Generative Pattern Dissemination for Collaborative Intrusion Detection

A thesis presented for the degree of Master of Science
by Mike Petersen

First Reviewer: Prof. Dr. Ulrich Bühler
Second Reviewer: Prof. Dr. Sebastian Rieger

Department of Computer Sciences
University of Applied Sciences Fulda
Fulda, Germany
July 2021

Contents

Chapter 1

Introduction

1.1 Motivation

1.2 Objectives

1.3 Structure

Chapter 2

Preliminaries

2.1 Collaborative Intrusion Detection

2.2 Gaussian Mixture Models

2.3 Hash Functions

We define a hash function as $h : A \to B : \boldsymbol{p} \mapsto \boldsymbol{u}$ where $A \subset \mathbb{R}^d$ with $d \in \mathbb{N}$ is the set of real input data points and $B = \{0, 1\}^l$ with $l \in \mathbb{N}$ the set of all bit sequences of fixed size l, with $l < d$. Inputs $\boldsymbol{p}$ to hash functions are called *messages* and outputs $\boldsymbol{u}$ are called *digests*. A collision occurs when two messages $\boldsymbol{p}_1 \neq \boldsymbol{p}_2$ are projected onto the same digest $\boldsymbol{u} = h(\boldsymbol{p}_1) = h(\boldsymbol{p}_2)$. We further refer to a *family* of hash functions $\mathcal{H} : A \to B$ as a collection of hash functions that have the same domain and range, share a basic structure and are only differentiated by constants.

2.3.1 Cryptographic Hash Functions

In order to be suitable for cryptographic applications, a hash function h should have, among others, three basic properties. The *first preimage resistance* defines, that for any given $\boldsymbol{u} \in B$ it is infeasible to determine any $\boldsymbol{p} \in A$ such that $h(\boldsymbol{p}) = \boldsymbol{u}$, allowing h to be effectively one-way only. Furthermore, the *second preimage resistance* defines, that for any given $\boldsymbol{p} \in A$ it is infeasible to determine any other $\boldsymbol{p}' \in A$, $\boldsymbol{p} \neq \boldsymbol{p}$ which produces the same output, i.e. $h(\boldsymbol{p}) = h(\boldsymbol{p}')$. Lastly, the *strong collision resistance* states, that is infeasible to determine any two distinct $\boldsymbol{p} \neq \boldsymbol{p}' \in A$ such that $h(\boldsymbol{p}) = h(\boldsymbol{p}')$.

2.3.2 Non-Cryptographic Hashes

Applications that do not require the hash function to be resistant against adversaries, e.g. hash tables, caches or de-duplication, are usually implemented by using a hash function that exhibits relaxed guarantees on the properties defined before in **??** in exchange for significant performance improvements.

2.3.3 Locality Sensitive Hash Functions

Introduced as an algorithm that solves the cR-near neighbour problem, locality-sensitive hashing (LSH) [IM98] maps similar messages to the same digest with higher probability than dissimilar messages. More formally, given a threshold $R \in \mathbb{R}^{>0}$, an approximation factor $c \in \mathbb{R}^{>1}$ and probabilities $P_1, P_2 \in \mathbb{R}^{\geq 0}$, a family $\mathcal{H} : A \to B$ is called (R, cR, P_1, P_2)-sensitive if for any two points $\boldsymbol{p}_1, \boldsymbol{p}_2 \in A$ and any hash function h chosen uniformly at random from $\mathcal{H}$ the following conditions are satisfied:

- if $||\boldsymbol{p}_1 - \boldsymbol{p}_2|| \leq R$ then $P[h(\boldsymbol{p}_1) = h(\boldsymbol{p}_2)] \geq P_1$,
- if $||\boldsymbol{p}_1 - \boldsymbol{p}_2|| \geq cR$ then $P[h(\boldsymbol{p}_1) = h(\boldsymbol{p}_2)] \leq P_2$.

In order for a $\mathcal{H}$ to be applicable for the cR-near neighbour problem, it has to satisfy the inequality $P_1 > P_2$.

Chapter 3

State of the Art

3.1 Data Dissemination in Collaborative Intrusion Detection

The key components for designing a CIDS architecture are described in [Vas16, p. 34]. Among them, data dissemination is particularly noteworthy as one of the fundamental components for the communication between members. A central aspect is the communication *overhead* that is introduced with the dissemination of alerts and knowledge within a CIDS, which in turn is heavily influenced by the CIDS architecture [Vas16, p.39], that can be primarily categorized into centralized, hierarchical and decentralized approaches [ZLK10] (see Figure **??**). Centralized architectures, consisting of multiple monitoring units and a central analysis unit [CM02][MI03], suffer from a Single Point of Failure (SPoF) and are limited in their scalability due to a bottleneck, which is introduced by the central analysis unit. However, the SPoF problem can be mitigated if individual components in such a system are implemented redundantly and form a centralized architecture only at the logical level. A bottleneck, on the other hand, is usually avoided by a distributed implementation, which can also logically be regarded as a central data repository. Hierarchical designs exhibit multiple monitoring and analysis units organized in a tree-based topology [PN97; Zha+01; Ngu+19]. These systems are restricted in their scalability by the respective instances on higher levels, whose failure results in a malfunction of the respective sub-trees [ZLK10]. Again, such disadvantages only play a major role in non-redundantly implemented systems. Additionally, there exist approaches that are inherently distributed. Distributed architectures [Vas+15; Gil+13; BA08; Fun+08; JWQ03], wherein each participating system has a monitoring and analysis feature and where the communication is based on some form of data distribution protocol, are considered as scalable by design. However, they depend on effective and consistent data dissemination and the data attribute selected for correlation may affect load distribution among participants [ZLK10].

The flow of information in centralized and hierarchical architectures is governed by the logical arrangement of components and their specific role. In centralized architectures, there is usually a central entity that controls communication. In hierarchical systems, the communication paths are mainly governed by the topological structure. In contrast to that, different techniques exist in distributed approaches. Whereas flooding refers to unfiltered data distribution to all members, selective techniques reduce the overhead by using, e.g., random walks [VF06], gossipping approaches [Das+06][GKM03] or publish-subscribe mechanisms. The latter provide flexible organization options and guarantee delivery. For example, subscriptions utilized in [JWQ03] are based on special attack forms.

The data dissemination strategy of a CIDS not only defines the communication paths as described above, but also influences what kind of data is exchanged between the CIDS members while also specifying format and level of granularity. This also includes the central aspects of *data privacy*, realized by utilization of, e.g., bloom filters [Vas+15][Loc+05], and *interoperability* that can be obtained by standardized formats, such as the Intrusion Detection Message Exchange Format [CM02][Dum+06].

In particular, there are two approaches to mention, that directly address the problems of communication overhead an privacy in this context. In [Loc+05], the authors present an approach for the efficient and privacy aware distribution of alert data in a distributed CIDS. Before exchanging information, the respective data is compressed by the utilization of a bloom filter data structure. Upon an alert, the relevant information, e.g. IP address, is inserted ino the bloom filter. Since the bloom filter contains information on suspicious hosts, it is called *watchlist*. The watchlist is shared among peers, which are selected by a network scheduling algorithm called *whirlpool*, that dynamically creates an overlay that defines peer relationships.

The authors state, that within data dissemination, there two challenges. First, the disclosure of sensitive data, e.g. IP addresses, to collaborative entities renders participation in the collaboration not an option. Second, the tradeoff between latency and accuracy of the information exchange and the required bandwidth. Centralized approaches disseminate information reliable and predictable, but they constitute a bottleneck. While distributed approaches scale well, information can be lost or delayed by partitioning the data among the peers.

In the context of reducing communication overhead, this approach addresses this challenge twofold. First, alert data is compressed when inserted into the bloom filter by hashing. Second, only peers exchange data, which reduces overhead significantly, when compared to a complete distribution. However, bloom filters are probabilistic data structure, which exhibit increasing false positive matches with increasing filling degree. Thus, when scaling this approach, the probability that innocent hosts are considered as malicious, increases.

Also, the authors in [Vas+15] state, that a CIDS needs to provide *scalability*, minimal message *overhead*, *privacy* of the exchanged alert data, an *domain awareness*, which describes the ability to constrain alert disseminatation to specific sub-domains of a network. Instead of randomly creating sub-domains as in [Loc+05], the authors suggest to incorporate network traffic similarities into account for this process.

data dissemination: extract specific features from alerts and add data into bloom filter; then utilize data disseminatation technique (e.g. flooding, partial flooding, gossipping) for sending bloom filters to other nodes (peers)

similarity (alert) correlation: when nodes receive data from other nodes, they compute their similarity value by performing logical operations on the bloom filters

similarity (alert) correlation: when nodes receive data from other nodes, they compute their similarity value by performing logical operations on the bloom filters, after calculating the similarity value, nodes will make use of a threshold value t to determine wheteher the similarity value is enough for joining a group (community creation)

Community formation: After the successful dissemination and correlation of the alert data, each sensor creates a matrix with its local knowledge of other sensors. Based on this knowledge and along with the utilized threshold, sensors can identify others and form a community with them to, afterwards, exchange more fine-grained alert data.

The problem of finding an optimal threshold value (golden standard) heavily depends on the network that is to be monitored.

To sum up, the following challenges in the context of data dissemination are found to be critical for the success of the overall system.

Minimal Overhead Detection latency can be affected by various factors, some of which are encountered in conventional IDS and others that are specifically relevant in the CIDS context. With regards to the data dissemination in CIDS, the computational overhead introduced by the communication of multiple monitors in the CIDS needs to be minimized, such that potential knowledge gains are available fast enough.

Privacy Members may not want to disclose data that contains information on system- and network states of their infrastructure, as it constitutes a privacy and security problem. This includes, among other things, legal aspects when it comes to sharing log and network data. Nonetheless, the exchange of this information is crucial for the effective operation of a CIDS.

Interoperability The individual components of the overall system, which were deployed in different system and network environments, should be able to interact with each other in the context of the CIDS. In addition to system-wide standards for data collection, processing and exchange, there exists a trade-off between interoperability and privacy.

Domain Awareness t.b.d. (a feature that increases scalability by reducing overhead and contributes to privacy by partially constraining communication; also increases accuracy, because only those alerts are shared that are relevant for w.r.t. to similarity of data etc.)

3.2 Similarity Hashing in Malware Detection

Searching for similar objects in large data sets is a fundamental challenge that has found important applications in many fields. An important subclass of similarity search is the nearest neighbour search problem, which becomes hard in the high dimensional case, when relying on exact algorithms like a linear scan ($O(n)$) as the best solution. However, many applications do not require an exact solution, such that in these cases a randomized algorithm can be used, which provides the correct solution with high probability in sublinear time [Dat+04]. Therefore, approximate and randomized solution algorithms are widely used in practice. There is popular approach known as locality-sensitive hashing (LSH) that allows searching in large databases for similar items in a randomized manner. This technique hashes similar input onto the same hash code with high probability for the purpose of populating a hash table. Since similar items are placed into the same buckets, this approach is suitable for data clustering or nearest neighbour search.

Also the field of cyber security has adopted methods suitable for similarity search, mainly for the analysis of polymorphic malware. This type of malware poses a significant challenge, since it changes its appearance over time in order to stay undetectable from antivirus software [WM18, p.91]. Traditional antivirus software uses cryptographic hash functions (SHA-256) to create file signatures. Such signatures are well suited to search for identical files in a knowledge database. However, due to the property of cryptographic diffusion, even minimal changes to the malware result in large differences in the resulting hash value. With the rapid evolution and proliferation of polymorphic malware, detection based on unique signatures no longer seems effective.

Based on these developments, detection schemes based on approximate matching algorithms have initially become the focus of research. In contrast to LSH, approximate matching functions are designed for producing digests of objects and a subsequent comparison of such digests, which yields a confidence value reflecting the similarity of two objects [MH17]. Popular approaches in this area are for example *ssdeep*, which implements context triggered piecewise hashing (CTPH) [Kor06] or *sdhash*, that makes use of bloom filters for the digest creation and comparison [Rou10].

3.3 Generative Algorithms and Intrusion Detection

Chapter 4

Generative Pattern Database

Three main challenges in the context of data dissemination in CIDS were identified. First, intrusion related data is usually of sensitive nature. Thus, the exchange mechanism must not compromise *privacy* policies and regulations. At the same time, the usability of the data has to be preserved. Second, the data that is subject of the exchange may exhibit large volumes. That constitutes a challenge, since the dissemination is desired to be executed with a *minimal overhead* in a timely and scalable fashion. Lastly, the *interoperability* of the CIDS with existing local IDS is an important aspect that influences the adoption and operational usability in practice.

In summary, existing approaches for data dissemination mainly provide mechanisms for exchanging alert data or single attributes, e.g. IP addresses, as they focus on the correlation of intrusion detection incidents that originate from different sensors. The exchange of actual training data is neglected, possibly due to high data volumes. Thus, these systems lack of mechanisms for the extraction and global persistence of novel attack patterns, e.g. zero day exploits, that can be used for the training of an intrusion detection sensor.

The approach that is presented in this chapter exchanges attack patterns by sharing generative machine learning models that have been trained on partitions of similar data points. Such a model-based dissemination enables the receiving side to sample a synthetic dataset that enhances existing local datasets. This provides two main advantages. First, no original data leaves a local network and therefore does not violate any privacy restrictions. Second, the data is compressed considerably by representing it in form of a generative model. In order to make that mechanism scalable, the monitored data is clustered using random projections. This way, similar data points are partitioned into globally common clusters and provide data parallelism, such that updates on the database is done efficiently in a cluster-wise fashion. Furthermore, this mechanism enables a similarity-based correlation of distributed intrusion events. The integration of both a similarity based correlation of intrusion incidents and a mechanism for sharing attack knowledge makes it possible to extract novel patterns of distributed attacks and provide them globally within the CIDS, resulting in an improved attack detection.

While Section 4.1 gives a high-level overview of the proposed architecture and its main processing primitives, details on the specific algorithms and strategies of the main services are described in Sections 4.2 to 4.5.

4.1 System Architecture

Logically, the proposed CIDS exhibits a hierarchical architecture (see Figure 4.1). For one, the global infrastructure G represents the collection of $M \in \mathbb{N}$ CIDS participants and their knowledge on an abstract level. For another, it provides specific services, that are globally available to each local infrastructure $L_m, m \in \{1, \ldots, M\}$ that includes all CIDS components and processes within the IT infrastructure boundaries of a corresponding CIDS member. Each local infrastructure L_m agrees to a specified feature extraction process that provides the monitoring data $\boldsymbol{X} \subset \mathbb{R}^d$ for the attack detection. Single data points of the monitoring data are referred to as $\boldsymbol{x} \in \boldsymbol{X}$ with a total number of features $d = |\boldsymbol{x}| \in \mathbb{N}$. In addition, the set of targets $Y \subset \mathbb{N}$ with instances $y \in Y$ is known and registered by every local infrastructure L_m. Furthermore, every L_m prodives an individual training dataset $D_m = \{(\boldsymbol{x}_n, y_n) : 1 \leq n \leq N_m\}$ of size $N_m = |D_m| \in \mathbb{N}$. CIDS communication across local boundaries occurs exclusively in a vertical direction. Thus, the exchange of information between individual L_m takes place indirectly via the global pattern database (PDB_G) and the global event channel (C_G). Each L_m includes a local pattern database (PDB_{L_m}), a local event channel (C_{L_m}) and an event-based data processing pipeline that consists of four services.

4.1.1 Pattern Database

Each instance of a pattern database (PDB) is realized as a key-value store. For the following algorithm descriptions, a PDB is treated as a hash table as defined in Section **??**, referred to by replacing the function variable with the service identifier, e.g. $PDB_G(k)$ being a specific slot or hash value related to a key k in the global pattern database. Note that if a specific hash function is already used to construct a key (e.g. Random Projection), the hash table will internally apply a distinct hash function on the key to ensure even distribution across the slots.

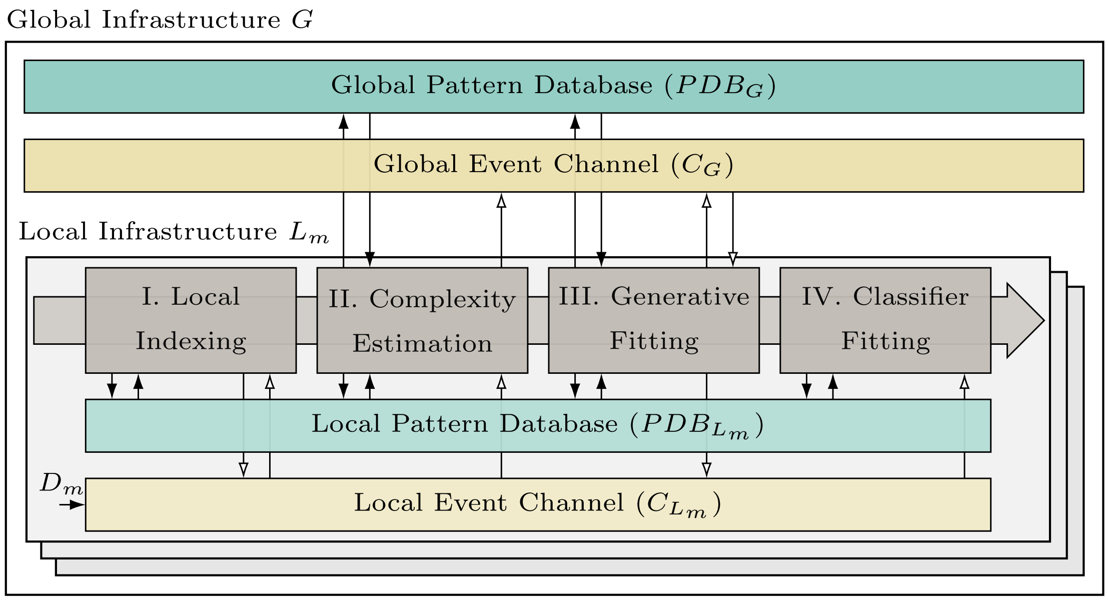
Global Infrastructure G
Global Pattern Database (PDB_G)
Global Event Channel (C_G)
Local Infrastructure L_m
I. Local Indexing
II. Complexity Estimation
III. Generative Fitting
IV. Classifier Fitting
Local Pattern Database (PDB_{L_m})
D_m
Local Event Channel (C_{L_m})

Figure 4.1: High level CIDS architecture.

4.1.2 Event Channel

Event channels provide a topic-based publish-subscribe messaging mechanism that is mainly used to distribute workloads among the service instances in the processing pipeline. Via the messaging system, service instances receive and emit events, on which upon the respective operations are triggered. Changes in a pattern database result in responses that in turn are leveraged as the respective events. In this fashion, updates are propagated throughout the processing pipeline, ensuring a timely consistency among the pattern databases.

4.1.3 Processing Pipeline

Local and global pattern databases serve exclusively as data sources and sinks for operations. The only exception is the initial import of datasets D_m into the *Local Indexing* service via the messaging system.

4.2 Local Indexing

The local indexing service listens on C_{L_m} for new data points from D_m. First, data points are buffered until a defined batch size is released. Subsequently, the batch X is scaled to the range $[-1, 1]$ by applying a feature-wise min-max normalization, given by

$$\boldsymbol{X}' = \frac{\boldsymbol{X} - min(\boldsymbol{X})}{max(\boldsymbol{X}) - min(\boldsymbol{X})} \cdot (b - a) + a, \quad \begin{array}{rr} a = & -1 \\ b = & 1 \end{array}.$$

After that, each element $\boldsymbol{x}' \in \boldsymbol{X}'$ is subject to both a *GRP* h with a global seed for the initialization of the projection plane $\boldsymbol{M}$ (see Section **??**) and a non-cryptographic hashing function g (see Section 2.3.2). The *GRP* clusters the data by mapping similar data points $\boldsymbol{x}'$ to the same hash value. The non-cryptographic hashing function on the other hand serves as a mechanism for deduplicating identical $\boldsymbol{x}'$. In order to insert a pair $(\boldsymbol{x}'_n, y_n) \in D_m$ into the PDB_{L_m}, at first two keys $k_\alpha, k_\beta \in K$ are constructed as

$$\begin{aligned} k_\alpha(\boldsymbol{x}'_n, y_n) &= p_x +\!\!+\, h(\boldsymbol{x}'_n) +\!\!+\, y_n, \\ k_\beta(\boldsymbol{x}'_n) &= g(\boldsymbol{x}'_n), \end{aligned}$$

Notation	Description
G	Global Infrastructure
L_m	Local Infrastructure m
PDB_G, PDB_{L_m}	Global Pattern Database, Local Pattern Database of L_m
C_G, C_{L_m}	Global Event Channel, Local Event Channel of L_m
$M \in \mathbb{N}$	Total number of CIDS participants
$m \in \{1, \ldots, M\}$	Local Infrastructure Identifier

Table 4.1: Summary of the architecture notation.

where $+\!\!+$ indicates the concatenation operator and p_x describes an arbitrary prefix constant. Secondly, if not already present, a hash table T_{m_n} is created and inserted into the slot $PDB_{L_m}(k_\alpha(\boldsymbol{x}'_n, y_n))$. Finally, $(\boldsymbol{x}'_n, y_n)$ is hashed into the slot $T_{m_n}(k_\beta(\boldsymbol{x}'_n))$. That nested indexing is depicted in Figure 4.2. Since a particular $h(\boldsymbol{x}')$ is a cluster of similar data points, we refer to it as a *region*. Therefore, every region is represented by one or more hash tables T_{m_n}, each of them containing data points that belong to the same label. Each execution of that insert operation is executed idempotently, such that the pattern database only returns a response upon the insertion of new data points. For each response, the corresponding region $h(\boldsymbol{x}'_n)$ is sent into the local event channel C_{L_m} as an event that indicates a change on a certain cluster within the local pattern database.

4.3 Complexity Estimation

A region is said to be complex, if it contains more than one unique label. Otherwise, a region is simple. Since the data within a region already represents a cluster, the existence of multiple classes indicates a more complex decision boundary. On that basis we differentiate how a region is processed in the subsequent services of the pipeline. Furthermore, the complexity state of a region may vary, depending on the scope it is observed. Note that since the projection matrix $\boldsymbol{M}$ is initialized with the same values in every L_m, all hashes that were computed by h are globally comparable. This means that similar data points from different datasets, e.g. $\boldsymbol{x}_i \in D_1$ and $\boldsymbol{x}^*_j \in D_2$ may be hashed to the same region $h(\boldsymbol{x}_i) = h(\boldsymbol{x}^*_j)$. However, it is also possible that that the corresponding labels $y_i \in D_1$ and $y^*_j \in D_2$ are not equal and therefore lead to a different global view on that region's complexity state. Given that, the complexity estimation module acts as a bridge between the local and global components and answers the question, which regions are considered to be complex in a global context. First, an event containing a region $h(\boldsymbol{x}'_n)$ is received. This event is then used for retrieving the set of unique labels $Y_{h(\boldsymbol{x}'_n)}$ for that particular region stored at PDB_{L_m}, which is defined as

$$\bigcup_{y \in Y} \Big\{ y_n \in \Big\{ T_{m_n}\Big(k_\beta(\boldsymbol{x}'_n)\Big)\Big\} : T_{m_n} \in \Big\{ PDB_{L_m}\Big(k_\alpha(\boldsymbol{x}'_n, y)\Big)\Big\}\Big\}.$$

Next, a key $k_\gamma \in K$ is constructed by concatenating an arbitrary prefix constant p_y with the region $h(\boldsymbol{x}'_n)$ and the id of the current local infrastructure m:

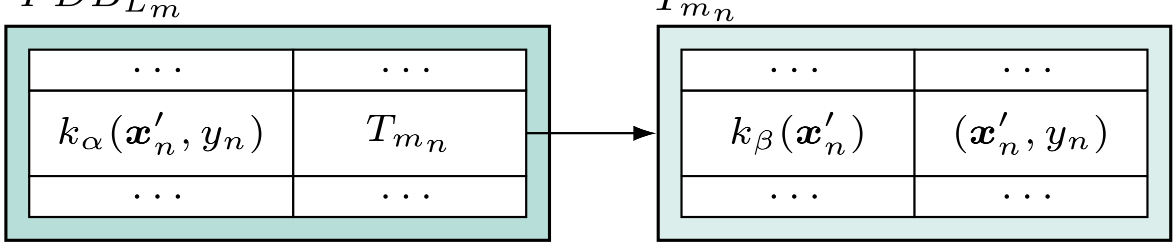
PDB_{L_m}
T_{m_n}
$k_\alpha(\boldsymbol{x}'_n, y_n)$
T_{m_n}
$k_\beta(\boldsymbol{x}'_n)$
$(\boldsymbol{x}'_n, y_n)$

Figure 4.2: Nested indexing in a local pattern database.

$$k_\gamma(h(\boldsymbol{x}'_n), m) = p_y + h(\boldsymbol{x}'_n) + m.$$

Then, $Y_{h(\boldsymbol{x}'_n)}$ is inserted into the global pattern database at the slot $PDB_G(k_\gamma(h(\boldsymbol{x}'_n), m))$. The global complexity $c_{h(\boldsymbol{x}'_n)} \in \{0, 1\}$ is determined by combining the respective label sets from all local infrastructures and evaluating its cardinality:

$$c_{h(\boldsymbol{x}'_n)} = \left| \bigcup_{m \in M} \left\{ PDB_G\Big(k_\gamma(h(\boldsymbol{x}'_n), m)\Big) \right\} \right| > 1.$$

Finally, the state $c_{h(\boldsymbol{x}'_n)}$ is inserted in the global pattern database by constructing a key $k_\kappa \in K$ with a prefix constant p_c and the region $h(\boldsymbol{x}'_n)$ as

$$k_\kappa(h(\boldsymbol{x}'_n)) = p_c + h(\boldsymbol{x}'_n)$$

and storing it in the slot $PDB_G(k_\kappa(h(\boldsymbol{x}'_n)))$. If the state has changed, a response is returned and the corresponding region $h(\boldsymbol{x}'_n)$ is sent as an event into the global event channel C_G.

4.4 Generative Fitting

4.5 Classifier Fitting

Chapter 5

Experimental Evaluation

Chapter 6

Conclusion

Bibliography

[PN97] Phillip A. Porras and Peter G. Neumann. “EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances”. In: *National Information Systems Security Conference* (1997).

[IM98] Piotr Indyk and Rajeev Motwani. “Approximate nearest neighbors: towards removing the curse of dimensionality”. In: *ACM Symposium on Theory of Computing* (1998).

[Zha+01] Zheng Zhang et al. “HIDE: a Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification”. In: *IEEE Workshop on Information Assurance and Security* (2001).

[CM02] Frédéric Cuppens and Alexandre Miege. “Alert correlation in a cooperative intrusion detection framework”. In: *Proceedings 2002 IEEE symposium on security and privacy*. IEEE. 2002, pp. 202–215.

[GKM03] Ayalvadi J Ganesh, A-M Kermarrec, and Laurent Massoulié. “Peer-to-Peer Membership Management for Gossip-Based Protocols”. In: *IEEE Transactions on Computers* (2003).

[JWQ03] R. Janakiraman, M. Waldvogel, and Qi Zhang. “Indra: A Peer-to-Peer Approach to Network Intrusion Detection and Prevention”. In: *IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (2003).

[MI03] Patrick Miller and Atsushi Inoue. “Collaborative intrusion detection system”. In: *22nd International Conference of the North American Fuzzy Information Processing Society, NAFIPS 2003*. IEEE. 2003, pp. 519–524.

[Dat+04] Mayur Datar et al. “Locality-sensitive hashing scheme based on p-stable distributions”. In: *Proceedings of the twentieth annual symposium on Computational geometry - SCG ’04*. Brooklyn, New York, USA: ACM Press, 2004, p. 253. ISBN: 978-1-58113-885-6. DOI: `10.1145/997817.997857`. URL: `http://portal.acm.org/citation.cfm?doid=997817.997857` (visited on 01/14/2022).

[Loc+05] M. E. Locasto et al. “Towards Collaborative Security and P2P Intrusion Detection”. In: *IEEE SMC Information Assurance Workshop* (2005).

[Das+06] Denver Dash et al. “When Gossip is Good: Distributed Probabilistic Inference for Detection of Slow Network Intrusions”. In: *AAAI National Conference on Artificial Intelligence* (2006).

[Dum+06] Claudiu Duma et al. “A Trust-Aware, P2P-Based Overlay for Intrusion Detection”. In: *International Workshop on Database and Expert Systems Applications* (2006).

[Kor06] Jesse Kornblum. “Identifying almost identical files using context triggered piecewise hashing”. en. In: *Digital Investigation* 3 (Sept. 2006), pp. 91–97. ISSN: 17422876. DOI: 10.1016/j.diin.2006.06.015. URL: https://linkinghub.elsevier.com/retrieve/pii/S1742287606000764 (visited on 01/21/2022).

[VF06] Vivek Vishnumurthy and Paul Francis. “On Heterogeneous Overlay Construction and Random Node Selection in Unstructured P2P Networks”. In: *IEEE International Conference on Computer Communications* (2006).

[BA08] Rainer Bye and Sahin Albayrak. *CIMD - Collaborative Intrusion and Malware Detection.* 2008.

[Fun+08] Carol J. Fung et al. “Trust Management for Host-Based Collaborative Intrusion Detection”. In: *Managing Large-Scale Service Deployment* (2008).

[Rou10] Vassil Roussev. “Data Fingerprinting with Similarity Digests”. en. In: *Advances in Digital Forensics VI.* Ed. by Kam-Pui Chow and Sujeet Shenoi. Vol. 337. Series Title: IFIP Advances in Information and Communication Technology. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 207–226. DOI: 10.1007/978-3-642-15506-2_15. URL: http://link.springer.com/10.1007/978-3-642-15506-2_15 (visited on 01/21/2022).

[ZLK10] Chenfeng Vincent Zhou, Christopher Leckie, and Shanika Karunasekera. “A survey of coordinated attacks and collaborative intrusion detection”. In: *Computers & Security* 29.1 (2010), pp. 124–140.

[Gil+13] Manuel Gil Pérez et al. “RepCIDN: A Reputation-Based Collaborative Intrusion Detection Network to Lessen the Impact of Malicious Alarms”. In: *Journal of Network and Systems Management* (2013).

[Vas+15] Emmanouil Vasilomanolakis et al. “SkipMon: A locality-aware Collaborative Intrusion Detection System”. In: *IEEE International Performance Computing and Communications Conference* (2015).

[Vas16] Emmanouil Vasilomanolakis. “On Collaborative Intrusion Detection”. PhD thesis. Darmstadt: Technische Universität Darmstadt, 2016. URL: http://tubiblio.ulb.tu-darmstadt.de/82583/.

[MH17] Vitor Hugo Galhardo Moia and Marco Aurélio Amaral Henriques. “Similarity Digest Search: A Survey and Comparative Analysis of Strategies to Perform Known File Filtering Using Approximate Matching”. en. In: *Security and Communication Networks* 2017 (2017), pp. 1–17. ISSN: 1939-0114, 1939-0122. DOI: 10.1155/2017/1306802. URL: https://www.hindawi.com/journals/scn/2017/1306802/ (visited on 01/24/2022).

[WM18] Michael E. Whitman and Herbert J. Mattord. *Principles of information security.* Cengage Learning, 2018. ISBN: 978-1-337-10206-3.

[Ngu+19] Tri Gia Nguyen et al. “Search: A collaborative and intelligent nids architecture for sdn-based cloud iot networks”. In: *IEEE access* 7 (2019), pp. 107678–107694.