ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240626

Detecting Patterns of Attacks to Network Security in Urban Air Mobility with Answer Set Programming

Gioacchino Sterlicchio[a,*,1] **and Francesca Alessandra Lisi**[b,1]

[a]DMMM, Polytechnic University of Bari, Italy
[b]DIB and CILA, University of Bari Aldo Moro, Italy
ORCID (Gioacchino Sterlicchio): https://orcid.org/0000-0002-2936-0777, ORCID (Francesca Alessandra Lisi): https://orcid.org/0000-0001-5414-5844

Abstract. The growth of unmanned aerial vehicles (UAVs) will make the sky more crowded and pose several challenges as regards safety and security. Enabling high-rate, low-latency and ultra-reliable wireless communication between UAVs and ground base is crucial to realize their large-scale usage in the future, especially in the field of Urban Air Mobility. Recently, cellular-connected UAVs have drawn significant attention as a promising technology for Automatic Dependence Surveillance Broadcast (ADS-B) Like communication, which leverages other types of communication such as 4G LTE. In this work, we address the current lack of ADS-B security features and propose to use Answer Set Programming (ASP) for finding contrast sequential patterns that characterize different attacks on the 4G LTE network. The experiments show that a declarative approach is feasible in this context, and that the implementation of span and gap constraints make the search for patterns more efficient and effective.

1 Introduction

In the past few years there has been a tremendous increase in the use of unmanned aerial vehicles (UAVs) in civilian applications, such as aerial surveillance, traffic control, photography and communication relaying. A new aviation market is developing as Urban Air Mobility (UAM), a new air transportation system for passengers and cargo in urban environments thanks to new advanced technologies [24]. The transportation task is performed by electric aircraft that take off and land vertically, remotely piloted, with a pilot onboard or autonomous (in the future). The aircraft used range from small drones for parcel deliveries to air taxis for passenger transport. Among the several features like safety, sustainability, privacy and affordability, we have focused on *security*. UAM has a complex ecosystem made by different interconnected technologies which makes it vulnerable to cyber-threats disrupting UAM operations or corrupt the data exchanged between stakeholders and supporting infrastructure systems, likely resulting in potentially high-impact risks for the broader aviation transportation system [14].

Cellular-connected UAV [33] is a promising technology to achieve the essential UAM requirements of high-capacity, low-latency and ultra-reliable wireless communications between UAVs and their associated ground entities, not only for supporting their safe operation, but also for enabling mission-specific rate-demanding payload communications. This kind of technology can be used together by unmanned aerial system traffic management (UTM) and ADS-B Like communication ensuring safety and security [28] to organize the operation of the UAV. Automatic Dependent Surveillance Broadcast (ADS-B) is a surveillance technology in aviation that automatically and periodically broadcasts flight information (altitude, heading, and position based on Global Navigation Satellite System). An evolution is ADS-B Like communication that is similar to ADS-B but uses other technologies such as 4G long-term evolution (LTE) networks and broadcast technologies such as Wi-Fi beacons, APRS, XBee, and LoRaWAN (long-range wide area network) [28].

The threats could lead to a loss of confidentiality, integrity, or availability of the data exchanged or stored across the UAM ecosystem [12]. Threats that could affect data confidentiality would allow access and disclosure of restricted information, including proprietary and personal privacy information. The loss of data confidentiality could result from providing unauthorized information access to an attacker. Threats that could affect the integrity of the data in the UAM systems would risk the modification or destruction of the data. Message tampering, service supplier spoofing, data injection, and data manipulation could lead to the loss of data authenticity and validity, resulting in a loss of data integrity. The availability of data in UAM systems could be affected by threats that target the reliable or timely access to information, such as Global Positioning System (GPS) jamming or denial of service attacks. Furthermore, the threats of GPS spoofing or man-in-the-middle attacks have the additional risk of potentially leading to the disruption of UAM operations and the loss of UAM vehicles in flight.

In this paper we consider the problem of detecting patterns of attacks to 4G LTE network security in UAM. Traces are sequential data that need to be analyzed to discover regularities that characterize attacks in contrast with the normal network activities. To this aim, we leverage the *Contrast Sequential Pattern Mining (CSPM)* task [4], and a declarative approach that exploits Answer Set programming (ASP) [19]. The work follows therefore the stream of research known as Declarative Pattern Mining (DPM). We consider a couple of attacks: the *authentication failure attack* on the attach procedure and the *numb attack* on the paging procedure [11]. However, the proposed methodology can be applied for other attacks to which the 4G network is vulnerable but also to 5G. Given that, this kind of analysis can be useful in UAM from at least two points of view: 1) better

* Corresponding Author. Email: g.sterlicchio@phd.poliba.it
1 Equal contribution.

understanding of the attack strategy suffered through a retrospective forensic analysis, and 2) use of detected patterns to monitor the behavior of the 4G network and identify an attack in progress online.

The paper is organized as follows. In Section 2 we overview the current research in network security and in DPM. In Section 3 we provide the necessary background on ASP and CSPM. In 4 we describe our ASP enconding for the data and the problem, and in Section 5 we report the experimental results obtained on the chosen datasets. Section 6 concludes the paper with final remarks.

2 Related works

Works related to this paper come from two research areas: Cybersecurity, and (Declarative) Pattern Mining.

First, we cite existing efforts that focus on security, privacy and availability on networks, with particular reference to formal methods for the security of 4G LTE. In [11], Hussain *et al.* investigate the security and privacy of three critical procedures of the 4G LTE protocol and propose LTEInspector which combine a model checker and a cryptographic protocol verifier. In [31] they focus on identifying non-trivial interactions, using an explicit-state model checker, between the different control-plane protocol layers of LTE. Broek *et al.* [32] and Khan *et al.* [15] both proposed changing pseudonym-based (PMSI) defense against IMSI catching attack.

Network security is also field of application for pattern mining algorithms. In [3], Buczak *et al.* survey methods of machine learning and data mining that can be successfully applied to intrusion detection. Sequence mining is one of them. Li *et al.* [18] use sequential pattern mining in real time to find out the frequency and sequence features of multi-stage attacks. In [16], the authors construct attack graph from transaction database using sequential pattern mining. Husák *et al.* [10] use sequential pattern mining and rule mining in the analysis of cyber security alert sharing SABU. Whereas sequence mining is widely explored in network security, the CSPM task has not been addressed so far in this application domain to the best of our knowledge.

In recent years there has been an increasing interest in DPM, a research stream in which the objective is to develop declarative approaches to pattern mining. Several encodings have been presented so far, to cover pattern mining tasks such as sequence mining [23, 6] and frequent itemset mining [13, 7]. ASP is widely used in DPM. The first proposal is described by Guyet *et al.* [8]. The authors explore the SPM problem with ASP and compare their method with a dedicated algorithm. Gebser *et al.* [6] use ASP for extracting condensed representations of sequential patterns. Samet *et al.* in [29] mine rare sequential patterns with ASP. Guyet *et al.* [9] present the use of ASP to mine sequential patterns within two representations of embeddings (fill-gaps vs skip-gaps) and compare them with Constraint Programming. A hybrid ASP approach is proposed by Paramonov *et al.* [25] which combines dedicated algorithms for pattern mining and ASP. In [20] *Lisi and Sterlicchio* propose an ASP-based approach to Contrast Pattern Mining. In [21], the same authors present the first ASP encoding for the CSPM problem. We will refer to this declarative approach to CSPM as *MASS-CSP* (*Mining with Answer Set Solving - Contrast Sequential Patterns*) hereafter. *MASS-CSP* is our starting point for the ASP encoding presented in Section 4.

3 Background

3.1 Contrast Sequential Pattern Mining

Contrast Sequential Pattern Mining (CSPM) is a data mining technique that aims to discover patterns in data which contrast with a set of negative examples [4]. Unlike traditional sequential pattern mining [22], which focuses on finding patterns that occur frequently in a dataset, CSPM looks for patterns that are significantly different between a set of positive and negative examples. CSPM is particularly useful for analyzing time-ordered sequences of transactions, events or actions to uncover hidden associations or trends that may not be apparent when analyzing each group separately. This can lead to new discoveries and insights into complex datasets where traditional pattern mining techniques may fall short. More formally, given two sequences dataset, D_1 labeled with the C_1 class and D_2 labeled as C_2. Each sequence s is represented as list of ordered items $s = \langle i_1, i_2, \ldots, i_n\rangle$ from an alphabet Σ and an attribute from a set A that represents the class label (e.g., positive/negative). First, we look sequential patterns p, defined as an ordered list of items $p = \langle a_1, a_2, \ldots, a_k\rangle$ such that each $a_i \in \Sigma$ and occurs consecutively in at least one sequence in S. The support of a sequential pattern p, $supp(p)$, is the number of sequences in which it occurs. Given a minimum support threshold *minsup*, we find all frequent sequential patterns p_i, such that $supp(p) \geq minsup$. Frequent sequential patterns are those that occur frequently enough within the dataset based on the specified support threshold. Then, given the *growth rate* from D_2 to D_1 of a sequential pattern p as $GR_{C_1}(p) = \frac{supp(p,\, D_1)/|D_1|}{supp(p,\, D_2)/|D_2|}$. If $supp(p, D_2) = 0$ and $supp(p, D_1) \neq 0$ then $GR_{C_1}(p) = \infty$. After, the growth rate from D_1 to D_2 of p is defined as $GR_{C_2}(p) = \frac{supp(p,\, D_2)/|D_2|}{supp(p,\, D_1)/|D_1|}$. If $supp(p, D_1) = 0$ and $supp(p, D_2) \neq 0$, then $GR_{C_2}(p) = \infty$. The contrast rate of p is defined as $CR(p) = max\{GR_{C_1}, GR_{C_2}\}$ and if $GR_{C_1}(p) = 0$ and $GR_{C_2}(p) = 0$ then $CR(p) = \infty$. We say that p is a contrast sequential pattern if $CR(p) \geq mincr$, where *mincr* is the minimum contrast rate threshold.

Table 1 shows a sequences dataset D of bank customers which is obtained by merging the datasets D_1 and D_2 that contain *compliant* and *non-compliant* customers, respectively. We start by finding sequential patterns first and given $minsup = 2$, $\langle deposit, withdrawal, transfer\rangle$ is a sequential pattern because it occurs in sequences 1, 2, and 3. Another example is $\langle deposit, withdrawal\rangle$ within all. Assuming we have found all the sequential patterns, we check whether these are contrasting for one of the two classes. Given $mincr = 2$, $p_1 = \langle deposit, withdrawal, transfer\rangle$ and the metrics $supp(p_1, D_1) = 2$, $supp(p_1, D_2) = 1$, $GR_{compliant}(p_1) = 2$, $GR_{non\text{-}compliant}(p_1) = 0.5$, and $CR(p_1) = 2$, p_1 is a contrast sequential pattern for *compliant* because $CR(p_1) \geq mincr$. Given $p_2 = \langle deposit, withdrawal\rangle$ and its metrics $supp(p_2\, D_1) = 2$, $supp(p_2, D_2) = 2$, $GR_{compliant} = 1$, $GR_{non\text{-}compliant} = 1$, p_2 is not a contrast sequential pattern for any of the two classes. In this case the pattern is present equally in the two classes. This toy example suggests how contrast patterns can help in identifying key differences in transaction sequences between compliant and non-compliant customers which can be used for fraud detection or for improving customer compliance measures within the bank domain.

3.2 Answer Set programming

Answer Set Programming (ASP) is a declarative programming paradigm [19, 2] that is based on logic programming and non-

Table 1. An example of sequence dataset concerning compliant/non-compliant bank customers.

ID	Sequence	Class
1	$\langle deposit, withdrawal, transfer\ \rangle$	*compliant*
2	$\langle deposit, withdrawal, transfer\rangle$	*compliant*
3	$\langle deposit, withdrawal, transfer\rangle$	*non-compliant*
4	$\langle deposit, withdrawal, deposit\rangle$	*non-compliant*

monotonic reasoning, which allows for the representation of incomplete or uncertain information.

Formally, an ASP *program* consist of a set of rules, where each *rule* has the following form: $a_1 \vee \ldots \vee a_n \leftarrow b_1, \ldots, b_k, not\ b_{k+1}, \ldots, not\ b_m$. It says that if $b_1, \ldots, b_k$ are true and there is not reason for believing that $b_{k+1}, \ldots, b_m$ are true then at least one of the $a_1, \ldots, a_n$ is believed to be true. The left hand side and the right hand side of the $\leftarrow$ are called *head* and *body* respectively. Rules without body are called *facts*. The head is unconditionally true and the arrow is usually omitted. Conversely, rules without head are called *denials* (or integrity constraints).

The semantics of an ASP program is defined in terms of *answer sets*, which are sets of atoms that represent solutions to the program, i.e. that satisfy all the rules in the program. Formally, an answer set for an ASP program P is a set of atoms A such that: 1) A satisfies every rule in P, i.e., for every rule r in P, there is a literal L in r such that L is true in A, and 2) A is minimal, i.e., no proper subset of A satisfies every rule in P. Denials are used to discard stable models, thus reducing the number of answers returned by the ASP solver.

ASP solvers (e.g. Clingo [5] and DLV [17]) use efficient algorithms based on non-monotonic reasoning and logic programming techniques to compute answer sets for given programs. These solvers typically employ *grounding* techniques to convert first-order logic into propositional form so that ASP solvers can be used for solving them as efficiently as possible with the current knowledge.

4 A Declarative Approach

In this section, we describe how to encode traces in ASP and how to mine them for finding contrast sequential patterns.

4.1 ASP encoding of traces

Table 2. Propositions for the numb and the authentication failure attacks.

Idx	Numb attack	Authentication failure attack
1	tracking_area_update_request	attach_complete
2	attach_accept	security_mode_complete
3	service_request	identity_request
4	attach_request	tracking_area_update_complete
5	detach_accept	identity_response
6	attach_complete	emm_information
7	authentication_request	authentication_failure
8	security_mode_command	tracking_area_update_request
9	identity_response	attach_request
10	emm_information	detach_accept
11	tracking_area_update_accept	security_mode_command
12	detach_request	tracking_area_update_accept
13	identity_request	authentication_request
14	extended_service_request	detach_request
15	authentication_response	attach_accept
16	authentication_reject	authentication_response
17	security_mode_complete	service_request
18	tracking_area_update_complete	

We consider several traces of a 4G-LTE formal modelling, each of them is labelled as normal or attack. Each row is a state where each value represents the truth value of a proposition (see Table 2 to understand order and name), where 1 means that the proposition is true in that state, 0 otherwise. Each state is separated by the semicolon symbol and only one proposition in true in each state. As regards the authentication failure attack, a frame of normal traces is given in Listing 1 where the sequence of normal events is $\langle \ldots,$ *security_mode_complete, attach_accept, attach_complete, detach_request* $, \ldots\rangle$. Conversely, Listing 2 shows the sequence $\langle \ldots,$ *authentication_failure, authentication_request, authentication_response, security_mode_command* $, \ldots\rangle$ that highlights the occurrence of an authentication failure attack.

Listing 1. An example of normal trace from authentication failure.
```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
-------------------------------------------
...
0,1,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0;
0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 1, 0, 0;
1,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0;
0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 1, 0, 0, 0;
...
```

Listing 2. An example of attack trace from authentication failure.
```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
-------------------------------------------
...
0,0,0,0,0,0,1,0,0, 0, 0, 0, 0, 0, 0, 0, 0;
0,0,0,0,0,0,0,0,0, 0, 0, 0, 1, 0, 0, 0, 0;
0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 1, 0;
0,0,0,0,0,0,0,0,0, 0, 1, 0, 0, 0, 0, 0, 0;
...
```

We assume traces to be uniquely indexed by integers, and in particular we will assume that a trace π^i is referred to by the integer i. We encode a trace π^i as a sequence by means of a set of facts matching the predicate $seq/3$. The atom $seq(t, p, i)$ models that $i \in \pi^i_p$. The only additional information to encode is whether each sequence is labeled as either normal or attack and this is done through the atom $cl(t, c)$ where $c \in \{normal, attack\}$. We denote by $P(\pi)$ the set of facts that encode the trace π.

Listing 3. ASP encoding of the normal trace shown in Listing 1.
```
cl(0,normal).
...
seq(0,4,security_mode_complete).
seq(0,5,attach_accept).
seq(0,6,attach_complete).
seq(0,7,detach_request).
...
```

Listing 4. ASP encoding of the attack trace shown in Listing 2.
```
cl(20,attack).
...
seq(20,4,authentication_failure).
seq(20,5,authentication_request).
seq(20,6,authentication_response).
seq(20,7,security_mode_command).
...
```

Examples of ASP encoding of traces are given by Listings 3 and 4. The first line of each says that the sequence has $normal/attack$ class, whereas the other lines describe which proposition is true in each timestamp of that sequence. In other words they describe the evolution of the system (normal or bad) over time.

4.2 ASP encoding of CSPM with constraints

As already mentioned in Section 3, we have considered the ASP encoding reported in [21] as a starting point for the work presented in this paper. The problem variant of CSPM addressed in [21] is a basic one. We start with a couple of observations on the limits of that variant. For illustrative purposes, let us consider the pattern $\langle a, b\rangle$ and the sequences $\langle a, b, c\rangle$ and $\langle a, c, c, b\rangle$. First, they do not deal with the number of gaps between one embedding and another. In other words, two consecutive items of a sequential pattern can be n gaps apart within a sequence, in the example 0 and 2 respectively. Secondly, $\langle a, b\rangle$ has support in both sequences but with different span, namely 1 and 3 respectively. Our work develops on the basis of these two observations because in various application domains, patterns that reflect certain characteristics are more informative.

In [27] there were defined many types of constraints on patterns and embeddings, among which the ones based on the notion of *gap* and *span* (illustrated in Figure 1). These constraints are useful in several ways: 1) by applying the span and gap constraints, we can reduce the number of candidate patterns that need to be generated and checked, which can significantly improve the efficiency of the mining process; 2) the span and gap constraints can help filter out patterns that do not make sense in the context of the data. For example, if we know that certain events should happen closely together in time, we can set a small span constraint to filter out patterns that have a large gap between them; 3) by applying the span and gap constraints, we can identify patterns that are meaningful and interesting, rather than just finding random combinations of items; 4) by limiting the number of items between two items in a pattern, we can improve the interpretability of the pattern and make it easier to understand the relationships between the items; 5) by limiting the number of items between two items in a pattern, we can reduce the noise in the data and focus on the most important items.

In the next subsections we briefly describe how we have implemented these constraints in *MASS-CSP*, more precisely in the ASP encoding for sequence mining originally described in [9] and exploited subsequently in [21] for CSPM. The full encoding is reported in the supplementary material [30]. The goal is to improve the efficiency of the overall ASP encoding of the CSPM problem. As discussed in [9], the gap and span constraints can be encoded in two ways in ASP. 1) We can use *denials* to delete answer sets that do not satisfy the constraints. The problem is that they act a posteriori during the test stage for validating candidate models. In this way we loose the benefits of applying the constraints. 2) A more effective method consists in introducing constraints in the generate stage for pruning the search space earlier. This is possible if we implement the gap and span constraints as *choice rules*. We have chosen to follow this second implementation strategy for the fill-gaps technique, that is the representation of embeddings adopted in *MASS-CSP* [21]. Note that, in [9] the implementation of the constraints as choice rules was presented only for the skip-gap technique. So our implementation complements previous work.

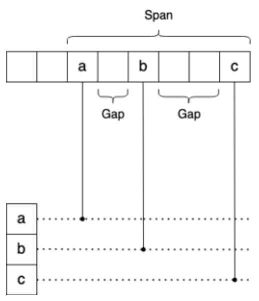

Figure 1. Illustration of the gap and span notions.

Span constraint

The span constraint specifies the minimum/maximum length allowed for a sequential pattern. As illustrated in Figure 1, it is the difference between its last item timestamp that is 8 and its first item timestamp, i.e. 3, and thus $\langle a, b, c\rangle$ has span 5 in that sequence. A span constraint requires that the pattern duration should be longer or shorter than a given time period. By setting a span constraint, we can focus on identifying shorter or longer sequences of events based on our specific requirements. For instance, if we set a short span constraint, we may discover frequent itemsets that occur closely together in a short period, while a larger span allows us to capture more spread-out occurrences. The maximal span constraint is anti-monotonic while the minimal span constraints is monotonic [26].

In our encoding, the predicate *seq(T, P, I)* defines the timestamp of *I* in sequence *T* as the integer position *P*. The idea is to add an argument to *occS/3* to denote the position of the occurrence of the first pattern item (Listing 5). Line 1 finds the first occurrence of pattern item within a sequence, Line 2 maintains the knowledge that the P-th pattern item has been mapped all along the further sequence indexes. Lines 3-4 make a comparison among the *minspan* and *maxspan* parameters (*#const* statement) and the difference between the first item position *IP* and the last item position *P*.

Listing 5. ASP encoding of the span constraint in the fill-gaps approach.

```
1  occS(T,1,P,P) :- seq(T,P,I), pat(1,I).
2  occS(T,L,P,IP) :- occS(T,L,P-1,IP), seq(T,P,_).
3  occS(T,L,P,IP) :- occS(T,L-1,P-1,IP), seq(T,P,C),
4        pat(L,C), P-IP+1>=minspan, P-IP+1<=maxspan.
```

Gap constraint

The gap constraint controls the minimum/maximum gap allowed between consecutive occurrences of items within a sequence. It specifies how many time units may intervene before an item is observed again. In Figure 1, the gap between a and b is 1 while 2 between b and c. Gap constraints are essential for capturing temporal relationships between events. Setting appropriate gap values helps identify patterns where there might be delays or interruptions between related events but still maintain their significance. The minimal and maximal gap constraints are anti-monotonic [26]. A gap constraint imposes a constraint on all embeddings, if an embedding does not satisfy the constraint, the whole pattern is unsatisfied.

Listing 6 shows the ASP encoding of the gap constraint with the *mingap* and *maxgap* parameters provided by the user using the *#const* statement. Only embeddings that satisfy *mingap* and *maxgap* will be considered pattern occurrences within a sequence (Lines 3-6). Lines 1-2 are similar to the corresponding ones in Listing 5.

Listing 6. ASP encoding of the gap constraint in the fill-gaps approach.

```
1  occG(T,1,P) :- seq(T,P,I), pat(1,I).
2  occG(T,L,P) :- occG(T,L,P-1), seq(T,P,_).
3  occG(T,L,P) :- occG(T,L-1,P-1), seq(T,P,C), pat(L,C),
4                 pat(L-1,C1), seq(T,P2,C1),
5                 P-P2-1>=mingap, P-P2-1<=maxgap.
```

Span and gap constraints can be jointly applied by merging the two encodings above as shown in Listing 7.

Listing 7. ASP encoding of the span+gap constraint in the fill-gaps approach.

```
occSG(T,1,P,P) :- seq(T,P,I), pat(1,I).
occSG(T,L,P,IP) :- occSG(T,L,P-1,IP), seq(T,P,_).
occSG(T,L,P,IP) :- occSG(T,L-1,P-1,IP), seq(T,P,C),
                   pat(L,C), pat(L-1,C1), seq(T,P2,C1),
                   P-P2-1>=mingap, P-P2-1<=maxgap,
                   P-IP+1>=minspan, P-IP+1<=maxspan.
```

5 Experiments

CSPM can be particularly useful to 4G-LTE for different reasons, e.g., optimizing network performance by analyzing contrast patterns and making informed decisions on network configurations, resource allocations, and traffic management strategies and ensuring quality of services requirements. Since our focus is on security, anomalies or unusual behavior in the network traffic can be detected by mining contrast sequential patterns, thus helping in identifying potential security threats. Contrast sequential patterns are those that characterize normal and attack behaviour given different traces.

In this paper, we consider a couple of attacks - namely the *authentication failure attack* and the *numb attack* [11] - as a case study. Table 4 shows example patterns for both attacks. More precisely they are the longest contrast sequential patterns found having 30% support across all sequences. They describe the behavior or timeline of events that leads to the attack. Interestingly, both attacks share common behavior with the exception of the *authentication_failure* event that occurs always in subsequence $\langle \ldots,$ *authentication_request, authetication_failure, authentication_request* $, \ldots \rangle$ in (a) but not in (b).

In the rest of the section we report the experimental results obtained on two different sets of execution traces for these attacks. More details about the results can be found in the supplementary material. The main goal is to show the feasibility of a declarative approach to CSPM in the context of network security. Also, experiments have been designed in order to provide a comparative evaluation between the basic ASP encoding of *MASS-CSP* reported in [21] and the ASP encodings proposed here that implement the span/gap constraints. In fact, in [21] the performance of *MASS-CSP* was evaluated on a couple of datasets. In the present work we consider a real-world case study that exploits CSPM by applying an improved version of *MASS-CSP* in order to speed up its performance and reduce the explosion in the number of patterns. We empirically show what are the advantages of adding constraints on pattern embeddings.

5.1 Experimental setup

For the evaluation we have used the events logs made available by [1] for the *authentication failure* and *numb attack* on 4G-LTE cellular network, and partition each event log into positive and negative traces. A comprehensive description of each log can be found in [1] and in their github repository. [2] Table 3 summarizes the main features of the datasets used in the experiments. The number within each dataset name is the number of traces equally distributed in normal and attack traces. We report the dataset name, the number of distinct symbols ($|\Sigma|$), the number of sequences ($|\mathrm{D}|$), the total number of symbols in the dataset ($||\mathrm{D}||$), the maximum and the average sequence length ($|\mathrm{T}|$), and the density that is calculated by $\frac{||D||}{|\Sigma||D|}$.

As an ASP solver in the experiments, we have used clingo 5.4.0 with default solving parameters, only the timeout has been set to 1 hour. The ASP programs have been run on a laptop computer with AMD Ryzen 5 3500U @ 2.10 GHz, 8GB RAM without using the multi-threading mode of clingo. Multi-threading reduces the mean runtime but introduces variance due to the random allocation of tasks. However, such variance is inconvenient for interpreting results with repeated executions.

[2] https://github.com/CLC-UIowa/SySLite

Table 3. Trace datasets for authentication failure and numb attacks.

Dataset	$\|\Sigma\|$	**|D|**	**||D||**	**max|T|**	**avg|T|**	**density**
Auth_failure_40	17	40	748	35	17.70	1.10
Auth_failure_80	17	80	1,796	78	21.45	1.32
Auth_failure_200	17	200	4,188	76	19.94	1.23
Auth_failure_400	18	400	9,265	91	22.16	1.29
Auth_failure_1000	18	1,000	24,024	96	23.02	1.33
Numb_attack_40	18	40	522	57	12.05	0.73
Numb_attack_80	18	80	1,186	63	13.83	0.82
Numb_attack_200	19	200	3,129	51	14.65	0.82
Numb_attack_400	19	400	5,865	73	13.66	0.77
Numb_attack_1000	20	1,000	15,836	98	14.84	0.79

5.2 *Results with the basic* MASS-CSP

In the first bunch of experiments we have tested the ability of the basic algorithm to extract patterns that characterize attack traces. To this aim we have run it under different configurations to understand the amount of extracted patterns, the execution times and the memory needed. We started by extracting patterns of minimum length of 2 and maximum length of 2 (*minlen* and *maxlen* parameters), then 3 up to 6 by varying the minimum support threshold (*minsup*) and the minimum contrast rate threshold (*mincr*). This work has been done for all versions of each dataset. In [21], we showed how the encoding works with different input size in time and memory by increasing or decreasing *minsup* and/or *mincr*. In this paper, we go further and analyze how an increase in the length of the patterns affects the memory consumption and the time taken, considering the grounding and solving time. Before continuing with the discussion it is worthy to point out that the number of patterns both in this section and in the following one is on a logarithmic scale so as to facilitate the comparison. Figures 2 and 3 summarize what we said before, i.e., when we are looking for longer patterns (from 2 to 6), the program gets bigger and thus, more patterns are found, time is much higher and memory grows up. We have a huge number of patterns because the basic encoding does not take into account more effective and efficient constraints on embeddings like span and/or gap.

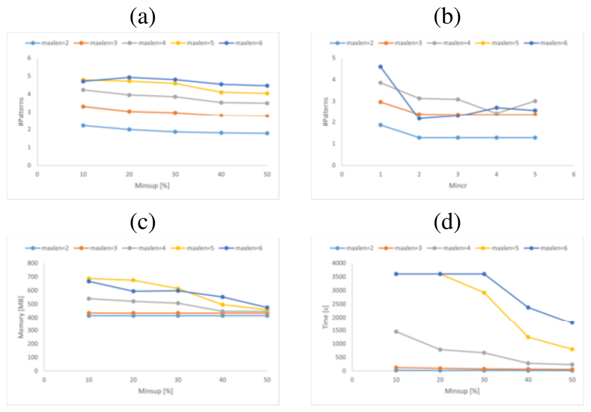

Figure 2. Number of patterns (a, b), memory consumption (c), and runtime (d) by varying the maximum pattern length *maxlen* on Auth_failure_1000.

5.3 *Results with the improved* MASS-CSP

In Section 4.2 we have described what type of constraints can be added to the basic version of *MASS-CSP* to enhance first of all effectiveness and efficiency but also improve searching for patterns by

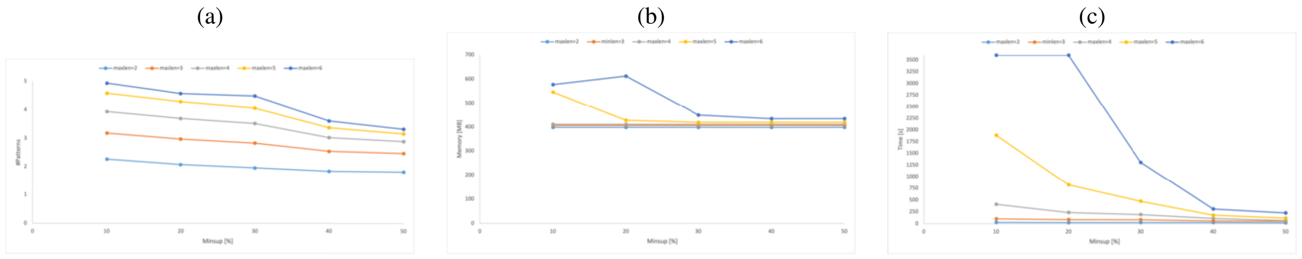

Figure 3. Number of patterns (a), memory consumption (b), and runtime (c) by varying the maximum pattern length *maxlen* on Numb_attack_1000.

filtering out useless ones according to the new constraints. For example, we would like patterns of behavior that have a certain temporal duration or that fall within a certain minimum and/or maximum temporal range. In the specific case study, contrast sequential patterns that describe sequential events during an attack may be useless if there are gaps between one item and another within the sequence. Interesting patterns are those that describe the evolution of the system whose items are sequential without gaps between them or understand the duration of bad sequential events that can led to an attack. This way we can accurately capture the crucial steps that may lead to detect an attack rather than stating that the evolution of the system is correct and is functioning normally.

To demonstrate the usefulness of the improvements described in Section 4.2, Figures 4 and 5 make a comparison between the basic *MASS-CSP* (dotted lines) and the improved one with the span/gap constraints (continuous lines). The advantage of adding constraints on the gap when calculating the embeddings of a pattern is clear. First, we have control over the type of pattern we want with the minimum and maximum gap. The pattern set is considerably reduced, extracting only those actually useful for our purpose with an advantage on time and memory as we act directly in the pattern generation phase, having a smaller ground program than the previous one. Analogously, using the span constraint, we are able to reduce the number of patterns and the execution times but it seems that we do not have improvement in memory consumption. This is because we are looking for patterns of a minimum and/or maximum duration and with a maximum length but we have no control over the gap between items within the sequence. Obviously the constraints on gap and span are successful when we want to find patterns because we reduce the search space and we are sure to eliminate superfluous ones. When we apply jointly the two constraints (Gap+Span), the number of patterns is lower than with the gap and span constrains taken separately but memory consumption and execution time are higher than with the gap constraint only. The gap constraint is the one that brings the best advantages in terms of overall performance.

6 Conclusions

This paper addresses the problem of detecting attack patterns to the network security in the context of UAM. As a case study we have considered only a couple of attacks to 4G LTE, namely the authentication failure and the numb attack. We have suggested that CSPM can be helpful and presented an ASP-based approach to mine contrast sequential patterns from 4G-LTE traces. The goal is to characterize normal and attack behavior on different type of attacks. The patterns found with our approach may be useful for post-attack analysis in order to understand the steps of the immediate attack and implement defensive mechanisms. Thanks to the generate-and-test approach of ASP, *MASS-CSP* does an exhaustive search of all the

(a)	⟨*attach_request, authentication_request, authentication_failure, authentication_request, authentication_response, security_mode_command, security_mode_complete, attach_accept, attach_complete, detach_request, detach_accept, attach_request, authentication_request, authentication_failure, authentication_request, authentication_response, security_mode_command, security_mode_complete, attach_accept, attach_complete, detach_request, detach_accept*⟩
(b)	⟨*attach_request, authentication_request, authentication_response, security_mode_command, security_mode_complete, attach_accept, attach_complete, detach_request, detach_accept, attach_request, authentication_request, authentication_response, security_mode_command, security_mode_complete, attach_accept, attach_complete*⟩

Table 4. Examples of longer attack patterns found with 30% support in (a) Auth_Failure_40 and (b) Numb_Attack_40.

possible patterns present in the data set to possibly attribute them to a specific class based on threshold values for contrast rate. So, depending on the values considered, a very large value may find the very likely (but not in the probabilistic sense) ones but may cause others to be missed. Conversely, a low value could lead to noise and therefore an increase in false positives. A lot is played on the threshold values of the measures considered (support and contrast rate).

Experiments have shown that applying a declarative approach is feasible. Also, results have highlighted the benefits of adding constraints to embeddings. In particular, the span and gap constraints allow pruning the set of patterns found, thus decreasing the memory consumption and the execution time. The mining process has been made more effective since constraints better limit the search for desired behaviors. The proposed implementation of the span and gap constraints for the fill-gaps representation of embeddings complements previous work in ASP-based sequence mining. Moreover, since the constraints are implemented in the sequential pattern search phase, the advantage is brought not only to CSPM tasks but also to any other pattern mining task that is based on sequence mining. Finally, since the only input is the set of execution traces, the approach applies also to other attacks on the same network or even on other networks such as 5G.

In this work, the negative behaviors, i.e. the attacks, have been identified because normal traces are also available with which to compare and find the differences. As future work, it would be interesting to explore the applicability of a declarative approach to discover anomalies or behaviors that occur with some regularity but not so frequently in traces where we do not know a corresponding class. From the ASP encoding point of view one could consider to use an incremental version of Clingo to expand the size of patterns in each iteration as well as a hybrid approach as shown in [21].

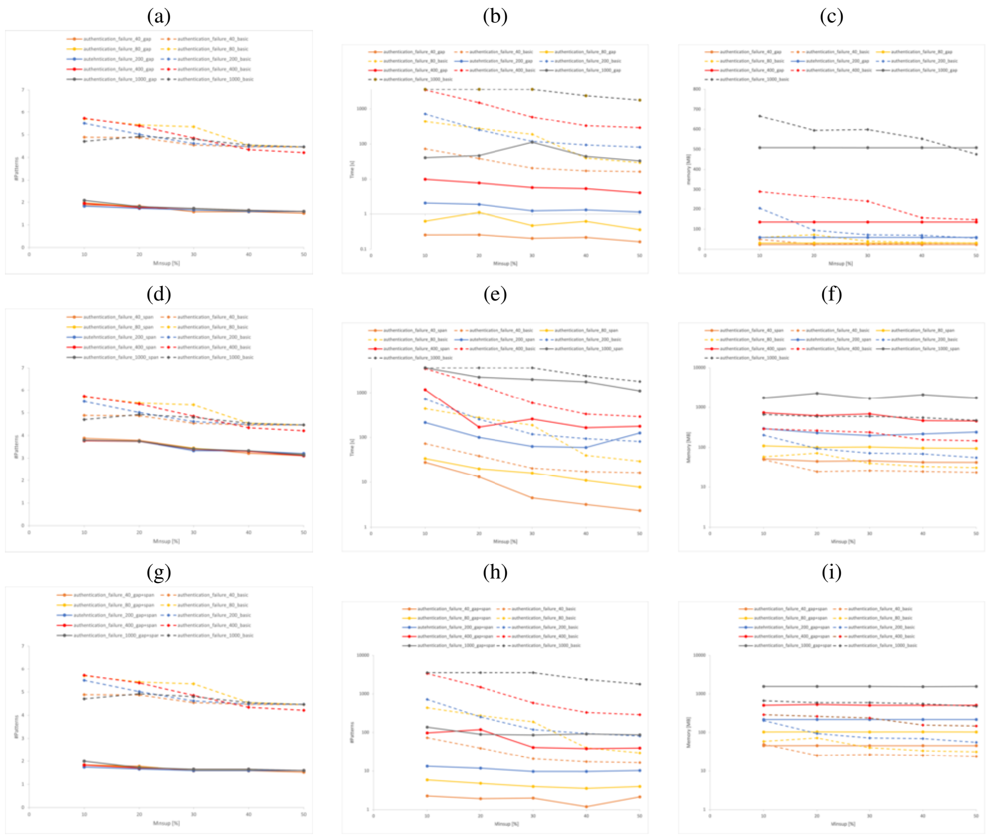

Figure 4. Comparison as regards number of patterns, execution time, and memory consumption between the basic ASP encoding and the encodings with gap (a-c), span (d-f), and gap+span (g-i) constraints on the datasets Auth_Failure with *mingap=0*, *maxgap=0*, *minspan=1*, *maxspan=10 minlen=2*, and *maxlen=6*.

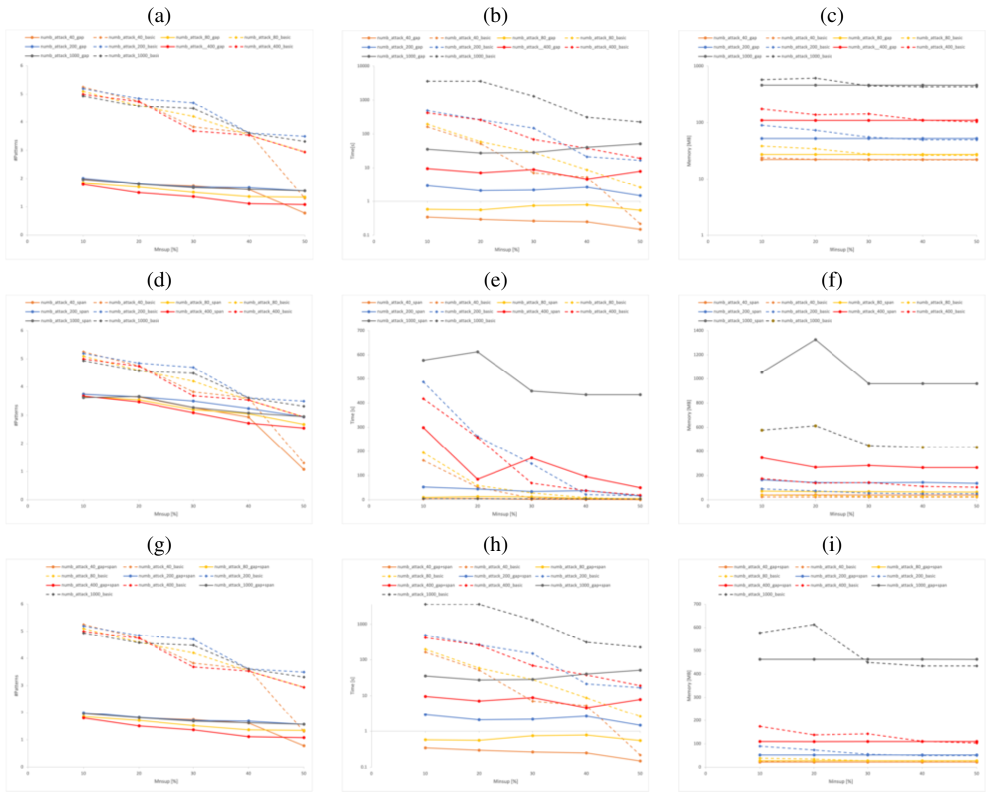

Figure 5. Comparison as regards number of patterns, execution time, and memory consumption between the basic ASP encoding and the encodings with gap (a-c), span (d-f), and gap+span (g-i) constraints on the datasets Numb_Attack with *mingap=0*, *maxgap=0*, *minspan=1*, *maxspan=10 minlen=2*, and *maxlen=6*.

Acknowledgements

This work was partially supported by the project FAIR - Future AI Research (PE00000013), under the NRRP MUR program funded by the NextGenerationEU.

References

[1] M. F. Arif, D. Larraz, M. Echeverria, A. Reynolds, O. Chowdhury, and C. Tinelli. Syslite: Syntax-guided synthesis of PLTL formulas from finite traces. In *2020 Formal Methods in Computer Aided Design (FMCAD)*, pages 93–103, 2020. doi: 10.34727/2020/isbn.978-3-85448-042-6_16.

[2] G. Brewka, T. Eiter, and M. Truszczynski. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011. doi: 10.1145/2043174.2043195. URL http://doi.acm.org/10.1145/2043174.2043195.

[3] A. L. Buczak and E. Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2):1153–1176, 2015.

[4] Y. Chen, W. Gan, Y. Wu, and P. S. Yu. Contrast pattern mining: A survey, 2022.

[5] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. Clingo= ASP+ control: Preliminary report. *arXiv preprint arXiv:1405.3694*, 2014.

[6] M. Gebser, T. Guyet, R. Quiniou, J. Romero, and T. Schaub. Knowledge-based sequence mining with ASP. In *IJCAI 2016-25th International joint conference on artificial intelligence*, page 8. AAAI, 2016.

[7] T. Guns, A. Dries, S. Nijssen, G. Tack, and L. De Raedt. Miningzinc: A declarative framework for constraint-based mining. *Artificial Intelligence*, 244:6–29, 2017.

[8] T. Guyet, Y. Moinard, and R. Quiniou. Using answer set programming for pattern mining. *arXiv preprint arXiv:1409.7777*, 2014.

[9] T. Guyet, Y. Moinard, R. Quiniou, and T. Schaub. Efficiency analysis of ASP encodings for sequential pattern mining tasks. In *Advances in Knowledge Discovery and Management*, pages 41–81. Springer, 2018.

[10] M. Husák, J. Kašpar, E. Bou-Harb, and P. Čeleda. On the sequential pattern and rule mining in the analysis of cyber security alerts. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, ARES '17, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450352574. doi: 10.1145/3098954.3098981. URL https://doi.org/10.1145/3098954.3098981.

[11] S. R. Hussain, O. Chowdhury, S. Mehnaz, and E. Bertino. LTEInspector: A systematic approach for adversarial testing of 4g LTE. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018. URL https://www.ndss-symposium.org/wp-content/uploads/2018/02/ndss2018_02A-3_Hussain_paper.pdf.

[12] C. Ippolito and K. Krishnakumar. An interface-based cybersecurity subsystem analysis on a small unmanned aerial systems. *AIAA SciTech*, 2019.

[13] S. Jabbour, L. Sais, and Y. Salhi. Decomposition based SAT encodings for itemset mining problems. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 662–674. Springer, 2015.

[14] A. Jordan, K. K. Jaskowska, A. Monsalve, R. Yang, M. Rozenblat, K. Freeman, and S. Garcia. Systematic evaluation of cybersecurity risks in the Urban Air Mobility operational environment. In *2022 Integrated Communication, Navigation and Surveillance Conference (ICNS)*, pages 1–15. IEEE, 2022.

[15] M. S. A. Khan and C. J. Mitchell. Trashing IMSI catchers in mobile networks. In *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 207–218, 2017.

[16] J. Lei and Z.-t. Li. Using network attack graph to predict the future attacks. In *2007 Second International Conference on Communications and Networking in China*, pages 403–407. IEEE, 2007.

[17] N. Leone, C. Allocca, M. Alviano, F. Calimeri, C. Civili, R. Costabile, A. Fiorentino, D. Fuscà, S. Germano, G. Laboccetta, B. Cuteri, M. Manna, S. Perri, K. Reale, F. Ricca, P. Veltri, and J. Zangari. Enhancing DLV for large-scale reasoning. In M. Balduccini, Y. Lierler, and S. Woltran, editors, *Logic Programming and Nonmonotonic Reasoning - 15th International Conference, LPNMR 2019, Philadelphia, PA, USA, June 3-7, 2019, Proceedings*, volume 11481 of *Lecture Notes in Computer Science*, pages 312–325. Springer, 2019. doi: 10.1007/978-3-030-20528-7_23. URL https://doi.org/10.1007/978-3-030-20528-7_23.

[18] Z. Li, A. Zhang, J. Lei, and L. Wang. Real-time correlation of network security alerts. In *IEEE International Conference on e-Business Engineering (ICEBE'07)*, pages 73–80. IEEE, 2007.

[19] V. Lifschitz. Answer sets and the language of answer set programming. *AI Magazine*, 37(3):7–12, 2016.

[20] F. A. Lisi and G. Sterlicchio. A declarative approach to contrast pattern mining. In A. Dovier, A. Montanari, and A. Orlandini, editors, *AIxIA 2022 - Advances in Artificial Intelligence - XXIst International Conference of the Italian Association for Artificial Intelligence, AIxIA 2022, Udine, Italy, November 28 - December 2, 2022, Proceedings*, volume 13796 of *Lecture Notes in Computer Science*, pages 17–30. Springer, 2022. doi: 10.1007/978-3-031-27181-6_2. URL https://doi.org/10.1007/978-3-031-27181-6_2.

[21] F. A. Lisi and G. Sterlicchio. Mining contrast sequential patterns with ASP. In R. Basili, D. Lembo, C. Limongelli, and A. Orlandini, editors, *AIxIA 2023 - Advances in Artificial Intelligence - XXIInd International Conference of the Italian Association for Artificial Intelligence, AIxIA 2023, Rome, Italy, November 6-9, 2023, Proceedings*, volume 14318 of *Lecture Notes in Computer Science*, pages 44–57. Springer, 2023. doi: 10.1007/978-3-031-47546-7_4. URL https://doi.org/10.1007/978-3-031-47546-7_4.

[22] C. H. Mooney and J. F. Roddick. Sequential pattern mining–approaches and algorithms. *ACM Computing Surveys (CSUR)*, 45(2):1–39, 2013.

[23] B. Negrevergne and T. Guns. Constraint-based sequence mining using constraint programming. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 288–305. Springer, 2015.

[24] H. Pak, L. Asmer, P. Kokus, B. I. Schuchardt, A. End, F. Meller, K. Schweiger, C. Torens, C. Barzantny, D. Becker, J. M. Ernst, F. Jäger, T. Laudien, N. Naeem, A. Papenfuß, J. Pertz, P. Shiva Prakasha, P. Ratei, F. Reimer, P. Sieb, and C. Zhu. Can Urban Air Mobility become reality? Opportunities, challenges and selected research results. *arXiv e-prints*, art. arXiv:2309.12680, Sept. 2023. doi: 10.48550/arXiv.2309.12680.

[25] S. Paramonov, D. Stepanova, and P. Miettinen. Hybrid ASP-based approach to pattern mining. *Theory Pract. Log. Program.*, 19(4):505–535, 2019. doi: 10.1017/S1471068418000467. URL https://doi.org/10.1017/S1471068418000467.

[26] J. Pei, J. Han, and W. Wang. Mining sequential patterns with constraints in large databases. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, page 18–25, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581134924. doi: 10.1145/584792.584799. URL https://doi.org/10.1145/584792.584799.

[27] J. Pei, J. Han, and W. Wang. Constraint-based sequential pattern mining: the pattern-growth methods. *Journal of Intelligent Information Systems*, 28(2):133–160, 2007.

[28] N. Ruseno, C.-Y. Lin, and S.-C. Chang. UAS traffic management communications: The legacy of ADS-B, new establishment of remote ID, or leverage of ADS-B-Like systems? *Drones*, 6(3), 2022. ISSN 2504-446X. doi: 10.3390/drones6030057. URL https://www.mdpi.com/2504-446X/6/3/57.

[29] A. Samet, T. Guyet, and B. Negrevergne. Mining rare sequential patterns with ASP. In *ILP 2017-27th International Conference on Inductive Logic Programming*, 2017.

[30] G. Sterlicchio and F. A. Lisi. Detecting Patterns of Attacks to Network Security in Urban Air Mobility with Answer Set Programming, July 2024. URL https://doi.org/10.5281/zenodo.13135192.

[31] G.-H. Tu, Y. Li, C. Peng, C.-Y. Li, H. Wang, and S. Lu. Control-plane protocol interactions in cellular networks. *ACM SIGCOMM Computer Communication Review*, 44(4):223–234, 2014.

[32] F. Van Den Broek, R. Verdult, and J. De Ruiter. Defeating IMSI catchers. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, pages 340–351, 2015.

[33] Y. Zeng, J. Lyu, and R. Zhang. Cellular-Connected UAV: Potential, challenges, and promising technologies. *IEEE Wireless Communications*, 26(1):120–127, 2018.