

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

1	Introduction	1
1.1	Motivation	1
1.2	Objectives	1
1.3	Structure	1
2	Preliminaries	3
2.1	Collaborative Intrusion Detection	3
2.2	Gaussian Mixture Models	3
2.3	Hash Functions	3
2.3.1	Cryptographic Hashes	3
2.3.2	Locality-Sensitive Hashes	4
2.3.3	Hash Table	4
2.3.4	Random Projection	4
3	State of the Art	7
3.1	Data Dissemination in Collaborative Intrusion Detection	7
3.2	Similarity Hashing in Malware Detection	9
3.3	Generative Algorithms and Intrusion Detection	10
4	Generative Pattern Database	11
4.1	High Level Overview	12
4.1.1	Example Integration	12
4.1.2	Clustering	13
4.1.3	Filtering and Compression	13
4.1.4	Global View	14
4.1.5	Creation	14
4.1.6	Usage	14
4.2	System Architecture	15
4.2.1	Pattern Database	15
4.2.2	Event Channel	16
4.2.3	Processing Pipeline	16
5	Experimental Evaluation	17
6	Conclusion	19

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

A hash function is a function that maps a large input set to a smaller target set. The elements of the input set are called *messages* or *keys* and may be of arbitrary different lengths. The elements of the target set are called *digests* or *hash values* and are of fixed size length. More specifically, we define a hash function as $h : A \rightarrow B : \mathbf{p} \mapsto \mathbf{u}$ where $A \subset \mathbb{R}^d$ with $d \in \mathbb{N}$ is the input set and $B = \{0, 1\}^k$ with $k \in \mathbb{N}$ the target set of all bit sequences of fixed size k , with $k < d$.

Typically, hash functions are used for the realization of, e.g. hash tables, data integrity checks, error correction methods or database indexes. Depending on the application, different requirements are imposed on the utilized hash function. In this context, the most important property of a hash function is the probability of a *collision*. A collision occurs when two keys $\mathbf{p}_1 \neq \mathbf{p}_2$ are projected onto the same hash value $\mathbf{u} = h(\mathbf{p}_1) = h(\mathbf{p}_2)$.

Two different types of hash functions were used to develop basic components of the generative pattern database. First, in Section 2.3.1, cryptographic hashes are defined by describing their strict properties. Next, in Section 2.3.2 locality sensitive hashes are introduced as a contrasting concept to cryptographic hash functions. Finally, a specific family of hash functions that is locality-sensitive is explained in Section 2.3.4.

2.3.1 Cryptographic Hashes

Security Requirements for cryptographic hash functions [Wil, p. 349]

Definition 1 (First preimage resistance). For any given $\mathbf{u} \in B$ it is infeasible to determine any $\mathbf{p} \in A$ such that $h(\mathbf{p}) = \mathbf{u}$.

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 $\mathbf{u} \in B$ it is infeasible to determine any $\mathbf{p} \in A$ such that $h(\mathbf{p}) = \mathbf{u}$, allowing h to be effectively one-way only. Furthermore, the *second preimage resistance* defines, that for

any given $\mathbf{p} \in A$ it is infeasible to determine any other $\mathbf{p}' \in A$, $\mathbf{p} \neq \mathbf{p}'$ which produces the same output, i.e. $h(\mathbf{p}) = h(\mathbf{p}')$. Lastly, the *strong collision resistance* states, that is infeasible to determine any two distinct $\mathbf{p} \neq \mathbf{p}' \in A$ such that $h(\mathbf{p}) = h(\mathbf{p}')$.

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.

2.3.2 Locality-Sensitive Hashes

We further refer to a *family* of hash functions $\mathcal{H} : A \rightarrow B$ as a collection of hash functions that have the same domain and range, share a basic structure and are only differentiated by constants. 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 $\mathcal{H} : A \rightarrow B$ is called (R, cR, P_1, P_2) -sensitive if for any two points $\mathbf{p}_1, \mathbf{p}_2 \in A$ and any hash function h chosen uniformly at random from \mathcal{H} the following conditions are satisfied:

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

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

2.3.3 Hash Table

We describe a hash table operationally by defining a hash function h , that maps the keyspace K into the slots of a hash table $T[0..S-1]$, $S \in \mathbb{N}$, i.e. $h : K \rightarrow \{0, 1, \dots, S-1\}$ with $S \ll |K|$. Thus, a message with key $k \in K$ hashes to slot $h(k)$. Additionally, $h(k)$ is the hash value of the key k .

2.3.4 Random Projection

Given a message $\mathbf{x} \in A = \{x \in \mathbb{R}^d : 0 \leq x \leq 1\}$ and a randomly selected hyperplane defined as $\mathbf{M} = (a_{ij}) \in \mathbb{R}^{d \times k}$ where $a \sim \mathcal{N}(0, I)$, a *gaussian random projection (GRP)* aims to (I) reduce the dimensionality from d to l dimensions and (II) provide a binary encoding by first projecting \mathbf{x} onto \mathbf{M} and subsequently applying the sign function to each element of the result, i.e. :

$$h(\mathbf{x}) = [h(\mathbf{x}, a_1), \dots, h(\mathbf{x}, a_k)] \text{ with } h(\mathbf{x}, a) = \text{sign}(\mathbf{x}^T a).$$

The resulting digest is a binary vector $h(\mathbf{x}) = \mathbf{u} \in B = \{0, 1\}^l$ that is commonly used as bucket index for storing \mathbf{x} in a hash table. For any two messages $\mathbf{x}_1, \mathbf{x}_2$, the probability of being hashed to the same bucket increases with a decreasing distance, which is given by the angular distance as

$$P[h(\mathbf{x}_1) = h(\mathbf{x}_2)] = 1 - \frac{\theta(\mathbf{x}_1, \mathbf{x}_2)}{\pi}$$

with $\theta(\mathbf{x}_1, \mathbf{x}_2) = \cos^{-1}\left(\frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{\|\mathbf{x}_1\| \|\mathbf{x}_2\|}\right)$.

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], gossiping 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 and 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 into 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 are 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 reliably and predictably, 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 dissemination 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 dissemination technique (e.g. flooding, partial flooding, gossiping) 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 whether 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 any policies and regulations related to data *privacy*. 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 *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 practical adoption into security architectures.

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. from 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, which is exploited as a data parallelism mechanism. Given that, bursty workloads can be served effectively in a cloud deployment. 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 4.2 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 ?? to ??.

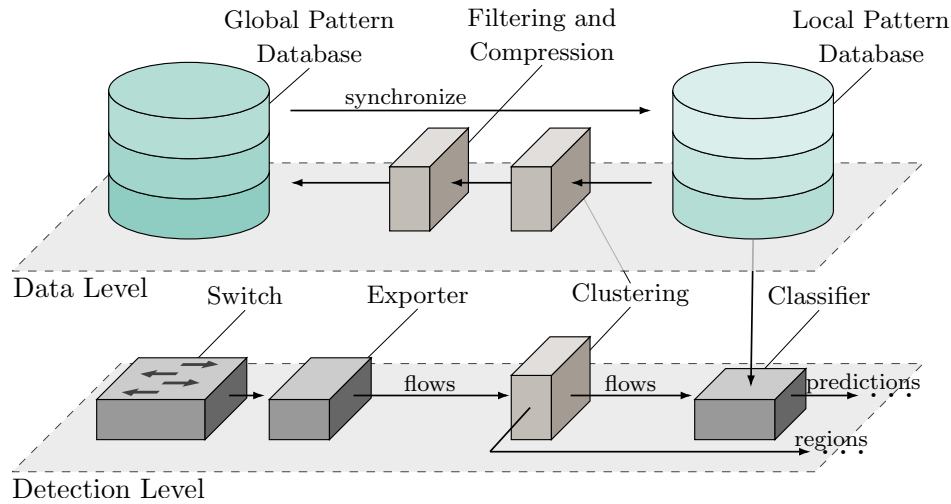


Figure 4.1: Integration of the approach into a generic NIDS.

4.1 High Level Overview

Several members exist in the CIDS, each of which manages an isolated IDS. Every IDS operates according to a specific set of rules, that is essentially based on the content of a local database. The goal of the generative pattern database is to close the knowledge gaps of local databases and thus increase the detection rate of associated IDSs. By providing a *global view* on all local databases, individual IDSs can benefit from the collective knowledge of the CIDS.

Section 4.1.1 starts with a reference example to illustrate the idea that is described above and discusses the integration of the CIDS into existing infrastructures. Subsequently, Section 4.1.2 and Section 4.1.3 show the key concepts that enable data distribution and correlation under the given requirements. Finally, Section 4.1.4 combines the individual elements to present the strategy for the creation and usage of the global view.

4.1.1 Example Integration

An exemplary integration of the CIDS into a generic NIDS is shown in Figure 4.1. The shown NIDS consists of three components, which can be found on the detection layer. A flow exporter computes statistical flow features based on the network packets of a switch. A discriminative model serves as a classifier that operates on the specific feature set that the exporter extracts. After completing the training of a classifier instance on a given dataset, it is deployed within the detection pipeline. There, the classifier receives a stream of network flows and predicts them.

The CIDS mainly integrates at the data level, where the training data for the classifier is provided by the *local pattern database*. At this point, the local pattern database only contains the *local view* of the intrusion detection data from that particular member. In order to provide a global view for that member, a synchronization process between its local pattern database and the *global pattern database* has to be initiated. Before the data is transferred to the global pattern database, is subject to a set of clustering, filtering and compression operations. This way, data privacy is maintained and the overhead is minimized by reducing the data volume. On the receiving side, the global pattern database combines data from all members to a global view.

Upon each update of the global pattern database, the new state of the global view is synchronized to each local pattern database and subsequently enhances the classifier on the detection level by extending the local intrusion detection dataset. Furthermore, an identical clustering operation as on the data level is applied on the detection level. Each incoming flow is assigned to a certain cluster, which is referred to as *region*. While the predictions from the classifier are suitable for detecting attacks that are known to the CIDS, regions are leveraged for the detection of novel data patterns and similarity-based correlations that uncover stealthy attacks, which are executed on the resources of multiple CIDS members simultaneously.

4.1.2 Clustering

Since the results of the clustering operation should be consistent, while its execution is distributed among all members of the CIDS, an unsupervised algorithm with few parameters for initialization has to be selected. Additionally, the algorithm should be scalable, since it is to be applied on whole databases on the data level and on streams of live data on the detection level. Given these requirements, random projection is a good choice.

By using random projection, real data points are clustered according to their angular distance. Furthermore, the projection result is a binary string, which can be used for storing similar data points into a common bucket of a hash table by using the binary string as an index. In the context of the generative pattern database, the combination of random projections and hash tables is exploited as the main controlling primitive for data persistence and retrieval. Instead of using randomly selected projection planes, a shared seed results in the application of a common projection function among all members of the collaboration. In other words, similar data points from different datasets are indexed to a common global bucket, i.e. region. Each region is subject to the transformations individually, which is utilized as a data parallelism mechanism. Thus, this approach is natively suited for cloud deployments where bursty workloads can be served effectively. Lastly, this mechanism enables a similarity-based correlation of distributed intrusion events. As incoming data is monitored on the detection level, the clustering is applied, which results in a pattern that can be used for novelty checks or global occurrences within the CIDS.

4.1.3 Filtering and Compression

Two types of data are extracted within individual regions. First, metadata of local datasets is collected by counting label occurrences, which serve as indicator for determining if the respective region needs to be subject to the second extraction type. Second, models that are trained with generative algorithms on local attack data are the exchange medium for disseminating information within the CIDS. This provides two main advantages. For one, no original data leaves a local network and therefore does not violate any privacy restrictions. For another, the data is compressed considerably by representing it in form of a generative model.

4.1.4 Global View

4.1.5 Creation

As shown in Figure 4.2, each view is partitioned by a common projection function into an identical set of regions. Each local view contains original datapoints from its respective dataset. In this example, there exist three different classes globally, which occur differently in each local dataset. Examining a specific region, the combination of unique classes within all local views determines its complexity on a global level. If a region contains more than one class, it potentially exhibits a non-linear decision boundary, hence it is called complex. Otherwise, a region is called simple. Attack data within a complex region, separated by its label, is used as training data for a generative algorithm. Subsequently, the resulting model is transferred to the global view.

4.1.6 Usage

A synchronization process disseminates the region complexity estimations and the generative models to all local views. By sampling data from multiple generative models, a synthetic dataset is assembled, which is blended into the respective local datasets for enhancing the subsequent training of a discriminative model.

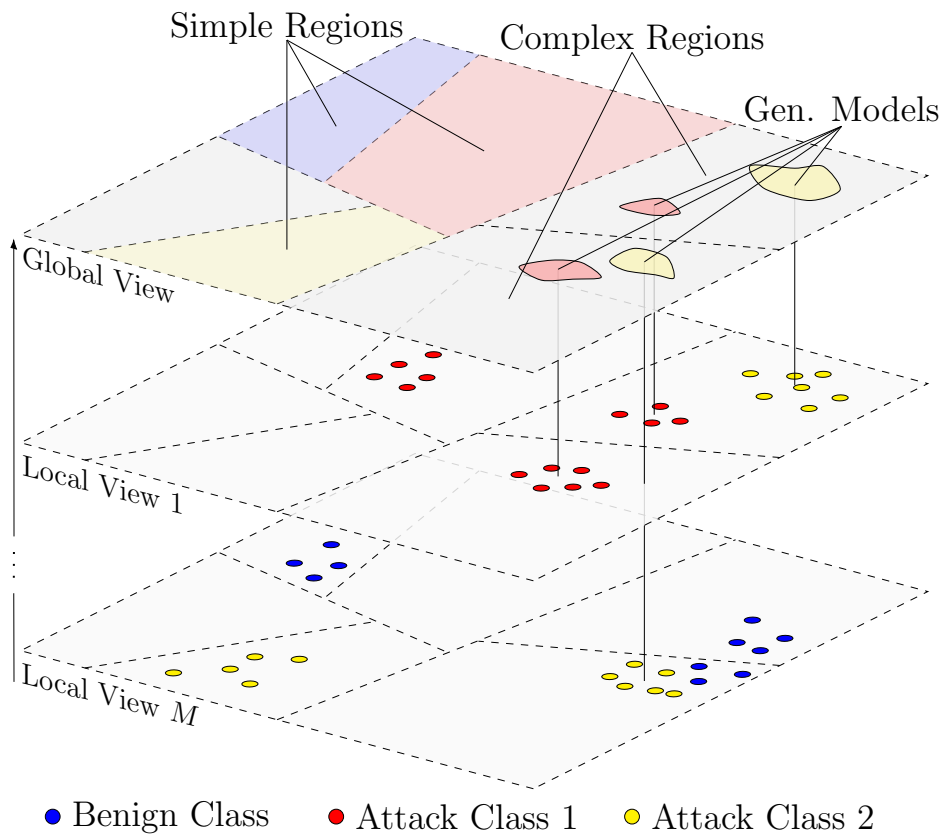


Figure 4.2: Building the global view by combining M local views.

4.2 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 $L_m, m \in \{1, \dots, M\}$ 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 $\mathbf{X} \subset \mathbb{R}^d$ for the attack detection. Single data points of the monitoring data are referred to as $\mathbf{x} \in \mathbf{X}$ with a total number of features $d = |\mathbf{x}| \in \mathbb{N}$. 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 provides an individual training dataset $D_m = \{(\mathbf{x}_n, y_n) : 1 \leq n \leq N_m\}$ 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.

4.2.1 Pattern Database

depending on the scope (either local or global), different tasks are considered local pdbs store intrusion detection datasets and corresponding metadata of the respective member; global pdbs store global metadata (combined information of local datasets) and the generative models

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 2.3.3, that is referred to by replacing its function variable with the identifier of the corresponding database, 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

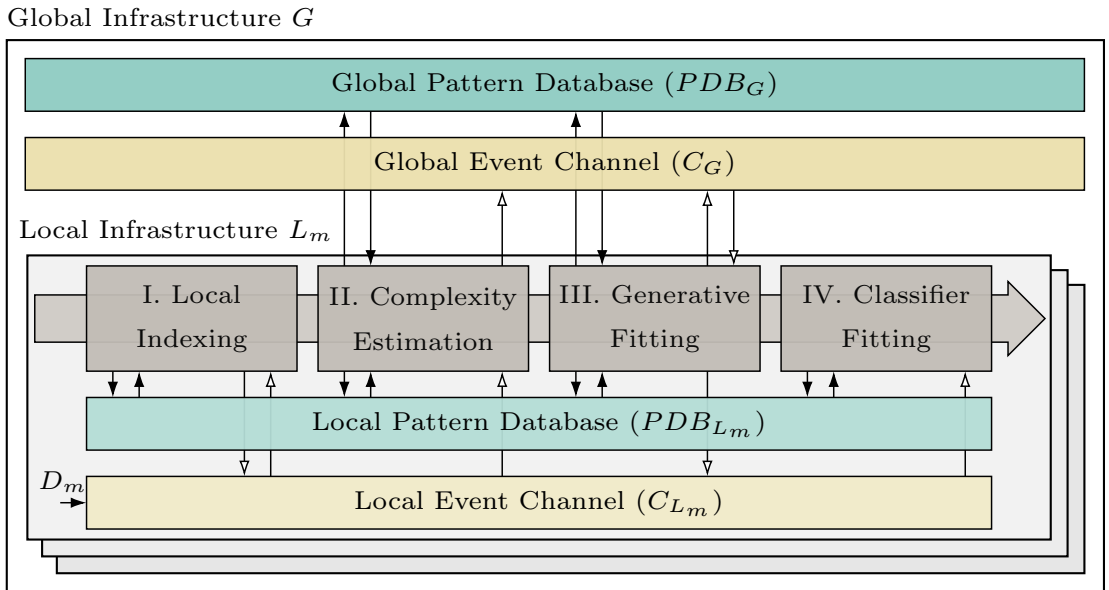


Figure 4.3: High level CIDS architecture.

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.

4.2.2 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.

4.2.3 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.

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
$m \in \{1, \dots, M\}$	Local Infrastructure Identifier

Table 4.1: Summary of the architecture notation.

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 Mieke. “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](https://doi.org/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](https://doi.org/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 Sheno. 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](https://doi.org/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](https://doi.org/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.
- [Wil] S. William. *Cryptography and Network Security - Principles and Practice, 7th Edition*. Pearson Education India. ISBN: 9789353942564. URL: <https://books.google.de/books?id=AhDCDwAAQBAJ>.