rajiv Ramdhany, Ruzanna Chitchyan, M. Ali Babar, Awais Rashid



www.elsevier.com/locate/jnca

PII: S1084-8045(17)30356-9
DOI: https://doi.org/10.1016/j.jnca.2017.10.016
Reference: YJNCA1996

To appear in: *Journal of Network and Computer Applications*

Received date: 8 August 2017
Revised date: 18 October 2017
Accepted date: 28 October 2017

Cite this article as: Faheem Ullah, Matthew Edwards, Rajiv Ramdhany, Ruzanna Chitchyan, M. Ali Babar and Awais Rashid, Data Exfiltration: A Review of External Attack Vectors and Countermeasures, *Journal of Network and Computer Applications,* https://doi.org/10.1016/j.jnca.2017.10.016

Data Exfiltration: A Review of External Attack Vectors and Countermeasures

Faheem Ullah[1], Matthew Edwards[2], Rajiv Ramdhany[2], Ruzanna Chitchyan[3], M. Ali Babar[1, 4], Awais Rashid[2]

[1]University of Adelaide, Australia & [4]IT University of Copenhagen, Denmark

[2]Security Lancaster, School of Computing and Communications, Lancaster University, UK,

[3]Department of Computer Science, University of Leicester, UK

ABSTRACT

Context: One of the main targets of cyber-attacks is data exfiltration, which is the leakage of sensitive or private data to an unauthorized entity. Data exfiltration can be perpetrated by an outsider or an insider of an organization. Given the increasing number of data exfiltration incidents, a large number of data exfiltration countermeasures have been developed. These countermeasures aim to detect, prevent, or investigate exfiltration of sensitive or private data. With the growing interest in data exfiltration, it is important to review data exfiltration attack vectors and countermeasures to support future research in this field. *Objective:* This paper is aimed at identifying and critically analysing data exfiltration attack vectors and countermeasures for reporting the status of the art and determining gaps for future research. *Method:* We have followed a structured process for selecting 108 papers from seven publication databases. Thematic analysis method has been applied to analyse the extracted data from the reviewed papers. *Results:* We have developed a classification of (1) data exfiltration attack vectors used by external attackers and (2) the countermeasures in the face of external attacks. We have mapped the countermeasures to attack vectors. Furthermore, we have explored the applicability of various countermeasures for different states of data (i.e., in use, in transit, or at rest). *Conclusion:* This review has revealed that (a) most of the state of the art is focussed on preventive and detective countermeasures and significant research is required on developing investigative countermeasures that are equally important; (b) Several data exfiltration countermeasures are not able to respond in real-time, which specifies that research efforts need to be invested to enable them to respond in real-time (c) A number of data exfiltration countermeasures do not take privacy and ethical concerns into consideration, which may become an obstacle in their full adoption (d) Existing research is primarily focussed on protecting data in 'in use' state, therefore, future research needs to be directed towards securing data in 'in rest' and 'in transit' states (e) There is no standard or framework for evaluation of data exfiltration countermeasures. We assert the need for developing such an evaluation framework.

Keywords: Data Exfiltration, Data Leakage, Data Theft, Data Breach, External Attack Vector, Countermeasure

1. Introduction

Data theft (formally referred to as data exfiltration) is one of the main motivators for cyber-attacks irrespective of whether carried out by organised crime, commercial competitors, state actors or even "hacktivists". Preventing data exfiltration is increasingly becoming a challenging task due to two main reasons. First, over the span of last few years, cyber-crime has transformed from an individual's act to an organizational act. This transformation has provided attackers (or often called hackers) with high budget, resources, and sophistication to become more professional in data exfiltration. Second, the existing data infrastructure contains several means (as shown in Fig. 1a) that are originally designed for the legitimate exchange of data but can be used for data exfiltration.

Fig. 1b shows one scenario of how these legitimate means can be leveraged for data exfiltration. *Stage-1*: The attacker researches the available means within an enterprise to find some weaknesses that can be exploited to exfiltrate data. *Stage-2*: Once the weakness has been identified, the attacker launches the attack to exploit the identified weakness. The attack can either be network-based or physical-based. In a network-based attack, an attacker may use different techniques such as phishing email, malware, or some kind of injections. In a physical attack, an attacker may leverage techniques like copying data to a removable device or even get hold on some printed documents. *Stage-3*: Once the attacker gains the required access to the sensitive data, the process

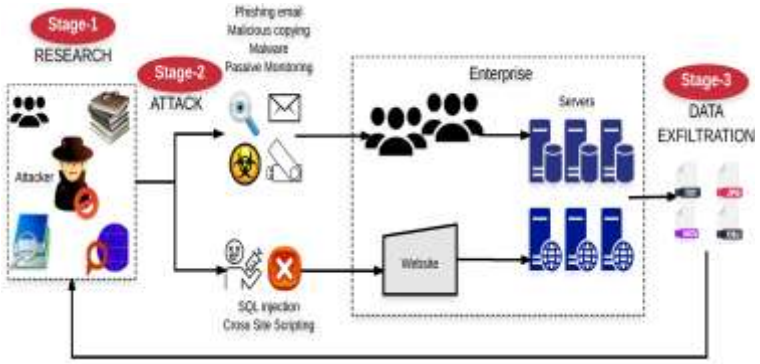


Fig. 1a: Some of the means of Data Exfiltration Fig. 1b: Data Exfiltration Scenario

of data exfiltration starts as shown in Fig. 1b. Due to the sophistication of attacks and several available means, there has been a number of data exfiltration incidents reported recently. According to ITRC Breach Report[1], until 3rd October 2017, there have been 1056 data exfiltration incidents in 2017. A popular incident reported in June 2017 was the leakage of data of 200 million American voters, which was compiled for Republican National Committee. The data contained personal information of voters that included name, home address, phone number, date of birth, and voter registration details. The leak occurred due to misconfiguration of the database holding the data. *Equifax* and *Deloitte* are two of the top companies, which have recently fallen victims to major cyber security attacks that have left millions of people's private and business data exfiltrated[2][3]. Both of the companies were entrusted with the responsibilities of safeguarding that very data that got stolen with known and unknown personal, professional, and business implications for private citizens as well as businesses. ITRC Breach Report has several other incidents among which the most noticeable are shown in Fig. 2. Recently, Verizon released 2017 Data Breach Investigations Report[4] that revealed some interesting aspects of this growing concern. According to this report, 75% of data exfiltration attacks were perpetrated by external attackers (i.e., hackers) while 25% of the attacks involved actors within the victim organization. This report also reveals that among all the data leaks, about 24% occurred in the financial sector, about 15% occurred in the healthcare sector, about 15% occurred in retail and accommodation sector, and about 12% occurred in public sector entities.

As evident from the above-mentioned statistics, such proliferation of data exfiltration is becoming a serious concern for business and government organizations. For businesses, data exfiltration may cause huge financial and reputational damage by disclosing trade secrets, information about future projects, and customers' profiles to competitors. For governments, the consequences of data exfiltration can be even more severe when national secrets or information about political settlements fall into the hands of adversaries. Driven by the need to address such a serious concern, security experts develop various security systems that include but not limited to Intrusion Detection System (IDS), Intrusion Prevention System (IPS), firewall, and Security Information and Event Management (SIEM). However, these systems fall short in dealing with data exfiltration due to unstructured and constantly evolving nature of data. To overcome this limitation, security experts and researchers come up with the idea of data exfiltration countermeasures, which specifically aim to detect, prevent, or investigate data exfiltration. Unlike traditional security systems (e.g., IDS, IPS, and firewall), which

[1] http://www.idtheftcenter.org/images/breach/2017Breaches/ITRCBreachReport2017.pdf
[2] https://www.theguardian.com/us-news/2017/sep/07/equifax-credit-breach-hack-social-security
[3] https://www.theguardian.com/business/2017/sep/25/deloitte-hit-by-cyber-attack-revealing-clients-secret-emails
[4] https://www.ictsecuritymagazine.com/wp-content/uploads/2017-Data-Breach-Investigations-Report.pdf

are supposed to act when an attacker tries to enter the enterprise, data exfiltration countermeasures are expected to work when an attacker is on its way back from the enterprise with the sensitive data.

We assert that there is an important need of providing an extensive overview and in-depth analysis of the available literature to connect the knowledge about the current state of the art on this topic. We have carried out an extensive literature review on data exfiltration to derive and use a classification of external attack vectors (which accounts for 75% of total data exfiltration attacks) and countermeasures. We have systematically analysed the countermeasures in terms of their contributions and limitations. We have reviewed the countermeasures based on their goals, that is, whether a particular countermeasure is designed to prevent, detect or investigate data exfiltration. We have also analysed the available countermeasures based on the state of data that can either be in

Fig. 2: Most Noticeable Data Exfiltration incidents in 2017

in use, in transit or at rest. Furthermore, we have provided a mapping of the attack vectors to the selected countermeasures to help understand which countermeasures are applicable against which attack vectors. This literature review has enabled us to identify the areas where immediate research is required to cope with the future challenges. Our review has attempted to answer the following Research Questions (RQs):

RQ1: What are the attack vectors used by remote attackers to steal data from an individual or an organization?

RQ2: What are the countermeasures for preventing, detecting and investigating data exfiltration?

RQ3: What are the challenges in preventing, detecting and investigating data exfiltration?

As our aim is to survey both attack vectors and a broad set of countermeasures — preventive, detective and investigative — we refer to this topic as "data exfiltration" rather than "data leakage prevention", which implies a specific focus on preventive measures. Both academia and industry refer to data exfiltration with other names too such as data theft, data leakage, data breach, or data stealing, therefore, these terms are used interchangeably in this paper. The terms external attacker, remote attacker, and hacker are also used interchangeably.

The rest of this paper is structured as follows. We begin with a discussion of the related work on existing surveys and compare them with our survey to justify the contributions of our survey in Section 2. Section 3 describes the methodology adopted for conducting this survey. Section 4 highlights the external attack vectors identified from the studied literature to address RQ1. Section 5 follows this by systematising the body of work on countermeasures to address RQ2. Following this, we present some of the future challenges for data exfiltration research in Section 6 to address RQ3. In section 7, we outline the limitations of our work. Finally, Section 8 concludes the paper.

2. Related work

2.1. Existing Literature Reviews

Data exfiltration is a highly active research area where literature has been reviewed from different perspectives. During our review process, we found five review papers on data exfiltration. We briefly summarize the key aspects of these review papers.

Alneyadi et al. [2] report a review on Data Leakage Prevention Systems (DLPs) challenges and countermeasures. Based on the analysis of 25 countermeasures, the authors identify some potential areas for future research that include contextual analysis, content analysis, term weighing techniques, internal misuse, and smartphone security. Shabtai et al. [3] present a review on data leakage detection and prevention. They propose a taxonomy for data leakage prevention solutions, discuss various data leakage prevention solutions in industry and academia, highlight the causes, locations, timings and motivators for data leakage, and at the end identify future trends for research in DLP. The review of Shabtai et al. includes only 37 DLPs from academic literature as this review is primarily focused on industrial DLPs. Raman et al. [4] literature survey describes the problem of DLP and highlights some basic approaches from the literature. They identify encryption, access control, semantic gap in DLP and collaboration as major challenges and test clustering and social network analysis as potential areas for research in DLPs. Raman et al. focus on challenges in data exfiltration and describe only three countermeasures from the academic literature. A review by Brindha and Shaji [5] highlights the issue of data exfiltration in the cloud environment with particular focus on data exfiltration caused by authorized entities. The authors highlight the leakage threats, causes of threats, types of information leaked, and data leakage channels. The number of papers included in the review process is unclear; however, this review does not report any countermeasure for data exfiltration. Tahboub and Saleh [6] report a survey of data leakage/loss prevention systems. This is a short survey paper limited to the definitions of data leakage, data states, capabilities of DLPs, and methods adopted by DLPs. The authors compare the scenarios of protecting data from outside (through tools such as Intrusion detection system, firewalls, and Security Information and Event Management (SIEM) systems) and inside through DLPs.

It is worth mentioning that it is a common practice to conduct multiple reviews on the same topic. For example, there are a number of reviews on network anomaly detection [7-11]. However, it is important to ensure that such reviews differ from each other in terms of objectives, included papers, and results. In the following section, we compare our review with existing related reviews.

2.2. Comparison to Existing Literature Reviews

In order to ensure the novelty and new contribution of our review, we thoroughly analyse the related reviews and compare them with our review in terms of objectives, included papers, and results.

Objective

Our review differs from the existing reviews in two ways: (1) whilst all the existing reviews report *challenges* in preventing or mitigating data exfiltration, our review reports the *attack vectors* used to exfiltrate data. By challenges, we mean factors such as several leakage channels, difficulty in managing access right, encryption, and steganography. An attack vector is a particular method or technique such as SQL injection, phishing, and passive monitoring that is used to exfiltrate data; (2) Due to undefined scope, the existing reviews provide some insights into insider attacks and unintentional data leakage. However, our review does not provide any such details rather provides insight into data exfiltration caused by malicious activities of a remote attacker. Whilst some of the countermeasures may also be applicable to insider attack vectors, such attacks are out of the scope of our review. We also consider the fact that data exfiltration is a broad research area and there are a number of resources from which data can be leaked such as computers, mobile phones, web servers, databases, virtual machines, printers, network, and IoT sensors. Hence, it is quite challenging to include papers from all domains. Therefore, the scope of this review is limited to data exfiltration from computers, web servers, databases, virtual machines, and network. We have excluded the papers that report data exfiltration from domains such as mobile devices, IoT devices, and printers.

Included papers

Our review significantly differs from the existing reviews in terms of the included papers. Our review has only 2, 2, and 0 common papers with [2], [3], and [4] respectively. A major reason for such a huge difference in the pool of papers is that the existing reviews (i.e. [2], [3], and [4]) are primarily focused on the insider attacks and industrial countermeasures while our review focuses on external attackers and research-based countermeasures. Whilst our review includes a total of 108 countermeasures, the existing reviews [2], [3], and [4] include a total

of 25, 37, and 3 countermeasures. The review of Brindha and Shaji [5] is focussed on data exfiltration challenges and does not report any countermeasures. Similarly, Tahboub and Saleh [6] define various tools and technologies relevant to data leakage but they do not report any countermeasure from the academic literature. The inclusion of a sufficiently large number of countermeasures (108) as compared to the number of countermeasures included in other related reviews (25, 37, and 3) provides a more comprehensive overview and supports the generalization and concreteness of our findings.

Results

The findings from our review do not overlap with the findings from the existing reviews. Unlike [2], [3], [4], and [5], which report the challenges faced by data leakage prevention systems, our review does not report any challenges, rather, it reports the data exfiltration attack vectors. Similar to [2] and [3], our review also presents a classification of the countermeasures, however, our criteria and the resulting classification are quite different to the classifications presented in [2] and [3]. The classification presented in [3] is based on multiple criteria that include data state, deployment scheme, leakage detection and prevention technique, and remedial actions. This classification overlaps with our classification to a certain degree. However, our classification is based on a single criterion (the technique employed), which enables us to present an in-depth classification of countermeasures into four levels unlike two-level classification presented in [3]. Similarly, the classification in [2] is quite shallow and categorises the countermeasures into high-level categories (two-level classification).

Some other novel aspects of our review, which are missing in related reviews, include investigative countermeasures, mapping of countermeasures to attack vectors, and mapping of countermeasures to data states. The open research challenges identified in our review differs significantly from the research areas or recommendations provided in other related reviews. [3] and [6] identify accidental data leakage and leakage of data from mobile devices as future research areas. These research challenges stem from insider activity and mobile domain covered in [3] and [6], which are outside the scope of our review. The articles [2] and [4] identify certain techniques such as text clustering, social network analysis, fingerprinting, and term weighting that are more likely to be adopted in future research in DLPs. Unlike [2] and [4], the future research challenges presented in our review relates to the quality and evaluation of the countermeasures that are explained in detail in Section 6.

Our Contributions

This paper provides a broad and structured overview on data exfiltration. In a nutshell, we make the following contributions in this paper:

- Summarize and categorize the most frequently used data exfiltration attack vectors to contextualize the in-depth discussion, mapping, and critical analysis of the countermeasures against data exfiltration attacks
- Provide an in-depth critical analysis of a large number (108) of data exfiltration countermeasures
- Present a four-level deep classification of the data exfiltration countermeasures
- Report investigative countermeasures for investigating data exfiltration (which is missing in related review papers)
- Provide a mapping of countermeasures to attack vectors
- Provide a mapping of countermeasures to data states to provide insights about which countermeasures are applicable to protect which state of data.
- Highlight several distinct open issues and challenges that require the immediate attention of the research community.

Table 1. A comparison of our survey with existing survey articles

Topics Covered	[2]	[3]	[4]	[5]	[6]	**Our Survey**
Data exfiltration attack vectors						✓
Classification of Data exfiltration attack vectors						✓
Critical analysis of countermeasures	✓					✓
Classification of countermeasures	✓	✓				✓
Investigative Countermeasures		✓				✓
Mapping of countermeasures to data states						✓
Mapping of countermeasures to attack vectors						✓
Future research challenges	✓	✓	✓	✓	✓	✓

3. Research Methodology

The taxonomy and analysis presented in this paper are based on a structured review of a large number of papers on data exfiltration. We followed a structured process of identifying and selecting the relevant papers from which the relevant data was extracted and analysed to answer the research questions. Table 2 shows the research questions and their respective motivators that stimulated our analysis of the reviewed papers.

Table 2. Research Questions of this structured survey.

Research Question	**Motivation**
RQ1: What are the attack vectors used by remote attackers to steal data from an individual or an organization?	To gain an understanding of the most frequently used remote attacks for exfiltrating data from an individual or an organization.
RQ2: What are the countermeasures for preventing, detecting and investigating data exfiltration?	To get an overview of the countermeasures designed and incorporated by research community for fighting against data exfiltration attacks.
RQ3: What are the challenging issues in preventing, detecting and investigating data exfiltration?	To unfold the research areas that requires further attention from researchers to enhance defences against data exfiltration attacks.

3.1. Data Source and Search Strategy

Seven computer science publication databases shown in Table 3 were each queried for four search terms: *"data exfiltration", "data leakage", "data breach" and "data theft".* We derived these search terms from a series of pilot searches, wherein various synonyms for data exfiltration were explored with the aim of finding a set of results which appeared most relevant and neither too broad nor too narrow. In the rest of this survey, we use the terms paper, study, and document interchangeably for referring to the papers selected for this survey.

Table 3. Database sources

Source	**URL**
ACM	http://portal.acm.org
Compendex	https://www.engineeringvillage.com
IEEE Xplore	http://ieeexplore.ieee.org
ScienceDirect	http://www.sciencedirect.com
SpringerLink	https://link.springer.com/
Wiley	http://onlinelibrary.wiley.com/
Web of Science	https://www.webofknowledge.com/

3.2. Selection of the Papers

Each of the four search terms were applied to each of the seven databases resulting in a total of 28 queries (4×7=28). After retrieving the documents from these databases, we reviewed the title of each document and

made a binary decision as to whether the full text would be relevant to the study's aims (i.e., it appeared to detail either exfiltration attack vectors or countermeasures). After the selection based on the title, the full text of each of the selected papers was reviewed. Some papers were discarded due to the lack of relevance of the full text to our research questions, bringing the pool of papers to its final total of 108 papers.

3.3. Data Extraction and Synthesis

After selecting 108 papers, we extracted the data using a pre-designed data exfiltration form for answering the research questions. The extracted attack vectors and countermeasures from the primary studies were analysed using qualitative analysis technique, namely thematic analysis [13]. We followed the six-step process developed by Braun and Clarke [13] to produce the results presented in Section 4, 5, and 6. The six steps include:

- *Familiarizing with the data:* Data extracted from papers and recorded in excel sheet were read carefully to get deep understanding of the data exfiltration attack vectors, the countermeasures, and the research gaps.
- *Generating initial codes:* After deep understanding of the extracted data, initial codes were assigned to the key points in the data.
- *Searching for themes:* The themes were analysed to divide the attack vectors and countermeasures into potential themes at multiple levels. For example, for countermeasures, the first layer was preventive, detective, and investigative, and then inside each theme, further potential themes were identified.
- *Reviewing themes*: The themes at all levels were reviewed and required modifications were made.
- *Defining and naming themes:* The name of the themes were reviewed and themes were renamed as and where required.
- *Producing report:* Our analysis produced two major categories of attack vectors, three major categories of countermeasures, and six open research areas. These categories of attack vectors, countermeasures, and open research areas are elaborated in section 4, 5, and 6 respectively.

4. RQ1. Data Exfiltration Attack Vectors

This section reports the results of data analysis about data exfiltration attack vectors. This section is meant to answer RQ1, "*What are the attack vectors used by remote attackers to steal data from an individual or an organization?*" Technical methods for extracting data from an individual or an organisation's systems range from obvious methods commonly seen in attacks to niche vectors and cutting-edge methods for covertly moving data past current detection systems. It is worth noting that the attack vectors used by external attackers to steal data are not limited to what is highlighted in this section. There exists a large number of such attack vectors that have already been widely described in the computer security literature. For the details on these attack vectors, readers are referred to [14-16]. We highlight only those attack vectors that are mostly reported in our included data exfiltration countermeasures (108 countermeasures). That means that either the countermeasure is directly designed or evaluated with these attack vectors or there is some level of reflection from the countermeasure that makes the authors believe that such a countermeasure is most suitable for defence against a particular attack vector. For example, we found around 31 countermeasures applicable against malware attacks, so malware attacks have been included. We found only two and one countermeasure applicable to inference attack vector and Sybil attack vector respectively, so we have not included inference attack and Sybil attack in this section. Whilst these attack vectors have already been described in the literature, there is no specific categorization of these attack vectors. We categorize the included attack vectors based on the guidelines of thematic analysis [13]. The identified attack vectors can be categorized in a number of ways such as who initiates the attack (insider or outsider), what is the medium used for data exfiltration (network or physical) or what is the size of data to be leaked? Since we do not cover attacks initiated by an insider in our review, we decided to categorize the data exfiltration attacks based on the used medium. Fig. 3 shows that our initial categories are network and physical.

The focus of our review is on countermeasures and not attack vectors or their categorization. The main motivation for categorizing and briefly highlighting the attack vectors is to contextualize the discussion on the reviewed data exfiltration countermeasures. Such a categorization and contextualization of attack vectors will help a reader to map and understand the countermeasures in relation to the attack vectors.

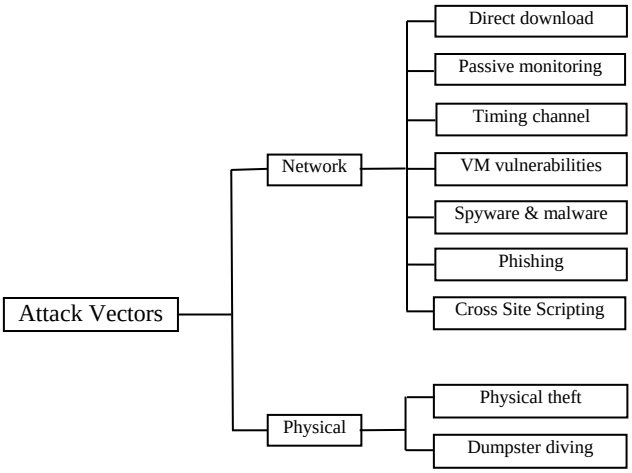


Fig. 3. Data Exfiltration Attack Vectors.

4.1. Network-based Attack Vectors

Network-based attack vectors include those vectors that use existing network infrastructure for stealing data from an (individual) organization. The identified network-based attack vectors are shown in Fig. 3 and described in following sub-sections.

4.1.1. Direct download

Perhaps the most direct method of data exfiltration for a remote attacker is manipulating a public-facing server into disclosing non-public information, such as through the well-known category of SQL injection attacks. The steps involved in identifying and exploiting such vulnerabilities are well-documented [17-19] and in many cases, the attacks are trivially automated for attackers by downloadable scripts.

4.1.2. Passive monitoring

Sniffing the wireless broadcast traffic is well-known but usually overlooked data exfiltration vector. With many wireless networks still insufficiently secure [20], and businesses making more and more use of wireless-connected laptops, tablets, smartphones and other devices, the threat of attackers passively listening to an organisation's traffic is a very real one, and many connected devices leak information [21]. Such broadcast-interceptions are not limited to typical wireless networks. A notable example can be the incident in 2009 when it was discovered that military adversaries of the United States in Iraq were able to access the video feeds of Predator drones by simply listening on the correct channel [22]. Any medium which broadcasts information to an unknown number of users – be it a satellite transmission or a corporate wireless network – should sufficiently be secured when it comes to confidential information.

4.1.3. Timing channels

This method of data exfiltration appears in the literature describing threats but rarely in the literature aiming to detect or prevent exfiltration, yet it is a plausible exfiltration vector for sophisticated attackers. A timing channel is an extremely subtle form of a hidden channel which works by sending innocuous packets to an external recipient at particular times, such that the time delay between packets represents a particular byte value [15]. Such a vector is very difficult to detect, as any traffic could potentially be carrying a timing channel, and the communicated information is not embedded in the packets themselves, merely in the delay between them. Examples are channel operating locally via network sockets [23] and even in variations on keyboard usage [24].

4.1.4. Virtual machine vulnerabilities

Modern businesses are increasingly making use of Virtual Machines (VM) hosted by a third party. However, virtualised computing infrastructure carries its own risks with regards to data exfiltration – including the risk of

a service provider being able to access data, even encrypted [25] – and a number of papers in our review point out covert channels available to attackers targeting such systems. These threats mostly come from co-residency, where a malicious virtual machine is set up on the same physical machine as a target virtual machine. Such a situation can be detected by the malicious machine [26, 27] using watermarks – unintrusive but identifiable 'tags' added to data – to identify the target for infiltration, and then a variety of covert channels can be employed to transfer data from one virtual machine to the other, including exploitation of the physical machine's cache [28], memory bus [29] or network sockets [23].

4.1.5. Spyware and Malware

Spywares are installed on user's computer to monitor user's activity and report back to a third party [30]. Such software is normally used by software providers to enhance their performance by sending relevant updates to users based on their activities. Such monitoring can be used to leak a user's information. Spyware includes malware, adware, cookies, web bugs, browser hijackers and key loggers [15]. Recently designed malware has the capability to scan user's personal computer for personal information and send back such information as an email attachment to all email contacts of the user. It is important to note that such an information leakage is initiated by an internal source, therefore, firewalls often fail to detect such exfiltration [14].

4.1.6. Phishing

A very common method for exfiltrating the personal data of an individual is that of phishing. It is a form of social engineering where the individual is invited (usually via email) to visit a fraudulent website, which is set up for exfiltrating personal information of visitors [31]. Upon visiting the fraudulent website, the user is asked to enter username, password, bank account number and similar details, which ultimately leads to landing sensitive personal information in the hands of the hacker. Some of the famous types of phishing attacks include Deceptive phishing, DNS-based phishing, and Search Engine phishing [32].

4.1.7. Cross Site Scripting

Cross Site Scripting (XSS) is another way of stealing personal information from an authenticated session by injecting a malicious script in an attacked website [33]. Once the malicious script executes, it gives an attacker full access to the information held by the trusted website [34]. XSS is a popular method among attackers for stealing information as evident from the recent OWASP ranking [35] where it is considered as the third biggest attack vector for leakage of personal and sensitive information.

4.2. Physical Attack Vectors

Physical attack vectors include those attacks that get unauthorized and illegal physical access to data and move it to a new physical location. The identified physical attack vectors are shown in Fig. 3 and described in the following sub-sections.

4.2.1. Physical theft

Due to a variety of options available to attackers, it is very challenging to detect physical exfiltration of data [15, 16]. Data can be easily leaked by copying it to CD/DVD, USB, floppy drive or even a laptop [15]. It is also possible to first print data and then hide printed material while leaving premises of an organization. Instead of copying or printing sensitive data, an attacker may steal some physical device on which sensitive data is stored [14]. This may happen due to weak physical security at an organization's premises or due to the carelessness of some individual employee who left a device unattended.

4.2.2. Dumpster diving

At times, organizations adopt weak practices for destroying information (both hard and soft) such as throwing printed documents or CDs into a dustbin. An adversary can find some sensitive information by scanning it [14]. For example, if an organization throws printed documents and CDs in a dustbin, it is quite possible that an adversary may search the dustbin and sees if there is some information of potential value.

A summary of reported attack vectors is shown in Table 4.

Table 4. Summary of attack vectors.

S#	Attack Vector	Description	Study Reference
1	Direct download	Attack on a public facing server to directly download data	[17-19]
2	Passive monitoring	Passively listening and observing an organization network traffic for identification and leakage of sensitive information	[20-22]
3	Timing channels	Monitoring timing (instead of actual contents) of network packets and extracting information based on exploitation of time delays between packets	[15, 23, 24]
4	Virtual machine vulnerabilities	Service provider (such as cloud provider) leveraging and providing VM facility has access to personal data of user Establishment of covert channel for exfiltration of data between two VMs hosted on same physical machine	[23, 25-29]
5	Spyware and malware	Software used by remote attackers to identify personal information in a computer and send it back to the attacker via some medium such as email attachment.	[14, 15, 30, 36]
6	Phishing	An individual is invited to visit a fraudulent website and visiting the website in turn leak the personal information of the individual.	[31, 37]
7	Cross site scripting	Injects malicious script in a website which in turn gives attacker an access to personal information of users held by the trusted website	[33-35]
8	Physical theft	Copying sensitive data to a removable device (CD, DVD, USB etc.) and taking device out of the organization Stealing data in printed form from an organization	[14-16]
9	Dumpster diving	Careless and weak practices of destroying sensitive information gives adversary an opportunity to search weakly destroyed material for information of his interest	[14]

5. RQ2. Countermeasures

This section reports results of the data analysis about data exfiltration countermeasures. This analysis is meant to answer RQ2, "*What are the countermeasures for preventing, detecting and investigating data exfiltration?*" As mentioned in Section 3.3, we employed Thematic Analysis method to produce the results reported in this section. The research community has presented a number of solutions to address the issue of data exfiltration. At the abstract level, these countermeasures can be divided into three categories based upon whether a countermeasure is preventing, detecting or investigating data exfiltration. In some cases, these aims overlap such as a particular countermeasure may both prevent and detect data exfiltration or another countermeasure may both detect and investigate an exfiltration attempt. But this is not the case always and wherever a countermeasure addresses multiple aims, it is clearly mentioned with the specific countermeasure. Our selected studies include preventive, detective and investigative countermeasures. The number and percentage of the selected studies related to each of these three categories are shown in Fig. 4. It can be seen that the research community is primarily focussing on preventive and detective countermeasures, while investigative countermeasures lack sufficient exploration.

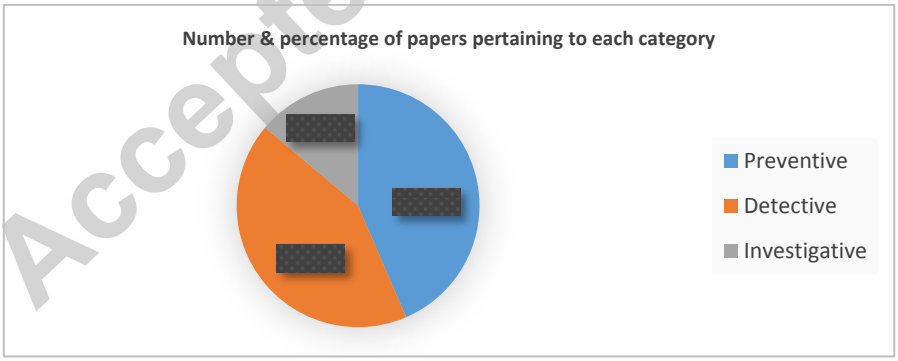


Fig. 4. Statistics of surveyed papers based on category

A possible reason for such a lack of focus on investigative countermeasures could be that individuals and organizations are more interested in protecting their data rather than identifying when, how and who leaked their data. It appears that researchers have focused on preventive countermeasures followed by detective countermeasures as deploying preventive and detective countermeasures is more effective (as compared to investigative countermeasure) for stopping or at least reducing data exfiltration.

Another important aspect of our research is to focus on the state of the data; in use, in transit, and at rest as depicted in Fig. 5. Data in use is the data that is not just stored rather being processed by an application or a

user. Data in transit is the data that is in the form of transmission from one node of a network to another node of the same or another network. Data at rest is the data that is stored in a storage device (hard drive or mobile device) and is not currently under any kind of processing. A data exfiltration countermeasure addresses data in single or combination of two states as shown in Fig. 6. Most of the countermeasure address data in "in use" state. A possible reason can be that data in use is susceptible to various kinds of vulnerabilities and it is challenging to protect data in this state [38].

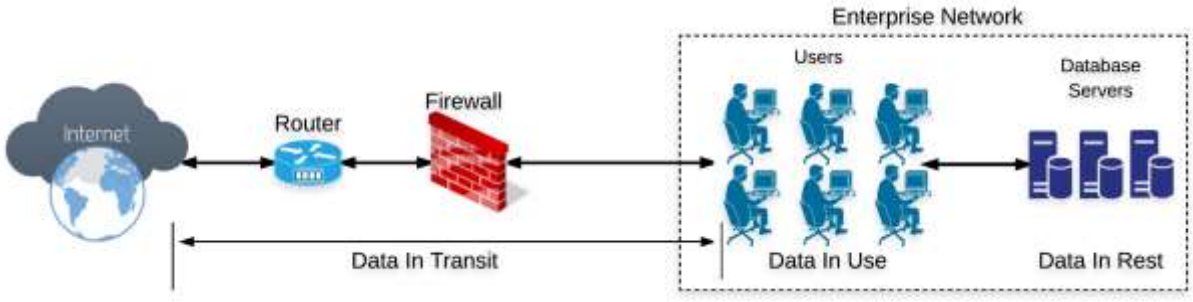


Fig. 5. Three states of Data

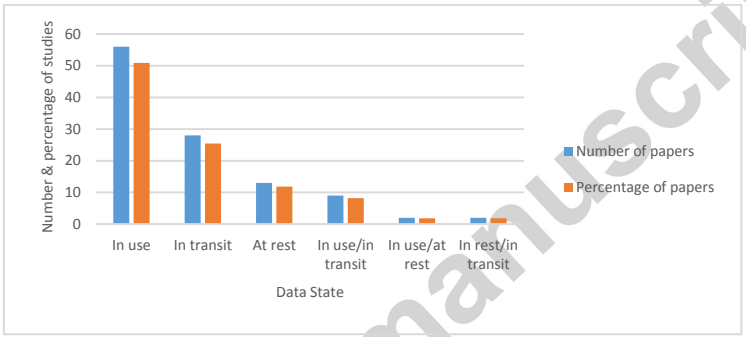


Fig. 6. Number and percentage of selected studies pertaining to each data state.

5.1. Classification of countermeasures

Since the number of countermeasures pertaining to each of the three basic categories (preventive, detective and investigative) was quite high, we have further categorized the countermeasures based on our thematic analysis as reported in Section 3.3. For the basis of such a categorization, we had several options such as types of applications, methodological approaches or domains. On one hand, categorizing the extracted countermeasures based on the types of applications would result in too many categories and on the other hand categorizing the countermeasures based on domains would result in very few categories. Therefore, our categorization of countermeasures is based on the methodological approaches used to address a particular aim (preventive, detective, investigative) relevant to data exfiltration. Our classification of countermeasures is shown in Fig. 7.

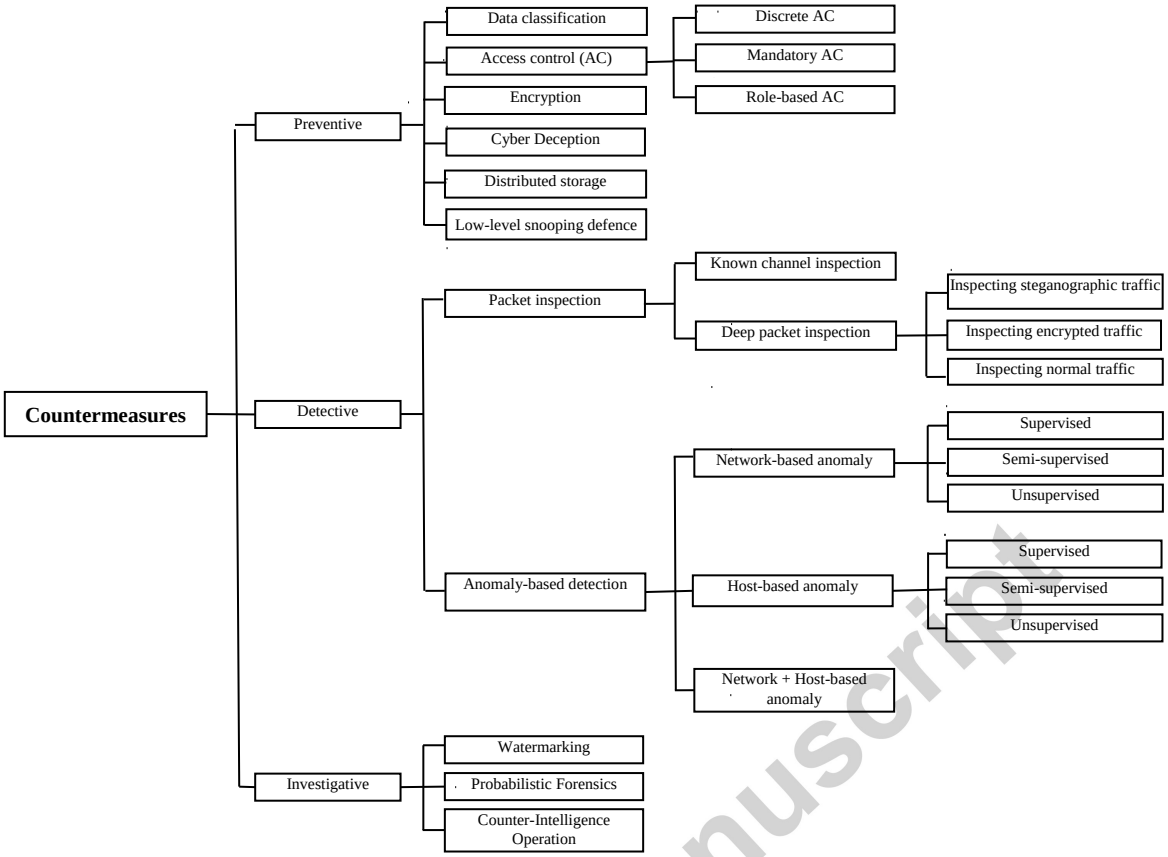


Fig. 7. Classification of Data exfiltration countermeasures

5.1.1. *Preventive countermeasures*

This category includes the countermeasures for preventing an attacker from stealing data. Preventive countermeasures are proactive and resistive in nature. Unlike reactive countermeasures (detective) that comes into play once an attack is detected, preventive countermeasures are more proactive to resist against data exfiltration attempts. These countermeasures are incorporated in the endpoint devices (such as PCs, Laptops, and Servers) to control access to the data resided on these devices or apply particular security tactics (such as encryption, data classification, and cyber deception) to help secure data against exfiltration attacks. Preventive countermeasures are primarily applicable to protect data that is currently under processing (data in use) or stored in a database server (data in rest). A simple scenario of how preventive countermeasures can defend against data exfiltration attack is depicted in Fig. 8. As shown, several preventive countermeasures (Access Control, Encryption, Cyber Deception, and Data Classification) have been employed by the enterprise. Data is classified into different classes based on the sensitivity of data and highly sensitive data is encrypted to enforce strong security measures. Similarly, a chunk of data is shown as negative data. This negative data is a fake copy of the highly sensitive data to deceive attackers. Access control is enforced to stop unauthorized access to the data. Preventive countermeasures are further divided into six categories. The papers pertaining to each category are shown in Fig. 9 and analysed in the following sections.

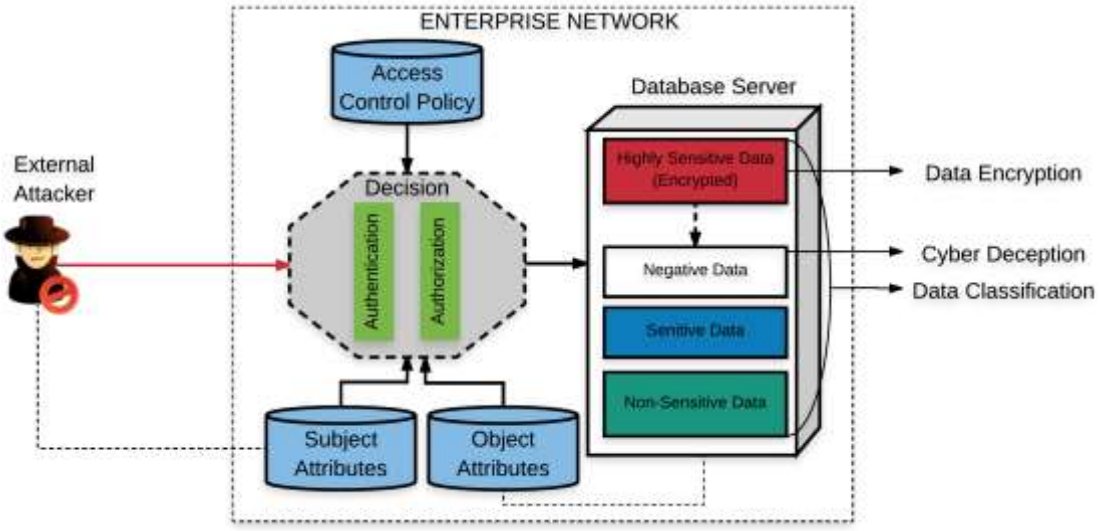


Fig. 8: Depiction of Preventive Countermeasures incorporated in an Enterprise

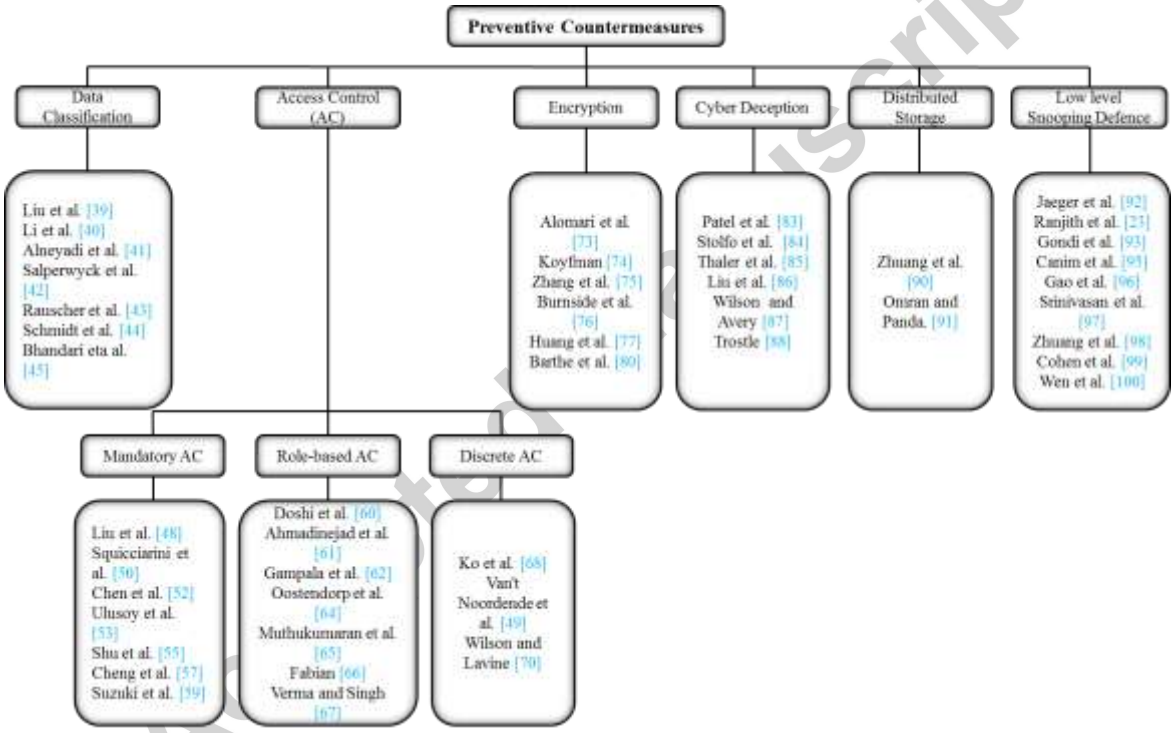


Fig. 9. Papers pertaining to each category in Preventive countermeasures

5.1.1.1. Data classification

In order to prevent data leakage, a number of preventive techniques classify data based on the sensitivity level. Then the preventive systems are incorporated to primarily prevent this sensitive chunk of data from data exfiltration attacks. Different approaches classify the data in different manners. Some classify it only as sensitive and non-sensitive, others classify into several categories, such as highly confidential, confidential, restricted, and public. Such classification helps network traffic monitoring systems and other security measures to treat data based on its sensitivity level. This means that security systems are more focussed on a specific chunk of data rather than the entire dataset. Monitoring of all data going in or out of the system at real-time is also a challenging and computationally expensive option. Therefore such a classification is critical to appropriately monitor the chunk of sensitive data among the huge amount of data in big data systems.

Liu et al. [39] present an approach, called SeCage, for separating secret data and corresponding code manipulating this secret data from the rest of the data (non-secret data) and code. Such a separation helps to prevent leakage of data through memory disclosure attack both at application and OS level. Entire data and code are divided into two compartments (secret compartment and main compartment) using static and dynamic analysis. This separation ensures that only functions inside the secret compartment can access the secret data. SeCage utilizes hardware-based virtualization to ensure strong isolation between secret and main compartments. If any function outside of the secret compartment tries to access the secret data, SeCage notifies hypervisor to handle the violation. The proposed approach protect against various attacks such as buffer over-read attack and hijacking attacks. Also, SeCage stops an attack from stealing data directly from services or through attacks such as timing channel and passive monitoring. Based on the results reported after evaluating the approach, this work seems quite solid, the incorporation of such an approach requires high processing capability. Similarly, the semi-automatic nature of SeCage deployment makes this approach relatively labour-intensive.

Using keyword filtering and data labelling, Li et al. [40] present a technique for preventing leakage of unstructured data from an organization. According to this technique, unstructured data is classified into four categories (highly sensitive, sensitive, internal, and public) based on sensitivity level. After classification, public-private key pairs are assigned to data of highly sensitive, sensitive and internal levels. Data of these three categories is labelled using EIGamal signature algorithm. When data leaves an organizational network, network protection server filters the unstructured data to check if the source IP is organization's network IP and destination IP is external. Next, network protection server filters the cover and attachments with the data using mirror traffic and sensitive keywords. If any sensitive keyword is detected, the public key is used to verify the label. If the label is verified, data is considered sensitive and is blocked. The monitoring of outgoing traffic via network protection server can protect against data leakage attacks such as SQL injection, phishing, and malware attacks, but such it is unclear how much delay has been caused by such extensive monitoring and verification. The real significance of the technique can only be obtained through a formal evaluation. Whilst authors outline the steps for implementing the proposed technique, there is no such evaluation presented that can specify the detection rate or delay caused by the incorporation of the proposed technique.

Alneyadi et al. [41] present an approach to classifying any document leaving the premises of an organization based on the secrecy level using statistical term weighing analysis and if a document is classified of high secrecy level then required blocking actions are taken. Term weighing is a way for defining the importance of each term in a document. The weight of each term inside a document is calculated using Term Frequency-Inverse Document Frequency (TF-IDF) function. Based on the outcome of TF-IDF, a document is classified into a level of secrecy (no restriction, restricted, and top secret). For testing the proposed DLP, 360 documents were divided into three categories (known, partially known, and unknown) based on whether the specific document is known to the DLP. The proposed DLP showed significant detection rates (96.67% and 80.83%) for known and unknown documents, but it failed to detect unknown documents (57.22%) to a reasonable accuracy rate. Similar to Li et al. [40], this approach will protect against data leakage via attacks like phishing, malware, however, it is not reported that how much delay has been introduced in the network flow due to the incorporation of this approach.

Salperwyck et al. [42] highlight an exfiltration vector posed by data analyses which rely on combining public data with private data, where they suspect some portion of the infrastructure around the public data may be attempting to exfiltrate the private information – such as spyware on a public machine. Their solution proposes that private data are kept on a specialised hardware device, which is designed to be resilient to tampering, and that the processing regarding the private data is handled on the specialised device, removing the need for it to be loaded into memory on an untrusted machine. This scheme is resilient to direct leakage of the whole of a private dataset through spyware attacks and demonstrates a solution to one of the trickier areas of policy enforcement. However, it leaves open secondary avenues such as leakage through examining the content of queries to the public dataset and of course the eventually distributed results of the analysis. The authors do not mention the implications of executing queries on a USB device in terms of query processing time for hidden data. This approach requires specialized hardware that might not be a feasible option for several individuals and organizations.

Rauscher et al. [43] present a VLAN-based architecture for reducing the risk of data leakage in a healthcare organization. The proposed approach is based on separating data based on sensitivity level and moving nodes from lower sensitivity zone to higher sensitivity zone. The entire VLAN of the organization is divided into

different network zones based on the sensitivity of data and nodes. Any data packet or node belonging to a particular zone cannot enter another zone unless it passes through specified filters and gateways. A network manager can promote a node from lower to a higher security zone. The proposed architecture reduces the possibility of data leakage via direct copy-paste, emailing, botnets and malware attacks, however, this approach can have significant performance plenty as a result of having a porthole and promoting nodes. With the requests for changing zones going through a network manager can have scalability issues, so an automation support is required.

Schmidt et al. [44] present Trustbox, a system designed to handle sensitive data by restricting users' ability to leak it. The system consists of a trusted virtual machine, which can access sensitive data via an IPSec tunnelled connection to a network storage device. A restricted set of applications allow users to manage an office suite or email but prevent the use of browsers or instant messaging applications – usage of which can be handled in a separate untrusted VM that has restricted ability to communicate with the trusted VM. Email communication is similarly monitored by a deep packet inspection tool. These are all features of a remote client, with some gaps in perimeter security that might be exploited (e.g., user encryption of data to be exfiltrated), but the authors also note that offline usage of the system – and access to data – is desirable, and address this via having the trusted VM check out sensitive data to the local encrypted virtual hard-drive. Whilst a combination of a locked-down virtual environment, secure local storage of data, and preventing the use of portable storage media enhances data protection against attacks like phishing, malware, and physical theft, this approach decreases the data transfer rate from 72.6 MB/sec to 41.3 MB/sec due to the full disk encryption. Similarly, starting an application (e.g., Microsoft Excel) inside a virtual machine instead of a physical machine leads to increased start-up time, which in this case increases from 671 milliseconds to 1104 milliseconds.

Bhandari et al. [45] present a framework for secure storage and retrieval of data from the cloud using data classification, detecting errors in data during transmission via Hashed Message Authentication Code (HMAC), and index building. With this approach, data is first classified into three sections (i.e., public, private, and limited) based on confidentiality, integrity, and availability. Then data is indexed using index building so that it can be searched after encryption. After encrypting data using RSA and adding HMAC to it, data is stored in a cloud. While retrieving the data, a user receives the requested data in encrypted form. The encrypted data is decrypted using the keyword provided by the data owner. A user can apply HMAC check to verify the integrity of the downloaded data. The encrypted transmission and verification of transmitted data help prevent against passive monitoring, however, the framework involves several manual steps that make the adaptation of the framework unrealistic. The authors argue that their approach is better than RSA for ensuring data security in the cloud; however, the paper does not provide any reasoning for justification of this point. The authors do not provide any details about the implementation of the framework.

5.1.1.2. Access control

It is important to enforce authentication and authorization mechanisms for ensuring that only legitimate users with the required credentials can access the data. Such access control mechanisms help the preventive systems to differentiate between legitimate and malicious users. Authentication verifies that a user who wants to access the system is already a registered user while authorization ensures that although the user is a registered one, even such a user should only be granted access to the resources for which the user has already registered. For effectively securing the data and other services, the rule of least possible privilege should be implemented so that users should be granted access to least possible required resource. Although, access control mechanisms provide sufficient defence against attackers who intends to steal data, but such mechanisms cannot be fully relied upon particularly while data is in transit [4]. Access control policies found in the literature can primarily be divided into three categories; mandatory, role-based and discretionary [46]. Access control is in itself a particularly rich area of study and a plethora of techniques has been proposed to control unauthorized access to resources. Of the several access control approaches, only a few are briefly described here.

5.1.1.2.1. Mandatory Access Control

Access to resources or data is controlled by the access policy defined by an administrator and enforced via the operating system. Data objects are labelled and such labels specify the sensitivity of the data objects and indicate who can access the particular data object. This is the strictest kind of access control but it is quite expensive to implement and maintain such an access control [47].

Liu et al. [48] attempt to prevent data leakage through the application of a form of encrypted file system coupled with an authentication server. Rather than actually encrypt the data, their scheme encrypts access functionality (passwords) defined when the file is created. This scheme means exfiltration attempts focused on data stored on disk remain entirely feasible. Policy enforcement is relegated to control of the appropriate passwords for files, the assumption being that passwords are given out in accordance with policy. In contrast with the far-reaching scheme of Van't Noordende et al. [49], who leverages mobile agents to allow content providers to have control over access to their data, this proposition more realistically reflects achievable objectives in security practices. However, it seems the combination of special user-space file system, encryption, and authentication server would most likely affect the I/O performance if a large amount of data is being processed. The paper does not provide any evaluation of the proposed approach.

Squicciarini et al. [50] began with work which looked at various accesses to data – such as indexing – which could violate privacy or data protection concerns with data stored in the cloud. They address this concern with nested JAR files, which bind data to a policy consisting of a service level agreement between tenant and service provider about the permitted access to and use of data. The authors take this further by introducing the notion of autonomous security-aware objects (SAOs) [51] which protect encapsulated data by applying security policies associated with data. The encapsulated data is protected by cryptographic techniques. The protection is defined in the formal security policy, which places conditions on actions relating to the data. The policy of decryption on the fly and transferring data to the content rendering application protects against unauthorized copying. Authors provide a detailed experimental evaluation of the proposed approach. The protection offered is more powerful as it enables different levels of authorisation and access to the protected data. It would appear from various design decisions – including that both government and service provider policies have priority over data owner policies – that the system intends to protect only against exfiltration via certain levels of malicious intrusion, and not, for example, a compromise of cloud service provider's systems.

Chen et al. [52] go further in the same direction and propose a joint software and hardware approach, called DataSafe, to self-protecting data at rest, in transit and during execution. Rather than the model of [50, 51], where access to data is managed with reference to agreed policies, Chen and colleagues consider the situation where a process is unvetted and the process operating on the data is unaware of policies. As well as preventing unauthorised access, the DataSafe architecture aims at preventing dissemination to unauthorised recipients. For preventing dissemination to unauthorized recipients, the high-level security policy for sensitive data is translated into hardware enforceable tags and a Secure Data Compartment (SDC) is created where these tags are associated with the sensitive data residing in the memory. The enforcement of unpassable output control in DataSafe prevent against data breaches through malware installed on authorized machines. The authors compare their idea with the related ideas to demonstrate the superiority of their approach in terms of translating software policies and runtime data tracking. However, the proposed approach has several limitations in terms of performance and cost. The redirection of file calls and setting of SDCs by the DataSafe software is a computationally expensive operation. Similarly, tracking of information flow by DataSafe hardware costs in terms of performance.

The traditional access control measures control access to the file level and not to the individual records in a file while it is quite possible that some records in a file may contain sensitive information. Ulusoy et al. [53] present a framework, GaurdMR, which provides fine-grained access control mechanism at the key-level in MapReduce systems [54] by dynamically creating authorized views of the data based on the roles of a user in an organization. GaurdMR consists of two main modules: Access Control and Reference Monitor. The access control module consists of filters and roles, which are used to create authorized views of data for a specific user. The reference monitor takes input from access control module about security policies and ensures the enforcement of these security policies in a MapReduce system. The datasets (Twitter textual data and Google image data) seem quite adequate for the required evaluation. The high processing and storage capability (8 core processor and 32 GB main memory) may hinder the adaptation of the proposed approach to ensuring controlled access in a system. The incorporation of this approach would also lead to a small overhead in performance as compared to other related approaches (Integrated and Vigiles System) shown by the authors in their experimental evaluation.

Shu et al. [55] present a security architecture, called Shield, for secure file sharing in a cloud environment. The proposed architecture is designed to restrict the data access capability of cloud server by migrating the task of encryption/decryption to the client side. At the same time, the architecture focuses on reducing the burden of

key's management on the client side by storing the security control information for each file in the form of a separate security control file along with the corresponding encrypted data file. Authors introduce a proxy server, which is responsible for authentication and access control. The proxy server leverages the security control information stored with the data file to authenticate and authorize various operations. This security architecture can provide defence against well know attacks that include DoS attack, passive monitoring, timing channel attacks, and rollback attacks. The authors provide sufficient evaluation and comparison with eCryptfs [56] (related approach) for justification of the proposed architecture. The evaluation demonstrates that a single proxy server can handle 45,000 user requests per second, which is quite admirable. Whilst the access control through a proxy server enhances security to prevent data leakages, it reduces the performance efficiency by 7%-13%.

Cheng et al. [57] present a hypervisor-based approach, AppShield, for protecting data, code and execution integrity of an application against OS level malware attacks. In this design, hypervisor separates the address space for application to be protected from the rest of the applications. After separation of address space, any access request from the kernel first gets authorised through system call and then permitted to access the protected application. A system call from the application to kernel may also contain sensitive data and AppShiled provides Spatial protection [58] to such data to ensure that OS access only authorized data. Based on the evaluation and implementation details, AppShield seems a reliable approach for controlling unauthorized access to resources. However, AppShield protection introduces a performance overhead of 0.2% to 10.3%.

Suzuki et al. [59] develop an operating system, called Salvia, with a focus on preventing data leakage. Salvia has two distinct file types: regular files, which can be protected with existing file access controls and "privacy files", which are related to certain data protection policies. These policies are enforced with reference to the contexts surrounding the process accessing a file – these ranging from data such as the connected wireless network ESSID to the system call history of a process. In the policies associated with "privacy files", system calls available to a process are made conditional upon the contexts of that process at the time a call is attempted, with system call functionality broadly grouped to enable easier policy-writing. The implementation of Salvia shows that compared to Linux OS, Salvia on average takes extra 10.35 us in processing a file. The results show that registering of entries in system call histories consumes extra memory. The experiments were conducted in 2007 on a machine with limited processing power and memory (1.0 GHz processor and 256 MB RAM).

5.1.1.2.2. Role-based Access Control

Unlike mandatory access control where data objects are labelled, in role-based access control users are assigned a particular role (e.g., developer, tester, accountant) and based on the role of the user, access is granted to various resources [47]. A number of studies propose role-based access control mechanisms.

Doshi et al. [60] present a multi-layer security framework for a database to prevent data leakage. The proposed framework consists of three security layers: (1) Role-based Access control: intercept user's requested query to check whether the user has the privilege for accessing requested table. The request query is forwarded to next layer (Row-based access control) only if user has required privilege else request is rejected (2) Row-based Access Control: If there is any security policy defined at the row level, the requested query is forwarded only if it complies with the security policy else request is rejected (3) Column-based Access Control: If there is any security policy defined at the column level, the requested query is entertained only if it complies with defined column-level security policy else request is rejected. The proposed security framework can protect web applications against SQL injection and blind injection attacks. The authors claim that the proposed approach does not introduce any overhead in performance and deployment, which seems unrealistic based on the fact that introduction of three security layers would definitely have some effect on query processing time. The evaluation provided is quite weak in the sense that the authors have tested the approach for a very simple and single scenario and the results obtained from such a scenario cannot be generalized.

Ahmadinejad et al. [61] present a view-based protection approach for preventing third-party applications from accessing user's profile shared on a social networking platform. In this approach, each user's profile undergoes sanitization transformation before being released to third-party applications. Such a transformation generates an alternative view of user's profile and disturbs the statistical correlation of data to disable third-party application from making any inference attack. This way queries generated by third-party application are evaluated on transformed view rather than original user's profile data. User profiles are represented as trees and analogues of common sanitization transformations (permutation, suppression, noise introduction, generalization) are formulated from these trees to form the alternate views. Defining transformation rules for semi-structured data

seems a significant contribution, however, this work is primarily theoretical and needs some rigorous experimental evaluation for confirmation of its reliability. Since most of the data on social media is in unstructured form, it will be interesting to look into the transformation rules for unstructured data too.

Gampala et al. [62] propose model controls access of a user using these steps: (1) User requests for uploading or downloading from cloud (2) user is authenticated by asking several personal questions and if user provides a wrong answer for four consecutive times, user is blacklisted, otherwise, the user is given an access (3) a one-time password is sent to the user's email and phone number for next log-in (4) if a user wants to upload a file, the file is encrypted using enhanced elliptic curve cryptography [63] (5) Next time, if the user wants to download file, the file is decrypted using elliptic curve cryptography. The proposed approach can help in preventing the access of a hacker, who had stolen the credentials (username and password) of a user or website admin using an attack vector such as phishing, spyware, or XSS, to personal information resided in the rest state in a cloud. However, this approach is not quite user-friendly as a legitimate user may not be willing to answer several questions each time for a login. It is unknown how the approach has been evaluated.

Oostendorp et al. [64] present a combination of perimeter security policies offered by firewalls and the more flexible Domain and Type Enforcement (DTE) operating system level access policies. They extend DTE over network communications by treating individual packets as objects with their own attributes: the type of information, the domain (of the source) and the DTE uid of the source process. Specified relations between DTE objects then define the access controls across a network. The authors describe the deployment of the system to control exported services, with DTE policies controlling access to local resources and permitting only valid clients to communicate with their respective servers. In this approach, firewalls will only allow ingress network traffic if a DTE policy exists for that particular flow between two processes belonging two DTE domains. As long as data flows occur between known DTE domains, access control can be applied. We see this as a plausible solution to police network-based data traffic between virtual machines or containers on a cloud platform. A set of containerised microservice instances can be determined to belong to a particular DTE domain; these can be forced to communicate only with microservices in another DTE domain as allowed by access policies. This ensures that rogue processes are not allowed to interact with microservices processing sensitive data. It is unclear, on the other hand, how this solution applies to an untrusted model where a software process in a known DTE domain needs to interact with another process in an unknown domain. Also, policy enforcement requires that inter-DTE-domain communication always occurs in a closed loop; if attackers were able to take over one of the processes, for instance by injecting code (e.g. stack smashing), then the solution will be ineffective.

Muthukumaran et al. [65] describe FlowWatcher for mitigating data exfiltration in web applications by monitoring HTTP traffic under the shadow of User Data Access (UDA) policy. The authors define the access policy language, UDA policy language, which is used to define the intended access model by relating users to the data they are going to interact. Since FlowWatcher only monitors HTTP communication, therefore, UDA policy language defines the access policy based on information contained in HTTP communication. This language is designed to accommodate evolving access control requirements dynamically. FlowWatcher sits between the user and web application to monitor HTTP requests and responses. After intercepting a user's request, FlowWatcher updates the access policy based on matching credentials of a user's request with updates rules. Next, FlowWatcher forwards the request to the web application and waits for the response. Upon intercepting the response, FlowWatcher checks if the response contains any data that should not be seen by the user. If any such data is found, the response is modified to remove this data else original response is forwarded to the users. FlowWatcher seems an effective countermeasure for preventing XSS attacks as it strips all HTML tags from the response body. The strength of this work lies in its rigours evaluation where authors wrote and tested UDA policies for 7 web applications. The effect on performance also seems negligible as results show a maximum of 10% increase in CPU utilization. As a whole, FlowWatcher seems an applicable approach for monitoring data in transit and in use for malicious exfiltration.

Fabian [66] addresses exfiltration threat posed by USB and similar storage devices by focusing on different means of disabling USB access, ranging from disabling the USB ports physically or in firmware to running a software suite which permits the use of only authorised devices. The author emphasises that the use of USB devices should be managed according to the role and responsibilities of the users in the organization. The work of Fabian provides some theoretical guidance for the security-aware usage of USB devices in an organization; asset-inventory in organisations need to be extended to output ports on machines and secure configurations for such machines must be put in place to avoid the use of output ports if not authorised. However, it does not

propose any innovative solution for stopping un-authorized copying to USB devices. Verma and Singh [67] describe an implementation of one such device-authentication suite, preventing unauthorised devices from communicating with a system. The work of Verma and Singh also does not provide any evaluation or details on detection rate of unauthorized devices

5.1.1.2.3. Discrete Access Control

In discretionary access control, the onus of regulating access to data objects fall on their respective owners [47]. Data owner also controls the level of access to the data. For example, the data owner may grant only read access to one user but another user may have both read and write access.

Ko et al. [68] implement a kernel-level access control mechanism to discover and notify an end user about data transmission (both authorized and unauthorized) and enables the end user to take the required action. The kernel level module works in collaboration with a user's space program and asks the user's permission for sending any kind of data out of the system. The whole process consists of four steps (1) Data in the form of messages being sent out of the system are captured at the kernel level (2) Sending process is paused to wait for the user's approval (3) Netlink [69] is used to present the captured data to the user in a compatible format (4) a user makes a decision that whether transmission of such a data should be allowed or not and the decision is transferred from a user's space to the kernel's space where corresponding decision is implemented. Giving end user the authority to have control over data transmission can stop data exfiltration initiated by malware and phishing attacks. However, this approach has several limitations that include (1) this approach needs some level of automation as relying on user's permission for each data transfer seems an unrealistic and labour-intensive job; (2) the proposed mechanism is not able to detect data exfiltration via covert channels and (3) this mechanism cannot detect data exfiltrated in encrypted form.

Van't Noordende et al. [49] espouse a more radical vision of information dissemination controls, by outlining an information infrastructure based on the employment of mobile software agents and providing a middleware to facilitate it. In their formulation, sensitive data is contained within 'confined rooms'. Parties wishing to gain access to some portion of the sensitive data send an autonomous agent to run locally, where it has access to the data in order to search for the particular information its owner requires. When a mobile agent has identified the information it wishes to return to its owner, it submits this communication to a gatekeeping agent, which can filter the outgoing information according to the local policy concerns, levy charges, or negotiate non-disclosure agreements. This solution, by advocating for in-situ access-brokering in proximity to the data, suggests a paradigm shift in the way policy is currently applied. Although it can be envisioned as an attractive measure to counteract SQL injection, XSS, phishing, and malware, we can foresee two significant barriers to implementation and deployment: 1) exact understanding of the gatekeeping agent is critical, both where it comes to the information it is protecting and the policies it is instructed to follow, 2) it requires a radical change in how information is stored and accessed in organisations. The authors rely on a theoretical discussion for justifying the validity of the approach and do not provide any experimental evaluation.

Wilson and Lavine [70] present a discretionary access control method for preventing data exfiltration via removable devices. The authors tie the dissemination of files via USB storage to the classification of a file. Files are appointed a distribution level and community of interest label, which is used to automate the decisions about whether or not to allow a file transfer to take place. The generation of the classification information may be more suitable for military domains than for ordinary organisations. The authors tested the proposed approach with a single data type (MS Word); the reliability of this approach (whilst promising), when extended to other types of data, remains to be seen.

5.1.1.3. Encryption

Encrypting data at rest or in transit is a common and mature approach to preventing attackers to gain an access to actual data. Though attackers may get access to the encrypted data (ciphertext), they will not be able to read the actual data (plain text) without having access to the required decryption key. Whilst data encryption is considered a good protective measure, it poses a significant challenge to the deployed detective systems; these often require unfettered access to the contents of transmitted data to detect if the transmission of data deemed confidential complies with defined policies. The use of TLS reverse proxies to enable inspection of HTTPs traffic is a popular solution within the industry [71, 72]. Encrypting data makes the job of detective systems difficult as they may be unable to monitor the actual data. However, recent advances in the detective systems

enable them to monitor encrypted data network traffic. Of the many encryption techniques presented in the literature, a few are described below.

Alomari et al. [73] compare several encryption schemes based on their suitability for storage purpose. According to [73], encryption schemes can be divided into three categories: (1) Software-based encryption: A special software is used to encrypt data inside a storage device; this type of encryption is economical but not too reliable (2) Controller-based encryption: An external hardware or chip performs encryption; since keys are stored deep inside hardware, it is a reliable scheme (3) Internal Disk Encryption: Encryption hardware is embedded inside storage device. This is the most efficient and reliable technique for encryption.

Koyfman [74] presents an automatic encryption approach in the context of customer database security. As motivation, he highlights that data in motion from a central repository to a single user is of low value compared to the repository itself. He presents a database system which automatically and transparently encrypts data, allowing adopters smooth migration to encrypted storage without requiring existing software to make adjustments to their storage procedures. A design does require the adoption of a separate key storage server, which can authenticate the use of keys – for decryption or encryption purposes – for particular clients. Authors incorporate Network-Attached Encryption (NAE) server for managing cryptographic keys throughout the enterprise. Such a system is suited to encryption management within an organisation or even between multiple ones for data at rest to prevent various attacks (SQL injection and XSS) on the database, but does not easily carry to mobile contexts, where network connectivity might not be possible, and where clients leave secured networks. He notes that ideally, the software would itself encrypt and decrypt all the data which it handled as it moved back and forth from storage, but the effort required in terms of (re)development makes this impractical.

Zhang et al. [75] consider a similar approach in a different unrelated context, transparently applying encryption within a file system. They note the utility of such a system for protecting leakage through vectors such as temporary files created by office applications, and the capacity it achieves for automatically applying encryption policy without requiring user action. Their schema for key management suggests that each individual has his/her own set of keys used in disk encryption, stored on a USB drive, which would seem to limit the utility of the method to individual use – files encrypted in this manner are not easily sharable between members of an organisation. This approach seems applicable to prevent attacks on data resided in the rest state in a file system. The automated nature of encryption as claimed by authors seems interesting, however, this work does not explicitly show any evaluation of the proposed encryption mechanism.

Contrary to the attempts to protect data at rest or in use, Burnside et al. [76] focus on protecting data in transit through an organisation's web pipeline. Whilst they accept the functionality of SSL connections for protection on the wire, they point out the unnecessary vulnerability exposed by sending the whole order documents through to service components. The service components only require specific portions of the supplied customer information, with various endpoints terminating the SSL protection as data is moved from the network. They present a novel end-to-end encryption solution whereby fields to be submitted as part of an order are encrypted in their web browser, then re-encrypted at a gateway in order to prevent exposure of the web service's internal structure. Whilst the prototype evaluation of the approach shows an additional 40-150 ms per transaction processing, the approach is quite suitable for preventing timing channel attacks and passive monitoring of the network traffic. The extra processing cost is a reasonable trade-off for increased security of data in transit.

Huang et al. [77] present a tool (GenoGaurd) to protect genomic data by incorporating a combination of encryption and encoding. Their approach ensures that decryption attempts via brute-force attacks using any key or password generate a plausible genomic sequence. The storage and retrieval of genomic data in biobank works in this way: (1) A patient provides biological data to a Certified Institution (CI) and chooses a password to be used for encrypting the data (2) CI generates genomic sequences from biological data, encodes sequences using Distributor Transforming Encoder (DTE) and encrypts the data with a given password (3) The generated ciphertext is stored in the biobank (4) During the retrieval from biobank, a patient or doctor requests for data from biobank (5) encrypted data is sent, which patient or doctor decrypts using the patient's password to get plaintext (6) plaintext is decoded to get the original data. GenoGaurd has been rigorously evaluated as a practical option to be applied to healthcare systems for protecting genomic data in rest state against brute-force attack. However, for full adoption of this approach, a reasonable solution is required for handling the situation where a legitimate user enters a wrong password.

At times, cryptographic schemes are attacked and secret material particularly keys are leaked, which can lead to exfiltration of entire data [78, 79]. Barthe et al. [80] present an approach to detect fault attacks [81] on cryptographic implementations to prevent leakage of concealed secrets. First, fault conditions (mathematical properties of a cryptographic algorithm which helps an attacker to find more information using simple calculations) are identified from the cryptographic algorithm to find the true essence of attacks and learn about potential new attacks. Next, an algorithm is designed which takes these fault conditions as input and identifies the areas having such fault conditions in the cryptographic implementation. The tool developed for evaluation of the algorithm demonstrates positive results in terms of detecting fault conditions. The proposed approach seems an effective solution for preventing leakage from the cryptographic algorithm in itself, but the incorporation of such approach may lead to additional costs for an organization.

5.1.1.4. Cyber Deception

Cyber Deception is an effective solution for defeating and exposing malicious intruders [82]. Adding an element of deceit to data via dummy records in the database is an emerging approach to preventing an attacker to gain access to actual data. It is very hard for an attacker to differentiate between real and dummy records after getting access to a database. An attacker cannot get any aggregated level details (e.g., size of database and growth rate of the database) from the database. The element of deceit can also be extended to data traversing a network. Techniques can be used to generate dummy network traffic that makes an attacker unable to gain any significant and meaningful information even from the maliciously captured network traffic.

Patel et al. [83] present a technique wherein any attacker stealing data appears to gain non-information due to the highly misleading storage pattern – multiple rows of data which appear to relate to one piece of information actually each only carries a small amount of encrypted true information. The generated non-information can be used as watermarks to trace attempted leaks. There are also practical considerations with the system in terms of low storage efficiency of such a scheme. The concept of the negative database can assist in preventing attacks such as SQL injection and XSS on data at rest in a database. However, at least until this moment, this approach has some major limitations. Storing negative data along with the actual data in a database increases the storage cost. The authors indicate that their implementation of the approach can handle *select* and *insert* query but not *update* query. Similarly, storing extra (negative) data in the database effects the query performance too.

Stolfo et al. [84] combine user behavioural profiling with the concept of a disinformation attack. The authors reason that legitimate users' use of search functionality is likely to be targeted and limited towards one or a few intended files, whereas a masquerade with their credentials is likely to perform widespread and untargeted searches. Monitoring for such widespread searches would be a suitable proxy for detecting masquerading attackers. The authors also place decoy files within the defensive system, these being generated documents, which appear to be tax forms or medical records – of interest to an attacker – but when loaded into memory carry a hidden alert code. The authors reason that ordinary users will have no reason to access these files. The countermeasure is suitable where an exfiltration attempt result in data search patterns (whether on the file system or in databases) that differ radically from a 'usual' pattern. The exfiltration vectors that may cause these types of patterns are potentially XSS, SQL injection, etc. Given the additional performance and storage cost associated with placing decoy files in the database, the idea of combining the behavioural profiling and placing decoy files seems impractical. Once the access is identified as malicious, there is hardly any significant advantage in sharing misinformation with the attacker.

Thaler et al. [85] present an approach based on Markov chains for deceiving data stealer and detecting data theft by dynamically creating/placing decoy folders alongside actual folders inside a database and monitoring the interaction of the data stealer with these decoy folders. Since manual creation of decoy folders is labour-intensive, therefore, this approach creates decoy folders dynamically using these four steps; (1) data is normalized to replace properties of actual folders with placeholders (2) using normalized data, Markov chain is learnt (3) using learnt model, decoy directory structure is created (4) the directory structure is instantiated to create decoy folders. These decoy folders do not give any information to the data stealer but any interaction with these decoy folders is detected as a suspicious activity. The automation brought into the creation of decoy folders is the real strength of this work. The approach is risky in the sense that it could detect only 51.57% of malicious attacks. This technique would also fail if attackers already know the project they want to steal.

Liu et al. [86] present an approach for preventing leakage of secrets from an encrypted database. The secrets considered are those gathered from the aggregated level details (size of database and growth rate of database)

and not those gathered by decrypting individual records inside a database. The authors come up with the solution to add a sufficient number of dummy records to the actual records before encrypting and outsourcing data to a cloud. Whilst the addition of dummy records would cost in terms of storage and query speed, this approach would avoid the big cost of secret leakage. Whilst the idea of preventing leakage from aggregated level details seems a fair additional security measure, it comes with increased cost of storage and decreased performance. It is labour-intensive to add dummy records manually to a database.

Wilson and Avery [87] present an approach based on cyber deception for mitigating data exfiltration in SaaS cloud. The proposed process of data protection in SaaS cloud can be described in these steps. (1) User uploads file embedded with MAC address, IP, Hostname, and UserID of the user to the cloud (2) Numbers and names within the document are replaced with random values to create a decoy document (3) Both the original document and decoy document are stored in the cloud (4) Some user of the cloud requests for a file stored in the cloud (5) MAC address of user is checked and if it is the same as the one embedded in the file, original document is returned, otherwise, the user is considered malicious and decoy document is returned. The approach is applicable to protect the rested data in the cloud against SQL injection attack where a malicious user is able to spoof the tuple [MAC address, IP, Hostname, UserID] and submit it with the query. The incorporation of this approach for enhancement of security would double the cost of storage as with every file, a similar decoy file will be stored. It is not clear whether decoy files are created automatically or by the user. This technique requires multiple copies of the same data if the user needs to access it from different host machines which again would increase the storage cost.

Trostle [88] focuses on preventing data exfiltration from a network. The author takes the example of a malicious process with access to data it wishes to transmit out of a network, which it might attempt via a timing channel, sending packets on a vector where its external collaborator can monitor traffic. Rather than packet or traffic analysis, they posit as a solution an encrypted mix-network wherein apparently distinct destination addresses are generated for packets headed to the same actual destination, and source addresses are randomly generated numbers, making traffic analysis by an outside observer significantly more difficult. This 'fuzzing' approach to prevent exfiltration shares similarities with the data-masking countermeasures proposed for handling network content (e.g., in [89]), but applies them to the network as a whole. The proposed approach seems effective for protecting data in transit against timing channel attack and passive monitoring. However, a full adaptation of the approach requires rigorous evaluation of the approach, which is missing in the paper. This approach may not be helpful in preventing attacks in a network with little traffic.

5.1.1.5. Distributed storage of sensitive data

Storing all sensitive data in one repository is discouraged as a successful attack can expose the whole repository. Storing sensitive data in multiple repositories offers some control over data leakage. Leveraging such a distributed approach for storing sensitive data ensures that not all target eggs are put in one basket and even if an attacker successfully gains access to one data store, the data stored in other places remains protected. Such a situation also alerts the data owner about the attack on one data chunk so that additional security measures can be put in place for other sensitive chunks of data.

Zhuang et al. [90] report an information leakage aware approach, StoreStim, to store data in multiple clouds in such a way that the risk of information leakage is reduced to a minimum. Whilst outsourcing data to multiple clouds, data is evenly divided among clouds but not the information contained in the data. This approach first ensures that information (not data) is evenly distributed among cloud providers, which ensures that not all important information is outsourced to a single cloud. This approach is based on the idea that outsourcing similar data chunks to the same cloud reduces leakage risk. The authors designed an algorithm, BSFhish for generating similarity-preserving signatures for data chunks. Using these signatures, another algorithm, SPClustering, divides data chunks among clouds in an optimized and leakage aware manner. Intelligent and planned outsourcing of data to several clouds seems a good idea for reducing the risk of data leakage in cloud environments, however, organizations may be reluctant in adopting such an approach due to the extra storage cost and complexity of data management.

Omran and Panda. [91] focus on secure storage of data in a cloud to prevent it from leakage. Their approach is based on the idea of dividing a single table into several tables to distribute sensitive attributes. The model works according to these steps: (1) Divide the table to scatter sensitive attributes to several tables (2) add dummy records to tables if records in a table are less than a set threshold (3) store resulting tables into separate clouds

(AWS and Azure) (4) Generate code for sensitive attributes and store it on client side (5) Start accessing data stored in different clouds using Bipartite matching with Hungarian algorithm. The countermeasure seems applicable for scenarios where a hacker gets direct access to the data storage server. There are several limitations such as if a single table is divided into several tables manually, then it is labour-intensive; the authors do not provide any evaluation of the approach to justify the effectiveness of the proposed approach.

5.1.1.6. Low-level snooping defence

Threat models must include the possibility that a malicious agent has or can gain privileged access to a machine running a process or storing data. This co-residency of the malicious agent with a data storing/processing entity provides an opportunity for low-level snooping techniques to slurp sensitive data. When co-residency is local, for example, an agent shares the same machine with legitimate sensitive-data-processing programs; memory-scanning techniques can be used to scrape data. Point of Sale memory-scraping malware is a notable example. Alternatively, co-residency can be in the form of a malicious agent gaining control of virtual machine provisioned on the same physical server computer as other legitimate virtual machines hosting data-computing applications. A number of papers report on reducing the window of opportunity when data is in use. This ranges from solutions to block unwarranted access to blocks of memory to techniques that minimise the time and volume of data needs to be held in memory for processing. Other papers propose techniques to prevent the establishment of a covert channel between co-hosted virtual machines, defences against memory analysis and address bus leakage.

Isolation of co-hosted VMs from each other is often more important than their isolation from the hypervisor. Jaeger et al. [92] describe a risk flow policy for regulating communications between virtual machines (VMs), particularly concerning covert channels, which previous Mandatory Access Control (MAC) enforcement failed to regulate. Their approach for enforcement is a hypervisor embedding a variant of the Chinese Wall policy, which allows any of a possible set of VMs to be launched; once launched the ability to launch another VMs is restricted according to the policy. The idea of Jaeger et al. is a good countermeasure for preventing the establishment of covert channels between co-hosted VMs. It can provide security support to data 'in use' in a virtual machine. However, an issue with this paper is that the proposed ideas are not implemented to see that to what level covert channels are prevented and detected.

Ranjith et al. [23] investigate a number of possible data exfiltration channels in cloud-hosted virtual machines. They suggest possible memory-related channels caused by both live migration of virtual machines and their shutdown, as well as networked channels between virtual machines. They find that the current memory management of existing hypervisors is sufficient, but they also demonstrate a detection-resistant form of (1) steganographic communication based on TCP headers, (2) a network-based covert channel that uses sockets for covert data transmission and (3) a timing covert channel. To remedy the TCP channel, they suggest that a controlled VM be placed between the network communications of two VMs, to intercept TCP communications and intentionally mask the possible communication headers. This paper is primarily focused on alerting the security experts about the existence of data leakage vulnerabilities in co-hosted VMs. Some of the remedies are proposed, but they are purely theoretical and require rigours evaluation.

Gondi et al. [93] focus on the lifetime of sensitive data in C programs. They argue that developers do not always clear sensitive data from memory at the earliest possible moment that can lead to unnecessary windows of opportunity for data exfiltration. This is of particular interest in legacy code, much of which is written in C and difficult to manually adjust. They have developed a static analysis and code transformation tool, SWIPE, which tracks sensitive data and erases it just after its last use, in order to mitigate this vulnerability. The approach seems viable for preventing memory-scraping attacks e.g. cold boot attacks where attackers have physical access to the machine or memory scanning malware such as Dexter [94]. The authors demonstrate that the effectiveness of the approach with a comprehensive evaluation of real-world C programs containing sensitive data (username and passwords) and the results are quite impressive with very low-performance overhead caused due to THE incorporation of SWIPE.

Canim et al. [95] quantify the risk posed by memory analysis of relational database management systems (DBMSs). Whilst many DBMSs have built-in encryption support to improve the security of data at rest, the encryption and decryption keys used to manage encrypted data transparently are usually available in memory,

leaving the data exposed for decryption. Existing solutions involve the provision for a secure hardware module managing some portion of the key hierarchy used in encryption. Whilst these solutions solve the issue of attackers capturing private keys held in memory, they do not address the problem when if large amounts of sensitive data are brought into memory during the processing of a query, a significant amount of data can be at risk of theft. The authors suggest a query optimisation process that avoids large amounts of sensitive data being loaded into memory. This strategy can protect sensitive data from malware and physical analysis attacks. There is a performance overhead of around 3% that still needs improvement.

Gao et al. [96] address the data leakage implications of supervisory functions available to hypervisors and privileged VMs in virtualised environments, specifically the ability of software in a privileged use-space to obtain memory pages from any guest OS. They highlight that this functionality has been instrumental in solving other security problems through VM introspection approaches, and as such the removal of the functionality is undesirable. Instead, they champion an application memory privacy protection scheme, which renders certain memory pages un-mappable from the hypervisor and thereby providing protection against malware that can exploit foreign mapping. The exfiltration threat being addressed here is the compromise of the privileged use-space only, not the hypervisor itself. The applications making specific runtime calls whenever sensitive operations are to be performed can themselves form an extra risk in the case of a malicious hypervisor as the procedure highlights the most sensitive memory pages to target.

Srinivasan et al. [97] focus on the threat of malware in a general sense but do cover data exfiltration. Their chief consideration is hiding data, which may be included in an executable or in memory whilst executing an application. They achieve this through three mechanisms: randomisation of the address space of data within executables (preventing easy location of sensitive data objects), holding encryption keys necessary for program operation in the hypervisor rather than the VMs' memory, and remote attestation of the integrity of executables. The idea presented here provides second line defence in combination to the first line defence by anti-virus. Such a multi-line defence may lead to performance overhead and additional cost, however, malware written to generate polymorphic code will defeat anti-virus software anytime, therefore, the three mechanisms proposed by authors should be part of the critical security controls.

Zhuang et al. [98] identify a data leakage possibility in the XOM model of processor security: the address bus. They demonstrate how a control flow graph of a running process can be constructed from the analysis of the address bus, and then matched to a known library. To mitigate this risk, they have implemented a memory-randomisation infrastructure, which means that the memory accesses on the address bus will appear random. They manage this through the use of a "hide cache" on the chip, which remembers the original addresses and a hardware unit that randomly permutes memory space. The memory-randomisation approach proposed by Zhuang et al. seems effective in preventing side channel attacks on the address bus which involves discovering re-use code, revealing IP and CFG matching that leads to leaking sensitive information. Apart from a possible leakage risk such as listening to moments of permutation and number of accesses between permutations, this approach is not applicable for preventing attacks on address bus in a multi-processor environment.

Cohen et al. [99] report a software-hardware approach to self-protecting data within Apache Hadoop, a computation and storage system for handling "Big Data". They rely on a trusted platform module (TPM) chip, which forms the basis of a chain of trust up from the hardware into software handling of data. Software components, on top of trusted bootloaders and other security components, include runtime monitoring of system changes and a trusted network connection component to provide protection against passive monitoring and timing channel attacks. The proposed approach faces several implementation issues such as complexity of configuration and lack of support and unity of various packages. This approach is not able to handle runtime manipulations and requires some level of automation for keys management for secure communications.

Wen et al. [100] present an approach, Gemini, for preventing data leakage by tracking data propagation using page fault interrupt mechanism of OS. Gemini tracks three types of data flows: (1) Data flow from sensitive data sources to memory pages by placing an interceptor (native API hook) between the data source and the underlying OS. Any memory page that reads data from the sensitive source is marked with SMP (Sensitive Memory Page) (2) data flow from sensitive memory page to another memory page (3) data flow from sensitive memory page to the outside world by placing an interceptor (native API hook) between memory page and outgoing channel. If data is marked with SMP (sensitive memory page), the data flow is blocked. The proposed approach will block malware that attempts to scrape memory and intercept data I/O calls. However, the reported

evaluation lacks clarity; and the adaptation of the approach would have severe effects on the performance of CPUs. A summarized view of the preventive countermeasures is presented in Table 5.

Table 5. List of preventive countermeasures.

Category	Paper	Data state	Contribution	Provide Protection against
Data Classification	Liu et al. [39]	In use	Leverages hardware-based virtualization to separate sensitive data from non-sensitive data	Buffer over-read attack, Hijacking attack, Passive monitoring, Timing channel attack
	Li et al. [40]	In use	Classifies the data into four sensitivity levels and blocks exfiltration of sensitive data	SQL Injection, phishing, or malware attacks, dumpster diving
	Alneyadi et al. [41]	In use/in transit	Classify document based on sensitivity using term weighing technique and stop leakage of sensitive documents	Malware attacks, Phishing attacks
	Salperwyck et al. [42]	At rest	Separates private data from public data and places private data on special hardware which is resilient to attack	Malware attacks
	Rauscher et al. [43]	In use	Separates data into different zones based on sensitivity level	Unauthorized copying, phishing attack, botnets, malware attack, dumpster diving
	Schmidt et al. [44]	In use	Separates sensitive data from rest of the data using virtualization technique and restricts access to sensitive data using authentication techniques.	Malware attacks, Phishing attack, Physical theft, dumpster diving
	Bhandari eta al. [45]	In use/at rest	Presents a technique for secure storage and retrieval of data from cloud by classifying data and deploying authentication procedures.	Passive monitoring
Mandatory AC	Liu et al. [48]	In use	Presents a technique for encryption and secure allotment of passwords according to access policy. Also, passwords are authenticated through an authentication server	Malware attacks
	Squicciarini et al. [50]	In use	An approach which binds data to access policy agreed between user and cloud provider	Unauthorized copying
	Chen et al. [52]	In use	A software-hardware based protection technique which binds data with hardware-enforced policies.	Malware attacks
	Ulusoy et al. [53]	In use	Instead of file level, implements access control measures at individual records level based on automatic creation of dynamic views of data	Malware attacks
	Shu et al. [55]	In use	An approach for secure file sharing where access is controlled through proxy server which utilizes the keys stored with each file	Denial of Service attack, Rollback attack, Passive monitoring, Timing channel attack
	Cheng et al. [57]	In use	Approach for protection of application's data from kernel level attacks where hypervisor provides the services of authorizing requests from kernel.	Malware attacks
	Suzuki et al. [59]	In use	Presents OS with two types of files access – regular and privacy. For privacy files, access polices are enforced at all levels.	Phishing attacks
Role-based AC	Doshi et al. [60]	In use	Secures database by filtering requested query through three security layers based on role of the user	SQL Injection, Blind Injection
	Ahmadinejad et al. [61]	In use	Restricts the access of third party applications to information only authorized to them by disturbing the correlation of data in user's profile	Inference attacks, Phishing attacks
	Doe & Suganya [62]	In use	An approach for authenticating user's access to data through one time password system to prevent data breaches in cloud	Phishing attack, malware attack, XSS
	Oostendorp et al. [64]	In use	Presents a DTE-based firewall that secures an organization's network via leveraging role-based access policies.	VM level attacks
	Muthukumaran et al. [65]	In use/in transit	Monitors network traffic to check if any user is trying to access any data against access policy. In such a scenario, user is restricted from accessing the data.	XSS attack
	Fabian [66]	In use	An approach for preventing data being copied to an un-authorized USB device by disabling USB ports	Unauthorized copying
	Verma & Singh [67]	In use	A technique for protecting data from being copied to un-authentic USB device	Unauthorized copying
Discrete AC	Ko et al. [68]	In use/ In transit	A module implemented at kernel level that monitors outgoing network traffic and asks for data owner's approval to allow or block transmission	Malware attack, Phishing attack
	Van't Noordende et al. [49]	In use	Leverages mobile agents to allow content providers to have control over access to their data	SQL Injection attack, XSS attack, Malware attack, Phishing attack
	Wilson & Lavine [70]	In use	Categorize files based on sensitivity level and sensitivity level decides whether file can be copied to a USB device or not.	Unauthorized copying
Encryption	Koyfman [74]	At rest	An approach that encrypts data automatically in a database and leverages a separate server for storing and authenticating encryption and decryption keys.	SQL Injection attack, XSS attack

	Zhang et al. [75]	At rest	Automatic encryption within a file system which store keys in a USB device.	SQL Injection attack, XSS attack
	Burnside et al. [76]	At rest/ In transit	End to end encryption technique for encrypting data both at rest and in transit	Timing Channel attack, Passive monitoring
	Huang et al. [77]	At rest/ In transit	Uses combination of encryption and encoding to securely store data	Brute Force attack
	Barthe et al. [80]	At rest	Approach for identification and prevention of fault attacks on cryptographic implementations	Fault attack
Cyber Deception	Patel et al. [83]	At rest	A technique for placing negative data in database alongside real data to disable an attacker to identify which data is real and which is negative	SQL Injection attack, XSS attack
	Stolfo et al. [84]	In use/At rest	Decoy files are placed in the cloud which disables the attacker to differentiate between real and fake data.	SQL Injection attack, XSS attack
	Thaler et al. [85]	At rest	A technique for automatic creation and placement of decoy files alongside actual files in the database	SQL Injection attack, XSS attack
	Liu et al. [86]	At rest	Add dummy records to data to prevent attacker from finding about aggregated level details (size of database, growth rate of database)	Inference attack
	Wilson & Avery [87]	At rest	Decoy document is created for each document and then both actual and decoy documents are stored. Actual document is returned only to authorized user based on the his credentials already embedded in actual document	SQL Injection attack
	Trostle [88]	In transit	An approach that generate random packets alongside actual packets which disables the attacker to identify which are packets of his interest as the source of packets is also hidden.	Timing Channel Attack, Passive Monitoring
Distributed Storage of sensitive data	Zhuang et al. [90]	At rest	Approach for securely storing data in multiple cloud which reduces risk of data leakage	
	Omran & Panda. [91]	At rest	Presents a technique for securely storing data in cloud by dividing single table into several tables based on sensitive attributes.	
Low Level snooping defence	Jaeger et al. [92]	In use	Uses hypervisor to ensure secure communication between VMs and prevent any covert channels.	Covert channel between VMs
	Ranjith et al. [23]	In use	A technique to prevent establishment of a covert channel between two VMs	Covert channel between VMs
	Gondi et al. [93]	In use	A technique that add extra functionality to a C code for an application which ensures that sensitive data gets removed from the code after its use.	Cold boot attack, Physical theft
	Canim et al. [95]	In use	A technique for securing decryption keys of a DBMS by preventing large amount of sensitive data being loaded to the memory.	Malware attacks
	Gao et al. [96]	In use	A technique that secures memory of an application by allowing it to register its address space with hypervisor that disables any foreign mapping to protected memory pages	Malware attacks
	Srinivasan et al. [97]	In use	A technique for hiding sensitive data during execution of an application	Malware attacks
	Zhuang et al. [98]	In use	Using hardware support and compiler optimization, this technique prevents leakage of data via address bus	Side channel attack
	Cohen et al. [99]	In use	Leverages Trusted Platform Module (TPM) chip to provide a software-hardware based protection to data stored in Hadoop	Timing channel, Passive monitoring
	Wen et al. [100]	In use	A technique for tracking memory page access to detect and prevent data leakage	Malware attacks

5.1.2. Detective countermeasures

Detective countermeasures aim to detect exfiltration attempts. Unlike preventive countermeasures that are proactive in nature, detective countermeasures are reactive that detect data exfiltration attacks and stop them wherever possible. These countermeasures are either deployed at the network level or host-level to monitor the network traffic and host access patterns to either look for transfer of sensitive information or observe abnormalities. In case of detecting transfer of sensitive information or observing any abnormality in the network behaviour or the host access behaviour, an alert is generated. A simple working of detective countermeasure is depicted in Fig. 10. This figure shows the incorporation of content inspection, host-based anomaly detection, and network-based anomaly detection. There are three scenarios of detection (i.e. host-based anomaly detection, content inspection, and network-based anomaly detection) depicted in this figure. Each scenario is denoted by S# (Scenario number). These scenarios are briefly highlighted here and explained in detail in their respective sections.

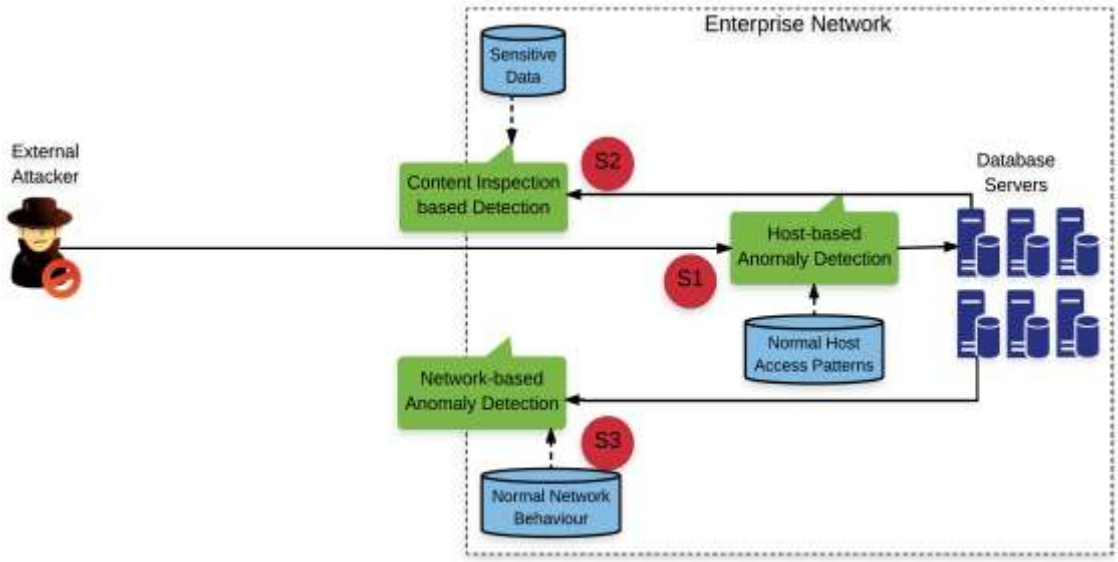


Fig. 10: Depiction of Detective Countermeasures

S1: If a host-based anomaly detector is installed, the access pattern will be compared with the already known normal pattern of access before allowing any access to the data. If the access pattern deviates from the normal access pattern, the access is termed as malicious and access is denied.

S2: If a content inspector-based detector is installed, the data on its way back towards the attacker will be examined by comparing data with an already created database of sensitive data before leaving the premise of the enterprise. If any sensitive data is detected, transfer of data will be either stopped or security administrator of the enterprise will be alerted.

S3: If a network-based anomaly detector is installed, the network traffic carrying the data on its way back will be monitored and compared with already know normal network behaviour. If network behaviour deviates from the normal behaviour, transfer of data will be either stopped or security administrator will be alerted.

Detective countermeasures face many problems common to automated security systems. They must identify rare and devastating events without getting in the way of legitimate uses of a system. The countermeasures pertaining to this category can further be categorized based on multiple criteria, which include functionality, detection time and technique. Based on functionality, there can be two types of detective countermeasures - one only detects data exfiltration but cannot prevent it, and the other first detects data exfiltration and then prevents it either automatically or alerting a system administrator. Based on the detection time, detective countermeasures can be divided into two categories; real-time and post-leaked. Our detailed categorization is based on techniques, according to which detective countermeasures can be classified into two categories; content inspection and anomaly-based detection. Fig. 11 shows the papers pertaining to each category in detective countermeasures.

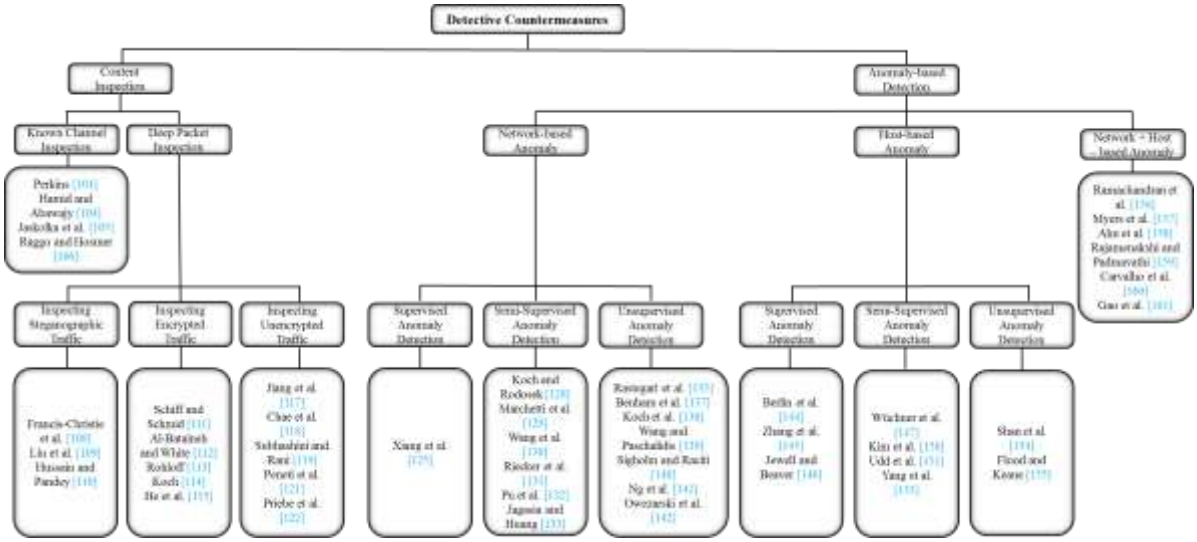


Fig. 11. Papers pertaining to each category in Detective countermeasures

5.1.2.1. Content inspection

One of the simpler approaches to detecting data exfiltration attempts is to inspect outgoing traffic searching for sensitive material through defined patterns, keywords or hashes. Proxy servers for each channel under inspection intercept outgoing data and submit it to content inspection systems for analysis. Thus, proxy servers (for email, IM or FTP channels) enable files' content examiningi using analysers that look for matches in keywords, personally identifiable information, hashes, defined patterns or files flagged by signatures. A positive match identifies the data as sensitive and thus an unauthorised attempt to exfiltrate data across network boundaries. Data exfiltration detection approaches based on content inspection monitors the network traffic in real-time and generate security alerts when an unauthorized data exfiltration attempt is detected. Based on the scope of the inspection, the content inspection can be classified into two categories; Known channel inspection and deep packet inspection.

5.1.2.1.1. Known channel inspection

This is a simple approach where outgoing network traffic is monitored on some known high-risk channel. This approach does not monitor all outgoing network traffic rather inspects only a known channel (such as email) that faces a high risk of being used by attackers for stealing the sensitive data. Perkins [101] in their general review of data leakage prevention suggests such a known-channel system for protecting data in motion and provides an anecdote illustrating their applicability. A particularly common channel for this sort of monitoring is email. The ubiquity and simplicity of email as a transfer mechanism, combined with the relative ease with which it can be monitored via a mail proxy, make it a good target for detection systems. More advanced approaches look at identifying whether emails should be sent based on the topics a recipient is able to access [102, 103], which has a dual benefit in that it prevents accidental leakage through the misdirected mail.

Email-borne phishing activities deceive users into volunteering information that may lead to their accounts or computers being compromised and eventually to sensitive data being accessed by hackers and exfiltrated. Hamid and Abawajy [104] proposes an approach based on clustering and profiling for detection of phishing activities in email communications. The main objective of this work is to help a common user differentiate between phishing and non-phishing emails and so avoid getting into an interaction with such emails. Based on data extracted from phishing emails, clusters of phisher's profiles are created that are expected to show various activities of phishers and help in accurately detecting phishing attacks. The proposed approach may also warn a sender who wants to send email to a legitimate new receiver.

Jaskolka et al. [105] present an algebraic approach to detecting exfiltration of data in motion via protocol-based covert channel using relations. The proposed technique consists of two steps. First, monitoring of the data transmission on the known communication channel; second, computing a relation between data observed on the channel and the confidential data. An organization wishing to monitor a sensitive communication channel installs a monitor on a particular channel. A monitor extracts and stores header field from packets transmitted on

the channel and computes a relation between the extracted information and the confidential information already configured with the monitor. Investigating this relation verifies whether or not any confidential information has been leaked. The implementation details of the proposed approach are not available. Hence, it is not clear how much delay would be caused due to computing the relation between header fields of leaving data and already stored confidential data. The detection rate achieved by the approach can be justified after proper implementation.

Raggo and Hosmer [106] separate the discovery of steganographic use and the identification of steganographic carriers from the extraction of the hidden content. They discuss the deployments of the method on a number of known channels systems. Such systems are limited to studying the content of the medium for which they are designed, and as such provide no protection against unexpected exfiltration channels. Naive design approaches would also fail to catch encrypted or steganographic files, but steganalysis modules for email filters are viable.

5.1.2.1.2. Deep packet inspection

Unlike known channel inspection, deep packet inspection monitors all outgoing traffic for an overlap with sensitive data. Such an approach provides higher level of detection capability as it ensures that not even a single data packet goes un-inspected

Inspecting steganographic traffic

One of the common techniques used by hackers is to hide the sensitive data inside other non-sensitive data so that the installed detective system cannot detect the sensitive data. Modern steganography works by identifying either redundant space within innocuous files or unused fields in common communication protocols (including the ubiquitous TCP/IP) and then encoding the message into these overlooked areas [107]. A number of approaches have been proposed for inspecting such hidden contents to detect data leakage attempts.

Sometimes sensitive information is concealed inside a video and exfiltrated from the network of an organization. Francis-Christie et al. [108] present an approach to detect and prevent exfiltration of data in video using a combination of passive (for detection) and active (for prevention) steganalysis. Data inside a video can be hidden in data frames, metadata or audio stream. The proposed approach uses DST and wavelet filters to remove hidden data from the frames of the video. The D4 wavelet of the video is generated and LH, HL and HH sub-bands are deleted to remove hidden data. These filters are installed between networks of an organization outside Internet so that filter removes hidden data from the video before it goes out to the public Internet. The proposed approach seems suitable for detecting and preventing exfiltration of sensitive data within video frames, however, the authors do not discuss the delay caused by such an intensive filtering process for each video leaving the organization's network. Since the proposed approach does not stop the egress of video and only removes the hidden data from the frames, it is not clear whether the removal of such hidden data causes any damage to the quality of the video.

Liu et al. [109] cover three related features for advanced DPI systems. The first is application identification – allowing network administers to enforce when traffic, which uses the same protocol (SSH, HTTP), is devoted to a particular application (webmail, video streaming, and social media). Tunnelling application traffic through nonstandard means is a typical technique for covert exfiltration. The second feature covered is that of generating and detecting content signatures – building fingerprints of sensitive files is important due to the scalability issue around large signature/file databases in data leakage systems – which they approach through extracting wavelet transforms: signatures, which preserve patterns against noise and distortions. Their particular focus is video content, and for this, they adopt a time-series approach, which allows matching from arbitrary start-points. The final feature they address is detecting steganographic exfiltration, which they address in a particular domain by comparing the wavelet coefficients of a signal and its de-noised version as features for a classifier trained on steganographic and non-steganographic audio signals. The proposed three-layer framework seems quite promising for monitoring network traffic, however, incorporation of such an approach for an organization may be very costly. The evaluation provided is based on a specific case (3 web applications), which is insufficient for generalization of the presented results.

Instead of detecting steganographic contents, Hussain and Pandey [110] present an approach that uses public key steganography to alert communicating parties if carrier data comes under an attack on its way. Once alerted, communicating parties can take the required measures to stop the attack or mitigate its effects. The normal communication between two parties consists of carrier data and overhead. This overhead contains reserved bits

in addition to other fields. These reserved bits can hide a secret message about attack alert. If carrier data comes under an attack on its way, this secret message hidden in reserved bits alerts the communicating parties that carrier data is under an attack. The secret alert message should be hidden in reserved bits using steganography otherwise an attacker can always find a way to stop sending alert to the communicating parties. The proposed work is quite superficial and left many issues unaddressed. For example, it is not addressed whether the instalment of the secret message within reserved bits is automatic or needs to be done manually. It is not clear whether the detection mechanism only detects modification of the carrier data or also detect any kind of passive monitoring of the carrier data.

Inspecting encrypted traffic

Another technique used by attackers to evade detection is to first encrypt the data and then steal it. Due to widespread adoption of SSL/TLS, the ability to initiate secure connections is common and usually necessary. Without specific countermeasures in place, network monitors would not be able to detect the contents of encrypted data packets being delivered via HTTPS or common utilities like SCP or SFTP [15]; and thus would not be able to identify sensitive data being extracted. A common practice in the industry to monitor encrypted traffic is to use reverse proxies to decrypt data at an intermediary point between backend servers and clients. Clients use a public key emanating from the reverse proxy to encrypt their requests to a remote server. The reverse-proxy will decrypt and inspect the client request and then engage in a TLS-encrypted communication with the remote server on behalf of the client. This solution, whilst effective, comes along with the serious invasion of privacy for legitimate users. On the other hand, encrypting network traffic preserves the privacy but current data leakage prevention systems find it challenging to inspect encrypted network traffic. To address the issue of inspecting encrypted network traffic for detecting data exfiltration, a number of techniques have been reported in the reviewed papers.

Schiff and Schmid [111] present an approach, PRI, that inspects encrypted network traffic in a privacy-preserving manner. This approach sets up a Software Guard Extensions (SGX) server that inspects all outgoing or incoming traffic based on the rules configured in the SGX server by the administrator. When user PC of an organization establishes a communication with a PC from outside, a session key is generated and shared with SGX server. Any data sent outside goes through SGX server, which first decrypts the data using session key and then inspects it based on the rules. In case any sensitive data is found, it is stored in Secure PRI storage and an alert is generated to let an administrator know that sensitive data is being leaked. An administrator can modify data exfiltration detection rules in PRI server to match the requirements of the particular organization. This approach is not altogether different from the encryption terminating capabilities of reverse-proxies. It, however, has dependency on SGX hardware (for supporting Intel SGX security rules), which may not be feasible for many organizations. Due to the unavailability of the required SGX hardware, even the authors could not evaluate the efficiency of the proposed approach.

Al-Bataineh and White [112] focus on the threat of data exfiltration via botnets, which routinely employ encryption to thwart data leakage prevention systems, and disguise their command-and-control channels in web traffic. They develop a classifier to detect data-stealing HTTP requests made by the Zeus botnet, which includes encrypted data where there should be plain text. The use of encryption identifies an exfiltration attempt, as randomly-distributed symbols should not (barring compression) appear in the traffic examined. The proposed approach seems a good fit for detecting data stealing via malware attacks. The authors provide a comprehensive evaluation of the approach and the accuracy level achieved seems quite promising. However, the processing time consumed in analysing the traffic still needs to be reduced to make the approach more practical.

Rohloff [113] presents an architecture for monitoring signatures in the encrypted data flow in the network to avoid exfiltration of any sensitive data. The approach is using homomorphic encryption to securely operate on encrypted data for detection of sensitive signatures. The system consists of (1) Data server (2) Data guard (3) Policy authority (4) Data recipient. The Policy authority handles key distribution and identification of sensitive signatures, which should be monitored in outgoing traffic. The Data recipient submits a request to Data guard and so data guard extracts data from the data centre and runs sensitive signature detection on the data. The Data guard sends results of detection process to the Policy authority that checks if data to be exported contains any sensitive signatures before sending the data to a recipient. The processing time taken by various modules (EvalAdd and EvalMult) is different, which causes unnecessary delays in the monitoring process. As a whole, inspecting all encrypted traffic via policy authority is a computationally expensive and time-consuming

approach. A possible solution could be to send only an encrypted bit to the policy authority for approval. The authors do not provide any details on the detection rate achieved by the proposed approach.

Network analysis approaches are doubly beneficial in that they can be applied to an intrusion detection system (IDS) for both intrusion and extrusion detection. Koch [114] includes extrusion detection as a core component of their design for next-generation IDS. They discuss three possible approaches to handling encrypted communications within this system: detecting misuse of the encryption protocols, altering protocols to allow packet payload analysis, and finally statistical approaches, which examine packet sizes and time intervals. The proposed approaches are just ideas and the authors do not go into the details of the implementation and discussing their pros and cons.

He et al. [115] propose an approach to detecting exfiltration of encrypted data from a cloud platform. The detection process consists of two steps (1) Monitoring of all outgoing network traffic to find whether traffic is encrypted or not. For identification of simple encryption, Deep Packet Inspection (DPI) has been used while for identification of proprietary and plaintext encryption, entropy technology has been utilized. (2) Determining whether initiation of such encrypted data movement is expected from a particular user or not. For this purpose, user behavioural profiles are built based on particular features (destination IP, port, and protocol types) using Multinomial Naïve Bayes algorithm [116]. If the encrypted outgoing traffic is not expected based on a user's profile, it is considered exfiltration and connection is shutdown to stop exfiltration. Whilst the detection accuracy (of around 90%) achieved by the proposed approach seems quite promising, the delay of around 45 sec caused in the network flow due to the traffic monitoring is still a serious concern and needs to be addressed. The details about profiling users are also missing in this work.

Inspecting unencrypted traffic

A variety of approaches exist that can inspect data that is neither hidden nor encrypted. The inspected data is compared with a database of sensitive data. If any match is found, the transmission is termed as malicious and a security alert is generated. Due to the difficulty in inspecting encrypted traffic, a possible strategy is to ban end-to-end encryption within an organisation; it is a common practice to force data exfiltrators to use the channels that are heavily guarded within an organisation instead of monitoring a large number of possible exfiltration channels.

Jiang et al. [117] highlight the role of peer-to-peer technologies as data leakage threats and focus on the contribution of content-scanning systems in perimeter data leakage prevention systems. A typical DLP system scans for the content of outgoing packets in a database of sensitive files for determining any significant data match. They contrast this problem with simpler signature-detection roles for DPI systems preventing virus propagation, where packet fragmentation is a smaller issue as the relative size of signatures is much smaller. The authors develop a formal representation of the content-matching problem for their own solution, which is linear in speed to the size of the individual packets and linear in memory to the database of sensitive files. The proposed approach is applicable for preventing various attacks such as SQL injection, XSS, malware, and phishing. The content scanning approach requires an up-to-date database of sensitive information. The filtering of network traffic may lead to delays in a busy and large network. The direct monitoring of the traffic generated by each user causes privacy concerns among users.

Chae et al. [118] present a privacy data leakage prevention method for P2P networks by removing privacy data from the file to be sent out of the organization network. When an external user sends a request for a file, a user inside the organization has to send the file. Before leaving the organization network, the file is checked for removal of privacy-sensitive data. The proposed approach seems suitable for prevention against malware attacks that gather and send out sensitive information out of the user's PC via a P2P channel. However, the proposed approach requires an updated database of private data of all users. Similarly, halting the network traffic for removal of privacy data from files cause delays that may not be acceptable in several cases.

Subhashini and Rani [119] presents an approach to be used in Data Leakage Prevention Systems (DLPs) for detection of confidential terms in a document. In order to train the dataset, documents (both confidential and non-confidential) are clustered using K-means algorithm. For detection of confidential terms, language modelling [120] is used for calculating the value of each term. The probability of a term represents the importance of a term in a cluster. Language models for both confidential and non-confidential documents are created by calculating the probability of each word in a document. Next, cosine similarity between clusters from

confidential and non-confidential documents is found and if any cosine similarity is above the threshold than that cluster from non-confidential document is included into the confidential cluster. Next, confidential score for each term in a confidential language model is calculated. If the confidential score for a term is above the threshold, the term is selected as confidential. The proposed idea can be utilized in DLPS against malware attacks. More rigorous evaluation is required since the paper does not clarify the detection accuracy of the proposed idea or its effects on the overall performance.

Peneti et al. [121] present an approach for data leakage prevention based on timestamp. The proposed approach consists of two phases: (1) Learning phase: all documents of an organization are collected to create clusters of data, calculate scores for key terms in the documents and assign a timestamp to each document based on the organization's schedule (2) Detection phase: the outgoing documents' timestamps are compared with the timestamps of the documents in the identified clusters; based on the results of the comparison, a decision is made whether or not the documents are allowed to be sent. We believe that the premise used for document classification (for the purpose of content-matching at the perimeter upon egress) is weak as the duration for which confidential documents remain highly sensitive in nature is indeterminate; the computation of timestamps may thus be inexact and flawed. The learning based approaches usually do not work for detecting unknown attacks. The evaluation also appears to be weak as there are not many details reported on the accuracy level achieved and the effects of the approach on the performance of the overall system.

Priebe et al. [122] propose CloudSafetyNet (CSN) - a monitoring framework that enables tenants of a cloud to monitor the flow of their data for detection of accidental data leakage. The process is based on mutual cooperation and trust among cloud tenants. Detection of data leakage consists of three steps: (1) Cloud tenant adds an encrypted security tag to a subset of text fields in HTTP request using JavaScript library (2) Tag monitors of all cloud tenants installed by cloud provider observes and stores security tags (3) Cloud tenants periodically checks and decrypt their tag stores to check for any foreign tag, which can indicate an accidental data leak about which the relevant tenant is informed. The framework is designed to defend against malicious tenants, who try to create false positives in leakage detection, by preventing Sybil attacks. The approach cannot detect leaks via all paths as it needs monitors to be installed along all paths, which is an expensive requirement. Managing the tags and storage for them is another issue with this approach. Most of the times, tenants may not prefer to interfere in another tenants affair and might not be willing to report such data leaks.

5.1.2.2. Anomaly-based detection

Anomaly-based detection is a technique, which compares ongoing pattern or behaviour (of network or host) with the expected pattern or behaviour and if any deviation is observed then it is termed an anomaly. Such unexpected pattern or behaviour is referred in different ways such as anomaly, exception, surprise, outliers, aberrations, and peculiarities [7, 123]. Unlike packet inspection where detection of data exfiltration is based on monitoring contents (actual data), anomaly-based detection techniques analyses the context such as source, destination, size, timing and format to detect data exfiltration [2]. Based on anomaly, detective techniques can be divided into three types; network-based anomaly detection, host-based anomaly detection, and network + host –based anomaly detection.

5.1.2.2.1. Network-based anomaly detection

Network-Based Anomaly (NBA) detection techniques monitor network traffic to determine whether communication flows differ from baseline conditions in terms of traffic volume, source/destination address pairs, diversity of destination addresses, and time of day or the (mis) use of particular network protocols. In different detection approaches, the detection is based on modelling different aspects of the network traffic such as IP flows, protocol membership, and packet jitter. This involves gathering metrics/observations about the normal behaviour of the network at different times of the day to compute the baseline condition. Any deviation observed may allude to suspicious activity like data leakage. The difficulty with this category of techniques lies in the determination of the baseline conditions for 'normal' user activity. Not only is this difficult to estimate - the dynamicity inherent in organisations may result in unpredictable spikes of network activity; the potentially high false positive rate can be damaging to business activity. A sufficient number of anomaly-based detection techniques have been presented and delineating all of them is challenging. Network-based anomaly detection approaches operate in three modes; supervised mode, semi-supervised and unsupervised [124].

Supervised mode

In supervised mode, training data sets are prepared for both normal and abnormal network traffic. Network traffic under monitoring is compared with both the classes (normal and abnormal) to decide which class it belongs to. If traffic belongs to the abnormal class then it is considered a malicious activity, which indicates towards a possible data leakage. It is important to note that approaches based on supervised mode are not able to identify new traffic patterns (that does not exist already in normal and abnormal training data sets) as either normal or abnormal.

Xiang et al. [125] present a network intrusion detection approach that is based on Extreme Learning Machine (ELM) [126, 127]. Authors have used a MapReduce implementation of ELM which enables it to scale horizontally. In this work, ELM is used for classification of intrusion attempts. This work is focussed on improving the speed and accuracy of machine learning techniques to help intrusion detection systems deal with big data efficiently. This objective is achieved by presenting MapReduce implementation of ELM.

Semi-supervised mode

In the semi-supervised mode, training data set is prepared only for normal network traffic. The network traffic under monitoring is compared with the prepared training data set for normal traffic and if it does not match, then traffic is termed as malicious. One drawback is that if a genuine user undertakes some activity that is legitimate but new, this approach would term it as malicious as because of unavailability of the training data, the new activity may not belong to existing class of normal traffic and may be considered an anomaly.

Koch and Rodosek [128] discuss an 'extrusion detection system' for identifying users engaged in an encrypted remote session. Their approach is based on the characteristics of traffic in remote sessions, where each keystroke is sent as a packet of a known size, and the size of server responses reveals whether a command has been executed. Using this restricted set of typing data, certain features such as typing speed, regularity and the order of delays between keypresses can be extracted, and compared to a profile generated for every authorised user. The cross-correlation between the session's features and each of the authorised users' profiles is used to select the user most likely to be in the session. Whilst the approach seems more applicable for preventing an insider from exfiltrating data in encrypted form, it can also be effective against malware attacks that first attempt to encrypt data and then steal it out of the network. A problem with the approach could be the effort required to generate each user's profile. Whilst learning the behaviour of a new legitimate user, the request for sending data of this new user despite being legitimate may be blocked.

Marchetti et al. [129] report an approach for network traffic monitoring to identify specific hosts inside a large network that show malicious activities, which lead to pawing ways for data exfiltration and other Advanced Persistent Threats (APTs). The proposed approach extracts certain features from the network traffic for each host over time and evaluates it for suspicious activities. These features include size of data transfer, number of data transfer and number of distinct destinations for data transfer. The evaluation result is presented in the form of a ranked list of all hosts, which indicates to organisation's security analyst where they need to focus to stop data exfiltration and APTs. This approach consists of: (1) Collection of network flow records going out of each host (2) Extraction of certain features from flow records (3) Normalization of collected features for comparison purpose (4) Computation of suspiciousness score based on the features that relates to present and past behaviour of host (5) Ranking of hosts based on their speciousness score. The proposed approach has been thoroughly evaluated over a network of 10,000 hosts and monitoring traffic for around 5 months. The level of accuracy for detecting data is quite promising. However, it is expected that an APT will be launched and detected, but the authors show the detection of simple exfiltration of 40 GB and 9 GB of data.

Wang et al. [130] propose a visual analytic based approach for detecting data exfiltration by monitoring network traffic. For demonstration, the authors use data set of IEEE VAST 2013 mini-challenge 3 that contains network traffic collected over a span of two weeks via NetFlow Collector. The abnormal traffic behaviour is identified based on payload size and duration. Extremely large payload size and long duration indicate data exfiltration. A graph between payload size and duration is generated to visualize the behaviour of these two parameters. Traffic lying in the top right portion of the graph is most likely suspicious for data exfiltration. The authors demonstrate the efficiency of the approach with a single specific dataset, which makes it harder to generalize the results for a large and diverse network traffic. The authors do not specify any specific threshold for defining payload size and duration as sufficiently large to be identified as an attempt of data exfiltration.

Riecker et al. [131] focuses on the cost and efficiency of intrusion detection system and proposes an approach based on the energy consumption of sensor nodes for wireless networks. If a sensor node comes under a malicious attack, its energy consumption either becomes too high or too low. In this system, a mobile agent is used that carries a battery status that shows the expected energy consumption of a sensor node. The energy consumptions of sensor nodes are predicted using linear regression model. This mobile agent moves from node to node and compares its current energy consumption with the expected energy consumption. If there is a huge deviation, it is considered that system has been under attack. The approach is designed to detect flooding and blackhole attack. Such a detection approach may work for a uniformly behaving wireless network but will fail to detect real attacks in a wireless network whose behaviour is variable. The approach does not seem suitable for large network due to the low storage capacity of currently available network nodes. The detection accuracy of 100% is based on simulation, which may be different if the system is implemented in the real world.

Pu et al. [132] present a network-behavioural detection system which is focused on one particular exfiltration threat – data-stealing Trojans. They identify three behaviours typical of such Trojans: an unusually upload-biased upload to download ratio, a connection kept open for a long time, and a master/slave dynamic in connection setups. They build a detection system focused on these elements, with the latter two behaviours forming a fast-acting detection system and the first implementing sanity checking by profiling the relative transfer rates to detect potentially mistaken decisions. The proposed system can help in preventing malware attacks by analysing the network behaviour. However, the current form of the system is quite immature and reported work has several ambiguities. Most importantly, the reported experimental evaluation is insufficient and does not convey any significant information about the performance of the system.

An appropriate deployment of network analysis can be useful in securing even networks deployed in insecure environments. Jagasia and Huang [133] examine the possibility of data theft in wireless sensor networks, where physical security of the network cannot be guaranteed. They suggest that attackers may introduce malicious nodes into a wireless network in order to observe its traffic – traffic analysis being both a potential threat and solution in this scenario. The authors propose a MAC-level detection system, which would allow the sensor network to identify these nodes which are attempting to transmit anomalous volumes of data. Using this system, coordinating nodes in the network – or a group of cooperating neighbours – would monitor their local nodes for high frequencies of transmission. The proposed approach is applicable to detect passive monitoring and timing channel attacks as it can detect some adversary listening to network traffic or gathering aggregated level information about the network by installing a malicious node. The authors present experimental evaluation in detail, however, the detection rate (of around 60%) does not seem quite impressive and needs some improvement that can be achieved by incorporating the points suggested for future extension of this work.

Unsupervised mode

In the unsupervised mode, training data set is neither available for normal network traffic neither for abnormal network traffic. This mode relies on the assumption that normal traffic instances are in majority and abnormal traffic instances are usually in minority [134]. Since this mode of data exfiltration detection does not require training data sets, it is widely adopted mode while designing approaches for data exfiltration detection.

The issue with signature-based intrusion detection systems is that the features for detection are extracted from the packet header, which does not really display the actual behaviour of network traffic in all cases. To handle this issue, Rastegari et al. [135] report a network intrusion detection technique that is based on the combination of statistical features and machine learning. First, seven statistical features based on entropy and volume are extracted from network traffic. Then the rule learning technique, ERS-NID [136], is used to generate a rule set for detecting network intrusion. The experimental results show that the approach can achieve a high detection rate and learn the behaviour of network in a short period of time. Whilst this work detects intrusion in a network, it can provide inspiration for designing an unsupervised data leakage prevention system.

The behavioural approach seems a rich area for exploration, however, it should be noted that not all abnormal traffic will be data exfiltration. Benham et al. [137] also advocate a behavioural approach to network modelling for data exfiltration, but one which is less sensitive to legitimate anomalies. They suggest a ‘Behaviour Engine’, which begins with no rules at all and gradually learns them by observing and querying a user’s use of a network. The aim here is to capture the tacit knowledge of users rather than just the explicit knowledge. Explicit knowledge is usually captured in requirements that can be transformed into rules for a decision engine. By automated system validation, the engine ensures new behaviour does not invalidate old behaviour that means the

system can grow without being 'trained' by an attacker to allow malicious behaviour. The authors do not provide any details on how the proposed engine would avoid learning an already present malicious behaviour as legitimate behaviour. A major issue with unsupervised anomaly detection techniques is the high false alert rates, which is usually verified through experimental evaluation. However, the authors do not provide any experimental evaluation that leaves the credibility of the proposed behaviour engine at stack.

Koch et al. [138] present a behaviour-based intrusion detection system for encrypted environments. Unlike traditional intrusion detection systems that require some signature database or learning process, this approach uses the concept of majority decision, which means statistics in the majority for a certain feature (delay between packets or number of packets) are considered benign traffic while those in minority are considered malicious. The detection system consists of three components (1) *Network Probe* logs network packets and converts them into compatible format for further analysis (2) *Packet Capture* analyses network packets forwarded to it by Network Probe to determine source and destination IP, timestamp, and payloads (3) *Detector* uses the concept of intersession correlation and intra-session correlation to perform similar measurements and classify the types of traffic in majority and minority. A major strength of this work is the real-time analytics of encrypted network traffic. The experiments are performed with a small dataset, which makes it hard to generalize the obtained results. It is quite possible that the system may require some high computational capability for handling large dataset and may give more false alarms.

Wang and Paschalidis [139] present two methods (model-based and model-free) for anomaly detection in dynamic networks. Both methods are based on change-based anomaly detection, which means normal network traffic is learnt and then network traffic is monitored for any deviation from normal traffic behaviour. Any deviation found on the network is termed as an anomaly. This method uses Probability Laws (PL) for learning normal network traffic behaviour and detecting anomalies in network behaviour. In order to efficiently estimate PLs for two- dimensional data, the authors propose a two-step method that divides this complex problem into two well-explored problems – estimating PLs for one-dimensional data and set cover problem. The proposed detection mechanism is applicable to stopping an attacker who intends to steal sensitive information through SQL injection attack. A problem with such an approach is the failure to detect zero-day attacks. Similarly, the anomaly detection rate of around 60% does not seem quite promising and needs improvement.

Sigholm and Raciti [140] also address data theft in wireless networks; though their focus is on mobile ad-hoc networks deployed for military operations. They apply a clustering approach to anomaly detection, using IP and transport-layer header information as the features. This approach is focussed on monitoring data that moves across organizational network boundary and prevent it from attacks such as SQL injection. Their initial validation of this method in a simulation is effective, but the flexibility demands of a combat-environment wireless network do not seem to be fully represented in the evaluation scenario. Identifying anomalous traffic in a fast-changing ad-hoc network would seem to be a demanding problem for any classification system.

Ng et al. [141] present an approach for detection of password guessing attempts and DoS attacks by analysing network log files using data mining techniques. The log file is parsed and the information is decoded and stored back in the file. The clustering algorithm is designed and applied to see if certain connections appear multiple times. The connections appearing multiple times indicates suspicious activity. A major limitation of the proposed approach is the lack of capability to detect the attack at runtime, which makes the approach more or less impractical. The authors do not provide any experimental evaluation of the work due to which several associated aspects such as detection rate and effects on performance remain unclear.

Owezarski et al. [142] present an unsupervised algorithm, UNADA, for classification of anomalies collected from honeypots. Such classification helps to automatically modify filtering rules deployed on routers and switches. This algorithm does not need to have a signature database, look for some signatures in network traffic or perform any learning process on the network traffic. For classification of anomalies, inspiration is taken from sub-space clustering, density-based clustering and evidence accumulation which are presented in [143]. The experimental results show that the approach successfully detected DoS and flooding attacks. The strength of the work lies in its ability to detect attacks at real-time with a high accuracy rate. However, the approach is time-consuming and computationally expensive.

5.1.2.2.2. Host-based anomaly detection

Instead of monitoring the network traffic, another possible approach is to monitor the host access pattern and compare it with the normal or expected pattern of access. One of the most critical areas of access inside a host is the database that is monitored for the types of accesses and the types of functions (download and upload size, and number of data transfer). The monitoring data are recorded and compared to normal access patterns for any deviations that may be data leakage. Similar to Network-based anomaly detection techniques, Host-Based Anomaly (HBA) detection techniques also operate in three modes; supervised, semi-supervised, unsupervised. Unlike network-based anomaly detection where training data sets are prepared based on normal or abnormal network traffic patterns, the training data sets are prepared based on normal and abnormal host access patterns.

Supervised mode

Berlin et al. [144] present a malicious behaviour detection system using windows audit logs. Logs are collected from users of an enterprise and sandboxed virtual machine. The reason for latter is that logs in an enterprise do not contain malicious behaviours; that is why authors need to add malicious behaviour logs for detection purpose so they run some benign malicious binaries on virtual machine. All collected log samples are labelled as "malware" or "benign". This labelling is done using VirusTotal that runs around 55 malware engines over the samples to find whether a sample is a malware or not. After labelling, features are extracted from the audit logs. Based on these features, the audit logs are classified using regularized regression machine learning classifier. The proposed approach helps detect malware attacks. This is a supervised learning approach, which cannot detect unknown malicious behaviour. For practical application of the approach, an updated database of benign and malware is required all the time. Similarly, windows audit logs do not capture 100% details and the problem can be relegated to Microsoft to improve their audit log capabilities.

Zhang et al. [145] focus on mechanisms for protecting university credentials, which are often stolen in order to gain unfettered access to information such as scholarly publications or uncensored website access. They have built a classifier to detect compromised VPN accounts based on unusual authentication patterns (e.g., geographically distinct locations, IP addresses, and resource usage patterns) and positively evaluate it on real authentication logs from two universities. The proposed approach detects phishing and malware attacks launched to steal users' access credentials. The results show a high level of detection accuracy. A major limitation of the work is that it cannot detect phishing and malware attacks at runtime, rather, collect and analyse data from the last one to two weeks to detect compromised credentials.

Jewell and Beaver [146] propose a model of user activity based on their system call behaviour – that is, their use of the system call library to open and read files or access the network. Whilst users do not typically emit system calls themselves, their typical use of programs is captured by these patterns; at the same time, a model of system calls rather than selected execution of programs would capture maliciously-injected code that a user would be unaware of. They demonstrate the identifiability of system calls in a small experiment, and report on initial efforts at profiling the system calls involved in exfiltration attempts. The authors demonstrate that their developed system detected several attacks such as brute force attack and un-authorized copying to USB. The strength of the model is that it is environment-neutral, which means it can cope up with any operating system or operational conditions. Whilst the authors claim that their approach has no significant effect on system's performance, it is unclear how data analytics have been provided for analysing such large size of data (over 1 GB per hour) without any effect on system's performance.

Semi-supervised mode

Wüchner et al. [147] present an approach for detection of malware based on analysis of data flow at OS level between different entities (processes or sockets) of the system via a graph. Data gathered from system level events through runtime monitors is represented in the form of quantitative data flow graph. The generated graph consists of nodes and edges where a node represents a system entity and an edge represents a data flow. Malware detection heuristics (replication, manipulation and quantitative) are defined based on malware behaviour databases [148] and academic malware analysis report [149]. If any entity or data flow in the graph matches any of the three heuristics, it is detected as malicious. The proposed approach outclasses the related approaches for malware detection in terms of time and performance efficiency. Based on prototypical implementation, the approach achieved a detection rate (96%); however, the dataset of malware (44 malware) was quite small. A large database of malware behaviour requires a great deal of effort.

Kim et al. [150] point out that monitoring the activity of a database can quickly generate huge bodies of data which are difficult to effectively analyse for anomalies. They propose an implementation of a density-based outlier detection technique. They evaluate a number of implementations of the Local Outlier Factor (LOF) algorithm, with the greatest reduction in processing time coming from the combination of a kd-tree index of neighbours and the Approximated k-Nearest Neighbours algorithm. The strength of the work lies in the usage of a large and real dataset (containing 297,019 records) for analysing the access patterns of a database inside an organization. Whilst the authors reduced the computational complexity of LOF algorithm for analysing the database to detect outliers, the proposed algorithm still cannot generate data exfiltration alerts at real-time.

Udd et al. [151] have leveraged a framework Bro [152] to develop an anomaly detection system for SCADA (Supervisory Control and Data Acquisition) networks. The authors have also developed a parser for traffic using IEC 60870-5-104 protocol. The developed system is a combination of two anomaly detection mechanisms; packet timing and automatic whitelisting. This system categorizes every packet into one of three whitelists; Address Resolution Protocol, pair communication or TCP control. Timing statistics are important as these are used to observe deviations in packet arrivals. The system works in three steps: (1) Parser ensures that traffic using IEC 60870-5-104 protocol is compatible with Bro framework (2) learning component categorize packets into whitelists and record timing statistics (3) detector compares each packet with the three whitelists and if it does not match with anyone, then it is considered abnormal network traffic and an alarm is raised. The authors test the developed anomaly detection system with a number of attacks such as malware, man-in-the-middle, and spoofing attack. The results show that system detected all attacks except spoofing attacks. An issue with the current form of the work is the cleaning of network data to make it compatible with bro framework.

Yang et al. [153] present an approach for collecting news stories and other reports on data stealers from online sources. The collected data is analysed to identify the behaviour patterns of data stealers for alerting organizations against the identified behaviours patterns. The process consists of five steps: (1) collection of online news stories (2) pre-processing the data (3) Extraction of named entities (Name, address, email, and age) from text (4) creating a theft record by categorizing the named entities into categories such as time, location, and loss (5) analysing the theft record to find about various aspects of data exfiltration. This approach can give individuals and organizations some insight into the behaviour of hackers. However, the approach has limitations such as not all the data exfiltration stories get public and inability to detect unknown attacks.

Unsupervised mode

DBMSs have their own known vulnerabilities when it comes to data exfiltration, particularly with regard to their common deployment as part of the web applications. Shan et al. [154] stand as an example of countermeasures which are developed specifically to prevent such misuse through SQL injection attack. They suggest that a "protective shell" be introduced into the DBMS which learns about the legal and illegal query strings for a given application. This additional protection means that anomalous queries injected by an attacker can be automatically rejected without any particular need for the application programmer to implement protective measures. Since the protective shell has to learn both legal and illegal query strings, such an approach will not be able to stop zero-day attacks. It is very likely that the introduction of this new layer affects query performance; the paper neither talks about specific performance issues nor provides any evaluation details.

Flood and Keane [155] propose a similar approach in the context of cloud services. In their version, a Finite State Machine is built from observations of training data. Rather than merely preventing data access, however, the authors propose an active countermeasure which allows cloud vendor to automatically deploy 'spores' of malicious code to query the clients (browsers) which are involved in deviations from expected behaviour, with the aim of gathering information on attackers trying to access confidential data.

5.1.2.3. Network + host-based anomaly detection

Some of the data exfiltration detection approaches take into account both the network logs and the host logs to identify abnormal behaviour. For example, Ramachandran et al. [156] develop a data exfiltration countermeasure system which relies on building a behaviour model of network traffic. They train their model on correlations between activity on a host (abnormal CPU and memory utilisation) and activity in the network, the grounding principle being that unusually-sized data transmissions are suspicious exfiltration behaviour when not correlated with the other indicators of hosts engaged in productive activity. The reported experimental results are very brief and surficial, which leaves several aspects unclarified. For example, the performance effects of

logging and analysing such a huge amount of data are not mentioned. The authors tested the system with only a single attack (file transfer) that is not very convincing.

There are trade-offs between monitoring network traffic and monitoring host storage systems. For example, individually innocuous actions on many machines within an organisation might add up to a breach of legitimate access, which a host-based analysis system would not detect but a network-based monitor would consider suspicious. Myers et al. [157] suggest a distributed system of cooperating servers work together to correlate log information and trigger rule-based responses. The proposed approach is applicable for a large organization with sufficient resources as it needs large storage and high computational capability.

Ahn et al. [158] present a model for detecting future unknown Advanced Persistent Threats (APTs) attacks based on analysis of data gathered from various sources. The proposed model consists of four steps: (1) Data collection: data is collected from most of the available resources that includes log files, database, network behaviour, anti-virus, and status information (2) Data processing: collected data is processed to make it meet certain requirements (format and compatibility) (3) Data Analysis: processed data is analysed using clustering, predictions, and behavioural analysis to find out about user behaviour, system status, network traffic and misuse of files (4) Result: if any abnormal behaviour is detected, administrator is alerted about it. The authors expect that this model can be considered as the basis for detecting malware, phishing and SQL injection attacks launched as part of a large APT. The need for collecting and analysing data from so many sources can have severe effects on a system's performance but there has been no information available on the performance issues.

Rajamenakshi and Padmavathi [159] present a system based on integration of behavioural analysis and access policy for detection of data exfiltration. The proposed system consists of three layers: (1) Data Collection layer collects data from host (file system logs and kernel logs), network (incoming and outgoing packets) and vulnerability profiles (created based on vulnerabilities scanner results) of vulnerable applications and processes on the host. (2) Exfiltration Detection Engine (EDE) analyse collected data for detecting data exfiltration (3) the last layer is the Presentation layer that is used by system administrators to visualize the threats and alerts. The authors attacked the system using various data exfiltration attacks such DNS attack, SYN flooding, port scanning, and a malicious file transfer, which the proposed approach detected successfully. However, the experimental results do not specify the details about detection accuracy or storage and computational requirements for the detection system. The approach is not capable of detecting data exfiltration via covert channels.

Carvalho et al. [160] present an approach for real-time detection of attacks on an organization's network based on analysing a large amount of malware data gathered from various resources on daily basis. The proposed system, OwlSight, works in seven steps; (1) OwlSights collects malware data from both external (Social media data) and internal sources (organizational network flow and logs) (2) Gathered data is brought into a single format (3) Data is clustered into events based on similarity (4) Duplicate copies are removed (5) Data is enriched by analysing it for locations, numbers, names, and URLs (6) Data for each event is stored in a separate database (Email DB, and Social media DB) (7) Big data analysis engine is used to analyse these databases and show real-time alerts to the security administrators of an organization. This work provides a promising platform for a fight against malware attacks. However, for practical adaptation of the platform, some rigorous experiments need to be conducted to evaluate the performance of the platform.

Gao et al. [161] present a framework, Haddle, for on-demand data collection and data analysis to find the leaked data and the attacker. The proposed model consists of two components; Data collector and Data analyser. Data collector collects data from file logs, network logs, and process logs. The Data analyser analyses the collected data to detect attacks based on four dimensions (abnormal directory, abnormal user, suspicious block proportion, and abnormal operation). Directories in Hadoop are fixed and if any directory in file dataset is found out of range, an attack may have happened, and corresponding blocks of data can be found by analysing records containing such an abnormal directory. The proposed framework is evaluated with a single attack scenario, which is insufficient to make the results generalizable. The paper provides high-level details and misses important details such as how data from various sources is brought into a standard format for further analysis.

A summarized view of detective countermeasures is presented in Table 6.

Table 6. List of detective countermeasures

Category	Paper	Data state	Contribution	Provide Protection against
Known channel inspection	Hamid & Abawajy [104]	In use	Creates clusters of malicious profiles based on extracted features to detect phishing contents in email.	Phishing attacks
	Jaskolka et al. [105]	In transit	Correlates data inside header of outgoing data packet on a specific channel with confidential data to find it any confidential data has been leaked	Protocol-based covert channel
Inspecting steganographic traffic	Francis-Christie et al. [108]	In transit	Detects and prevents exfiltration of hidden data in a video.	Video-based covert channel
	Liu et al. [109]	In transit	A DLPS which identifies the application generating particular traffic, generating and detecting signatures for outgoing data and detecting steganographic exfiltration.	Audio-based covert channel
	Hussain & Pandey [110]	In transit	A technique using which any data packet under attack in network immediately informs communicating parties that I am under attack using alert message concealed in reserved bits	Passive Monitoring
Inspecting encrypted traffic	Schiff & Schmid [111]	In transit	Places a server for monitoring outgoing traffic which decrypts data and inspects for sensitive data	SQL Injection attack, XSS attack, Phishing attacks
	Al-Bataineh &White [112]	In transit	Leverages encryption as an opportunity instead of a challenge for detection of data exfiltration	Malware attacks
	Rohloff [113]	In transit	Detection and prevention of sensitive data being leaked in outgoing encrypted network traffic	SQL Injection attack, Phishing attacks, Passive monitoring,
	Koch [114]	In transit	Presents three possible approaches for handling detection of leakage in encrypted communication	SQL Injection attack, XSS attack, Phishing attacks, Passive monitoring, Timing channel attack
	He et al. [115]	In transit	Detects exfiltration of data in encrypted form by correlating origin of data transmission with the user profile to identify if such a transmission can be expected or not.	SQL Injection attack, XSS attack, Phishing attacks,
Inspecting unencrypted traffic	Jiang et al. [117]	In transit	A DLPS inspired from content scanning engine which monitors outgoing traffic for sensitive information	SQL Injection attack, XSS attack, Malware attacks, Phishing attacks
	Chae et al. [118]	In transit	Leverages privacy data removing technology to remove sensitive data from an outgoing file in a P2P network	Malware attacks
	Subhashini & Rani [119]	In use	Presents method for calculating score for each term and thereby detecting confidential terms in a document. This method can be used in other DLPs	Malware attacks
	Peneti et al. [121]	In transit	Attaches a time stamp with each document and later monitors the time stamp of each outgoing document. If time stamp of outgoing document is old, so it is allowed to leave else transmission is blocked.	SQL Injection attack, XSS attack, Phishing attacks, Malware attacks
	Priebe et al. [122]	In transit	Trust-based approach among tenants of cloud for detecting accidental leakage of data	VM level attacks, Sybil attacks
NBA Supervised	Xiang et al. [125]	In use	Presents a fast Extreme Machine Learning technique for helping IDS to deal with big data.	Brute-force attack, malware attack, Rootkit attack
NBA Semi-supervised	Koch & Rodosek [128]	In transit	User features are extracted from network traffic and compared with already created user's profiles. Any user whose profile doesn't match with already existing profiles is considered an attacker.	Malware attacks
	Marchetti et al. [129]	In transit	Extracts features from network traffic and generates a rank list of hosts based on suspicious score	Advanced Persistent Threat
	Wang et al. [130]	In transit	Detects abnormal traffic behaviour based on payload size and duration	
	Riecker et al. [131]	In transit	Monitors the energy consumption of network nodes and any abnormal consumption is considered malicious	Flooding attack, Black-hole attack
	Pu et al. [132]	In transit	Detection of data exfiltration based on three different behaviour-based criteria	Malware attacks
	Jagasia & Huang [133]	In transit	Identifies nodes in the network which transmit abnormal size data and are termed as malicious	Passive monitoring, Timing channel attack
NBA Unsupervised	Rastegari et al. [135]	In use	Features are extracted from the network traffic and then rule learning technique is used to develop rules for intrusion detection	Port Scanning attack
	Benham et al. [137]	In transit	Behaviour-based detection system which starts from zero and slowly gets trained based on user's behaviour. This system is designed to be less sensitive to legitimate abnormal traffic.	Advanced Persistent Threats
	Koch et al. [138]	In transit	Extracts features from network traffic and perform majority and minority statistics. Statistics in minority are considered malicious	SQL Injection attack, Brute-force attack
	Wang & Paschalidis [139]	In transit	Learns normal traffic behaviour and then monitors deviations. Any deviation from normal network traffic is considered malicious	SQL Injection attack
	Sigholm & Raciti [140]	In transit	Behaviour-based detection based on clusters formed from features (IP, header information) in an ad hoc wireless network.	SQL Injection attack
	Ng et al. [141]	In transit	Analyse network logs files to detect brute force attack	Brute force attack, DoS attack
	Owezarski et al. [142]	In transit	An approach for classification of anomalies which helps in deployment of filtering rules on routers, switches to detect data exfiltration	Flooding attack, DoS attack
HBA Supervised	Berlin et al. [144]	In use	Detect malicious behaviour based on analysis of windows logs	Malware attacks
	Zhang et al. [145]	In use	Technique for identification of stolen authentication credentials based on features (geographic location, IP etc.) using machine learning	Malware attacks, Phishing attacks
	Jewell & Beaver [146]	In use	A technique for creating profiles of legitimate and malicious system calls and comparing future system calls with these profiles to decide whether they are legitimate or malicious	Brute force attack, unauthorized copying

HBA Semi-supervised	Udd et al. [151]	In transit	Packet arrival time is used for detection. Any packet that deviates from expected time arrival specify malicious activity.	Malware attack, Man-in-the-middle attack, spoofing attack
	Wüchner et al. [147]	In use	Correlates OS level data flow with already defined malware behaviour and any deviation is recorded as malicious	Malware attacks
	Kim et al. [150]	In use	Monitors database activity in a more effective manner using density-based outlier for detecting data exfiltration	SQL Injection attack, XSS attack
	Yang et al. [153]	In use	A technique for collection of news stories and other reports on data stealers from online sources and performing statistical analysis to identify the behaviour patterns of such data stealers and help organizations to be careful against such behaviours	
HBA Unsupervised	Shan et al. [154]	In use	A technique for training database to protect itself from SQL injection attacks	SQL Injection attack
	Flood & Keane [155]	In use	A framework that helps in detection of data exfiltration in multi-cloud system	VM level attacks
Network + Host-based Anomaly	Ramachandran et al. [156]	In use/in transit	Correlates the CPU activity with network activity and any large size data transfer which doesn't correlate with host CPU is considered exfiltration of data	SQL Injection attack, XSS attack, Malware attacks, Phishing attack
	Myers et al. [157]	In use/in transit	A combination of network and host-based analysis to detect exfiltration	Malware attacks
	Ahn et al. [158]	In use/in transit	A model for detecting APTs based on logged data gathered from both network and host resources	Malware attack, SQL Injection attack, Phishing attack
	Rajamenakshi & Padmavathi [159]	In use/in transit	A framework for gathering and analysing data from network and host resources for detecting data exfiltration	DNS attack, SYN flooding, and port scanning
	Carvalho et al. [160]	In use/in transit	Framework for gathering network and host data and analysing it for detecting data exfiltration.	Malware attacks
	Gao et al. [161]	In use/in transit	A framework for gathering and analysing network logs, file logs and process logs to detect data leakage	

5.1.3. Investigative countermeasures

After successful data exfiltration, it is usually not possible to reverse its impact. However, there ought to be systems that can identify when, how, and who exfiltrated data. Such forensic analysis can be valuable for fixing security weaknesses and taking appropriate actions after a breach. Investigative countermeasures investigate data exfiltration incident to identify how and when data might have been leaked and wherever possible such countermeasures can help trace attacker(s). Investigating a data exfiltration incident can help in mitigating the effects of an attack. Information gathered during such investigation can also be useful for other enterprises. Of course, at times it may not be possible for an enterprise to share such sensitive information with broader community due to confidentiality reasons. However, sharing the leads that may not violate an enterprise's security policies would be quite fruitful for other enterprises. Investigative countermeasures found in our review has been further divided into three categories – Watermarking, Probabilistic Forensics, and Counter-Intelligence Operation. Fig. 12 shows the papers pertaining to each category in investigative countermeasures.

5.1.3.1. Watermarking

Watermarking is a technique that is often particularly useful in post-hoc detection of exfiltration vectors, as well as holding previously discussed benefits for in-time detection at organisational perimeters. Watermarking is popular for protecting copyrights and preventing from illegal copying of data. The relatively simple measure of a small signature inserted into documents for identification purposes can help a distributor trace how data found in the public domain was leaked.

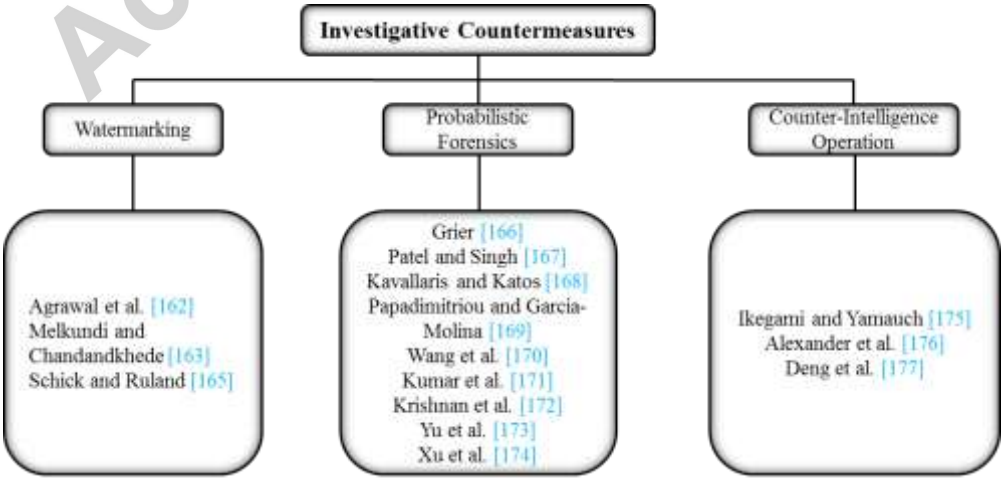


Fig. 12. Papers pertaining to each category in Investigative countermeasures

Agrawal et al. [162] motivate the need for watermarking techniques that are applicable to relational databases. They highlight the challenging nature of this problem as databases require watermarks both be unobtrusive and spread across multiple fields and collections that may be arbitrarily reordered or from which only a subset may be selected– and with possibly arbitrary restrictions on the types of manipulation permitted in each field. They go on to design a technical approach to watermarking relational databases wherein randomly-selected bit values in certain tuples of numeric data are specified and altered, with random generation using a secret key as the seed. They demonstrate that this system is robust to both detection and various evasion techniques with the help of experimental evaluation on a sufficiently large dataset, and note drawbacks such as the possibility of additively watermarking datasets and small differences introduced into the original data. The proposed technique is applicable only for numeric data and cannot watermark textual data.

Melkundi and Chandandkhede [163] proposes a watermarking technique for watermarking both textual and numeric data in a relational database to address the issue of data ownership and copyright. The technique also is capable of verifying extracted watermarks by comparing them with original watermarks using Levenshtein distance algorithm. Unicode characters [164] are used to watermark textual data as these characters provide invisibility feature. For watermarking numeric data, LSB of the binary form of numeric attribute is flipped. The reported experimental results are quite convincing as the proposed technique detected around 98% of attempts made to insert, delete or alter the watermark. This technique inserts watermarks in every attribute of every tuple, therefore, it is very unlikely that any attempt for insertion, deletion or alteration of a watermark will go undetected. However, it is unclear that to what level this watermarking process is automatic as manual watermarking in such depth would be a highly labour-intensive job.

Schick and Ruland [165] apply watermarking to the non-repudiation of forwarding information – preventing a receiver of protected data from denying that they had access to it, and allowing a provider of data to track the last party granted access to a particular copy of some data. They manage this through a data transfer process, which transmits encrypted data. The decryption process produces two files: an incremented encrypted file, which the user will use to pass on the data to another party, and a visibly watermarked file intended for viewing, with the watermark tied to the key used for decryption. A data distributor finding a file in public will thus be able to detect which authorised party last had access to it from this watermark. The work presented in this paper is superficial and needs comprehensive testing in a real organization. The ideas are limited to restricting the illegal forwarding of image data and would not work for other forms of data.

5.1.3.2. Probabilistic Forensics

This category contains countermeasures that reconstruct a digital activity, which might have led to data exfiltration. A digital activity is constructed from the traces (such as filesystem metadata) logged in a system. These countermeasures leverage probabilistic approaches to reconstruct a digital activity from the traces. The reconstructed digital activity is analysed to identify the timing, techniques employed, and party (attacker) behind a data exfiltration attack. Probabilistic forensics are further explained in the following countermeasures.

Grier [166] focuses on the detection of whether a file has been copied. Copy operations usually leave no trace within files systems investigated as access timestamps does not distinguish copy operation from other forms of file access. Grier attempts to disambiguate these situations by examining the differences in timestamps caused by folder copy operations and other accesses, the former notably leaving traces in the form of a cut-off date for the earliest timestamp in a folder. Grier demonstrates a method in a case study, where he detects copying which occurred roughly two months prior to the investigation. However, this approach needs to be tested in a real-world file system to determine its accuracy. The current form of the approach does not distinguish legitimate and illegitimate copy operations, which is addressed by Patel and Singh [167] who developed a fuzzy inference system for filtering on this detection methodology.

Kavallaris and Katos [168] use a similar methodology to defend against pod-slurping attacks, where data has been covertly copied to a USB device. They base their method on the access timestamps of files and the file transfer rates of devices, demonstrating that, with specific countermeasures, host systems storing confidential data can not only detect but protect themselves against duplication to unauthorised devices. The approach seems quite suitable for preventing copying of data to an unauthorized USB device. However, the detection rate (50%-

60%) is unimpressive and more sophisticated statistical models need to be incorporated in the approach for improving the detection rate of malicious copying.

Papadimitriou and Garcia-Molina [169] focus on the 'traitors tracing problem', where trusted third parties may leak data, which have been shared with them, and a distributor must decide which party is to blame. They specifically consider schemes where the original data cannot be 'perturbed' – made less sensitive through the partial obfuscation of some content, or 'watermarked' – tagged with an identifier. They develop an agent guilt model to decide upon the likelihood of one or more of the third parties being responsible for a leak, based on the rarity of the data, the amount of data leaked, and the estimated probabilities of the data being accessed by other means. They cover variants of the problem where fake information objects may be randomly provided along with requested data to act as agent-level watermarks. Such an approach may give birth to a level of mistrust between the data owner and the third party, which may have long-term business consequences. The approach is based on probability and tracking result may catch the party not actually responsible for data leak. In that case, the data owner may face some serious case of blaming the wrong party for data exfiltration.

Wang et al. [170] model the detection of a traitor as an incremental process of refinement of a possible set of suspects – appearing to leave aside the possibility of the data being obtained through other means – and simulate the traceability impact of a data distributor adopting a uniformly random data allocation scheme. Whilst this result may be useful groundwork, random distribution of data to possible trusted parties seems unlikely in a business context, where specific data sharing relationships as proposed in [169].

Kumar et al. [171] compare a number of data leakage detection and data allocation strategies for this same 'traitors tracing' problem, though note that the leakage may be accidental. In modelling the problem, they address allocation where only a sample of a total dataset is required. In the sample formulation, they consider random allocation, overlap allocation (in which the least-distributed data items are released), and an allocation scheme that minimises the maximum relative overlap between any two agents (reduces overlap for the requesting agent). The authors implemented the algorithms and concluded that scheme that minimises the maximum relative overlap between any two agents should be adopted for increasing the chances of tracing the data leaker. The experiments are performed with a very small dataset. For more comprehensive comparison of these algorithms, these experiments need to be performed with a large and generalized dataset.

Concerned with the availability of trustworthy data on accesses to possibly compromised hosts, Krishnan et al. [172] move the monitoring of hosts to the hypervisor, with a forensic audit log which records disk accesses, system calls (to the host) and chains of access to memory across processes. They build a query interface for this audit log to allow for richer mining operations and go on to build a proof-of-concept implementation which selectively blocks the guest OS from sending network packets which contain monitored data, and also identifies applications involved in attempting to send packets or install such applications. A limitation of this work is the security of the hypervisor itself. If the hypervisor's security gets compromised, it will lead to undermining the integrity and confidentiality of all logged data.

Yu et al. [173] apply both static analysis and runtime tracking to the problem of tracing the propagation of sensitive data, with the aim of detecting malicious information flows in programs– which are not necessarily due to exploits. They implement this as a kernel module for dynamic tracing combined with a pre-processing phase for static analysis and test its performance with a password stealing attack and SQL injection from an ordinary web server configuration. Whilst successful, and with limited impact on system performance, they note that this method requires rules to be manually written regarding sensitive data.

Xu et al. [174] present an investigation technique based on Bayesian Belief Network (BBN) to identify the path and responsible party for leakage of data. BBN helps investigators in characterizing a relationship between hypothesis and evidences to establish a conclusion about the path and the responsible party for data exfiltration. BBN works according to conditional probability, which means the more the evidence to support a hypothesis, the more likely it is that the hypothesis is true and ultimately more accurate will be the conclusion. According to the proposed technique, after the construction of BBN for data exfiltration, investigators apply the Normalized Likelihood analytical approach to identify most probable targets. This technique helps direct investigation in the right direction at a very early age but requires investigators to fully analyse and understand the underlying hypothesis. The proposed technique might not be applicable for small organizations with limited budget and resources, as this technique requires expert investigators having access to many resources for collecting evidence. In addition, the proposed idea needs some rigorous evaluation.

5.1.3.3. Counter-Intelligence Operation

Counter-Intelligence is incorporated by enterprises to track back an attack towards an attacker. In this operation, an attacker is usually indulged with fake data and during this process; information is gathered about the attacker. Sometimes tools are employed to target the machine of an attacker based on the information gathered during the attack and retrieve useful information about the adversary party. Counter-intelligence operation is explained in the following countermeasures.

Ikegami and Yamauch [175] propose a counter-intelligence approach for identifying an attacker who tries to steal classified information. The classified information inside a computer is replaced with dummy data that contains an embedded exploration program, which has the capability to collect and transmit information from an attacker's computer back to a specified investigation system. The dummy function is called as soon as there is an attempt to steal the classified information. When the stolen dummy data reaches an attacker's computer, the exploration program embedded in dummy data starts collecting and sending information about an attacker's computer back to the investigation system. The attacker is identified by analysing the obtained information. However, there has been no formal evaluation of the proposed approach; hence, it is not clear how to replace the same exact file to be leaked with a dummy file. The approach requires sufficient information (OS and rootkit) about an attacker's computer, however, most often such information is not available.

Alexander et al. [176] take this further, suggesting a counter-intelligence operation be conducted to trace an intrusion back to its source and deploy malware designed to reveal the identity, objective and success of an attacker. They survey rootkit techniques which would be suitable for deployment in such an operation, ranking them on their stealth, information access and data exfiltration capabilities, and develop a new rootkit technique for their domain. The idea of hiding the tracking tool on the target computer is interesting; however, the implementation of both tracking and hiding tool still remains open issues.

Deng et al. [177] propose another counter-intelligence approach, Tracing and Revoking Leaked Access Credentials (TRLAC), for tracing those who illegally distribute access credentials of a cloud application and initiate a revoking mechanism for depriving them of access rights. According to this approach, the credentials of each authorized user is encoded using identity-based broadcast encryption [178] and optimal fingerprint codes [179]. Authorized users are divided into groups and their credentials are encrypted with the data to be outsourced. In case credentials are leaked, TRLAC starts a two steps process of tracing the culprits; 1) a tracing technique traces code-word associated with illegal credentials. 2) the tracing technique finds the set of code-words associated with culprits whose access rights are revoked. The experimental results are quite convincing, however, the approach involves a lot of manual effort before outsourcing to a cloud-based infrastructure. A summarized view of investigative countermeasures is shown in Table 7.

Table 7. List of most relevant investigative countermeasures

Category	Paper	Data state	Contribution
Watermarking	Agrawal et al. [162]	At rest	A technique for watermarking textual data in relational database
	Melkundi & Chandandkhede [163]	At rest	A technique for watermarking both textual and numeric data in a relational database
	Schick & Ruland [165]	At rest	An approach that leverages watermarking technique to track the authorized user who shared data illegally.
Probabilistic Forensics	Grier [166]	In use	A method to find out whether a file has been copied or not based on examination of file system.
	Patel & Singh [167]	In use	A technique for providing ease in differentiating between copy operation and rest of the operations
	Kavallaris & Katos [168]	In use	An approach for preventing data being copied to an un-authorized USB device
	Papadimitriou & Garcia-Molina [169]	In use	A model for identification of guilty trusted party who has leaked data
	Wang et al. [170]	In use	A technique based on probabilistic analysis to identify the insider who leaked data
	Kumar et al. [171]	In use	An approach for smart data allocation which helps in tracing the data leaker
	Krishnan et al. [172]	In use	An approach for post-exfiltration analysis to identify what, when and how based on reconstruction of events using logged data from the hypervisor
	Yu et al. [173]	In use	A technique for identification of the path used for exfiltration of data using statistical analysis
	Xu et al. [174]	In use	A technique based on probabilistic analysis to identify path and responsible party for data leakage
Counter-Intelligence Operation	Deng et al. [177]	In use	An approach for tracing the traitor who illegally distribute access credentials of a cloud application
	Ikegami & Yamauch [175]	In use	An approach that detects, prevents and tracks the attacker
	Alexander et al. [176]	In use	A technique for hiding detective and investigative software from the attacker using rootkit

5.2. Mapping of Attack Vectors to Countermeasures

Fig. 13 shows the mapping of the attack vectors identified in section 3 onto the countermeasures reported in section 4.3. This mapping provides a reader with an understanding of which attack vectors are addressed by which countermeasures. It is imperative to mention here that some of the preventive and detective countermeasures adopt an aggressive and proactive approach (instead of reactive) for protecting data. This means that these countermeasures focus on securing data instead of defending it against a specific attack and therefore, it is quite challenging to map such countermeasures to a particular attack vector. Such countermeasures are not shown in Fig. 13. For example, Zhuang et al. [90] propose to distribute data smartly among multiple clouds which would reduce the risk of leaking all data in a single attack. The technique proposed in [90] is more focused on reducing the risk of data leakage and do not include discussion or reference to any attacks. The countermeasures shown in Fig. 13 do not include investigative countermeasures as investigative countermeasures presented in section 5.1.3 report countermeasures for investigating data leakage in a generic sense and are not specific to particular attack vectors. The countermeasures that address attack vectors other than those reported in Section 3 (such as brute-force attack and flooding attack) are also not shown in Fig. 13.

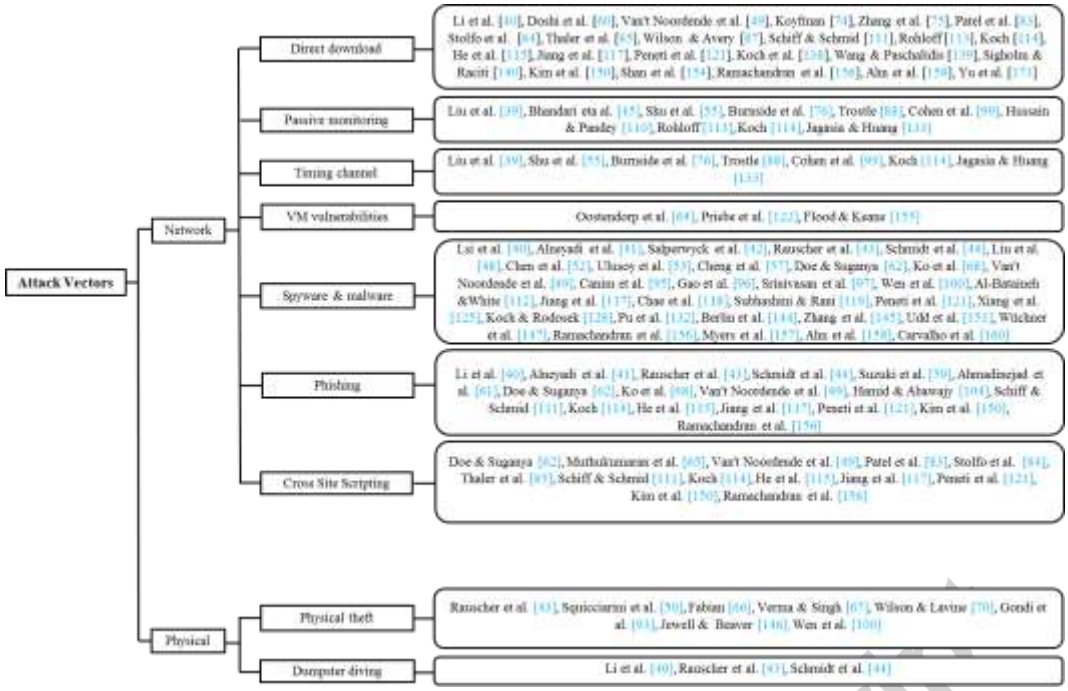


Fig. 13. Mapping of Attack Vectors to Countermeasures

6. RQ3. Open and Future Challenges

This section reports the findings from analysing the data extracted to answer RQ3, "*What are the challenging issues in preventing, detecting and investigating data exfiltration?* Cyberspace is vulnerable to a variety of attacks. According to one statistics, the total annual cost of cybercrime is around $400 billion [180]. Data exfiltration is the main motivator for these attacks. Researchers and practitioners have developed many methods and systems to protect data from outside [6]. These systems include Intrusion Detection System, Intrusion Prevention Systems, Security Information and Event Management (SIEM) system, Anti-malware, and Firewalls. Unlike IDS, IPS, SIEM, Anti-malware, and Firewalls, data leakage detection and prevention systems protect data from inside [6]. Whilst these data exfiltration countermeasures have recently attracted the attention of the research community, there exist several open and challenging issues. Fig. 14 shows some of the major open and challenging issues identified by our review that stem from the in-depth and critical analysis of attack vectors (Section 4) and countermeasures (Section 5).

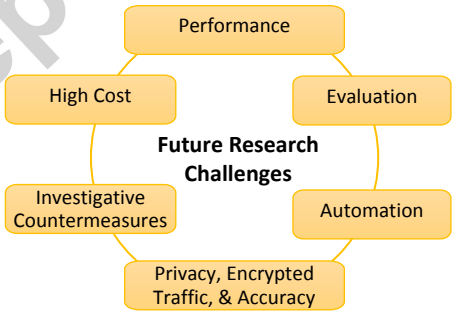


Fig. 14. Future Research Challenges in Defence against Data Exfiltration

6.1. Performance

The in-depth critical analysis of around 108 countermeasures reveals that performance is one of the most critical qualities for systems designed for preventing, detecting, or investigating data exfiltration. The fact reported in Data Breach Investigation Report [181] of Verizon Inc. endorses our observation, which states that in 86% of the cases, the evidence of data breach attack was recorded by the installed detection system, which could not analyse and report the data breaches in a timely fashion. In our review, we found a number of countermeasures (e.g., [39], [40], [41], [42], [44], [48], [52], [53], [55], [57], [76], [83], [84], [86], [87], [95], [97], [112], [113],

[115], [118], [142], [145], [150]) that perform poorly in terms of response time and resource utilization. The major reason for such poor performance is the large size, high speed, and heterogeneous nature of data dealt with by these systems. For instance, an enterprise as large as HP generates 1 trillion events per day or around 12 million events per second [182]. It is imperative to collect and analyse this big data in real-time and without causing significant delay in any data transmission process. Hence, there is a significant potential for further research that can help address the performance issues in these systems. We assert that research is required to investigate the incorporation of Big Data tools and technologies (such as Hadoop and Spark) in Data Leakage detection and prevention systems. Another potential direction for improving performance is the classification of data based on the sensitivity level (to separate sensitive data) and then dealing with each sensitivity level accordingly. Whilst our review reports some work on data classification in the context of data exfiltration (Section 5.1.1.1), we assert that more research will be productive in this area. Furthermore, the selection of features from security event data is another potential approach for improving performance and it needs to be investigated that how feature selection tools and technologies can be developed and incorporated in defence against data exfiltration attacks.

6.2. Evaluation

In our review, we observed that quite a large number of countermeasures are either not evaluated at all ([23], [40], [45] , [48] , [49] , [62] , [75] , [88] , [91] , [67] , [92] , [105] , [110], [111], [114], [137], [141], [154], , [160], [174], , [175]) or evaluated weakly ([60], [70], [100], [109], [132], [138], [147], [158], [161], [165], [171]). We consider an evaluation as weak evaluation when the system is evaluated with a small dataset (e.g. [132], [138], [147], [171]), or just one type of data (e.g. [70], [100], [165]), or restricted to a specific case (e.g. [109], [160]). Without rigorous evaluation, the claims made about a countermeasure would unlikely be taken seriously. One solid reason for the absence of such rigours evaluation in a number of studies is the lack of a generic evaluation framework. We strongly emphasis that a generic framework needs to be developed for the assessment of data exfiltration countermeasures that can guide researchers and practitioners on how to evaluate their systems. One important aspect for such a framework should be the consideration and development of a comprehensive dataset. We found that there exist some datasets ([119], [121], [150], [162]) for evaluating data exfiltration countermeasures, however, these datasets are small in size and limited in quality. These limitations make the datasets unable to reflect the actual strengths and weaknesses of the proposed countermeasure. It is also important to keep updating such a dataset with evolving patterns of normal and malicious activities so that a proposed countermeasure can be evaluated with a variety of established and newly introduced attacks. Another pressing issue that the evaluation framework needs to consider is the assessment of Advanced Persistent Threat (APT). Since APTs are highly sophisticated and planned attacks operated under low and slow mood, it is quite challenging to reasonably launch such attacks and test the effectiveness of a countermeasure in the face of these attacks as evident from the experimental evaluation in [129] and [158]. We assert that evaluation framework should be able to guide for evaluating a countermeasure with APTs. For improving the standard of evaluation, we also encourage close collaboration between academia and industry. Such a collaboration would enable the academicians to gather feedback from industry about their approaches and develop systems in collaboration with industry that can help evaluate such system on real-world projects.

6.3. Automation

Our review reveals that a number of countermeasures ([39], [43], [45], [68], [86], [91], [99], [128], [163], [174], [177], [173]) require manual efforts during deployment and operations such as (a) having a dedicated network administrator to promote a node from one security zone to another based on the sensitivity of the node [43]; (b) involvement of user for approval of each single data transfer out of user's computer [68]; (c) manual addition of dummy records to the database for cyber deception [86]; (d) manually dividing a single table into several tables [91]; (e) involvement of expert investigators for manually gathering evidence of data leakage to identify the data leaker [174]; (f) manual writing of rules regarding sensitive data [173]; and (g) semi-automatic deployments [39]. A lack of automation impacts performance of the overall system in terms of deployment and response time. The involvement of network administrator and investigating experts increases the cost and makes the incorporation of such systems quite challenging for enterprises. Similarly, personal users are often reluctant to pay attention to security alerts and approvals during data transmission. We assert that there is an important need for future research to bring automation into the deployment and operation of the software systems designed for preventing, detecting, or investigating data exfiltration. The dependency on a dedicated human should be reduced to a minimum to make these systems more acceptable for enterprises and personal use.

6.4. Privacy, Encrypted Traffic, and Accuracy

With respect to data exfiltration countermeasures, the three terms (i.e. Privacy, Encrypted Traffic, and Accuracy) are closely related. In our review, we found a number of studies such as [105], [108], [117], [119], [121], [169] that are set in violation to users' privacy. These countermeasures directly monitor the outgoing network traffic generated by users and scan it for detecting sensitive information. Whilst these data exfiltration countermeasures are in most cases installed by enterprises to defend themselves against data exfiltration, such approaches lead to a mistrust between users' and their corresponding enterprises. To address the privacy and security concerns, the approach of encrypting data before sending it out is broadly adopted. However, encrypting data makes the countermeasures, which rely on examining the content of outgoing traffic, unable to detect any malicious data exfiltration. To address this issue, countermeasures have been developed as reported in our review ([111-115]) that can examine the encrypted traffic to detect data exfiltration. However, these countermeasures are facing serious issues of accuracy as it is quite challenging to accurately detect data exfiltration by only examining the ciphered data. We are of the view that the adoption of encrypted traffic protocols is on such a broad scale, that it is unrealistic to opt for their removal or resist their adoption. Apart from their wide-scale adoption, we also believe that it is quite risky not to adopt encrypted traffic transmission approach because it leaves open the option of data exfiltration via passive monitoring. Based on our analysis, we assert that a good deal of exploration is required to address the accuracy concerns of detecting data leakage and prevention systems that examine encrypted network traffic.

6.5. Investigative Countermeasures

In our review, we analysed and categorized data exfiltration countermeasures into preventive, detective, and investigative categories. As evident from Fig – 2, research community primarily remains focussed on preventive and detective countermeasures. Whilst the preventive and detective countermeasures are quite crucial for fighting data exfiltration, we believe investigative countermeasures are equally important. Investigative countermeasures help in identifying the medium used for exfiltration and in tracking attackers. Such information is quite critical for securing an individual and an organization from data exfiltration in future. Information gathered during such an investigation also helps in mitigating the effects of data exfiltration incident. Furthermore, identifying and prosecuting attackers sends a very strong message to other potential attackers that there are systems in place to track and catch them. Hence, we believe that there is an urgent need of investing more efforts in developing and evaluating investigative approaches for identifying *how*, *when* and *who* in the aftermath of a data exfiltration attack. We also encourage the research community to explore the true potential of investigative countermeasures to enable them to accurately identify *how*, *when* and *who* as well as guide for recovery and mitigating the effects of data exfiltration.

6.6. High Cost

Cost is one of the primary concerns both for individuals and especially enterprises while deciding upon the incorporation of a particular system, tool, or technology in their infrastructure. By cost, we mean the expenditure to develop, operate, and maintain a system. In our review, we found a number of countermeasures ([42], [83], [84], [86], [87], [90] , [111], [122], [174]) whose cost make them very much unlikely to be incorporated by enterprises. The incorporation of these countermeasures leads to high costs in a number of ways such as (a) leveraging a specialized hardware for storing sensitive data [42]; (b) installing SGX hardware for implementing security rules [111]; (c) installing tag monitors via all paths by cloud provider for monitoring security tags [122]; (d) high dependency on security experts [174]; and (e) storing negative data in a database for cyber deception [83] , [84] , [86] , [87] , [90]. We assert that the incorporation of specialized hardware should be discouraged in the design of data exfiltration countermeasures as deploying and maintaining hardware on a large scale would be very much unfeasible for enterprises. Similarly, a thorough investigation is required to explore ways of reducing the cost for storing and maintaining negative data in cyber deception approaches.

7. Limitations

As mentioned in Section 4, the attack vectors used by external attackers to steal data are not limited to what is presented in section 4. There are two reasons for such a limitation: (1) There exists a large number of attack vectors as reported in [14-16] and covering all of them is quite challenging (2) Our review is focussed on

countermeasures and not on attack vectors. The motivation for including attack vectors is to contextualize the discussion on the countermeasures. Therefore, we only included those countermeasures that are most frequently reported in the countermeasures. Similarly, a wide range of literature exists that directly or indirectly address data exfiltration in various domains (mobile computing, IoT devices, Printers); it may not be possible for a single review like ours to cover all such literature. For instance, a wide range of papers exist on access control or encryption but it was not the intention of this review to include all those studies. However, during the selection of the studies for this review, we tried to include most recent and most relevant literature on data exfiltration. Having said that, we cannot ignore the limitation of human biasedness in the inclusion or exclusion of papers in literature reviews. Similarly, extending the survey by following citations from included publications could unveil larger bodies of work on exfiltration methods which did not match our queries. We opted not to pursue this extension due to the increased manual workload it would require being disproportionate to the potential gain in coverage, but this does mean that our survey is unlikely to match papers which avoid the terminology used in our queries.

8. Conclusion

Data exfiltration is a serious and ongoing issue in the field of information security. The number and capabilities of data exfiltration vectors are growing at an alarming rate. The existence and emergence of such attack vectors make the data exfiltration countermeasures critical for an organization's security. These countermeasures are designed to prevent, detect, and investigate data exfiltration attempts. This paper reports a survey of a large number of papers published on data exfiltration external attack vectors and countermeasures. The aim of this review is to systematically select and critically review the literature to understand the state of the art of the data exfiltration external attack vectors, countermeasures, and identify the gaps to motivate future research in this area. Based on the analysis of the reviewed papers, we have built a taxonomy of external attack vectors leveraged by attackers to steal data. That taxonomy has been leveraged to provide a detailed classification of data exfiltration countermeasures. The identified countermeasures have been discussed and critically analysed in terms of their contributions and limitations. We have also analysed the countermeasures with respect to their applicability against particular external attack vectors. Another critical overview provided by our review is the applicability of countermeasures for particular data state (1: in use, 2: in transit, and 3: at rest), which gives an insight into what particular data states are mostly attacked and how countermeasures protect data in these particular states. The results of this review have identified several open research areas that need to be explored for further understanding the data exfiltration attack vectors and developing appropriate countermeasures.

Our review has revealed several tensions in the design of data exfiltration countermeasures with regards to performance, encryption, privacy, accuracy, automation, evaluation, and cost. The extensive monitoring and analysis of large size network and host activities by these countermeasures requires high computational and storage capability. The lack of such capability leads to poor performance and response time. Our work highlights the critical importance of privacy and ethical concerns when developing data exfiltration countermeasures. This is pertinent given the increasing concerns over a surveillance society. Furthermore, the increasing adoption of encryption (for security and privacy) makes a number of countermeasures obsolete. The high-level dependency on human experts and hardware devices make these countermeasures very expensive to be incorporated by enterprises. Therefore, we assert on making these countermeasures more software-intensive rather than hardware-intensive and bringing more automation into the design of these data exfiltration countermeasures. Our analysis of the trends in the literature has revealed that there remains the issue of practical deployment and evaluation for most of the existing countermeasures. We believe close and intensive collaborative ties between academia and industry can help address this issue. We have observed that existing research is focussed on preventive and detective countermeasures and significant research is needed on developing investigative countermeasures, which are equally important for defence against data exfiltration attacks. Our results show that most of the countermeasures focus on protecting data in 'in use' state, therefore, the future research needs to focus on securing data in 'rest' and 'in transit' states. We hope that the insights provided in this paper will give academic researchers and industry practitioners with new directions and motivations for enhancing research and development efforts to devise, evaluate, and deploy new and innovative countermeasures for securing against data exfiltration attacks.

References

1. Hipolito, J.M., *Anatomy of a Data Breach. Available at https://www.trendmicro.com/vinfo/us/threat-encyclopedia/web-attack/110/anatomy-of-a-data-breach.* 2011.
2. Alneyadi, S., E. Sithirasenan, and V. Muthukkumarasamy, *A survey on data leakage prevention systems.* Journal of Network and Computer Applications, 2016. **62**: p. 137-152.
3. Shabtai, A., Y. Elovici, and L. Rokach, *A survey of data leakage detection and prevention solutions*. 2012: Springer Science & Business Media.
4. Raman, P., H.G. Kayacık, and A. Somayaji. *Understanding data leak prevention*. in *6th Annual Symposium on Information Assurance (ASIA'11)*. 2011.
5. Brindha, T. and R. Shaji. *An analysis of data leakage and prevention techniques in cloud environment*. in *2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. 2015. IEEE.
6. Tahboub, R. and Y. Saleh. *Data leakage/loss prevention systems (DLP)*. in *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*. 2014. IEEE.
7. Chandola, V., A. Banerjee, and V. Kumar, *Anomaly detection: A survey.* ACM computing surveys (CSUR), 2009. **41**(3): p. 15.
8. Patcha, A. and J.-M. Park, *An overview of anomaly detection techniques: Existing solutions and latest technological trends.* Computer networks, 2007. **51**(12): p. 3448-3470.
9. Garcia-Teodoro, P., et al., *Anomaly-based network intrusion detection: Techniques, systems and challenges.* computers & security, 2009. **28**(1): p. 18-28.
10. Hodge, V. and J. Austin, *A survey of outlier detection methodologies.* Artificial intelligence review, 2004. **22**(2): p. 85-126.
11. Gogoi, P., et al., *A survey of outlier detection methods in network anomaly identification.* The Computer Journal, 2011. **54**(4): p. 570-588.
12. Van Acker, S., D. Hausknecht, and A. Sabelfeld. *Data Exfiltration in the Face of CSP*. in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. 2016. ACM.
13. Braun, V. and V. Clarke, *Using thematic analysis in psychology.* Qualitative research in psychology, 2006. **3**(2): p. 77-101.
14. Gordon, P., *Data Leakage-Threats and Mitigation.* InfoSec Reading Room, 2007.
15. Giani, A., V.H. Berk, and G.V. Cybenko. *Data exfiltration and covert channels*. in *Defense and Security Symposium*. 2006. International Society for Optics and Photonics.
16. Winterfeld, J.A.a.S., *Chapter 6 - Logical Weapons.* 2014.
17. Clarke, J., *Chapter 4 - Exploiting SQL Injection.* 2009.
18. Moyle, S., *"The blackhat's toolbox: SQL injections,"* Network Security2007.
19. Revelli, A., *Chapter 4 - Exploiting SQL injection.* 2012.
20. Potnuru, M., *Limits of the Federal Wiretap Act's Ability to Protect Against Wi-Fi Sniffing.* Michigan Law Review, 2012: p. 89-117.
21. Goldman, A.D., et al. *Plugging the leaks without unplugging your network in the midst of Disaster*. in *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*. 2012. IEEE.
22. Shakarian, P., J. Shakarian, and A. Ruef, *Introduction to cyber-warfare: A multidisciplinary approach*. 2013: Newnes.
23. Ranjith, P., C. Priya, and K. Shalini, *On covert channels between virtual machines.* Journal in Computer Virology, 2012. **8**(3): p. 85-97.
24. Shah, G., A. Molina, and M. Blaze. *Keyboards and Covert Channels*. in *Usenix security*. 2006.
25. Hay, B. and K. Nance. *Circumventing cryptography in virtualized environments*. in *Malicious and Unwanted Software (MALWARE), 2012 7th International Conference on*. 2012. IEEE.
26. Bates, A., et al. *Detecting co-residency with active traffic analysis techniques*. in *Proceedings of the 2012 ACM Workshop on Cloud computing security workshop*. 2012. ACM.
27. Bates, A., et al., *On detecting co-resident cloud instances using network flow watermarking techniques.* International Journal of Information Security, 2014. **13**(2): p. 171-189.
28. Xu, Y., et al. *An exploration of L2 cache covert channels in virtualized environments*. in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. 2011. ACM.
29. Wu, Z., Z. Xu, and H. Wang. *Whispers in the hyper-space: high-speed covert channel attacks in the cloud*. in *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*. 2012.
30. Doyle, E., *Not all spyware is as harmless as cookies: block it or your business could pay dearly.* Computer Weekly, 2003: p. 32.
31. Jakobsson, M. and S. Myers, *Phishing and countermeasures: understanding the increasing problem of electronic identity theft*. 2006: John Wiley & Sons.
32. Computer and Associates, *Types of Phishing Attacks. Available at http://www.pcworld.com/article/135293/article.html.* 2007.

33. Huang, L.-S., et al. *Protecting browsers from cross-origin CSS attacks*. in *Proceedings of the 17th ACM conference on Computer and communications security*. 2010. ACM.
34. Heiderich, M., et al. *Scriptless attacks: stealing the pie without touching the sill*. in *Proceedings of the 2012 ACM conference on Computer and communications security*. 2012. ACM.
35. *OWASP. Top Ten Project. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#tab=OWASP_Top_10_for_2013*. 2013.
36. Farrow, R., *Network Defense Is Your Desktop Being Wiretapped?* NETWORK MAGAZINE-SAN FRANCISCO-, 2003. **18**(8): p. 52-53.
37. *Internet security threat report.* April 2008. **XIII**.
38. (ASPG), A.S.P.G., *The three states of digital data http://aspg.com/three-states-digital-data/#.WD-BzNV96Uk.* Major Northeast University, 2014.
39. Liu, Y., et al. *Thwarting memory disclosure with efficient hypervisor-enforced intra-domain isolation*. in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015. ACM.
40. Li, H., et al. *Leakage Prevention Method for Unstructured Data Based on Classification*. in *International Conference on Applications and Techniques in Information Security*. 2015. Springer.
41. Alneyadi, S., E. Sithirasenan, and V. Muthukkumarasamy. *Detecting Data Semantic: A Data Leakage Prevention Approach*. in *Trustcom/BigDataSE/ISPA, 2015 IEEE*. 2015. IEEE.
42. Salperwyck, C., et al. *GhostDB: hiding data from prying eyes*. in *Proceedings of the 33rd international conference on Very large data bases*. 2007. VLDB Endowment.
43. Rauscher, R. and R. Acharya. *A network security architecture to reduce the risk of data leakage for health care organizations*. in *e-Health Networking, Applications and Services (Healthcom), 2014 IEEE 16th International Conference on*. 2014. IEEE.
44. Schmidt, M., et al. *Trustbox: A security architecture for preventing data breaches*. in *2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing*. 2011. IEEE.
45. Bhandari, A., A. Gupta, and D. Das, *A framework for Data Security and Storage in Cloud Computing.*
46. Samarati, P. and S.C. de Vimercati. *Access control: Policies, models, and mechanisms*. in *International School on Foundations of Security Analysis and Design*. 2000. Springer.
47. Smyth, N., *Security+ Essentials . Available at http://www.lulu.com/au/en/shop/neil-smyth/security-essentials/ebook/product-12565906.html.* 2010.
48. Liu, D., P. Bai, and H. Jiang. *Using the user space file system to protect file*. in *Apperceiving Computing and Intelligence Analysis (ICACIA), 2010 International Conference on*. 2010. IEEE.
49. van't Noordende, G., F.M. Brazier, and A.S. Tanenbaum. *Guarding security sensitive content using confined mobile agents*. in *Proceedings of the 2007 ACM symposium on Applied computing*. 2007. ACM.
50. Squicciarini, A.C., G. Petracca, and E. Bertino. *Adaptive data protection in distributed systems*. in *Proceedings of the third ACM conference on Data and application security and privacy*. 2013. ACM.
51. Squicciarini, A.C., G. Petracca, and E. Bertino. *Adaptive data management for self-protecting objects in cloud computing systems*. in *Proceedings of the 8th International Conference on Network and Service Management*. 2012. International Federation for Information Processing.
52. Chen, Y.-Y., P.A. Jamkhedkar, and R.B. Lee. *A software-hardware architecture for self-protecting data*. in *Proceedings of the 2012 ACM conference on Computer and communications security*. 2012. ACM.
53. Ulusoy, H., et al. *GuardMR: Fine-grained security policy enforcement for MapReduce systems*. in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*. 2015. ACM.
54. Dean, J. and S. Ghemawat, *MapReduce: simplified data processing on large clusters.* Communications of the ACM, 2008. **51**(1): p. 107-113.
55. Shu, J., Z. Shen, and W. Xue, *Shield: a stackable secure storage system for file sharing in public storage.* Journal of Parallel and Distributed Computing, 2014. **74**(9): p. 2872-2883.
56. Halcrow, M.A. *eCryptfs: An enterprise-class encrypted filesystem for linux*. in *Proceedings of the 2005 Linux Symposium*. 2005.
57. Cheng, Y., X. Ding, and R.H. Deng. *Efficient virtualization-based application protection against untrusted operating system*. in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*. 2015. ACM.
58. Chhabra, S., et al. *SecureME: a hardware-software approach to full system security*. in *Proceedings of the international conference on Supercomputing*. 2011. ACM.
59. Suzuki, K., K. Mouri, and E. Okubo. *Salvia: a privacy-aware operating system for prevention of data leakage*. in *International Workshop on Security*. 2007. Springer.
60. Doshi, J.C. and B. Trivedi. *Hybrid intelligent access control framework to protect data privacy and theft*. in *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*. 2015. IEEE.
61. Ahmadinejad, S.H., P.W. Fong, and R. Safavi-Naini. *Privacy and Utility of Inference Control Mechanisms for Social Computing Applications*. in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. 2016. ACM.
62. Doe, N.P. and V. Suganya. *Secure service to prevent data breaches in cloud*. in *Computer Communication and Informatics (ICCCI), 2014 International Conference on*. 2014. IEEE.

63. Gampala, V., S. Inuganti, and S. Muppidi, *Data security in cloud computing with elliptic curve cryptography.* International Journal of Soft Computing and Engineering (IJSCE), 2012. **2**(3): p. 138-141.
64. Oostendorp, K.A., et al. *Domain and type enforcement firewalls*. in *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*. 2000. IEEE.
65. Muthukumaran, D., et al. *FlowWatcher: Defending against data disclosure vulnerabilities in web applications*. in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015. ACM.
66. Fabian, M. *Endpoint security: managing USB-based removable devices with the advent of portable applications*. in *Proceedings of the 4th annual conference on Information security curriculum development*. 2007. ACM.
67. Verma, S. and A. Singh. *Data theft prevention & endpoint protection from unauthorized USB devices—Implementation*. in *2012 Fourth International Conference on Advanced Computing (ICoAC)*. 2012. IEEE.
68. Ko, R.K., A.Y. Tan, and T. Gao. *A Mantrap-Inspired, User-Centric Data Leakage Prevention (DLP) Approach*. in *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*. 2014. IEEE.
69. He, K.K., *Kernel korner: why and how to use netlink socket.* Linux Journal, 2005. **2005**(130): p. 11.
70. Wilson, D. and M.K. Lavine. *A discretionary access control method for preventing data exfiltration (DE) via removable devices*. in *International Conference on Digital Forensics and Cyber Crime*. 2009. Springer.
71. Villanueva, J.C., *Forward Proxy vs Reverse Proxy. Available at http://www.jscape.com/blog/bid/87783/Forward-Proxy-vsReverse-Proxy.* 2012.
72. Yousef and Bakhdlaghi, *Snort and SSL/TLS Inspection. Available at https://www.sans.org/reading-room/whitepapers/detection/snort-ssl-tls-inspection-37735* 2017.
73. Alomari, M.A., K. Samsudin, and A.R. Ramli. *A study on encryption algorithms and modes for disk encryption*. in *2009 International Conference on Signal Processing Systems*. 2009. IEEE.
74. Koyfman, A. *Securing sensitive data with the ingrian datasecure platform*. in *International Conference on Financial Cryptography and Data Security*. 2005. Springer.
75. Zhang, X., et al. *Research and application of the transparent data encpryption in intranet data leakage prevention*. in *Computational Intelligence and Security, 2009. CIS'09. International Conference on*. 2009. IEEE.
76. Burnside, M. and A.D. Keromytis. *F3ildcrypt: End-to-end protection of sensitive information in web services*. in *International Conference on Information Security*. 2009. Springer.
77. Huang, Z., et al. *GenoGuard: Protecting genomic data against brute-force attacks*. in *2015 IEEE Symposium on Security and Privacy*. 2015. IEEE.
78. Dan and Goodin, *Stolen RSA data used to hack defense contractor. Available at https://www.theregister.co.uk/2011/06/06/lockheed_martin_securid_hack/.* 2011.
79. Labs, R.F.R., *Anatomy of an attack. Available at http://blogs.rsa.com/anatomy-of-an-attack/.* 2011.
80. Barthe, G., et al. *Synthesis of fault attacks on cryptographic implementations*. in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 2014. ACM.
81. Boneh, D., R.A. DeMillo, and R.J. Lipton. *On the importance of checking cryptographic protocols for faults*. in *International Conference on the Theory and Applications of Cryptographic Techniques*. 1997. Springer.
82. Nunes, E., et al., *Cyber-deception and attribution in capture-the-flag exercises*, in *Cyber Deception*. 2016, Springer. p. 151-167.
83. Patel, A., N. Sharma, and M. Eirinaki. *Negative database for data security*. in *Computing, Engineering and Information, 2009. ICC'09. International Conference on*. 2009. IEEE.
84. Stolfo, S.J., M.B. Salem, and A.D. Keromytis. *Fog computing: Mitigating insider data theft attacks in the cloud*. in *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on*. 2012. IEEE.
85. Thaler, S., J. den Hartog, and M. Petkovic. *Towards Creating Believable Decoy Project Folders for Detecting Data Theft*. in *IFIP Annual Conference on Data and Applications Security and Privacy*. 2016. Springer.
86. Liu, J. *Preventing Leakages of Business Secrets from Encrypt Data Stored in the Cloud*. in *High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on*. 2015. IEEE.
87. Wilson, D. and J. Avery, *Mitigating Data Exfiltration in Storage-as-a-Service Clouds.* arXiv preprint arXiv:1606.08378, 2016.
88. Trostle, J. *Applying network address encryption to anonymity and preventing data exfiltration*. in *MILCOM 2008-2008 IEEE Military Communications Conference*. 2008. IEEE.
89. Tsuda, H., et al., *Inter-cloud data security for secure cloud-based business collaborations.* Fujitsu Sci Technol J, 2012. **48**(2): p. 169-176.
90. Zhuang, H., et al. *StoreSim: Optimizing Information Leakage in Multicloud Storage Services*. in *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*. 2015. IEEE.
91. Omran, O.M.B. and B. Panda. *A new technique to partition and manage data security in cloud databases*. in *Internet Technology and Secured Transactions (ICITST), 2014 9th International Conference for*. 2014. IEEE.
92. Jaeger, T., R. Sailer, and Y. Sreenivasan. *Managing the risk of covert information flows in virtual machine systems*. in *Proceedings of the 12th ACM symposium on Access control models and technologies*. 2007. ACM.
93. Gondi, K., et al. *Swipe: eager erasure of sensitive data in large scale systems software*. in *Proceedings of the second ACM conference on Data and Application Security and Privacy*. 2012. ACM.

94. Dan and Goodin, *"Dexter" malware steals credit card data from point-of-sale terminals. Available at https://arstechnica.com/security/2012/12/dexter-malware-steals-credit-card-data-from-point-of-sale-terminals/.* 2012.
95. Canim, M., et al., *Building disclosure risk aware query optimizers for relational databases.* Proceedings of the VLDB Endowment, 2010. **3**(1-2): p. 13-24.
96. Gao, H., et al. *Preventing Secret Data Leakage from Foreign Mappings in Virtual Machines*. in *International Conference on Security and Privacy in Communication Systems*. 2011. Springer.
97. Srinivasan, R., et al. *A multi-factor approach to securing software on client computing platforms*. in *Social Computing (SocialCom), 2010 IEEE Second International Conference on*. 2010. IEEE.
98. Zhuang, X., T. Zhang, and S. Pande. *HIDE: an infrastructure for efficiently protecting information leakage on the address bus*. in *ACM SIGPLAN Notices*. 2004. ACM.
99. Cohen, J.C. and S. Acharya. *Incorporating hardware trust mechanisms in Apache Hadoop: To improve the integrity and confidentiality of data in a distributed Apache Hadoop file system: An information technology infrastructure and software approach*. in *2012 IEEE Globecom Workshops*. 2012. IEEE.
100. Wen, Y., J. Zhao, and H. Chen. *Towards Thwarting Data Leakage with Memory Page Access Interception*. in *Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference on*. 2014. IEEE.
101. K. Perkins, C., *Data Loss Protection.* 2013.
102. Zilberman, P., et al., *Analyzing group E-mail exchange to detect data leakage.* Journal of the American Society for Information Science and Technology, 2013. **64**(9): p. 1780-1790.
103. Zilberman, P., et al. *Analyzing group communication for preventing data leakage via email*. in *Intelligence and Security Informatics (ISI), 2011 IEEE International Conference on*. 2011. IEEE.
104. Hamid, I.R.A. and J.H. Abawajy, *An approach for profiling phishing activities.* Computers & Security, 2014. **45**: p. 27-41.
105. Jaskolka, J., R. Khedri, and K.E. Sabri, *Investigative support for information confidentiality.* Journal of Ambient Intelligence and Humanized Computing, 2015. **6**(4): p. 425-451.
106. Raggo, M.a.H., Chet, *Chapter 11 - Mitigation Strategies. Available at http://www.sciencedirect.com/science/article/pii/B9781597497435000110.* 2013.
107. Cantrell, G. and D.D. Dampier. *Experiments in hiding data inside the file structure of common office documents: a stegonography application*. in *Proceedings of the 2004 international Symposium on information and Communication Technologies*. 2004. Trinity College Dublin.
108. Francis-Christie, C. and D. Lo. *A Combination of Active and Passive Video Steganalysis to Fight Sensitive Data Exfiltration through Online Video*. in *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*. 2016. IEEE.
109. Liu, Y., et al. *SIDD: A framework for detecting sensitive data exfiltration by an insider attack*. in *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*. 2009. IEEE.
110. Hussain, I. and N. Pandey. *Carrier data security using public key steganography in ZigBee*. in *Innovation and Challenges in Cyber Security (ICICCS-INBUSH), 2016 International Conference on*. 2016. IEEE.
111. Schiff, L. and S. Schmid, *PRI: Privacy Preserving Inspection of Encrypted Network Traffic.* arXiv preprint arXiv:1604.04465, 2016.
112. Al-Bataineh, A. and G. White. *Analysis and detection of malicious data exfiltration in web traffic*. in *Malicious and Unwanted Software (MALWARE), 2012 7th International Conference on*. 2012. IEEE.
113. Rohloff, K. *Privacy-Preserving Data Exfiltration Monitoring Using Homomorphic Encryption*. in *Cyber Security and Cloud Computing (CSCloud), 2015 IEEE 2nd International Conference on*. 2015. IEEE.
114. Koch, R. *Towards next-generation intrusion detection*. in *2011 3rd International Conference on Cyber Conflict*. 2011. IEEE.
115. He, G., et al. *A Novel Method to Detect Encrypted Data Exfiltration*. in *Advanced Cloud and Big Data (CBD), 2014 Second International Conference on*. 2014. IEEE.
116. Rennie, J.D., et al. *Tackling the poor assumptions of naive bayes text classifiers*. in *ICML*. 2003. Washington DC).
117. Jiang, J., et al. *Skip finite automaton: A content scanning engine to secure enterprise networks*. in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. 2010. IEEE.
118. Chae, C.-J., et al., *A privacy data leakage prevention method in P2P networks.* Peer-to-Peer Networking and Applications, 2016. **9**(3): p. 508-519.
119. Subhashini, P. and B.P. Rani. *Confidential Terms Detection Using Language Modeling Technique in Data Leakage Prevention*. in *Proceedings of the Second International Conference on Computer and Communication Technologies*. 2016. Springer.
120. Katz, G., Y. Elovici, and B. Shapira, *CoBAn: A context based model for data leakage prevention.* Information Sciences, 2014. **262**: p. 137-158.
121. Peneti, S. and B.P. Rani. *Data leakage prevention system with time stamp*. in *Information Communication and Embedded Systems (ICICES), 2016 International Conference on*. 2016. IEEE.
122. Priebe, C., et al. *Cloudsafetynet: detecting data leakage between cloud tenants*. in *Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security*. 2014. ACM.
123. Ampah, N.K., et al., *An intrusion detection technique based on continuous binary communication channels.* International Journal of Security and Networks, 2011. **6**(2-3): p. 174-180.

124. Bhuyan, M.H., D.K. Bhattacharyya, and J.K. Kalita, *Network anomaly detection: methods, systems and tools.* IEEE Communications Surveys & Tutorials, 2014. **16**(1): p. 303-336.
125. Xiang, J., et al. *Using extreme learning machine for intrusion detection in a big data environment*. in *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*. 2014. ACM.
126. Huang, G.-B., et al., *Extreme learning machine for regression and multiclass classification.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2012. **42**(2): p. 513-529.
127. Huang, G.-B., Q.-Y. Zhu, and C.-K. Siew, *Extreme learning machine: theory and applications.* Neurocomputing, 2006. **70**(1): p. 489-501.
128. Koch, R. and G.D. Rodosek. *User identification in encrypted network communications*. in *2010 International Conference on Network and Service Management*. 2010. IEEE.
129. Marchetti, M., et al., *Analysis of high volumes of network traffic for Advanced Persistent Threat detection.* Computer Networks, 2016.
130. Wang, W., B. Yang, and V.Y. Chen. *A visual analytics approach to detecting server redirections and data exfiltration*. in *Intelligence and Security Informatics (ISI), 2015 IEEE International Conference on*. 2015. IEEE.
131. Riecker, M., et al., *Lightweight energy consumption-based intrusion detection system for wireless sensor networks.* International Journal of Information Security, 2015. **14**(2): p. 155-167.
132. Pu, Y., et al., *Data stolen trojan detection based on network behaviors.* Procedia Computer Science, 2013. **17**: p. 828-835.
133. Jagasia, M. and D. Huang. *Distributed data-theft detection in wireless sensor networks*. in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. 2009. IEEE.
134. Portnoy, L., E. Eskin, and S. Stolfo. *Intrusion detection with unlabeled data using clustering*. in *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001*. 2001. Citeseer.
135. Rastegari, S., C.-P. Lam, and P. Hingston. *A Statistical Rule Learning Approach to Network Intrusion Detection*. in *IT Convergence and Security (ICITCS), 2015 5th International Conference on*. 2015. IEEE.
136. Rastegari, S., P. Hingston, and C.-P. Lam, *Evolving statistical rulesets for network intrusion detection.* Applied Soft Computing, 2015. **33**: p. 348-359.
137. Benham, A., H. Read, and I. Sutherland. *Network Attack Analysis and the Behaviour Engine*. in *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*. 2013. IEEE.
138. Koch, R., M. Golling, and G.D. Rodosek, *Behavior-based intrusion detection in encrypted environments.* IEEE Communications Magazine, 2014. **52**(7): p. 124-131.
139. Wang, J. and I.C. Paschalidis. *Robust anomaly detection in dynamic networks*. in *Control and Automation (MED), 2014 22nd Mediterranean Conference of*. 2014. IEEE.
140. Sigholm, J. and M. Raciti. *Best-effort Data Leakage Prevention in inter-organizational tactical MANETs*. in *MILCOM 2012-2012 IEEE Military Communications Conference*. 2012. IEEE.
141. Ng, J., D. Joshi, and S.M. Banik. *Applying Data Mining Techniques to Intrusion Detection*. in *Information Technology-New Generations (ITNG), 2015 12th International Conference on*. 2015. IEEE.
142. Owezarski, P. *A Near Real-Time Algorithm for Autonomous Identification and Characterization of Honeypot Attacks*. in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*. 2015. ACM.
143. Mazel, J., et al. *Sub-space clustering, inter-clustering results association & anomaly correlation for unsupervised network anomaly detection*. in *Proceedings of the 7th International Conference on Network and Services Management*. 2011. International Federation for Information Processing.
144. Berlin, K., D. Slater, and J. Saxe. *Malicious behavior detection using windows audit logs*. in *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*. 2015. ACM.
145. Zhang, J., et al. *Safeguarding academic accounts and resources with the university credential abuse auditing system*. in *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*. 2012. IEEE.
146. Jewell, B. and J. Beaver. *Host-based data exfiltration detection via system call sequences*. in *ICIW2011-Proceedings of the 6th International Conference on Information Warfare and Secuirty: ICIW*. 2011. Academic Conferences Limited.
147. Wüchner, T., M. Ochoa, and A. Pretschner. *Malware detection with quantitative data flow graphs*. in *Proceedings of the 9th ACM symposium on Information, computer and communications security*. 2014. ACM.
148. *Symantec. Malware database,*. Nov 2013.
149. Bayer., U., *Large-Scale Dynamic Malware Analysis.* PhD thesis, Technische Universit¨at Wien, 2009.
150. Kim, S., et al., *Application of density-based outlier detection to database activity monitoring.* Information Systems Frontiers, 2013. **15**(1): p. 55-65.
151. Udd, R., et al. *Exploiting Bro for Intrusion Detection in a SCADA System*. in *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*. 2016. ACM.
152. Paxson, V., *Bro: a system for detecting network intruders in real-time.* Computer networks, 1999. **31**(23): p. 2435-2463.
153. Yang, Y., M. Manoharan, and K.S. Barber. *Modelling and Analysis of Identity Threat Behaviors through Text Mining of Identity Theft Stories*. in *Intelligence and Security Informatics Conference (JISIC), 2014 IEEE Joint*. 2014. IEEE.

154. Shan, L., D. Xiaorui, and R. Hong. *An adaptive method preventing database from SQL injection attacks*. in *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*. 2010. IEEE.
155. Flood, J. and A. Keane. *A Proposed Framework For The Active Detection Of Security Vulnerabilities In Multi-Tenancy Cloud Systems*. in *Emerging Intelligent Data and Web Technologies (EIDWT), 2012 Third International Conference on*. 2012. IEEE.
156. Ramachandran, R., S. Neelakantan, and A.S. Bidyarthy. *Behavior model for detecting data exfiltration in network environment*. in *Internet Multimedia Systems Architecture and Application (IMSAA), 2011 IEEE 5th International Conference on*. 2011. IEEE.
157. Myers, J., M. Grimaila, and R. Mills. *Insider threat detection using distributed event correlation of web server logs*. in *International Conference on Information Warfare and Security*. 2010. Academic Conferences International Limited.
158. Ahn, S.-H., N.-U. Kim, and T.-M. Chung. *Big data analysis system concept for detecting unknown attacks*. in *16th International Conference on Advanced Communication Technology*. 2014. IEEE.
159. Rajamenakshi, R. and G. Padmavathi. *An Integrated Network Behavior and Policy Based Data Exfiltration Detection Framework*. in *Proceedings of the Fifth International Conference on Fuzzy and Neuro Computing (FANCCO-2015)*. 2015. Springer.
160. Carvalho, V.S., M. João, and J.P. Magalhães. *OwlSight: Platform for Real-Time Detection and Visualization of Cyber Threats*. in *Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2016 IEEE 2nd International Conference on*. 2016. IEEE.
161. Gao, Y., et al. *Haddle: A Framework for Investigating Data Leakage Attacks in Hadoop*. in *2015 IEEE Global Communications Conference (GLOBECOM)*. 2015. IEEE.
162. Agrawal, R., P.J. Haas, and J. Kiernan, *Watermarking relational data: framework, algorithms and analysis.* The VLDB journal, 2003. **12**(2): p. 157-169.
163. Melkundi, S. and C. Chandankhede. *A robust technique for relational database watermarking and verification*. in *Communication, Information & Computing Technology (ICCICT), 2015 International Conference on*. 2015. IEEE.
164. Consortium, U., *The Unicode Standard Version 6.2–Core Specification.* Unicode Inc., Mountain View, California, USA, 2012.
165. Schick, R. and C. Ruland. *Data Leakage Tracking–Non-Repudiation of Forwarding*. in *International Conference on Informatics Engineering and Information Science*. 2011. Springer.
166. Grier, J., *Detecting data theft using stochastic forensics.* digital investigation, 2011. **8**: p. S71-S77.
167. Patel, P.C. and U. Singh. *Detection of data theft using fuzzy inference system*. in *Advance Computing Conference (IACC), 2013 IEEE 3rd International*. 2013. IEEE.
168. Kavallaris, T. and V. Katos, *On the detection of pod slurping attacks.* computers & security, 2010. **29**(6): p. 680-685.
169. Papadimitriou, P. and H. Garcia-Molina. *A model for data leakage detection*. in *2009 IEEE 25th International Conference on Data Engineering*. 2009. IEEE.
170. Wang, X., J. Shi, and L. Guo. *Towards analyzing traceability of data leakage by malicious insiders*. in *International Conference on Trustworthy Computing and Services*. 2012. Springer.
171. Kumar, A., et al. *Comparative evaluation of algorithms for effective data leakage detection*. in *Information & Communication Technologies (ICT), 2013 IEEE Conference on*. 2013. IEEE.
172. Krishnan, S., K.Z. Snow, and F. Monrose, *Trail of Bytes: New Techniques for Supporting Data Provenance and Limiting Privacy Breaches.* IEEE Transactions on Information Forensics and Security, 2012. **7**(6): p. 1876-1889.
173. Yu, J., et al. *LeakProber: a framework for profiling sensitive data leakage paths*. in *Proceedings of the first ACM conference on Data and application security and privacy*. 2011. ACM.
174. Xu, F., et al. *A bayesian belief network for data leakage investigation*. in *Proceedings of the 2nd international workshop on Security and forensics in communication systems*. 2014. ACM.
175. Ikegami, Y. and T. Yamauchi. *Attacker Investigation System Triggered by Information Leakage*. in *Advanced Applied Informatics (IIAI-AAI), 2015 IIAI 4th International Congress on*. 2015. IEEE.
176. Alexander, J.S., T. Dean, and S. Knight. *Spy vs. spy: Counter-intelligence methods for backtracking malicious intrusions*. in *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*. 2011. IBM Corp.
177. Deng, H., et al. *Tracing and revoking leaked credentials: accountability in leaking sensitive outsourced data*. in *Proceedings of the 9th ACM symposium on Information, computer and communications security*. 2014. ACM.
178. Delerablée, C. *Identity-based broadcast encryption with constant size ciphertexts and private keys*. in *International Conference on the Theory and Application of Cryptology and Information Security*. 2007. Springer.
179. Nuida, K., et al., *An improvement of discrete Tardos fingerprinting codes.* Designs, Codes and Cryptography, 2009. **52**(3): p. 339-362.
180. McAfee, C.f.S.a.I.S.-. *Net Losses: Estimating the Global Cost of Cybercrime Economic impact of cybercrime II.* White Paper, 2014.

181. Inc., V., *Data Breach Investigation Report. Available at http://www.verizonenterprise.com/resources/reports/rp_2010-data-breach-report_en_xg.pdf. Accessed on 11th July, 2017.* 2010.
182. Alliance, C.S., *Big Data Analytics for Security Intelligence. Available at https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Big_Data_Analytics_for_Security_Intelligence.pdf. Accessed on 11th July, 2017.* 2013.

Faheem Ullah

Faheem Ullah is currently pursuing the Ph.D. program in software engineering with the School of Computer Science, The University of Adelaide, Australia. He has been working with research centre, Centre for Advanced Research in Engineering (CARE), on various research projects related to computer science. His current research primarily focuses on software architecture, cyber security, and big data analytics.

Matthew Edwards

Matthew Edwards is a PhD Student and Senior Research Associate at the Security Lancaster Research Centre. His research interests lie at the intersection of human-centred security issues, open source intelligence and machine learning. His doctoral thesis focuses on methods of attribute valuation for identity resolution tasks, and he has previously published work regarding schema-building and sampling concerns for online identity resolution. He is currently investigating methods for detecting and preventing mass-market fraud campaigns.

Rajiv Ramdhany

Rajiv Ramdhany is a senior research associate at BBC Research and Development. Previously, he was a senior researcher with the Next Generation Distributed Systems Group at Lancaster University, where he worked on Adaptive Systems Software for Ad Hoc Networks. He has worked on improving the adaptability of operating systems features such as protocol stacks and concurrency support, particular in embedded systems and other resource-constrained environments. He also has a background in middleware protocols. His research interests include the Internet of Things for smarter cities, multi-user exploitation of commodity Wireless Sensor Networks and communication protocols for ad hoc networks. Rajiv received his Ph.D. in 2011 for his work on ‘‘Supporting the dynamic deployment of ad hoc routing protocols’’ and an M.Sc. in Distributed Systems in 2003 at Lancaster University. In between, Rajiv has worked as a lecturer in Computer Science at the University of Mauritius.

Ruzanna Chitchyan

Ruzanna Chitchyan is a lecturer in Software Engineering the Department of Computer Science, University of Leicester. Her current research interests are in requirements modelling and analysis in general and aspect-oriented requirements engineering and architecture design in particular. She has worked on several major EC projects on this topic (e.g. AOSD-Europe, AMPLE, and DiVA), and, throughout the years, has actively participated in the Early Aspects workshops.

Muhammad Ali Babar

Muhammad Ali Babar is currently a Professor with the School of Computer Science, The University of Adelaide, Australia. He is also an Honorary Visiting Professor with the Software Institute University of Nanjing, China. He has authored/co-authored over 180 peer-reviewed research papers at premier

software engineering journals and conferences, such as the ACM Transactions on Software Engineering and Methods, the IEEE Software, and ICSE.

Awais Rashid

Professor Awais Rashid is Director of Security Lancaster Research Centre, one of the UK's Academic Centres of Excellence in Cyber Security Research. His research interests focus on security of cyber physical systems (CPS) and studies of adversarial and non-adversarial behaviours pertaining to cyber security. He leads a number of research projects, including a project as part of the Research Institute in Trustworthy Industrial Control Systems (RITICS). He also co-leads the Security and Safety theme within the UK hub (PETRAS) on Security, Privacy and Trust in the Internet of Things and a CHIST-ERA project on human and technical responses to cyber-physical infrastructures under attack.