Detecting Attacks in Network Traffic Using Normality Models: The Cellwise Estimator*

Felix Heine, Carsten Kleiner, Philip Klostermeyer, Volker Ahlers, Tim Laue, and Nils Wellermann

Hannover University of Applied Sciences & Arts, Hannover, Germany
{firstname.lastname}@hs-hannover.de

Abstract. Although machine learning (ML) for intrusion detection is attracting research, its deployment in practice has proven difficult. Major hindrances are that training a classifier requires training data with attack samples, and that trained models are bound to a specific network.
To overcome these problems, we propose two new methods for anomaly-based intrusion detection. Both are trained on normal-only data, making deployment much easier. The first approach is based on One-class SVMs, while the second leverages our novel Cellwise Estimator algorithm, which is based on multidimensional OLAP cubes. The latter has the additional benefit of explainable output, in contrast to many ML methods like neural networks. The created models capture the normal behavior of a network and are used to find anomalies that point to attacks. We present a thorough evaluation using benchmark data and a comparison to related approaches showing that our approach is competitive.

Keywords: Network Intrusion Detection · Machine Learning · Anomaly Detection · Multidimensional Data · OLAP Cubes · Iceberg Condition

1 Introduction

In this paper, we introduce a novel approach to anomaly-based intrusion detection systems (IDS). Most commercial IDS are based on static rules that need to be maintained and are thus being called rule-based. In the research community, many approaches try to exploit machine learning (ML) to build what is often called an anomaly-based IDS. A common claim is that anomaly-based approaches are capable of detecting new types of attacks, in contrast to rule-based IDS, that can only detect known attacks. While the latter is true for sure, the correctness of the former claim depends on the exact method.

Many publications use some form of classification, either binary or multi-class classification, to build a model that is used to detect attacks [6], with neural networks and decision trees being among the most popular algorithms. However, a classifier needs training data containing samples for each attack type

* The GLACIER project has been funded by the German Federal Ministry of Education and Research under grant no. 16KIS0950.

Heine, F., Kleiner, C., Klostermeyer, P., Ahlers, V., Laue, T., Wellermann, N.:
Detecting attacks in network traffic using normality models: the cellwise estimator.
In *Foundations and Practice of Security. 14th International Symposium, FPS 2021*, pp. 265–282. Springer 2022.

The final authenticated version is available online at https://doi.org/10.1007/978-3-031-08147-7_18

that it should later detect. This means such a model is also not capable to detect novel types of attacks whose patterns do not resemble attacks from the training data.

Furthermore, it is unclear how to deploy such approaches in practice. Experiments from Al-Riyami et al. [3] show that it is in general not possible to use models trained on one network in another network. This implies that deployment must include a training process using data from the target network. However, to train a classifier, this data must include attack samples, which need to be labeled as such. This makes deployment extremely complex, rendering these approaches impractical.

To mitigate this problem, we propose to apply one-class classification to network intrusion detection. One-class classification uses normal-only training data and builds a model of the normal data to later detect deviations from the learned normal behavior. This approach implies that there is no need to include attacks in the training data, making the deployment process much easier.

We compare two different one-class classification methods. On one hand, we use the well-known **one-class support vector machine (OSVM)** algorithm [18], and on the other hand an approach developed at our group called **cellwise estimator (CE)**. The latter is based on modeling traffic patterns of the network on various aggregation levels, e.g. traffic between two hosts, traffic using a certain protocol, traffic between two sub-nets, traffic from a single host using a certain protocol, and so forth. The original idea stems from our paper [9], however in this publication, we introduce an important extension called **default cells** that model how likely new traffic patterns are. This means we focus on attacks that show up as unusual traffic patterns, e.g. DoS attacks, port scans, data exfiltration, lateral movement, C2 communication, see [22]. As we only use traffic meta data and no content, encrypted traffic is no issue.

A main benefit of CE compared to most ML methods including OSVM is **explainability**, which is crucial for practical application, since it helps security operations center (SOC) staff to understand alerts and react better [2]. With typical ML methods, the user has no clue *why* an alert has been issued. We, in contrast, can generate explanations for anomalies, thus moving towards explainable security [24].

Furthermore, we evaluate both methods using benchmark data and compare the results to related results from the literature. The evaluation shows that CE performs better than OSVM on the UNSW benchmark, with the additional benefit of explainability. Overall, we present the following **contributions**:

- Two methods for intrusion detection that use normal-only data for training: the CE and OSVMs
- A comparative evaluation of these methods and discussion of the results
- A comparison to the detection quality of a rule-based IDS (Suricata)

The paper is organized as follows. In Sect. 2, we give an overview of related work in terms of intrusion detection systems using machine learning with a specific eye on anomaly-based approaches. We move on explaining the concept of the CE and the application of OSVMs in Sect. 3. In Sect. 4, we present and

discuss our evaluation results. Finally, we conclude and present an outlook to future work.

2 Related Work

There is a plethora of evaluations of different classification algorithms used for machine learning-based intrusion detection [4,14,20]. For this section, we primarily chose publications that used network data for their evaluation, since this is also the main type of data we focused on using during our own evaluation and which benchmarked their approach using the UNSW-NB15 dataset [15] to improve comparability with our research.

Nixon et al. [17] evaluated the potential of unsupervised autoencoder neural networks with different layer depths as a low-cost alternative for anomaly detection. They used two different approaches to determine the anomaly threshold from data streams and evaluated their approach on the KDD'99 and UNSW-NB15 datasets. For UNSW-NB15 they reported an accuracy of 0.791 and a F1 score of 0.703. As a benchmark, they compared their results to Naïve Bayes and Hoeffding Adaptive Tree approaches, which achieved an accuracy of 0.929 (NB) and a F1 score of 0.832 (HAT) at a much higher computational cost.

Tama et al. [1] proposed a two-stage meta classifier with a hybrid feature selection beforehand and two meta classifiers, i.e., Rotation Forest and Bagging. They evaluated their contribution using the suggested UNSW-NB15$_{\text{test}}$ split from the main dataset and achieved the best results using 19 features, including *service* and *state*. They reported an accuracy value of 0.85797, false positive rate (FPR) of 0.117 and precision/recall of 0.88/0.868 at best.

Tufan et al. [23] created an anomaly-based flow-level IDS pipeline by using an ensemble learning model approach, consisting of two machine learning algorithms, namely a base classifier using Naïve Bayes, a k-nearest neighbors algorithm, logistic regression, and a SVM into a convolutional neural network (CNN), as a case study specifically on probing attack types (e.g., ping sweeping, port scans) and used the reconnaissance attack category from the UNSW-NB15 dataset as a benchmark. They introduced a feature selection workflow and with a smaller set of 10 features, including *dsport*, *state* and *service*, they reported an impressive F1 score of 0.9902 and an area under curve (AUC) value of 0.9990 in their results section for this attack type.

Gharaee et al. [7] used an SVM approach for anomaly detection in combination with a genetic algorithm for feature selection. They evaluated their algorithm on the KDD'99 and UNSW-NB15 datasets for different attack types. In a similar manner Chowdhury et al. [5] combined SVM-based anomaly detection with simulated annealing to select three random features. They evaluated their algorithm on the UNSW-NB15 dataset, achieving an accuracy of 0.9876, a FPR of 0.0009, and a false negative rate of 0.0115. Zhang et al. [26] used a one-class SVM (OSVM) approach, which trains an anomaly detection model with normal data only. They use the KDD'99 dataset for evaluation, making their results not directly comparable to ours.

Khan et al. [11] conducted a study in which five different supervised ML classifiers have been benchmarked using the UNSW-NB15 dataset, namely Decision Tree, a Random Forest classifier, a Gaussian Naïve Bayes classifier, an AdaBoost classifier and a Gradient Boost classifier. They achieved their best results using the Random Forest approach and reported an accuracy of 0.986 and a F1 score of 0.983. Prior to this, a feature extraction was carried out, but it was not elucidated which features were ultimately selected in the process.

3 Concept

In this chapter, we describe both how we use **One-class SVMs** to find attacks, and how the **Cellwise Estimator** works.

We start with a general overview. The basic setting in both approaches is that we use training data that contains only normal (i.e. attack-free) flows from the target network. From this training data, we derive a model that describes the normal operation of the network. Using this model, we score new traffic during the inference phase. The score expresses how well the new traffic matches the normality model. In the evaluation, we figure out how well the scores (i.e. normality of traffic) reflect whether the traffic contains attacks or not.

This is a fundamentally different approach compared to the typical machine learning approach of training a classifier. In this approach, the classifier sees examples for each type of attack during training and tries to identify similar patterns later.

An important assumption underlying this approach is that deviations from the learned normal behavior of the network will, at least to some part, be related to attacks. This implies that the attack-free part of the network data is in some form regular, i.e. that the traffic follows a common pattern, at least to some amount. As such, the method might work even better in industrial networks where machines communicate in a regular fashion compared to office networks where humans communicate.

To train both the OSVM and the CE, *hyperparameters* need to be configured. These parameters influence exact behavior and thus the detection capabilities. To validate hyperparameter settings, labeled validation data is required. In consequence, this means that our requirements are not completely fulfilled. However, this is an intrinsic problem of all anomaly-based approaches. An interesting approach is to adapt the hyperparameters during operation as suggested by [17] for the threshold. The CE has a slight benefit as we obtained already good results with standard settings, while a grid search was necessary for the OSVM to work properly.

3.1 One-class SVM

OSVMs are a special kind of SVMs that are trained on normal-only records. They model an area in the data space where normal records reside. The inference data is scored based on its distance to the normal area in the data space [26].

As SVMs can only handle numeric data, we need to convert all categorical columns of the data to numeric columns. We use one-hot encoding to accomplish this, however, some columns like port numbers have high cardinality leading to high-dimensional data. As a remedy, in the case of ports, we only use the well-known ports and convert the other ports (> 1024) to frequency groups depending on how often the port occurs in the training data.

We also tested Isolation Forests [13] as an alternative, however OSVMs yielded superior results thus we only include OSVMs in the paper due to lack of space.

3.2 Cellwise Estimator

In this section, we describe the cellwise estimator. For more details please refer to its original proposal in [9]. However, the original proposal does not include the concept of default cells, that proved to be very important in the evaluation.

The base idea is to use various aggregation levels in an automated fashion to model network traffic. Data cubes [8] and Online Analytical Processing (OLAP) [12] offer an appropriate model. In this terminology, **dimensions** are categorical attributes that define aggregation levels, like IP address, port, protocol etc., and **metrics** are continuous attributes that are aggregated, e.g. summed up, like a connection count or the sum of the transferred data amount in bytes.

Assume we have four dimensions *srcip, destip, destport*, and *protocol*. Then, a **cell** is described by a four-tuple containing one entry for each dimension: either a specific value for this dimension, or a star $*$ meaning *any*. Thus the cell (1.2.3.4, *, 443, tcp) contains all tcp traffic on port 443 originating from IP 1.2.3.4 to any destination host. The set of all cells is called a **cube**.

The time dimension is handled by chopping the data into time slices of a configurable size, e.g. 20 minutes or one hour. During training, for each cell a time series is collected with one value per time slice. During inference, each time slice is converted to a data cube and evaluated as soon as it is complete, i.e. during operation, new alerts are generated for each time slice independently.

For each cell, we store one or more models that describe the normal traffic pattern in this cell. Here, various classes of models (Gaussian, time series, etc.) are possible in general. A Gaussian model for the cell (1.2.3.4, *, 443, tcp) could for example describe the normal amount of traffic in the cell during a defined time period.

Three problems have to be solved. Firstly, even with a moderate number of dimensions, the number of possible cells is quite large; in general, we have $\prod_{i=1}^{N}(|A_i| + 1)$ possible cells where N is the number of dimensions and $|A_i|$ is the cardinality of dimension i (e.g., the number of different IP addresses). Secondly, many of the more specific cells will have no or only very sporadic data making it difficult to find a robust model. Thirdly, we do not know how to score traffic that does not match an existing model.

To solve these problems, we first define an **iceberg condition** that the training data for a cell must meet for the cell to be included in the final model. In our experiments, we only use cells where traffic occurs regularly, excluding all

sporadic patterns. Assume that host 1.2.3.4 communicates with 5.6.7.8 regularly using tcp port 443 (https). Thus a cell (1.2.3.4, 5.6.7.8, 443, tcp) is built. There is no other host that 1.2.3.4 contacts via 443/tcp regulary, i.e. no other cell (1.2.3.4, *target*, 443, tcp) is build for other target IP addresses. Note that there are efficient algorithms to find all cells fulfilling an iceberg condition [25]. The set of all iceberg cells is called an **iceberg cube**.

Continuing the example, assume that during the inference phase, there is traffic from 1.2.3.4 to 3.4.5.6 via 443/tcp. This traffic would only show up in aggregated cells like (1.2.3.4, *, 443, tcp), where it might not be detected. Thus the question is: is it normal for host 1.2.3.4 to contact other hosts apart of 5.6.7.8 via 443/tcp? To answer this, we introduce **default cells**. In this case the default cell is (1.2.3.4, ?, 443, tcp). During training, all traffic from 1.2.3.4 via 443/tcp that does not match an iceberg cell is collected. Finally, a model for this traffic is build. This means that we can score how unusual the contact to 3.4.5.6 is, depending on whether host 1.2.3.4 has contacted other hosts apart of 5.6.7.8 in the past. Note that no iceberg condition is applied to default cells. Instead, the set of default cells is derived from the iceberg. Thus a default cell can also be an empty cell during training, making new traffic occurring in this cell during inference very suspicious. To summarize, the default cell (1.2.3.4, ?, 443, tcp) contains all data from the cell (1.2.3.4, *, 443, tcp) that is not in a specific cell (1.2.3.4, *ip*, 443, tcp) of the iceberg.

Default cells enable the model to differentiate better between entities that have regular communication patters with only a few targets (typically servers, production systems), and entities that have many spontaneous connections like human-operated office PCs. Furthermore, the combinatorial approach of the cells makes it possible to perform this differentiation on various levels, i.e. to model that the communication patterns of some host on port 443 are very noisy, while the patterns on port 22 (ssh) are quite regular, as the user browses various different web-pages while contacting only a single ssh server. In the consequence, browsing a new web page would not trigger an alert, while contacting a new ssh server would do so.

Overall, the training process runs through the steps of the `train_ce()` function shown in Lst. 1.1. First, all cells in the iceberg cube are computed using a configurable condition. From the iceberg cube cells, the default cells are derived. The union of the iceberg cells and the default cells is the set of all cells that we are going to compute models for. Then, all training records are assigned to each matching cell. Having collected the data for each cell, we are ready to build a statistical model for each cell. The model cube returned only contains the model parameters and no training records any more.

The inference phase is described in the `inference_ce()` function. It starts by assigning the inference data to the cells. Using the statistical models, an anomaly score is calculated for each cell separately that tells us how unusual the data in this cell is. A configurable threshold then determines whether the score is large enough to qualify as an anomaly. All anomaly cells are collected.

Listing 1.1. Training and Inference Procedures of the CE.

```
1  FUNCTION train_ce(train_data, iceberg_cond):
2      iceberg_cube := build_iceberg(train_data, iceberg_cond)
3      default_cube := create_default_cells(iceberg_cells)
4      model_cube := iceberg_cube UNION default_cube
5      assign_records_to_cells(model_cube, train_data)
6      FOR EACH cell IN model_cube:
7          build_models(cell)
8      RETURN model_cube
9
10 FUNCTION inference_ce(model_cube, inference_data, threshold):
11     assign_records_to_cells(model_cube, inference_data)
12     anomalies := list()
13     FOR EACH cell IN model_cube:
14         evaluate_model(cell)
15         IF score(cell) > threshold:
16             anomalies.append(cell)
17     anomaly_groups := group_anomalies(anomalies)
18     RETURN anomaly_groups
```

In a final phase, the anomaly cells are grouped into anomaly groups. This is necessary as an anomaly might be visible in multiple cells due to the hierarchical nature of the cube data. As an example, large amounts of connections towards host 1.2.3.4 on port 443 *might* also show up as anomalies in cells (*, 1.2.3.4, 443, tcp), (*, 1.2.3.4, *, tcp), and (*, 1.2.3.4, *, *). On one hand, this is a kind of redundancy; on the other hand, different cells might give different hints to the analyst. As a solution, we do not remove redundant cells but rather bundle them as **anomaly groups** using the most generic cell as root. In the example, (*, 1.2.3.4, *, *) is the root for the group. For each anomaly in the group, text is generated that describes the cell, the cell model and explains why the cell data is considered unusual. Such texts look like this:

```
Anomaly from a rare source ip to destination port 143.
A rare value for 'source ip' in this context is any value except:
  '59.166.0.0', '59.166.0.1', [...]
The following anomalies were found:
- Connection count (value 39) is not within 0.00 +- 0.01 * 4.
```

This indicates that data has been found in a default cell that has been empty during training. The threshold (configurable) is four times the std. dev. of the Gaussian distribution. For constant values, we set the std. dev. to 0.01 to avoid division by zero errors. A GUI could provide a drill through feature to show the data underlying each cell and each cell model, i.e. as a time series.

Finally, we note that the CE approach is generic: it could also be applied to anomaly detection in other domains as long as there are suitable dimensions in

the data to define aggregation levels. However, in this paper, we focus on the IDS domain.

3.3 Comparison and Output

The One-class SVM approach and the CE are quite different approaches. The base granularity of network data is a flow record (a single connection). An OSVM builds a model that looks at each record individually and assigns it a score. In contrast, we group the records to different aggregation levels and compare the aggregated values to models built during training for the same aggregation levels. This means that the output is on a different scale: individual connections vs. groups of connections. This does not exclude that we cannot issue alerts about individual connections, in cases that there is only a single connection in a cell, e.g. in a default cell that is supposed to be empty (no traffic expected). However, connections that belong together are grouped in a natural way and thus reduce the overall number of alerts. We will show the result of this reduction in the evaluation.

4 Evaluation

In this section we present evaluation results of the CE as well as other detection approaches. We have tested the CE with the following publicly available datasets: UNSW-NB15 ([15]), CICIDS2017 ([19]), TON IoT ([16]). Even though each of these datasets has its own issues, those are the best datasets publicly available to our knowledge. Due to space constraints, we cannot present all evaluation results here, hence we will focus on the most representative examples.

4.1 Effectiveness of CE

In this subsection we will discuss the quality of the attack detection of our CE approach on the mentioned datasets.

Note that the CE has a large set of hyper-parameters, such as time slice size, iceberg conditions, used cell types, cell aggregation function as well as cell model specific parameters, only the most influential of which we can discuss here due to space limitations. Our experiments showed that different iceberg conditions do not have a big impact on the results and will thus not be analyzed further. In addition, when changing the size of the time slices used during training and inference, we did observe changes in detection quality, but no general pattern. Due to this non-uniform behaviour we will use time slices of 20 minutes for the remainder of this work to make the results comparable. Note that this behavior of the hyper-parameters is considered specific for the data sets used in this paper. When applying the CE to other data and/or domains, these parameters are likely relevant. This requires further investigation (cf. section 5.2).

We use the well-known receiver operating characteristic (ROC) curves displaying the behavior of the true positive rate (TPR) of the algorithm as a function of the FPR along with the AUC as quality measure. A random classifier

achieves a straight line from (0,0) to (1,1) and thus an AUC of 0.5 whereas a perfect classifier has an AUC of 1.0.

For the UNSW-NB15 dataset, we used the fields *srcip*, *dstip*, *dsport*, *proto*, *state* in our tests. The CICIDS17 dataset has a lot more numerical features overall, but is missing the *state* field, so we tested the CE on *srcip*, *sport*, *dstip*, *dsport* and *proto*. The network part of the TON IoT dataset was also missing the *state* field while having a *service* field, so we tested the CE on the fields *srcip*, *sport*, *dstip*, *dsport*, *proto* and *service*. We only use a single count metric that indicated the number of flows in the current aggregation. Using this metric, a Gaussian model is build for each cell.

We have focused our feature selection on those features which facilitate User and Entity Behavior Analysis (UEBA) [10, Req. 2] and which are processable by our prototype. This set of features differs from the sets other approaches tend to use, since some of them use automated feature selection for their evaluation and most of their choices do not contain standard flow identifying characteristics like *srcip, sport, dstip* or *dsport*, thereby forfeiting the advantages of UEBA.

For the UNSW-NB15 dataset, we used a cross-validation approach, with data from all but one hour used for training and data from the remaining hour for testing. For this, we chose a subset of the dataset from 2:00 until 12:00 on day 2, as this range contains continuous data without gaps. That means that e.g. results for 3 to 4 have been obtained by using training day 2 from hour 2 to 3 and 4 to 12 and testing on data from hour 3 to 4. According to our approach, attacks are removed from the training part of the data. Fig. 1 shows the corresponding ROC curves as mean of the cross-validation iterations. Note that here the AUC is computed based on flows by adding up scores from all cells containing the flow. This is done in order to be able to compare results to other approaches later, even though the computation works on cells as explained in Sec. 3.

The approach without default cells (left side) results in bad detection quality, as we obtain an AUC of only 0.62. However, the AUC is much higher if we include default cells to detect attacks, making them a key detection feature of the CE. With a mean AUC of 0.98, we can conclude that the CE can detect attacks well using default cells.

The CICIDS17 dataset includes five days of network traffic between a Monday and a Friday, with a different set of simulated attacks on each day, except Monday. Instead, this day serves as a reference with benign traffic only, so we chose to train the model on this day and to test on the other ones without cross-validation. Training and testing happened between 1 am and 1 pm.

Taking a look at overall ROC curves with 292906 attack flows and 2511658 benign flows, we achieved a maximum AUC value of 0.995. However, if we take a closer look onto isolated AUC values for different types of simulated attacks, we can observe rather different values for individual attacks.

The left part of Fig. 2 shows exemplary curves for the attack type of fuzzers in the UNSW dataset. This attack type produces a large number of flows towards a target host or socket (the respective flow counts are also shown in Fig. 2). This graph shows good detection capabilities of the CE with a mean AUC of 0.97.

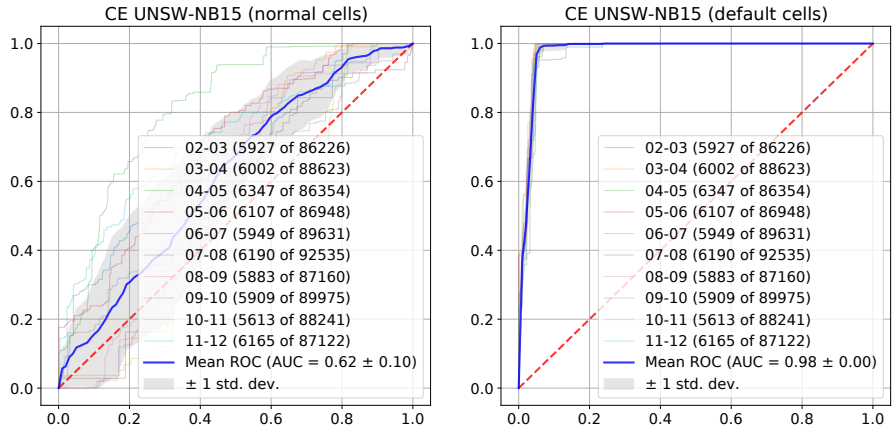

Fig. 1. UNSW overall result for normal and default cells.

For other attack types in this dataset the AUC values are similar, irrespective of the number of flows belonging to the attack. This behavior can not be observed in the results on the CICIDS dataset, shown in the right part of Fig. 2. There we can see that the detection capabilities differ for different types of attacks. In particular the Heartbleed attack consisting of only very few flows has a very low AUC. In general, the CE struggles to detect attacks with small flow numbers as those likely go unnoticed even with default cells. For the UNSW dataset this has not been the case, probably because those few flows have been between previously unseen endpoints, so that they have been detected as abnormal.

Reporting too many anomaly cells could cause resource problems when applying the CE in a SOC. Thus, it would be desirable to report as few anomaly cells as possible while still covering the largest possible number of attacks. To assess how well anomaly cells and attack flows overlap we will use two measures, namely:

- Recall as percentage of attack flows that occur in at least one cell with an anomaly score
- Precision as percentage of the anomaly cells containing an attack

In the left part of Fig. 3, the precision and recall values are displayed for different thresholds. The threshold value determines the anomaly score above which a cell is considered an anomaly. These scores are now computed on cell- not on flow-level in line with the concept of the CE. Naturally, if we raise this threshold we miss some of the attack flows as they are only contained in cells with small anomaly score, reducing the recall. On the other hand, the graph shows that the precision increases with a higher threshold as cells are more likely to contain an attack, if the anomaly score is larger.

Another important measure is *how interesting* a cell containing an attack is, meaning how many of the flows in an anomaly cell are actually attack flows. Cells

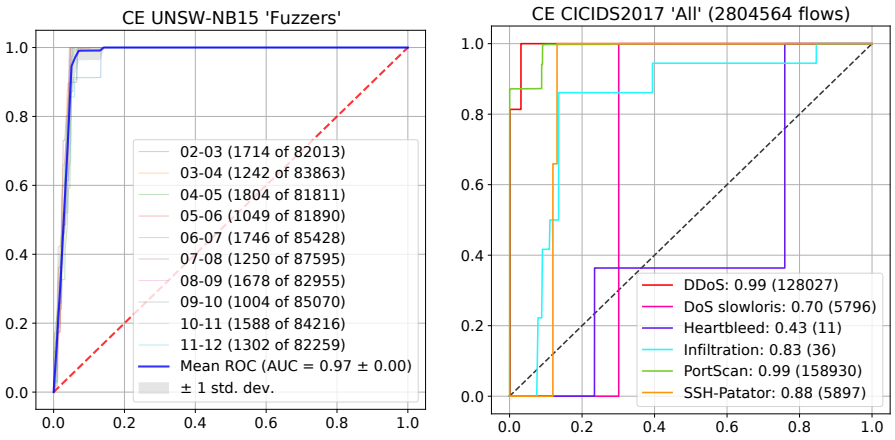

Fig. 2. UNSW results for attack type Fuzzers and CICIDS results for all attack types

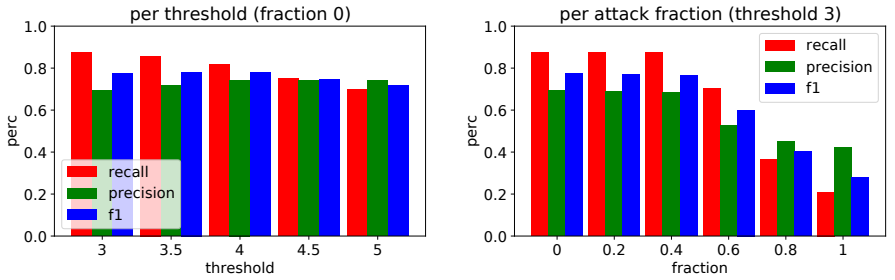

Fig. 3. Detailed Analysis of Anomaly Cell Quality and Importance

with few number of attack flows may lead to huge workload in an SOC while only addressing very few relevant flows. So, ideally the fraction of attack flows in the anomaly cells should be high. As conclusion from above (F1-measure is almost constant for different thresholds), it makes sense to continue with the smallest threshold tested (i.e. 3) focusing on higher recall and analyze the precision and recall values as a function of the fraction of attack flows. The result is shown in the right part of Fig. 3. We can see that up to a fraction of 0.4 the precision, recall and F1 scores are somewhat high, then drop and are rather low (below 0.5) from 0.8 onward. This means that about half of the anomaly cells reported consist of at least 50% attack flows. This is a very good result, as those cells capture attacks very well, so treatment of these cells will be an efficient way to treat these attacks.

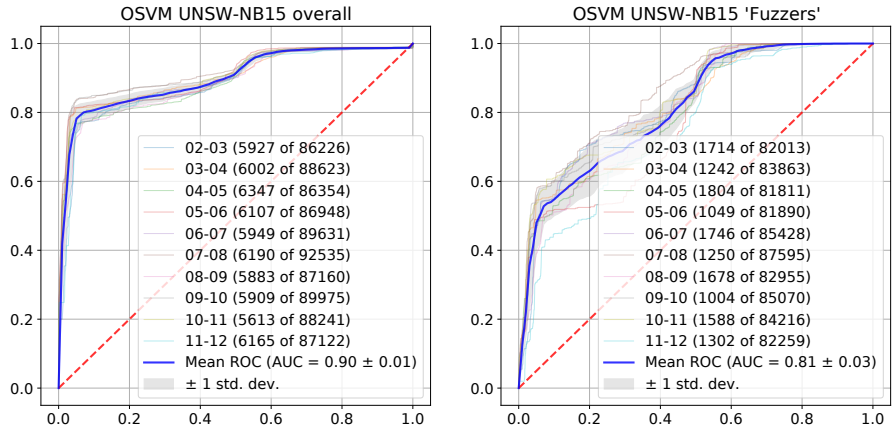

Fig. 4. OSVM results for UNSW using connection meta-data.

4.2 Comparison of CE and OSVM

In this subsection we will compare the results of the CE presented above with the results the OSVM approach (cf. section 3.1). In addition to the actual detection quality assessment, it has to be pointed out again, that the advantage of the CE in general is the explainable model used for the detection, whereas the OSVM acts like a black box.

The OSVM has been provided with more features than the CE, since providing more information about each individual flow is more appropriate for this algorithm. Namely, *sbytes*, *dbytes*, *spkts*, *dpkts*, *srcip* and *dstip* have been provided as numeric attributes and *dsport_cat*, *dsport_fcat*, *sport_cat*, *sport_fcat* as categorical or frequency categorical attributes from the UNSW dataset. The OSVM has been trained on a sample of 100000 flows due to its very high training time, but results have been similar to using the full training set.

The results are displayed in Fig. 4. The left graph showing detection capabilities over all types of attacks should be compared with Fig. 1 (right side) for the CE. We can see that the AUC is higher for the CE than for the OSVM, but both perform well. Due to the different approach (CE grouping flows into cells, whereas OSVM classifies individual flows) the graph of the CE shows a more step-wise behavior for some test sets, whereas the OSVM graph is more continuous. However, the CE reaches the TPR of 1.0 much earlier which is important for attack detection as detecting all attacks is desirable. This should come at the smallest possible false positive rate to avoid costly unnecessary work in the SOC. A positive aspect of the OSVM is the small variance of the detection quality with regard to the actual time slot under investigation, which has only been achieved in the CE by using default cells.

As for the CE we have also examined the behavior of the OSVM over different types of attacks. The right graph in Fig. 4 shows the curve for the fuzzers attack

Method	Overall Alerts		False Alerts	
	μ	σ	μ	σ
CE	336	50	107	53
OSVM	12084	3745	7282	3735

Table 1. Number of Alerts Generated from Detection for Methods (per hour)

type which had slightly lower detection quality than average for the CE (cf. left part of Fig. 2). The graph shows that the sub-par behavior is the same for the OSVM and actually the difference in the detection quality is much larger here than it is overall between CE and OSVM (fuzzers: AUC 0.81 vs. 0.97, overall: 0.90 vs. 0.98). We can conclude that the detection of fuzzers seems to be difficult in general and this is not a specific behavior of the CE. Similarly to the overall evaluation, we see again that the OSVM's curve is more delicate, yet reaches the TPR much later than that of the CE. For this attack type there is also a much larger variation over the different testing intervals than for the OSVM overall. This is comparable to the variation of the CE. In general, the standard deviation is larger for the OSVM, e. g. on the attack type Analysis with very small number of flows per attack the AUC result has been 0.90 with a standard deviation of 0.17. This again emphasizes the higher difficulty of detecting attacks with small numbers of flows.

Another big advantage of the CE can be inferred from Tab. 1, namely the number of generated alerts is significantly lower for the CE than for the OSVM. The total number of alerts is about 3% and the number of false alerts about 1.5% of those in the OSVM. This is very important from a practical perspective as each alert requires the attention of people in the SOC and the less alerts need attention the less personnel in the SOC is required. The difference is also due to the approach: whereas the OSVM generates an alert for each suspicious flow, the CE only generates an alert for a significant number of flows together that are reflected in a single cube cell.

In summary, we can conclude that the CE has a detection quality that is even better than the OSVM, while providing the additional advantage of an explainable detection model as opposed to a black box and in addition reducing the number of alerts requiring attention from the SOC.

4.3 Comparison to other IDS

We have also experimented with the well-known IDS Suricata[1] which is rule-based, i. e. requires detection rules for the attacks and will thus only detect previously known attacks. The detection quality depends significantly on the ruleset fed into Suricata. In order to compare the results to our CE, which does not require rules for known attacks, we have used Suricata on the UNSW-NB15 dataset with the publicly available Proofpoint Emerging Threats open rule set[2].

[1] https://suricata.io/
[2] https://rules.emergingthreats.net/

Severity	Alerts	Recall	Prec	FPR
≤ 1	779	0.01	0.95	0.00
≤ 2	2,211	0.03	0.94	0.00
≤ 3	2,729	0.04	0.94	0.00
Anomalies	38,844	0.03	0.04	0.04

Table 2. Results for Suricata and UNSW-NB15 (day 2)

Suricata rules refer to a variable HOME_NET, which has to be set. For the UNSW-NB15, we used the following subnets: 192.168.0.0/16, 10.40.0.0/16, 59.166.0.0/16, 149.171.126.0/24; cf. the network diagram in [15].

Representative results are shown in Tab. 2. Suricata classifies alerts by severity with 1 being most important, the number of alerts shown is summated. The results show a very high precision of the generated alerts and almost no false positives. Thus, the alerts generated by Suricata are always relevant.

The issue with the generated alerts in Suricata as opposed to detections by CE and OSVM is the extremely low recall. Only an extremely small fraction of potential attacks are detected and the problem is worse the higher the severity is selected. This leads to f1-measures around 0.02 (severity 1) to 0.07 (severity 3) which are significantly lower than the values of the CE shown in Fig. 3.

Suricata additionally detects anomalies which represent unexpected content in packet structure or protocol and might be considered as potential attacks to increase the recall. However, as shown in Tab. 2, there is a very high number of anomalies, but those do not really help in detecting attacks as can be seen from both low precision and recall. Even though the FPR is still low the ratio of undetected attacks is still very high. In summary, Suricata with the given rule-set issues valuable high-precision alerts but misses most of the attacks. Thus it needs to be complemented with an anomaly-based system like the CE.

5 Conclusion

In this section, our results are summarized and and an outlook on future work is given.

5.1 Summary

This paper introduces two novel approaches to detecting attacks in network traffic that operate on a normal behavior model of the network. The advantage of such approaches is that any kind of attack, even if not seen previously, can be detected as long as it has an impact on the traffic in a network which is likely true for almost any relevant attack. The approach to use a one-class support vector machine has shown to provide good detection capabilities, however at the expense of generating a high number of alerts, since it reports every suspicious network flow. Also, it does not provide reasoning as to why this flow is suspicious.

The main contribution of this paper is the second approach using a Cellwise Estimator described in Sect. 3. As has been shown in Sect. 4 the detection capabilities of the CE are even better than the OSVM, achieving an AUC value of 0.98 over all attack types in the UNSW-NB15 dataset. In addition, the CE aggregates suspicious flows into cells, resulting in a significantly lower number of alerts (roughly 1/50 of the OSVM). Moreover, the cells contain information about the reason for flagging them as suspicious, significantly simplifying the work of the security analyst. Finally, the training as well as the analysis time has been significantly lower for the CE when compared to the OSVM (training about 1/30 of the time, analysis 1/3).

The detection quality of both approaches varies over different kinds of attacks, however for all kinds it is much better than that of widely used IDSs such as Suricata. Even though not all hyper-parameters of the CE have yet been completely optimized, the system at the current state is already very powerful.

5.2 Future Work

Even given the promising results presented above, there is a lot of work remaining. As has been explained, the optimization of all hyper-parameters of the CE approach is not yet complete. In particular, we had experiments where the detection quality varied over different sizes of time slices used to create the model. There may be a relation to the actual length of an attack, as an attack spanning over a full time slice might be considered normal behavior. On the other hand large time slices might lead to short attacks staying unnoticed because a cell's anomaly score might remain below the threshold. This relationship has to be analyzed further to be able to optimize this hyper-parameter. Also, using other aggregation functions to compute the scores of a cell beyond the currently used count, as well as cell models of higher complexity than the Gaussian models, is a promising route for future research.

For the used sample datasets as well as in practical usage scenarios feature engineering could be improved in the future. Other research shows that selecting a different set of features for a specific dataset (e.g. [21] for the CICIDS) might also improve the correctness of the prediction scores for other types of attacks. This could be applied here as well. Also, for a practical deployment, general rules on how to determine the relevant features to train the model need to be developed. The need to train the model for a specific network is still there and seems impossible to overcome.

A general issue with the sample datasets available is that the number of attacks is potentially higher than in real-world situations. Thus, a more realistic evaluation scenario would be to add individual attacks to a clean normal behavior dataset and then determine the detection capabilities. However, information about which flow belongs to which attack would be required, which is currently not available for the sample datasets.

Building a model might also benefit from larger training time ranges than have been available in the sample datasets used. The longer the training, the

more precise the model will become, potentially resulting in even better detection. On the other hand, it is important that the data used to learn the normal behavior consists of benign data only. This is more difficult to achieve, the longer the training period is in real circumstances.

Although in this paper we have used default cells only in the classification of network anomalies, we believe that both the method and the approach of using default cells are transferable to other areas of classification as well.

References

1. Adhi Tama, B., Comuzzi, M., Rhee, K.H.: Tse-ids: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. IEEE Access **7** (07 2019), https://doi.org/10.1109/ACCESS.2019.2928048
2. Akinrolabu, O., Agrafiotis, I., Erola, A.: The challenge of detecting sophisticated attacks: Insights from soc analysts. In: Proceedings of the 13th International Conference on Availability, Reliability and Security. ARES 2018, ACM, New York, NY, USA (2018), https://doi.org/10.1145/3230833.3233280
3. Al-Riyami, S., Coenen, F., Lisitsa, A.: A re-evaluation of intrusion detection accuracy: Alternative evaluation strategy. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. p. 2195–2197. CCS '18, ACM, New York, NY, USA (2018), https://doi.org/10.1145/3243734.3278490
4. Buczak, A.L., Guven, E.: A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials **18**(2), 1153–1176 (2016), https://doi.org/10.1109/COMST.2015.2494502
5. Chowdhury, M.N., Ferens, K., Ferens, M.: Network intrusion detection using machine learning. In: Proc. Int. Conf. Security Manag. (SAM). pp. 1–7 (2016)
6. Faraj, O., Megías, D., Ahmad, A.M., Garcia-Alfaro, J.: Taxonomy and challenges in machine learning-based approaches to detect attacks in the internet of things. In: Proceedings of the 15th International Conference on Availability, Reliability and Security. ARES '20, ACM, New York, NY, USA (2020), https://doi.org/10.1145/3407023.3407048
7. Gharaee, H., Hosseinvand, H.: A new feature selection ids based on genetic algorithm and svm. In: 2016 8th International Symposium on Telecommunications (IST). pp. 139–144 (2016), https://doi.org/10.1109/ISTEL.2016.7881798
8. Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., Pirahesh, H.: Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. Data mining and knowledge discovery **1**(1), 29–53 (1997), https://link.springer.com/article/10.1023/A:1009726021843
9. Heine, F.: Outlier Detection in Data Streams Using OLAP Cubes. In: Kirikova, M., Nørvåg, K., Papadopoulos, G.A., Gamper, J., Wrembel, R., Darmont, J., Rizzi, S. (eds.) New Trends in Databases and Information Systems. pp. 29–36. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-67162-8_4
10. Heine, F., Laue, T., Kleiner, C.: On the evaluation and deployment of machine learning approaches for intrusion detection. In: 2020 IEEE International Conference on Big Data (Big Data). pp. 4594–4603 (12 2020), https://doi.org/10.1109/BigData50022.2020.9378479
11. Khan, S., Eswaran, S., Honnavalli, P.: Performance Evaluation of Advanced Machine Learning Algorithms for Network Intrusion Detection System, pp. 51–59. Springer Singapore (01 2020), https://doi.org/10.1007/978-981-15-3020-3_6

12. Kimball, R., Ross, M.: The data warehouse toolkit: the complete guide to dimensional modeling. John Wiley & Sons (2011)
13. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation-based anomaly detection. ACM Trans. Knowl. Discov. Data **6**(1) (mar 2012), https://doi.org/10.1145/2133360.2133363
14. Mishra, P., Varadharajan, V., Tupakula, U., Pilli, E.S.: A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Communications Surveys & Tutorials **21**(1), 686–728 (2019), https://doi.org/10.1109/COMST.2018.2847722
15. Moustafa, N., Slay, J.: UNSW-NB15: A Comprehensive Data set for Network Intrusion Detection systems. In: 2015 Military Communications and Information Systems Conf. pp. 1–6 (Nov 2015), https://doi.org/10.1109/MilCIS.2015.7348942
16. Moustafa, N.: A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. Sustainable Cities and Society **72**, 102994 (2021), https://doi.org/10.1016/j.scs.2021.102994
17. Nixon, C., Sedky, M., Hassan, M.: Autoencoders: A low cost anomaly detection method for computer network data streams. In: Proceedings of the 2020 4th International Conference on Cloud and Big Data Computing. pp. 58–62 (08 2020), https://doi.org/10.1145/3416921.3416937
18. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Comput. **13**(7), 1443–1471 (Jul 2001), https://doi.org/10.1162/089976601750264965
19. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In: ICISSP. pp. 108–116 (2018), https://doi.org/10.5220/0006639801080116
20. Singh, G., Khare, N.: A survey of intrusion detection from the perspective of intrusion datasets and machine learning techniques. International Journal of Computers and Applications **0**(0), 1–11 (2021), https://doi.org/10.1080/1206212X.2021.1885150
21. Singh Panwar, S., Raiwani, Y.P., Singh Panwar, L.: Evaluation of network intrusion detection with features selection and machine learning algorithms on CICIDS-2017 dataset. In: International Conference on Advances in Engineering Science Management & Technology (ICAESMT) (2019), https://doi.org/10.2139/ssrn.3394103
22. Strom, B.E., Applebaum, A., Miller, D.P., Nickels, K.C., Pennington, A.G., Thomas, C.B.: Mitre att&ck®: Design and philosophy (2020), https://attack.mitre.org/docs/ATTACK_Design_and_Philosophy_March_2020.pdf
23. Tufan, E., Tezcan, C., Acarturk, C.: Anomaly-Based Intrusion Detection by Machine Learning: A Case Study on Probing Attacks to an Institutional Network. IEEE Access **9**, 50078–50092 (01 2021), https://doi.org/10.1109/ACCESS.2021.3068961
24. Viganò, L., Magazzeni, D.: Explainable security. In: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS PW). pp. 293–300 (2020), https://doi.org/10.1109/EuroSPW51379.2020.00045
25. Xin, D., Han, J., Li, X., Wah, B.W.: Star-Cubing: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration. In: Freytag, J.C., Lockemann, P., Abiteboul, S., Carey, M., Selinger, P., Heuer, A. (eds.) Proceedings 2003 VLDB Conference, pp. 476–487. Morgan Kaufmann, San Francisco (1 2003)
26. Zhang, M., Xu, B., Gong, J.: An anomaly detection model based on one-class SVM to detect network intrusions. In: 2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN). pp. 102–107 (2015), https://doi.org/10.1109/MSN.2015.40