VAE-MAD-GAN FOR HIDS

TIM KAELBLE

ScaDS-AI
November 2020 –

ABSTRACT

Short summary of the contents in English. . . a great guide by Kent Beck how to write good abstracts can be found here:

https://plg.uwaterloo.ca/~migod/research/beckOOPSLA.html

ZUSAMMENFASSUNG

Kurze Zusammenfassung des Inhaltes in deutscher Sprache. . .

CONTENTS

1 EXPOSEE

1.1 EINLEITUNG

Notes :

- use advances of sequence detection from NLP

Angriffe auf Computersysteme bestehen schon seit diversen Jahren. Die häufig verwendeten auf Blacklists basierenden Abwehrmechanismen reichen nicht aus um viele drohende Gefahren abzuwenden. Das liegt hauptsächlich daran, dass weder Abwandlungen von Angriffen, noch unbekannte Angriffe erkannt werden können. Ein wesentlicher Vorteil liefert hier die Angriffserkennung über Anomalien. Im Gegensatz zu dem erwähnten Blacklist Ansatz, muss nicht jeder Angriff der abgewehrt werden soll bekannt sein. Stattdessen wird versucht das Normalverhalten eines Systems zu ermitteln und jegliche Abweichung als Anomalie einzustufen. Nun bieten verschiedene Systeme verschiedene charakteristische Merkmale um das Verhalten zu beschreiben. Eine häufig verwendete Information für die Charakterisierung bieten zum Beispiel System-Logs [3].

In dieser Arbeit werden System-Calls verwendet. Sie bieten eine sehr abstrakte Betrachtung auf Betriebssystemebene. Programme auf einer Festplatte können meist erst Schaden anrichten, sobald sie ausgeführt werden. Dabei führen sie betriebssystemspezifische System-Calls aus, welche über verschiedene Tools wie zum Beispiel Sysdig [11] ausgelesen werden können. Die Schwierigkeit im Vergleich zu dem Untersuchen der Logs besteht darin, die großen Datenmengen zu bewältigen, welche schon bei kleineren Anwendungen anfallen. Die Probleme in der Verarbeitung von sehr großen Datenmengen konnten unter anderem durch die Verwendung selbst lernender Algorithmen erfolgreich angegangen werden. Im realen Einsatz solcher Verteidigungsmechanismen besteht eine weitere Schwierigkeit darin, dass das IDS Zugriff auf den Kernel des zu überwachenden Systems benötigt. Diese wird in dieser Arbeit allerdings nicht behandelt, da lediglich die Algorithmen selbst, jedoch nicht die praktische Umsetzung in einem potentiellen Betrieb betrachtet wird.

In verschiedenen Arbeiten wurden bereits die Abfolge von System-Calls betrachtet, doch nur in wenigen Arbeiten werden auch die Parameter zur Anomalieerkennung verwendet. Eine der ersten Arbeiten von Forrest et. al [1] betrachtet lediglich die Sequenzen der System-Calls. Maggi et al. verwenden zusätzlich auch Parameter und verweisen in ihrer Arbeit [8] auf diverse verschieden Ansätze. In dieser

Arbeit soll versucht werden die Hinzunahme eines Parameters, wie zum Beispiel den Dateipfad (sofern vorhanden) bei schreibenden und lesenden Befehlen, mit Hinblick auf die Erkennungsquote des IDS zu untersuchen.

Nachdem definiert wurde welche Information untersucht wird, stellt sich zu Beginn der Entwicklung einer Anomalieerkennung die Frage, wie das Normalverhalten der Systeme erfasst werden soll. Abstrakt betrachtet werden bei der Untersuchung von System-Calls zeitvariante und potentiell multivariate Datenstreams betrachtet. Besonders erfolgreich haben sich dabei Long-Short-Term-Memory (LSTM) Netzwerke gezeigt. Sie haben den Vorteil auch Zusammenhänge mit größerer zeitlicher Verzögerung noch zu erkennen [5] und können in unterschiedlichsten Architekturen einen Nutzen bringen.

In dieser Arbeit sollen zwei Forschungsfragen verfolgt werden.

- Ist die Zunahme von Parametern bei der Anomalieerkennung mittels System-Calls eine Verbesserung?

  Welche Parameter kommen in Frage?

- Kann der Erfolg von LSTM-Netzwerken auf die Erkennung von Anomalien in der Cyber-Sicherheit übertragen werden?

Des Weiteren wird, je nach Erfolg der ersteren Fragen noch optional folgendes untersucht:

- Können aktuelle Verbesserungen des Lernverhaltens durch GAN auch hier Anwendung finden?
  - MAD-GAN [7]
  - VAE-MAD-GAN [9]

Um diese Forschungsfragen angemessen behandeln zu können müssen zunächst Grundlagen aus verschiedenen Bereichen gelegt werden. Zum einen werden unterschiedliche Herangehensweisen zur Überwachung von Systemen betrachtet und erläutert wieso es für diese Anwendung sinnvoll ist eine Host-Based Intrusion Detection zu wählen. Zum anderen müssen die Grundlagen für den verwendeten Algorithmus gelegt werden. Dazu gehören Grundlagen zu neuronalen Netzen sowie die Erweiterungen der LSTM Netzwerke.

Ein großer Teil der Implementierungsarbeit wird die Vorverarbeitung der Daten darstellen. Diese soll mit der genaueren Untersuchung des gesamten Algorithmus in einem weiteren Kapitel dargestellt werden. Nachdem die verwendete Software analysiert wurde, wird eine Auswertung auf dem LID-DS [6] Datensatz durchgeführt. Dieser bietet den Vorteil, dass in einer reproduzierbaren Art System Calls aufgenommen wurden. Des Weiteren werden zusätzlich die System Call Parameter zur Verfügung gestellt.

Im letzten Teil der Arbeit soll dann eine Schlussfolgerung aus den zuvor gewonnenen Ergebnissen gezogen werden. Hauptsächlich sollen die gestellten Forschungsfragen untersucht werden. Konnte

mit einem hinzugezogenen Parameter ein Mehrwert erzielt werden? Bieten sich LSTM-Netzwerke auch für die Anomalieerkennung im IT-Sicherheitsbereich an?

2 VERWANDTE ARBEITEN

2.1 ANOMALIEDETEKTION MIT SYSTEM CALLS

2.2 UNTERSUCHEN VON ZEITREIHEN MIT LSTM

3 GRUNDLAGEN

3.1 ANOMALIE DETEKTION

3.1.1 *Ansätze*

- Whitelist
- Blacklist

  aufzählen aller Angriffe
- Normalverhaltern

  Definiere was normal ist

  Analogie menschliches Immunsystem → Auch NN Analogie zur Natur <++>

3.1.2 *Host Based Intrusion Detection*

3.1.3 *System Calls*

3.2 LONG SHORT-TERM MEMORY RECURRENT NEURAL NETWORK

Neuronale Netze bestehen aus verschiedenen Ebenen von Knoten (auch Neuronen genannt), die miteinander verbunden sind. Die Neuronen erhalten Eingangssignale und falls diese eine bestimmte Bedingung erfüllen (z.B. Schwellwertüberschreitung) wird das Ausgangssignal des Neurons verändert. Wird die Bedingung nicht erfüllt, sendet das Neuron ein Signal, welches die nachfolgenden Neuronen nicht beeinflusst. Der schematische Aufbau wird in Abbildung 3.1 gezeigt.

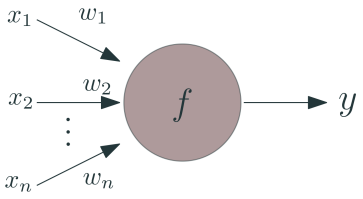

Figure 3.1: Aufbau eines Neurons

Da ein einzelnes Neuron noch keine komplexen Aufgaben lösen kann, ist der Aufbau in Netzen entscheidend. Die Neuronen sind in sogenannten *Layers* (z dt. Ebenen) angeordnet (siehe Abbildung 3.2). Es gibt eine *Input-Layer*, quasi beliebig viele *Hidden-Layers* und

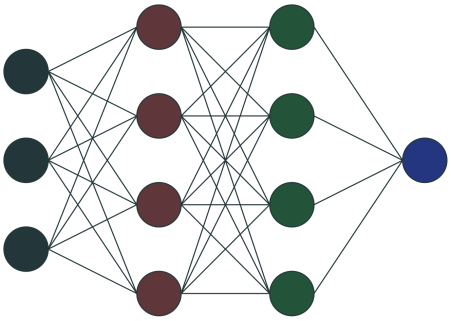

Figure 3.2: Schematische Darstellung eines neuronalen Netzes

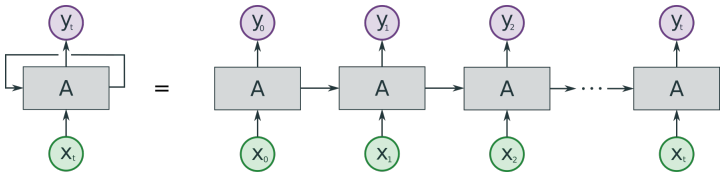

Figure 3.3: „Ausgerollte“ Darstellung eines RNN Neurons (inspiriert von [10])

eine *Output-Layer*. Die einzelnen Knoten sind jeweils mit der nachfolgenden Layer verbunden. Ist der Ausgang eines Knotens auf einer Layer mit einer vorherigen oder derselben Layer verbunden, spricht man von einem *Feedback* oder *Recurrent Neural Network* (RNN) (siehe Abbildung 3.3), sonst von einem *Feedforward Neural Network* (FNN). Mit Knoten, die eine extra Verbindung zu sich selbst haben, können frühere Eingaben Einfluss auf die Behandlung der nächsten Eingabe haben. Der einzelne Knoten merkt sich seine Ausgabe, welche im nächsten Zeitschritt als weiteres Eingabesignal dient. Dadurch wird es ermöglicht auch zeitlich abhängige Sequenzen zu erlernen, da die Behandlung eben auch vorherige Geschehnisse mit einbezieht. Lernt ein FNN zum Beispiel die Eingabe (A,B,A,B,A,B,A,...), besteht eine 50% Chance, dass die nächste Ausgabe ein B ist, sofern die letzte nicht bekannt ist. Weiß man, dass die letzte Ausgabe ein B war, können wir davon ausgehen, dass eigentlich ein A folgen sollte. Die Feedback Struktur RNNs ermöglicht also einen ersten Ansatz, um zeitlich abhängige Muster zu erkennen.

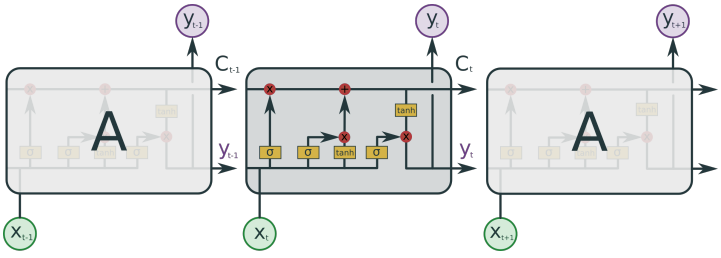

Figure 3.4: Schematische Darstellung eines Knotens in einem LSTM NN, mit Input, Output und Forget Gate (inspiriert von [10]).

Long Short-Term Memory (LSTM) ist eine Erweiterung der RNNs. Hauptziel der LSTMs ist es, das Lernen der zeitlich abhängigen Muster zu verbessern. Auch RNNs haben dieses Ziel, doch mit diesen Netzwerken kann schon ein Abstand von 10 diskreten Zeitschritten zwischen den abhängigen Ereignissen nicht überbrückt werden [4]. So kann ein RNN zum Beispiel im Satz „Die Wolken am *Himmel*“ das Wort *Himmel* vorhersagen, doch bei der Satzfolge „Die Person kommt aus Frankreich. ... . Die Person spricht *französisch*.“ wird es Schwierigkeiten haben. Dies soll die LSTM Zelle erweitern und zudem durch verbesserte Fehlerkorrektur für bessere Lernergebnisse sorgen. Umgesetzt wird diese Anforderung, indem an jedem Knoten eine *Memory Cell* (Gedächtniszelle) angebracht wird. Sie ist mit sich selbst verbunden und gibt den Zellstatus an. Mit Hilfe dieser Information soll eine Abhängigkeit auch über einen längeren Zeitraum gefunden werden. Der Zellstatus C_{t-1} zum Zeitpunkt $t-1$ hat dann im nächsten Zeitschritt t einen Einfluss auf die Zelle C_t und somit auch auf die Ausgabe y_t. Die Weitergabe des Status wird in Abbildung 3.5 dargestellt.

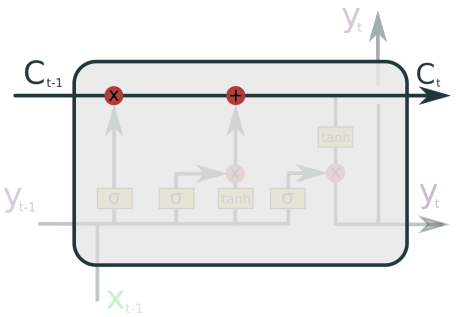

Figure 3.5: Weitergabe des Zellstatus innerhalb eines Knotens (inspiriert von [10]).

Einfluss auf den Zellstatus haben zwei verschiedene *Gates* (zu dt. Gatter/Tore). Im ersten Schritt wird entschieden, welche Information aus dem vorherigen Zeitschritt keinen Einfluss mehr auf den Zellstatus haben sollen. Dies wird mit dem *Forget Gate* umgesetzt und ist in Abbildung 3.6 zu sehen. Informationen aus dem Speicher, die keinen Einfluss mehr haben sollen, können so entfernt werden. In dem Sprachbeispiel könnte das Genus (*grammatikalisches Geschlecht*) gespeichert werden, um so eine grammatikalisch korrekte Vorhersage zu machen. Kommt nun allerdings ein neues Pronomen, sollte das bisher gespeicherte Genus keinen Einfluss mehr haben.

Das *Input Gate* soll im nächsten Schritt angeben, welche neuen Informationen in den Zellstatus C_t aufgenommen werden. Dies erfolgt in zwei Schritten, zunächst wird mit i_t ermittelt, welche Information geupdated werden soll. Im Vektor $\tilde{C}$ ist der eigentliche Werte (wie z.B. das Genus) enthalten, welcher den zuvor vergessenen Wert ersetzen soll (vgl. Abbildung 3.7) .

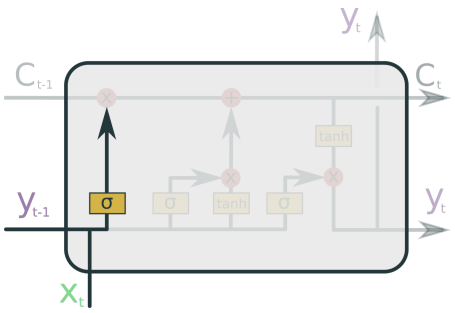

Figure 3.6: Einfluss des Forget Gates auf den Zellstatus (inspiriert von [10]).

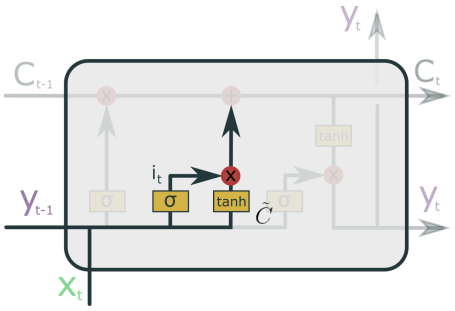

Figure 3.7: Einfluss des Input Gates auf den Zellstatus (inspiriert von [10]).

Wie der Zellstatus C_t nun die Ausgabe beeinflusst, wird über das *Output Gate* geregelt (siehe Abbildung 3.8). Dies soll in unserem Sprachbeispiel entscheiden, ob die Information des Genus für die Vorhersage des nächsten Wortes eine Rolle spielt. [2] [10]

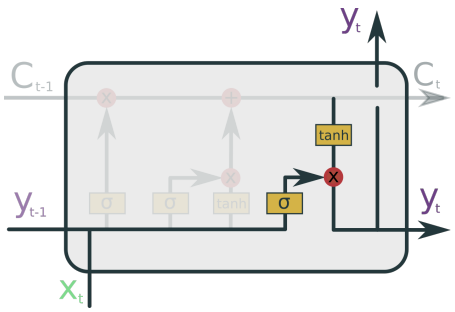

Figure 3.8: Das Output Gate regelt den Einfluss des Zellstatus auf die Ausgabe des Neurons (inspiriert von [10]).

Die verschiedenen Gates können so als ein weiteres kleines NN in jedem Knoten der LSTM Netze betrachtet werden, welche einen zeitlichen Zusammenhang besser erkennen sollen.

4 REALISIERUNG

5 ERGEBNISSE

5.1

6

FOLGERUNGEN

BIBLIOGRAPHY

[1] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. "A sense of self for Unix processes." In: *Proceedings 1996 IEEE Symposium on Security and Privacy*. 1996, pp. 120–128. DOI: 10.1109/SECPRI.1996.502675.

[2] Felix A. Gers, Jürgen A. Schmidhuber, and Fred A. Cummins. "Learning to Forget: Continual Prediction with LSTM." In: *Neural Comput.* 12.10 (Oct. 2000), pp. 2451–2471. ISSN: 0899-7667. DOI: 10.1162/089976600300015015. URL: http://dx.doi.org/10.1162/089976600-300015015.

[3] S. He, J. Zhu, P. He, and M. R. Lyu. "Experience Report: System Log Analysis for Anomaly Detection." In: *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. 2016, pp. 207–218. DOI: 10.1109/ISSRE.2016.21.

[4] Sepp Hochreiter. "Untersuchungen zu dynamischen neuronalen Netzen." In: (Apr. 1991).

[5] Sepp Hochreiter and Jürgen Schmidhuber. "LSTM Can Solve Hard Long Time Lag Problems." In: *Proceedings of the 9th International Conference on Neural Information Processing Systems*. NIPS'96. MIT Press, 1996.

[6] *Leipzig Intrusion Detection Data Set*. URL: https://www.exploids.de/lid-ds/.

[7] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. "MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks." In: *Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series*. Ed. by Igor V. Tetko, Věra Kůrková, Pavel Karpov, and Fabian Theis. Cham: Springer International Publishing, 2019, pp. 703–716. ISBN: 978-3-030-30490-4.

[8] F. Maggi, M. Matteucci, and S. Zanero. "Detecting Intrusions through System Call Sequence and Argument Analysis." In: *IEEE Transactions on Dependable and Secure Computing* 7.4 (2010), pp. 381–395. DOI: 10.1109/TDSC.2008.69.

[9] Zijian Niu, Ke Yu, and Xiaofei Wu. "LSTM-Based VAE-GAN for Time-Series Anomaly Detection." In: *Sensors* 20.13 (2020), p. 3738. ISSN: 1424-8220. DOI: 10.3390/s20133738. URL: http://dx.doi.org/10.3390/s20133738.

[10] Christopher Olah. *Understanding LSTM Networks*. Aug. 2015. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[11] *Seeing is Securing For containers, Kubernetes and cloud services*. URL: https://sysdig.com/.