A Cost-Effective Cloud Event Archival for SIEMs

Adriano Serckumecka[1] Ibéria Medeiros[1] Bernardo Ferreira[2] Alysson Bessani[1]
[1]LASIGE, Faculty of Science, University of Lisboa, Portugal
[2]DI, FCT, Universidade NOVA de Lisboa & NOVA LINCS, Portugal

***Abstract*—Security Information and Event Management (SIEM) systems have been adopted by organizations to enable a holistic monitoring of malicious activities in their IT infrastructures. SIEMs receive events from diverse devices of the organization's IT infrastructure (e.g., servers, firewalls, IDS), correlate these events, and present reports for security analysts. Given the large number of events collected by SIEMs, it is costly to store such data for long periods. Since organizations store a relatively limited time-frame of events, the forensic analysis capabilities severely become reduced. This concern limits the organizations' ability to store important information about the past cybersecurity-related activity, limiting forensic analysis. A possible solution for this issue is to leverage public cloud storage services, exploiting their low cost and "infinite" scalability. We present SLiCER an archival system for long-term storage that makes use of a multi-cloud-based storage system to guarantee data security and ensures cost-effectiveness by grouping events in blocks and using indexing techniques to recover them. The system was evaluated using a real dataset and the results show that it is significantly more cost-efficient than competing alternatives.**

***Index Terms*—Security, long-term archival, SIEM, indexing, multiple clouds, cost-effectiveness**

I. Introduction

Security Information and Event Management (SIEM) systems provide a centralized point of analysis for correlating security-related events generated by a multitude of devices (firewall, IDS, IPS, anti-virus, syslog, etc.). One of the most important benefits of having such systems is the possibility to access these events in retrospective to understand and investigate why and how security incidents happened.

Modern SIEMs archive collected events for a short period of time, varying from a week to one year [1], [2]. This happens due to the amount of storage required since large organizations collect thousands of events per second. Accommodating this large data set is not only a matter of buying more storage space (i.e., disks), but also the management effort for dealing with such data (which is critical and privacy-sensitive). Additionally, these systems need to support efficient queries on large-scale storage, which is not the case in many SIEMs backed by traditional databases. The consequence is a limited support for forensic activities, especially in situations when incidents could only be explained by already deleted events. For instance, certain advanced security threats exploring zero-day vulnerabilities take on average 320 days until being discovered [3]. Furthermore, many countries have data protection laws that require the exact date when a breach of sensitive data was discovered to notify the affected parties promptly. As such, some standardization bodies have recommending to store events for one or more years. For instance, the COBIT (Control Objectives for Information and related Technology) and NIST (National Institute of Standards and Technology, SP800-53 and SP800-92) have recommended storing event log data for one year [4] and 5 years [5], respectively.

An alternative for increasing the SIEM event storage capacity is the use of public cloud storage services, such as Amazon S3 and Azure Blob Storage. The costs seem promising: a GB/month costs USD 0.01 in S3 [6], which means that a 10TB event database costs USD 1200/year with all the backup and security management being performed by the cloud provider. However, many SIEM users still have concerns about moving this sensitive data to a third-party provider that needs to be completely trusted.

Another concern is how to make this encrypted data searchable, while recovering events in a cheap and fast way when needed. Searchable Symmetric Encryption (SSE) (e.g., [7]–[9]) is the most well-known technique to search over encrypted data stored in the cloud. However, this technique is not the most appropriate for a large-scale archival data store that will rarely be queried, besides being susceptible to leakage attacks.

In this paper we propose a simpler and more pragmatical approach for implementing such archival, in which data is organized and indexed before going to the cloud, in such a way that queries can be processed at the client side. For ensuring data security, we employ a cloud-of-clouds storage system where data is encrypted and efficiently replicated through several clouds [10]. For ensuring efficient and secure event retrieval, we organize events in blocks according to time periods and devices that generated them, storing them with indexes that help recovering events in a fast and cheap way.

This approach, named SLiCER, is being implemented in a system which runs together with the SIEM on the organization premises, collecting events to be stored on the cloud, and supporting queries on such archival. Besides security (ensured through encryption and the use of cloud-of-clouds storage), the most important requirement of our proposal is its cost-effectiveness. As mentioned before, cloud storage space is cheap, but reading data and performing requests from the cloud can be very expensive. Therefore, to minimize the costs of using cloud storage services, we propose a domain-specific data and query model that enables high rates of compression and associates small indexes to event blocks, decreasing the amount of data that needs to be stored and retrieved. We evaluate the system experimentally using 14.4GB of real data and compare different indexed and no indexed approaches.

In summary, the contributions of the paper are:

1) a cloud-backed archival data model that enables a cost-efficient storage and retrieval of events.
2) a system that implements this model, organizes events in data blocks, generates their indexes, and uses these indexes to recover data stored in the clouds;
3) an experimental evaluation using real SIEM events comparing different indexed and no indexed algorithms.

II. Related Work

To the best of our knowledge, no previous work tries to solve the same problem we are addressing here: efficient and secure archival of security events using public clouds. However, there are several initiatives broadly related to ours that deserve mentioning, namely: SIEMs in the cloud, efficient forensics in archival data, and cloud-of-clouds archival.

a) SIEMs and the cloud: There are many analytic and archival systems currently available that can be used as a SIEM: Splunk [11], Solr [12], Elastic [13], to cite just a few. These systems aim to provide an analytics platform, offering features to generate alerts and reports, becoming closer to a complete SIEM. Elastic, for instance, employs Apache Lucene [14] as its background indexing engine, which creates inverted indexes for the stored documents (or events), used to perform keyword searches. Splunk is very similar, aggregating a few different functionalities such as a correlation engine and the use of a proprietary scheme to index the data.

Many of these systems are also offered as a cloud-managed deployment. The providers maintain an online infrastructure in the cloud and charge the user by the volume of events ingested by the system. These systems could be used as an archival system but would be an expensive tool as they offer a richer set of features. SLiCER does not aim to offer the full capabilities of a SIEM or data analytics engine, instead, it provides only long-term event archival and retrieval in the cloud, with security and reducing the operational costs.

b) Efficient forensics in archival data: VAST [15] is a forensic analytics platform to capture and retain network activities, storing and providing queries over the network historical data. It has some similarities to Elasticsearch [16] as a distributed platform with indexing and searching capabilities. However, it uses a different approach to index data, employing bitmap indexes. Although the indexes employed in VAST are quite efficient, they significantly increase the amount of storage: even with compression, VAST indexes require 37% of the size of the raw stored data. As we show in Section VI, SLiCER employs much smaller indexes than VAST.

c) Cloud-of-Clouds storage: There is a plethora of works that advocate the use of multiple cloud storage services to provide availability and fault tolerance, and improving reliability. Of particular interest are data-centric solutions such as RACS [17], DepSky [18], and their follow up works (e.g., [10], [19], [20]), which do not require any server in the cloud. These systems employ techniques such as erasure codes, secret sharing and quorum replication to disperse encrypted data on several cloud providers, hence ensuring stored data survives loss, corruption, leakage, and vendor lock-in, as long as a pre-defined fraction of the providers are still correct.

SLiCER complements these systems by providing a client-side data processing engine that allows content-based querying on stored data.

III. The SLiCER Approach and System

SLiCER aims to overcome and address the storage constraints of modern SIEMs, providing a way to retain security events for long periods by leveraging low cost cloud storage services. SLiCER can store events[1] generated from different sources. The core of the system is responsible for organizing the received events for efficient storage and creating indexes for retrieval in the clouds, both in terms of performance and monetary costs. Secure storage is fulfilled by using a cloud-of-clouds storage system with encryption [10], and the information retrieval is done by searching over the indexes.

In the following sections, we present the SLiCER architecture, and the indexing methods supported by it.

A. Architecture

SLiCER is composed of three main components, namely, *Event Archiver*, *Query Manager*, and *Cache*. Figure 1 shows the general architecture of the system and its interaction with the SIEM infrastructure and public clouds.

a) Event archiver: This component implements a data model to organize and prepare events before they are stored in the cloud. Briefly, in this model, events are grouped by device and time range, with an unique index being created for each event block. The goal is to reduce the number of requests and unnecessary downloads from the cloud, since these are expensive when compared to storage costs. In more detail, the Event Archiver performs the following sequence of tasks:

(1) *Receives and organizes events.* SLiCER monitors the arrival of new events coming from the SIEM infrastructure (devices, SIEM, or Logger[2]). Events are separated by the devices that generated them.

(2) *Creates event blocks.* For each device, SLiCER generates a block after a certain time (e.g., one hour) or when the amount of data received reaches a certain size (e.g., 10MB).

(3) *Creates indexes of event blocks.* An index is created for each block of events by using a specific index method (see Section III-B).

(4) *Caches the indexes.* Each index generated in the previous step is copied to the *Cache* in order to ensure that queries can be run locally, decreasing the costs of reading data from the cloud. Also, the block of events can be sent to the cache if the administrator wants to keep the most recent events locally.

(5) *Sends blocks and indexes to the cloud.* Each block and its respective index are compressed, encrypted, and sent to the cloud.

[1] Which are basically small *documents*, following the information retrieval terminology.

[2] A SIEM archival that stores events for short periods (e.g., six months).

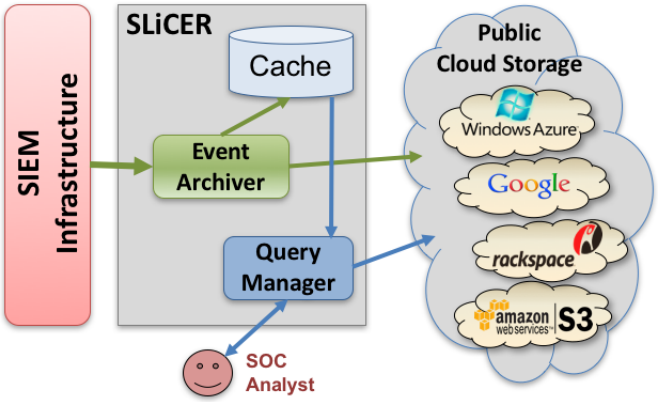

Fig. 1. The SLiCER system architecture.

b) Query Manager: SIEM operators access the archived events stored in the clouds through this component. It implements a query model to allow the execution of queries. The query model is defined by three parameters: a subset of devices (D) which contains the events the analyst wants to search, an interval (I) which defines a start and end time to a query, and a subset of keywords (K) which the analyst wants to search over.

The Query Manager performs the following steps: (1) *Receives queries and gets the indexes of event blocks.* The query manager receives and checks the queries syntax, and looks for the indexes of blocks, from the cache, for the devices specified in the query. The indexes can also be obtained from the cloud, in case they are not in the cache. (2) *Executes queries and retrieves event blocks.* The queries are executed over the index of each block within the parameters of search. The result is the event block names that are going to be read from the cloud. (3) *Extracts events.* The blocks are decrypted, decompressed, and the events matching the query criteria are extracted and cached. (4) *Deliver events.* The resulting set of events is delivered to the SIEM analyst.

c) Cache: This is the local storage where recently-fetched indexes and data blocks downloaded from the clouds are stored. Using a cache allows users to quickly execute new queries by downloading only the missing files, thus reducing the operation and data transfer costs of the system. This feature is important in practice for two reasons. First, indexes can be quite small, which means that a large number of them can be kept in the cache to save accesses to the cloud. Second, it is common in forensics to start with more general queries that are progressively refined, so it is important that such followup queries can be processed locally.

B. Indexing Methods

In SLiCER, queries can be performed by specifying a device, a time range, and/or keywords that may have appeared in a security event (e.g. an IP address). A benefit from organizing events in blocks according to device and time ranges, is that the first two types of queries are very easy to process, since SLiCER can directly return the requested blocks. However, keyword queries require an index to be efficient.

Indexes are auxiliary data structures used to efficiently locate and retrieve data without having to scan the whole block. In our case, the data comprises text events coming from different devices in different formats, i.e., with various fields and sizes. This section presents the different indexing methods available in SLiCER, including the no-index method in which data is not indexed at all and two different methods of indexing such event content, namely inverted index and probabilistic Bloom filter.

a) No-Index: In this method, events are still organized in blocks by device and time range, but blocks are not indexed. This means that device and time queries will be more efficient, since blocks can be directly retrieved and there are no additional indexes to download, nonetheless keyword queries require blocks to be linearly scanned.

b) Inverted index: It organizes documents (in our case events) to a list of terms, mapping its locations and frequency in a file. The index files contain a list of keyword references and is created based on all distinct keywords existent in a data block. As blocks are organized the same way as in No-Index, our data model permits to create a separated file index for each event block, which is used to perform searches without needing to download the data previously.

c) Bloom filter index: Uses Bloom filters [21] to store keywords appearing in each field of the events in a block. Being probabilistic, it can answer if a keyword may be in a set or definitely is not in a set, meaning that it can generate false positives, but never false negatives. Similarly to the inverted index, this type of index can be used to index all distinct keywords within a block.

IV. Cost Model

The costs of recovering data are calculated taking into account the kind of data (event blocks and indexes), its size and number, and the price of requests, i.e., the GET request for recovering data. However, the cost of a query is not associated with the number of events found by a query, but the number and size of blocks scanned.

The size and time range of the data blocks can be adjusted by the analyst in advance and can influence the cost of a query. Bigger blocks can force the download of unnecessary data, turning the query expensive. On the other hand, smaller blocks can generate many GET requests to retrieve the necessary blocks, which can also result in an expensive query. Therefore, the challenge is to find the best cost-benefit trade-off between the number of blocks and its size.

As the system we propose can be configured with or without an index, the final query cost is affected by this configuration, meaning that for the former the cost must include the number and size of the indexes and the GET requests to get them, while for the latter this cost does not exist.

Equation 1 represents the simplified query cost model. It is composed of the costs of requesting and downloading the

indexes, plus the costs of requesting and downloading data blocks from the cloud.

$$Q_{cost} = (P_{request} * N_{indexes} + P_{download} * S_{indexes}) + (P_{request} * N_{blocks} + P_{download} * S_{blocks}) \quad (1)$$

V. Implementation

We implemented SLiCER as a userspace daemon in Java. The system has two main components: event achiever and query manager, as described in Section III (the cache is just a local directory).

The SLiCER event archiver uses a no-index and two indexing algorithms: inverted indexes and bloom filters. Inverted indexes are built through Apache Lucene v7.7.0 [14], while Bloom filters use a long fast implementation[3] [21]. Also, we resort to the XZ compression algorithm[4] to compress event blocks and their indexes.

The process of sending files to the clouds is done using the Charon cloud-of-clouds storage system [10], which implements an improved version of DepSky [18] under a file system interface.

VI. Evaluation

The objective of the experimental evaluation was to answer the following questions:

1) How much SLiCER reduces the storage space needed for the events?
2) Is SLiCER able to reduce the time to search data in the cloud, compared with other solutions?
3) Is SLiCER able to reduce the cost of recovering data from the cloud, compared with other solutions?

We performed two types of tests: experimental evaluation of our prototype and cost simulations based on the prices of Amazon S3 [6]. For the former, we deployed SLiCER on a Dell desktop 7040 with a Intel core i7 CPU 3,4-3,8 GHz, 16 GB RAM, and a HDD 7200rpm 1TB.

In Section VI-A we present the methods and algorithms we used to process the dataset and evaluate our system. In Section VI-B we present the characterization and processing of the dataset, and in Section VI-C we present the evaluation of SLiCER in execution of queries and cost simulation.

A. Methods and Algorithms

We evaluated several variants of SLiCER and compare it with a simplistic "cloud archival" approach, which we called *Strawman*.

In the *Strawman* (SM) design, each event block is created for a time range, meaning that it contains all events generated by all devices in a given period. No indexes are created for the blocks. This design reflects a simplistic approach in which collected events are simply sent to the cloud from time to time in blocks, and such blocks are downloaded back in case they need to be queried. To retrieve data from the cloud, the final query cost is defined only by all blocks downloaded for the time-range specified in the query, taking into account their sizes and the GET requests.

No-index (NI) is one of the variants of SLiCER in which blocks generated following SLiCER's data model are created without indexes.

Index represents the main configuration of SLiCER, in which event blocks are indexed. We defined two variants of this design: Lucene and Bloom filters indexes.

[3]https://github.com/vvcogo/java-longfastbloomfilter

[4]https://git.tukaani.org/

TABLE I
Index sizes.

Algorithms	SM	NI	BF	LC
Index Size (MB)	-	-	41	601
Index size % (raw)	-	-	0,29%	4,17%

B. Dataset Characterization and Processing

a) Dataset characterization: The dataset was obtained from a test SIEM environment of a large power utility company, which mimics a subset of the utility production system, and is used to train staff. The events were collected in the CEF (Common Event Format) format, from 11 devices (firewall, antivirus, DB server, etc.), with 2 of them being collectors that receive events from other devices not directly connected to the SIEM. The number of events collected each day is similar. During a period of five days, 8861999 events were gathered, each one with an average size of 1.7 kB, leading to a total of 14.4 GB of data.

This real dataset has a limited time range, however, it represents a typical time window used in most queries (between 1 to 5 days) executed by a Security Operation Center (SOC), as the owner of this dataset.

b) Dataset processing: We processed the dataset using Strawman (SM) and the three variants of SLiCER: NoIndex (NI), Lucene (LC), and Bloom Filter (BF), as discussed before.

After SLiCER and SM finished organizing the 14,4 GB of events into blocks and compressing them, the data was reduced to 220 MB, about 70x of compression using XZ algorithm. The indexes were also compressed to optimize the storage.

Table I show the index sizes in MB and the percentage compared to the raw data size. Lucene index size widely contrasts with Bloom Filter since Lucene indexes all terms of a document, being deterministic. However, after compression, the Lucene index is about 6 times smaller than its uncompressed version. This way, LC can be cheaper than SM approach when the returned data blocks are less than 35% of all blocks in that time range. Nevertheless, the NI approach is cheaper than LC since it does not have to download indexes, remaining cheaper when the returned data blocks are more than 7%, approximately. Finally, Bloom Filter presents the smallest index due its probabilistic algorithm. BF maintain its advantage even when lots of blocks are returned by a query.

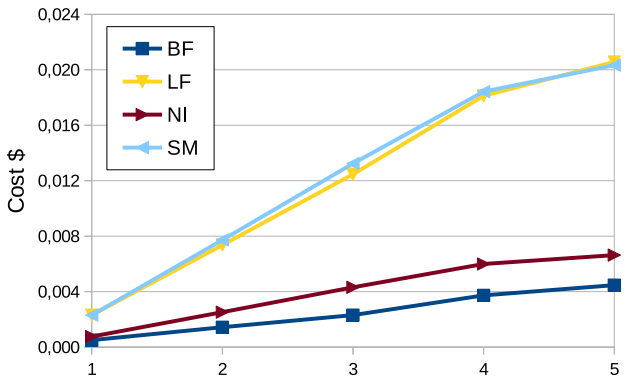

Fig. 2. Query cost considering time ranges from 1 to 5 days of events.

NI will be cheaper than BF only when the sum of indexes and blocks sizes were greater than the size of all data blocks involved in that query.

This analysis allows us to answer to question 1, where we can see that applying the SLiCER data model can reduce next to 70 times the size of the data stored in the cloud, including the indexes.

C. Query Cost

The cost of retrieving data from the clouds is intrinsically related to how specific and refined a query is, independently of the dataset size. The differences between this dataset and a bigger one are the available time range for queries and the monthly storage cost. Therefore, the cost is dependent of the number and size of data blocks and their indexes that will be downloaded. The number of blocks will determine the GET operations cost to request the data from the cloud and the size will define the cost to download data from there.

a) Varying time range: In order to evaluate SLiCER's query cost, we defined a query to represent a typical scenario, when a query can filter and reduce the necessary download to less than half of the query data range. The following query is formed by three default parts: the devices, parameters, and time range.

In [10.10.25.2]
where [dst=10.10.2.12, shost=collect2]
from [1 to 5 days]

Therefore, [10.10.25.2] is the device, [dst=10.10.2.12, shost=collect2] are two different keywords, and [1 to 5 days] the time range. The query was executed 5 times: the first is relative to the dataset collected on the first day, the second execution comprises the first and second days, and so on.

The data generated by this device represents 33% of the whole dataset, distributed in 92 data blocks, and the same number of indexes, when applicable. For example, in the query using 5 days of data and LC as algorithm, the blocks that match were 43, representing 46% of the total data blocks.

Figure 2 shows the monetary cost of executing the query (in USD), based on Equation 1, looking for events in different sizes of the dataset (i.e., time range between 1 to 5 days). When calculating these costs, we considered that no data was cached (event blocks or indexes).

For all executed queries, the SM and LC were the most expensive because the former downloads all data blocks for all time ranges and the latter downloads the indexes which have big sizes.

To answer questions 2 and 3, we conclude that both NI and BF have better results in terms of execution time and cost than the SM and LC. However, since the size of indexes is a disadvantage of Lucene, it is preferable to opt for a method that is slightly less efficient but considerably cheaper. In this way, BF is the best solution that fits in this assumption.

b) False positives: As we mentioned before, Bloom filters can generate false positives due to its probabilistic approach. The number of blocks unnecessarily downloaded was few, about 1% of the total, not impacting the final costs. However, in some cases this number can be higher due to the form that blocks are indexed.

VII. Conclusion

The paper presents the SLICER approach and system to improve the capacity of SIEMs for long-term events retention and their memory for future forensic analysis. The approach, on one hand, leverages public cloud storage, exploiting their low costs and "infinite" scalability. On the other hand, it employs a data model that groups events in blocks using different indexing techniques to efficiently recover data, ensuring cost-effectiveness in storing and recovering data to/from the clouds. We evaluated three variants of SLICER with a real dataset provided by a power utility company. The results showed that the system is significantly more cost-efficient than competing alternatives. Finally, although we focused on improving the storage capabilities for security events for SIEMs, SLiCER can also be adapted to work as a long-term storage for other purposes or systems that generate different types of events.

Acknowledgment

This work is supported by the EC through the DiSIEM project (H2020-700692), and FCT through the IRCoC project (PTDC/EEISCR/6970/2014) and the LASIGE Research Unit (UID/CEC/00408/2019). Bernardo Ferreira was supported by FCT/MCTES, through strategic project NOVA LINCS (UID/CEC/04516/2013) and project HADES (PTDC/CCI-INF/31698/2017).

References

[1] I. Alienvault. (2016, February) Best practices for configuring your usm installation. [Online]. Available: www.alienvault.com/forums/discussion/6705/q-a-from-webcast-best-practices-for-configuring-your-usm-installationl

[2] B. Walther. (2013, November) Arcsight data retention. [Online]. Available: https://wikis.uit.tufts.edu/confluence/display/exchange2010/ArcSightDataRetention

[3] L. Bilge and T. Dumitras, "Before we knew it: an empirical study of zero-day attacks in the real world," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 833–844.

[4] S. Gordon. (2010, October) Siem best practices to work. [Online]. Available: www.eslared.org.ve/walc2012/material/track4/Monitoreo/Top_10_SIEM_Best_Practices.pdf

[5] K. Kent and M. Souppaya. (2016, February) Nist-guide to computer security log management. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-92.pdf

[6] I. Amazon Web Services. (2017, July) Amazon s3 pricing. [Online]. Available: https://aws.amazon.com/s3/pricing/

[7] C. Orencik, A. Selcuk, E. Savas, and M. Kantarcioglu, "Multi-keyword search over encrypted data with scoring and search pattern obfuscation," *International Journal of Information Security*, pp. 251–269, 2016.

[8] M. Naveed, M. Prabhakaran, and C. A. Gunter, "Dynamic searchable encryption via blind storage," in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 639–654.

[9] B. Ferreira, J. Leitao, and H. Domingos, "Muse: Multimodal searchable encryption for cloud applications," in *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, pp. 181–190.

[10] R. Mendes, T. Oliveira, V. Cogo, N. Neves, and A. Bessani, "Charon: A secure cloud-of-clouds system for storing and sharing big data." *IEEE Transactions on Cloud Computing*, Accepted on IEEE Transactions on Cloud Computing (2019).

[11] I. Splunk. (2018, January) Splunk siem. [Online]. Available: https://www.splunk.com/en_us/products/quick-start-bundles/siem.html

[12] A. S. Foundation. (2019, March) Apache solr. [Online]. Available: http://lucene.apache.org/solr/

[13] I. Elastic. (2019, March) Elastic platform. [Online]. Available: https://elastic.co/

[14] A. S. Foundation. (2017, July) Apache lucene 6.6.0 documentation. [Online]. Available: http://lucene.apache.org/core/6_6_0/

[15] M. Vallentin, V. Paxson, and R. Sommer, "Vast: A unified platform for interactive network forensics," in *13th USENIX Symposium on Networked Systems Design and Implementation, 2016)*, pp. 345–362.

[16] C. Gormley and Z. Tong, *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine*. O'Reilly, 2015.

[17] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "Racs: A case for cloud storage diversity," in *Proceedings of the 1st ACM Symposium on Cloud Computing*, 2010, pp. 229–240.

[18] A. Bessani, M. Correia, B. Quaresma, F. Andre, and P. Sousa, "Depsky: dependable and secure storage in a cloud-of-clouds," *ACM Transactions on Storage (TOS)*, vol. 9, no. 4, p. 12, 2013.

[19] T. Oliveira, R. Mendes, and A. Bessani, "Exploring key-value stores in multi-writer byzantine-resilient register emulations," in *Proc. of the 20th Int. Conf. On Principles Of DIstributed Systems – OPODIS'16*, 2016.

[20] D. Dobre, P. Viotti, and M. Vukolić, "Hybris: Robust hybrid cloud storage," in *Proceedings of the ACM Symposium on Cloud Computing*. New York, NY, USA: ACM, 2014, pp. 12:1–12:14. [Online]. Available: http://doi.acm.org/10.1145/2670979.2670991

[21] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.