A Multi-agent Case-based Reasoning Intrusion Detection System Prototype

Jakob Michael Schoenborn[1,2] and Klaus-Dieter Althoff[1,2]

[1] University of Hildesheim, Universitaetsplatz 1, 31141 Hildesheim, Germany
schoenb@uni-hildesheim.de,
[2] German Research Center for Artificial Intelligence (DFKI), Trippstadter Str. 122, 67663 Kaiserslautern, Germany

Abstract. The number of actors, costs, and incidents in terms of internet criminality is rising each year as many devices in our daily routines become increasingly connected to the internet. 'Security by design' is gaining increased awareness in software engineering, but it is not to be expected to catch all security issues as the range of potential security issues and the creativity of the attackers are both seemingly endless. Thus, we propose a multi-agent case-based reasoning system to detect malicious traffic in a computer network. We mainly rely on the commonly used UNSW_NB15 data set including 82332 training cases with mostly numeric attributes, but the application design is open to operate with other data sources, such as NSL-KDD and CICIDS-2017 as well.
Purpose. The aim of the proposed system is to detect malicious network traffic and alert the security engineer of a company to take further actions such as blocking the source IP address of the potential attacker.
Findings. We were able to successfully detect seven out of ten attacks with an average true-positive rate of 82,56 % and leave the remaining attacks (Analysis, Backdoor, Worms) for further investigation and improvements.
Implications and value. The results are close to other research results with room for improvement. Due to the nature of a multi-agent framework, this application could be integrated into other existing intrusion detection systems and serve as an add-on.

Keywords: Case-based Reasoning, SEASALT, Intrusion Detection System, Multi-Agent System

1 Introduction

Supply chains and production processes become increasingly more digital and connected with online services through industry 4.0 and a higher rate of globalization. Additionally, the pandemic forced organizations further into digitisation by communicating to their employees located in their newly established home offices. Through the increasing necessary network connections, the dependence on a faultless and uninterrupted network is also increasing as well. The recent world events showed the negative influence for multiple countries as soon as the production and shipping of goods is interrupted. This can be seen as an analogy for computer network traffic: if important computers are not reachable in the network, if important services cannot be reached, the production rate of a company decreases - in the worst case, the company has to shut down its production entirely. It is not only the loss of income through the not produced goods, which negatively affects the company, but also further indirect consequences such as damage of reputation and legal consequences by not fulfilling service level agreements (SLA)[3] [18] or, in case of a data leak, general data protection regulation (GDPR) or California privacy rights act (CPRA). For Europe, the European Union Agency for Cybersecurity publishes annually the NIS[4] Investments report. According to this report, the top components of direct incident costs are *incident response costs* (33 %), *costs related to data recovery and business continuity management* (22 %) and *loss of productivity* (19 %) [7].

Worldwide, the investment in cyber threat intelligence is annually increasing by 16 % [7] with the top three investments by solution type are *cloud access security brokers* (33 %), *vulnerability assessment* (25 %) and *web application firewalls* (25 %) [6], which matches the needs in a stable network environment as mentioned above and where our approach can be supportive. Nevertheless, medium-sized companies and startups are financially not able to invest in security or do not have a dedicated ransomware defense program, e. g., 44 % of the companies in the health sector [7]. The lack of security professionals and the lack of established certified processes, e. g. ISO27001, enables potential attackers to enter and attack the computer network of a company. It is not uncommon for companies to have multiple thousands of known vulnerability issues - even for larger IT companies. Thus, a network intrusion detection system to detect potential attacks can be helpful in the mitigation of an attack and consequently to reduce the effective costs.

For the support on intrusion detection for any interested individual or company, we develop a case-based reasoning (CBR) intrusion detection system (IDS). CBR is a methodology cycling through four steps: retrieve, reuse, revise, retain. Generally, CBR follows the paradigm "Similar problems have similar solutions", thus retrieving experience from old situations (cases) to solve a new occurring problem. For any given data package, we *retrieve* the most similar data package (case) and might *reuse* its solution (label) to flag an incoming data package as potential attack. If certain attributes are missing or the usage is not immediately possible, we might *revise* the case and query the system again. Based on the results, a knowledge engineer might decide to *retain* the new case into the case base. Another possible consequence of the *retain* step might also only be to adjust the similarity measures, which is one of the knowledge containers for the knowledge representation, along the casebase, adaptation knowledge, and vocabulary according to Richter [11]. As there are different kinds of attacks and different kinds of (training-) data sets, we propose a multi-agent system accord-

[3] Def. SLA: *An explicit statement of expectations and obligations that exist in a business relationship between two organizations: the service provider and customer.* [**?**]

[4] Network and Information Systems Directive

ing to the SEASALT architecture [2] to ensure the scalability of the system. Using this way of modularization allows us to initialize and query CBR agents whenever they are needed and adjust resources accordingly to the amount of incoming data packages (potential attacks). Using only a single classifier was not feasible due to limited computational resources triggered by a large casebase[5].

In the following sections, we begin to take a closer look at related work and similar approaches in the literature. This will provide us insight in which data sets are most commonly used and how other approaches fare in terms of detecting malicious traffic. Section 3 describes the concept of our application, including the reasoning behind the knowledge modeling and case representation. Consequently, we provide a brief overview on the application itself in Section 4 and continue with an evaluation of our approach in contrast to other approaches in Section 5. The paper closes with a conclusion and an outlook into future work.

2 Literature review

The aim of the literature review is to identify categories of AI methods and to find common data sets, which are used to measure the detection rate of malicious traffic. As database, we chose IEEExplore[6], arXiv[7], and Google Scholar[8]. We filtered for articles, which have been published in conferences using peer reviews as review criteria[9]. The time frame ranges from 2015 to 2022; from 2014 backwards, the number of relevant literature rapidly decreases. After additional filtering, we identified 206 relevant articles.

2.1 Data sets

In terms of data sets, we could identify the following usage:

Four ‘main’ data sets can be identified: KDD-CUP-99, NSL-KDD, UNSW-NB15 and CIC-IDS-2017. The former both were predominantly existent in the early literature, also before 2015, with both data sets releasing in 1999. NSL-KDD has been steadily updated and seems to be the most used data set up until today, while KDD-CUP-99 lost its popularity. 2018 was a year where multiple new datasets have been created[10], but seemingly have not been used furthermore. Instead, the latter two, UNSW-NB15 and CIC-IDS-2017 are newer data sets, rising in popularity and are becoming possible alternatives. Figure 1 presents an overview of the popularity of the mentioned data sets in the relevant literature since 2015.

[5] It has to be tested, whether approaches to optimize the retrieval on large casebases will ease this limitation; see also Section 5.1: Limitations.

[6] https://ieeexplore.ieee.org/Xplore/home.jsp

[7] https://arxiv.org/

[8] https://scholar.google.com/

[9] Assumed to exist, if not explicitly mentioned.

[10] But their share was 10 % or lower, thus, not depicted in this graphic.

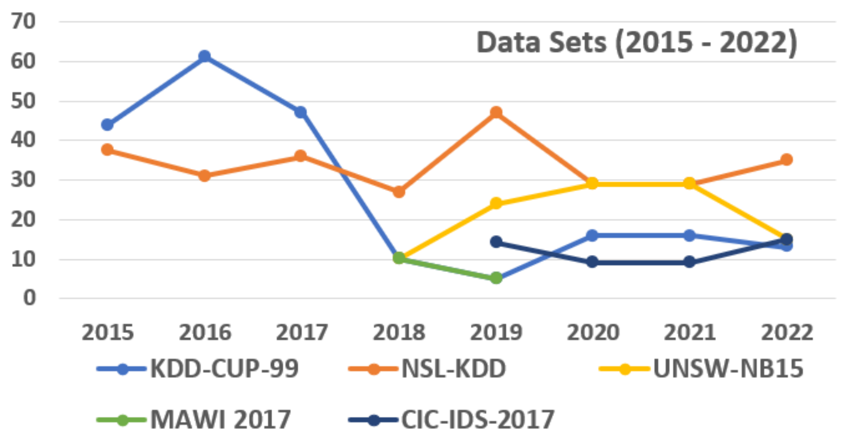


Fig. 1. Used datasets and their share in the relevant literature since 2015. x-axis: year of publication; y-axis: number of publications using the corresponding data set.

2.2 Classifiers

In terms of classifiers, we could identify the following usage:

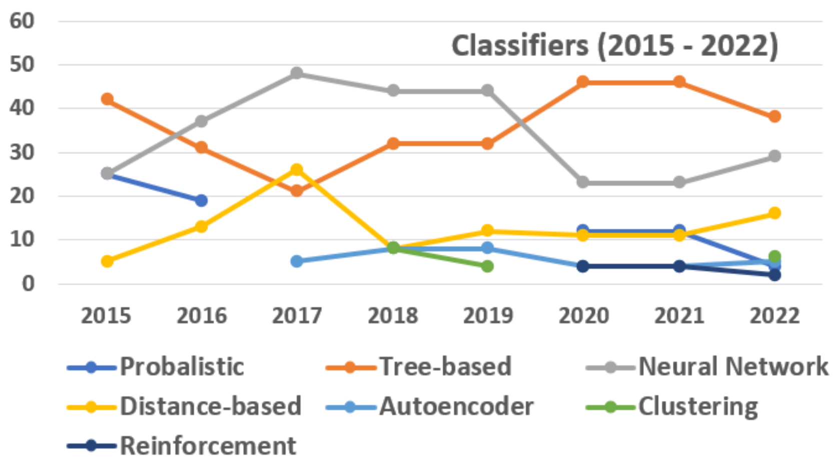


Fig. 2. Used classifiers and their share in the relevant literature since 2015. x-axis: year of publication; y-axis: number of publications using the corresponding classifier.

Unsurprisingly, as the most approaches are based on machine learning algorithms, neural networks (mostly with feature-selection techniques) and tree-based classifiers (mostly random forests to prevent overfitting) are the most dominant classifiers since 2015 up until today. Distance-based models (mostly support vector machines) also achieved reasonable results, but overall worse than its contenders.

2.3 Related Work

In this section, we briefly present related work, which is using the data set UNSW_NB15[11]. To our best knowledge, our approach is the first in the CBR domain, thus, referring to related approaches using the same dataset.

Ullah and Mahmoud focus on identifying malicious traffic in IoT networks. The authors propose a two-level hybrid anomalous activity system [15]. The first level distinguishes traffic between 'normal' and 'anomalous' using flow-based features extracted from the CICIDS2017 and UNSW-NB15 dataset. If an anomaly activity is detected, the flow is forwarded to the level-2 model to find the category of the anomaly, using recursive feature elimination, synthetic minority over-sampling technique, edited nearest neighbours for cleaning the aforementioned datasets and random forest classifier for the level-2 model [15]. Their results propose a 97 % F1 score with both respectively 97 % precision and recall across all attacks. A validation using a real-world scenario to validate the model is promised, but seemingly never published[12].

Anwer et al. present '*A Framework for Efficient Network Anomaly Intrusion Detection with Features Selection*', applying different strategies by using filter and wrapper feature selection methodologies [1]. For classification, J48 and Naïve Bayes algorithms are used. By trying to find the lowest required number of attributes, the authors achieve an accuracy of 80 % across all attacks using 18 features. Future work suggests using support vector machines, artificial neural networks and a majority voting scheme between all classifiers to increase the accuracy.

Wu and Guo propose '*LuNet: A Deep Neural Network for Network Intrusion Detection*' [19]. The authors focus on decreasing the false-positive rate of the system, motivated by the efficiency of the system overall, which often is not accounted for in publications with high detection rates, according to the authors. LuNet is highly focused on convolutional and recurrent neural networks - these learn input traffic data in sync with a gradually increasing granularity such that both spatial and temporal features of the data can be effectively extracted [19]. The results are separated in eight different algorithms, reaching at best an accuracy of 82.78 % with a FPR of 4.72 %.

3 Concept

3.1 Dataset

The concept of the prototype relies to some extend on the chosen dataset. While we propose a multi-agent approach, each agent needs to establish its casebase and cases. For the agents that will be evaluated in section 5, we chose the UNSW-NB15 dataset based on the recommendations by Divekar et al. (2018): "*In summary the results strongly indicate that UNSW-NB15 can satisfactorily substitute*

[11] See Section 3 for the reasoning of this choice.

[12] The authors continued their work in IDS in IoT, but generated new (flow-based) data set.

the archaic KDD CUP 99 dataset and even NSL-KDD when used to train machine learning anomaly-based NIDSs." [5] and by Ring et al. (2019): "*Further, we'd like to give a general recommendation for the use of the [...] CICIDS 2017 and UNSW-NB15 data sets. [...] CICIDS 2017 and UNSW-NB15 contain a wide range of attack scenarios.*" [12]. Both authors published an extensive literature review on the current datasets; their recommendation seems to confirm the trend shown in Figure 1. As KDD-CUP-99 and NSL-KDD both are based on 42 attributes, we are optimistic for our approach to work similar with those data sets - but the confirmation remains open for future work.

UNSW_NB15 has been created by N. Moustafa and J. Slay and spans over 47 different attributes, which can be sub-categorized into basic features, connection features, content features, time features, additional generated features, and labeled features. The attack categories are labeled as 1, while normal traffic is labeled as 0 [9]. Table 1 provides a brief description of the attack categories, so that the interested reader is able to gain a picture of the attacks we are trying to prevent from happening.

The dataset is split into training data (82332 packages, thus, in sum 82332 cases) and testing data (175341 packages). For a detailed description of the 47 features, we refer to the original publication by Moustafa and Slay [9].

Analysis	a type of variety intrusions that penetrate the web applications via ports, emails, and web scripts.
Backdoor	a technique of bypassing a stealthy normal authentication, securing unauthorized remote access to a device.
DoS	intrusion which disrupts the computer resources, to be extremely busy in order to prevent the authorized requests from accessing a device.
Exploit	a sequence of instructions that takes advantage of a vulnerability to be caused by an unintentional behavior on a host or network.
Fuzzers	attacker attempts to discover security loopholes in a network by feeding it with massive inputting of random data to make it crash.
Generic	technique that establishes against every block-cipher to collision without respect to the configuration of the block-cipher.
Reconnaissance	can be defined as a probe; an attack that gathers information about a computer network to evade its security controls.
Shellcode	an attack in which the attacker penetrates a slight piece of code starting from a shell to control the compromised machine.
Worms	an attack whereby the attacker replicates itself in order to spread on other computers. Often, it uses a computer network to spread itself.

Table 1. Description of the attack categories by Moustafa and Slay [9]

.

3.2 Case-based reasoning agents

In terms of efficiency and scalability, we suggest a multi-agent system with at least one agent per attack category and one agent for normal data traffic.

We begin with two agents on top (the first layer): A BurpAgent and WiresharkAgent. Those agents are trained and fed with data of their respective commonly used programs "Burp" and "Wireshark". Both tools are commonly used in the domain: the former, for intercepting and manually manipulating traffic but also capturing traffic in general, the latter mainly for capturing and filtering traffic. Wireshark is known for creating "packet capture"-files (.pcap), which are the foundation of the training- and test datasets of all discussed data sets. Those .pcap file are usually translated into .csv files. These files can be imported by the WiresharkAgent, which then can flag each packet with "normal" or "suspicious" traffic. Suspicious traffic can be forwarded to the next layer to further identify the incoming packet. This leaves us for the UNSW-NB15 data set with ten agents - which can be multi-threaded, if the amount of incoming traffic makes it necessary. The agents can easily be incorporated into multi-agent frameworks such as the SEASALT architecture [2]. The structure is also depicted in Figure 4 in Section 4.

We[13] use case-based reasoning agents, each containing four knowledge containers according to Richter [11]: *vocabulary, similarity measure, adaptation knowledge, casebase.*

In terms of *vocabulary* structure, we use an attribute-value representation, as the measurable data contains 35 attributes in addition to twelve derived attributes. No set of attributes contains unknown values; thus only complete situations are evaluated. Correlations between certain attributes could not be detected, yet. Certainly, attributes contain correlation to attack categories, which will be covered next in the *similarity measure* container.

Following the weighted Hamming similarity measure

$$sim(q,p) = \sum (g_i \times sim_i(q_i, p_i) \mid 1 \leq i \leq n) \tag{1}$$

as Richter and others suggested, we utilize the local-global principle [3,11,16]. For local measures sim_i, we inspect the attributes A_i based on their minimum and maximum values and calibrate a symmetrical polynomial function with heavily decreasing similarity for differing attributes based on the variability of an attribute. The narrower the data points of an attribute, the stronger decreasing the similarity function. For the amalgamation function, we set values for the non-negative real weight vector coefficients $g = (g_1, ..., g_n)$, normalized to $\sum g_i = 1$ [11].

For the values of g, we calculate the average value of each attribute ranging over the whole data set and also the average value filtered by each attack category. This enables us to identify attributes which seem to hint at a certain attack

[13] The remainder of this section is mostly similar to Schoenborn et al. [13]. The overall structure described below has not drastically changed since. However, we slightly changed the used similarity metrics and weights for our agents, leaving us with a different implementation and overall better results.

for given values. For example, *spkts* (=‘*source packets*’) depicts the source-to-destination packet count with the following calculated average values for each attack category, reading: “For an exploit attack, 37,7 packets have been sent on average from the source to the destination”:

AVG	Analysis	Backd.	DoS	Exploit	Fuzzer	Generic	Normal	Recon	Shellc.	Worm
18,67	3,12	4,39	*28,9*	**37,7**	11,8	2,8	6,97	6,97	6,07	*16,78*

While the average on the whole data set is at 18,67 and has lower values for other attacks, Exploit points out with an average value of 37,7 packets from source to destination. This confirms the intuitive expectation of an exploit attack: *to exploit* means in the IT-security context to systematically abuse known security issues of a given system. However, it needs to be tested, which security issues the target might have - resulting into multiple requests and consequently an increased amount of packets running from source-to-destination. Therefore, *spkts* receives a higher weight than other attributes for the exploit agent. The more distinct an attribute-value, the higher the weight. Figure 3 further illustrates the construction.

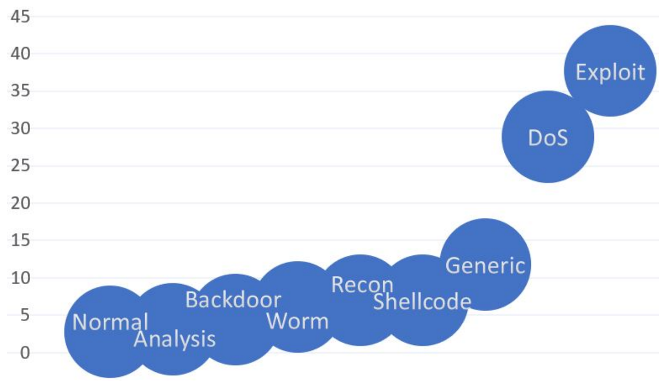


Fig. 3. Example for weight selection for the attribute *spkts*. *y*-axis depicts the average values calculated for the attribute *spkts*. Exploits and DoS attacks can mostly be easily classified, e. g., if the value is > 35, it is most likely an exploit. Thus, exploit- and DoS agents receive a higher weight for the attribute *spkts*. If the value is inside the interval [23,32], it is most likely a DoS attack. Most other values < 15 cannot easily be assigned to an attack category, thus, *spkts* receiving a lower weight for the respective agents.

On a similar notion, this situation also holds true for the denial of service (DoS) attack: with a value of 28,9, it is also distinct enough from other attack categories, which range on average between 2,8 and 16,78. Therefore, we are also able to identify attribute values, which are not the maximum, but still unique to a certain attack category - and use this information to increase the weight of the given attribute for the corresponding agent (see Figure 3). We repeat this process for each agent and each attribute.

Each agent is trained to detect its respective attack, i. e., a DoS agent only contains cases labeled with denial of service attacks. Thus, the *case base* contains the experiences based on the training data set. We store each line of the data set as a case, resulting in 82332 cases overall. However, there is still room left for improvement regarding the two conflicting goals: having the case base as large as possible for increased competence knowledge, while having the case base as small as possible for better efficiency, relative to the available resources.

For each package in the testing data set, each agent votes by submitting its n most similar cases to a coordination agent. For now, it will be left open for discussion in Section 5 whether n should be 1 to submit only the most similar case or to calculate the average similarity of $n > 1$ cases to reduce the risk of outliers. For our experiments, we choose $n = 10$ to remove outliers and gain insight whether the similarity of other similar cases is decreasing correctly, as to be expected. The votes with the highest similarities will be reported to the (human) user. After receiving the results, the user may decide which agent is ultimately correct - leaving the responsibility and legal liability to the human user - and might choose to start further actions to stop the attack, such as blocking the source IP address of the potential attacker.

4 Implementation

We implemented the system described in Section 3 by using myCBR 3.4 and the programming language JAVA. MyCBR is an open-source similarity-based retrieval tool and software development kit (SDK)[14] and has been further developed by students of the University of Hildesheim and by the authors, hence the increased version number. MyCBR 3.4 and the prototype presented in this contribution are available for free under the LGPL licence at Github[15].

Figure 1 provides a brief overview of the multi-agent system. We would like to emphasize that each agent underlies the SEASALT architecture by Bach [2]. First, the user will be provided with a simple graphical web interface, asking to import either a Burp-Export file, a Wireshark .csv Export, or the UNSW-NB15 training- or test data set. In either cases, the Coordination Agent will forward the data to the corresponding agents. However, in the first three cases, the agent will print average statistics on the data visibly to the IDE console or log file. This is especially important for the knowledge engineer to view and control the training data.

In case of importing training data, each agent will be initialized with the given training data, corresponding to its agent type. Each topic agent extents the abstract class *Agent*, which forces each agent to implement methods relevant for any CBR functionality, such as initializing a myCBR project file (.prj) and initializing the four knowledge containers.

For String (text) values, such as protocol, service, and state, we use either the Levenshtein similarity function, or check for equality - depending on the

[14] see http://mycbr-project.org/index.html
[15] see https://github.com/jmschoenborn

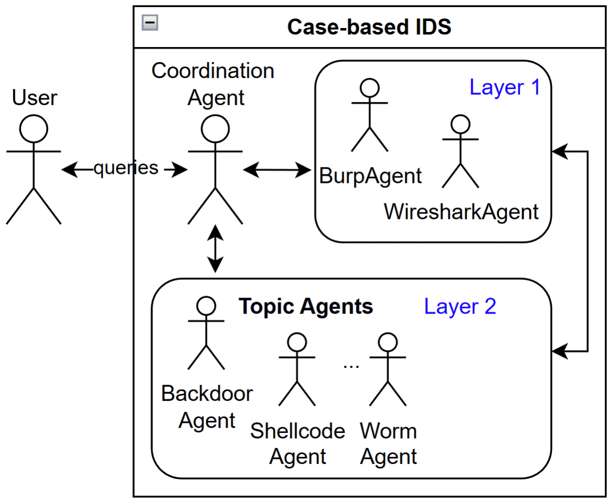


Fig. 4. Overview of the implementation, including ten topic agents.

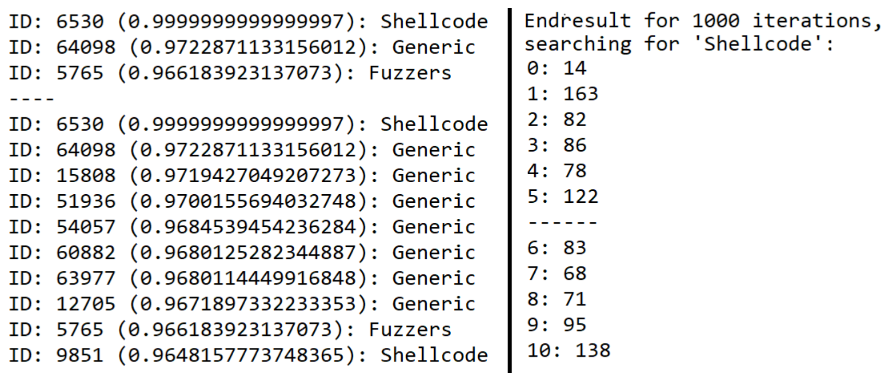
attribute. For example, for protocol, we check whether the same protocol has been used, as a lexicographical distance is not applicable here (in contrast to the IP source address, as it might be interesting whether the attack is originating from the same subnetwork). For the most numeric attributes, a symmetrical polynomial function has been established. Exception here is the attribute “Port”: Port numbers are assigned in various ways, based on three ranges: System Ports (0-1023), User Ports (1024-49151), and the Dynamic and/or Private Ports (49152-65535); the different uses of these ranges are described in RFC6335[16]. We treat the similarity of ports to the three groups accordingly. After all knowledge containers are initialized with given values of the knowledge engineer and the training data, the myCBR files are stored to the local disk and the initialization process of the agent is finalized. Using the stored files allows us to load agents for the testing data set on the fly. The agents can easily be adjusted to fit for other training data sets, such as mentioned in Section 2.1, as well.

In case of importing the testing data set, already established agents are activated by the coordination agent. Additionally, the user provides a positive number a for the minimum number of different attack categories that should be displayed in the result and a positive number $c \geq a$ for the number of cases that should be presented. This allows the user to receive a broader picture of the similarity distribution between multiple attack categories to prevent missing out on ambiguous results.

Before going through the test dataset, we filter the test dataset by attacks and confront the pool of all agents with a certain attack for training purposes.

[16] see https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

For example, consider we filter the dataset by the attack ‘Shellcode’. The goal is to identify, whether the corresponding agent can identify its attack, i. e., wins the majority votum. Figure 5 shows (left) one exemplary end result after one voting iteration. The first three values (above the horizontal line) are **not** the overall three best cases, but instead the best cases of three distinct attacks[17]. For the test case labeled as Shellcode, the ShellcodeAgent provides the best case with 99,99 % similarity (ID 6530). However, 7 votes of the GenericAgent made it into the best 10 cases (below the horizontal line). Thus, in terms of a majority vote, the attack has been incorrectly identified (False-Positive). Figure 5 shows (right) the end result of the Shellcode Agent. The result reads as follows: Out of 1000 iterations, the attack ‘Shellcode’ has been voted for by 0,1,2,....,10 times by the Shellcode Agent. 6 or more votes are treated as ’success’ (→ the sum of all results below the horizontal line). The more votes, the better the detection rate of an attack.



```
ID: 6530 (0.9999999999999997): Shellcode
ID: 64098 (0.9722871133156012): Generic
ID: 5765 (0.966183923137073): Fuzzers
----
ID: 6530 (0.9999999999999997): Shellcode
ID: 64098 (0.9722871133156012): Generic
ID: 15808 (0.9719427049207273): Generic
ID: 51936 (0.9700155694032748): Generic
ID: 54057 (0.9684539454236284): Generic
ID: 60882 (0.9680125282344887): Generic
ID: 63977 (0.9680114449916848): Generic
ID: 12705 (0.9671897332233353): Generic
ID: 5765 (0.966183923137073): Fuzzers
ID: 9851 (0.9648157773748365): Shellcode
```

```
Endresult for 1000 iterations,
searching for 'Shellcode':
0: 14
1: 163
2: 82
3: 86
4: 78
5: 122
------
6: 83
7: 68
8: 71
9: 95
10: 138
```

Fig. 5. (left) Example result after one voting iteration; (right) End result of the ShellcodeAgent. Both are used to analyze the performance of the classifiers for the domain expert to validate the systems output and performance, e. g., validating the similarity assessment.

Table 2 presents the results for all ten agents for the described test phase. The agents Analysis, Backdoor, DoS, Shellcode, Worm are not able to identify their attacks, whereas Normal and Generic agents achieve good results. Exploit, Fuzzer, and Reconnaissance agents achieve acceptable, but improvable results. We explicitly listed the counter for false-positives: for example, in 575 of 1000 cases where Analysis was the labelled attack, the Exploit agent won the majority vote instead. Further investigating the amount of false-positives, the DoS, Exploit and Reconnaissance agents cause a high amount of disturbance.

[17] As mentioned before, this is done to identify how close different attacks are to each other. This can be helpful to adjust similarities regarding false-positive results.

Vote x	Analys	Backdo	DoS	Exploi	Fuzzer	Generi	Normal	Recon	Shell	Worm
0	908	823	481	192	162	14	124	131	325	119
1	38	53	112	23	95	5	50	102	267	10
2	33	67	147	20	65	7	33	45	155	1
3	9	12	35	27	61	12	10	24	98	0
4	8	43	59	40	70	9	15	21	53	0
5	4	2	48	23	72	18	23	5	51	0
6	0	0	46	61	73	14	9	15	25	0
7	0	0	14	186	72	20	15	13	12	0
8	0	0	33	58	88	20	13	11	13	0
9	0	0	8	87	116	141	33	90	0	0
10	0	0	17	283	126	740	675	603	1	0
False-Pos.	Analys	Backdo	DoS	Exploi	Fuzzer	Generi	Normal	Recon	Shell	Worm
Analysis	-	0	0	0	0	0	0	0	0	0
Backdoor	0	-	0	0	0	0	0	0	0	0
DoS	104	183	-	98	48	8	10	50	77	1
Exploits	575	451	502	-	200	16	127	221	259	87
Fuzzers	15	14	16	15	-	10	32	3	12	0
Generic	0	2	0	2	1	-	0	0	0	12
Normal	17	0	1	0	9	0	-	0	2	0
Recon	67	94	109	71	47	1	24	-	254	12
Shellcode	0	3	1	0	1	0	0	0	-	0
Worms	0	0	0	0	0	0	0	0	0	-
TPR	0	0	11.80	67.50	47.50	93.50	74.50	67.20	05.10	0
FPR	77,80	74,70	62.90	18.60	30.60	03.50	19.30	27.40	60.04	11.20
Precision	0	0	15.80	78.40	60.82	96.39	79.42	71.04	7.83	0
Recall	0	0	11.80	67.50	47.50	93.50	74.50	67.20	5.10	0
F1	0	0	13.51	72.54	53.34	94.92	76.88	69.06	6.17	0

Table 2. Training scenario: 1000 cases per agent are presented. The table depicts how well agents are able to detect test cases with their respective label. The top half of the table presents the results of the majority votum, reading: 'Out of 1000 cases, the attack *column name* has been voted for by x agents for n times.'

5 Evaluation

5.1 Limitations

During the first test runs we encountered a few challenges with the data set, which resulted into limitations to this prototype version.

1. *Redundancy*
   As to be expected, the training data set contained multiple redundant cases, containing the same attribute-value pairs. If these cases turn out to be the most similar case for a given testing data input query case, the majority vote will easily be flooded by the redundant cases. A relatively quick fix to this challenge would be to simply remove redundant cases and remain with one pivotal case. The occurrence of a large amount of redundant cases might contain context information, which should not easily be discarded. However, a more elegant and efficient way would be a proper introduction of case base maintenance under the aspect of pivotal cases, and coverage and reachability of cases in a casebase as introduced by Smyth & Keane [14].

2. *Same case, different attack category*
   We identified multiple cases with exactly the same attribute-value pairs, but different attack category labels (249-Analysis, 710-DoS, 1413-Reconnaissance, 1416-Exploits, 3421-Fuzzers). During the training phase, and tests within the training data set, this resulted into a 100 % similarity for a given case for multiple different attack categories, which can easily lead to an increased rate of false-positives.

5.2 Results

Table 3 presents our results of querying the topic agents with the test data set. We confronted the agents with 50 000 cases randomly selected out of the test data set[18]. Each activated topic agent voted with their $n = 10$ most similar cases. Out of this pool of best cases (100 with 10 active agents), the 10 most similar cases overall have been chosen. Each correct vote will be counted: if there are at least six correct votes, the query is considered as classified[19]. All correct votes will be summarized and we provide all necessary variables to calculate Precision $\left(\frac{TPR}{TPR+FPR}\right)$, Recall $\left(\frac{TPR}{TPR+FNR}\right)$, F1 $\left(2\frac{precision * recall}{precision + recall}\right)$.

	Analys	Backdo	DoS	Exploits	Fuzzers	Generic	Normal	Recon	Shell	Worm
Count	869	739	4567	12916	7366	6653	12591	3806	439	53
TPR	0,00	0,00	70,73	94,59	86,97	98,15	99,42	97,32	30,77	0,00
FPR	0,00	0,00	74,48	84,28	72,03	59,34	32,86	64,80	26,32	0,00
TNR	100,00	100,00	25,52	15,72	27,97	40,66	67,14	35,20	73,68	0,00
FNR	100,00	100,00	29,27	5,41	13,03	1,85	0,58	2,68	69,23	0,00
Precision	0,00	0,00	26,22	59,11	70,39	98,30	98,37	54,38	28,57	0,00
Recall	0,00	0,00	70,73	94,59	86,96	98,14	99,41	97,32	30,76	0,00
$F1$	0,00	0,00	38,26	72,76	77,81	98,23	98,89	69,78	29,63	0,00

Results of other approaches (no CBR) using the same dataset	
Wheelus et al. [17]	ranging from 69 % to 83 % TPR for all attacks
Pratomo et al. [10]	69,21 % TPR on average for all attacks
Mebawondu et al [8]	76,96 % TPR for all attacks
Ullah and Mahmoud [15]	97 % F1 score with 97 % precision and recall
Anwer et al. [1]	accuracy of 80 % using 18 features
Wu and Guo [19]	accuracy of 82.78 % at best, FPR of 4.72 %
Our proposed approach	82,56 % TPR (excluding Analysis, Backdoor, Worms)

Table 3. Results of quering the MAS using the test dataset. Additionally, direct comparison to other approaches using the same data set.

As the results in Table 3 show and as the preliminary results of Table 2 suggested, we receive very good results for Normal and Generic agent and acceptable results for Exploits, Fuzzers and Reconnaissance agents. As described

[18] The complete test data set contains 175341 entries.
[19] Either true-positive or false-negative

earlier, DoS, Exploits and Reconnaissance agents are disturbing with too many false-positive results. This is very likely the result of an improper similarity modeling of those agents and will be further investigated in future work. Excluding Analysis, Backdoor, and Worm Agents - as these attacks cannot be detected by the proposed system, yet - leaves us with 82,56 % TPR which is competitive with similar approaches. The exclusion of the mentioned agents is easily done by having the multi-agent structure.

To receive better results, we plan to integrate SHAP (SHapley Additive exPlanations) as a technique for explaining the output by assigning importance values to the input features. The SHAP value of a feature represents the contribution of that feature to the difference between the actual prediction and the average prediction for all possible combinations of features. This will allow us to further sharpen the similarity measures, more precisely, the weights of the attributes. Furthermore, it can aid in model debugging, interpretation, and communication of results. Nevertheless, the results indicate a possibility of using CBR in an important domain - the IT security - with promsing results.

6 Conclusion

We present a transparent multi-agent based CBR system for supporting intrusion detection in a network using the UNSW_NB15 data set for training and testing. Each topic agent of the multi-agent system contains its own casebase, similarity measure, vocabulary, and adaptation knowledge. The modeling, e. g., the assessment of similarity measures and weights, has mostly been done based on our expertise of the domain and based on the identification of distinct attribute-value pairs, characterizing given attack categories described in Section 3.

Despite the limitations described in Section 5.1, we were able to detect normal and generic traffic - competitive with other (non-CBR) approaches. Four agents need further adjustments to achieve better results, while four agents need to be remodeled. Especially the Worm agent seems to suffer from a low casebase (44 cases in the training set). Nevertheless, Worms contain by far the most distinct and characteristic values, which leaves us optimistic to receive better results after further fine-tuning of the local similarity measures.

For further adjustments and future work, we are looking to integrate the KDD-CUP-99 data set and the NSL-KDD data set. As the results of Anwer et al. [1] suggests, it seems possible to achieve reasonable results with a relative low number of attributes. Thus, it could be helpful to find overlapping attributes in the three different data sets and include these in the casebase of the corresponding agents. This way, we can further emphasize on the usefulness of a multi-agent system by using different sources to enrich our agents.

Furthermore, the modeling can possibly be supported by introducing SHAP values, as we are already planning to integrate explainability to the system a next milestone. Explainability further increases the transparency of the system and should help both, the knowledge engineer, and the user of the system, to understand its decision making process.

References

1. Anwer, H. M.; Farouk, M.; Abdel-Hamid, A. (2018): *A framework for efficient network anomaly intrusion detection with features selection.* In: 9th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 2018, pp. 157-162.
2. Bach, K. (2013): *Knowledge Acquisition for Case-Based Reasoning Systems*, Ph. D. thesis, University of Hildesheim, 2013. URL: http://www.dr.hut-verlag.de/978-3-8439-1357-7.html.
3. Bergmann, R. (2022): *Experience Management: Foundations, Development Methodology, and Internet-Based Applications.* In: volume 2432 of Lecture Notes in Computer Science, Springer, 2002.
4. Verma, D. (2004): *Supporting Service Level Agreements on IP Networks.* In: Proceedings of IEEE/IFIP Network Operations and Management Symposium, 92(9), (pp. 1382-1388). NY, USA.
5. Divekar, A.; Parekh, M.; Savla, V.; Mishra, R. and Shirole, M. (2018): *Benchmarking datasets for Anomaly-based Network Intrusion Detection: KDD CUP 99 alternatives.* In: IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), Kathmandu, Nepal, 2018, pp. 1-8.
6. Enisa - European Union Agency for Cybersecurity (2021): *NIS Investments 2021.* URL: https://www.enisa.europa.eu/publications/nis-investments-2021. Last validation: 04/11/2023. (p. 15).
7. Enisa - European Union Agency for Cybersecurity (2022): *NIS Investments 2022.* URL: https://www.enisa.europa.eu/publications/nis-investments-2022. Last validation: 04/11/2023. (pp. 10,44,72).
8. Mebawondu, J. O., Alowolodu, O. D., Mebawondu, J. O., Adetunmbi, A. O. (2020): *Network intrusion detection system using supervised learning paradigm.* In: Scientific African 9, e00497, 2020.
9. Moustafa, N. and Slay, J. (2015): *UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set).* In: Military Communications and Information Systems Conference (MilCIS), 2015, pp. 1–6.
10. Pratomo, B. A.; Burnap, P.; Theodorakopoulos, G. (2018): *Unsupervised approach for detecting low rate attacks on network traffic with autoencoder.* In: International Conference on Cyber Security and Protection of Digital Services (Cyber Security). pp. 1–8, 2018.
11. Richter, M. M. (1995): *The knowledge contained in similarity measures*, Invited Talk at the First International Conference on Case-Based Reasoning, ICCBR'95, Sesimbra, Portugal, 1995.
12. Ring, M.; Wunderlich, S.; Scheuring, D.; Landes, D.; Hotho, A. (2019): *A survey of network-based intrusion detection data sets.* In: Comput. Secur. 86: 147-167.
13. Schoenborn, J. M. and Althoff, K. D. (2022): *Multi-Agent Case-Based Reasoning: a Network Intrusion Detection System.* LWDA 2022: Lernen, Wissen, Daten, Analysen, Hildesheim, pp. 258-269.
14. Smyth, B., Keane, M. T.: *Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems.* In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes. pp. 377–383. Morgan Kaufmann (1995).
15. Ullah, I. and Mahmoud, Q. H. (2019): *A Two-Level Hybrid Model for Anomalous Activity Detection in IoT Networks.* In: 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2019, pp. 1-6.

16. Wess, S. (1995): *Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik: Grundlagen, Systeme und Anwendungen (translated: Case-based problem solving in knowledge-based systems for decision support and diagnostic: basics, systems and applications)*. Ph.D. thesis, University of Kaiserslautern, Infix-Verlag, 1995.
17. Wheelus, C., Bou-Harb, E., Zhu, X. (2018): *Tackling class imbalance in cyber security datasets*. In: IEEE International Conference on Information Reuse and Integration (IRI). pp. 229–232, 2018.
18. Wu, L.; Garg, S. K.; Buyya, R. (2015): *Service Level Agreement (SLA) Based SaaS Cloud Management System*. ICPADS 2015: 440-447
19. Wu, P. and Guo, H. (2019): *LuNet: A Deep Neural Network for Network Intrusion Detection*. In: IEEE Symposium Series on Computational Intelligence (SSCI), Xiamen, China, 2019, pp. 617-624, 2019.