Better anomaly detection for access attacks using deep bidirectional LSTMs

Henry Clausen[1], Gudmund Grov[2], Marc Sabate[1], and David Aspinall[1,3]

[1] University of Edinburgh, UK {henry.clausen, m.sabate, david.aspinall}@ed.ac.uk
[2] Norwegian Defence Research Establishment (FFI), Norway Gudmund.Grov@ffi.no
[3] The Alan Turing Institute, London UK

Abstract. Recent evaluations show that the current anomaly-based network intrusion detection methods fail to detect remote access attacks reliably [10]. Here, we present a deep bidirectional LSTM approach that is designed specifically to detect such attacks as contextual network anomalies. The model efficiently learns short-term sequential patterns in network flows as conditional event probabilities to identify contextual anomalies. To verify our improvements on current detection rates, we re-implemented and evaluated three state-of-the-art methods in the field. We compared results on an assembly of datasets that provides both representative network access attacks as well as real normal traffic over a long timespan, which we contend is closer to a potential deployment environment than current NIDS benchmark datasets. We show that by building a deep model, we are able to reduce the false positive rate to 0.16% while detecting effectively, which is significantly lower than the operational range of other methods. Furthermore, we reduce overall misclassification by more than 100% from the next best method.

1 Introduction

We present a short-term contextual model of network flows that aims at improving detection rates of remote access attacks. Remote access attacks are used to gain control or access information on remote devices by exploiting vulnerabilities in network services, and are involved in many of today's data breaches [1]. A recent survey [10] showed that these attacks are detected at significantly lower rates than more high-volume probing or DoS attacks. We present the construction of a short-term contextual model of network flows, and show how this model handles suspicious behaviour. Our idea is to capture probability distributions over sequences of network flows that quantify their overall likelihood, much like a language model. We hypothesise that this improves the detection of low-rate access attacks. Our model is based on deep bidirectional LSTM networks.

Recently, deep learning models such as LSTMs have been a popular tool in network intrusion detection [2, 7, 12]. However, persistent failings in evaluations have made it difficult to assess the performance and real-world applicability of currently proposed methods to access attack detection, and have lead to a chaotic and convoluted NIDS landscape [10].

To avoid these pitfalls and demonstrate that our approach delivers a significant improvement in detection rates and real-world applicability, we evaluated our model carefully on two modern network intrusion detection datasets. Furthermore, we reimplemented and evaluated three state-of-the-art methods on these datasets and compared their performance against ours. By carefully selecting input parameters based on their sequential interdependence as well as increasing model complexity in terms of depth and efficient input embedding compared to preceding models, we are able to detect remote access attacks at a false positive rate of 0.16%, a rate at which none of the comparison models are able to detect any attacks reliably.

This work provides the following novel contributions:

1. We present a new and efficient contextual network flow model based on a deep bidirectional LSTM model. It is specifically designed to detect low-volume network access attacks, and significantly improves on current results through selected input parameters as well as increased model depth and efficient input embedding which enables us to detect attacks at a low false positive rate of around 0.16%.
2. We perform a careful evaluation to avoid common failings and demonstrate that our model is capable of both detecting attacks while remaining stable and consistent over time, using two modern datasets.
3. We reimplement and evaluate three prominent anomaly-based intrusion detection models as benchmarks as well as including a smaller and more shallow version of our model. We perform an appropriate, discerning, and comparative evaluation of their performance and conclude that none of these models are able to detect remote access attacks reliably at the false positive rate we achieve.

1.1 Overview

In verbal or written speech, we expect the words "I will arrive by ..." to be followed by a word from a smaller set such as "car" or "bike" or "5pm". Similarly, on an average machine we may expect DNS lookups to be followed by outgoing HTTP/HTTPS connections. These short-term structures in network traffic are a reflection of the computational order of information exchange. Attacks that exploit vulnerabilities in network communication protocols often achieve their target by deviating from the regular computational exchange of a service, which should be reflected in the generated network pattern.

Table 1(a) depicts a flow sequence from an XSS-attack. Initial larger flows are followed by a long sequence of very small flows which are likely generated by the embedded attack script trying to download multiple inaccessible locations. Flows of this size are normally immediately followed by larger flows, as depicted in Fig. 1, which makes the repeated occurrence of small HTTP flows in this sequence very unusual.

Table 1(b) depicts a regluar SMB service sequence while Table 1(c) depicts a *Pass-the-hash* attack via the same SMB service. As shown, the flows to port

Src	Dst	DPort	bytes	# packets
A	B	80	247956	315
A	B	80	7544	13
A	B	80	328	6
A	B	80	2601	10
A	B	80	328	6
A	B	80	328	6
A	B	80	380	7
A	B	80	328	6
			⋮	

(a) XSS-attack, A=192.168.10.50, B= 172.16.0.1

Src	Dst	DPort	bytes	# packets
D	C	N33	600	5
C	D	445	77934	1482
D	C	N33	600	5
C	D	445	5202	10

(b) Benign SMB, C=C6267, D=C754

Src	Dst	DPort	bytes	# packets
C	D	445	4106275	2830
C	D	445	358305611	242847

(c) *Pass-the-hash* attack via SMB

Table 1: The left side depicts a flow sequence from an XSS-attack.The right side depicts a benign SMB-sequence (top), and a sequence from a *Pass-the-hash* attack via the same SMB service.

N33 necessary to trigger the communication on the SMB port are missing while the second flow is significantly larger than any regular SMB flows due to it being misused for exfiltration purposes.

The underlying idea of our model is to predict probabilities of connections in a host's traffic stream conditional on adjacent connections. The probabilities are assigned based on the connection's protocol, network port, direction, and size, and the model is trained to maximise the overall predicted probabilities.

To assign probabilities, we map each connection event to two discrete sets of states, called vocabularies, according to the protocol, the network port, and the direction of the connection for the first, and according to number of transmitted bytes for the second. The size of the vocabulary is chosen large enough to capture meaningful structures without capturing rare events that can deteriorate prediction quality. We then designed a deep bidirectional LSTM (long short-term memory) network that takes bivariate sequences of mapped events as input to efficiently capture the conditional probabilities for each event.

1.2 Outline

The remainder of the paper is organised as follows. Section 2 discusses the current state of network intrusion detection. Section 3 explains the methodology and architecture of our model as well as the data preprocessing. Section 4 explains the advantages of the datasets used in this work as well as current state of the art models that we compare our results with. Section 5 discusses our detection rates on attack traffic, the false positive rate on benign traffic, and compares our results with those of the implemented comparison models. Section 6 concludes our results.

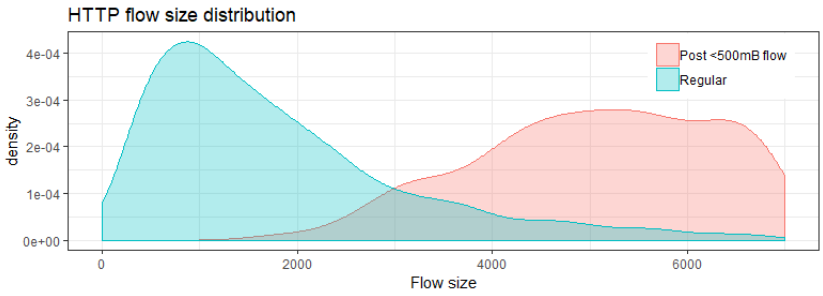


Fig. 1: HTTP flow size distribution overall, and if preceded by an HTTP flow smaller than 500 bytes.

2 Related work and evaluation pitfalls

LSTM-models for web attack detection, such as by Yu et al. [19], improve detection rates of simpler preceding models such as Song et al. [16]. They rely on deep packet inspection, and are often targeted at protecting selected webservers rather than network-wide, due to a lack of computational scalability and increasing traffic encryption. Methodologically, vocabularies are created from string sequences with well-known NLP methods, while our work provides a new vocabulary-construction method suitable for traffic metadata.

The majority of LSTM-based metadata approaches rely on labelled attack data for classification, and do not have the scope of anomaly-based models to detect previously unseen attacks. A prominent example of this comes from Kim et al. [7], who classify flow sequences based on 41 numeric input features. Anomaly-based approaches, such as ours, mostly rely on iterative one-step ahead forecasts, with the forecasting error acting as the anomaly indicator. This is for instance done in GAMPAL by Wakui et al. [18], who use flow data aggregation as numerical input features, which are computationally easier to process, but cannot encapsulate high-level information such as the used protocol, port, or direction. These models are best used for detecting high-volume attacks. Apart from our work, only Radford et al. [12] create event vocabularies from flow protocols and sizes. We use a more sophisticated model in terms of stacked recurrent layers and embeddings for more input features, which results in higher detection rates, as demonstrated in see Sect. 2.1.

Notable work outside of network traffic includes Tiresias [15]and DeepLog [5]. The design of Tiresias has similarities to ours, but the scope of the model is attack forecasting rather than intrusion detection, and relies on both different input data in the form of IDS logs as well as different evaluation metrics.

DeepLog is combined with a novel log parser to create a sequence of symbolic log keys, which is then also modelled using one-step forecasting. The authors achieve good detection results in regulated environments such as Hadoop with limited variety of events (e.g., 29 events in Hadoop). Here, our model goes further

by being applied to a much more heterogeneous data source and creating a more than 30 times larger vocabulary.

2.1 Evaluation pitfalls

According to Nisioti et al. [10], the trustworthiness of published low volume access attack detection rates is debatable due to evaluation shortcomings. We designed our evaluation to avoid four common pitfalls that are regularly seen:

Outdated datasets Two datasets and their derivatives, DARPA-98 and KDD-99, have been extensively used to benchmark network intrusion detection models [11]. However, both datasets are now more than 20 years old and have been pointed out as significantly flawed and prone to give overoptimistic results [17].

Lack of attack class distinction Most intrusion datasets include attack events from both low volume access attack classes such as R2L (Remote-to-Local) and U2R (User-to-Root) as well as attacks like DoS or port scans which generate a large number of events. If reported detection rates do not distinguish between different attacks or attack classes, performance metrics will be dominated and potentially inflated by DoS and probing attacks.

Arbitrary false positive rates There is no agreed upon value for a suitable false positive rate in network intrusion detection. This leads many authors to report very high detection rates at the expense of having unrealistically high false positive rates, often around 5% and above. In our evaluation, we report overall AUC scores, which describe the separation of benign and anomalous traffic.

Lack of long-term evaluation To be effective, an intrusion detection system has to produce a consistently low false positive rate in the presence of concept drift. A crucial aspect when assessing the deployability of an intrusion detection system is the long-term stability of a trained model [8], which is often neglected in the literature. We include a dataset focused on long-term traffic evolution in our evaluation to demonstrate the stability and deployability of our model.

3 Experiment Setting

3.1 Session construction

To order the raw network flows, we first gather all outgoing and incoming flows for each of the hosts selected for examination according to their IP address.

The traffic a host generates is often seen as a series of *session*, which are intervals of time during which the host is engaging in the same, continued, activity [13]. In our context, flows that occur during the same session can be seen

as having strong short-term dependencies. We therefore group flows going from or to the same host to sessions using an established statistical approach [13]:

If a network flow starts less than α seconds after the previous flow for that host, then it belongs to the same session; otherwise a new session is started. If a session exceeds β events, a new session is started.

We chose the number of $\alpha = 8$ seconds as we have found that on average around 90% of flows on a host start less than 8 seconds after the previous flow, a suitable threshold to create cohesive sessions according to Rubin-Delanchy et al. [13]. We introduced the β parameter in order to break up long sessions that potentially contain a small amount of malicious flows, and estimated $\beta = 25$ to be a suitable parameter, detection rates do not seem to be very sensitive to the exact choice of β though.

3.2 Contextual modelling

Each session is now a sequence of flows that are assumed to be interdependent. We observed in an initial traffic analysis that the protocol, port, and direction of a flow as well as its size are highly dependent on the surrounding flows, which motivates their use in the modelling process. We treat flows as symbolic events that can take different states, much like words in a language model. The state of a flow is defined as the tuple consisting the protocol, network port, and the direction of the flow. We consider only the server port numbers, which indicate the used service, in the state-building process. We introduce the following notation:

M:	number of states	N_j:	the length of session j
C:	number of host groups	$x^{i,j}$:	the state of flow i in session j
S:	number of size groups	c^j:	the host group
N^i_{embed}:	embedding dimension	$s^{i,j}$:	size group of flow i in session j
N^i_{hidden}:	LSTM layers dimension		

The collection of all states is called a *vocabulary*. For prediction, the total size of a vocabulary directly correlates with the number of weights needed to be inferred in an LSTM network, thus influencing the time and data volume needed for training. Too large vocabularies also lead to decreased predictive performance by including rare events that are hard to predict [4]. We therefore bound the total number of states and only distinguish between the $M-2$ tuples of protocol, port, and direction most commonly seen on a machine, with less popular combinations being grouped as "Other". Furthermore, the end of a session is treated as an additional artificial event with its own state. The total vocabulary size is then given by M.

Our experimentation has shown that detection rates improve when including the size as an additional variable, especially for brute force web attacks. Rather than making a point estimate of the size, we want to produce a probability distribution for different size intervals. This provides better accuracy for situations in which both small and large flows have a similar occurency likelihood. We group

flows into S different size quantile intervals, with the set of all size intervals forming a third vocabulary.

Hosts are grouped according to their functionality (Windows, Ubuntu, servers, etc.) a distinction that can easily be performed using signals in the traffic. The group is provided to the model as an additional input parameter and forms a third vocabulary.

3.3 Trained architecture

We use a deep bidirectional LSTM network which process a sequence in both forward and reverse direction to predict the state and size group of individual flows. The architecture of the network we trained is depicted in Fig. 2. The increased model complexity we present has not been explored in previous LSTM applications to network intrusion detection, and enables us to boost detection rates while lowering false positive rates, which we demonstrate in Section 2.1.

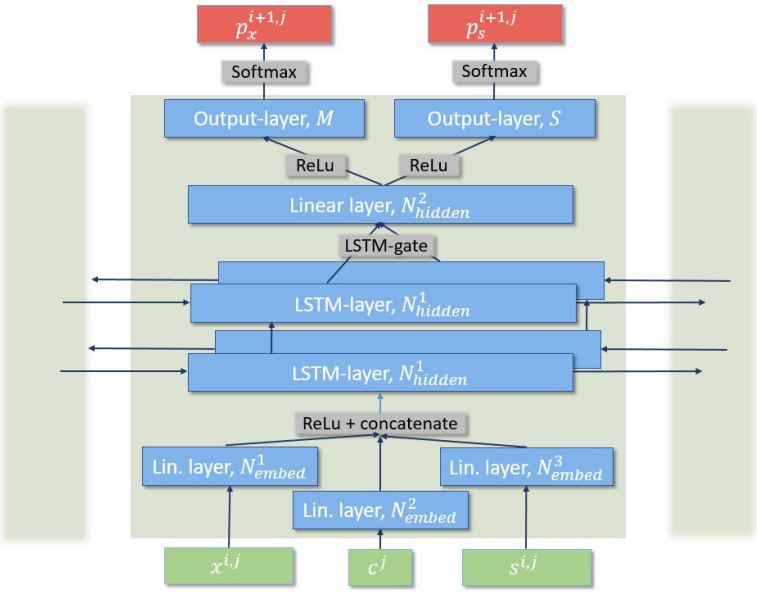


Fig. 2: Architecture of the trained bidirectional LSTM network.

Embedding First, each of the three inputs of the three vectors is fed through an embedding layer, which assigns them a vector of size N^i_{embed}, $i \in \{1, 2, 3\}$. This embedding allows the network to project the data into a space with easier temporal dynamics. This step significantly extends existing designs of LSTM models for anomaly detection and allows us to project multiple input vocabularies simultaneously without a large increase in the model size. By treating the

state, the size group, and the host group as separate dictionaries, we avoid the creation of one large vocabulary of size $M \times C \times S$, which makes training faster and avoids the creation of rare states [4].

LSTM-layer In the second step, the vectors are concatenated and fed to a stacked bidirectional LSTM layer with N^1_{hidden} hidden cells. This layer is responsible for the transport of sequential information in both directions. The usage of bidirectional LSTM layers compared to unidirectional ones significantly improved the prediction of events at the beginning of a session and consequently boosted detection rates within short sessions. Increasing the number of LSTM layers from one to two decreases false positive rates in longer sessions while maintaining similar detection rates.

Output layer The outputs from the bidirectional LSTM layers are then concatenated and fed to an additional linear hidden layer of size N^2_{hidden} with the commonly used rectified linear activation function. We added this layer to enable the network to learn more non-linear dependencies in a sequence. We found that by adding this layer, we are able to capture complex and rare behaviours and decrease false positive rates.

Finally, we use softmax output layers of size M and S, which provide us with two probability vectors, $p_x^{i,j,k}$ and $p_s^{i,j,l}$. The prediction loss for both the state is then given by the log-likelihood;

$$\text{lh}_x^j = \sum_{i=1}^{N_j} \text{lh}_x^{i,j} = \sum_{k=1,i=1}^{M,(N_j)} x_k^{(i),j} \cdot \log(p_x^{i,j,k})$$

with the size group loss being calculated in the same way. We calculate the total loss as the sum of the state loss and the size group loss.

After the training, we use the network to determine the anomaly score of a given input session via the average of the predicted likelihoods, as this measure is independent of the session length:

$$\text{AS}^j = 1 - \sum_{i=1}^{N_j} \Big(\exp(\text{lh}_x^{i,j}) + \exp(\text{lh}_s^{i,j}) \Big) / N_j$$

An anomaly score close to 0 corresponds to a benign session with a very high likelihood while a score close to 1 corresponds to an anomalous session with events which the network would not predict in the context of previous events.

As we mentioned above, too large vocabularies can cause problems both for model training and event prediction. We achieved the best results for $M = 200$ for the available data and computational resources. The size of the size group was chosen smaller with $S = 7$. Host groups were determined for each dataset individually. We trained on a quad-core CPU with 3.2 GHz, 16 GB RAM, a single

NVIDIA Tesla V100 GPU. Training each model could be achieved in under three hours. We chose $N^1_{\text{embed}} = 10$, $N^2_{\text{embed}} = N^3_{\text{embed}} = 5$ for the embedding layers, and $N^1_{\text{hidden}} = N^2_{\text{hidden}} = 50$ for the hidden layers, which achieve the best results without overfitting. We trained each model for 500 epochs. The parameters of the network are optimised to minimise the total loss in minibatches of size 30 using the ADAM optimiser. The optimal value for the learning rate was found to be 0.0003, and decays by a factor of 2 after each ten subsequent epochs the training set. We use a parameter weight decay of 5×10^{-4} per epoch to avoid overfitting, and a drop-out rate of 0.5. Our implementation uses PyTorch.

4 Datasets and benchmark models

4.1 Dataset assembly

We selected two publicly available datasets that complement each other: CICIDS-17 [14], and UGR-16 [9]. The CICIDS-17 dataset contains traffic from a variety of modern attacks, while the UGR-16 dataset's length is suitable for long-term evaluation. We train models with the same hyperparameters on each dataset to demonstrate the capability of our model to detect various attacks and perform well in a realistic environment.

CICIDS-17 [14] This dataset, released by the Canadian Institute for Cybersecurity (CIC), contains 5 days of network traffic collected from 12 computers with attacks that were conducted in a laboratory setting. The attack data of this dataset is one of the most diverse among NID datasets and contains SQL-injections, Heartbleed attacks, brute-forcing, various download infiltrations, and cross-site scripting (XSS) attacks.

UGR-16 dataset [9] Released by the University of Grenada in 2016, this dataset contains real-world network flows from a Spanish ISP. It contains both clients' access to the Internet and traffic from servers hosting a number of services. The data contains a wide variety of real-world traffic patterns. The main advantage of this dataset over previous ones is its usefulness for evaluating IDSs that consider long-term evolution and traffic periodicity, as it spans from March to August of 2016.

4.2 Detection method

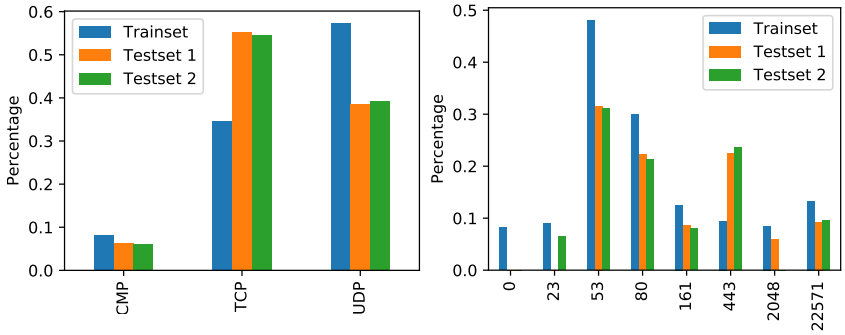
We identify an individual session as malicious if it contains labelled attack traffic from the seven remote access attacks in the CICIDs-17 dataset. We use the the 99.9% anomaly-score quantile as a simple threshold T for our model to distinguish between malicious and benign. By determining T from the training data, we control the expected false positive rate in the test data. Finding an appropriate threshold value is a compromise between higher detection rates and lower false positive rates. Our chosen threshold would translate to about one false alert every three days for host with an activity similar to the CICIDS-17 data, and about one false alert every seven days for hosts in the UGR-16 dataset,

which is about 20-50 times lower than the false-positive rates that our benchmark models were evaluated on. We additionally provide AUC-scores, a performance measure independent of a particular threshold choice (see next Section).

4.3 Dataset split

To evaluate detection rates, we split the CICIDS-17 data into a test set and a training set. We selected the four hosts in the data that are subject to remote access attacks, two web servers and two personal computers. We choose our test set to contain the known attack data while the training data should only contain the benign data. Due to the short timespan of the dataset, we have to train on traffic from all five days, with the test data intervals being placed around the attack. For this reason, we evaluate temporal model consistency only on the UGR-16 data. In total, the test set contains 14 hours of traffic for each host while the training set contains 31 hours of traffic. We chose our training data to contain about 10 000 sessions per host if possible. Overall, we included for the data 24128 sessions in the training and 32414 sessions in the test set for the CICIDS-17, whereas we included 50010 sessions in the training and 100018 sessions in the test set for the UGR-16 data.

To test long-term stability and robustness of our model against concept drift, we split the UGR-16 data into one training set interval and two test set intervals, for which we can compare model performance. The training set interval stretches over the first month, with the first test set interval containing the sessions from the following two months, and the second test set interval containing the last two months. We then isolated traffic from five web-servers that provide a variety of services that show behavioural evolution. Fig. 3 depicts the changes of these servers in terms of protocol and port usage over the different intervals.



(a) Protocols (b) Ports

Fig. 3: Temporal change in protocol and port usage over the different train and test intervals across selected servers in the UGR-16 dataset.

4.4 Benchmark comparison models

We compare our detection and false positive rates against three network anomaly models that we have re-implemented and re-evaluated.

A recent and well cited survey by Nisioti et al. [10] identified the **UNIDS model by Casas et al.** [3] as achieving the highest detection rates of remote access attacks on the KDD-99 dataset, so we chose to include this method as our first benchmark. The authors rely on subspace-projection and density-based clustering (DBSCAN) for outlier detection, and achieve detection rates of access attacks at around $80-85\%$ on the KDD-99 dataset with a false positive rate of 3.5%.

Niyaz et al. [6] present a more recent deep-learning model combines anomaly detection and classification by using sparse autoencoders and detection through reconstruction error. The authors classify individual flows and claim a detection precision of 99% with a recall of 97.5%, even higher than the UNIDS model.

Finally, **Radford et al.** [12] predict sequences of individual flows between pairs of hosts using a two-layer LSTM network. Flows are tokenised according to their protocol and size, and the model detects 60% of the attacks at a false positive rate of about 2% on the CICIDS-2010 dataset. This model is closest to ours in terms of contextual anomaly detection from flow metadata, and achieves the best results out of the three bechmark models during our evaluation. We include it to highlight the improvements our design choices yield over other contextual LSTM-models.

Lastly, we include a more **shallow version of our model**, depicted in Fig. 4, to highlight the benefits of a deeper structure. This model only contains one LSTM-layer, and no linear layer before the output layer.

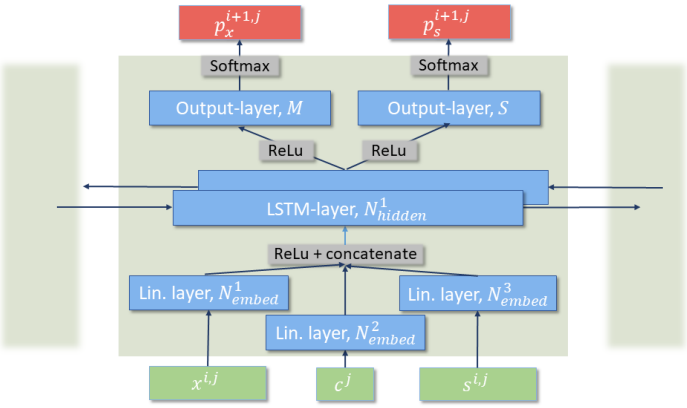


Fig. 4: Architecture of the shallow LSTM-model version we use as a benchmark.

5 Evaluation

We demonstrate that we can build an accurate and close-fitting model of normal behaviour with the model described in Section 3. We train models for each dataset separately, but without any change in the selected hyperparameters, i.e. number of hidden cells, vocabulary size, learning rate etc.

5.1 Detection rates

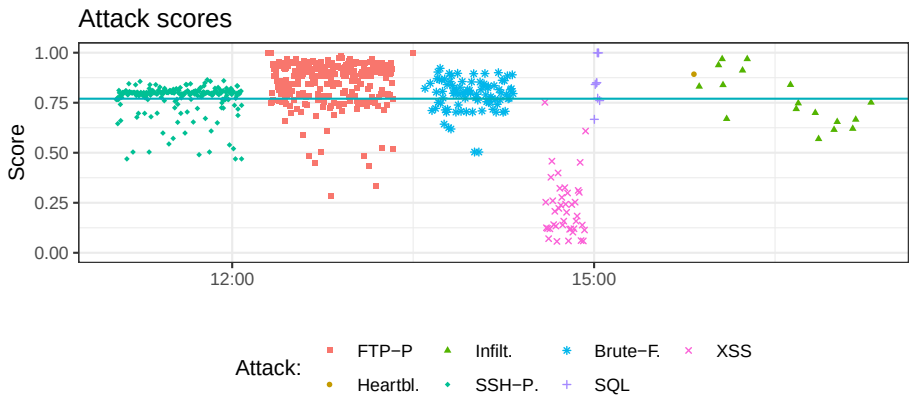


Fig. 5: Score distribution for access attacks contained in the CICIDS-17 dataset.

As described above, we estimate detection rates using traffic of various remote access attacks in the CICIDS-17 dataset. Table 3 and Fig. 5 depict anomaly score distributions and detection rates for traffic from seven different types of attacks.

Most notable is that scores from all attacks except cross-site scripting (XSS) are significantly higher distributed than benign traffic, with median scores lying between 0.75 and 0.89. Detection rates with our chosen threshold of 0.77 are highest for Heartbleed attacks (100%), followed by FTP and SSH brute-force attacks and SQL-injections, where 91%, 74%, and 75% of all affected sessions are detected. Detection rates are lowest for XSS and Infiltration attacks. The overall detection rates we achieve are in a similar range as most unsupervised methods in Nisioti et al.'s evaluation [10], but with significantly better false positive rates.

XSS and infiltration attacks cause the victim to execute malicious code locally. Heartbleed and SQL injections on the other hand exploit vulnerabilities in the communication protocol to gain exfiltrate information, and are thus more likely to exhibit unusual traffic patterns, visible as completely isolated TCP-80 flows for SQL attacks, or unusual sequences of connections initiated by the attacked server during Heartbleed attacks.

	scores and det. rates				comparison det. rates			
	min	max	median	det. rate	UNIDS	Radford	Niyaz	shallow m.
Brute Force Web	0.50	0.92	0.80	0.66	0.0	0.0	0.07	0.28
FTP-Patator	0.28	1.00	0.82	0.91	0.07	0.0	0.03	0.38
Heartbleed	0.89	0.89	0.89	1.00	0.0	0.0	0.0	0.0
Infiltration	0.57	0.97	0.75	0.41	0.0	0.166	0.0	0.0
SQL-injection	0.67	1.00	0.84	0.75	0.0	0.0	0.02	0.21
SSH-Patator	0.47	0.86	0.80	0.74	0.0	0.0	0.0	0.67
XSS	0.06	0.75	0.20	0.00	0.0	0.0	0.20	0.0

Table 2: Anomaly score distributions and detection rates for malicious sessions in the CICIDS-17 dataset, as well as detection rates for comparison models on the right. Depicted are the minimum, maximum, and median score for sessions in each attack class along with the rate of sessions exceeding the detection threshold (99.9% quantile).

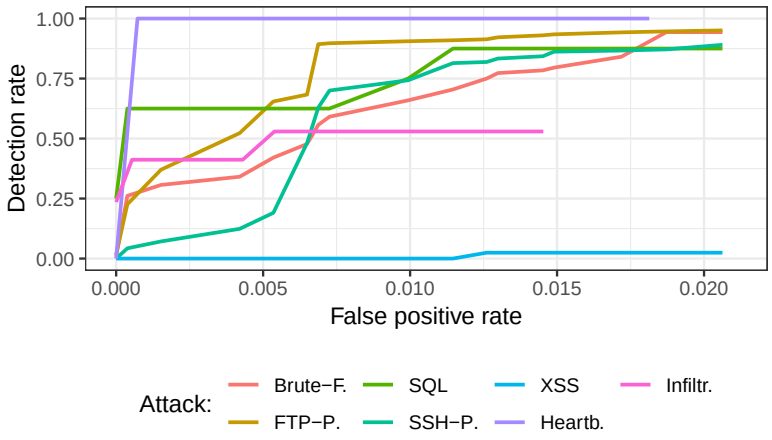


Fig. 6: ROC-curves for different attack types in the CICIDS-17 dataset.

Brute-force attacks on the other hand cause longer sequences of incoming connections to the same port of a server, in this case to port 21 for FTP, 22 for SSH, and 80 for web brute-force. Especially for port 80, such sequences are not necessarily unusual, which explains the difference in detection rates between web brute-force, which our model does not detect reliably, and FTP and SSH brute-force, which are detected at a higher rate. Depending on how much benign traffic the particular sessions are overlayed, the estimated anomaly scores can vary. Brute-Force attacks are not low in volume, and spread over many sessions since we introduced a maximum session length. For these types of attack, our model therefore only has to flag a smaller percentage of malicious sessions the attack generates to detect anomalous behaviour.

Fig. 6 provides *ROC* (Receiver operating characteristic) curves for each attack type. As seen, for Heartbleed, FTP brute-force, SQL injection, and infiltration attacks, our model starts detecting attacks with close to zero false positives.

Comparison models We now compare detection rates between our model and the three models described in Section 4.4 that we chose as benchmarks.

All three models ultimately detect anomalies when an anomaly score exceeds a threshold, which controls the balance between low false positive rates and high detection rates and usually depends on the given data. To create a fair comparison, we chose threshold providing similar false positive rates of 0.01%, e.g. the 99.9% anomaly score quantile of the training data, which is necessary for assessing suitability for real-world deployment as we have argued in Section 4.2. Table 3 depicts detection rates for each model. As seen, none of the other models are achieving any meaningful detection rates at a false positive rate this low.

	1-AUC scores				
	Our model	UNIDS	Radford	Niyaz	shallow m.
Brute Force Web	**0.016**	0.49	0.027	0.32	0.048
FTP-Patator	**0.0025**	0.011	0.0048	0.16	0.0052
Heartbleed	**0.0003**	0.0057	0.032	0.077	0.012
Infiltration	0.046	**0.033**	0.35	0.15	0.11
SQL-injection	**0.005**	0.44	0.497	0.39	0.019
SSH-Patator	**0.009**	0.013	0.035	0.011	0.005
XSS	0.127	**0.02**	0.03	0.16	0.13
Average	**0.044**	0.144	0.135	0.18	0.091

Table 3: 1-AUC scores for our model and the implemented comparison models on the CICIDS-17 dataset. Fat numbers indicate the best value for each attack.

To assess the overall separation between benign and malicious traffic for each model, we calculated $1-AUC$ *(Area under ROC curve)* scores by varying the thresholds for each model, and calculating the area under the ROC-curve, depicted in Fig. 6 and Table 3.

In comparison to the other benchmark models, our shallow model is capable of making some detection at the chosen false positive rate, but cannot reach the levels of our deeper model. While brute-force attacks are still detected, more nimble attacks such as Heartbleed or XSS are less distinctive from benign traffic for the shallow model. It is remarkable that by adding the described additional layers, we are able to more than double the overall detection power, as indicated by the 1-AUC-scores.

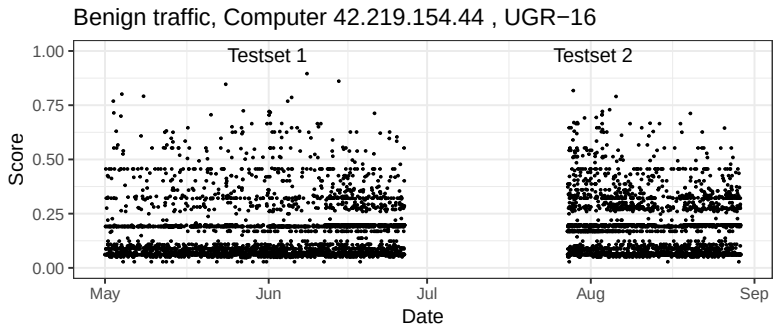


Fig. 7: Anomaly score distribution for benign sessions in the UGR-16 dataset

5.2 Benign traffic and long-term stability

We assess the performance of our model on benign traffic by looking at the quantiles and visual distribution of sessions scores. The plots and tables in Table 4 as well as Fig. 7 depict the score distribution of benign sessions for each dataset in the corresponding test sets.

	50%q	90%q	99.9%q	max score	Pr(>T)
UGR-16, testset 1	0.14	0.33	0.76	0.9	0.0018
UGR-16, testset 2	0.17	0.37	0.72	0.86	0.0014

Table 4: Anomaly score quantiles for benign sessions in each testset in the UGR-16 data, along with the maximum session score and the percentage of sessions exceeding the detection threshold (false alerts), averaged over all hosts.

As shown, the centres of the distributions are concentrated very well in the lower region of the $[0, 1]$ interval, with about 50% of all sessions receiving scores in the region between 0.1 and 0.25. High scores are rare, with only very small percentages exceeding our chosen detection threshold of T. If we look at the hosts in the UGR-16 data, on average less than 0.15% of all assumed-benign sessions exceed the threshold, which would translate to fewer than ten false-alerts over the span of four months on a host with similar activity rates.

A clear banding structure is visible in the plotted distributions, with most session scores being very concentrated on narrow intervals. These scores represent frequently reoccurring activities that generate similar traffic sequences. Fig. 7 shows that these banding structures remain virtually unchanged over several months. Similarly, Score distributions remain stable over several months, as depicted in Table 4. Although more investigations are required for definitive conclusions, these results indicate that the identified contextual structures in network traffic remain relatively stable over time.

6 Conclusion

Our proposed model presents a new and promising angle to anomaly-based intrusion detection that significantly improves detection rates on the types of network attacks with the lowest detection rates. We use an anomaly-based approach that does not rely on specific notions of attack behaviours, and is therefore better suited at detecting unknown attacks rather than regular misuse- or signature-based systems. By assigning contextual probabilities to network events, our model improves detection rates of low-volume remote access attacks and outperforms current state-of-the-art anomaly-based models in the detection of several attacks while retaining significantly lower false positive rates. Furthermore, our model retains low false positive rates for periods stretching several months. Our results provide good evidence that using contextual anomaly detection may in the future help decrease the threat of previously unseen vulnerabilities and malware aimed at acquiring unauthorised access on a host. We specifically focused on short-term anomalies as they are often an unavoidable byproduct of an attack thus very difficult for an attacker to avoid without preexisting control over the victim device or other network devices.

6.1 Resilience and evasion

Evasion tactics and corresponding model resilience against them have been a concern in the development of NIDS. We specifically focused on short-term sequential anomalies as they are often an unavoidable byproduct of attack sequences, and it is thus very difficult for an attacker to pertube attack sequences that rely on a specific exploit without preexisting control over the victim device or other network devices. We therefore believe that our model is relatively robust against evasion, however we identified potential improvements for future work.

A specific evasion tactic that has been discussed extensively in the context of machine learning is model poisoning in the training/retraining phase. As our model uses symbolic features instead of numerical ones, there is little possibility to introduce gradual shifts, and attempts to introduce new sequences would likely exceed the anomaly threshold. Additionally, without control over the victim device, the attacker can only introduce alterations in one direction. Lastly, we showed in Section 5.2 that short-term contextual traffic patterns remain stable over several months, which means that retraining of our model is only necessary at a low rate and attackers will have to wait for a long time to execute successful model poisoning.

An issue we encountered is the overlay of malicious and benign traffic. Currently, the existence of known traffic patterns in a session can deplete the overall anomaly score of a session. A potential evasion tactic could therefore conceal an attack behind benign communication with the victim device, an already common approach for C&C communication. Possible improvements for this issue are a refined notion of a session that groups related traffic better, and a better scoring method that identifies smaller anomalous sequences in an otherwise normal sequence of flows.

Acknowledgements

We are grateful for our ongoing collaboration with our industry partners (blinded) on this topic area, who provided both support and guidance to this work. Discussions with them have helped reinforce the need for a better evaluation and understanding of the possibilities that new intelligent tools can provide.

Full funding sources are currently blinded.

References

1. M-trends 2015: A view from the front lines. Tech. rep. (2015), https://www2.fireeye.com/rs/fireye/images/rpt-m-trends-2015.pdf
2. Bontemps, L., Cao, V.L., McDermott, J., Le-Khac, N.A.: Collective anomaly detection based on long short-term memory recurrent neural networks. In: Future Data and Security Engineering. pp. 141–152. Springer (2016)
3. Casas, P., Mazel, J., Owezarski, P.: Unsupervised network intrusion detection systems: Detecting the unknown without knowledge. Computer Communications **35**(7), 772–783 (2012)
4. Chen, W., Grangier, D., Auli, M.: Strategies for training large vocabulary neural language models. arXiv preprint arXiv:1512.04906 (2015)
5. Du, M., Li, F., Zheng, G., Srikumar, V.: Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1285–1298. ACM (2017)
6. Javaid, A., Niyaz, Q., Sun, W., Alam, M.: A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS). pp. 21–26. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) (2016)
7. Kim, J., Kim, J., Thu, H.L.T., Kim, H.: Long short term memory recurrent neural network classifier for intrusion detection. In: 2016 International Conference on Platform Technology and Service (PlatCon). pp. 1–5. IEEE (2016)
8. Koutrouki, E.: Mitigating concept drift in data mining applications for intrusion detection systems. arXiv preprint arXiv:1010.4784 (2018)
9. Maciá-Fernández, G., Camacho, J., Magán-Carrión, R., García-Teodoro, P., Therón, R.: UGR '16: A new dataset for the evaluation of cyclostationarity-based network IDSs. Computers & Security **73**, 411–424 (2018)
10. Nisioti, A., Mylonas, A., Yoo, P.D., Katos, V.: From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods. IEEE Communications Surveys & Tutorials **20**(4), 3369–3388 (2018)
11. Özgür, A., Erdem, H.: A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015. PeerJ Preprints **4**, e1954v1 (2016)
12. Radford, B.J., Apolonio, L.M., Trias, A.J., Simpson, J.A.: Network traffic anomaly detection using recurrent neural networks. arXiv preprint arXiv:1803.10769 (2018)
13. Rubin-Delanchy, P., Lawson, D.J., Turcotte, M.J., Heard, N., Adams, N.M.: Three statistical approaches to sessionizing network flow data. In: 2014 IEEE Joint Intelligence and Security Informatics Conference. pp. 244–247. IEEE (2014)

14. 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)
15. Shen, Y., Mariconti, E., Vervier, P.A., Stringhini, G.: Tiresias: Predicting security events through deep learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 592–605. ACM (2018)
16. Song, Y., Keromytis, A.D., Swtolfo, S.: Spectrogram: A mixture-of-markov-chains model for anomaly detection in web traffic (2009)
17. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications. pp. 1–6. IEEE (2009)
18. Wakui, T., Kondo, T., Teraoka, F.: Gampal: Anomaly detection for internet backbone traffic by flow prediction with lstm-rnn. In: International Conference on Machine Learning for Networking. pp. 196–211. Springer (2019)
19. Yu, Y., Liu, G., Yan, H., Li, H., Guan, H.: Attention-based bi-lstm model for anomalous http traffic detection. In: 2018 15th International Conference on Service Systems and Service Management (ICSSSM). pp. 1–6. IEEE (2018)