

Experimenting with neurosymbolic AI for defending against cyber attacks

Magnus Wiik Eckhoff^{a,b}, Jonas Halvorsen^a, Bjørn Jervell Hansen^a, Martin Eian^c,
Vasileios Mavroeidis^b, Robert Andrew Chetwyn^b, Geir Skjøtskift^c and Gudmund Grov^{a,b,*}

^a *Norwegian Defence Research Establishment (FFI), Norway*

*E-mails: Magnus-Wiik.Eckhoff@ffi.no, Jonas.Halvorsen@ffi.no, Bjorn-Jervell.Hansen@ffi.no,
Gudmund.Grov@ffi.no*

^b *Department of Informatics, University of Oslo, Norway*

E-mails: vasileim@ifi.uio.no, roberac@ifi.uio.no

^c *mnemonic, Norway*

E-mails: meian@mnemonic.no, geir@mnemonic.no

Abstract.

It is generally accepted that all cyber attacks can not be prevented, and it is therefore necessary to have the ability to detect and respond to cyber attacks. Both connectionist and symbolic approaches are currently being employed for this purpose, but far less work has been done on the intersection of the two. This paper argues that the cyber security domain holds significant potential for applying neurosymbolic AI. We identify a set of challenges faced in cyber security today, and from this, we propose a set of neurosymbolic use cases that can help address the challenges. Feasibility is demonstrated through multiple experiments that apply neurosymbolic AI to cyber security. We find a significant overlap between the challenges in cyber security and the promises of neurosymbolic techniques, making it an interesting research direction for both the neurosymbolic AI and cyber security communities.

This paper is an extended version of a paper published at the NeSy 2024 conference [49]. The main additional contributions are further experimental evidence for our hypothesis that NeSy offers real benefits in this domain and a more in-depth treatment of knowledge graphs for cyber security.

Keywords: AI, neurosymbolic AI, cyber security, incident detection and response

1. Introduction

Protecting assets in the cyber domain requires a combination of *preventive measures*, such as access control and firewalls, and the ability to *defend* against cyber operations when the preventive measures were not sufficient.¹

Our focus in this paper is on defending against offensive cyber operations, and before going into details, we put in place some concepts and terminology:

*Corresponding author. E-mail: Gudmund.Grov@ffi.no.

¹It is generally accepted that preventive measures are not sufficient in cyberspace. An analogy is that we still need smoke detectors and a fire brigade, even if we take all possible preventive measures to reduce the risk of a fire.

Terminology & concepts The focus of this paper is to defend *assets* against *threats* in cyberspace. An *asset* can be anything from information, physical infrastructure, and people - to the internal processes of an enterprise. *Threats* manifest themselves in the form of *cyber operations* (or *cyber attacks*) conducted by an *adversary* (or *threat actor*). In the context of defending, the term *incident* is used for an event that is deemed to have a negative impact on assets. This event can potentially be an attack, and the process of defending comes under the area of *incident management* [25].

This is typically carried out in a *Security Operations Centre* (SOC), which consists of people, processes, and tools [40]. One of the objectives of a SOC is to detect and respond to threats and attacks, where *security analysts* play a crucial role. Knowledge of threats in the cyber domain is developed by conducting intrusion analysis and producing and consuming *Cyber Threat Intelligence* (CTI). Networks and systems to be protected are monitored, and *events* – e.g., network traffic, file changes, or processes executing on a host – are forwarded and typically stored in a *security information and event management* (SIEM) system, where events can be investigated, correlated and enriched – and queried. We will use the term *observations* for such events resulting from monitoring. Suspicious activity that is observed may raise *alerts*, which may indicate an incident that has to be analysed and responded to in the SOC. Finally, *Neurosymbolic AI* [43], which aims to combine connectionist and symbolic AI, will be abbreviated *NeSy*.

Why is a SOC relevant for NeSy? A SOC essentially conducts abductive reasoning by observing traces and identifying and analysing their cause in order to respond. This involves sifting through masses of events for suspicious behaviour, an area in which extensive research has been conducted for several decades using statistics and machine learning. Identifying the cause of observed suspicious behaviour requires situational awareness, achieved by combining different types of evidence, applying reasoning, and deriving knowledge. There are various ways in which evidence and knowledge can be represented, such as structured events and alerts, unstructured reports, and semantic knowledge [82, 122].

In a SOC, the ability to *learn* models to detect suspicious activities and the ability to *reason* about identified activities to understand their cause and respond to them is thus required. These abilities are at the core of NeSy, and our hypothesis is as follows:

A SOC provides an ideal environment to study and apply NeSy with great potential for both scientific and financial impact.

Some early work has explored NeSy in the cyber security domain [31, 55, 59, 90, 102, 108] and our goal with this paper, which extends [49], is to showcase the possibilities and encourage the NeSy community to conduct research in the SOC field² with an emphasis on experiments.

Methodology: The identified SOC challenges are derived from a combination of existing published studies, the experience and expertise of the authors, and further discussions with SOC practitioners. The use cases result from reviewing NeSy literature in the context of the identified challenges, and the preliminary experiments conducted are based on a subset of the identified use cases.

Contributions: This paper is an extended version of [49], published at the NeSy 2024 conference. We outline how AI is used today in a SOC and identify and structure a set of challenges faced by practitioners who use AI. We then create a set of promising use cases for applying NeSy in the context of a SOC, review current NeSy approaches in light of them, and demonstrate feasibility through proof-of-concept experiments. In this paper, we extend [49] with a more profound treatment of the use of knowledge graphs when defending against cyber attacks, and crucially, we address the main limitation of [49] – limited experimental evidence – with the following experiments:

- Our experiment with *Logic Tensor Networks* (LTN) [12] in [49] is remade with an LTN using the same structure as a published ML model for network intrusion detection [116], and extended with further experiments addressing explainability aspects and prioritising crucial knowledge.

²This paper focuses on the use of NeSy to defend against cyber attacks. Consequently, approaches that address NeSy in the context of AI security, such as [90], are not considered in scope.

- 1 – A NeSy technique called *Embed2Sym* [9] is explored to analyse and contextualise alerts from intrusion detec- 1
- 2 tion systems. 2
- 3 – Building on [23, 24], we explore the integration of large language models with a symbolic approach for *threat* 3
- 4 *hunting*. 4
- 5 – We extend our previous work using data-driven enrichment of (symbolic) knowledge [123] with experiments 5
- 6 using newly released data and explore the advantages NeSy provides for this challenge. 6

7 *Paper structure:* In §2, we describe the typical use of AI in a SOC and the identified challenges. In §3, we make 7

8 the case for NeSy and introduce the different NeSy techniques discussed in this paper. In §4, we outline the NeSy 8

9 use cases and suggest NeSy techniques to address them. In §5, we describe the proof-of-concept experiments, before 9

10 we conclude in §6. 10

11 2. Challenges faced when using AI in a SOC 11

12 12

13 MAPE-K (Monitor-Analyse-Plan-Execute over shared Knowledge) [68] is a common reference model to struc- 13

14 ture the different phases when managing an incident.³ For each phase of MAPE-K, we below discuss the use of AI, 14

15 including underlying representations, and identify key challenges security practitioners face when using AI.⁴ 15

16 2.1. Monitor 16

17 17

18 In the *monitor* phase, systems and networks are monitored and telemetry is represented as sequences of *events*. 18

19 An *event* could, for instance, be a network packet, a file update, a user that logs on to a service, or a process being 19

20 executed. Events are typically structured as key-value pairs. In a large enterprise, tens of thousands of events may 20

21 be generated per second. In this phase, a key objective is to detect suspicious behaviours from events and generate 21

22 *alerts*, which are analysed and handled in the later phases of MAPE-K. 22

23 23

24 This is a topic where machine learning (ML) has been extensively studied by training ML models on the vast 24

25 amount of captured event data (e.g., [3]). A challenge with such data is the lack of *ground truth*, in the sense that for 25

26 the vast majority of events we do not know if they are benign or malicious. As most events will be benign (albeit 26

27 we do not know which ones), one can exploit this assumption and use unsupervised methods to train anomaly 27

28 detectors. This is a common approach. For at least research purposes, synthetic data sets from simulated attacks are 28

29 also commonly used [72]. However, synthetic data sets suffer from several issues [8, 37, 67] and promising results 29

30 in research papers using synthetic data often fail to be reproduced in real-world settings – whilst anomaly detectors 30

31 often create a high number of false alerts⁵ [18, 124]. Our first challenge, which has also been identified by the 31

32 *European Union Agency for Cybersecurity* (ENISA) [106], identifies this performance issue for ML models under 32

33 real-world conditions: 33

34 34

35 **Challenge 1.** *Achieve optimal accuracy of ML models under real-world conditions.* 35

36 36

37 As benign software and malware are continuously updated, the notion of *concept drift* is prevalent and ML models 37

38 therefore must be re-trained regularly. There are some approaches that take such concept drift into account [6, 101]. 38

39 In addition to the need for scalability, due to the large amount of data, real-world conditions introduce a significant 39

40 level of noise (i.e. aleatoric uncertainty) in the data, which is not well reflected in synthetic data. 40

41 41

42 We know the ground truth of the associated alerts and events for previous incidents that have been handled. 42

43 Compared to the set of all events, the alerts related to incidents make up only a tiny fraction. Typically, the majority 43

44 of events will be benign, resulting in data sets that are heavily unbalanced. This imbalance is a challenge during 44

45 45

46 46

47 ³Other common reference models are the OODA (Observe-Orient-Decide-Act) loop [16] and the IACD (Integrated Adaptive Cyber Defence) 47

48 framework [32]. 48

49 ⁴There have recently been a vast number of proposals for using *large language models* (LLMs) across MAPE-K. Here, we include what we 49

50 currently consider the most promising uses of LLMs and refer to [15, 95] for a more complete discussion. We provide a more detailed account 50

51 of LLMs in the NeSy context in §3 and §4. 51

⁵As most events are benign, the base rate fallacy is important in this domain.

both training and inference of machine learning models. Still, the previous incidents are vital as they are labelled and contain relevant data – either in terms of actual attacks experienced or false alerts that should be filtered out. One important challenge, also recognised by ENISA, is the ability to exploit such labelled “incident data sets” and train ML models based on them:

Challenge 2. *Learning with small (labelled) data sets (from cyber incidents).*

New knowledge about threats, attacks, malware, or vulnerability exploits is frequently published, often in the form of threat reports and advisories. The traditional, and still most common, method for threat detection is *signature-based*, where knowledge is encoded (often manually) as specific patterns (called signatures). Detection is achieved by matching events with these signatures and generating alerts. Although signature-based methods have their limitations, such knowledge could improve the performance of ML-based detection models, creating the need for the ability to extract relevant knowledge and include it in the ML models:

Challenge 3. *Extract knowledge (including about threats, malware, and vulnerabilities) and enrich ML-based detection models with it.*

In recent years, some work has been done to encode cyber knowledge into AI systems to improve detection [14, 55]. In addition to reports, dedicated knowledge bases, formal ontologies, and knowledge graphs can be used to enrich ML models with such knowledge. There are also attempts to extend the coverage domain for such ontologies, including the *unified cyber ontology* (UCO) [126] and the SEPSES knowledge graph [71]. To represent cyber threat intelligence (CTI), a commonly used schema is *Structured Threat Information eXpression* (STIX) [99], with the associated *Threat Actor Context* (TAC) ontology [88]. However, the maintenance of ontologies and knowledge graphs has its own challenges, as real-world domains are rarely static, meaning new concepts emerge, existing ones evolve, and old ones may become obsolete. In addition, domain-specific ontologies (tailored for specific use cases) often do not rely on a shared foundation, i.e. *foundational ontologies*. This causes duplicate efforts with incompatible structures (reinvention and misalignment of concepts) for common concepts, resulting in fragmented knowledge representation, which makes cross-domain data integration complex and error-prone. This might impact inferencing, ultimately leading to incomplete or incorrect results.

A widely used knowledge base for threat actors and attacks is MITRE ATT&CK [94]. ML, and in particular *natural language processing* (NLP), is being explored for extracting CTI into symbolic forms (e.g., STIX) [87] or to map it to MITRE ATT&CK [80]. *Large Language Models* (LLMs) are also explored for this topic [53, 81]. Some limitations have been identified [134], but this is an active area of development with improvements made all the time. We are unfamiliar with approaches that combine and integrate such knowledge with ML models for detection trained on events.

A different approach to identify malicious behaviour is *cyber threat hunting*. This is a *hypothesis-driven* approach in which hypotheses are formulated iteratively (typically using CTI) and validated using event logs, as well as other sources of information [121]. Automating this process is our final challenge for the monitor phase:

Challenge 4. *Automated generation of hypotheses from CTI and validation of hypotheses using observations for threat hunting.*

Utilizing AI to facilitate threat hunting is continuously explored [97]. However, most have focused on supporting hypothesis generation by extracting relevant CTI, and using both ML/NLP [42] and symbolic AI [112]. In addition, symbolic AI has been used to support validation [24]. Although symbolic and subsymbolic approaches have been applied, a more comprehensive, integrated, and effective approach is needed.

2.2. Analyse

The goal of the analyse phase is to understand the nature of the observed alerts, determine possible business impact, and create sufficient situational awareness to support the subsequent plan and execute phases.

Both malware and benign software continuously evolve. This makes it difficult to separate malicious from benign behaviour [106], even with continuous detection engineering efforts. For example, an update to benign software may cause a match with an existing malware signature and may also appear as an anomaly in the network traffic. As

1 a result, most of the alerts would be either false or not sufficiently important for further investigation [18], causing 1
2 so-called *alert fatigue* among security analysts in a SOC. The analysis phase is, therefore, labour-intensive, where 2
3 security analysts must plough through and analyse a large number of alerts – most of them false – to decide their 3
4 nature and importance: 4

5 **Challenge 5.** *The volume of alerts leads to alert flooding and alert fatigue in SOCs.* 5
6

7 Understanding the nature of alerts is essential, and studies have shown that a lack of understanding of the under- 7
8 lying scores or reasoning behind the alerts have led to misuse and mistrust of ML systems [100]. Studies [18], along 8
9 with guidance from ENISA [106], have highlighted the need for alerts to be reliable, explainable, and possible to 9
10 analyse. The use of explainable AI to support this has shown some promise [34], and both knowledge graphs [18] 10
11 and LLMs⁶ [57, 61, 70] have been identified as promising approaches. 11

12 An alert is often a single observation and needs to be placed into a larger context to determine an *incident* and 12
13 provide the necessary *situational awareness* as a result of an analysis [38]. Such contextualisation includes enriching 13
14 alerts with relevant knowledge from previous incidents, common systems behaviour, infrastructure details, threats, 14
15 assets, etc. The same attack – or the same phase of an attack – is likely to trigger many different alerts. Different 15
16 ML techniques, particularly clustering, have been studied to fuse or aggregate related alerts [74, 127]. In addition 16
17 to understanding an incident and achieving situational awareness, contextualisation will also help a security analyst 17
18 understand individual alerts. Similarly to challenge 3, contextualization of alerts will involve extracting a symbolic 18
19 representation from a vast amount of available (and typically unstructured) information. 19

20 A cyber attack conducted by an advanced adversary will, in most cases, go through several phases to reach its 20
21 objectives, creating a need to discover the relationships (between the alerts) across the different phases of an attack. 21
22 A common reference model to relate such phases is the *cyber kill chain*, originally developed by Lockheed Martin 22
23 and later refined into the *unified cyber kill chain* [110]. Other formalisms that enable modelling different phases of 23
24 attacks include *MITRE Attack Flow* [93] and the *Meta Attack Language (MAL)* [60]. Different approaches have been 24
25 studied to relate the different phases, including symbolic approaches [104], AI planning [5, 92], knowledge graphs 25
26 [24, 75], state machines [132], clustering [51] and statistics [53]. However, this research topic is considerably less 26
27 mature compared with ML models for detection in the monitor phase. We summarise the challenges of combining, 27
28 understanding and explaining observations in the following challenge: 28

29 **Challenge 6.** *Combine observation with knowledge to analyse, develop, and communicate situational awareness.* 29
30

31 Developing cyber situational awareness requires connecting a plethora of different sources, such as alerts and details 31
32 about infrastructure and threats. There have been proposals to use knowledge graphs to combine these different 32
33 sources to support analysis [82, 122], including providing explanations [18]. 33

34 When an incident is understood and sufficient situational awareness is achieved, a suitable amount of resources 34
35 have to be allocated to handle the incident. There may be multiple incidents requiring some prioritisation between 35
36 them. This involves understanding the risk and potential impact of the incident, including any mitigating actions 36
37 that may be taken in subsequent MAPE-K phases: 37

38 **Challenge 7.** *Understanding the risk, impact, importance, and priority of incidents.* 38
39

40 **2.3. Plan & Execute** 40
41

42 The last two phases of MAPE-K, plan and execute, focus on responding to detected incidents. This involves 42
43 finding suitable responses in the plan phase and preparing and executing the response(s) in the execute phase. From 43
44 an AI perspective, research in these phases is less mature than in the monitor and analyse phases. Therefore, we will 44
45 only focus on the plan phase, which we consider to have more interesting AI-related challenges. 45
46

47 To plan a suitable response, three promising AI techniques are *AI planning* (e.g., [47]), *reinforcement learning* 47
48 (RL – e.g., [58, 98]) and *recommender systems* (e.g., [109]). Each of these techniques has pros and cons: AI planning 48
49 requires considerable knowledge and formulation of the underlying environment, reinforcement learning requires a 49
50

51 ⁶E.g. Microsoft security co-pilot, Elastic Security generative AI [22], and an Elastic/LongChain initiative [33] 51

considerable amount of interactions/simulations (often in the millions) and recommender systems typically require extensive knowledge of previous events. In certain cases, a quick response time is necessary, which means this level of interaction would be too time-consuming. When generating response actions, their risk and impact must be taken into account (including the risk and impact of not acting), which is an unsolved problem when using AI. Moreover, when proposing a response action, an AI-generated solution must be able to explain both what the response action will do and why it is suitable for the given problem:

Challenge 8. *Generate and recommend suitable response actions in a timely manner that take into account both risk and impact and are understandable for a security analyst.*

To support such generation, there are several frameworks and formal ontologies that can be used, such as MITRE *D3FEND* [63], *RE&CT* [10] and *CACAO playbooks* [89].

2.4. Shared knowledge

The ‘K’ in MAPE-K stands for *knowledge* shared across the phases. We have, for instance, seen knowledge about threats and the infrastructure being protected used across different phases. Moreover, this knowledge takes different forms and representations (structured and unstructured) and is analysed using different techniques (symbolic and sub-symbolic). In addition to consuming knowledge, it is also important to share knowledge with key stakeholders, both technical and non-technical [128]. This may be a report about an incident for internal use (e.g., to board members) or sharing threat information with a wider community, which lead us to our final challenge:

Challenge 9. *Generating suitable incident and CTI reports for the target audience.*

2.5. Summary

We have shown the need to learn and reason across MAPE-K and that both symbolic and connectionist AI are being used across the phases. We have identified several challenges, which we in the following section will address from a NeSy perspective.

3. Neurosymbolic AI to defend against cyber attacks

Kahneman’s [62] distinction between (fast) instinctive and unconscious ‘*system 1*’ processes from (slow) more reasoned ‘*system 2*’ processes has often been used to illustrate the NeSy integration of neural networks (system 1) and logical reasoning (system 2). This interdisciplinary approach integrates neural networks adept at learning from vast amounts of unstructured data, with symbolic representations of knowledge and logical rules to enhance the interpretability and reasoning capabilities of AI systems. Building on the above analogy, system 1 can, in a SOC, be seen as the ML-based AI used to identify potentially malicious behaviour in the monitor phase. Here, a large amount of noise needs to be filtered out from the large amount of events (thus a need for speed and scalability). System 2 is the reasoning conducted in the analysis phase, where deeper insight is required, and the need for scalability is less significant. This dichotomy of requirements entails that neither end-to-end pure statistical nor pure logical approaches will be sufficient, and a NeSy combination seems ideal. Three commonly used reasons for pursuing NeSy are to design systems that are *human auditable and augmentable*, can *learn with less* and provide *out-of-distribution generalisation* [48]. We have seen examples of each of these in the challenges described in §2: the use of knowledge to contextualise, analyse, and explain alerts; generate and explain response actions; learn from (relatively few) incidents; and handle concept drifts and noise in order to achieve high accuracy of ML models under real-world conditions.

There are multiple studies on the current trends in neurosymbolic artificial intelligence [13, 43, 117], which we will not repeat here. Instead, we will briefly describe the NeSy techniques we have found relevant for the use cases in §4, categorised according to the taxonomy first introduced by Henry Kautz during his AAAI Robert S. Engelmore memorial lecture [65]. This taxonomy was later revised at the 2024 Neurosymbolic AI summer school [66]. This revised Kautz taxonomy consists of the following eight categories:

- 1 – *Symbolic* \rightarrow *Neuro* \rightarrow *Symbolic*: Techniques where symbolic input (like language) is converted into non-
2 symbolic input (like vectors) that is passed to a neural network. The output of the neural network is then turned
3 back into symbols (like a category or a sequence of symbols).
- 4 – *Symbolic*[*Neuro*()]: A symbolic problem solver that can call a neural subroutine when needed.
- 5 – *Neuro* \Leftrightarrow *Symbolic*: A neural system converts non-symbolic inputs into symbolic structures, which are pro-
6 cesses by a symbolic reasoning system. The errors can then be propagated back through the symbolic system
7 to the neural system.
- 8 – $f : \textit{Symbolic} \rightarrow \textit{Neuro}$: Standard deep learning with training based on symbolic rules. By generating training
9 input-output pairs from symbolic rules, a neural system can train on the pairs. After training, the system can
10 correctly handle unseen data.
- 11 – *Neuro*_{*Symbolic*}: Techniques in which symbolic rules are used as templates for structure within a neural compo-
12 nent. This means that the expressed logic determines the general structure of a neural network.
- 13 – *Neuro*^{*Symbolic*}: A neural system is structured in a manner that provides an explicit world model to help it reason
14 about future consequences of decisions.
- 15 – *Neuro*|*L*(*Symbolic*): Neural systems where symbolic knowledge is encoded in the loss function.
- 16 – *Neuro*[*Symbolic*()]: A neural engine that can call a symbolic reasoning system. This comprises systems with
17 deliberate system 2 reasoning with a system 1 engine.

18 Below we briefly describe the NeSy techniques used in §4 following this taxonomy. Note that (1) we only cover a
19 subset of the categories and only discuss the ones to which a relevant NeSY technique belongs; (2) many techniques
20 have aspects that mean they can fit into multiple categories. When this is the case, we have chosen the most relevant
21 category.

22 3.1. *Symbolic*[*Neuro*()]

23 **Differentiable Probabilistic Answer Set Programming (dPASP)** [46] is based on furnishing Answer Set Pro-
24 gramming (ASP) [17] with neural predicates as interface to both deep learning components and probabilistic
25 features in order to afford differentiable neurosymbolic reasoning. dPASP is suitable for detecting under
26 incomplete information, abductive reasoning, analysis of competing hypotheses (ACH) [54], and what-if rea-
27 soning.

28 **PyReason** [2] is a python framework supporting both differentiable logics and temporal extensions. Additionally,
29 it enables temporal reasoning over graphical structures with fully explainable traces of inference.

30 3.2. *Neuro* \Leftrightarrow *Symbolic*

31 **Neuro Symbolic Concept Learner (NS-CL)** [86] builds models to learn visual perception, including semantic in-
32 terpretation of the images without explicit supervision. It learns visual concepts, words, and semantic parsing
33 jointly.

34 **Neuro-Symbolic Inductive Learner (NSIL)** [28] is an approach in which a neural network learns to extract latent
35 concepts from raw data, while jointly learning a mapping of symbolic knowledge to latent concepts.

36 **Differentiable Inductive Logic Programming (dILP)** [35] is a framework that seeks to combine the advantages
37 of inductive logic programming (ILP) [96] with the advantages of neural networks. It enables learning logic
38 programs from noisy and structured examples.

39 **DeepProbLog** [85] and **DeepStochLog** [133] incorporate reasoning, probability, and deep learning, by extending
40 probabilistic logic programmes with neural predicates created from a neural classifier.

41 **Neural Probabilistic Soft Logic (NeuPSL)** [111] is a neurosymbolic framework where the output from the
42 trained neural networks is in (symbolic) *Probabilistic Soft Logic* (PSL) [11]. This enables reasoning over
43 low-level perceptions of deep neural networks.

44 **NeurASP** [135] is an extension of Answer Set Programming that incorporates neural networks. This is achieved by
45 treating the output from the neural classifier as a probability distribution over the atomic facts in the ASP. The
46 ASP rules can also be used to improve the training of the neural networks.

Deep Symbolic Learning (DSL) [29] is a neurosymbolic system that learns a set of perception functions, mapping images to symbols while also learning a symbolic function over the symbols in an end-to-end fashion.

STAR [114] combines LLMs with ASP. Knowledge is extracted to predicates using an LLM. ASP can then be employed to reason over the extracted knowledge.

Logic.py [69] is an approach to solving search-based problems with LLMs. The LLMs formalise a given problem in a domain-specific language *Logic.py*, which can be solved using a symbolic constraint solver.

Embed2Sym [9] extracts latent concepts from a neural network architecture and assigns symbolic meanings to these concepts. This enables solving tasks involving both perception and reasoning.

3.3. $f : S_{symbolic} \rightarrow Neuro$

Recurrent Reasoning Networks (RRNs) [56] is a neurosymbolic method for training a deep neural network to perform ontology reasoning. The RNN model is able to reason with an accuracy that is close to symbolic methods, while being more robust.

3.4. $Neuro_{symbolic}$

Logical Neural Networks (LNN) [115] are designed to simultaneously provide key properties of both neural nets (learning) and symbolic logic (knowledge and reasoning), enabling both logical inference and injecting desired knowledge into the neural architecture.

Logic Tensor Networks (LTN) [12] is an approach where a membership function for concepts is learnt based on both labelled examples and abstract (logical) rules. LTN introduces a fully differentiable logical language, called *real logic*, where elements of first-order logic can be used to encode the underlying knowledge.

3.5. $Neuro[S_{symbolic}()]$

Modular Reasoning, Knowledge and Language (MRKL) [64] systems present a neurosymbolic architecture to improve the utility of LLMs. The system consists of a set of expert modules and a router that routes incoming natural language to appropriate modules. The modules can be both neural (e.g. LLMs or vision modules) or symbolic (e.g. a calculator or an API call).

Phenomenal Yet Puzzling [113] presents an approach for inductive reasoning with language models. Inductive reasoning is done through *iterative hypothesis refinement*, and consists of three steps: proposing, selecting, and refining hypotheses. When coupled with a symbolic interpreter, accurate feedback can be given to refine the hypotheses.

Large Language Models Are Neurosymbolic Reasoners [36] investigates the application of LLMs as symbolic reasoners in text-based games. The LLM agents are given information about their role, observations, a set of valid actions arising from both the game environment, and a symbolic module. With this, agents can interact with the environment and solve text-based games involving symbolic tasks.

Symbolic Deep Reinforcement Learning (SDRL) [84] is a framework in which symbolic planning is introduced into deep reinforcement learning. This enables both high-dimension sensor input and symbolic planning.

Knowledge Graph Enhanced Retrieval Augmented Generation [76] deals with the limitations of large language models — like hallucinations and difficulty with factual data — by integrating knowledge graphs (KGs) for more reliable and contextually grounded outputs. This approach constructs a richer semantic context through ontology-based schemas and vector embeddings, enabling more effective retrieval and reasoning.

4. Neurosymbolic AI use cases to improve defending against cyber attacks

From the challenges in §2, we here outline a set of NeSy use cases we believe are promising. For each use case, we identify suitable NeSy tools and techniques that show potential. We note that this work is incomplete and should be seen as a starting point (see §6). Moreover, this section is speculative by nature, but we provide some evidence in terms of existing work and experiments conducted in §5 for selected use cases.

4.1. Monitor

The ability to integrate relevant knowledge into ML-based detection models (challenge 3) falls directly under the NeSy paradigm, and could both improve performance under real-world conditions (challenge 1) and help reduce the number of false alerts (challenge 5):

Use case 1. *Use (symbolic) knowledge of threats and assets to guide or constrain ML-based detection engines.*

A similar case for such a NeSy use case is made in [108]. *Logical Neural Networks (LNN)* [115] enables injecting knowledge about threats, vulnerabilities, or infrastructure into the neural architecture. Here, it both learns from data and considers cyber security knowledge. Similarly *Logic Tensor Networks (LTN)* [118], can learn from data while taking into account knowledge. Cyber security specific knowledge can be encoded as real logic rules, and help constrain the training of the neural detection engine. LTN has been studied to detect suspicious behaviour [14, 102] and is the topic of one of our experiments in §5. Knowledge about threats and assets can provide good indications of when (and even how) we can expect concept drift. Existing works on addressing concept drift in general [41], and tailored for network intrusion detection [6], are mainly based on identifying concept drift from the data or the model’s performance. Embedding knowledge about the expected change into an LNN or an LTN could help retraining models to be resilient to concept drift.

In challenge 2, we highlighted the need to learn from (relatively small) data sets, which is one of the key features of NeSy [48]:

Use case 2. *Learn detection models from a limited number of (labelled) incidents.*

Additional embedded knowledge in an LNN or LTN can help reduce the amount of required training data. *NS-CL*[86] has shown that it can be trained on a fraction of the data required by comparable methods – albeit in a different domain with different data sources. NeSy-based inductive logic programming variants, such as *∂ILP* [35], would also be able to learn from small data sets. The learnt logic program will also be inherently explainable (see challenge 6).

Threat hunting involves generating suitable hypotheses, applying and validating them, then updating and iterating (challenge 4). Work has started investigating LLMs for this challenge [107]. It has been argued for symbolism in LLMs [52], and based on that we define an LLM-based NeSy threat hunting use case:

Use case 3. *LLM-driven threat hunting using symbolic knowledge and reasoning capabilities.*

LLMs have been used for hypothesis generation in other domains [113], which can be further investigated for threat hunting. Similarly, LLMs used as a symbolic reasoner in text-based games [36] also seems interesting to port to such threat hunting.

Hypothesis generation is typically driven by CTI, which can be captured in a knowledge graph. The integration of LLMs and knowledge graphs is an active research field [76, 105]. In addition, symbolic or computational methods could be used for other steps in the hunting process, including: planning how to answer the hypothesis; reasoning about available data sources to execute this plan; ensuring correct translation to the required query language⁷ to validate the hypothesis using the observations; and finally, reason about the results from the execution and provide input for any refinement of the hypothesis for a new hunting iteration. Additionally, answer set programming techniques, such as dPASP [46], can leverage existing LLM-ASP integrations to perform threat hunting, thus utilizing both knowledge and reasoning [114].

4.2. Analyse

A prominent characteristic of NeSy is its ability to combine learning and reasoning. Such a combination is desirable in a SOC, and our next use case, which cuts across the monitor and analyse phases, addresses several of the challenges from §2:

⁷Events are stored in a SIEM system, which will have a query language.

Use case 4. *Incorporate learning of detection models with the ability to reason about their outcomes to understand and explain their nature and impact.*

In [108], the case for such integration of detection and analysis using NeSy is also made. One way to achieve this is to simultaneously train a neural network (for detection) with related symbolic rules that can be used for contextualisation, analysis, and explanation (challenge 6). Two NeSy techniques that can accomplish this are *Deep Symbolic Learning (DSL)* [29] and *Neuro-Symbolic Inductive Learner (NSIL)* [28]. LLM-based *Modular Reasoning, Knowledge and Language (MRKL)* systems [64], which consists of both neural and symbolic modules, provides a promising architecture for explaining the nature and impact of an outcome. Answer set programming-based neurosymbolic techniques like *dPASP* [46] and *NeurASP* [135] – and *DeepProbLog* [85] and *DeepStochLog* [133], which incorporates reasoning, probability and deep learning – also seem promising for this use case.

Use case 4 is rather generic and can be broken down into several smaller sub-cases. The first such sub-case is the extraction of symbolic alerts, in order to support alert contextualisation, analysis, and explanation:

Use case 5. *Extracting alerts in a symbolic form.*

In [55], symbolic alerts are extracted from a *graph neural network (GNN)* based detection engine. A combination of *GNNExplainer* [136] and *DL-Learner* [79] is used to extract symbolic alerts. The symbolic rules learnt by both DSL and NSIL may also provide such symbolic alert representation, and the use of e.g. *oILP* for detection will learn symbolic alerts by design. *Embed2Sym* [9] can be used to encode symbolic alerts by utilizing its ability to extract concepts from neural networks and assign a symbolic meaning, while LLM-based methods, such as *Logic.py* [69], have shown the ability to produce symbolic representations.

A SOC typically receives a large volume of threat intelligence, which is too large to thoroughly analyse manually. Such intelligence is used to contextualise alerts, and it is thus desirable to enrich the SOC's knowledge bases with relevant intelligence reports:

Use case 6. *Use statistical AI to enrich or extract symbolic knowledge.*

This use case addresses challenge 6. In §2, we discussed several approaches to extract knowledge in a suitable symbolic form from reports [80, 81, 87]. NeSy-based LLM approaches, such as *STAR* [114] and *Logic.py*, could also be used to extract knowledge in a suitable symbolic form amendable for reasoning.

This ability to reason is crucial as the intelligence report may be incorrect or superseded for different reasons, including underlying (aleatoric) uncertainty, deterioration over time, or they may come from sources one does not fully trust. It may also simply not be relevant for our purposes, or more importantly, intelligence reports may conflict with our existing knowledge or our own observations. It is therefore desirable to have the ability to quantify and reason about knowledge, including the level of trust, from both our own observations and existing knowledge:

Use case 7. *Reason about and quantify knowledge.*

This use case aims to address challenge 7. It may play a role in the implementation of a technique known as *risk-based alerting* [125], which involves using data analysis to determine the potential severity and impact of alerts and incidents. *Probabilistic attack graphs* [50] has been used to add probabilities to CTI to support such quantification. One potential NeSy approach for this use case is *Recurrent Reasoning Networks (RRNs)* [56]. RRNs could be used to train an ML model from observations to reason about our existing knowledge graph, e.g., to quantify or identify inconsistencies. Another NeSy example is *Neural Probabilistic Soft Logic (NeuPSL)* [111], as the output from the neural networks is in *Probabilistic Soft Logic (PSL)* [11], which can be treated by probabilistic graphical models. *NeurASP*, *dPASP*, *DeepProbLog*, and *DeepStochLog* also seem applicable here.

As discussed in §2, a cyber attack conducted by an advanced adversary will consist of multiple phases. The ability to relate these phases is essential when developing cyber situational awareness (challenge 6):

Use case 8. *Relate the different phases of cyber incidents.*

One concrete NeSy use case would be to merge the statistics- and semantics-driven approaches outlined in [7]. Further, *PyReason* [2] enables temporal reasoning over graphical structures, such as knowledge graphs, and can be used to exploit the temporal aspect of relating the different phases. The second experiment in §5 addresses this use case by exploring temporal reasoning using a combination of LLMs, temporal logic, ASP and plan recognition. The ontological reasoning supported by RRNs also seems promising for this type of problem.

4.3. Plan & execute

Neurosymbolic reinforcement learning (NeuroRL) [1] combines the respective advantages of reinforcement learning and AI planning. NeuroRL can learn with fewer interactions compared to traditional RL by using inherent knowledge. This ability makes it more applicable than both RL and AI planning when (near) real-time response is required and a complete model of the environment is infeasible. Moreover, it has the promise of more explainable response actions, whilst a reasoning engine could, in principle, help to take into account both risk and impact⁸. Thus, this seems like a promising approach for challenge 8:

Use case 9. *Generating impact and risk-aware explainable response actions in a timely fashion using neurosymbolic reinforcement learning.*

Symbolic Deep Reinforcement Learning (SDRL) [84] is directly applicable to this use case. Other neurosymbolic reinforcement learning techniques have been used in offensive cyber security settings for penetration testing⁹ [31]. Whilst there are some commonalities with our challenges, defending has their own peculiarities. For example, speed, risk, impact, and explainability are more prominent when defending against cyber attacks.

4.4. Shared knowledge

A widely applied form of symbolic AI in the context of cyber security is in semantic ontologies. Ontologies provide a formal and structured way of representing knowledge that both humans and machines can interpret while accounting for interoperability across systems. Built upon symbolic AI principles, ontologies focus on knowledge representation, logic, and reasoning, using well-defined structured models of the world. They define concepts, their composition, and their relationships within a domain, and they provide a clear distinction between the data (and the information itself) and the underlying model that defines how that information is organised, represented, and processed. This paradigm enables model evolution without data disruption, a known limitation of traditional relational models and other data serialisation formats that inherently combine representations and data elements. This powerful paradigm allows for a more seamless integration of federated and siloed (in too many cases heterogeneous) data and can provide ensembles of contextual knowledge graphs in support of answering complex questions for decision-making. In addition, ontologies are the backbone of a knowledge base that can guide learning, ensure consistency, facilitate inference, and provide explainability, making neurosymbolic systems more capable of handling real-world, knowledge-intensive tasks. Our final use case directly addresses challenge 9. Cyber threat intelligence is commonly shared in both structured and unstructured forms. LLMs are extensively studied for generating reports and this is also the case for cyber defence [95]. It is important that the information generated is accurate, something [76] can help with. The generation process is likely to use symbolism (e.g., knowledge graphs [105]). The reports need to be correct, which is an area in which symbolic AI can help [52]. We, therefore, rephrase challenge 9 as a NeSy use case:

Use case 10. *Generation of incident reports and CTI reports tailored for a given audience and/or formal requirements, using (symbolic) knowledge and LLMs.*

4.5. Summary

We have outlined ten different uses of NeSy that can address the challenges outlined in §2, and identified promising NeSy techniques that can serve as a starting point. Table 1 summarises the relationship between these use cases and the underlying challenges from §2. In addition, we indicate which use case and challenge each of the experiments in §5 addresses.

⁸We note that there are additional challenges when generating risk and impact-aware responses, such as both deriving the requirements in the first place and representing them in a suitable way.

⁹Penetration tests are simulated attacks against the infrastructure and assets being protected, for instance, to identify vulnerabilities.

	Use case 1	Use case 2	Use case 3	Use case 4	Use case 5	Use case 6	Use case 7	Use case 8	Use case 9	Use case 10
Challenge 1	✓ E1									
Challenge 2		✓								
Challenge 3	✓ E1									
Challenge 4			✓ E4							
Challenge 5				✓ E3	✓ E3					
Challenge 6				✓ E3	✓ E3	✓ E5		✓ E2,3,5		
Challenge 7							✓ E3			
Challenge 8									✓	
Challenge 9										✓

Table 1

Relationship between challenges, use cases and conducted experiment. '✓' indicates that a given challenge is addressed by the given use case, while 'EN' indicates that the challenge/use case is addressed by experiment N.

5. Proof-of-concept experiments

This section provides experimental evidence for our hypothesis that a SOC is an ideal environment for studying neurosymbolic approaches. The selection criteria we have used for the experiments is a combination of covering a broad set of challenges and use cases, as seen in table 1, and that is sufficiently mature and feasible to conduct within our time frame. A consequence of the latter criteria is that the experiments only cover the monitor and analyse phases of MAPE-K, as we believe we find the most mature NeSy approaches there. We also note that our approaches should be seen as proof-of-concepts, and are far away from being in a state that can be used in a operational setting in a SOC. We have conducted the following five experiments:

- In Experiment 1 (§5.1), we address use case 1 (*using knowledge of threats and assets to guide ML-based detection engines.*) This is shown by using Logic Tensor Networks (LTN) [12] to illustrate how cyber security knowledge in symbolic form can be used to improve an ML-based detection engine as well as improving explainability.
- In Experiment 2 (§5.2), we address use case 8 (*relating different phases of cyber incidents*). Here, LLMs and ASP are used to elicit and reason about adversary attack patterns and observed alerts for situational awareness.
- In Experiment 3 (§5.3), we address use case 4 (*learning detection models with the ability to reason about their outcomes*). This experiment also addresses elements from use case 5, use case 7 and use case 8. Here, a NeSy solution based on the Embed2Sym [9] approach is explored to contextualise alerts. That is, we use ASP and formalised domain knowledge to label clusters of embeddings according to what cyber kill chain phase they are likely to represent.
- In Experiment 4 (§5.4), we address use case 3 (*LLM-driven threat hunting with symbolic knowledge and reasoning*). Here, we build on [23, 24] by exploring the integration of LLMs with a symbolic approach based on knowledge graphs for threat hunting.
- In Experiment 5 (§5.5), we address use case 6 (*use statistical AI to enrich or extract symbolic knowledge*). Additionally, some elements of use case 8 is addressed. Here, we extend our previous work using data-driven enrichment of (symbolic) knowledge [123] with experiments using newly released data and explore the advantages NeSy provides for this challenge.

5.1. Experiment 1: LTN for knowledge-aware intrusion detection

ML-based intrusion detection systems need to learn how to correlate data and their classes¹⁰, capturing both simple and complex relationships. However, information that is not present or prevalent in the data might not be

¹⁰Classes could, for instance, represent malicious or benign data – or specific attack steps or attack techniques. This depends on the type of classifier.

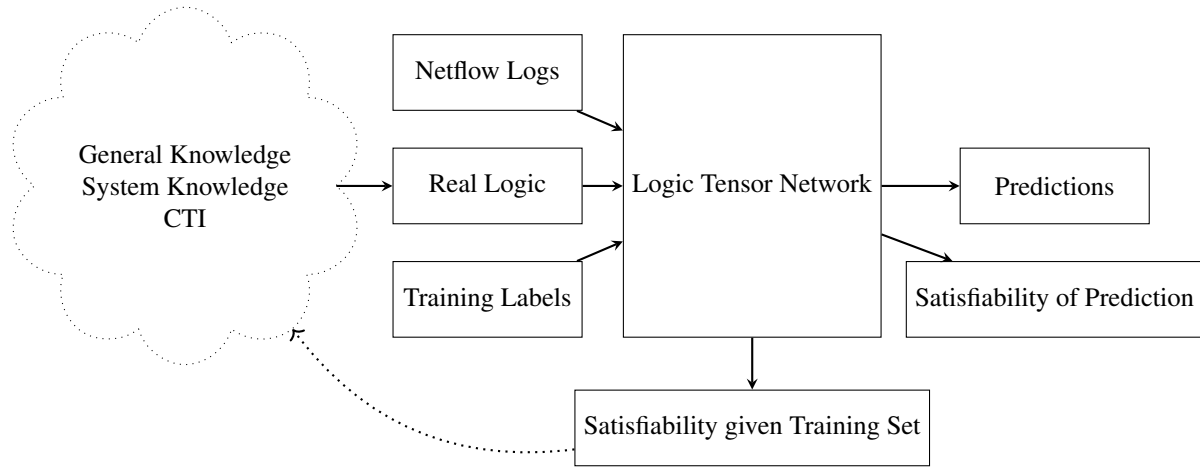


Fig. 1. Overview of LTN-based approach

used, even if it is obvious to an analyst. For this reason, we use a Logic Tensor Network (LTN) [12] to learn from data while being guided by expressed knowledge. Most people intuitively know that a vulnerability in Microsoft Word is not a danger for machines without the software installed. Neural networks, on the other hand, need to learn this by seeing it repeatedly in training data. Common sense knowledge, such as this, can easily be expressed as logic statements, which are used to help guide the learning of the LTN model. In detection engineering, analysts often have information they use to support the detection process that is not expressed explicitly in the logs used by the detection engine. This information can come from the knowledge or experience of analysts or other sources of information, such as cyber threat intelligence reports.

This experiment addresses use case 1 and is placed in the monitor phase of MAPE-K. Here, the goal is to detect malicious traffic by training LTN-based classifiers to detect two types of malicious traffic: brute force attacks and cross-site scripting (XSS) attacks. A *brute force attack* is a trial-and-error approach that, for instance, tries to guess the correct password, while *cross-site scripting (XSS)* attacks essentially inject malicious code into webpages.

We train two LTN-based classifiers: one classifier that separates brute force attacks from benign traffic and one classifier that separates XSS attacks from benign traffic.¹¹ Both classifiers use aggregated traffic in the form of *NetFlow* entries [26]. A *NetFlow* entry contains information about traffic between two distinct ports on distinct IP addresses for a given protocol within a given time frame (which may vary). It will typically contain information on the number of packets and the amount of data transferred, in addition to a wide range of other features. For our experiment, we used more than 80 different features extracted from the *NetFlows*.

The LTN-based intrusion detector will generate an alert if a *NetFlow* entry is classified as brute force or XSS. This alert will typically be manually inspected by a SOC analyst – or, as we will see in other experiments below – further enriched by e.g. other NeSy approaches.

Figure 1 shows an overview of the approach. As the learning is supervised, the model takes the ground truth label for each *NetFlow* entry as input in addition to the *NetFlow* entries themselves. The main difference from a standard fully connected neural network is that we encode and provide knowledge as *real logic* statements [118]. This could, for instance, be general knowledge, knowledge about the systems being protected, or cyber threat intelligence (CTI) about the threat we are trying to detect. Real logic is a fully differentiable first-order fuzzy logical language, supporting connectives and quantifiers [12]. This enables expressing knowledge that is hard or even impossible to express by purely adding extra information to the data points.¹² The statements are used in the classifier’s symbolic part, while the labels and *NetFlows* are used in its neural part.

¹¹We consider training two binary classifiers more realistic than a multi-class approach for this particular problem. However, note that we also did train a single multi-class network with comparable results.

¹²As an example, we can express that “all privilege escalation attacks are preceded by an initial access” by the following statement:
 $\forall x : \exists y : \text{privEsc}(x) \implies (\text{initAccess}(y) \wedge \text{before}(y, x))$

The experiments seek to answer the following research question:

Will a classifier enriched with knowledge perform better and provide better insight into what has been learnt than a purely data-driven classifier?

The experiment consists of two parts: In the first part, a three-layered fully connected neural network is trained and used as a baseline. In the second part, a LTN with the same underlying neural network structure is enriched with additional knowledge [12]. In both cases, 70% of the data is used for training and 30% for testing. The experiment is inspired by [102], where LTN is used in a similar fashion to distinguish benign NetFlows from multiple classes of attack NetFlows. Our experiment takes this further by also encoding cyber security knowledge into the LTN and comparing a classifier with and without the knowledge¹³.

We use the CICIDS2017 data set [119] for our experiments. This data set simulates benign traffic and attacks over five days, varying the attacks performed each day. In our experiment, we use the subset called "Tuesday morning". The labelled flows are categorized into three classes: "Benign", "Web Attack – Brute Force" and "Web Attack - XSS". The classes are significantly imbalanced, with 168,000 flows in the benign class and 2,159 flows in the remaining classes. Such imbalance between benign traffic and attacks is common and will be significantly more imbalanced in real life. The data set is partitioned into a 30/70 split between a training set and a test set. To account for class imbalance during training, we undersample the benign class to match the number of samples from the attack classes and the benign class. We include all features from the NetFlows into the sample except for the properties *source IP*, *destination IP*, and *source port* as they will cause overfitting and not generalise well. We one-hot encode the *destination port* with the 22 most common ports being their own feature and the remaining ports regarded as "other ports". We also one-hot encode the *protocol* feature. All features are normalised using min-max normalisation. Each NetFlow is represented as a vector of length 92.

The LTN consists of one predicate for class membership:

$$P(x, class).$$

This predicate is configured as a fully connected multilayer perceptron with an architecture of 92 input features, two hidden layers of size 256, and an output layer of size one. ELU is used as the activation function of the hidden layers, while sigmoid is used for the output layer [27].

Real logic statements are used to shape the training of the neural network. The idea is that such statements should be created by a cyber security analyst and be based upon knowledge about the system, the current threat landscape, and any other relevant information the analyst has. Training consists of updating the neural network P to maximise the accumulated truth value of the axioms [12]. In this experiment, we first define the following axioms:

$$(1) \quad \forall x \in B : P(x, \text{Benign}) \quad (2) \quad \forall x \in BF : P(x, \text{Brute_force}) \quad (3) \quad \forall x \in X : P(x, \text{XSS})$$

The first three axioms describe how all flows in the training set that are labelled as a given class ($B : \text{benign}$, $BF : \text{brute force}$, $X : \text{xss}$), should be a member of that class. This encodes the information of the baseline neural network with no additional knowledge. We then define the following axioms:

$$(4) \quad \forall x \in NWS : \neg(P(x, \text{Brute_force}) \vee P(x, \text{XSS})) \quad (5) \quad \forall x \in IT : \neg(P(x, \text{Brute_force}) \vee P(x, \text{XSS}))$$

The fourth axiom describes how all NetFlows not going to or from a web server ($NWS: \text{Not Web server}$) cannot be a web attack. Both the XSS and brute force attack in this data set are for web servers specifically. The information about what is not a web server is knowledge extracted from the topology of the network we are tasked to defend (i.e., the victim of the attack). The fifth axiom defines all traffic between machines in the victim network ($IT: \text{Internal traffic}$) as not part of an attack. This is because we expect all XSS or brute force attacks to come from outside

¹³Similar work using LTN for intrusion detection has been done in [14]. Note that this work was published after our initial LTN experiments [49], which we extend here.

the organisation’s network. The knowledge encoded in axioms 4 and 5 is elementary common knowledge, and we expect domain experts to express more complex relationships. Still, it is sufficient for our work.

We trained one baseline model and one LTN model for each of the two attack classes. Both the baseline and LTN models use the same training and test sets, and have the same configuration of the underlying neural network. We trained both models over 80 epochs with a batch size of 250. The results are presented in table 2.

Labels	Baseline Neural Network			Logic Tensor Network		
	Precision	Recall	F1	Precision	Recall	F1
Brute Force	0.066	0.847	0.122	0.154	1.000	0.267
XSS	0.028	0.929	0.055	0.104	0.964	0.188

Table 2

Results from LTN Experiment.

The data set tries to reflect realistic data and is therefore highly unbalanced, with 98.7% NetFlows being normal benign traffic. As seen by challenge 5, alert flooding is a problem in a SOC with a need to balance high recall, i.e. a balance between too many false alerts and the potential of missing an attack. This will depend on external factors; e.g. we may have intelligence that indicates a specific type of attack is imminent, and high recall is therefore essential.

The results show that both solutions have high recall when distinguishing benign traffic from attacks. The precision for both solutions is fairly low; however, this is to be expected as the data set is unbalanced. Most importantly, we can see that the precision of the LTN classifier is more than double that of the baseline classifier (0.154 vs 0.066 for brute force attacks; 0.104 vs 0.028 for XSS), indicating that adding knowledge can improve classification.

As a comparison, we look at two related works using purely neural techniques to create an NIDS on similar data sets. MLP4NIDS [116] uses a multi-layer perceptron (MLP) to create a multi-class classifier on the CIC-IDS-2017 data set. Kim et al. [73] use a convolutional neural network as a multi-class classifier, testing it on the CSE-CIC-IDS 2018 data set. This paper only looks at a subset of CIC-IDS-2017 with only two attacks: XSS and brute force. Both works show good results overall. However, as both of them are trained on 16 attack types as opposed to two in the LTN classifier, we see that the result for classifying XSS and brute force attacks is comparable or worse in both cases. For MLP4NIDS, all XSS and brute force attacks are misclassified. In Kim et al., the F1 score for XSS is 0.65 and 0.0 for two parts of the data set, compared to 0.27 by the LTN. The F1 score for brute force attacks was 0.3 and 0.0 for the same parts compared to 0.19 for the LTN classifier. It is worth noting that the accuracy function defined by Kim et al. is actually the F1 score. In general, the LTN classifier outperforms or performs on par with [73, 116] for XSS and brute force attacks. However, the LTN classifier performs worse than the state-of-the-art classifiers on average over all classes. We reiterate that the key observation in this experiment is that when trained under the same conditions, a baseline neural classifier can be improved by the additional knowledge included using an LTN.

5.1.1. Learning insights

Real logic statements in a LTN are an effective way of injecting knowledge into a neural classifier. They can also help in understanding and influencing the model’s training and focus. Next we explore different ways the explainability aspects of LTN can be used by a SOC analyst.

During training of the LTN, the goal is to maximize the aggregated truth of all the provided statements. This is done by deriving the loss of the model from the aggregated truth. Real logic is fuzzy logic, and we would not expect that all statements hold for all cases. After the training is completed, one can analyse how well the rules hold on all the provided training data to provide insights into how the model works. This can also be used as feedback to the analyst to help change or tweak the rules. In figure 2(a), we plot the satisfiability of the five statements for the training set. Here, we can see that after training, rules four and five are generally satisfied. They are the rules that reduce false positives (false alerts) for the two attack classes. We can also see that the performance for the third rule, which classifies XSS, is significantly lower. This is in accordance with the results in table 2.

When creating real logic statements, there is a risk of creating a statement that does not accurately reflect the data. This can be the result of errors made when defining rules or due to incorrect intuitions. For example, a rule asserting that only computers are targets of attacks does not accurately capture reality, as mobile phones are also targets. If a

bad rule is introduced, we would expect that the LTN would have a hard time satisfying this rule at the same time as the other rules. We can therefore use the low satisfiability of a rule (after training) as an indication that there is a problem with the rule. The fact that the LTN was not able to find a way to make the rule true is an indication that it does not describe the data correctly. To demonstrate this, we conduct a small experiment where we introduce a new rule that is obviously not true stating that all traffic labelled as benign should be classified as brute force:

$$(6) \forall x \in B(P(x \in B, Brute_force)).$$

After training an LTN with this incorrect rule, we can see in figure 2(b) that the satisfiability of the false rule is significantly lower than the other rules.

In addition to expressing rules, an analyst may be interested in expressing the relative importance of different rules. For example, a rule relating to a rare attack with limited consequences may be given a low priority. Conversely, a rule expressing something very prevalent and critical may be prioritised. To reflect this, we add weights to the rules to give a simple way of assigning the importance of a rule compared to the others, where a high weight results in the statement contributing more to the total aggregated truth.

To provide additional insights, the analyst can investigate the satisfiability of the different rules for a given NetFlow. For the majority of NetFlows, we expect all rules to hold, as this is what the LTN is optimising for. However, if some rules are not satisfied, we could pass the information on to an analyst to provide additional explanation and context for their analysis.

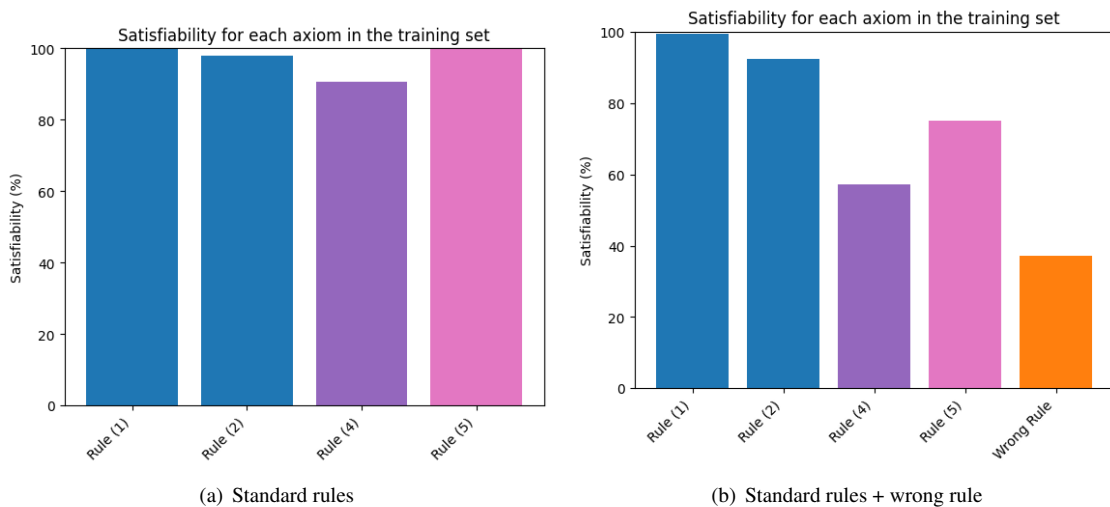


Fig. 2. Average satisfiability on training set (after training)

To summarise, this experiment illustrates the potential of using NeSy to embed additional knowledge into ML models to detect suspicious behaviour. We also see how LTNs can help analysts understand what the model learns and how it predicts. We have showcased that a neural classifier can be improved by adding knowledge in the form of real logic statements. When using the knowledge-enriched LTN, the number of false alerts was reduced without impacting recall. The LTN also provides multiple techniques to improve the model's explainability. By examining the satisfiability of the statements after training, we can gain valuable insight into what the model has learnt or what it is not able to learn. With this information, we can change and improve the defined statements. The satisfiability can also be used to gain insight into the model's predictions. This shows promise for using NeSy to enrich ML-based models with (symbolic) knowledge.

5.2. Experiment 2: LLMs and ASP for situational awareness

Experiment 1 provided an example of how alerts can be raised. In §2 we have also discussed the need for supporting alert analysis. In this experiment, we demonstrate such analysis by illustrating the use of NeSy to relate different phases of an attack (use case 8). Here, alerts sequenced by time are mapped to adversary attack patterns, gleaned from textual CTI reports into symbolic form using statistical methods (use case 6). The experiment is inspired by existing work such as: neurosymbolic plan recognition [4], attack plan recognition [5], and the use of LLMs to extract both *linear temporal logic* (LTL) [39] and CTI (in the form of MITRE ATT&CK *tactics* or *techniques*)¹⁴ [53, 103, 137].

An LLM is first used to elicit formal representations of attack patterns described in CTI reports, affording us a rapid way to convert CTI to symbolic knowledge. Here, we use the NL2LTL Python library [39] to extract representations of attack patterns in LTL_f [30], a temporal logic for finite traces. We are using Open AI's GPT-4 model with few-shot learning. We define a custom pattern template *ExistenceEventuallyOther* to express the LTL property $\diamond a \wedge \circ \diamond b$. Each prompt parses two lines from the attack description and finds the appropriate ATT&CK technique (from the allowed symbols), and an LTL formula. We combine the formulas from each prompt into one long LTL formula expressing the entire attack pattern. The following is one prompt generated by NL2LTL:

Prompt generated by NL2LTL

```
Translate natural language sentences into patterns:
ALLOWED_PATTERNS: ExistenceEventuallyOther
ALLOWED_SYMBOLS: T1548 (Abuse Elevation Control Mechanism), T1552 (Unsecured Credentials),
T1133 (External Remote Service), T1059 (Command and Scripting Interpreter ) [...]

## Few shot learning example
NL: The adversary logs into the Kubernetes console.
This leads to: The adversary can view plaintext AWS keys in the Kubernetes console.
PATTERN: ExistenceEventuallyOther
SYMBOLS: T1133, T1552

## Query
NL: Attackers used a known exploit to allow the loading of a DLL that facilitated altering SAML-rules
This Leads to: Attackers modifies SAML-rules allowing lateral movement to occur.
```

A conceptual adversary attack pattern, sequencing MITRE ATT&CK techniques is visualised in Figure 3.

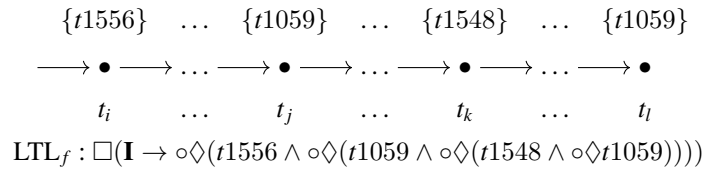


Fig. 3. Adversary attack pattern

Each ' t_{xxx} ', where x is a number, is a unique technique from the ATT&CK framework, \mathbf{I} is the initial state, and \square , \circ and \diamond are the 'always', 'next' and 'eventually' operators in LTL. Next, *telingo* [21] is used to postdict possible attacks. *telingo* is a temporal ASP solver implementing *temporal equilibrium logic* for *finite traces* (TEL_f) which

¹⁴A MITRE ATT&CK *tactic* describes why an adversary performs an action, while a MITRE ATT&CK *technique* describes how the action is performed [94].

is syntactically similar to LTL_f but semantically slightly weaker [20]. The `telingo` program encoding the problem is shown in figure 5. The program details the sequences of observed alerts (lines 5–11), the LTL_f representations of known attack patterns (lines 14–21), and the relationships between alerts and ATT&CK techniques (lines 25–28).

The attack patterns in the program are acquired by the elicitation step described above, and the sequences of observed alerts are assumed to come from a SIEM system. That is, the alerts produced are in a structured form amenable to be represented as Prolog/ASP terms. We assume that this conversion of alerts to symbolic form (use case 5) exists (see e.g. [55]). Furthermore, they are temporally ordered, inducing a sequence of alerts (where a_x is an alert in symbolic form), as shown in figure 4.

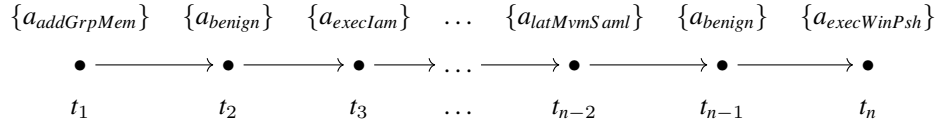


Fig. 4. Trace of alert observations

```

1  #program initial.
2  1 { plan(plan1;plan2) } 1 .
3
4  %Trace of alerts
5  &tel{&true
6      ;> alert(addGrpMem)
7      ;> alert(benign)
8      ;> alert(execIam)
9      ;> alert(latMvmSaml)
10     ;> alert(benign)
11     ;> alert(execWinPsh)}.
12
13 %Adversary attack plans
14 :- plan(plan1), not &tel{
15     >? techn(t1556)
16     & > (>? techn(t1059)
17     & > (>? techn(t1548)
18     & > (>? techn(t1059))))}.
19 :- plan(plan2), not &tel{
20     >? (techn(t1556)
21     & > (>? (techn(t1059) | techn(t1548))))}.
22
23 #program dynamic.
24 %abduce technique based on alert
25 1 {techn(t1556); techn(t1548)} 1 :- alert(addGrpMem) .
26 1 {techn(t1059)} 1 :- alert(execIam) .
27 1 {techn(t1548)} 1 :- alert(latMvmSaml) .
28 1 {techn(t1059)} 1 :- alert(execWinPsh) .

```

Fig. 5. Telingo program encoding the problem.

Finally, we assume that all the alerts produced can be associated with ATT&CK techniques, which is the case for many signature-based alerts. Note, however, that it is a many-to-many relationship: an alert can be an indicator for several techniques, and a technique can have several alert indicators. This knowledge¹⁵ can be represented in ASP

¹⁵Extracted from alert rules found at <https://github.com/SigmaHQ/sigma>.

with choice rules, as illustrated below:

```

1  {t1556; t1548} 1 ← aaddGrpMem
2
3  {t1059} ← aexecIam
4
5  {t1548} ← alatMvmSaml
6
7  {t1059} ← aexecWinPsh

```

The outcome of the program’s execution is that there are two stable models. This is shown in figure 5, and tells us that it is plausible that the input trace is an instance of the attack plan. Had there been no stable models, the conclusion would have been that this could not have been the case.

The feasibility of using the outlined approach in practice on real data is a matter which requires further study. However, we note that the LTL-satisfiability problem is PSPACE-COMplete and the TEL-satisfiability is EXPSPACE-COMplete [19] and hence both are considered intractable. Furthermore, the alert traces used in the experiment were much shorter than what can reasonably be expected, and the attack plan formulas are quite basic with respect to length and operator usage. This naive approach is unlikely to scale to realistic alert traces that can contain 100.000+ alerts. Thus, it would be worthwhile to look at: (1) less expressive approaches, such as metric temporal logic, that have been shown to be efficient and scalable in practice for large scale temporal event processing [130]; and (2) apply alert-reducing techniques such as filtering and alert grouping (i.e. merging similar alerts to a single instance) to shrink the alert traces into a manageable size.

5.3. Experiment 3: Neurosymbolic alert contextualisation

Deciding which alerts are important and require attention, and understanding how they belong in the bigger picture, is essential in a SOC. However, this requires contextualising alerts with knowledge, such as about the systems and networks in which the alerts were raised, cyber threat intelligence, and background knowledge accumulated by analysts over time. This is illustrated in figure 6, where the context allows an analyst to follow a continuous path through alerts and log events.

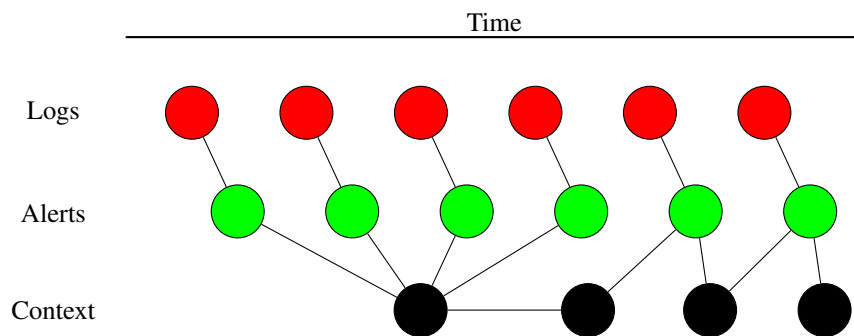


Fig. 6. Illustration of how context contributes to creating a continuous path through logs and alerts.

In this experiment, we assume the existence of rule-based and anomaly-based alerts, which differ widely in contextual richness, and we aim to classify these alerts by the cyber kill chain step to which they are likely to belong. The experiment mainly addresses use case 4, but also includes aspects of use cases 5, 7 and 8.

Rule-based alerts are generally contextually richer than anomaly-based alerts; the former are mostly hand-crafted and contextualised with descriptive knowledge as to what type of suspicious behaviour it detects (e.g. which MITRE ATT&CK technique the alert indicates), while the latter are generally more primitive alerts that flag any abnormal attribute values (that deviate from normally seen values during training). Thus, these are less descriptive about what behaviour is detected. Hence, it is, for example, easier to associate cyber kill chain phases with rule-based alerts than

with anomaly-based alerts.¹⁶ On the downside, contrary to anomaly-based alerting, rule-based alerting is unable to detect novel and previously unseen suspicious behaviour. Both types of alerts are thus useful for detecting cyber attacks.

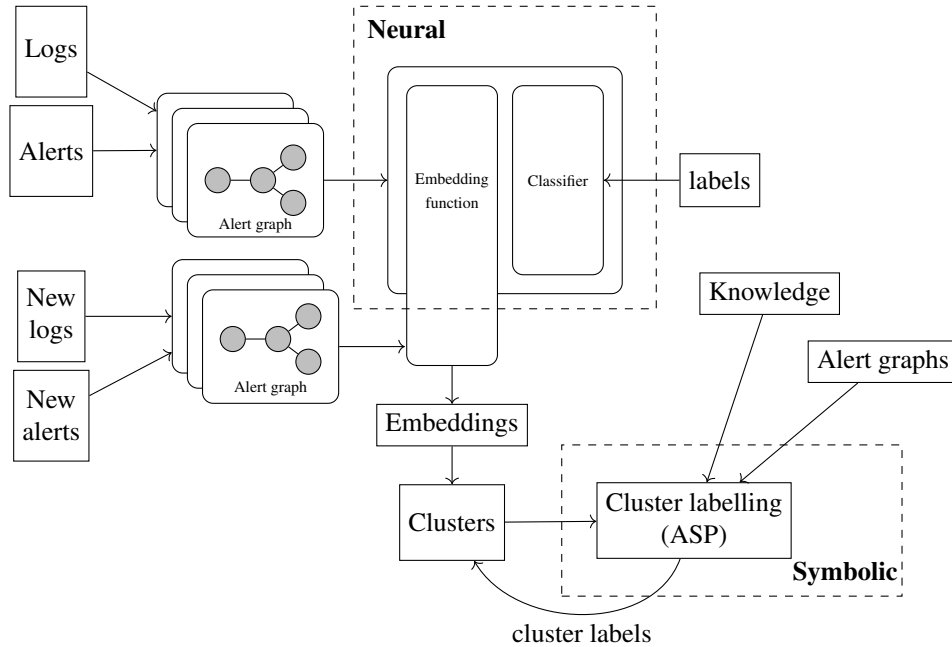


Fig. 7. Embed2Sym adapted to our experiment

For this experiment

we wish to classify alerts according to cyber kill chain steps, yet we have alerts with a highly varying degree of contextual richness on which to do so.

The approach we explore in this experiment is that of using ASP and formalised domain knowledge to label clusters of alert embeddings according to what cyber kill chain phase they are likely to represent. Here, the embeddings are created using a neural component and then clustered into groups. The effect of this is that the clusters are likely to contain both descriptive and non-descriptive alerts. The task we formalise in ASP is essentially an optimisation problem, where we use weak constraints to promote cluster labelling. Specifically, we encode the following two label assignment preferences:

- The assigned cluster labels (i.e. the predicted cyber kill chain phase of the alerts in the cluster) should comply with (any) domain knowledge about the detection rules that were the origin of the alerts in the respective cluster.

Example: the cluster that contains an alert generated by a rule that detects MITRE ATT&CK technique T1548 should ideally be labelled *Privilege Escalation*.

- When there are alerts in different clusters that share some context (e.g., same users, or overlapping source and destination addresses), then the cluster labels should be assigned in such a way that the temporal order of the alerts and the relative order of the cyber kill chain align.

Example: assume that $alert_1 \in cluster_1$ and $alert_2 \in cluster_2$, they both involve username *user*, and $alert_1$ happens before $alert_2$. Then $cluster_1$ should ideally be labelled with a cyber kill chain stage that occurs before the label assigned to $cluster_2$, e.g. *Lateral Movement* before *Privilege Escalation*.

¹⁶We note, however, that a detection rule can be (and often is) an indicator of more than one TTP (tactic, technique or procedure) or cyber kill chain phase. It is, therefore, non-trivial to uniquely identify it.

Turning to the technical details, we follow the approach presented in [9], referred to as *Embed2Sym*, where a neural perception and reasoning component is combined with a symbolic optimisation component to extract learnt latent concepts. The neural component is decomposed into two functions: a perception and a reasoning function. The latter function (reasoning) is designed to solve a downstream task, whereas the perception function creates vector embeddings of the input data. By solving the downstream task, the reasoning stage discovers structure in the data relevant to the domain. This is fed back to the perception function, influencing the vector embeddings. Finally, the embeddings are clustered, and symbolic optimisation using ASP is used to label the clusters according to the latent concepts.

This experiment is built upon logs and alerts from the data set described in [77] and [78]. We are mainly interested in alerts, yet most of the contextual information remains in the log messages; hence we need the latter as well in order to adequately contextualise alerts. The logs and alerts are collected from a testbed emulating a small enterprise where a multi-step attack is being performed. The data also includes ground truth, making it possible to see exactly where in the logs and alerts the attack is captured, and also what hostile activity gave rise to each of these log lines and alerts.

We transform this data into graph form, projecting descriptive features that can be extracted from log messages onto the alerts they are associated with. We end up with alert graphs, that includes nodes representing other objects such as network resources, MITRE ATT&CK techniques, detection rules etc. These can be extracted from the log or alert information and can be used to link alerts through paths in the graph. The latent concepts of interest are the cyber kill chain stages. The cyber kill chain stages in the experiment are based upon the stages used in the attack described in [77]:

1. Reconnaissance
2. Initial intrusion
3. Obtain credentials
4. Privilege escalation
5. Lateral movement

Our instantiation of *Embed2Sym* is shown in figure 7. The downstream task for the *Embed2Sym* reasoning function is in this case to classify alerts according to hostile activity, utilising the ground truth labelling in the data. This leads to a perception function whose output is an embedding function influenced by context learnt by the reasoning function.

For the next step in the process, we consider the set of alerts that we wish to analyse. Using the embedding function, these alerts are transferred into the embedding space, where they are clustered. The intuition behind this step is that the downstream task of classifying the alerts according to the actual hostile activity should force the embeddings of alerts from the same stages of the attack closer together in the vector space.

The final step is to apply symbolic reasoning, utilising the alert graph and formalised domain knowledge, to label the clusters accordingly. The task is encoded as an ASP program, shown in figure 9, and the clustered alerts are represented as ASP facts, as shown in figure 8.

```
1 alert_in_cluster(alert1, cluster1).
2 happensAt(alert1, 100000).
3 dst_host(alert1, "10.0.0.1").
4
5 detected(alert1, "T1000").
6 src_user(alert1, "John").
7 dst_user(alert1, "Jack").
8 src_host(alert1, "192.168.0.10").
```

Fig. 8. ASP instance encoding

Starting with the encoding of the instance data, lines 1-3 of figure 8 encode that an alert belongs to a cluster, that it happened at a certain unix epoch time, and the alerted event occurred on a specific host, respectively. Line 5 encodes that the alert was generated by a rule that detects instances of MITRE ATT&CK tactic T1000, while lines

```

1 1 % cyber kill chain / TTP domain knowledge
2 2 kill_chain_phase(latMvmt,5). kill_chain_phase(privEsc,4). kill_chain_phase(obtainCreds, 3).
3 3 kill_chain_phase(initialIntrusion,2). kill_chain_phase(recon,1).
4 4 kc_before(P, Ptac) :- kill_chain_phase(P,S), kill_chain_phase(Ptac,Stac), S < Stac.
5 5
6 6 phase_ttp_map(latMvmt, ("T1021.004";"T1021")).
7 7 phase_ttp_map(privEsc, ("T1548.003";"T1055";"T1078")).
8 8 phase_ttp_map(recon, ("T1083";"T1595.002")).
9 9 phase_ttp_map(obtainCreds, ("T1110";"T1110.001")).
10 10 phase_ttp_map(initialIntrusion, ("T1190";"T1078")).
11 11
12 12 % temporal rule
13 13 happensBefore(Alert, AlertTac, before(T1,T2)) :- happensAt(AlertTac, T2), happensAt(Alert, T1), T1 < T2.
14 14
15 15 % rules establishing feature-based links
16 16 %% 1. if on same host, then two alerts must a) share users OR b) share connection
17 17 %% 1.b
18 18 common_feature(Alert1, Alert2, same_source_dest(Dest,Source)) :- dst_host(Alert1, Dest), dst_host(Alert2, Dest),
19 19 src_host(Alert1,Source), src_host(Alert2, Source), Alert1 != Alert2.
20 20 %% 1.a
21 21 common_feature(Alert1, Alert2, same_source_user(Dest,User)) :- dst_host(Alert1, Dest), dst_host(Alert2, Dest),
22 22 dst_user(Alert1,User), dst_user(Alert2, User), Alert1 != Alert2.
23 23 common_feature(Alert1, Alert2, same_source_user(Dest,User)) :- dst_host(Alert1, Dest), dst_host(Alert2, Dest),
24 24 dst_user(Alert1,User), src_user(Alert2, User), Alert1 != Alert2.
25 25 %% 2. if on different hosts, must be src->dst
26 26 common_feature(Alert1, Alert2, source_to_dest(Dest)) :- dst_host(Alert1, Dest), src_host(Alert2, Dest), Alert1 != Alert2.
27 27 %% symmetric
28 28 common_feature(Alert1, Alert2, Reason) :- common_feature(Alert2, Alert1, Reason). % symmetric
29 29
30 30 % Cluster labeling
31 31 cluster(X) :- alert_in_cluster(_,X).
32 32 label(X) :- kill_chain_phase(X,_).
33 33 label(X) :- benign_class(X).
34 34 benign_class(benign).
35 35
36 36 %% Assign label to each cluster.
37 37 {assigned_label(Cluster, Label) : label(Label)} = 1 :- cluster(Cluster).
38 38 :- assigned_label(Cluster1, Label), assigned_label(Cluster2, Label), Cluster1 != Cluster2.
39 39 alert_label(Alert,Label) :- alert_in_cluster(Alert,ClusterID), assigned_label(ClusterID,Label).
40 40
41 41 % Optimization constraints
42 42 %% Constraint a)
43 43 label_ttp_mismatch(Alert,Label,T) :- detected(Alert,T), alert_label(Alert, Label), not phase_ttp_map(Label,T).
44 44 :- label_ttp_mismatch(Alert,Label,T). [1@2,Alert,Label,T]
45 45
46 46 %
47 47 %% Constraint b)
48 48 kill_chain_sequence_mismatch(S,Stac) :- common_feature(Alert, AlertTac, Just), kc_before(S,Stac), alert_label(Alert,S),
49 49 alert_label(AlertTac,Stac), S != Stac, not happensBefore(Alert,AlertTac,_).
50 50 :- kill_chain_sequence_mismatch(S,Stac). [1@1,S,Stac]
51 51
52 52 #show assigned_label/2.

```

Fig. 9. ASP program encoding the problem.

6-8 encode the IP-address that initiated the event that led to the alert, the username associated with the event, and the username that originally initiated the event (e.g. a *su* operation), respectively. Lines 1-3 encode facts that all alerts will contain, while lines 5-8 depend on the underlying alert event type (e.g., network traffic or process execution).

Proceeding to the encoding of the task itself (figure 9), the first part establishes some basic cyber kill chain and TTP domain knowledge. That is, lines 2-4 define the cyber kill chain phases, and their relative ordering in the chain, while lines 6-10 map MITRE ATT&CK techniques to the cyber kill chain phases used in our experiment.

The next part introduces rules pertaining to the order and shared features of alerted events. The rule in line 13 captures the temporal order of alerted events, while lines 13-28 capture shared features between alerts, such as sharing users, source and destination addresses, etc.

The following part, shown in lines 31-34, deals with defining what constitutes labels and clusters, while lines 37-38 are responsible for allocating labels. That is, the choice rule in line 37 ensures that each cluster is assigned a label, and line 38 ensures that each cluster is only assigned a single label. Line 39 is a convenience rule that in practice classifies an alert based on the assigned label of the cluster it belongs to. Finally, lines 43-50 capture the two weak constraints that encode the optimisation tasks described in the beginning of this section.

For the experimental run itself, we clustered 1900 alerts from the data set into six clusters (number of cyber kill chain steps plus one 'benign'). Of these 1900 alerts, 290 had information that associated them with MITRE

```

1  [...]
2  Answer: 6
3  assigned_label("recon",recon) assigned_label("benign",benign) assigned_label("privilege_escalation",privEsc)
4  assigned_label("cracking",obtainCreds) assigned_label("webshell",initialIntrusion) assigned_label("reverse_shell",latMvmt)
5  Optimization: 0 1
6  OPTIMUM FOUND
7
8  Models      : 6
9    Optimum   : yes
10 Optimization: 0 1
11 Calls       : 1
12 Time        : 96.698s (Solving: 0.01s 1st Model: 0.00s Unsat: 0.00s)
13 CPU Time    : 96.659s

```

Fig. 10. Clingo console output

ATT&CK tactics. For convenience, we identified the clusters with meaningful names in order to make validation easier (e.g. cluster2 was named "webshell"). We then ran the ASP encoding of the task and instance data through the `clingo` ASP grounder and solver [45], which was able to correctly classify the labels, as shown in figure 10.

Although this experiment was limited to detecting one specific instantiation of a cyber kill chain within a generated but realistic data set, we believe that the results indicate that the approach is feasible for classifying alerts according to cyber kill chain steps even when contextual information regarding the alerts varies widely. We note, however, that deciding if a model is stable and optimal for a disjunctive ASP program with optimization statements is $CO-NP^{NP}$ -COMPLETE in the grounded propositional case and thus well in the realm of infeasibility [44]. Similarly to the `telingo`-based experiment 2, this approach is unlikely to scale to realistic alert traces that can contain 100.000+ alerts, hence it would be worthwhile to look at applying alert-reducing techniques such as filtering and alert grouping.

5.4. Experiment 4: NeSy-driven threat hunting

The previous experiments have focused on intrusion detection and subsequent analysis of the raised alerts. This experiment focuses on a different approach to discovering malicious behaviour called *threat hunting* (see challenge 4). The experiment addresses use case 3 and explores the efficacy of leveraging LLMs to develop a taxonomy of behavioural indicators for the (symbolic) *indicators of behavior* (IOB) approach to threat hunting [23, 24]. This symbolic threat hunting approach utilises an ontology and semantic reasoning to infer a set of contextualised adversarial behaviors across a series of logged security event data. Whilst this IOB-concept has previously been demonstrated [23, 24], it lacks a taxonomy of behaviours and reusable IOB identifiers. Since the IOB knowledge base is an emerging concept, it requires continuous additions to its knowledge base to increase maturity. We explore the efficacies of LLMs to aid in the semi-automated development of the IOB taxonomy and knowledge base.

Natural language processing techniques have been utilised for extracting Indicators of Compromise (IOCs) from CTI Reports [83] and, more recently, extended to the utilisation of LLMs for extracting IOCs from CTI reports [129]. Here, we explore the implementation of LLMs for developing an IOB taxonomy. The utilisation of LLMs for generating detection logic from a conceptualised task has been demonstrated in industry [120]. Motivated by this work, we explore the automated development of semantic rule-based reasoning into our taxonomy in this experiment.

Threat hunting involves the generation of suitable hypotheses, followed by applying and then validating the hypotheses (see challenge 4). This experiment uses a scenario-driven approach to IOB development, where the scenario is a hypothesis describing what an adversary, tool or general user is trying to achieve. For a given scenario, the LLM is tasked with generating a set of low-level behavioural indicators that analyse syntax, commands and other properties at a low-level of details and reason over these indicators to infer a higher level of abstraction.

A simple example of such a scenario is `notepad.exe` being launched by the Windows built in *system user*¹⁷ and then establishing a network connection. This example is a set of low-level behaviours. By combining these low

¹⁷<https://learn.microsoft.com/en-us/windows/security/identity-protection/access-control/local-accounts>

level behaviours, a higher level of abstraction can be inferred as an interactive tool being launched by the system user for the purpose of networking. We can chain this behaviour together with other inferred behaviours to get an even higher level of abstraction.

To semi-automate this scenario-driven approach to developing an IOB taxonomy, we require an LLM that can:

- Contextualise elements of the cyber security domain.
- Contextualise how adversaries behave.
- Generate behavioural scenarios and transform these scenarios into a chain of events.
- Contextualise how tools, systems or programs operate in a given scenario.
- Generate a set of reusable IOB identifiers.
- Relate low-level security events together to form a higher level of abstraction and context.
- Transform an IOB scenario into a symbolic representation.
- Transform the detection logic for low-level events into semantic web rules (SWRL)¹⁸.
- Be both granular and descriptive to provide context commonly missing from MITRE ATT&CK technique procedures [24].

These requirements are a mix of concepts and subject areas, where tailored LLMs may struggle to fulfill some requirements and excel in others. As a results of this, we primarily focus on general-purpose LLMs rather than purpose-built LLMs. We compare the following models for their ability to semi-automate the development of an IOB taxonomy:

- GPT4-Omni
- GPT3.5-Turbo
- Llama 3.2-3b¹⁹
- SecurityLLM²⁰

The first three models are general purpose, while the last model is purpose-built for the cyber security domain. Various GPT-models have been used in the cyber security domain for a variety of purposes[95], including payload generation for offensive security tasks, leveraging knowledge from the MITRE ATT&CK framework and detection engineering. Llama has been available for research [91], and SecurityLLM is based on Llama with the intent to provide cyber security guidance [139], including threat hunting, cyber kill chains and MITRE ATT&CK. Both Llama 3.2-3b and SecurityLLM were used offline for this experiment.

The scale of symbolic security event data makes it inefficient to process line-by-line via an LLM, due to context size and memory limits. To illustrate, the symbolic event data file generated from a single node in an emulated attack scenario used for our experiment contains 490, 248 lines, and 24, 161, 441 characters. Instead, we leverage the LLM to generate IOB scenarios with relevant rule based detection logic, which utilise a reasoning engine for decision making. We define a set of constraints through prompt instruction and by embedding tasks within the message to the model to generate these scenarios.

LLMs perform more accurately when prompts utilise chain-of-thought prompting [131] and least-to-most prompting [138], which we combine in a hybrid approach. Chain-of-thought prompting is used to create a set of behaviors, an abstract definition, a summary, detection logic and the semantics are different concepts. Least-to-most prompting is used to create sub-tasks for the model to maintain accuracy and reduce the risk of LLM hallucination. Each sub-task has an updated prompt instruction set. Each model is tested without a system prompt. The GPT-models are non-configurable, hosted by the OpenAI, and operated in the web browser. Therefore, no configurations are available to share. Llama and SecurityLLM are open and their configurations are included during assessment.

Each model has two tasks:

1. Create the IOB scenario.
2. Create the symbolic representation.

¹⁸<https://www.w3.org/submissions/SWRL/>

¹⁹<https://huggingface.co/meta-llama/Llama-3.2-3B>

²⁰<https://huggingface.co/ZySec-AI/SecurityLLM>

(A) Prompt instruction for creating an IOB scenario

I want you to use a scenario-based threat hunting method for high-level abstractions.
 All scenarios are post-compromise scenarios. The threat actor has access to the network.
 Atomic level / Low level behaviors require a detection analytic.
 A detection analytic analyses the data logged by Sysmon in the Sysmon schema.
 Focus on the context that commands, parameters and syntax are trying to achieve.
 Focus on the context of these commands, parameters and syntax as a chain of events.
 Focus on Windows enterprise environments.
 Provide an extensive taxonomy rather than a few examples.
 Behaviors should focus on common behaviors associated with adversaries.
 Summarise with a table at the end.

(B) Task instruction for creating an IOB scenario

I want to create a taxonomy of adversarial behavioral indicators. Do not repeat the MITRE ATT&CK framework, I want to compliment it with more detailed levels of abstraction. This is a more detailed level of abstraction of what an adversary is trying to achieve over a chain of events.
 For this task focus on behaviours related to command prompt and how and adversary can leverage command prompt for its goals. Focus on the relation command prompt has with other aspects of adversarial behaviours.
 This taxonomy and the behaviors is a tiered approach where very low level atomic behaviors are sub behaviors of more abstract concepts. Several medium level behaviors may form a higher level abstraction which is the adversaries main goal.
 Be detailed with the high level abstraction, model it as a scenario and how an adversary may achieve it through its sub behaviours. The high level abstraction should be a reasoned chain of sub behaviors.
 Each behavior has a behavior ID value assigned to it. Prefix with a behavior category and suffix with a number value. Provide detection analytics for the low level behaviors where possible.
 Summarise at the end with a table with the following: Behavior ID, Description, Commands where relevant and Detection Analytics

(C) Prompt instruction for transforming an IOB detection into SWRL syntax

Use the SWRL syntax for reasoning rules.
 Command Line is a dataproperty named "commandLine" in the ontology.
 A process is a member of the "Image" class.
 A process individual will have the data property "commandLine" and contain some string to check.
 The ontology prefix is "threat".
 Each Behavior ID is a class.

(D) Task instruction for transforming an IOB detection into SWRL syntax

I will represent an example SWRL rule. Can you transform the example into an SWRL rule for each individual command where relevant.
 $Image(?e) \wedge commandLine(?e, ?c) \wedge swrlb : contains(?c, "nc - u/server/share") \rightarrow LB1001(?e)$

Fig. 11. Prompts used for NeSy-driven threat hunting.

Behavior ID	Description	Commands
HB001	Data Exfiltration via Command Prompt	
MB001	Collect Sensitive Data	
LB001	Search for Files of Interest	<code>dir /s *,dir C:\Users*\Documents*.doc*</code>
LB002	Copy Files to Temporary Location	<code>xcopy C:\Users*\Documents*.doc* C:\Temp\</code>
LB003	Compress Files for Exfiltration	<code>compress.exe /c /s C:\Temp*.doc*</code>
MB002	Stage the Data for Exfiltration	
LB004	Encrypt Data for Safe Transmission	<code>cipher.exe /e C:\Temp\compressed_data.zip</code>
LB005	Rename Files to Avoid Detection	<code>rename C:\Temp\compressed_data.zip C:\Temp\data.enc</code>
LB006	Move Data to Staging Area	<code>move C:\Temp\data.enc \StagingServer\Share</code>
MB003	Exfiltrate the Data	
LB007	Exfiltrate via HTTP	<code>curl -X POST -d @C:\Temp\data.enc http://example.com/upload</code>
LB008	Exfiltrate via FTP	<code>ftp -n -s:ftp_commands.txt</code>
LB009	Use Email for Exfiltration	<code>powershell Send-MailMessage -From "attacker@domain.com" -To "attacker@external.com" -Subject "Data" -Attachments "C:\Temp\data.enc"</code>

Fig. 12. An example scenario generated by GPT-4 Omni for generating an IOB taxonomy. This scenario is transformed into an OWL ontology and processed by a reasoning engine. The behaviors IDs are a URI and identifying property used in the ontology. SWRL reasoning logic is generated from the list of commands.

The prompt task specification, seen in figure 11, is developed based on the requirements for symbolic threat hunting, as elucidated in the requirements of an LLM for symbolic threat hunting. Each task specification defines the constraints and explicit requirements when processing the prompt. Prompts (A) and (B) are examples of how the scope of the task is defined for creating an IOB scenario and a generalised taxonomy of IOBs. Prompts (A) and (B) develop a holistic set of interconnected threat actor behaviours forming a scenario. An output of the prompt can be seen in figure 12.

Prompts (C) and (D) demonstrate the task specification for transforming IOBs into a set of user defined reasoning rules. These user-defined reasoning rules utilise the Semantic Web Rule Language (SWRL) for more nuanced symbolic reasoning. Overall, prompts (A)-(D) demonstrate the generation of symbolic representation aspects of this experiment.

An output of prompt task specifications (A) and (B) is found in figure 12. Note that each IOB has a Behavior ID. This Behavior ID is a uniform resource identifier, used to uniquely identify each IOB in the ontology. The prefix is a naming convention to ascertain which level an IOB is. *L* is for Low. *M* is for Medium and *H* is for High. More information on these levels can be found in [24] and [23].

5.4.1. Results and observations from NeSy-driven threat hunting

GPT4-Omni was the best-performing model. Without a system prompt, it produced various IOB scenarios and transformed these into a taxonomy. Like most of the models, it arbitrarily chose an IOB ID rather than generating the same each time when a system prompt was not provided. The model worked best with a system prompt, producing consistent IOB scenarios that can be concatenated into a behavioural taxonomy. An example scenario output by *GPT4-Omni* can be seen in figure 12.

The scenario in figure 12 creates the top-level behaviour *H01* (“Data Exfiltration via Command Prompt”) and generates a set of associated behaviours for this scenario. This is the optimal output expected from the LLM. Unlike Llama, this model was capable of producing an SWRL rule-set based on the possible command examples present in the IOB scenario. However, without the contextualised prompt instructions, it would create its own taxonomy.

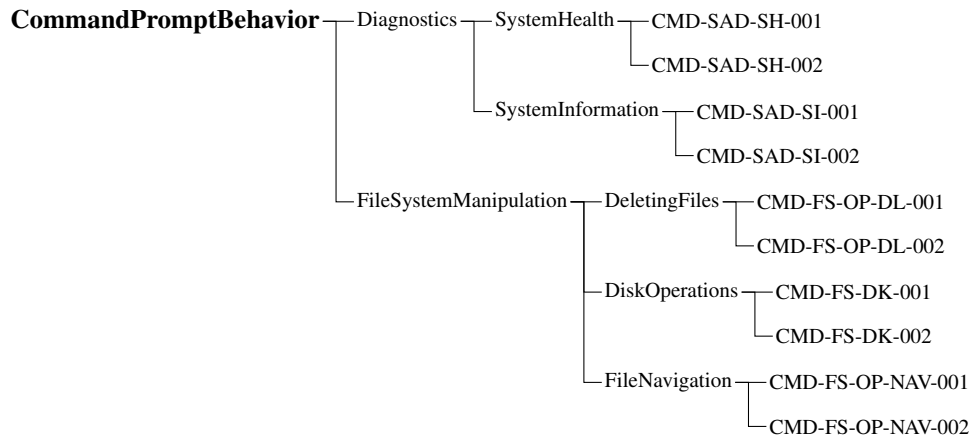


Fig. 13. Example class hierarchy of IOB Behaviors for the CommandPromptBehavior class. Each indent is a subclass. This class hierarchy was generated by GPT-4 Omni.

Instead of using SWRLs built-in regular expressions²¹ to trigger within the "commandLine" data property, it preferred to create a "commandLine" class and reason over a "hasCommand" relationship. After defining the scope in the prompt instruction and stating the task, it was capable of correctly generating the SWRL relationships. An example class hierarchy of IOBs can be seen in Figure 13. These behaviours and sub-behaviors were generated by the LLM, merged and output as an OWL2 schema²².

The same limitations found in GPT4-Omni were also present in *GPT3.5*, and there was little variance between the findings for these two models. *Llama 3.2-3b* primarily used its default configuration.²³ When including a system prompt, Llama was able to perform the IOB scenario generation task and develop a general taxonomy of behaviours. However, the model is prone to inconsistencies with its output. Without a system prompt, the model arbitrarily chooses its IOB identifier, making repeatable scenarios challenging without human input. The model had challenges when generating SWRL, generating nonsensical URIs to domain concepts that didn't exist. Similarly, inventing its own schema and annotation property. An example SWRL rule had to be provided in its task description to ensure the model consistently produced SWRL rules in the correct syntax. Without this in-context learning the model was prone to hallucinations. Once this context was provided the model consistently output SWRL rules for any detection logic generated in the taxonomy task. An example output can be found in figure 14, where events that match the conditions are classified as the relevant IOB class.

LB1001 SWRL Behavior Classification Rule:

$\text{Image}(?i) \wedge \text{commandLine}(?i, ?c) \wedge \text{swrlb:contains}(?c, "nc -u //server/share") \rightarrow \text{LB1001}(?i)$

LB1002 Behavior SWRL Classification Rule:

$\text{Image}(?i) \wedge \text{commandLine}(?i, ?c) \wedge \text{swrlb:contains}(?c, "ms14-064") \rightarrow \text{LB1002}(?i)$

LB1003 Behavior SWRL Classification Rule:

$\text{Image}(?i) \wedge \text{commandLine}(?i, ?c) \wedge$
 $\text{swrlb:contains}(?c, "Invoke-WebRequest -Uri https://...") \rightarrow \text{LB1003}(?i)$

Fig. 14. Example output of SWRL rule generation based on IOB scenario - Generated by Llama 3.2

²¹<https://www.w3.org/submissions/SWRL/#8>

²²<https://www.w3.org/TR/owl2-primer/>

²³The following notable settings were used: 28 discrete model layers, three CPU threads, the temperature was 0.5 and the context length was 8,192.

SecurityLLM primarily used its default configuration²⁴, and performed the worst in this experiment. System prompts were not possible, which limited the ability to narrow the scope for the experiment. The model is capable of providing abstract definitions for IOB concepts and the process of developing a taxonomy of behaviours, but not any of the other tasks, regardless of how prompts and tasks were tweaked.

To summarize, except *SecurityLLM*, all models fulfilled all the requirements for developing an IOB taxonomy, but there were difficulties in handling symbolic aspects for all cases. Without prompt instructions defining the context and the ontology schema, each model would create its own taxonomy with variance between each of them. This variance makes it difficult to integrate when the concepts, rules and relationships vary each time. Once this context, constraints and rules were established, each model (except *SecurityLLM*) were capable of transforming the taxonomy into an OWL2 schema. The GPT-models were capable of creating SWRL rules without an example rule provided, whereas Llama had difficulties in understanding this context and tried to form its own ontology for rules.

This experiment bridges the gap between the lack of an IOB taxonomy [23, 24] and the symbolic approach to threat hunting, thus demonstrating the value of NeSy. The symbolic approach to threat hunting has previously shown that reasoning engines can infer complex adversarial behaviours [23, 24], but rely on user-defined rule-based reasoning. We have shown that LLM-models can aid in the semi-automation of IOB development and automating the transformation of IOB scenarios into symbolic representations amenable for such reasoning.

5.5. Experiment 5: Data-driven enrichment of semantic kill-chain models

We have previously [123] applied data-driven enrichment of symbolic knowledge to help incident responders answer the questions:

- “What did most likely happen prior to this observation?”
- “What are the adversary’s most likely next steps given this observation?”

One of the major issues we faced in this research was the lack of sufficient data on computer security incidents. MITRE Engenuity recently published the tool *Technique Inference Engine (TIE)*²⁵, which uses a recommender model to infer a list of related techniques given a list of observed techniques. The data set used to train the TIE model is available on Github²⁶, and covers more than 6,000 computer security incidents. This data set is significantly larger than the data set used in [123]. In the following, we give a short description of the methodology and tools described in [123], as well as the following new contributions:

- A comparison of TIE and the tools presented in [123]
- New experiments using the TIE data set, including a discussion of the new results and conclusions
- A new analysis of the TIE data set and the the data set used in [123]

The available data for our method and TIE is a set of known incidents. Each incident contains an unordered set of MITRE ATT&CK techniques and sub-techniques. TIE uses this data to train a recommender model, which, when given a set of observed techniques as input, will output a set of techniques that most likely were used in the same incident. This gives incident responders guidance on what they should investigate, i.e., which techniques to look for. It does not cover the temporal aspect, i.e. what happened just before and after a specific observation of a technique. TIE’s approach does not include symbolic knowledge – it is purely data-driven. To answer our two questions above, i.e. the most likely prior and next steps with the available data sets, a NeSy approach is needed.

Our first step is the symbolic part, i.e., to formally model our knowledge of techniques. Every technique requires a set of abilities to be executed. Furthermore, every technique provides a set of abilities when executed. We developed a vocabulary of these abilities and mapped them to all the techniques and sub-techniques in ATT&CK. We then developed a tool²⁷, which when given a set of techniques and the mapped abilities as input, would output a set of

²⁴The following notable settings were used: 32 discrete model layers, three CPU threads, the temperature was 0.5 and the context length was 8,192.

²⁵<https://mitre-engenuity.org/cybersecurity/center-for-threat-informed-defense/our-work/technique-inference-engine/>

²⁶https://github.com/center-for-threat-informed-defense/technique-inference-engine/blob/main/data/combined_dataset_full_frequency.json

²⁷<https://github.com/mnemonic-no/provreq>

stages with the techniques that are possible to execute at each stage. The stages represent a temporal ordering of the technique: A technique in stage $n + 1$ depends on one or more techniques in stage n .

```

"tool_available": {
  "T1587.001": 83,
  "T1588.002": 181,
  "T1588.004": 16,
  "T1588.003": 27,
  "T1587.002": 9,
  "T1588.001": 37,
  "T1587.003": 18,
  "T1588": 45,
  "T1587": 48,
  "T1587.004": 7,
  "T1588.006": 7,
  "T1588.005": 8
}

```

Fig. 15. The number of times that techniques have provided the ability `tool_available`. In total (including the TIE data set), we observed 486 instances of a technique providing the ability `tool_available` to another technique observed in the same incident. In our original data set, this ability had 100 observed instances. The most frequently observed technique is Obtain Capabilities: Tool (T1588.002) with 181 occurrences, which corresponds to a Markov chain transition probability of $p = 181/486 \approx 0.3724$.

Our second step was to apply the symbolic model to add temporal information to the data set. For each incident in the data set, we record each instance of an ability being provided by one technique in that incident to another technique that requires that ability in the same incident. We transform the data set to a set of abilities, where each ability contains a set of techniques and a count of how many times we have observed each technique provide that ability to another technique in the same incident. Figure 15 shows an example of the technique counts for the ability “`tool_available`”.

Finally, we implemented a tool²⁸ that uses the technique counts from the previous step to determine the transition probabilities of a Markov chain, as explained in [123]. We then used Markov chain Monte Carlo simulations to determine the most likely technique prior to the observed technique. Our conclusions in [123] were that this approach is able to determine the prior technique with high probability, but if we try to determine a long attack chain, e.g. from observed exfiltration all the way back to initial access, then the most likely attack chain still has a very low probability. The example given for *Exfiltration over C2 Channel* (T1041) had a probability $p = 0.0202$, which is too low to be useful to an incident responder. In [123], we speculated that a larger data set might improve the performance for long attack chains.

After running the same experiments on the TIE data set, our results are similar. In one of the examples from [123], we see a clear improvement in the probability when we try to predict the prior technique: the most probable attack chain for the technique *User Execution* (T1204) had the probability $p = 0.6977$ in [123], while with the TIE data set this increased to $p = 0.8086$. In general, however, we see lower probabilities with the TIE data set compared to the data set in [123], e.g. with the *Exfiltration over C2 Channel* (T1041) example above. On closer examination, the reason for this result is that the TIE data set is more varied than our original data set as it covers a much larger number of techniques, and the technique observations are more evenly distributed.

To illustrate the difference between the data sets, we extracted the maximum Markov chain transition probability for each of the abilities in the transformed data set and created a box plot, shown in figure 16. The plot shows that the TIE data set has a lower median than the original data set, which means that in general a long attack chain generated from the TIE data set will have a lower probability than one created from the original data set (used in [123]).

²⁸<https://github.com/mnemonic-no/provreq-mcmc>

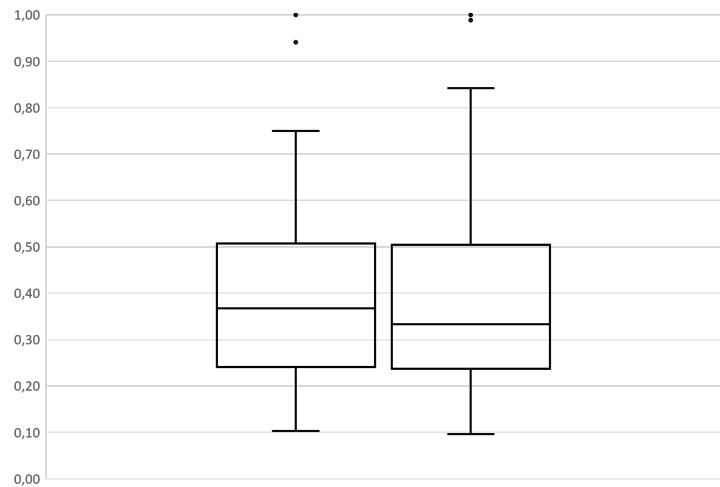


Fig. 16. Box plot with whisker boundary 1.5 interquartile range, showing the original data set to the left and the TIE data set to the right. The data used to create the box plot is the highest Markov chain transition probability for each ability. The outliers close to 1.00 are abilities where either a single technique was observed or where a single technique had an overwhelming number of observation compared to the other techniques. The outlier techniques for the original data set were Process Discovery (T1057), Container and Resource Discovery (T1613), and Gather Victim Org Information: Business Relationships (T1591.002), while the outlier techniques for the TIE data set were Process Discovery (T1057), Network Share Discovery (T1135), Modify Authentication Process: Network Device Authentication (T1556.004), and Gather Victim Org Information: Business Relationships (T1591.002).

Our conclusions from [123] are unchanged after testing our tools on the TIE data set: we are able to answer the questions in the introduction. However, our remarks that the low probability of long attack chains due to a lack of training data are not valid. Attackers are different and they use a varied set of techniques. Furthermore, new techniques are added to ATT&CK with each new revision. Based on the new experiments with the large TIE data set, we conclude that our approach is unlikely to give useful results for very long attack chains, and that our tools should rather be used iteratively during incident response: predict the most likely prior step, investigate, and then repeat the process once the prior attack step is confirmed.

6. Conclusion

Our main goal with this paper has been to showcase and demonstrate through experiments the possibilities for NeSy in cyber security, focussing on problems within SOCs. We hope this will help stimulate a concerted effort in studying NeSy in this domain. The use of NeSy for defending against cyber attacks is in its infancy, with some work having appeared over the last few years, including using NeSy for detection [14, 102], generating symbolic alerts [55] and extracting semantic knowledge from reports [87]. In addition, there exists work using NeSy in the cyber security domain that falls out of the scope of our paper, such as [90], where the focus is on adversarial attacks.

We have demonstrated that a considerable amount of symbolic and statistics-based AI is studied in SOC settings, and using it in real-world settings presents several challenges. We believe NeSy can address many of these challenges. Others have made some of the same points [59, 108], but not to the extent as we do here.

We have contributed by defining a set of NeSy use cases to address identified challenges, and mapping promising NeSy approaches to the use cases that serve as a starting point for further research. Several of the approaches have been demonstrated in our experiments, which are the main new contributions of this paper compared with [49]. An overview of the challenges, use cases, and experiments in this paper is presented in table 3. This work is just a start, and we both hope and expect that many new use cases and promising NeSy approaches that we have not covered here will appear in the not-too-distant future.

A challenge with AI in the cyber security domain is available data sets. Due to issues such as privacy, confidentiality, and lack of ground truth, researchers tend to use synthetic data, which have their limitations [8, 67].

MAPE- K Parts	Use case	Description	Challenges addressed	Experiments
Monitor	Use case 1	Use (symbolic) knowledge of threats and assets to guide or constrain ML-based detection engines.	Challenge 1, 3	Experiment 1
	Use case 2	Learn detection models from a limited number of (labelled) incidents.	Challenge 2	
	Use case 3	LLM-driven threat hunting using symbolic knowledge and reasoning capabilities.	Challenge 4	Experiment 4
Analyze	Use case 4	Incorporate learning of detection models with the ability to reason about their outcomes to understand and explain their nature and impact.	Challenge 5, 6	Experiment 3
	Use case 5	Extracting alerts in a symbolic form.	Challenge 5, 6	Experiment 3
	Use case 6	Use statistical AI to enrich or extract symbolic knowledge.	Challenge 6	Experiment 5
	Use case 7	Reason about and quantify knowledge.	Challenge 7	Experiment 3
	Use case 8	Relate the different phases of cyber incidents.	Challenge 6	Experiment 2, 3, 5
Plan & execute	Use case 9	Generating impact and risk aware explainable response actions in a timely fashion using neurosymbolic reinforcement learning.	Challenge 8	
Shared Knowledge	Use case 10	Generation of incident reports and CTI reports tailored for a given audience and/or formal requirements, using (symbolic) knowledge and LLMs.	Challenge 9	

Table 3

Overview of use cases with related challenges and experiments

Furthermore, such data sets tend to focus only on detection (monitor phase), containing only events, and lack the additional (symbolic) knowledge, which is important in SOCs and for our use cases. An important first step will be to develop synthetic data sets that contain both events for detection and necessary knowledge in order to address the use cases. This can either be achieved by extending existing “detection data sets” [72] with the necessary knowledge or by developing new “NeSy data sets” from scratch.

Acknowledgements This work was partially funded by the European Union as part of the European Defence Fund (EDF) project AInception (GA No. 101103385). Views and opinions expressed are, however, those of the authors only and do not necessarily reflect those of the European Union (EU). The EU cannot be held responsible for them. We would also like to thank the anonymous reviewers for the constructive feedback, which has helped improve the paper’s quality.

References

- [1] Kamal Acharya, Waleed Raza, Carlos Dourado, Alvaro Velasquez, and Houbing Herbert Song. Neurosymbolic reinforcement learning and planning: A survey. *IEEE Transactions on Artificial Intelligence*, pages 1–14, 2023.
- [2] Dyuman Aditya, Kaustuv Mukherji, Srikar Balasubramanian, Abhiraj Chaudhary, and Paulo Shakarian. PyReason: software for open world temporal logic. In *Proceedings of 2023 Spring Symposium on Challenges Requiring the Combination of Machine Learning and Knowledge Engineering (AAAI-MAKE 2023)*; arXiv preprint arXiv:2302.13482, 2023.
- [3] Zeeshan Ahmad, Adnan Shahid Khan, Cheah Wai Shiang, Johari Abdullah, and Farhan Ahmad. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1):e4150, 2021.
- [4] Leonardo Amado, Ramon Fraga Pereira, and Felipe Meneguzzi. Robust neuro-symbolic goal and plan recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11937–11944, 2023.
- [5] Adam Amos-Binks, Joshua Clark, Kirk Weston, Michael Winters, and Khaled Harfoush. Efficient attack plan recognition using automated planning. In *2017 IEEE symposium on computers and communications (ISCC)*, pages 1001–1006. IEEE, 2017.
- [6] Giuseppina Andresini, Feargus Pendlebury, Fabio Pierazzi, Corrado Loglisci, Annalisa Appice, and Lorenzo Cavallaro. Insomnia: Towards concept-drift robustness in network intrusion detection. In *Proceedings of the 14th ACM workshop on artificial intelligence and security*, pages 111–122, 2021.
- [7] Andy Applebaum. Finding Dependencies Between Adversary Techniques. Presented at the FIRST 2019 conference, 2019. Available at <https://www.first.org/resources/papers/conf2019/1100-Applebaum.pdf>.

- [8] Giovanni Apruzzese, Pavel Laskov, and Johannes Schneider. Sok: Pragmatic assessment of machine learning for network intrusion detection. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, pages 592–614. IEEE, 2023.
- [9] Yaniv Aspis, Krysia Broda, Jorge Lobo, and Alessandra Russo. Embed2Sym: Scalable Neuro-symbolic Reasoning via Clustered Embeddings. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 19, pages 421–431, 2022.
- [10] ATC-project. RE&CT. <https://atc-project.github.io/atc-react/>. Accessed: February 25, 2025.
- [11] Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. *Journal of Machine Learning Research*, 18(109):1–67, 2017.
- [12] Samy Badreddine, Artur d’Avila Garcez, Luciano Serafini, and Michael Spranger. Logic Tensor Networks. *Artificial Intelligence*, 303, 2022.
- [13] Tarek R Besold, Artur d’Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luis C Lamb, Priscila Machado Vieira Lima, Leo de Penning, et al. Neural-symbolic learning and reasoning: A survey and interpretation 1. In *Neuro-Symbolic Artificial Intelligence: The State of the Art*, pages 1–51. IOS press, 2021.
- [14] Alice Bizzarri, Brian Jalaian, Fabrizio Riguzzi, and Nathaniel D Bastian. A neuro-symbolic artificial intelligence network intrusion detection system. In *2024 33rd International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9. IEEE, 2024.
- [15] Clint Bodungen. *ChatGPT for Cybersecurity Cookbook*. Packt Publishing, 2024.
- [16] John R Boyd. The essence of winning and losing. *Unpublished lecture notes*, 12(23):123–125, 1996.
- [17] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.
- [18] Bushra A. Alahmadi and Louise Axon. 99% False Positives: A Qualitative Study of SOC Analysts’ Perspectives on Security Alarms. In *Proceedings of the 31st USENIX Security Symposium*, 2022.
- [19] Pedro Cabalar. Temporal asp: From logical foundations to practical use with telingo. In *Reasoning Web International Summer School*, pages 94–114. Springer, 2021.
- [20] Pedro Cabalar, Roland Kaminski, Philip Morkisch, and Torsten Schaub. telingo= asp+ time. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 256–269. Springer, 2019.
- [21] Pedro Cabalar, Roland Kaminski, Torsten Schaub, and Anna Schuhmann. Temporal answer set programming on finite traces. *Theory and Practice of Logic Programming*, 18(3-4):406–420, 2018.
- [22] Susan Chang and Pedro Jaramillo. Behind the scenes of Elastic Security’s generative ai features – a quantitative approach to prompt tuning and LLM evaluation. <https://www.elastic.co/blog/elastic-security-generative-ai-features>. Accessed: March 4, 2025.
- [23] Robert Chetwyn, Martin Eian, and Audun Jøsang. Onto hunt - a semantic reasoning approach to cyber threat hunting with indicators of behaviour. In *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 853–859, 09 2024.
- [24] Robert Andrew Chetwyn, Martin Eian, and Audun Jøsang. Modelling indicators of behaviour for cyber threat hunting via sysmon. In *Proceedings of the 2024 European Interdisciplinary Cybersecurity Conference*, EICC ’24, pages 95–104, New York, NY, USA, 2024. Association for Computing Machinery.
- [25] Paul Cichonski, Tom Millar, Tim Grance, Karen Scarfone, et al. Computer security incident handling guide – revision 2. *NIST Special Publication*, 800(61):1–147, 2012.
- [26] Benoit Claise. Cisco systems netflow services export version 9. Technical report, Cisco, 2004.
- [27] Djork-Arné Clevert. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.
- [28] Daniel Cunnington, Mark Law, Jorge Lobo, and Alessandra Russo. Neuro-symbolic learning of answer set programs from raw data. *arXiv preprint arXiv:2205.12735*, 2022.
- [29] Alessandro Daniele, Tommaso Campari, Sagar Malhotra, and Luciano Serafini. Deep symbolic learning: Discovering symbols and rules from perceptions. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23), Main Track*, pages 3597–3605, 2023.
- [30] Giuseppe De Giacomo, Moshe Y Vardi, et al. Linear temporal logic and linear dynamic logic on finite traces. In *Ijcai*, volume 13, pages 854–860, 2013.
- [31] Ryan Kerr Steven Ding and Li Li Adrian Taylor. Accelerating autonomous cyber operations: A symbolic logic planner guided reinforcement learning approach. In *Proceedings of the International Conference on Computing, Networking and Communications (ICNC 2024)*, pages 641–647, 2024.
- [32] BK Done, KD Willett, DW Viel, GW Tally, DF Sterne, and B Benjamin. Towards a capability-based architecture for cyberspace defense. *2016, Concept Paper Approved for Public Release, US Department of Homeland Security, US National Security Agency Information Assurance Directorate, and the Johns Hopkins University Applied Physics Laboratory, AOS-16-0099*, 2016.
- [33] Elastic. Elastic AI assistant. https://github.com/elastic/kibana/tree/main/x-pack/solutions/security/plugins/elastic_assistant. Accessed: February 25, 2025.
- [34] Håkon Svee Eriksson and Gudmund Grov. Towards XAI in the SOC – a user centric study of explainable alerts with SHAP and LIME. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 2595–2600, 2022.
- [35] Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.
- [36] Meng Fang, Shilong Deng, Yudi Zhang, Zijing Shi, Ling Chen, Mykola Pechenizkiy, and Jun Wang. Large language models are neuro-symbolic reasoners. In *Proceedings of the AAI conference on artificial intelligence*, volume 38, pages 17985–17993, 2024.
- [37] Robert Flood and David Aspinall. Measuring the complexity of benchmark nids datasets via spectral analysis. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 335–341. IEEE, 2024.

- [38] Ulrik Franke, Annika Andreasson, Henrik Artman, Joel Brynielsson, Stefan Varga, and Niklas Vilhelm. Cyber situational awareness issues and challenges. In *Cybersecurity and Cognitive Science*, pages 235–265. Elsevier, 2022.
- [39] Francesco Fuggitti and Tathagata Chakraborti. Nl2ltl—a python package for converting natural language (nl) instructions to linear temporal logic (ltl) formulas. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 16428–16430, 2023.
- [40] Konstantinos Fysarakis, Vasileios Mavroeidis, Manos Athanatos, George Spanoudakis, and Sotiris Ioannidis. A blueprint for collaborative cybersecurity operations centres with capacity for shared situational awareness, coordinated response, and joint preparedness. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 2601–2609. IEEE, 2022.
- [41] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.
- [42] Peng Gao, Fei Shao, Xiaoyuan Liu, Xusheng Xiao, Zheng Qin, Fengyuan Xu, Prateek Mittal, Sanjeev R Kulkarni, and Dawn Song. Enabling efficient cyber threat hunting with cyber threat intelligence. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 193–204. IEEE, 2021.
- [43] Artur d’Avila Garcez and Luis C Lamb. Neurosymbolic ai: The 3 rd wave. *Artificial Intelligence Review*, 56(11):12387–12406, 2023.
- [44] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.
- [45] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Multi-shot ASP solving with clingo. *CoRR*, abs/1705.09811, 2017.
- [46] Renato Lui Geh, Jonas Gonçalves, Igor Cataneo Silveira, Denis Deratani Mauá, and Fabio Gagliardi Cozman. dPASP: A Comprehensive Differentiable Probabilistic Answer Set Programming Environment For Neurosymbolic Learning and Reasoning. *arXiv preprint arXiv:2308.02944*, 2023.
- [47] Nirnay Ghosh and Soumya K Ghosh. A planner-based approach to generate and analyze minimal attack graph. *Applied Intelligence*, 36:369–390, 2012.
- [48] Alexander Gray. IBM Neuro-Symbolic AI Workshop 23-27 Jan 2023. <https://ibm.github.io/neuro-symbolic-ai/blog/nsai-wkshp-2023-blog/>, 2023. Opening address.
- [49] Gudmund Grov, Jonas Halvorsen, Magnus Wiik Eckhoff, Bjørn Jervell Hansen, Martin Eian, and Vasileios Mavroeidis. On the use of neurosymbolic ai for defending against cyber attacks. In *International Conference on Neural-Symbolic Learning and Reasoning*, pages 119–140. Springer, 2024.
- [50] Andreas Gylling, Mathias Ekstedt, Zeeshan Afzal, and Per Eliasson. Mapping cyber threat intelligence to probabilistic attack graphs. In *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 304–311. IEEE, 2021.
- [51] Steffen Haas and Mathias Fischer. Gac: graph-based alert correlation for the detection of distributed multi-step attacks. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC ’18*, page 979–988, New York, NY, USA, 2018. Association for Computing Machinery.
- [52] Kristian Hammond and David Leake. Large language models need symbolic ai. In *Proceedings of the 17th International Workshop on Neural-Symbolic Learning and Reasoning, La Certosa di Pontignano, Siena, Italy*, volume 3432, pages 204–209, 2023.
- [53] Md Ariful Haque, Sachin Shetty, Charles A Kamhoua, and Kimberly Gold. Adversarial technique validation & defense selection using attack graph & attack matrix. In *2023 International Conference on Computing, Networking and Communications (ICNC)*, pages 181–187. IEEE, 2023.
- [54] Richards J Heuer Jr. Analysis of competing hypotheses. *Psychology of intelligence analysis*, pages 95–110, 1999.
- [55] Anna Himmelhuber, Dominik Dold, Stephan Grimm, Sonia Zillner, and Thomas Runkler. Detection, explanation and filtering of cyber attacks combining symbolic and sub-symbolic methods. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 381–388. IEEE, 2022.
- [56] Patrick Hohenecker and Thomas Lukasiewicz. Ontology reasoning with deep neural networks. *Journal of Artificial Intelligence Research*, 68:503–540, 2020.
- [57] Paul RB Housel, Priyanka Singh, Siamak Layeghy, and Marius Portmann. Towards explainable network intrusion detection using large language models. *arXiv preprint arXiv:2408.04342*, 2024.
- [58] Zhisheng Hu, Minghui Zhu, and Peng Liu. Adaptive cyber defense against multi-stage attacks using learning-based pomdp. *ACM Transactions on Privacy and Security (TOPS)*, 24(1):1–25, 2020.
- [59] Brian Jalaian and Nathaniel D. Bastian. Neurosymbolic AI in Cybersecurity: Bridging Pattern Recognition and Symbolic Reasoning. In *MILCOM 2023 - 2023 IEEE Military Communications Conference (MILCOM)*, pages 268–273, 2023.
- [60] Pontus Johnson, Robert Lagerström, and Mathias Ekstedt. A meta language for threat modeling and attack simulations. In *Proceedings of the 13th international conference on availability, reliability and security*, pages 1–8, 2018.
- [61] Victor Jüttner, Martin Grimmer, and Erik Buchmann. ChatIDS: Advancing explainable cybersecurity using generative AI. *International Journal On Advances in Security*, 17(1):2, 2024.
- [62] Daniel Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011.
- [63] Peter E Kaloroumakis and Michael J Smith. Toward a knowledge graph of cybersecurity countermeasures. *The MITRE Corporation*, 11, 2021.
- [64] Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, et al. Mrkl systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *arXiv preprint arXiv:2205.00445*, 2022.
- [65] Henry Kautz. The third AI summer: AAAI Robert s. Englemore memorial lecture. *AI Magazine*, 43(1):105–125, 2022.

- [66] Henry Kautz. A taxonomy of neuro-symbolic ai or are we already there: Neuro symbolic summer school 2024 lecture. <https://neurosymbolic.github.io/nsss2024/>, 2024. Accessed: February 24, 2025.
- [67] Anthony Kenyon, Lipika Deka, and David Elizondo. Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets. *Computers & Security*, 99:102022, 2020.
- [68] Jeffrey O Kephart and David M Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.
- [69] Pascal Kesseli, Peter O’Hearn, and Ricardo Silveira Cabral. Logic. py: Bridging the gap between llms and constraint solvers. *arXiv preprint arXiv:2502.15776*, 2025.
- [70] Abderrazak Khediri, Hamda Slimi, Ayoub Yahiaoui, Makhlof Derdour, Hakim Bendjenna, and Charaf Eddine Ghenai. Enhancing machine learning model interpretability in intrusion detection systems through shap explanations and llm-generated descriptions. In *2024 6th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pages 1–6. IEEE, 2024.
- [71] Elmar Kiesling, Andreas Ekelhart, Kabul Kurniawan, and Fajar Ekaputra. The sepses knowledge graph: an integrated resource for cybersecurity. In *International Semantic Web Conference*, pages 198–214. Springer, 2019.
- [72] Ilhan Firat Kilincer, Fatih Ertam, and Abdulkadir Sengur. Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Computer Networks*, 188:107840, 2021.
- [73] Jiyeon Kim, Yulim Shin, Eunjung Choi, et al. An intrusion detection model based on a convolutional neural network. *Journal of Multimedia Information System*, 6(4):165–172, 2019.
- [74] Igor Kotenko, Diana Gaifulina, and Igor Zelichenok. Systematic literature review of security event correlation methods. *IEEE Access*, 10:43387–43420, 2022.
- [75] Kabul Kurniawan, Andreas Ekelhart, and Elmar Kiesling. An att&ck-kg for linking cybersecurity attacks to adversary tactics and techniques. In *International Semantic Web Conference (ISWC) - Posters and Demos*, October 2021.
- [76] Kabul Kurniawan, Elmar Kiesling, and Andreas Ekelhart. Cykg-rag: Towards knowledge-graph enhanced retrieval augmented generation for cybersecurity. In *RAGE-KG 2024 Workshop at ISWC 2024*, November 2024.
- [77] Max Landauer, Florian Skopik, Maximilian Frank, Wolfgang Hotwagner, Markus Wurzenberger, and Andreas Rauber. Maintainable log datasets for evaluation of intrusion detection systems. *IEEE Transactions on Dependable and Secure Computing*, 20(4):3466–3482, 2022.
- [78] Max Landauer, Florian Skopik, and Markus Wurzenberger. Introducing a new alert data set for multi-step attack analysis. In *Proceedings of the 17th Cyber Security Experimentation and Test Workshop*, pages 41–53, 2024.
- [79] Jens Lehmann. Dl-learner: learning concepts in description logics. *The Journal of Machine Learning Research*, 10:2639–2642, 2009.
- [80] Zhenyuan Li, Jun Zeng, Yan Chen, and Zhenkai Liang. AttackKG: Constructing technique knowledge graph from cyber threat intelligence reports. In *European Symposium on Research in Computer Security*, pages 589–609. Springer, 2022.
- [81] Jiehui Liu and Jieyu Zhan. Constructing knowledge graph from cyber threat intelligence using large language model. In *2023 IEEE International Conference on Big Data (BigData)*, pages 516–521. IEEE, 2023.
- [82] Kai Liu, Fei Wang, Zhaoyun Ding, Sheng Liang, Zhengfei Yu, and Yun Zhou. A review of knowledge graph application scenarios in cyber security. *arXiv preprint arXiv:2204.04769*, 2022.
- [83] Zi Long, Lianzhi Tan, Chaoyang He, and Shengping Zhou. Collecting indicators of compromise from unstructured text of cybersecurity articles using neural-based sequence labelling. *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019.
- [84] Daoming Lyu, Fangkai Yang, Bo Liu, and Steven Gustafson. Sdrl: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2970–2977, 2019.
- [85] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. *Advances in neural information processing systems*, 31, 2018.
- [86] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *International Conference on Learning Representations*. International Conference on Learning Representations, ICLR, 2019.
- [87] Francesco Marchiori, Mauro Conti, and Nino Vincenzo Verde. STIXnet: A novel and modular solution for extracting all STIX objects in CTI reports. In *Proceedings of the 18th International Conference on Availability, Reliability and Security*, pages 1–11, 2023.
- [88] Vasileios Mavroeidis, Ryan Hohimer, Tim Casey, and Audun Jøsang. Threat actor type inference and characterization within cyber threat intelligence. In *2021 13th International Conference on Cyber Conflict (CyCon)*, pages 327–352, 2021.
- [89] Vasileios Mavroeidis and Mateusz Zych. Cybersecurity playbook sharing with stix 2.1. *arXiv preprint arXiv:2203.04136*, 2022.
- [90] Stefano Melacci, Gabriele Ciravegna, Angelo Sotgiu, Ambra Demontis, Battista Biggio, Marco Gori, and Fabio Roli. Domain knowledge alleviates adversarial attacks in multi-label classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):9944–9959, 2021.
- [91] Meta. Introducing LLaMA: A foundational, 65-billion-parameter language model — ai.meta.com. <https://ai.meta.com/blog/large-language-model-llama-meta-ai/>. Accessed: November 06, 2024.
- [92] Doug Miller, Ron Alford, Andy Applebaum, Henry Foster, Caleb Little, and Blake Strom. Automated adversary emulation: A case for planning and acting with unknowns. *MITRE CORP MCLEAN VA MCLEAN*, 2018.
- [93] MITRE. Attack flow. <https://github.com/center-for-threat-informed-defense/attack-flow.html>. Accessed: February 25, 2025.
- [94] Mitre. Mitre ATT&CK. <https://attack.mitre.org/>. Accessed: February 25, 2025.
- [95] Farzad Nourmohammadzadeh Motlagh, Mehrdad Hajizadeh, Mehryar Majd, Pejman Najafi, Feng Cheng, and Christoph Meinel. Large language models in cybersecurity: State-of-the-art. *arXiv preprint arXiv:2402.00891*, 2024.
- [96] Stephen Muggleton. Inductive logic programming. *New generation computing*, 8:295–318, 1991.
- [97] Boubakr Nour, Makan Pourzandi, and Mourad Debbabi. A survey on threat hunting in enterprise networks. *IEEE Communications Surveys & Tutorials*, 2023.

- [98] Jakob Nyberg, Pontus Johnson, and András Méhes. Cyber threat response using reinforcement learning in graph-based attack simulations. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–4. IEEE, 2022.
- [99] OASIS. Introduction to STIX. <https://oasis-open.github.io/cti-documentation/stix/intro.html>. Accessed: February 25, 2025.
- [100] Sean Oesch, Robert Bridges, Jared Smith, Justin Beaver, John Goodall, Kelly Huffer, Craig Miles, and Dan Scofield. An assessment of the usability of machine learning based tools for the security operations center. In *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pages 634–641, 2020.
- [101] I. Olarra, H.B. Meeuwissen, P. de Haan, and Rob van der Mei. Telosian: Reducing false positives in real-time cyber anomaly detection by fast adaptation to concept drift. In *International Conference on Information Systems Security and Privacy*, February 2025.
- [102] Darian M Onchis, Codruta Istin, and Hogege Eduard-Florin. Advantages of a neuro-symbolic solution for monitoring IT infrastructures alerts. In *2022 24th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 189–194. IEEE, 2022.
- [103] Vittorio Orbinato, Mariarosaria Barbaraci, Roberto Natella, and Domenico Cotroneo. Automatic mapping of unstructured cyber threat intelligence: An experimental study:(practical experience report). In *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*, pages 181–192. IEEE, 2022.
- [104] Xinming Ou, Sudhakar Govindavajhala, Andrew W Appel, et al. Mulval: A logic-based network security analyzer. In *USENIX security symposium*, volume 8, pages 113–128. Baltimore, MD, 2005.
- [105] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–20, 2024.
- [106] C Pascu and M Barros Lourenco, editors. *Artificial intelligence and cybersecurity research – ENISA research and innovation Brief*. European Union Agency for Cybersecurity, 2023.
- [107] Filippo Perrina, Francesco Marchiori, Mauro Conti, and Nino Vincenzo Verde. Agir: Automating cyber threat intelligence reporting with natural language generation. In *2023 IEEE International Conference on Big Data (BigData)*, pages 3053–3062, 2023.
- [108] Aritran Piplai, Anantaa Kotal, Seyedreza Mohseni, Manas Gaur, Sudip Mittal, and Anupam Joshi. Knowledge-enhanced neurosymbolic artificial intelligence for cybersecurity and privacy. *IEEE Internet Computing*, 27(5):43–48, 2023.
- [109] Nikolaos Polatidis, Elias Pimenidis, Michalis Pavlidis, Spyridon Papastergiou, and Haralambos Mouratidis. From product recommendation to cyber-attack prediction: Generating attack graphs and predicting future attacks. *Evolving Systems*, 11:479–490, 2020.
- [110] Paul Pols and Jan van den Berg. The unified kill chain. *CSA Thesis, Hague*, pages 1–104, 2017.
- [111] Connor Pryor, Charles Dickens, Eriq Augustine, Alon Albalak, William Yang Wang, and Lise Getoor. Neupsl: Neural probabilistic soft logic. In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 4145–4153. International Joint Conferences on Artificial Intelligence Organization, 8 2023. Main Track.
- [112] Sara Qamar, Zahid Anwar, Mohammad Ashiqur Rahman, Ehab Al-Shaer, and Bei-Tseng Chu. Data-driven analytics for cyber-threat intelligence and information sharing. *Computers & Security*, 67:35–58, 2017.
- [113] Linlu Qiu, Liwei Jiang, Ximing Lu, Melanie Sclar, Valentina Pyatkin, Chandra Bhagavatula, Bailin Wang, Yoon Kim, Yejin Choi, Nouha Dziri, et al. Phenomenal yet puzzling: Testing inductive reasoning capabilities of language models with hypothesis refinement. *arXiv preprint arXiv:2310.08559*, 2023. To appear in the Twelfth International Conference on Learning Representations (ICLR 2024).
- [114] Abhiramon Rajasekharan, Yankai Zeng, Parth Padalkar, and Gopal Gupta. Reliable Natural Language Understanding with Large Language Models and Answer Set Programming. *arXiv preprint arXiv:2302.03780*, 2023.
- [115] Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Shajith Ikbal, Hima Karanam, Sumit Neelam, Ankita Likhyan, and Santosh Srivastava. Logical neural networks. *arXiv preprint arXiv:2006.13155*, 2020.
- [116] Arnaud Rosay, Florent Carlier, and Pascal Leroux. Mlp4nids: An efficient mlp-based network intrusion detection for cicsids2017 dataset. In *Machine Learning for Networking: Second IFIP TC 6 International Conference, MLN 2019, Paris, France, December 3–5, 2019, Revised Selected Papers 2*, pages 240–254. Springer, 2020.
- [117] Md Kamruzzaman Sarker, Lu Zhou, Aaron Eberhart, and Pascal Hitzler. Neuro-symbolic artificial intelligence: Current trends. *Ai Communications*, 34(3):197–209, 2022.
- [118] Luciano Serafini and Artur S d’Avila Garcez. Learning and reasoning with logic tensor networks. In *Conference of the Italian Association for Artificial Intelligence*, pages 334–348. Springer, 2016.
- [119] Iman Sharafaldin, Arash Habibi Lashkari, Ali A Ghorbani, et al. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018.
- [120] Dan Shiebler. Writing detection rules with llms, Jul 2024.
- [121] Xiaokui Shu, Frederico Araujo, Douglas L Schales, Marc Ph Stoecklin, Jiyong Jang, Heqing Huang, and Josyula R Rao. Threat intelligence computing. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 1883–1898, 2018.
- [122] Leslie F Sikos. Cybersecurity knowledge graphs. *Knowledge and Information Systems*, 65(9):3511–3531, 2023.
- [123] Geir Skjotskift, Martin Eian, and Siri Bromander. Automated att&ck technique chaining. *Digital Threats*, 6(1), February 2025.
- [124] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, pages 305–316. IEEE, 2010.
- [125] Splunk. Splunk RBA. <https://splunk.github.io/rba/>. Accessed: February 25, 2025.
- [126] Zareen Syed, Ankur Padia, Tim Finin, Lisa Mathews, and Anupam Joshi. Uco: A unified cybersecurity ontology. In *Workshops at the thirtieth AAAI conference on artificial intelligence*, 2016.

- [127] Tobias Syvertsen. A comparison of machine learning based approaches for alert aggregation. Master thesis, University of Oslo, 2023. Available from: <https://www.duo.uio.no/handle/10852/104437>.
- [128] Eleni Tsekmezoglou, Rossen Naydenov, Marianthi Theodoridou, and Apostolos Malatras, editors. *ENISA Threat Landscape Methodology*. European Union Agency for Cybersecurity, 2023.
- [129] PeiYu Tseng, ZihDwo Yeh, Xushu Dai, and Peng Liu. Using llms to automate threat intelligence analysis workflows in security operation centers. *arXiv preprint arXiv:2407.13093*, 2024.
- [130] Dingmin Wang, BERNARDO CUENCA GRAU, PRZEMYSŁAW A WAŁĘGA, and PAN HU. Practical reasoning in datalogmtl. *Theory and Practice of Logic Programming*, pages 1–31, 2024.
- [131] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22*, Red Hook, NY, USA, 2024. Curran Associates Inc.
- [132] Florian Wilkens, Felix Ortman, Steffen Haas, Matthias Vallentin, and Mathias Fischer. Multi-stage attack detection via kill chain state machines. In *Proceedings of the 3rd Workshop on Cyber-Security Arms Race*, pages 13–24, 2021.
- [133] Thomas Winters, Giuseppe Marra, Robin Manhaeve, and Luc De Raedt. DeepStochLog: Neural stochastic logic programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10090–10100, 2022.
- [134] Maxime Würsch, Andrei Kucharavy, Dimitri Percia David, and Alain Mermoud. LLMs perform poorly at concept extraction in cybersecurity research literature. *arXiv preprint arXiv:2312.07110*, 2023.
- [135] Zhun Yang, Adam Ishay, and Joohyung Lee. NeurASP: Embracing neural networks into answer set programming. *arXiv preprint arXiv:2307.07700*, 2023.
- [136] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.
- [137] Yizhe You, Jun Jiang, Zhengwei Jiang, Peian Yang, Baoxu Liu, Huamin Feng, Xuren Wang, and Ning Li. Tim: threat context-enhanced ttp intelligence mining on unstructured threat data. *Cybersecurity*, 5(1):3, 2022.
- [138] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.
- [139] Zysec. ZySec-AI/SecurityLLM · Hugging Face — [huggingface.co](https://huggingface.co/ZySec-AI/SecurityLLM). <https://huggingface.co/ZySec-AI/SecurityLLM>. Accessed: November 06, 2024.