Graph-Based Modelling and Prediction of Adversary Behaviour from Honeypot Traces

David Guilherme*†, Miguel Faísco*†, Hans P. Reiser†, Ibéria Medeiros*
*LASIGE, DI, Faculdade de Ciências, Universidade de Lisboa, Portugal
†Department of Computer Science, Reykjavik University, Reykjavik, Iceland
fc56333@alunos.ciencias.ulisboa.pt, fc56954@alunos.ciencias.ulisboa.pt, hansr@ru.is, ivmedeiros@fc.ul.pt

***Abstract*—High-interaction honeypots provide detailed observations of attacker behaviour, but the collected data is often used in a limited and largely offline manner. In this short paper, we present a *Honeypot Session Enrichment* prototype that intends to enable session-aware characterisation of adversaries and prediction of their future behaviour based on real-time observations during honeypot execution. Its approach combines fine-grained eBPF-based tracing with an external processing pipeline that links observed events to a structured knowledge representation and augments them with local malware classification using static and dynamic features. Building on this representation, we employ a graph neural network to model attacker behaviour and try to predict future actions from partial observations of an ongoing session. A preliminary evaluation was performed to assess the feasibility of performing such analysis locally and in real time, and to evaluate the extent to which early-stage observations enable accurate prediction of attacker behaviour and intent.**

***Index Terms*—Honeypots, Knowledge Graphs, Machine Learning, Attacker Behaviour**

I. Introduction

Internet-connected systems represent high-value targets for adversaries. Effective defences depend not only on preventing intrusions but also on understanding how attacks proceed over time. Structured knowledge bases such as MITRE ATT&CK [1] provide a systematic way to describe adversary behaviour. But, translating these abstract taxonomies into actionable insights about live systems is a significant challenge.

High-interaction honeypots (*HiH*, for short) provide detailed traces about attacker interactions, capturing rich information about their behaviour within a controlled environment. However, observations and traces are typically not converted into structured representations that support systematic prediction of an attacker's next steps or inference of their overall objective. As a result, the information collected by honeypots may not be utilised to its full potential.

To address this gap, in this short paper, we present a *Honeypot Session Enrichment* (`HSE`) prototype that employs a pipeline to characterise ongoing sessions and support analysis of adversary behaviour from live honeypot observations, intending to predict future adversarial actions. At the core of `HSE`, eBPF-based monitoring of a HiH provides fine-grained tracing of attacker activity. These events are mapped to a knowledge graph (KG), and enriched with on-host malware classification using static and dynamic features collected during the session. Then, a neural network model operates over the KG to try to predict the next steps of ongoing attacks, and, therefore, attackers' behaviours. We conducted a preliminary evaluation of the `HSE` prototype, aiming to assess i) the feasibility of performing such HiH sessions' analysis locally and in real time, and ii) the extent to which early-stage observations enable accurate prediction of attacker behaviour and intent. The preliminary results are promising, showing that: a) our unified KG can capture information from HiH's sessions, enrich it with malware classification, and correlate the enriched data with external malware information; b) the neural network model can operate over the KG and enables the characterisation and prediction of attackers' behaviours; c) the processing and local analysis time consumption is acceptable and feasible for a real-time `HSE` operating.

The contributions of this short paper are: (1) the *Honeypot Session Enrichment* (`HSE`), a prototype capable of analysing sessions from a HiH honeypot with an eBPF module, through the usage of KGs and neural network models, which enable the prediction of adversarial behaviours and the classification of malware uploaded during said sessions. (2) A preliminary evaluation of the `HSE`, in which the results show its usefulness for what it was designed, i.e., its capacity to accurately classify malware and predict incoming attacker activity.

II. Background

This section introduces the key concepts and tools upon which our proposed approach is built.

Adversary behaviours are commonly assumed to unfold as structured, sequential steps, and each step can show different levels of observable malicious intent [2]. Cataloguing and classifying these behaviours is therefore important for the design and organisation of defensive systems. Two complementary frameworks support this effort. MITRE ATT&CK is a globally accessible knowledge base of adversary tactics and techniques derived from real-world observations that, beyond cataloguing offensive methods, provides detection and mitigation guidance for each entry [1]. Building on this classification, the Common Attack Pattern Enumeration and Classification (CAPEC) catalogue documents recurring attack patterns by decomposing each into its constituent behaviours (execution flow) and annotating them with MITRE ATT&CK labels [3].

A ***Knowledge Graph*** represents entities as nodes and relations as typed directed edges, typically stated as triples $\langle subject, predicate, object \rangle$ [4], [5]. In cybersecurity, this is

a natural way to connect heterogeneous objects such as malware families, MITRE ATT&CK techniques, and abstract attack steps, while representing the semantics of how they relate, rather than flattening them into independent tables or documents [5]. KGs are an attractive structure for integrating honeypot observations with external taxonomies over time, since they enable the addition of new entities and relation types rather than redesigning a monolithic schema [6].

Model training on a KG typically requires learned representations that respect relational structure in the graph and, when behaviour is observed as an ordered trace, temporal structure in that sequence. Common methods for this purpose are graph neural networks (GNNs) [7] and recurrent models [8].

A ***honeypot*** is a decoy system or service intended to attract attacker interaction while recording activity for later analysis [9]. They are described by the attacker's interaction level (low, medium, and high), with high interaction (i.e., HiH) being the one with richer traces.

Malware analysis aims to characterise what malicious samples can do and how they behave in practice, supporting risk assessment and defensive response. It is commonly divided into *static* analysis that inspects binaries and metadata without execution (e.g., headers/sections/imports), and *dynamic* analysis that observes runtime effects via network activity, system/kernel-level monitoring such as syscalls, and before/after system state.

III. Related Work

This section presents relevant related work on the domains our work focuses on.

A. Knowledge Graph Frameworks and Behaviour Models

On the knowledge representation side, Rastogi et al. [10] developed MALONT2.0, a malware ontology for Android threats, and Falcarin et al. [11] constructed a cybersecurity KG aggregating heterogeneous repository data for security professionals. Yang et al. [12] built an MITRE ATT&CK-grounded KG for advanced persistent threat detection, and Kincl et al. [13] benchmarked KG embedding methods specifically for Android malware detection, matching state-of-the-art performance at reduced computational cost.

On the behavioural prediction side, Mohammadzad et al. [14] proposed a dynamic model for attacker next-action prediction, while Camacho González [15] applied Bayesian networks to forecast attacker sequences from simulated data. Xu et al. [16] introduced a web attack behaviour graph combining explicit and implicit causal relationships, analysed via a multi-agent LLM framework. Most architecturally relevant to this work, Soylu et al. [17] proposed a hybrid GNN-GRU model for cyberattack prediction from heterogeneous network data, combining relational structure modelling with temporal sequence learning, closely related to our design.

B. Honeypots

Honeypots are an established technique for defending against cyberattacks. Well-known examples include Cowrie [18], a medium-interaction honeypot simulating vulnerable *SSH* and *Telnet* servers, T-Pot [19], a system that integrates multiple honeypots into a single framework with a comprehensive interface. For high-interaction honeypots, HIHAT [20], PwnyPot [21], and HoneyIoT [22] are among the open-source solutions currently available, and HoneyICS [23] is a specialised solution for Industrial Control Systems. Although this type of honeypot captures non-automated adversaries and more advanced threats, most still lack integrated malware analysis, and often rely on third-party tools to analyse and classify those samples later, potentially missing key behaviours specifically meant for the target system.

Virtualised environments have become standardised in high-interaction honeypots. One example of the technologies used to monitor these environments is eBPF (extended Berkeley Packet Filter [24], [25]), a framework that enables the execution of safe, sandboxed user code in response to kernel events. This allows for stealthy and near real-time exporting of said events (attacker behaviours) to a controlling machine.

C. Machine Learning Based Malware Analysis

Recent works in Machine Learning (ML) based malware analysis have become increasingly more heterogeneous, with novel techniques for feature selection, extraction, and classification. Liu, Wang et al. [26] extracted malware features for clustering using opcode n-grams and gray-scale images, while Xu, Ray et al. [27] have used memory access patterns to identify hardware-assisted threats. Also, recent work on incorporating artificial intelligence into multiple stages of malware analysis [28] has emerged. In classical ML, numerous studies have been conducted on malware analysis. Vinayakumar et al. [29], [30] evaluated classical ML algorithms and deep learning algorithms over public and private datasets. Sharmeen et al. [30], [31] proposed a semi-supervised framework used to analyse ransomware samples from VirusShare [32] and VirusTotal, while Carlin et al. [30], [33] tested the performance of different models. These studies yielded mixed results, with some classical models performing the best in some cases, while neural networks performed the best in others.

Although all of these systems present significant advances in their respective fields, there must be a comprehensive framework that integrates high-interaction honeypots, real-time malware classification, and attacker behaviour profiling, all at once, in a local environment. Such a desired framework is what we propose and explain next.

IV. Proposed HSE Architecture

In order to solve the aforementioned problems, we propose a *Honeypot Session Enrichment* (HSE) prototype, with the main objective being to provide as much tangible and comprehensive information about live honeypot sessions and their techniques, using KGs and ML models, able to i) classify malware, ii) characterise adversarial behaviours, and iii) predict possible malicious actions.

Figure 1 illustrates the architecture of the proposed framework, which is organised as follows. The **green** arrows relate

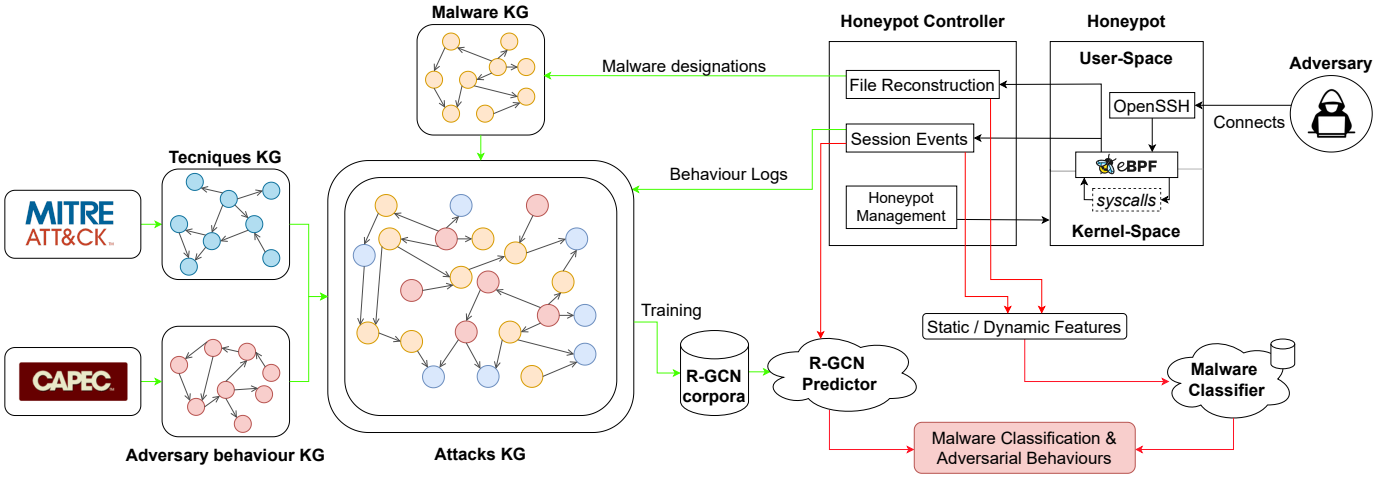

Fig. 1: Architecture of the proposed Honeypot Session Enrichment prototype.

to the training phase, which includes the building of the KGs and the training of the *R-GCN Predictor*, while the **red** arrows relate to the processing pipeline of a session received by the honeypot. The remaining **black** arrows describe the process of the transformation of attacker behaviours in a session into processable events by both the *R-GCN Predictor* and the *Malware Classifier*. Next, we describe the components of the framework: the *Honeypot* itself, the multiple KGs used to build the unified *Attacks KG*, the Adversarial Behaviour model (the R-GCN Predictor in the figure), as well as the processes of training and testing separate models for classifying uploaded malware samples and indicating their adversarial behaviours.

A. Honeypot

The high-interaction honeypot (HiH), integrating both static and dynamic analysis and an eBPF module, is designed to resemble a legitimate system with weak credentials. Attackers can easily gain access to and interact with it as if it was a real target, since it encourages them to perform their typical attack techniques (as they usually would). When the honeypot is running, its eBPF module listens for kernel-level events on the honeypot VM and forwards them to the host (the *Honeypot Controller*) via a listener. These events are generated by selected tracepoints (i.e., system calls) within the honeypot VMs' kernel and are triggered when a user interacts with one of them (e.g., opening a file, deleting a file, establishing a connection, etc.). The Honeypot Controller receives the events from eBPF, processes and stores them in a comprehensive log system. Table I shows the tracepoints the honeypot uses, which will produce events to classify uploaded malware and build attack profiles for incoming sessions.

This system further allows for both the reconstruction of files uploaded by attackers (*File Reconstruction* for static analysis) and the live observation and recording of all behavioural traces produced by said files (*Session Events* for dynamic analysis), which is useful for the implementation of malware analysis systems. eBPF was chosen as the base component of the honeypot because previous studies [24] showed that despite its flaws regarding high loads and stealthiness, it quickly processes and relays events: a priority for honeypot systems.

1) Collected Logs: Logs (from *Session Events*) collected by the honeypot are used in two phases: training and real-time processing. However, it is important to understand how logs are stored and which information is most relevant for behaviour analysis and prediction purposes. Such information, for example, can be the attacker's IP address connected to the honeypot and the type of action he performs (e.g., opening or executing a file), which can reveal and characterise an attacker's behaviour. (Appendix A details a log example).

For training, logs are used to build the KG for training the Adversarial Behaviour model (the R-GNC Predictor in Fig. 1), allowing it to learn directly from previous attacks. Then, when the framework is active, i.e., in the real-time processing phase, logs are inspected, and trigger framework outputs using the said KG and the honeypot itself.

2) Machine-Learning Based Malware Classifier: To gain insight into malware sent to our honeypot without relying on third-party services at runtime, we extended the honeypot system by integrating a local ML classifier. This classifier was trained on selected features from 6,342 VirusTotal reports from existing malware samples, and its goal is to classify new samples sent by attackers to the honeypot in real-time, while the attacker's session is still ongoing. This is achieved through

TABLE I: List of honeypot eBPF tracepoints

Purpose	Tracepoints	Outcome
Track process relations	*sched_process_fork* *sched_process_exit*	Detect process fork Detect process exit
Reconstruct Sessions	*bprm_committed_creds*	Commands Executed
External connections	*sys_enter_connect*	Trace Outgoing TCP
File mods / uploads	*do_sys_openat2* *filp_close* *chmod_common* *path_unlink* *sys_enter_write*	Trace file operations (open, write, etc.)

TABLE II: List of features that compose the malware dataset

Category	Feature
Static (Details)	Selected Header Fields
	Import List
	Export List
	Section List
	Shared Libraries
Dynamic (Behaviour)	Files Opened
	Files Deleted
	Files Written
	Files with Changed Attributes
	Shell Commands
Network	IP Traffic
Class	Malware Class Label

static and dynamic feature extraction, both provided by the honeypot system, as previously explained. Table II presents the selected features, organised by category. With these samples compiled into a dataset, we tested various ML models to determine which was best for our use case (see Section V-C1).

B. Attacks Knowledge Graph Construction

The proposed *Attacks KG* is merged from three different developed KGs: one focused on MITRE ATT&CK techniques (*Technique KG*), another on adversaries' behaviours (*Adversary Behaviour KG*), and a third on malware (*Malware KG*).

The *Malware KG* and *Technique KG* are constructed using malware specimens harvested from the honeypot environment over the previous annual cycle as the primary dataset. All these malware are modelled as discrete nodes interconnected with specific techniques from the MITRE ATT&CK framework, thus enabling the systematic mapping of behaviours such as Execution, Defence Evasion, Reconnaissance, and Discovery. Furthermore, these technical nodes are linked via directed edges to their respective mitigation strategies, providing high-fidelity features that establish a structural correlation between observed adversarial tactics and proactive defensive countermeasures. Appendix B illustrates an *Attacks KG*, where these two KGs are represented by the blue and yellow nodes.

The *Adversary Behaviour KG* is modelled using the CAPEC attacks, which can have two variants: *sequential* and *non-sequential*. The former is represented by the execution flow, where each discrete phase of an attack pattern is a node. These nodes are instantiated from logs retrieved from the honeypot and are systematically categorised using MITRE ATT&CK techniques to provide a standardised behavioural classification. On the other hand, *non-sequential* attacks, i.e., those that do not depend on an attack sequence, can be mapped directly into MITRE ATT&CK techniques. The logs that will enrich the KG will represent behaviours for each step of the execution flow of an attack, and they will also represent MITRE ATT&CK techniques for CAPEC attacks that do not depend on a sequence to be performed. The Attacks KG of Appendix B shows these two cases, where the red nodes on the right represent CAPEC attacks and those on the left represent behaviours from the execution flow of one of them.

By integrating these components, we construct the unified and final *Attacks KG* and enrich it with timestamped honeypot logs. This KG leverages semantic interoperability by identifying entities common to multiple ontologies, such as the MITRE ATT&CK techniques shared between the *Adversary Behaviour KG* and the *Technique KG*. These shared nodes facilitate cross-domain linkage, similar to the relational mapping established between the *Technique KG* and *Malware KG*.

The primary objective of this multi-layered KG is to enhance the reasoning heuristics of the *Adversarial Behaviour Model* (explained next). By providing a rich semantic context, the KG enables the model to maintain analytical coherence even during non-linear or fragmented attack sequences. This architecture further facilitates predictive impact assessment and the identification of emerging threats when encountering novel adversarial patterns previously unobserved.

C. Adversarial Behaviour Model

The primary objective of the model is the autonomous characterisation of user behaviour to infer underlying adversarial intent, supporting the following predictive tasks: *forecasting the attacker's next action* and *identifying the attack's final goal*. The model operates on the unified Attacks KG, where it connects CAPEC attacks to their execution flows by observing the real-time behaviours and comparing them with the representative instance logs of the MITRE ATT&CK techniques.

To achieve this, we start our architecture with an R-GCN model, a specific GNN that handles the heterogeneous relational structure of the Attacks KG by assigning a distinct weight matrix to each relation type, allowing semantically distinct edge types to be treated independently and enabling effective modelling of the complex inter-entity relationships encoded in the graph.

The objective is to deploy the model once it is improved and performing well directly within the honeypot environment for real-time inference. As attacker actions are recorded, the model continuously analyses the observed behavioural sequence, predicting the next likely action, the attacker's terminal objective, and, if it has enough data, the expected impacts on a real system. Given this real-time operational requirement, the model is designed to balance predictive accuracy with computational efficiency.

D. `HSE` Operation

Our `HSE` prototype integrates heterogeneous data sources into a unified pipeline spanning from data ingestion to real-time behavioural prediction, as illustrated in the left half of Figure 1. It contains three sequential phases: knowledge graph construction, model training, and operational deployment.

In the first phase, the Attacks KG is constructed from three primary data streams. Raw malware specimens harvested from the honeypot are mapped to the MITRE ATT&CK framework to extract relevant tactics and techniques, filtered according to their significance to the research scope. These technical attributes are then enriched with behavioural abstractions derived from CAPEC execution flows and correlated with ground-truth activity logs retrieved from the honeypot, which function as empirical instances that instantiate the abstract behavioural nodes within the graph.

In the second phase, the populated Attacks KG is used to train the model. The structured semantic context provided by the KG enhances the model's reasoning, improving its accuracy in both threat detection and predictive forecasting [34]. The relational structure of the graph enables the model to generalise across novel attack vectors, supporting the identification of previously unobserved behavioural patterns that deviate from known sequences.

The model was built to receive logs directly, as shown in the following query: {`"type": "exec", "filename": "/usr/bin/grep", "dport": -1, "session": "sshd-15565"`}.
Once the model predicts the next behaviour and the likely malware to be deployed, it produces the following tuple as output: <`Type, Detected technique, Forecast`>, where `Type` is the current action type recorded in the log, `Detected Technique` is the MITRE technique assigned to the current log entry, and `Forecast` is the model's prediction for the subsequent technique. Additionally, once the session ends, the model reports a set of performance metrics (see Section V-A) and a malware prediction, which is evaluated by comparing it against the classification produced by the *Malware Classifier* component.

Lastly, in the third phase, both the trained R-GCN Predictor and honeypot ML classifier models are evaluated through hold-out testing using historical log datasets from prior honeypot iterations, ensuring longitudinal validity. Upon successful evaluation, the models are deployed to the live honeypot environment to perform real-time predictive inference and malware classification on incoming attacker activity.

V. Preliminary Results

This section presents a preliminary evaluation of `HSE`, regarding the points i) and ii) stated in the introduction.

A. Metrics

During our preliminary experiments, we used the following metrics to measure the performance of our honeypot session enrichment framework models. The metrics can be described as: ***Accuracy*** determines how many of the model's predictions were correct. ***Precision*** determines how many of the predictions for a given class were correct, while ***Recall*** determines how many of the actual instances of a class were correctly predicted. ***F1-Score*** provides a harmonic mean between precision and recall, in order to understand if these two metrics are correctly balanced. Precision, Recall, and F1-Score require averaging across all classes, since they can only be applied individually per class in our models. Appendix C contains the formulas of these metrics.

B. Honeypot Capturing Effectiveness

In order to understand the honeypot's effectiveness at capturing and keeping attackers, we let the honeypot machine run, undisturbed and open to connections, for two consecutive weeks. After these two weeks, the honeypot received a total of 289,413 connections from 2,632 different IP addresses. This indicates an average of 19 connections per IP address, suggesting persistent probing and automated attacks instead of random scanning of open ports.

Furthermore, from these connections, 224,205 managed to successfully login into the honeypot. This high number of logins relative to the total number of connections (around 77%) further demonstrates the honeypot's effectiveness not only in attracting but also engaging attackers. Overall, these results show that our high-interaction honeypot equipped with an eBPF module is capable of handling high volumes of traffic over a relatively long timeframe, and at being enticing enough for attackers to go beyond a simple scan.

C. Malware Classifier Performance

This section presents the evaluation of the ML-based classifier component in the proposed framework, specifically in our honeypot. Firstly, we discuss the process for choosing the correct ML model for our use case, followed by the model's performance over samples received by the honeypot.

1) Determine the Best Machine Learning Classifier: With the dataset being compiled and finalised (as explained in IV-A2, and in order to determine the right classifier model to use in the framework, we tested the models Random Forest (RF), Gradient Boost (GB), Naïve-Bayes (NB), and Support Vector Classifier (SVC). These four models were firstly trained and tested over an 80/20 split of the dataset, in order to extract the most determining features, with the intention of reducing the column size of the dataset. This reduction is necessary since most of the original dataset columns used list-based attributes that, once expanded during preprocessing, resulted in a very high-dimensional feature space (around 25,587, in our case). With the aforementioned reduction, however, the dataset was reduced to approximately 900 columns.

With the newly reduced dataset, the three best-performing models – RF, GB, and SVC – were evaluated using 5-fold cross-validation to determine the selected model for our system. The model parameters were configured as follows (scikit-learn v1.7.0): RandomForestClassifier with a balanced class weight (`class_weight='balanced'`), with a 25 maximum depth over 600 estimators (`max_depth=25, n_estimators=600`), and 0.75 maximum features (`max_features=0.75`), GradientBoostingClassifier with a 25 maximum depth over 600 estimators (`max_depth=25, n_estimators=600`), and SVC with rbf kernel (`kernel='rbf'`). The results are presented in Table III.

While GB achieved the best overall performance, it had the longest fitting times (295.13 seconds), which makes it less

TABLE III: Model performance using the dataset with reduced feature size.

Metric / Model	**RandomForest**	GradientBoost	SVC
Fit time	**34.39s**	295.13s	2.67s
Accuracy	**0.932**	0.936	0.901
Precision	**0.932**	0.934	0.895
Recall	**0.932**	0.936	0.901
F1 - Score	**0.933**	0.935	0.903

suitable for a real-time honeypot with short time response and future implementations in which we plan to add retraining. Conversely, SVC had the fastest fitting times but the lowest performance. In contrast to these two models, RF provides satisfactory performance metrics while maintaining acceptable fitting times, and so was the model selected for our use case.

2) Classifier Performance: Throughout the honeypot system's runtime, attackers uploaded 52 different files, a total of 1,564 times. All of these files were presumed as being malware, and thus analysed by our malware classification model. Since VirusTotal reports were used for the training and testing of the classifier, the samples received in the honeypot were sent to VirusTotal, in order to establish a baseline for the classifier's evaluation.

Unexpectedly, of these 52 different files, 12 (around 23.0%) were benign. Since we did not account for this, these samples will always be misclassified. Nevertheless, it is quite interesting, and we concluded that it could be part of a preliminary probing phase of an attack. Of the remaining 40 files, 8 (around 23.0%) were classified by VirusTotal as classes not included in our dataset. This is an oversight which has to be worked on, both with the acquisition of new samples for the dataset and the implementation of model retraining in later iterations.

With this, out of the final 32 samples, our ML-based classifier incorrectly classified 1 sample. This brings the final total down to 31/52 correct classifications, or about 59.6%. While the ML classifier managed to accurately classify almost all of the samples it was trained for, the following iterations must address the gaps found in this test.

D. Adversary Behaviour results

This section presents the results of our initial evaluations on the current version of the adversarial behaviour model (the R-GCN), using a set of 1,776 pre-recorded honeypot sessions. This set contained only sessions in which malware was deployed or executed at some point. For next-behaviour forecasting, the model achieved an accuracy of 0.751, a precision of 0.164, a recall of 0.822, and an F1-score of 0.274, as shown in Table IV.

Although some scores are low, the two most relevant metrics are accuracy, which represents how often the model correctly predicted the next behaviour, and recall, which indicates how often the correct behaviour appeared among the top five forecasted options chosen by the model. These two metrics have much better scores compared with the precision metric but in order to achieve a more robust and reliable system, we aim to improve in this aspect.

The evaluation of the malware prediction feature was conducted on a smaller sample of 61 sessions, yielding an accuracy of 81.9%. However, this result is limited by a data constraint: most recorded malware sessions conclude with the deployment of a crypto miner, so most logs in the Attacks KG originate from crypto miner sessions. This creates a bias in which the model tends to classify nearly every session as a miner, a limitation we are aware of and working to address.

TABLE IV: Performance of the Adversarial Behaviour model

Model	**Accuracy**	Precision	**Recall**	F1 - Score
Results	**0.751**	0.164	**0.822**	0.274

E. Discussion

Based on the preliminary results obtained and taking into account points i) and ii), we can observe and conclude the following. As far as what insights about attacker behaviour can be obtained from the observations on the honeypot, we were capable of obtaining information on the type of attack that is being performed, predicting next steps with some confidence. With the proposed system, MITRE ATT&CK techniques employed by an attacker, and a classification of any malware sent by this same attacker, can be analysed in real-time, without relying on any external services.

From the behaviour observed within a honeypot session, specifically by its interactions and uploaded malware, it is possible to infer what the attacker aims to exploit during this same session, as well as what objectives it intends to achieve with this. From previous attack information, we were able to create a KG that combines techniques, attack patterns, malware and instances of real behaviours. This representation helps identify patterns and similarities that can support the detection and prediction of future attacks. Lastly, given the fitting time of the Random Forest model, it is possible to adopt `HSE` in run-time.

VI. Conclusions

This short paper presented the *Honeypot Session Enrichment* (`HSE`) prototype, a system to analyse sessions from a live honeypot environment using local KGs and ML models, without relying on external services, with the aim of predicting adversarial behaviours and classifying malware. The `HSE`'s results, although preliminary, showed that it is capable of attracting and capturing attackers, accurately classifying malware, and predicting incoming attacker activity. Although its approach has shown that it can be useful in helping security analysts, `HSE` needs improvements in its different components, and new capabilities in order to improve its precision in predicting attackers' next steps and behaviours.

Acknowledgments

We thank the anonymous reviewers and our shepherd, Sergio Portillo, for the constructive comments that helped improve this paper. This work has been supported by the RU Research Fund and the Frostbyte cybersecurity lab, partially supported by P2030 through project I2DT, ref. COMPETE2030-FEDER-00389100, an ITEA4 European project (ref. 22025), FCT through the LASIGE Research Unit, ref. UID/408/2025, and has received co-funding from the European Union's Digital Europe programme under project NCC-IS (Ref. 101127453).

REFERENCES

[1] MITRE, "Common attack pattern enumeration and classification," 2020. [Online]. Available: https://attack.mitre.org

[2] Y. D. Abbasi, N. Ben-Asher, C. Gonzalez, D. Kar, D. Morrison, N. Sintov, and M. Tambe, "Know your adversary: insights for a better adversarial behavioral model," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 38, 2016.

[3] CAPEC, "MITRE ATT&CK," 2015. [Online]. Available: https://capec.mitre.org/index.html

[4] C. Peng, F. Xia, M. Naseriparsa, and F. Osborne, "Knowledge graphs: Opportunities and challenges," *Artificial intelligence review*, vol. 56, no. 11, pp. 13 071–13 102, 2023.

[5] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier *et al.*, "Knowledge graphs," *ACM Computing Surveys (Csur)*, vol. 54, no. 4, pp. 1–37, 2021.

[6] B. Subagdja, D. Shanthoshigaa, Z. Wang, and A.-H. Tan, "Machine learning for refining knowledge graphs: A survey," *ACM Computing Surveys*, vol. 56, no. 6, pp. 1–38, 2024.

[7] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European semantic web conference*. Springer, 2018, pp. 593–607.

[8] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," *arXiv preprint arXiv:1801.01078*, 2017.

[9] N. Ilg, P. Duplys, D. Sisejkovic, and M. Menth, "A survey of contemporary open-source honeypots, frameworks, and tools," *Journal of Network and Computer Applications*, vol. 220, p. 103737, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S108480452300156X

[10] N. Rastogi, R. Christian, S. Dutta, and Y. Park, "An ontology-driven, dynamic knowledge graph for android malware," in *ACM Annual Conference on Computer and Communications Security*, 2021.

[11] P. Falcarin and F. Dainese, "Building a cybersecurity knowledge graph with cybergraph," in *Proceedings of the 2024 ACM/IEEE 4th International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS) and 2024 IEEE/ACM Second International Workshop on Software Vulnerability*, 2024, pp. 29–36.

[12] Y. Yang and H. Chen, "Construction of domain-specific knowledge graph for advanced persistent threat behaviour analysis and detection," in *Companion Proceedings of the ACM on Web Conference 2025*, 2025, pp. 1480–1484.

[13] J. Kincl, T. Eftimov, A. Viktorin, R. Šenkeřík, and T. Pavleska, "Comprehensive benchmarking of knowledge graph embeddings methods for android malware detection," *Expert Systems with Applications*, p. 127888, 2025.

[14] M. Mohammadzad, J. Karimpour, and F. Mahan, "Cyber attacker's next action prediction on dynamic real-time behavior model," *Computers and Electrical Engineering*, vol. 113, p. 109031, 2024.

[15] P. Camacho Gonzalez, "Probabilistic prediction of the behaviour of cyber attackers," Degree project in Computer Science and Engineering, second cycle, KTH, School of Electrical Engineering and Computer Science (EECS), 2020. [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-282875

[16] Y. Xu and Z. Chen, "Family classification based on tree representations for malware," in *Proceedings of the 14th ACM SIGOPS Asia-Pacific Workshop on Systems*, 2023, pp. 65–71.

[17] M. Soylu and R. Das, "A hybrid graph neural network model for predicting cyber attacks from heterogeneous and dynamic network data," *IEEE Access*, 2025.

[18] Michel Oosterhof and Contributors, "Cowrie: SSH/Telnet Honeypot," 2026. [Online]. Available: https://github.com/cowrie/cowrie

[19] Deutsche Telekom Security GmbH and M. Ochse, "T-Pot 24.04.1," 2024. [Online]. Available: https://github.com/telekom-security/tpotce

[20] H. project team, "Virusshare," 2007. [Online]. Available: https://hihat.sourceforge.net

[21] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder, "A survey on honeypot software and data analysis," *arXiv preprint arXiv:1608.06249*, 2016.

[22] C. Guan, H. Liu, G. Cao, S. Zhu, and T. La Porta, "Honeyiot: Adaptive high-interaction honeypot for iot devices through reinforcement learning," in *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2023, pp. 49–59.

[23] M. Lucchese, F. Lupia, M. Merro, F. Paci, N. Zannone, and A. Furfaro, "Honeyics: A high-interaction physics-aware honeynet for industrial control systems," in *Proceedings of the 18th International Conference on Availability, Reliability and Security*, 2023, pp. 1–10.

[24] N. W. S. Nuutinen, M. Faísco, M. Petrusic, I. Medeiros, and H. P. Reiser, "Evaluating eBPF as an alternative to virtual machine introspection for high-interaction honeypot implementation," in *Latin-American Symposium on Dependable Computing*. Springer, 2025, pp. 238–254.

[25] Microsoft, "eBPF for Windows," https://github.com/microsoft/ebpf-for-windows.

[26] L. Liu, B. sheng Wang, B. Yu, and Q. xi Zhong, "Automatic malware classification and new malware detection using machine learning," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 9, pp. 1336–1347, 2017. [Online]. Available: https://doi.org/10.1631/FITEE.1601325

[27] Z. Xu, S. Ray, P. Subramanyan, and S. Malik, "Malware detection using machine learning based analysis of virtual memory access patterns," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, 2017, pp. 169–174.

[28] A. Djenna, A. Bouridane, S. Rubab, and I. M. Marou, "Artificial intelligence-based malware detection, analysis, and mitigation," *Symmetry*, vol. 15, no. 3, 2023. [Online]. Available: https://www.mdpi.com/2073-8994/15/3/677

[29] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, and S. Venkatraman, "Robust intelligent malware detection using deep learning," *IEEE access*, vol. 7, pp. 46 717–46 738, 2019.

[30] M. G. Gaber, M. Ahmed, and H. Janicke, "Malware detection with artificial intelligence: A systematic literature review," *ACM Computing Surveys*, vol. 56, no. 6, pp. 1–33, 2024.

[31] S. Sharmeen, Y. A. Ahmed, S. Huda, B. Ş. Koçer, and M. M. Hassan, "Avoiding future digital extortion through robust protection against ransomware threats using deep learning based adaptive approaches," *IEEE Access*, vol. 8, pp. 24 522–24 534, 2020.

[32] A. Greenhill, "Virusshare," 2026. [Online]. Available: https://virusshare.com/about

[33] D. Carlin, P. O'Kane, and S. Sezer, "A cost analysis of machine learning using dynamic runtime opcodes for malware detection," *Computers & Security*, vol. 85, pp. 138–155, 2019.

[34] E. Gilliard, J. Liu, and A. A. Aliyu, "Knowledge graph reasoning for cyber attack detection," *IET Communications*, vol. 18, no. 4, pp. 297–308, 2024.

APPENDIX A
SYSTEM LOG EXAMPLE

System Log Example

```
{
  "session":
      "94.***.***.***/15968-sshd-2291",
  "subsession": "none",
  "pid": 15998,
  "ts": 1772367008576812147,
  "type": "file",
  "filename": "/home/admin/.bashrc",
  "fd": 3,
  "flags": 32768
}

{
  "session":
      "94.***.***.***/15968-sshd-2291",
  "subsession": "mal-15998",
  "pid": 15998,
  "ts": 1772367008576813826,
  "type": "exec",
  "filename":
   "/home/admin/4dds1p6j0dgof5t07tifmim5eh",
  "fid": "8.1:5289",
  "args":
   ["/home/admin/4dds1p6j0dgof5t07tifmim5eh"],
  "kind": "TRACKED"
}
```

Listing 1: Example of file access and execution events captured from an SSH session

The log example shown in Listing 1 comprises the following four most important attributes:

- **session**: indicates the IP address of the user and the session id;
- **subsession**: indicates if the event was executed from a main malware process pid or a child pid of this malware;
- **pid**: process id that triggered the event;
- **type**: type of event, where `file` signals the opening of a file; `file_close` signals the closing of a file; `exec` signals the execution of a file; `fork` creation of a child process; and `exit` close of the current process.

Some event types may also have extra attributes: `type:file` relates to file opening and writing (determined by an extra attribute named `written: <bool>`), along with a `filename`. `type:exec` events also carry this `filename` attribute, along with other special attributes such as `args` (which lists the arguments used in command execution), and `kind`, which can have a value of `normal` or `TRACKED`. This differentiates between the execution of regular system files and files uploaded or modified by an external user (i.e. an attacker). Furthermore, when a `TRACKED` file is executed, all events generated by it or its children will be tainted with `subsession: mal-<pid>`, differentiating it from other events, mainly for analysis purposes.

APPENDIX B
ATTACK KNOWLEDGE GRAPH OVERVIEW

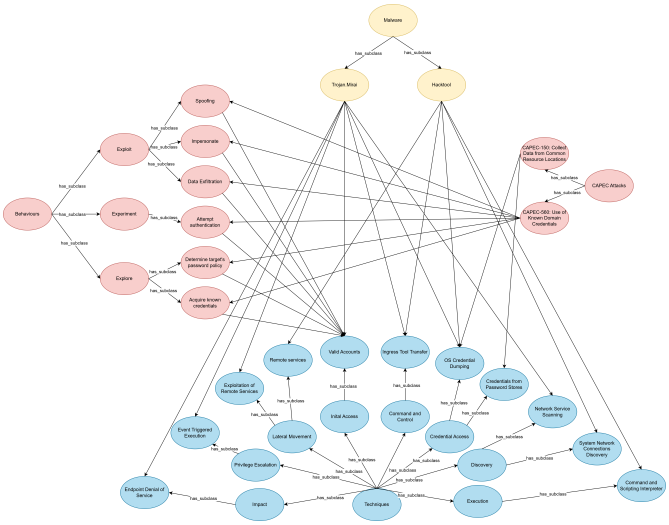

Fig. 2: Simplified overview of Attacks KG

APPENDIX C
EVALUATION METRICS

TABLE V: Evaluation metrics

Metric	Formula
Accuracy	$\frac{\sum_{i=1}^{K} TP_i}{N}$
Precision_i	$\frac{TP_i}{TP_i + FP_i}$
Recall_i	$\frac{TP_i}{TP_i + FN_i}$
F1-Score_i	$2 \cdot \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$