Learning Intrusion Tracking from Simulations

Nikolaos Kakouros, Pontus Johnson, Mathias Ekstedt
Division of Network and Systems Engineering
KTH Royal Institute of Technology
Stockholm, Sweden
Email: {nkak, pontusj, mekstedt}@kth.se

***Abstract*—This paper presents an approach to using Graph Neural Networks (GNNs) for intrusion detection, leveraging the MAL simulator for generating synthetic attack data. The proposed system utilizes telemetry events and their associations to predict the state of compromise in a network.**

***Index Terms*—cyber security, graph learning**

I. Introduction

A. Intrusion detection has a long history

Intrusion detection systems are an indispensable part of modern security infrastructure for detecting unauthorized access and anomalous behavior in computer and network systems. Rule-based detection is one of the earliest methods employed for this task, where manually written or generated rules match network traffic or system behavior against a database of known signatures. Signature matching is deterministic, leading to very low false positive rates. However, the reliability of these systems is contingent on their signatures being up-to-date and limited to what is included in their signature database. An alternative approach has been anomaly detection which promises to detect unknown or zero-day attacks by identifying deviations from benign system behavior. Originally, profile templates were used that were populated with system logs and metrics from the system under normal operation3. This method was limited by the available profile templates and the simple statistical descriptions of logs and metrics. Following advancements in machine learning, unsupervised learning was employed to replace profile-based detection with predictive models. Given a mixed, unlabeled dataset of benign and malicious logs, these models learn to associate each log with the appropriate class they belong to (benign or malicious). These methods have the potential to uncover hidden patterns in data that are indicative of behavior anomalies, but do not generalize well in new environments or even slight changes in the behavior of the underlying system.

B. A lack of labeled training data

A reason behind the shortcomings of anomaly detection is that anything that constitutes sufficient deviation from normal behavior will be flagged as suspicious behavior. Such deviations often appear without malicious intent, such as when new uses enter the system, seasonal habit changes occur, or system architecture changes.

Supervised learning can produce predictive models to achieve high precision and recall across various circumstances and systems, provided the training dataset covers these scenarios. The model is trained via a feedback process that corrects it for any wrong label applied to the input data.

C. MAL: We have a simulator

Introduce the MAL simulator developed for generating synthetic attack data.

D. GNNs are suitable DNNs

Explain why Graph Neural Networks (GNNs) are appropriate for this application due to their ability to handle graph-structured data.

E. Outline

II. Related Work

A. Intrusion detection

Review existing intrusion detection techniques and their limitations.

B. MAL

Discuss the features and capabilities of the MAL simulator.

C. GNNs

Overview of GNNs and their applications in various domains, including cybersecurity.

D. Provenance graphs

Definition of provenance graphs and examples from the literature.

III. Requirements for GNN Training

A. GNNs need a graph

Explain the necessity of graph structures for GNNs.

B. The information needed should be in the neighborhood

Detail the requirement for relevant information to be accessible within the graph neighborhood.

C. Additional requirements...

IV. MAL 2.0

Explain how MAL 2.0 extends MAL 1.0

A. Telemetry events

- Define telemetry events and their role in the proposed system.
- Telemetry events are connected to each other sequentially through a next association.
- Telemetry events need to be connected to attack steps through a trigger association, which needs to include an accuracy estimate.
- Telemetry events can also be associated to assets, in the case of our prototype, accounts and IP addresses.

Here are examples of telemetry events:

```
asset VM {
    | root_shell(IP ip)
        e meterpreter_detected(IP ip) [0.1]
}
```

where the event meterpreter_detected will be emitted by some threat detection system in 10% of the cases where the attacker has gained root shell access to the VM. Note that we no longer have false positives, because they will be generated by user emulators. Here is another example:

```
asset Bucket {
    | list_objects(Account account, IP ip)
        e list_object(Account account, IP ip) [1.0]
}
```

where the event list_object will be logged with 100% accuracy if the attack step list_object is compromised.

B. Next associations

- Describe the sequential linking of telemetry events.
- One good thing with creating a dedicated time step graph as suggested above, is that it does not need to include the complete attack graph, as it can be limited to the nodes that are N-neighbors to the log events.

C. Trigger associations

- Explain the associations between telemetry events and attack steps, including accuracy estimates.
- Note that we no longer need confusion matrices as the false positives will be generated by the legitimate events generator (user emulation).

D. Information association

- Discuss how telemetry events are connected to assets such as accounts and IP addresses.
- Telemetry events need to be connected to attack steps through a trigger association, which needs to include an accuracy estimate.

V. Example

A. A simple MAL spec

Listing 1 shows the simple MAL specification we will use in the subsequent sections to demonstrate how training of the GNN and intrusion detection work. The specification models a tiny part of typical cloud storage infrastructure. A user uses an IP address to connect to the cloud provider and assumes an identity (modeled as `Account`) to authenticate. The exact authentication mechanism is unimportant, and we consider `Account.assume` to perform authentication as well. The user can then list the available buckets and the objects they store and download the stored objects, provided that the used account has read permissions on the accessed buckets.

Listing 1. Example MAL spec used

```
#version: "0.0.0"
#id: "example"

category Cloud {
  asset IPAddress {
    | assume
  }

  asset Account {
    | assume
      -> projects.list_buckets,
         projects.buckets.list_objects
  }

  asset Project {
    | list_buckets
      ! log_project_list_buckets(Account account,
          ↪ IPAddress address)[1.0]
      -> buckets.attempt_list_objects
  }

  asset Bucket {
    | attempt_list_objects
      -> list_objects

    & list_objects
      ! log_bucket_list_objects(Account account,
          ↪ IPAddress address)[1.0]
      -> objects.read
  }

  asset Object {
    | read
      ! log_object_read(Account account, IPAddress
          ↪ address)[1.0]
  }
}

associations {
  Account [account] 1 <-- assumes --> * [addresses]
      ↪ IPAddress
  Project [project] 1 <-- contains --> * [buckets]
      ↪ Bucket
  Account [s_reader] * <-- storage_read --> *
      ↪ [projects] Project
  Bucket [bucket] 1 <-- stores --> * [objects]
      ↪ Object
}
```

The above MAL spec includes telemetry event definitions for various attack steps.

B. A simple instance model

Listing 2 shows a simple instance model in a self-explanatory pseudo-language. The corresponding instance graph is shown in fig. 1. The resulting attack graph is shown in fig. 3. The attacker starts at the `assume` attack step (of the `pontusj` account asset) in the center.

Listing 2. Example instance model

```
Project project_26
Bucket bucket_49
Bucket bucket_75
```

```
Object data_113
Object data_118
Object data_231
Object data_236
Object data_321
Object data_546
Account pontusj
Account mathiase
IPAddress 17_233_12_98
IPAddress 130_237_44_15

uses(pontusj, 17_233_12_98)
uses(mekstedt, 130_237_44_15)
storage_read(pontusj, project_26)
contains(project_26, [bucket_49, bucket_75])
stores(bucket_49, [data_113, data_118])
stores(bucket_75, [data_231, data_236, data_321,
    ↪ data_546])
```

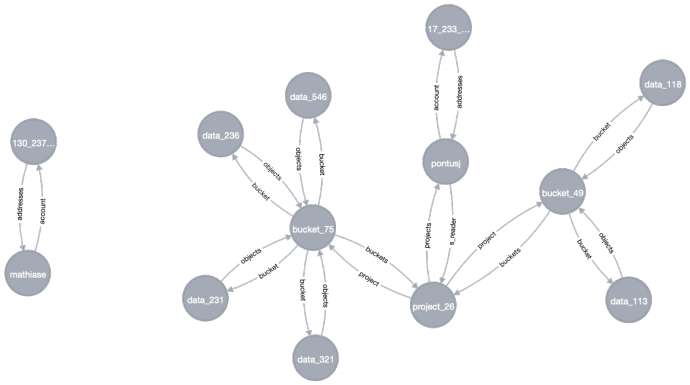

Fig. 1. The example instance model graph.

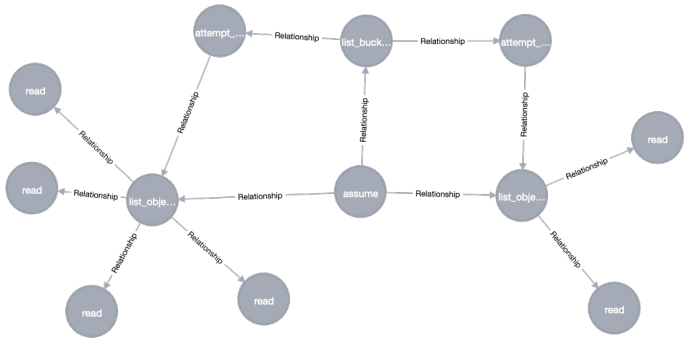

Fig. 2. The resulting attack graph for the example instance model.

C. The resulting attack graph

From the attack graph and an event log (synthetic or real), at each time step, we can generate a graph to present to the GNN 3.

D. The resulting GNN

Show the graph structure fed into the GNN for training 4.

E. Results from a small training round

Provide preliminary results from a small-scale training session with the GNN.

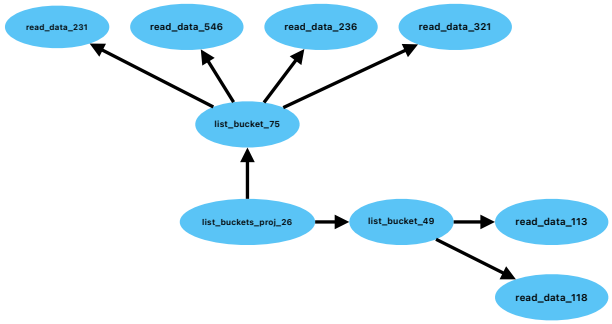

Fig. 3. The resulting attack graph.

VI. Simulation

Now, at each simulation step, events are triggered according to the specification. For instance, when the attacker compromises bucket_75.list_objects (list_bucket_75 in Figure 4), the attacker now needs to provide the employed account and IP. A list_object telemetry event is automatically produced, containing information about the concerned asset and attack step, as well as the account and the IP address. In this manner, the simulated log is generated.

VII. Graph Neural Network Training

From the attack graph and an event log (synthetic or real), at each time step, we can generate a graph to present to the GNN. Each event is connected with a trigger association to the associated attack step (as defined in the MAL specification), with an information associations to the employed account and IP, and a next association to the most recently triggered event, thus producing a graph like the one in Figure 4.

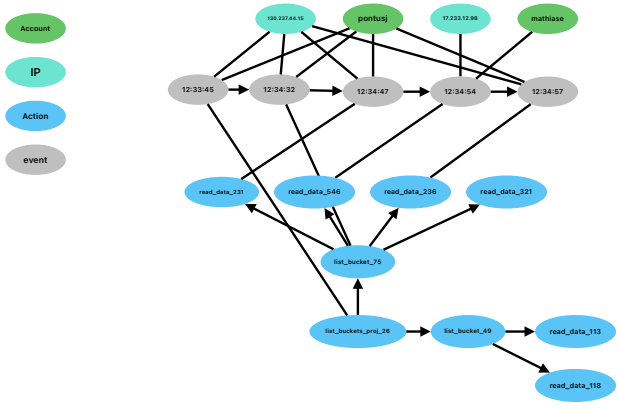

Fig. 4. Input graph for GNN training.

As mentioned, this graph should be useful to the GNN, as all the information relevant to assess the state of compromise of a node is within the neighborhood.

A. Future work

Suggest directions for future research and improvements.

VIII. Conclusion

A. Summary

Recap the main points and findings of the paper.

B. Impact

Discuss the potential impact of the proposed system on the field of intrusion detection.

C. Next steps

Outline immediate next steps for further development and research.

IX. Discussion

A. Challenges and limitations

Discuss potential challenges and limitations of the proposed approach.

- One consequence of this encoding is that the fan-out of certain nodes will become quite large.

B. Scalability

Address concerns about scalability and how they are mitigated.

- Parts of the graph where no events are recorded during the time window can therefore be omitted.