Using Indicators of Behavior to Contextualise SQL Injection Attacks

Robert Chetwyn, László Erdődi

University of Oslo, Department of Informatics,
Gaustadalleen 23B, N-0373 Oslo, Norway
laszloe@ifi.uio.no; laszloe@ifi.uio.no

Abstract: This paper investigates the enhancement of behavior-based cyberthreat hunting through the application of ontologies and semantic reasoning, specifically targeting SQL injection (SQLi). We use the OWL 2 Web Ontology Language and Resource Description Framework (RDF) data models to transform SQL injection event data into a semantically enriched format; suitable for advanced reasoning and context-aware analysis. Our approach introduces a detailed methodology for developing Indicators of Behavior (IOBs), which involves mapping SQLi events to ontological concepts and enriching these events with relational contexts derived from the ontology. This process enables the abstraction and summarization of security events into higher-level behaviors, offering a comprehensive understanding of attack sequences, adversary objectives, and potential next steps. Through the integration of ontological modelling and semantic reasoning, we establish a framework for identifying complex cyber threats. Comparative analysis indicates that our context-aware system provides meaningful context to security events, beyond the capabilities of traditional single-event signature-based methods and basic binary outputs from many machine learning classifiers, thus potentially leading to more effective defenses against SQL injection attacks.

Keywords: Indicators of Behavior; IOB; Threat Hunting; Cybersecurity; SQL Injection; Symbolic AI; Ontology

1 Introduction

The evolving cyber threat landscape has shifted from simplistic attacks targeting individual systems to sophisticated campaigns and threats that demand more proactive and contextualised defence strategies [1] [2]. Traditional Indicators of Compromise (IOCs), which comprise technical artifacts such as IP addresses, file hashes and unusual processes [3] are crucial forensic elements that support the indication of malicious activity but provide isolated instances of such activities. Without supplementary context, these IOCs do not reveal the underlying adversarial intent. It is the role of security analysts to process, investigate, contextualise and triage security alerts generated by these IOCs. When investigating alerts, security

analysts must answer "*what*" occurred, "*how*" something occurred and potentially "*why"* it occurred. Alert fatigue and manual investigations are a common challenge for security analysts [6]. IOCs evidence the *"what"* and only evidence the *"why"* when correlated with some threat intelligence [4] [5]. IOCs do not typically explain the *"how"*, requiring analysts to leverage knowledge from other sources.

Machine learning driven approaches for detection and anomaly clustering models commonly label or group security events, providing evidenced *"what"* contexts to security events but commonly lack an explained "*how"* and "*why"* context [6]. Without supportive Explainable AI methods, it is still the responsibility of security analysts to derive the missing contexts. Large language models are an emerging paradigm that show promise in providing explainable summaries to IOCs but face challenges in scalability, context, hallucinations, and concept understanding.

To overcome these limitations, the concept of Indicators of Behavior (IOBs) has been introduced as a semantic extension to threat hunting and cyber threat intelligence [7] [8]. IOBs focus on contextualising the intent behind sequences of security events and the causality of these events to reveal attacker behaviors and strategies, answering the "*what" "why"* and *"how"* of a security event sequence [4] [5]. This behavioral, intent based approach enables reasoning about the adversary's objectives in a holistic manner, providing explainable and contextualised event sequences for further investigation.

In this work we demonstrate how IOBs using semantic web technologies, such as ontologies and reasoning, can be modelled and applied to web application security events. By transforming these security events into semantically enriched representations we demonstrate how IOBs go beyond manual investigation and detection logics by providing reasoned, contextualised and explained event sequences to security analysts.

For this work we have formulated the following research questions:

- How can ontologies and semantic reasoning be applied to improve the context of SQL injection behaviors?
- What are the benefits of using Indicators of Behavior (IOBs) enriched with ontological contexts in threat hunting?
- How does our proposed methodology compare to traditional signature-based and machine learning approaches regarding context?

This paper is organised as follows. Section 2 provides background information on SQL injection attacks and foundational concepts relevant to our approach. Section 3 reviews related work, highlighting gaps in current detection methods. Section 4 details our ontology-based methodology for modelling Indicators of Behavior from SQL event data. Section 5 presents experimental results from capture-the-flag scenarios demonstrating the effectiveness of our approach. Section 6 evidences our results. A discussion can be found in Section 7. Finally, conclusions are drawn in Section 8.

2 Background

2.1 SQL

Structured Query Language (SQL) is a domain-specific language used for querying and managing relational databases. SQL database management systems (DBMS) are widely adopted across industries due to their ability to efficiently manage large volumes of structured data. Common SQL DBMSs, such as MySQL, PostgreSQL, Microsoft SQL Server, and Oracle Database, are used in a variety of applications, from small-scale websites to enterprise-level systems.

The SQL syntax has a logical order for which the syntax must follow. Each syntactical element of this logical order aids us in interpreting the behavior of the query.

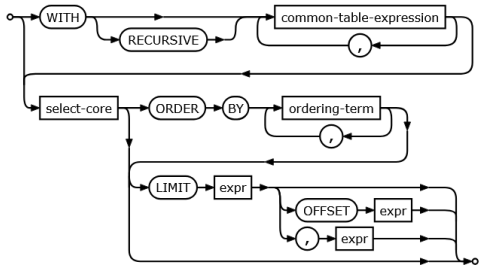

Figure 1
Example of the logical flow in a simple SQL SELECT query [10]

An example of how the SQL syntax logically flows can be seen in Figure 1. Any user of SQL, adversarial or non-adversarial, is bound to the scope of the SQL syntax and its logical flow. SQL queries in general are used to retrieve, update, insert, or delete data. The SQL syntax is based on simple, declarative commands that make it easy to understand and use.

An important characteristic of SQL queries is that input data is often used to construct the query. A simple example for validating user credentials during login would be:

select userid from users where username=input1 and password=input2

If the SQL input does not have appropriate input validation, then an adversarial SQL query can have serious consequences. SQL injection is a type of security vulnerability where an attacker manipulates SQL queries by injecting malicious input into a query's parameters [10]. This can allow unauthorised access to a database, enabling attackers to view, modify, or delete sensitive data. By crafting specific input, attackers can alter the intended behavior of the SQL query,

potentially bypassing authentication, extracting confidential information, or even executing administrative database commands.

2.2 Cyber Threat Hunting

Threat hunting is a proactive approach aimed at identifying potential threats that have evaded automated detection systems. It involves searching through network activity, security logs and other correlated data to uncover suspicious behavior that may indicate a compromise or confirm a hypothesis [11]. The key aspect of threat hunting is proactivity. Threat hunting teams actively seek out IOCs, IOBs that can be aligned with the tactics, techniques, and procedures (TTPs) of attackers. This proactive approach, doesn't rely on responding solely to alerts generated by detection systems. In this research we use semantic IOB based threat hunting approaches to reason over our security event data generated related to MySQL servers. We chose MySQL servers since previous semantic IOB based threat enhance these threat hunting approaches twofold. Firstly, we transform the SQL operator syntax into a set of IOB analytics that infer the adversary's behaviour. Second, we contextualise the database assets. By using an ontology, we are able to perform a hybrid threat hunting approach in a reasoned, machine-readable and semi-automated way.

2.3 Indicators of Behavior

IOBs are intent-based threat hunting analytics that focus on identifying the context or purpose behind a series of security events [4] [5] [7] [8]. This intent is the goal an adversary is trying to achieve or what the inferred goal is based on the semantics of the interconnected security events. They aid in establishing context to the individual security events and what they aim to achieve in a broader context, rather than an individual tactic or technique.

Intent-based analytics can adapt to unknown threats by recognising patterns of activity that indicate malicious intent, even if the exact techniques haven't been seen before. By focusing on what requirements are needed for the malicious intent, how it is achieved and how it can be inferred is an adaptive approach compared to single event detections. By reasoning over the syntax of the SQL query we can infer what that individual query is trying to achieve and its wider context with other related events.

2.4 Ontologies

Ontologies are a set of concepts and semantics with associated subjects that define a domain. The RDFS [12] and OWL 2 ontology [13] schemas provide a standard for transforming security event data into machine-readable semantic data. Our

research enhances the threat hunting process by transforming security event data into semantic data to semi-automate the threat hunting process and perform reasoning over this semantic data. The outputs of our findings produce reusable knowledge.

3 Related Work

3.1 Ontologies

An ontological approach for IOBs has been proposed in [4] and [5]. Whilst their approach can abstract behaviors from low-level security events, the approach is limited to Windows Sysmon logs. We extend this approach by applying the methods to the SQL domain. An ontology for SQLi prediction and prevention is proposed by [14] however their explainability and abstractions are vague and designed for simple classification. An ontology for detecting blind SQL injection is proposed [15] but the context is focused primarily on the attackers toolset. Whilst these ontology approaches have been proposed within the SQLi domain, they lack the context around the nature of the attack and the context of the assets being attacked and limited contexts around the attack origins. Within our work we contextualise these organisational assets alongside the behaviors to determine what an adversary is trying to achieve.

3.2 Knowledge Bases

Contexts and information sources provide both complexity and confusion if not properly sourced. Examples include the explainability of logged security events and the information sources relevance.

Knowledge bases such as MITRE ATT&CK [16], Common Attack Pattern Enumeration and Classification (CAPEC) [17] and Common Weakness Enumeration (CWE) exist for contextualising adversarial activities and their relevance to an organisation. Whilst providing a generalised descriptive abstraction for general knowledge, they lack more granular contexts for many security events. In this instance we focus on SQLi.

A logged SQLi event can be attributed to Technique T1190 [19], several CAPEC IDs including CAPEC-66 [20], CAPEC-108 [21] and several CWE IDs such as CWE-89 [22]. Other domain sources such as OWASP also provide generalised information for SQLi [23].

The Common Vulnerabilities and Exposures (CVE) [24] knowledge base provides a plethora of common vulnerabilities and exposures for systems and software in an organisation and how to possibly detect them. However, if there's a deviation in the adversary's exploit these detections might not trigger. Moving from indicators of compromise to indicators of behavioural intent will provide better contexts and oversight into what the adversary is trying to achieve.

Whilst providing a plethora of information sources for security analysts they have limited contexts as to why a specific SQLi event is relevant, if its a true positive or a false positive and its relevance to internal organisational assets. Similarly there are limited detection analytics provided for such cases. Utilising a knowledge graph and ontology to represent behavioural intent and additional knowledge contexts allows analysts to both infer behaviours and query for event data associated with behaviours.

3.3 Machine Learning Driven Detection

Explainability is a challenge with machine learning outputs. Whilst machine learning approaches increase the detection rate of SQL injection attacks in an automated and scalable way they often lack the descriptive contexts during an investigation phase. A binary classification of malicious or not malicious still requires a manual investigation to ascertain context, provenance and relevance. A machine learning framework for SQLi is proposed by [25]. Whilst their detection rates are promising there's a lack of granular context around these detections and prevention, the dataset is unavailable for post analysis or threat hunting. Similar challenges regarding these contexts can be found in other approaches such as [26] [27] do not provide their data and are simple classifiers. Research conducted by [28] uses a binary labelled dataset to classify and prevent SQLi attacks but lack ground truth as to why. An approach by [29] provides visualisation to the ML outputs but these outputs again binary classifiers. Overall, these methods are a tool to identify key security events in a forensic investigation. Our approach focuses on temporal chain of events; an adversarial procedure. Therefore, we correlate events and reason over these events to determine their context. Our method transforms these security events into semantic data for reasoning and semi automated threat hunting with context to enable more effective threat hunting to provide more mature and useful contexts for threat hunting and forensic investigations.

3.5 Comparisons Against AI Methods

The ontology presented in this paper contextualises behavioral patterns by explicitly modelling SQL syntax with mutable, organisation-specific context. Ontological rules can be tailored or extended as organisational requirements or attack patterns evolve, without requiring retraining on volumes of labelled attack data. Comparing

this ontology based approach against other machine learning approaches demonstrates the value of an ontological based approach for context enrichment.

Supervised Learning achieves high accuracy when large, representative labelled datasets are available. However, these models are strongly influenced by their training distributions and require retraining for each new domain to capture organisational context or any shifts in application logic [38] [39].

Unsupervised Learning is capable of identifying unusual behavior but generates more alerts that security analysts must investigate, contextualise and triage. Similarly, these unsupervised learning models are only effective when a baseline has been measured [40]

Large Language Models have been proposed for SQL injection detection [fart] but these models are susceptible to hallucinations, requiring careful prompt engineering, context updates and introduce challenges in holistic investigation. Rather than continuously establish the context and frame this context in a Large Language Model's prompt we use semantic reasoning.

4 Methodology

In this section, we outline the data collection process, ontology design and reasoning techniques used to detect SQL injection behaviors. We describe how event logs are transformed into semantic data and how IOBs are modelled in the ontology to support threat hunting.

4.1 Data Generation

Any interaction with an entity is logged as a security event, regardless of benign or malicious intent. These logged events are used as our dataset. We utilise Apache [30] logging to record each interaction. The logs use the default LogFormat Directive [30] providing the GET and POST request events on the web server alongside the IP address. Since we are capturing and reasoning behavioural intent from SQL queries this is all we require from the logs. We anonymise each unique IP address to a random integer value. The logs containing the original IP address of the adversary is unavailable for public access.

We preprocess each security event and transform it into semantic data for use in our designed ontology. To achieve this we use an automated Python script that asserts security event data to their relevant semantics.

For analysing the semantic data we utilise the Stardog IDE platform [31]. We chose Stardog because it provides a suite of visualisation tools, data mapping, IDE, access control and advanced reasoning. Similarly, this suite is optimised for handling

larger datasets. Protégé [32] was incapable of handling our datasets without long delays, making the approach unrealistic in a real world scenario.

The following process summarises the data generation and transformation for this research:

- Participants establish a session with our Capture the Flag (CTF) environment and interact with the environment.
- All participant interactions are captured as HTTP GET/POST requests and logged by the CTF entities.
- All logged events are extracted and transformed into semantic data.

4.2 Creating the Ontology

To semantically represent different domain concepts, contexts and perform reasoning, we design an ontology for this threat hunting approach. An ontology is a formal representation of a set of concepts applied to a domain. In this research, we create a formal representation of the SQL syntax and our entity contexts for the purpose of inferring behavioral intent. Similar approaches have been able to achieve reasoning over complex and variable security event data in other contexts [4] [5]. We apply this approach to the domain of SQLi. We utilise the OWL 2 [13] semantic web ontology language for expressivity and advanced reasoning. The benefit of OWL 2, comparatively to RDFS, is that we can be more expressive in our contextualised model and enable more efficient querying across the knowledge graph with restrictions [33]. Similarly, we are able to create equivalence concepts [33], an important aspect of reasoning if a user tries to encode values.

The ontology we created provides a generalised semantic model of the SQLi domain and a knowledge graph. The knowledge graph is the CTF event data transformed into semantic data, providing a method to further contextualise and reason across this event data.

4.2.1 Modelling the Organisational Environment

For entity based threat hunting we require an entity context. These entities are the critical assets within our organisation. To provide this entity context we utilise our own hosted CTF environments. We chose our own hosted CTF environments to provide this context since there's a limitation on available SQLi datasets with context to conduct mature threat hunting. Datasets such as [34] provide the SQLi payload but only provide a binary classifier. These datasets are more useful for supervised learning detection methods, whereas we are conducting a proactive threat hunt for adversarial behavioral activity.

Each CTF environment has a set of entities that a participant tries to exploit, in our environments these entities are web servers and databases. We then contextualise these assets within the ontology. In doing so, these contexts provide more relevance for testing our approach rather than generic labelled datasets. A query targeting a known organisational entity is of higher priority than a general scan on the network.

4.2.2 Representing Events in the Knowledge Graph

Every logged security event is a subject stored in the knowledge graph. Each security event stored in the knowledge graph is a member of the *Event* class. This *Event* class is where most rules, relationships, and reasoning interact throughout the knowledge graph. The *Event* class contains several object properties [13] that are used to infer the *Event* class when present. These properties include the *eventID*, *command* and SQL operator clause properties. The *eventID* property is a chronologically ordered eventID that is assigned during pre-processing. The *command* property is the logged HTTP GET/POST request. This *command* property is the primary source of behavioral context.

If an SQL operator is present in the HTTP GET/POST request it is assigned an SQL operator property. These SQL operator properties are boolean properties and a domain of their associated low-level behavior. This provides a method to infer that an event is a specific IOB. Each operator property is assigned an IOB as its domain. This provides a method to classify *Events* as *Behaviors*.

4.2.3 Relating Events in the Knowledge Graph to Form Event Chains

Rather than reasoning only on single events, we can establish better contexts and meaning over related security events. We relate events in the knowledge graph sequentially - forming a temporal chain of next pair security events. These temporal event chains allow us to traverse events in sequence, accumulating and tracking the behaviors within each event and providing a holistic context. We name this relation as *related* in the ontology.

For our use case, we utilise the anonymised IP addresses for correlation. Each anonymised IP address represents a user session. For each event that was instantiated in a user session we relate it's next logged event in the session. This reduces the computational complexity of transitive relations and avoids a combinatorial explosion of event sequences. Similarly we avoid cycles in reasoning logic with this approach by traversing through the chain in a single direction.

Simply if events $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$ are instantiated by user $\mathcal{U}$ then $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$ are related.

We also relate events by their CTF scenario they were detected in via the relationship *detectedIn*. This allows to see the overall behavioural activities in a given scenario. In other works [4] [5] this relationship specifies the machine the event was logged on. Due to there only being a single web server for each of our

scenarios we instead used this grouping for simplicity. Relationships for different use cases can be implemented using this method.

4.2.4 Modelling IOBs in the Knowledge Graph

Modelling IOBs is a multi-tiered approach [4] [5], utilising low, medium and high-level IOBs. Low-level IOBs are the individual SQL operators and their meaning. Medium-level IOBs are a causal combination of low-level IOBs. High-level IOBs are more detailed adversarial behaviors. The tiered system is interconnected where medium-level IOBs are a combination of low-level IOBs and high-level IOBs are a combinatino of medium-level IOBs.

An IOB is its own class and a subclass of Behavior. This provides a method to query for any *Event* that is a *Behavior* or for any *Event* that is a specific IOB. Each IOB has an abstract description that explains what the user is trying to achieve and an explanation as to what SQL operator is required. We assign an individual to each IOB. Each individual serves as a node in our relationship via the *partOf* and *indicatesBehavior* relationships. The purpose of the *indicatesBehavior* relationship is to simply identify which *Events* indicate a specific *Behavior*. When this *indicatesBehavior* relationship is established we state the *Event* is also *partOf* the *Behavior* it indicates.

The purpose of the *partOf* relationship is to serve as a direct and indirect relationship used for reasoning medium and high-level IOBs. If two *related Events* are *partOf* some *Behavior* then each event is inferred as being *partOf* their relationships *Behavior*. This allows us to model chains of behaviors.

The primary difference between both relationships is that *indicatesBehavior* is only the *Behaviors* and *Event* indicates whilst *partOf* is inherited. We automate the process of assigning these relationships with the Semantic Web Rule Language (SWRL). SWRL is a syntax for expressing logical inference rules in semantic web technologies. Examples of these SWRL rules and the syntax can be seen in the following rules:

SWRL Inference Rule for partOf direct relationship when an event indicates a specific behavior:

$$Event(?\,e) \wedge indicatesBehavior(?\,e, ?\,b) \rightarrow partOf\,(?\,e, ?\,b)$$

SWRL Inference Rule for partOf indirect relationship when two related events indicate a specific behavior:

$$Event(?\,e1) \wedge partOf\,(?\,e1, ?\,b1)) \rightarrow Event(?\,e2) \wedge partOf\,(?\,e2, ?\,b2) \wedge related(?\,e1, ?\,e2)$$

For explainability we can query the context behind an event indicating a specific behavior. Determining what behavior is a part of another another behavior and what those related connections are. In using this approach we go beyond basic binary

classifiers by contextualising the interconnectivity of these events, their relations to the modelled behaviors and the explainable contexts for each logged event. The following example inference rule shows how we develop this concept and establish behavioural context around interconnected events.

4.2.5 Contextualising Organisational Assets in the Knowledge Graph

For entity based threat hunting it is important to identify valuable and high risk assets to find potential threats [35]. In doing so, threat hunts focus on these assets as its scope - prioritising the high risk assets and increasing the security posture. In our scenarios these assets are the databases, their contents and the expected user interactions.

We keep an inventory of the database assets that our CTF scenarios contain, including their schema, tables and columns and include them within a *KnownSchema* class- this acts as our hypothetical organisation. When an incoming query tries to select data from any schema we define the relationship as *Event from Schema*. The specified schema individual is then classified to one of three classes. These classes are as follows:

- **KnownSchema** - Any known database schema is a collection of tables and columns present in the database.
- **UnknownSchema** - Any unknown database schema is a requested collection of tables and columns that are not present in the database.
- **InformationSchema** - The schema that provides database metadata [19]
- **Table** - A table within a schema.
- **Column** - A column within a table and a schema.

We can then infer the following:

SWRL Inference Rule for Event Selecting from Schema Classification

$$Event(?\,e) \wedge from(?\,s) \wedge KnownSchema(?\,s) \rightarrow fromKnownSchema(?\,e, ?\,s)$$

Whilst this relationship is not mandatory it adds a level of granularity if required for manual threat hunting queries.

Beyond entity driven threat hunting, contextualising our database assets works in tandem with ISO 27001 Annex A 5.9 - Inventory of information [36] and other associated assets, providing a contextualised asset for any risk register the organisation maintains. Similarly, whilst specific signature-based detection configurations can be applied to specific assets they can't be reasoned on. By defining the context of our assets to our organisation we can integrate IOBs into this context.

Due to the limited scope of our user interactions, we model the user assets with simple concept classes. We define our inbound network requests to be sourced from *KnownUsers* or *UnknownUsers*. The distinction being, known users are a collection of authorised users for an asset.

4.3 Inference Rules for IOBs

We implement a variety of user defined rules to extend the basic ontology inferences and indicate IOBs. These rules are a combination of SQL keywords, the context of what the operators are trying to achieve, any assets they are trying to interact with and the context of the user.

Each event containing SQL syntax will at minimum have a low-level IOB indicated with it. Take the following example log from a subset of our captured date:

$$\text{username} = \text{admin};\ \text{passwd} = \text{aa}' \text{ AND } 6131 = 4759 \text{ AND } '\text{qJlo}' = '\text{qJlo};$$

The example event contains the ′ escape syntax, the AND operator syntax and a TRUE/FALSE operator syntax. These individual operators are extracted from the events *command* data property for reasoning and used as a classifier. Individually these SQL syntactical operators represent limited reasoning but when combined form a basic IOB which we use for classification. This IOB abstraction is "user is trying to exploit a true/false statement" and is represented with a unique IOB ID. This abstraction is represented in a reasoning rule.

We can then query for any Event X that is Behavior A or indicatesBehavior A returning both the grouped classified events and/or their indicates behavior neighbours.

5 Experiments

We use each CTF scenario as an individual organisational network. Each CTF scenario has its own set of logged security data, a unique challenge flag and a set of assets. We use the challenge flag as a domain-specific true positive of compromise for our use case. These logs contain malicious and non malicious participant data.

Within each scenario we conduct a structured threat hunt for all entities in a given CTF scenario, returning all attributed behaviors to logged events. We then pivot on the returned behaviors and associated events for more in-depth analysis. This process enables entity and behavior based threat hunting within the knowledge graph.

6 Results

Here we report on the application of the IOB framework applied to two capture-the-flag scenarios that simulate SQL injection attacks. This section presents the inferred behavioral patterns and analysis of event chains that illustrate the framework's effectiveness.

6.1 Hunting for Behaviors in Scenario 1

For Scenario 1 the participant is required to enumerate the database content, pivot on this discovered content and find the username and password stored within the database. To achieve these steps we define the following set of low-level behavior abstractions:

- **MB1050** - User is trying to enumerate the database tables via the Information Schema. The Information Schema provides information about the tables.
- **MB1051** - User is trying to enumerate the columns via the Information Schema. The Information Schema provides information about the columns.
- **MB1053** - User is trying to enumerate the schemas in the database via the Information Schema. The Information Schema provides information about the schemas.

We state that if any set of events indicate either of these three medium-level IOBs and originate from the same user then classify each event as high-level *HB1005*. Behavior *HB1005* is defined as *the user has attempted enumeration of the database structure, including columns, table names and schemas*.

Finally we analyse the related events using SPARQL property paths to generate an event chain. If the event chain is related to a successful login classify the whole set as *HB1006*. This level of abstraction is *an unauthorised login with legitimate credentials after several enumeration attempts on the database*.

Querying for all events classified as the high-level behaviors returns the low-level event data that has been abstracted.

Overall, we can use the same analytics and reasoning to discover each CTF participant that successfully bypassed the system. The SPARQL property path provides the procedure the participant took, including the prose description of each behavior for further context.

6.2 Hunting for Behaviors in Scenario 2

In this CTF scenario we threat hunt for any activities detected in *CTF scenario 2*. We identify our entities as any database object in the *KnownSchema*. These entities contain sensitive information. We want to identify interactions with these assets that exhibit unauthorised access. In our use case we define unauthorised access as any interactions originating from an *UnknownUser*.

To achieve this as a semantic threat hunt, we define a high-level behavior with identifier *HB1001*. This abstract prose description is the following:

An unauthorised user has interacted with a known database schema and accessed a sensitive data asset within this schema.

The logical semantic rule of *HB1001* analyses any *Event* that is instantiated by an *UnknownUser* that interacted with any *Object* within a *KnownSchema* via a *SELECT* statement. This *Object* may be any *Table* or *Column* that contains the sensitive data asset. As we are aware of this sensitive d ata asset is within our infrastructure we specify that the *Table* must contain the *tableName* attribute with the value *secretflagtable* and the *Column* must contain the *columnName* attribute with value *flag*.

This rule is automatically transformed into a SPARQL query as seen in Figure 2. This query uses the SPARQL syntax to interact directly with the ontology similar to how SQL interacts with a database. All analytics are stored as reusable queries in the knowledge graph. The query as displayed in Figure 2.

```
SELECT * {
    ?event rdf:type sql:Event ;
           sql:command ?command .
    ?event sql:byUser ?user .

    ?event sql:from ?table .
    ?table sql:tableName ?tableName .

    FILTER CONTAINS(?tableName, "
    secretflagtable") .
    FILTER REGEX(?columnName, "flag")
}
```

Figure 2
SPARQL query for detecting behavior HB1001

This query returns positive confirmation of a set of events matching reasoned behavior. When filtering on the distinct unauthorised user set we see a total of 12 unique users have achieved their goal of accessing our sensitive information entity for further investigation. One such investigation is the adversary's procedure. To generate a temporal event chain for a single user we use the following SPARQL query seen in Figure 3.

```
SELECT ?eventID ?event ?user ?command (
    GROUP_CONCAT(?behaviorValue; separator=
    ", ") AS ?behavior) {
    ?event sql:related* sql:f83eea6c-58af-4
    edf-af80-3e99da0da741 .
    ?event sql:eventID ?eventID .
    ?event sql:byUser ?user .
    ?event sql:command ?command .

    OPTIONAL {
        ?event sql:partOf ?behaviorValue .
    }
} GROUP BY ?eventID ?event ?user ?command .
  ORDER BY ?eventID
```

Figure 3
SPARQL path query for identifying behaviors leading to event classified as behavior HB1001

This query identifies a path of zero or more occurrences of any related event that terminates at the defined URI: *f83eea6c-58af-4edf-af80-3e99da0da741*. This URI is the returned result from *HB1001*. This query then orders *Events* by their *eventID* attribute - a chronologically assigned eventID for each logged event. The returned result of this path query provides security analysts with a procedural event chain showing each procedure the adversary took to finally achieve their goal. Similarly it contains all benign and malicious related adversarial events.

Reusing our analytics for our independent dataset, that represents attacks coming from a sophisticated adversary, we were able to repeat and identify the correct security events. Whilst the adversary's procedure varied from the previous participants, the SQLi syntax driven IOBs were correctly assigned and asserted.

7 Discussion

This section discusses the implications of the results, highlighting strengths, limitations, and future directions for integrating Indicators of Behavior (IOBs) into SQL injection threat hunting.

Our ontology effectively models adversarial intent by representing the syntax of SQL queries—capturing the meaning behind keywords, operators, and syntax. Transforming security events into a knowledge graph allows both contextualisation and identification of user behavior. The resulting provenance graph of temporally related events supports semi-automated, entity and behaviour based threat hunting, enabling deeper investigation with explainable event chains.

Reasoning on SQL syntax and contextual domain information helps infer behavioral intent, prioritise relevant detections, and pivot on associated events. By abstracting adversarial goals into a hierarchy of behaviors through rule-based reasoning, we enable querying both low-level event data and higher-level behavior classifications. These reusable analytics improve explainability and enrich

contextual understanding beyond traditional signature- or machine learning-based methods.

Despite the limited context of anonymised IP addresses and SQL commands, our approach performs effective inductive and deductive reasoning. Integrating organisational asset context enables focused threat hunts, linking events with behaviors to facilitate a more precise and semi-automated process. Compared to supervised and unsupervised learning approaches, our ontology-based method offers enhanced situational awareness through explainable correlations of event chains.

Many isolated low-level behaviors have limited impact, but combined, they yield meaningful detections. Thus, rich contextual modelling is critical for effective behavioral detection. While this study focuses on SQL injection, the methodology is extensible to other attack vectors such as cross-site scripting and command injection.

7.1 Limitations

Because the primary intent of our vulnerable SQL scenarios is to educate offensive security techniques in a classroom setting, we lack more sophisticated threat actor payloads. Our captured attack activities occur over a very short period between the adversary's initial activities and the eventual compromise. However, these temporal behavioral event chains match a smash-and-grab style activity where the adversary has less sophisticated methods unlike advanced persistent threats [37]. However, whilst the adversary's' skills are limited, the same behavioral concepts can be applied since we focus on what the adversary is trying to achieve based on the syntax the SQL statement. A syntax that the adversary is bound by.

Another limitation is a lack of a taxonomy for IOBs. In this research we provide a proof of concept set of behaviours that can be applied to an SQL domain. A taxonomy of IOBs will provide a reusable set of analytics without the constant need for in-house IOB development.

Conclusions

In conclusion, the research demonstrates the efficacy of ontologies and semantic reasoning in abstracting indicators of behavior, with a specialised focus on detecting SQL injection attacks. By transforming SQLi events into semantic data, the research provides a context-rich detection and reasoning paradigm that goes beyond conventional pattern matching and classification.

The creation of reasoned IOBs and the use of ontological relationships contextualize and clarify the interpretative process, allowing for better understanding of events holistically. This work demonstrates that incorporating semantic technologies into threat hunting approaches significantly enhances their contextual awareness and analytical depth of a threat hunt. In this paper, the methods used to contextualise

both security events and organisational assets offers a sophisticated and context rich threat hunting investigation.

References

[1] Adaptive Office Solutions., "The Evolution of Cyber Attacks: A Decade of Change in the US and Canada", URL: https://www.adaptiveoffice.ca/blog/the-evolution-of-cyber-attacks-a-decade-of-change-in-the-us-and-canada/, Aug 2024

[2] Weids, D., "The Evolving Threat Landscape: Understanding Cyber Threats in the Digital Age.", Boardroom Cybersecurity, APress, 2024, URL: https://doi.org/10.1007/979-8-8688-0785-5_1

[3] Bianco, D., "The pyramid of pain", March 2014 URL: https://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html

[4] Team Cyware., "The Evolution of Cyber Threat Intelligence: From Threat Detection with IOCs to Defense Intelligence with IOBs", CYWARE Blog, Oct 2025, URL: https://www.cyware.com/blog/the-evolution-of-cyber-threat-intelligence-from-threat-detection-with-iocs?utm_content=351190447&utm_medium=social&utm_source=linkedin&hss_channel=lcp-10407612

[5] Any Run., "How Indicators of Compromise, Attack, and Behavior Help Spot and Stop Cyber Threats", Any Run Blog, 16 Aug 2025, URL: https://any.run/cybersecurity-blog/iocs-iobs-ioas-explained/

[6] Rastogi N *et al*., "Survey Perspective: The Role of Explainable AI in Threat Intelligence", arxiv, 3 Mar 2025, URL: https://www.arxiv.org/abs/2503.02065v1

[7] Chetwyn, R. A., Eian, M. and Jøsang, A., Modelling indicators of behaviour for cyber threat hunting via sysmon. In Proceedings of the 2024 European Interdisciplinary, Cybersecurity Conference, EICC '24, page 95–104, New York,NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3655693.3655722

[8] Chetwyn, R. A., Eian, M. and Jøsang, A., Ontohunt - 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, 2024. doi:10.1109/CSR61664.2024.10679394

[9] SQLLite. (n.d.). *Select*. SQLLite Overview. https://www.sqlite.org/lang_select.html

[10] kingthorin, zbraiterman, OWASP., "SQL Injection", OWASP Foundation, (n.d), URL: https://owasp.org/www-community/attacks/SQL_Injection

[11] Bianco, D., "Introducing the PEAK Threat Hunting Framework", Splunk Blogs, 18 April 2023, URL:

https://www.splunk.com/en_us/blog/security/peak-threat-hunting-framework.html

[12] W3C., "RDF Scheme 1.1", W3C Recommendation, 25 Feb 2014, URL: https://www.w3.org/TR/rdf-schema/

[13] W3C., "OWL 2 Web Ontology Language Document Overview (Second Edition)", W3C Recommendation, 11 Dec 2012, URL: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/

[14] Naveen Durai, K., Subha, R. and Haldorai, A., A novel method to detect and prevent sqlia using ontology to cloud web security. Wireless Personal Communications, 117(4):2995-3014, Mar 2020. doi:10.1007/s11277-020-07243-z

[15] Dora, J. R., Hluch´y, L. and Nemoga, K., Ontology for blind sql injection. Computing and Informatics, 42(2):480-500, 2023. doi:10.31577/cai_2023_2_480

[16] MITRE., MITRE ATT&CK Framework, (n.d.), URL: https://attack.mitre.org/

[17] MITRE, CAPEC, 6 July 2023, URL: https://capec.mitre.org/

[18] MITRE, Common Weakness Enumeration, 22 Mar 2024, URL: https://cwe.mitre.org/about/faq.html

[19] MITRE. Exploit public-facing application, Accessed : Nov 2023. URL: https://attack.mitre.org/techniques/T1190/

[20] CAPEC. Capec-66, Sql injection, Accessed: Jul 2018. URL: https://capec.mitre.org/data/definitions/66.html

[21] CAPEC. Capec-108, Command line execution through sql injection, Accessed: Jul 2018. URL: https://capec.mitre.org/data/definitions/108.html

[22] CWE, Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection'), Accessed: Feb 2024. URL: https://cwe.mitre.org/data/definitions/89.html

[23] OWASP and kingthorin, Sql injection. URL: https://owasp.org/www-community/attacks/SQLInjection

[24] MITRE., Common Vulnerabilities and Exposures (CVE), (n.d), URL: https://www.cve.org/

[25] Demilie, W. B., and Deriba, F. G., Detection and prevention of sqli attacks and developing compressive framework using machine learning and hybrid techniques. Journal of Big Data, 9(1), Dec 2022. doi:10.1186/s40537-022-00678-0

[26] Ross, K.. SQL injection detection using machine learning techniques and multiple data sources. doi:10.31979/etd.zknb-4z36

[27] Gupta, A., Tyagi, L. K. and Mohamed, A., A machine learning methodology for detecting sql injection attacks.In 2023 3rd International Conference on Technological Advancements in Computational Sciences (ICTACS), pp. 184-191, 2023. doi:10.1109/ICTACS59847.2023.10390153

[28] Alghawazi, M., Alghazzawi D. and Alarifi, S.. Deep learning architecture for detecting sql injection attacks based on rnn autoencoder model. Mathematics, 11(15), 2023, URL: https://www.mdpi.com/2227-7390/11/15/3286, doi:10.3390/math11153286

[29] Misquitta J. and Asha S., Sql injection detection using machine learning and convolutional neural networks. In 2023 5th International Conference on Smart Systems and Inventive, Technology (ICSSIT), pp. 1262-1266, 2023, doi:10.1109/ICSSIT55814.2023.10061019

[30] Apache HTTP Server Project, Apache HTTP Server Version 2.4 Log Files, Accessed: Jan 2025, URL: https://httpd.apache.org/docs/current/logs.html

[31] Stardog, Stardog latest documentation, Accessed: Jan 2025, URL: https://docs.stardog.com/

[32] Protégé, Protégé platform, Accessed: Jan 2025, URL: https://protege.stanford.edu/

[33] Munir, K., Sheraz Anjum, M., The use of ontologies for effective knowledge modelling and information retrieval. Applied Computing and Informatics, 14(2):116-126, 2018., doi:10.1016/j.aci.2017.07.003

[34] Sajid576., SQL Injection Dataset: kaggle.com. https://www.kaggle.com/datasets/sajid576/sql-injection-dataset, 2022

[35] HuntIO, Types of threat hunting, Accessed: Sep 2024. URL: https://hunt.io/glossary/types-of-threat-hunting

[36] ISMS.Online. Iso 27001:2022 annex a 5.9 – inventory of information and other associated assets. URL: https://www.isms.online/iso-27001/annex-a/5-9-inventory-of-information-other-associated-assets-2022/#:~:text=ISO%2027001%3A2022%20Annex%20A%20Control%205.9%20is%20named%20Inventory,take%20steps%20to%20protect%20them

[37] Salim, D. T., Singh, M, M. and Keikhosrokiani, P.. A systematic literature review for apt detection and effective cyber situational awareness (ecsa) conceptual model. Heliyon, 9(7):e17156, 2023, URL: https://www.sciencedirect.com/science/article/pii/S2405844023043645,doi: 10.1016/j.heliyon.2023.e1716

[38] Irunga *et al.,* "Artificial Intelligence Techniques for SQL Injection Attack Detection" ICIIT '23: Proceedings of the 2023 8th International Conference on Intelligent Information Technology, 13 July 2023, DOI: https://doi.org/10.1145/3591569.3591576

[39] Abdullah. H, Abdulazeez Adnan., "Detection of SQL Injection Attacks Based on Supervised Machine Learning Algorithms: A Review", Vol. 5, No. 2 (2024): INJIISCOM: Vol. 5, 24 Apr 2024, DOI: https://doi.org/10.34010/injiiscom.v5i2.12731

[40] Mehdta. D *et al*., "SQLIML: A Comprehensive Analysis for SQL Injection Detection Using Multiple Supervised and Unsupervised Learning Schemes", Vol. 4 SN Computer Science A. 281, 24 Mar 2023, DOI: https://link.springer.com/article/10.1007/s42979-022-01626-8

[41] Zhang. Y *et al.*, "A Method of SQL Injection Attack Detection Based on Large Language Models," 2024 2nd International Conference on Computer Network Technology and Electronic and Information Engineering (CNTEIE), Changchun, China, 2024, pp. 154-158, doi: 10.1109/CNTEIE66268.2024.00035