SN Computer Science (2024) 5:445
https://doi.org/10.1007/s42979-024-02744-1

ORIGINAL RESEARCH

The LeWiS Method: Target Variable Estimation Using Cyber Security Intelligence

Leigh Chase[1] · Alaa Mohasseb[1] · Benjamin Aziz[1]

Received: 15 June 2022 / Accepted: 22 February 2024

Abstract

Information technology plays an increasingly important role in the provision of essential services. For these systems and networks to be reliable and trustworthy, we must defend them from those who would seek to compromise their confidentiality, integrity and availability. Security intelligence tells us about the tactics, techniques and procedures used by threat actors for these very purposes. In this paper, we introduce a novel method for learning malicious behaviours and then estimating how likely it is that a system has been compromised. One of the difficulties encountered when applying machine learning to cyber security, is the lack of ground truth on which to train supervised techniques. This is often compounded by the volume, variety and velocity of data which is far greater than can be processed using only human analyses. The technique known as LeWiS, created and described by the authors, includes data preparation and processing phases that learn and later predict the presence of threat actors using a model of their behaviours. The method addresses the problems of scale and veracity, by learning indicators of attack via feature extraction from security intelligence that has been obtained through empirical methods. This approach shows promising classification performance for detecting learned malicious behaviours, within synthesised systems' event data.

Keywords Cyber security · Machine learning · Threat intelligence · Estimation methods · STIX · TTPs

Introduction

Information security is an enduring challenge and one whose importance is underscored simply by reading the popular media [3]. The infection of the UK's National Health Service (NHS) by the WannaCry ransomware in 2017 highlights the widespread, real-world impact of sophisticated cyber attacks. It also suggests lessons we can learn about preparedness and the need for continuous evolution of our response efforts [29]. Today's information systems are complex—they combine legacy with emerging technologies, are made-up of heterogeneous systems and provide more varied services than ever before. In parallel, we can see an upward trend in the frequency of cyber attacks as well as a diversification of the actors to whom they are attributed [9]. As identified by Kumar [16], these factors demonstrate the need for creativity and novel approaches to how we construct and apply defensive measures. Machine learning for advanced threat detection is one example of innovation in the field.

There are many ways in which to apply machine learning to cyber security and a similar range of reasons as to why one may wish to do so. Generally, machine learning offers ways to supplement, augment and enhance human efforts within this field. Given the scale and complexity of cyber security, it is simply impractical to scale human endeavour to address comprehensively the problems that it raises. The volume, variety and velocity of attacks, attack techniques and hostile actors exceeds that which can be countered using human expertise and skills, alone. An important motivation for applying machine learning

This article is part of the topical collection "Advances on Web Information Systems and Technologies" guest edited by Joaquim Filipe, Francisco José Domínguez Mayo and Massimo Marchiori.

✉ Leigh Chase
up348663@myport.ac.uk

Alaa Mohasseb
alaa.mohasseb@port.ac.uk

Benjamin Aziz
benjamin.aziz@port.ac.uk

[1] School of Computing, University of Portsmouth, Portsmouth, UK

Published online: 18 April 2024

to cyber security, then, is to build systems that combine human expertise with the scalability of machines.

Whilst often portrayed as a single domain, security actually contains a far greater number of nested topics and important challenges. These are as diverse as software reverse engineering, Security Information and Event Management (SIEM), generative signature techniques and policy-based management—all of which require different methods, algorithms and processing techniques [28]. In this paper we are concerned with the predictive power of machine learning and the use of Cyber Threat Intelligence (CTI) as a framework for knowledge representation. The literature highlights the increasing importance of CTI [12], but research into intelligence-specific learning methods remains limited. Techniques such as those described by Alghamdi [1] and Amit et al. [2] demonstrate applications using host- and network-sourced telemetry. However, they also highlight the risks of over-fitting and the failure to generalise beyond a small number systems. In-line with Shaukat et al. [28], they illustrate the impact that the lack of labelled data has on broadening the utility (and appeal) of learning schemes within this domain. These are critical issues and form the principal motivation for this work.

Herein, we introduce a novel new technique named "Learning With the Structured Threat Information Expression language" (LeWiS). It is a method for estimating target variables within structured security event data, based on supervised machine learning. It trains on intelligence material to learn the semantics for indicators of attack (IoAs), then a behavioural model is constructed and applied to predict whether systems' event data indicates a compromise and if so, by whom. To represent CTI information during training and prediction, the Structured Threat Information Expression (STIX) language is used. STIX is an open-source standard maintained by the Organization for the Advancement of Structured Information Standards [23]. The LeWiS method is designed to address the problems associated with (an absence of) labelled data and the tendency to over-fit. Experimentation using the MITRE ATT &CK framework [17] (for training) and synthetic system events as estimation targets, indicates LeWiS achieves promising classification performance when provided with sufficient information about an attacker's tactics, techniques and procedures (TTPs). The LeWiS method is novel because it operates purely on the semantics of data modelled using STIX. Using IoAs as the supervision data to predict indicators of compromise (IoCs) is a new approach, as is the way in which system events (telemetry) and the IoAs are mapped into a common information space. This means it can be applied between different systems and is both learning and classifying in a fashion that generalises insofar that STIX can model the behaviours of threat actors and the systems they target.

"Related Work" section of this paper discusses related work. "Proposed Approach" section describes the LeWiS technique in detail and includes some worked-examples using real CTI. This section also provides some background on STIX, insofar that it is relevant to the main discussion. "Results of Experimentation" section provides the results of experiments conducted to validate and verify the technique during development. The final "Conclusion and Further Work" section includes conclusions and a short discussion on further work.

Related Work

Specialised research into learning the semantics of hostile actor TTPs remains limited. Much of the prior art focuses on the application of probabilistic, or more recently "deep learning" methods to problems such as anomaly detection, network event classification and behavioural analysis. These efforts include a very broad range of disciplines—examples include two-level Bayesian networks and Markov models for use in unbalanced reporting environments [35], clustering and classification within Internet protocol (IP) packet analysis [8] and the use of random forests in generalised network anomaly detection use-cases [10]. Within the wider literature we see the various parallels between general machine learning challenges [27] and those more specific to cyber security [28]. Outside of academic research, techniques are being used within commercial products by technology providers—such as Darktrace [7], Vectra [30] and Expanse [24]. Commercial implementations tend to focus more on methods that prioritise data for human analysts, underpinned by attempts to find similar and dissimilar behaviours within computer networks. More targeted examples consider the role of machine learning within specialised sub-domains, such as software defined networking [34]. Contemporary research also includes an attacker's perspective on artificial intelligence (AI)— such as the potential for its misuse [15] and the evasion of defensive measures based upon it Xu et al. [32].

The importance of both using and sharing CTI is reflected in government thinking. This is evidenced by special publication 800–150 from the National Institute for Standards and Technology, which identifies shared situational awareness, improved security posture, knowledge maturation and greater defensive agility as the principal benefits of sharing [13]. The UK's Cyber Security Information Sharing Partnership established by the National Cyber Security Centre, actively promotes government-industry sharing of threat intelligence materials for mutual benefit [21]. Fransen et al. [11] discusses the advantaged of using CTI when attempting to improve our understanding of malicious TTPs in enterprise environments. Riesco et al. [25] notes some of

the difficulties associated with CTI-sharing communities—viz. the tendency to consume more than one provides- and introduces an innovative sharing mechanism using a smart contract-type structure, underpinned by blockchain technology. The Authors do not explicitly cover machine learning, but the topics on modelling tactics and the need for ontology are familiar to this work. Prior art by two of the same authors introduces the idea of semantic inference using CTI— making explicit reference to the STIX standard, as well as to the Web Ontology Language [26]. Authors in Jungsoo et al. [14] outline a technique to automatically generate Malware Attribute Enumeration and Characterisation (MAEC) records using STIX-formatted CTI. This idea focuses more on automation and parsing within an intelligence 'workflow', but has clear parallels with the supervised estimation techniques introduced herein.

On the subject of ontology, Blackwell [6] describes an original approach to the formulation and expression of security incident data. This is paired to a three-level architecture for planning and preparing defensive measures. This work is especially notable because it predates what are now common standards in CTI data management. In Ben-Asher et al. [5] the authors introduce semantics as a useful abstraction built upon base data elements— such as the direction of flow, distribution of protocols, packets per hour and related metrics within IP networks. This includes a specific example on command and control channels that is especially relevant to the detection of malware. Importantly, the authors also introduce the idea of granularity and the resolution with which observations are made—showing some accordance with how transferable the resulting ontology is. Many works identify the problems associated with constructing a meaningful abstraction to model the behaviours of cyber actors—Wali et al. [31] sets out the problems and proposes a novel bootstrapping approach, that aims to reduce the burden placed on designers and engineers. This technique combines an existing ontology with a literal text book, demonstrating an approach that seeks to maintain the currency and relevance of the knowledge that has been modelled. Interestingly, this work is a contemporary of the very early release of STIX in 2012–2013.

Most closely related to this paper is Zheng et al. [33], within which the authors explicitly make the link between STIX and machine learning. Crucially, Zheng et al. [33] highlights the constrained nature of how machine learning has been applied in the prior art— namely that it uses limited, very specific data for similarly bounded purposes. The authors' experimentation combined multi-layer perceptron networks with gradient boosting decision trees, to address different facets of the problem domain. Verification found very good classification performance for certain types of attack against web services—as high as 96.2% under some conditions. Finally, Mittal [19] describes the relevance of combining vector and graph-based representations to good effect. The authors introduce a structure they call the Vector Knowledge Graphs that seeks to blend the expressive quality of knowledge graphs with the crispness and functional nature of vectors.

The LeWiS method introduced in this paper offers an alternative to previous approaches. This centres on a new semantic model that combines subjects and object-predicates, with a supervised classification scheme to detect and attribute known-malicious TTPs. This is discussed in "Proposed Approach" section.

Proposed Approach

Modelling causation within cyber security data is difficult for a great many reasons, principal among them is the twin problem of data consistency and completeness [20]. LeWiS attempts to learn the semantics of how malicious actors compromise their targets. This information is articulated using STIX intrusion sets and the relationships they have to attack-pattern, malware and tool objects. Machine learning is applied via supervised techniques and trained using TTP information provided by ground-truth data. Predictions classify the behaviours within system telemetry using the pre-trained model—this data is also represented as STIX. The method is inherently repeatable and designed with automation in mind—similarly, it can be retrained whenever new or revised TTP data becomes available. LeWiS comprises three internal steps: pre-processing, processing and learning, which are combined into a single 'pipeline' of operations. The purpose of the three steps is to compartmentalise the activities taking place within each step. This follows the standard software engineering practices related to coupling and cohesion, but also adopts a style familiar to most machine learning engineers. Specifically, each step combines a series of related actions, the aggregated output of which is passed to the next step until processing is complete. The three steps are as follows:

- *Pre-processing—concerned with data acquisition, normalization, feature extraction and constructing the subject-(object-predicate) data structure*
- Processing—concerned with the constructing the vector representations using the subject-(object-predicate) data
- *Learning—performs the fitting functions and predictions by applying the models to observable event data*

The three steps follow standard conventions for machine learning 'pipelines'. The pre-processing and processing steps acquire and format the source intelligence data such that they can be used for learning. These are required because the 'raw' STIX objects must be parsed into a format that

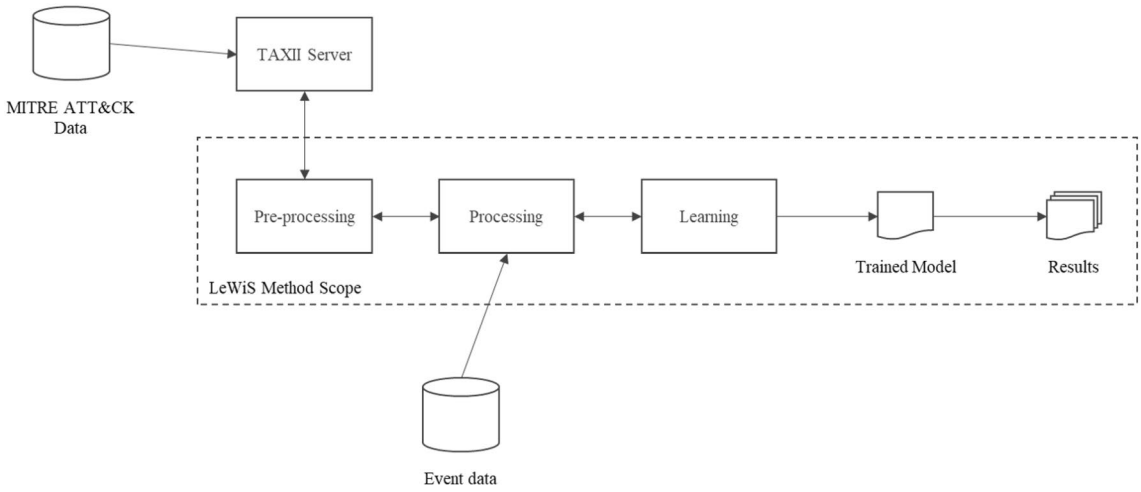

Fig. 1 Steps of the LeWiS method—shown as a simple flow diagram

is suitable as input to the learning algorithms. The LeWiS method uses 'shallow' learning (rather than deep techniques/artificial neural networks) and so a numerical representation of the TTP data within the intelligence is required. The pre-processing and processing steps perform the work required to construct and populate a feature space in which the shallow learning techniques can be applied. These are elements discussed in Pre-processing, Processing and Learning sections. They are also summarised in Fig. 1.

Description of the Dataset

All input data (for training and prediction) are formatted as STIX [23]. Formally, this is a "schema that defines a taxonomy of cyber threat intelligence" [23] and employs a linked-data structure whose information architecture describes four main entities:

- STIX domain objects (SDOs)[1];
- STIX cyber observable objects (SCOs)[2];
- STIX relationship objects (SROs)[3]; and
- STIX meta objects (SMOs).[4]

The STIX standard includes a data model for each type, which extends the common data types also defined in the standard. The common data types include binary, boolean, dictionary, enumeration, external reference (effectively pointers, that may be strings or hash values), floating point numbers, hashes, hexadecimal numbers, identifiers (which are themselves integers), kill chain phase, list (array-like), open vocabulary (string representation of a customisable language), string and timestamp. The data model for each object type is quite large, so references are added as footnotes in the list above. To work with STIX entities, software development kits (SDKs) typically model them as objects within a program—following standard Object Oriented Programming (OOP) conventions. Each object implements class structures consistent with the STIX standard, with member variables that contain the object's data stored in standard data types. The LeWiS technique, in common with other STIX-processing programs, serialises all STIX objects using Java Script Object Notation (JSON). Such objects are then transferable between programs, systems and networks. Actor behaviours (TTPs) are articulated using the intrusion-set SDO, which in turn has relationships with other domain objects modelled using the SRO entity. Each relationship has a dedicated SRO whose source, target and type attributes

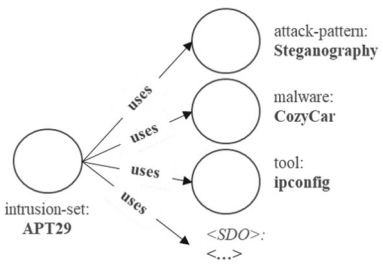

Fig. 2 Summary of intelligence relationships

[1] https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#_nrhq5e9nylke.

[2] https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#_mlbmudhl16lr.

[3] https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#_cqhkqvhnlgfh.

[4] https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html#_mq8oo9k9rb2.

Table 1 STIX object type counts

Type	Number
ATT &CK STIX objects	
Relationship	10990
Attack-pattern	692
Tool	70
Malware	424
Intrusion-set	125
Course-of-action	266
X-MITRE-tactic	14
X-MITRE-matrix	1
Marking	1
Identity	1

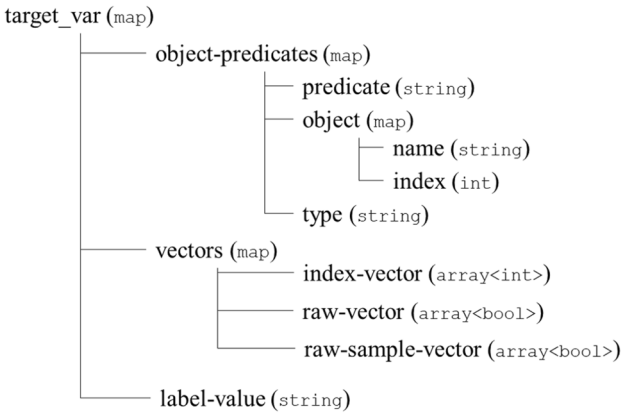

Fig. 3 Example of the SOP map structure

describe the required association. This is shown in Fig. 2. SROs can be used to link any of the other three STIX entities together. This paper focuses only on the relationships from intrusion sets to attack-pattern, malware and tool types.

SDOs and SROs are used for training and SCOs / SDOs for variable estimation. Training was completed using the MITRE Enterprise ATT &CK dataset, which was acquired from the organization's official TAXII (Trusted Automated Exchange of Intelligence Information) server [17]. It was chosen for its rich, real-world (not synthetic) intelligence on threat actors' TTPs. Given the MITRE Corporation's standing within the field, it is considered a trusted source of ground-truth. The data used in this paper is a 'snapshot' whose contents reflect the latest information available at the time it was downloaded (May 2021). The ATT &CK data is packaged as a single STIX Bundle [22], within which each SDO has a unique identifier contained within its id attribute. ATT &CK contains 10,990 SROs that model relationships between 1594 SDOs. The domain objects are distributed thus: 692 attack-patterns, 125 intrusion-sets, 424 malware types, 70 tools, 266 course-of-action records, 14 x-mitre-tactics, 1 x-mitre-matrix, 1 identity; and 1 marking definition. This is shown in Table 1.

Pre-processing

Pre-processing is used to transform STIX data into a format suitable for the Processing functions. This step selects only the features that are required by the subsequent process step, which are modelled as STIX object types within the scheme. The extracted object types are given in Table 1. At the time of writing, the pre-processing step does not extract the 'Marking-Definition' object type as it contains only metadata about the intelligence and is not used by LeWiS.[5] Similarly, whilst the `X-MITRE-Tactic` and `X-MITRE-Matrix` pre-processed and parsed into STIX objects, they are not used within subsequent steps (processing or learning). This is because they are part of MITRE's internal information model rather than part of the STIX base standard. The base standard is applied to offer greater generalisation and at the time of writing, the additional MITRE objects do not extend the LeWiS method in a meaningful way. All data follow the same preparation techniques and the approach can be applied to any valid STIX-formatted CTI. This allows generalisation beyond single (or a small number of) systems. The source STIX datasets are parsed into a map of subject-(object-predicate) (SOP) data structures, the design for which is original to LeWiS. The SOP map's keys are the intrusion-set descriptors extracted from the SDOs. Their corresponding values are a nested map that contains the relationship data. This is shown in Fig. 3.

This structure offers a way to efficiently serialize the data used to train the supervised learning model. It is advantageous because it is simple yet enforces a formal structure. In turn, this ensures TTP data are expressed consistently and in a platform-agnostic fashion. The SOP map is stored and processed in JavaScript Object Notation (JSON) format:

"target-var": {"object-predicates": [{"predicate": string, "object": "name": string, "index": int, "type": string}], "vectors": {"attack-pattern-vector": [int], "malware-vector": [int], "tool-vector": [int], "raw-ap-vector": [bool], "raw-m-vector": [bool], "raw-t-vector": [bool], "raw-sample-vector": [bool] }, "label-value": int }

We can consider a simple example using the "APT29" intrusion-set.

[5] This metadata would likely be used to define information handling controls for use in an enterprise IT environment. This could include privacy markings, data classifications or use of the 'Traffic Light Protocol' method.

{"predicate": "uses", "object": {"name": "Malicious Link", "index": 189}, "type": "attack-pattern" }

This indicates that this actor "uses" the "Malicious Link" attack-pattern and that this pattern can be found at offset 189 in the set of all attack-patterns included in the STIX bundle. For APT29, the training set contains 109 Object-Predicate structures like this. These records act as qualifiers for the behavioural associations of the target intrusion-set—they are extracted during Pre-processing and parsed into this format. The Object-Predicates represent the qualities by which we can distinguish the classification targets—viz. they define the separability criteria that give the entire process meaning. Ultimately, this is a 'template' for learning features extracted from STIX data sets and provides a scaffold for preparing and fitting samples for classification. For later reference, the Subject (root key of the map) is what the learning algorithms are attempting to classify—this is discussed in "Learning" section.

Processing

Processing generates vector representations for use in the Learning phase. Vectors are required because classifiers expect array-like input with consistent types—neither 'native' STIX nor the SOP structure conform to this. Processing has two inputs: SOP data for training and SCO data for prediction. Outputs are n-dimension vectors which are then passed to the Learning phase. These are stored under the vectors key in the SOP map data structure—meaning that eventually both the semantic relations and the vectors used to train the model are contained within a single data structure—extending the principles outlined by Mittal [19]. The intrusion-set relationship attributes are stored in the Vectors key of the SOP data structure. It contains another nested map within which are the vector representations of the object-predicates. LeWiS defines two types of vector: index vectors and raw vectors. The former contains the unique indices (integers) of all SDOs with which the target variable has an outbound relationship. For example, the APT29 has the following relationships with tool SDOs (non-exhaustive):

{"*predicate*": "*uses*", "*object*": {"*name*": "*AdFind*", "*index*": *5*}, "*type*": "*tool*"},

{"*predicate*": "*uses*", "*object*": {"*name*": "*Tor*", "*index*": *39*}, "*type*": "*tool*"},

{"*predicate*": "*uses*", "*object*": {"*name*": "*Mimikatz*", "*index*": *69*}, "*type*": "*tool*"}

For each intrusion-set, index vectors for the associated attack-pattern [Eq. (1)], malware [Eq. (2)] and tool [Eq. (3)] objects are created. Note that the index vector contents are not ordered. As the volume of underlying CTI increases, these allow for fast reading of the attack-pattern, malware and tool data for a given intrusion-set. For APT29 the complete tool vector is [Eq. (4)].

$$apt29_{attack-pattern-vector} = \langle ap_0, ap_1, ..., ap_n \rangle \quad (1)$$

$$apt29_{malware-vector} = \langle m_0, m_1, ..., m_n \rangle \quad (2)$$

$$apt29_{tool-vector} = \langle t_0, t_1, ..., t_n \rangle \quad (3)$$

$$apt29_{tool-vector} = \langle 5, 56, 59, 64, 62, 27, 37, 39, 69, 65 \rangle \quad (4)$$

Note [Eq. (5)] is the set of object-predicates in the SOP map for this intrusion-set.

$$|apt29_{x-vector}| = |op_x| \quad (5)$$

Modelling Relationship Occurrence

The raw vectors are a boolean representation of each intrusion set's relationships, based on a simple "has" or "has not" evaluation. The total length of each vector is equal to the cardinality of the corresponding set of SDOs within the STIX bundle and is initialised with 0 values. Because the ID of each SDO is unique, if a target variable has an association with an SDO a 1 is stored in the vector at the location defined by the index. Therefore, each intrusion-set entry in the SOP map contains a Boolean vector named "raw_< *xxx* >_vector", where *xxx* is the short name for the SDO type. Continuing the example started above, there are 70 tool SDOs in the ATT &CK data and "APT29" has an association with 10 of them. The $apt29_{raw\_t\_vector}$ for this intrusion-set has 70 components of which only 10 take the value 1 (at the indices specified in $apt29_{tool-vector}$)—all other values are 0. The same approach is taken to populate the $apt29_{raw\_ap\_vector}$ (attack-patterns) and $apt29_{raw\_m\_vector}$ (malware).

Alpha–Beta Re-sampling

In reality, threat actors are not characterised by the sum of all of their behaviours. Figure 2 shows APT29 is linked to the "Steganography" attack-pattern, the "CozyCar" malware and the "ipconfig" tool. The ATT &CK data contains 109 SROs for which the source reference is the ID of the APT29 intrusion-set. STIX-formatted intelligence does not include occurrence information for TTPs (only for observations). That is to say, it does not specify how frequently a given intrusion-set uses any given attack-pattern, malware or tool. If only the entire feature vector for each intrusion-set was used to train the supervised model, then the learning algorithm could only make classification decisions when all of the attack-pattern, malware and tool relationships were presented. In plain terms, when all of the target threat actor's TTPs were contained in a single piece of telemetry! To address this, LeWiS includes a technique referred to as "α -β re-sampling" (designed for LeWiS by the authors), which

creates additional vectors [Eq. (6)] whose components are subsets of the original 'full' feature vector [Eq. (7)].

$$|f(v_x, \alpha, \beta)| \tag{6}$$

$$f(v_x, \alpha, \beta) \subset v : v = \{x_0, x_1, ..., x_n\} \tag{7}$$

where

- f is the re-sampling function,
- v_x is the intrusion-set vector
- $_x$ is the class label
- α is the re-sampling rate
- β is the re-sampling mask
- v is the set containing the original vector components

The processing phase summarises all of this information in a single array-like structure. It contains the attack-pattern, malware and tool vectors [Eq. (8)].

$$v_x = \langle\langle ap_0, ap_1, ..., ap_n\rangle, \langle m_0, m_1, ..., m_n\rangle, \langle t_0, t_1, ..., t_n\rangle\rangle \tag{8}$$

A naïve way to approach re-sampling is to simply create sets from the feature vectors, then to calculate the Power Set $\mathcal{P}(v_x)$ in order to create component vectors representing each combination of elements. This resolves to Eq. 9, which does not scale well if the size of the feature vectors (the value of n) becomes very large.

$$_nC_k = \frac{n!}{r!(n-r)!} = \binom{N}{k} \tag{9}$$

Selecting Values for *alpha* and *beta*

In CTI it is practical to assume that the size of the attack-pattern, malware and tool vectors will increase over time. The α and β terms are also used to avoid the "curse of dimensionality" [4]. The α value is array-like and contains integers used to define the maximum number of samples to be taken from each sub-vector [example, Eq. (10)]. The sampling values for each sampled vector are user-defined but must be in the interval [0, $|v|$], where v is the feature vector for which the sample rate is defined (attack-pattern, malware or tool). Setting a value is a matter of context and for most cases where the CTI input data are of a manageable size, each vector can be used in its entirety. The α sampling approach is useful when the size of these vectors becomes large, or when the user wishes to target only a specific subset of attack-patterns, malware or tools.

$$\alpha = \langle max_{ap}, max_m, max_t\rangle \tag{10}$$

The β value is also array-like and contains three vectors of integers [example, Eq. (11)]. It is used to mask values that are not of interest to the re-sampler. The three component vectors represent the attack-pattern, malware and tool SDOs thus allow users to be selective over what is not used during re-sampling. This is of particular use to those building models to learn TTP patterns used against specific systems. For instance, to ignore all malware and tool SDOs relating only to Microsoft Windows operating systems where the user is only concerned with Linux targets.

$$\beta = \langle\langle 12, 34\rangle, \langle 7\rangle, \langle 28\rangle\rangle \tag{11}$$

Learning

LeWiS functions in a de-coupled fashion and does not prescribe any particular learning methods. What LeWiS is really providing is a semantic approach to learning actor TTPs, using the SOP data structure. This avoids the need to develop highly specialized logic that applies only on a system-by-system basis. The bulk of the work done by this technique is representational—viz. building a domain model that is consistent, platform-agnostic and can be used for both training and prediction. LeWiS has been tested using support vector machine (SVM), decision tree classifier (DTC) and logistic regression (Logit) algorithms. The normalized confusion matrix was used to measure classification performance. The results are discussed in "Results of Experimentation" section.

These techniques were chosen because they all support large class registries, handle multiple data types and are interpretable. We avoid the deep learning methods used by Zheng et al. [33] and Wali et al. [31], to preserve transparency and 'explainability' within the Learning phase. Furthermore each represents a different approach to classification and offers a high-degree of configuration potential. As with other aspects of the technique, the process for choosing the 'best' classifier is domain-specific and not something that can specified without particular uses in mind. All development, training, testing and configuration activities were completed using the Python "sklearn" (Sci-kit Learn) module—these techniques were not implemented from scratch. Currently, the approach affords limited options for re-training and does not include reinforcement techniques. The ATT &CK data used to train with LeWiS was imported into the local work space—it was not 'online' in any sense and so the models reflect the "as-was" view of the data, according to when it was acquired.

System telemetry is noisy and the detection methods for actor TTPs must combine events that happen over time. These events are parsed into SCOs containing the corresponding instance data (one per event), which are then aggregated to populate an Observed Data SDO.

Constraints and Drawbacks

Generally, any attempt to detect threat actors using structured CTI will introduce limitations and compromises. Consequently, there are practical considerations to be made when seeking to apply LeWiS at scale. Whilst STIX provides a useful system- and technology-agnostic language, basing LeWiS upon it in the ways described does introduce some constraints. These are summarised as follows:

1. Intelligence-oriented design: LeWiS operates exclusively on CTI rather than other types of cyber security data. The use of STIX objects (SDOs, SROs and SCOs) was an intentional design decision and whilst this allows it to be applied across different systems, this flexibility comes at the expense of working with other types of security data.
2. Representational completeness: in parallel with it being intelligence-oriented, LeWiS can only be applied to CTI expressed using STIX and objects that are fully compliant with its conventions. Data that cannot be (or simply are not) modelled using the STIX language cannot be processed using LeWiS.
3. Learning power: the learning potential of the technique is bounded by how successfully SDOs and SROs can be used to train a model. Then, it has a further dependency on how well system event data can be mapped as SCOs that are used as input to the learning process.
4. Specificity: STIX is a standard and is therefore prescriptive. The properties that any SDO, SRO and SCO may have are defined in the standard ([23])—as are the data types those properties can take. LeWiS, therefore, can only ever be as specific as the standard allows.
5. SCO parsing overheads: to perform estimation, LeWiS requires all input data to be parsed in to the Observable (SCO-formatted) objects described above. In enterprise networks this is not a trivial task and requires an explicit mapping to be defined for each data source type. Further work is required to identify which sources yield the most effective classification performance when modelled as SCOs.
6. Operational readiness: LeWiS is probabilistic and so estimating the presence of an attacker within a network is not a discrete-valued process. Instead it is a way of suggesting where specialists (analysts, engineers, etc.) should focus their attention based on the presence of behaviours that appear consistent with a trained model of suspicious activity.
7. Distinguishable events: some legitimate (non-malicious) actions are difficult (or very difficult) to distinguish from illegitimate (malicious) actions and LeWiS does not introduce any additional measures to improve this. Whilst this is by no means a LeWiS-specific problem, the technique will struggle to be accurate when a threat actor's TTPs are closely related (or are mimicking) 'proper' administrative practices—examples include webshells and potentially unwanted applications (PUAs).
8. Inheritance of the STIX language: LeWiS is constructing an intermediate representation (IR) of the underlying STIX data—which are themselves an IR of the 'raw' CTI. Equally, LeWiS is not in itself, a language. This means LeWiS inherits the constraints of the STIX language, which relates to the comments about representational completeness (above) and the limits of what LeWiS can model. However, this approach further limits the technique by offering no way to model the organisation of a system to which LeWiS is being applied. That is to say it operates purely on intelligence, because that is how STIX is designed. This constrains the performance and ultimate effectiveness of the technique.

The impact of these constraints depends on how one might wish to apply the LeWiS method. Generally, the overall result is that they add computational overheads and complexity to already challenging intelligence processing workloads. Similarly, effective use of LeWiS requires domain-specific knowledge and a detailed understanding of both STIX and the construction of ensembles of shallow machine learning techniques. Further consideration of these topics, including potential enhancements and mitigations, are discussed in "Conclusion and Further Work" section.

Results of Experimentation

Whilst this is an exploratory technique, the approach has shown promising results when classifying intrusion-set objects. Applying LeWiS to the latest ATT &CK data (version 9, at the time of writing) yields 125 unique class labels, which includes the null-actor. The ATT &CK training data includes intelligence on 124 intrusion sets and 1186 objects of types attack-pattern, malware and tool. Sparse matrices within SOP structures were common simply because ATT &CK contains intelligence on a broad range of TTPs. This variety means intrusion-sets or attack-patterns can make good discriminators. The SOP generation counts are show in Fig. 4.

The component feature vectors and resulting 'full' feature vector are defined in Table 2. Re-sampled vectors each have the same dimensions. When applying LeWiS to the ATT &CK data, it generates 3138 SOP structures within the map that span all intrusion-set SDOs.

The data within MITRE ATT &CK is different for each intrusion set. This is because more information is known

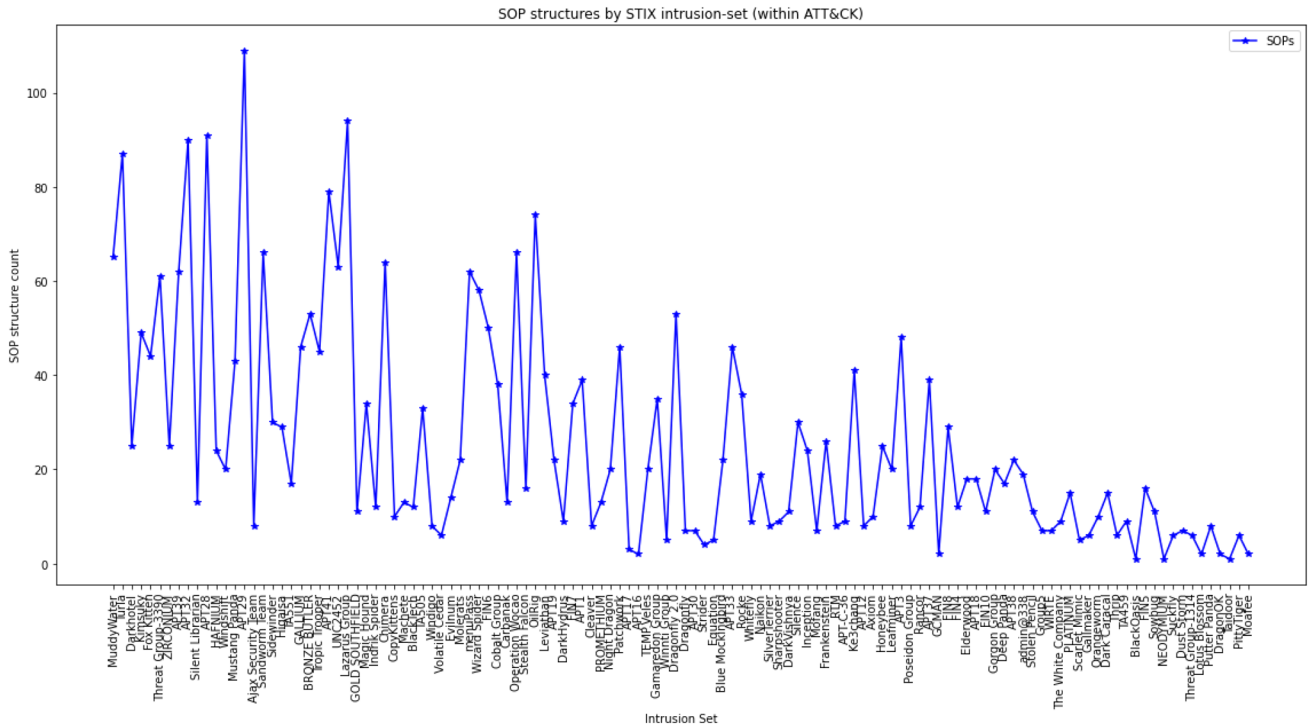

Fig. 4 SOP object counts by intrusion-set

Table 2 Vector definitions

Type	Size
Vectors	
Attack Pattern	692
Malware	424
Tools	70
Full feature	1186

about the TTPs of some actors than it is for others. Consequently, this yields different representations when LeWiS is applied. For instance, a comparatively large amount is known about group APT29 and their activities. LeWiS generates 152 SOP records, an attack pattern vector of length 106, a tool vector of length 14 and a Malware vector of length 32. We can use the shorthand `APT29{SOP:152, AP:106, T:14, M:32}` to express this. Similarly, the ATT &CK data contains a good amount of information on Kimsuky—specifically `Kimsuky{SOP: 99, AP: 89, T: 4, M: 6}`. For actors about which far less TTP information is known, we see a proportional reduction in the number of SOP objects and the length of their vectors. The groups Elderwood and BlackOasis are good examples: `Elderwood{SOP:18, AP:9, T:0, M:9}` and `BlackOasis{SOP:1, AP:1, T:0, M:0}`. A full listing of the SOP objects and vector sizes for each actor is provided in Table 7, within the Additional Tabular Data in "Appendix" section.

Testing was completed using SVM, DTC and Logit learning algorithms. These were evaluated and the best models chosen according to their average classification accuracy. Specimen vectors were synthesised to test the classifiers, however as these are not created from live telemetry the output is considered advisory. Initial exploration suggests the SVM, DTC and Logit techniques produce similar performance—maximum average accuracy was 59.1%, minimum was 43.7% using a common set of synthesized vectors. Representative classification performance is provided in Table 8, which was calculated and averaged over multiple runs/configurations of the technique.

A logical extension to this research is greater evaluation of how the re-sampling techniques can improve overall performance. The technique proved predictably sensitive to 'hedging' its classification decisions where intrusion-sets exhibit common features from across the attack-pattern, malware and tool objects. This offers some insight into the practical separability of actor TTPs using these SDOs, but it also suggests further analysis is required to understand these sensitivities more fully. The α-β re-sampling technique provides some mitigation by compensating the 'reporting bias' within the training data (where the distribution of TTP information is unbalanced). Using the ATT &CK data, the depth

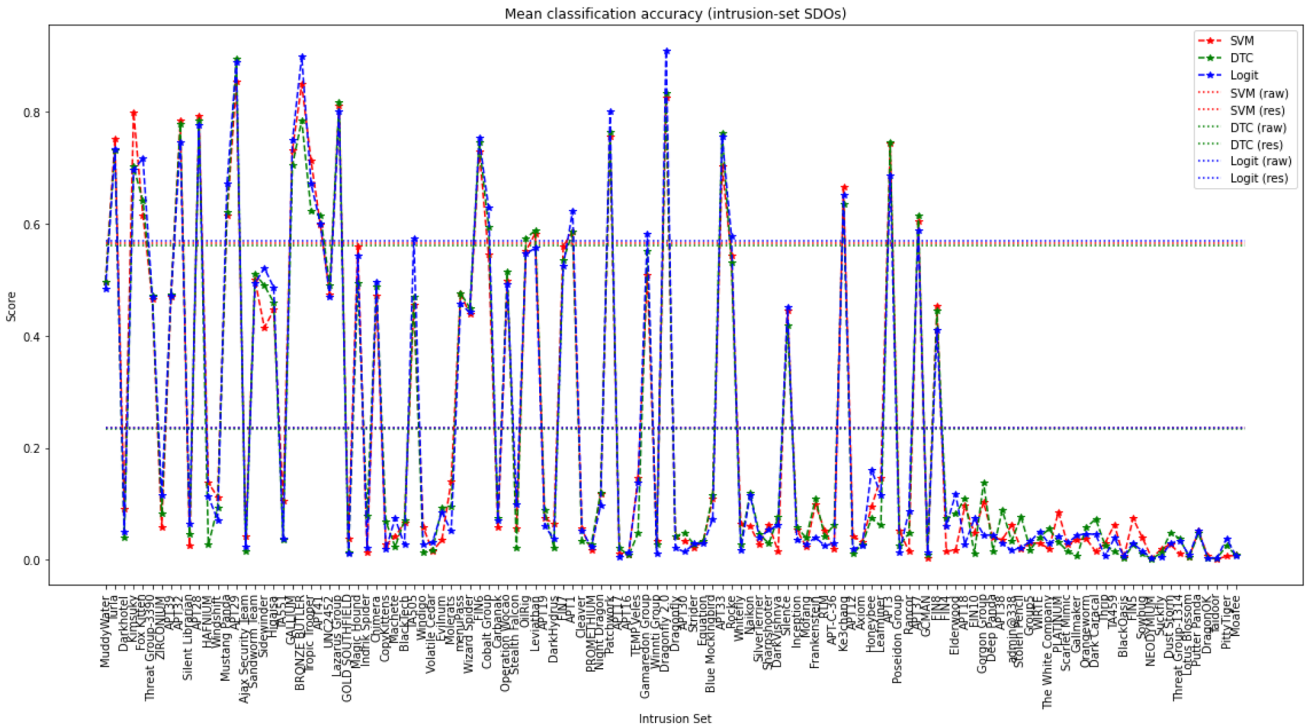

Fig. 5 SOP Classification Scores

of information on attack-pattern and malware SDOs mean the classification logic is typically biased towards these types. As stated in the "Introduction" and Related Work sections, the abstraction and knowledge representation layers built atop STIX are a vital part of the technique's portability and generalisation. Performance across the three algorithms varied by small degrees and the general trend confirms that actors with more ranging TTP information yield the highest classification performance. The training set contained a notable imbalance in the number the attack-pattern, malware and tool SDOs. The first two are far more populous than the last, however the combination of attack-pattern and malware relationships appears to be more indicative of specific actors when all three SDOs are present. Where tool types were dominant these proved a positive discriminator. Interestingly, this suggests the technique might be effective in detecting actors involved in 'living off the land' attacks. Generally, in the case of actors for whom the training data was sparse, training performance was far lower than is desirable. The performance scores are shown in Fig. 5, which graphs the mean accuracy of the classifiers with respect to each Intrusion Set. The horizontal lines show confidence thresholds weighted by these scores for each classifier. For each classifier there are two lines representing the 'raw' and 're-sampled' results. The first range (beginning $y = 0$ to the first line) is effectively zero-to-low confidence, whilst the middle range is low-to-medium. The upper range represents the highest confidence—i.e., where the classification results indicate human analysis / intervention is required. These ranges are included for illustrative purposes and to frame the classification results in a realistic context—viz. one in which decisions need to be made based upon the confidence inspired by a classifier's accuracy.

Peak performance (not averaged) was produced for the APT29, Dragonfly 2.0 and BRONZE BUTLER intrusion sets—reaching the 89th percentile under optimum conditions. With little-or-no re-sampling the overall classification performance was poor—giving a mean of 26.8%. This was caused by the sparse data in evidence for certain intrusion sets or where there was considerable duplication of TTPs between actors. When suppressing sets that gave very poor performance and performing basic re-sampling, the mean performance rose to 59.1%, with 15 intrusion sets performing well above this. Performance was markedly worse for intrusion sets about which little is known—this is as one might expect for these actors the classifiers scored poorly because the lack of TTP data meant there was little to discriminate these actors from others. DragonOK and Taidoor classification was especially poor—effectively yielding zero. This could likely be addressed with weights that compensate for these tendencies but this was not attempted by this research. The lack of TTP data also meant α-β re-sampling was impractical. The principle difficulty is that when re-sampling is applied on actors for whom a large amounts of TTP

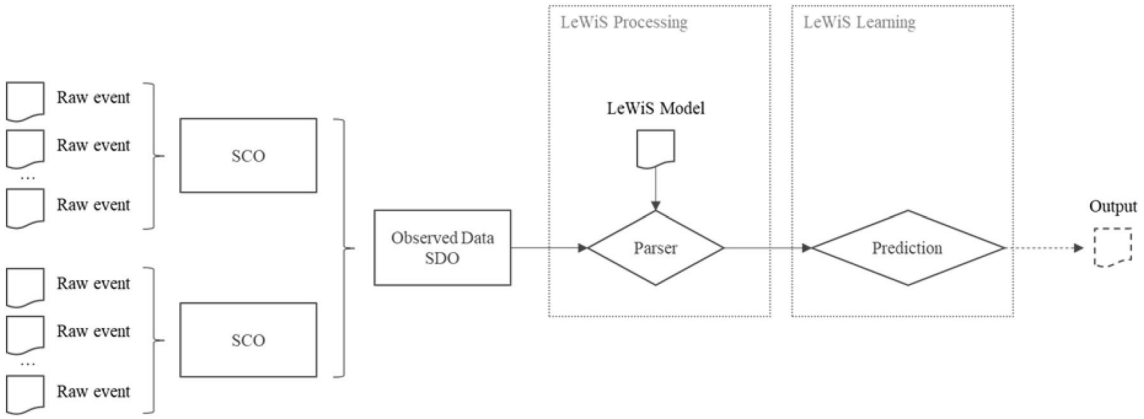

Fig. 6 Telemetry processing using SCOs and SDOs

data was available, the re-sampled vectors may contain the same information as the full (raw) feature vector for these 'lesser known' intrusion sets. This is problematic because if a mask vector is provided for β when re-sampling vectors for the 'greater known' sets, it is possible that this may inadvertently remove relationships that are statistically significant to the classification decision. In practice, there is no way to know this without relying on input from expert users.

Extended Experimentation

The MITRE ATT &CK data are maintained by an open source community of active contributors, meaning the CTI contained within changes over time. This is helpful when assessing techniques such as LeWiS because it is internally consistent and allows one to meaningfully evaluate how well an approach generalises. Table 6 is similar to Table 1 but shows the distribution of STIX objects within ATT &CK data obtained in April 2022 [18], rather than April 2021 [17]. By way of context, MITRE have adopted a twice-yearly release schedule for ATT &CK that features both an April and an October release. Between these times additional CTI may be added, but MITRE maintain this schedule to provide some formality to the overall process. Of greatest note within the newer data is the number of SROs: 15,262 in v11 (April 2022) versus 10,990 in v9 (April 2022). This indicates a higher level of "connectedness" (38.9%) despite far more modest increases in actual intelligence objects (attack-patterns: 3.9%, intrusion-sets: 9.6%, malware: 19.58%, tools: 11.4%, courses-of-action: 0.38%). Thus, the v11 data provides a higher density of relationship modelling than the v9 data and this is significant, because analysis of experiments run with LeWiS on v9 data suggests strong relationship modelling (i.e., between SDOs) improves overall predictive performance. To extend the experimentation described above, LeWiS has also been applied to the v11 data and the additional SROs (and SDOs) that it contains (Fig. 6).

The v11 ATT &CK data yields a different profile of LeWiS SOP objects by intrusion-set and this is illustrated in Fig. 7. The contrasting measures of aggregated classification performance are shown in Figs. 5 and 8. Broadly, the results strengthen the hypothesis that a higher resolution of relationship data improves classification performance for the affected intrusion-sets. The caveat given here is important: "certain intrusion-sets", rather this than simply conferring a more widely-observed benefit. Though overall performance is improved, average classification is not (in practice) a useful measure because the ability to repeatably classify certain actors well (and not others) is surely of more use than a technique that provides middling (and therefore indecisive) performance across a range. Use of α-β re-sampling again improved performance, where re-sampled vectors found peak classification performance of 0.896 (SVM), 0.898 (DTC) and 0.894 (Logit). Interestingly, Fig. 8 illustrates the slightly more eccentric performance of LeWiS using the v11 data—showing stronger separability between classes than in the v9 set. The top ten classification performances for each classifier are given in Tables 3, 4 and 5. Whilst overall performance was mildly improved (averages by classifier), the v11 data shows more distinctly where the technique works well and where it does not (Table 6).

This eccentricity serves a useful purpose in that it informs the decision: "Using the ATT &CK data, for which intrusions sets can I attempt a meaningful detection attempts with LeWiS?". Besides what can be achieved with tuning and optimisation (such as α-β re-sampling) this examination sheds light on the types of data needed to make effective classifications. In turn, this is useful in the context of CTI development and forensic readiness planning. Ultimately, LeWiS is a parametric modelling technique and its performance appears biased towards datasets that have a stronger relational component—this is to be expected, given how the SOP structures are constructed and then used for classification. The telemetry processing techniques (Fig. 6) also

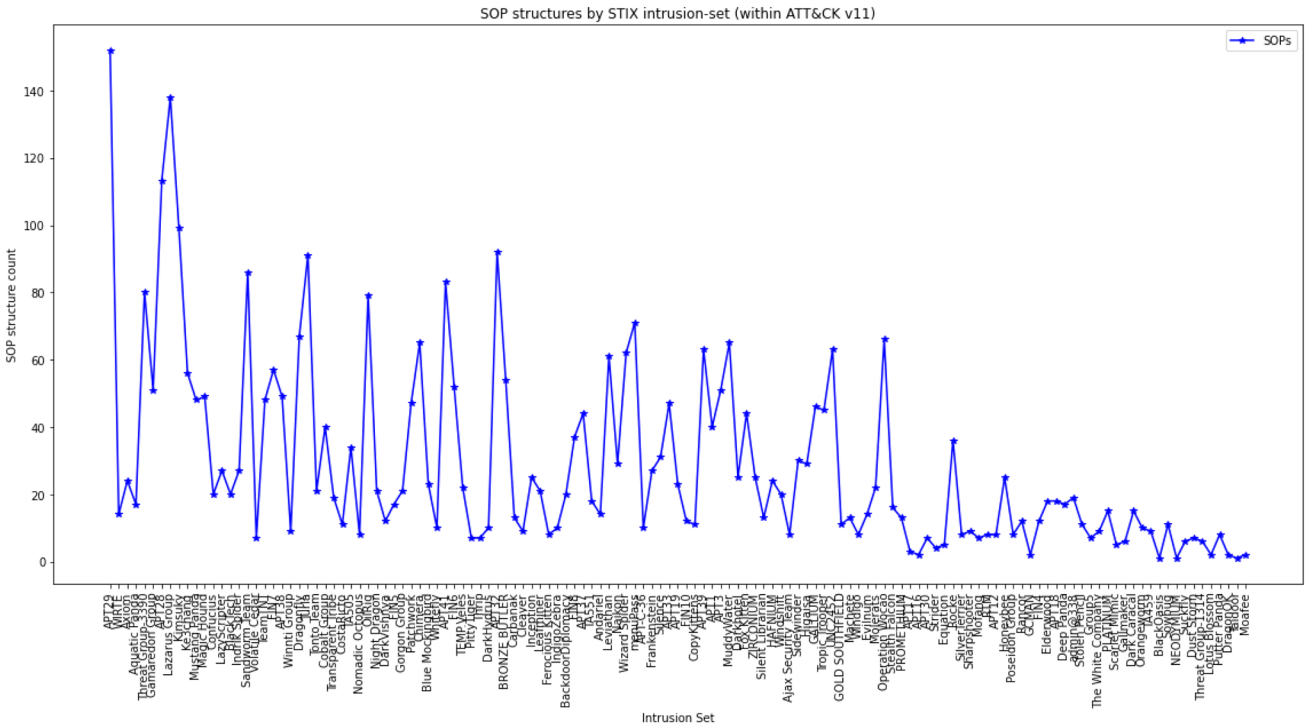

Fig. 7 SOP object counts by intrusion-set (v11 data)

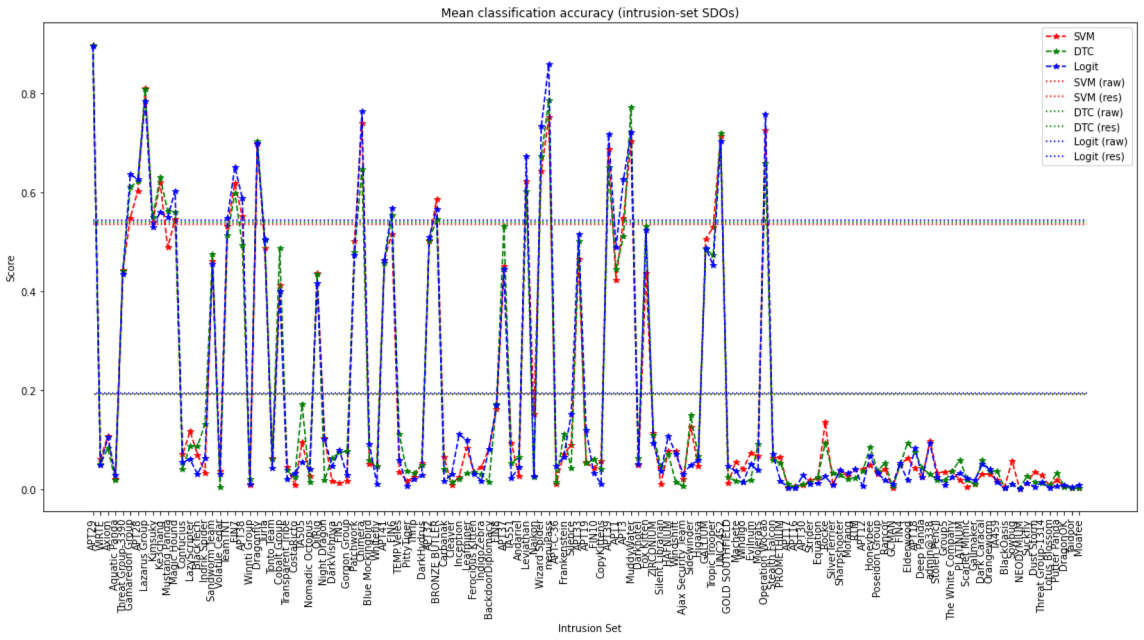

Fig. 8 SOP classification scores (v11 data)

influence how well represented the system under review is, in the context of the classification space. Relational data within the training/truth set, therefore, has a significant bearing on the technique's overall effectiveness. An interesting extension for LeWiS would be to make more use of "property data" in combination with the relational data that has been described. Here, property data refer to information about specific entities within the SDO space—such as

Table 3 Top 10 classification performance, SVM

Intrusion-set	Accuracy	Precision
SVM		
APT29	0.896	0.697
Lazarus Group	0.810	0.700
menuPass	0.751	0.622
Chimera	0.740	0.583
Operation Wocao	0.723	0.666
UNC2452	0.713	0.519
MuddyWater	0.703	0.482
Dragonfly	0.697	0.511
APT39	0.687	0.596
Wizard Spider	0.643	0.631

Table 4 Top 10 classification performance, DTC

Intrusion-set	Accuracy	Precision
DTC		
APT29	0.898	0.714
Lazarus Group	0.808	0.748
menuPass	0.785	0.614
MuddyWater	0.772	0.610
UNC2452	0.719	0.667
Dragonfly	0.704	0.624
Wizard Spider	0.672	0.572
Operation Wocau	0.658	0.589
APT39	0.650	0.601
Chimera	0.646	0.504

Table 5 Top 10 classification performance, logit

Intrusion-set	Accuracy	Precision
Logit		
APT29	0.894	0.758
menuPass	0.858	0.711
Lazarus Group	0.784	0.590
Chimera	0.783	0.521
Operation Wocao	0.757	0.644
Wizard Spider	0.733	0.557
MuddyWater	0.721	0.489
APT39	0.718	0.526
UNC2452	0.703	0.601
Dragonfly	0.700	0.630

descriptive information for attack-patterns, source code for malware and execution paths for tools (for example). Intuitively, such information could be modelled as a Label-Property Graph and these ideas are discussed as part of "further work" in "Conclusion and Further Work" section, below.

Table 6 STIX object type counts (v11)

Type	Number
ATT &CK STIX objects (v11)	
Relationship	15262
Attack-pattern	719
Tool	78
Malware	507
Intrusion-set	137
Course-of-action	267
X-MITRE-data-component	109
X-MITRE-data-source	38
X-MITRE-tactic	14
X-MITRE-matrix	1
Marking	1
Identity	1

Conclusion and Further Work

The overall classification performance is notable for intrusion-sets on whom the training corpus contains a suitable volume of information. Specifically, it is the relational data (modelled as SROs) within the training/truth set that have the largest impact on the technique's effectiveness. Performance is improved using the re-sampling process applied to the SOP map and so the more varied and voluminous the training data are, the more re-sampling can be effectively performed to 'tune' performance. The utility of this approach is also demonstrated through the ability to create specimen vectors for prediction against models trained on real CTI data sets. This finds broader applications within intrusion detection system and firewall testing, or when simulating security incidents for the purposes of personnel development. The results presented in "Results of Experimentation" section and Tables 3, 4 and 5 provide a broad sense of the method's overall classification performance. However, it is clear from the results that the volume and specificity of data within the underlying CTI is a significant influencing factor. Where the SOP structures are plentiful and the STIX objects well-connected, the performance is promising—with the intrusion-sets `APT29`, `Lazarus Group` and `menuPass` yielding strong accuracy and precision. The intrusion-sets for which there is far less data reduced classification performance for the entire MITRE data quite substantially. This offers the basis for determining to what degree one might trust the results of the LeWiS method, based on the intrusion-set it is trying to detect. Similarly, where classification performance is consistently poor (such as `FIN5`, `Gorgon Group` and `Night Dragon`) additional intelligence is required to use the approach meaningfully. In these cases one can see how the standard MITRE ATT &CK data are insufficient in volume and STIX object connections to

muster genuine human confidence. Conversely, for those concerned about the presence of specific threat actors the technique does offer a way to target and focus their intelligence gathering operations. For the more 'mainstream' actors, the technique provides acceptable classification performance and so could be considered reliable. Using only the MITRE data, therefore, suggests for performance that is 'better than guessing' one should focus on the intrusion-sets for which a sufficiently large collection of SOP structures can be created.

The multi-class implementations for each of the classifiers tested set the weights for each class to 1.0. This was done to avoid introducing skew or bias that was not inferred within the training data, but also because predictions were made using only specimen vectors. Real-world scenarios would introduce greater context given by the type of system being monitored and business-level information about the threats faced. Operators applying LeWiS to actual systems may wish to bias the classification decision depending on factors, such as:

- '*Guilty knowledge*' held by a user that would inform proper classification decisions, but cannot be (or is not) encoded within the learning methods;
- Trustworthiness or known-accuracy of the intelligence on which the LeWiS model was trained;
- To reflect the quality, or some other attribute, of the data provided by the system under scrutiny that affects the classification results; and
- To use the classification to scale, or become a coefficient of, a value external to LeWiS process—such as a calculated risk score.

Whilst the Boolean vectors bring relatively large dimensionality, they are simple-valued and have comparatively low storage complexity. A more elegant solution may be preferable in future iterations however, since the vector sizes scale linearly with the growth of intelligence material and they will likely become unwieldy. Improvements can also be made to the re-sampling function by applying masking operations (such as exclusive-OR logic) to create the combinations required. It is interesting to consider whether additional semantics might improve overall performance—for example, the introduction of second-order logic and conditionals that do not treat all relationships as equal. The ideas that underpin the SOP data structure could be extended to include statefulness with respect to the actor. This might further qualify their presence within a network/system and may also give some insight into what further actions they might perform. This may be especially useful in real-world scenarios, where an actor has already compromised some part of a system and those charged with its defence seek to understand how the threat could move laterally or gain a toehold in other systems. This initial, exploratory version of LeWiS is attempting to simply determine the presence of an actor—in reality this resolves to a binary classification of the system under review being in one of two states: compromised or not-compromised. Further development of LeWiS could see it applied in a more differentiated fashion such that it can work within the 'degrees of compromise' that exist in the real world. In so doing, it might offer insights into post-compromise defensive techniques to isolate, mitigate and manage the effect of hostile actors already operating within a network or system. "Extended Experimentation" section introduces how a label-property graph could be used to improve overall representation of the systems under review. The relationship (SRO) data has proven vital in both the v9- and v11-based experiments, but a logical extension of the technique would be model the entire CTI data as a graph whose vertices are SDOs and edges SROs. Properties can then be applied (to both vertices and edges) to further instrument LeWiS and increase the technique's expressive power. In many respects, this is a generalisation of the SOP model introduced herein, but one in which all relationships are mapped between each and every vertex in the graph. This may also create opportunities to explore the application of graph traversal algorithms to the classification process.

Perhaps the most compelling extension to the LeWiS method is generalising it to predict other target variables. This would entail training models whose classification targets are not only intrusion sets, but broaden to include Infrastructure, Malware and Vulnerability SDOS. Infrastructure estimation is an example of finding new TTPs through generalisation of known data. Similarly for Malware SDOs, telemetry would be used to estimate the presence of a particular implant within a system (rather than whom may be responsible for it). This may offer opportunities for detection outside of more more conventional means (such as intrusion detection and endpoint security technologies). Estimating Vulnerability SDOs could establish the presence of a specific vulnerability, or set of vulnerabilities within the monitored infrastructure purely thought intelligence processing.

Finally, STIX does not include a mechanism to state how common is any particular relationship between SDOs. This could be of real significance in machine learning and be used to improve the resolution of the models; avoiding the need to train only on binary relationships (i.e. one exists, or it does not) and allowing a more comprehensive scheme to be defined that uses the degree to which a relationship is present. The re-sampling technique described herein provides a partial solution to the problem, but greater improvements could likely be made by adding 'strengths' to the underlying data model. This is of course, not a trivial undertaking and it is necessary to remember that this additional attribute would require greater empirical information that might otherwise be used to construct attack sets. Furthermore, one has to

assume imperfect knowledge of the TTPs for any threat actor and because the 'strength' attribute is a function of other observable data, it may be difficult to manage bias when working in real-world settings.

Appendix: Additional Tabular Data

See Tables 7 and 8.

Table 7 SOP and vector sizes by MITRE ATT &CK group

Group	SOP	AP	T	M
APT29	152	106	14	32
WIRTE	14	11	1	2
Axiom	24	16	0	8
Aquatic Panda	17	15	0	2
Threat Group-3390	80	56	12	12
Gamaredon Group	51	47	1	3
APT28	113	87	9	17
Lazarus Group	138	112	4	22
Kimsuky	99	89	4	6
Ke3chang	56	45	8	3
Mustang Panda	48	43	1	4
Magic Hound	49	44	3	2
Confucius	20	19	0	1
LazyScripter	27	20	4	3
BlackTech	20	14	1	5
Indrik Spider	27	19	4	4
Sandworm Team	86	71	4	11
Volatile Cedar	7	5	0	2
TeamTNT	48	44	3	1
FIN7	57	40	4	13
APT38	49	43	2	4
Winnti Group	9	6	0	3
Dragonfly	67	57	8	2
Turla	91	65	13	13
Tonto Team	21	15	4	2
Cobalt Group	40	34	3	3
Transparent Tribe	19	14	0	5
CostaRicto	11	5	3	3
TA505	34	25	1	8
Nomadic Octopus	8	7	0	1
OilRig	79	58	11	10
Night Dragon	21	16	3	2
DarkVishnya	12	10	2	0
FIN5	17	11	4	2
Gorgon Group	21	17	2	2
Patchwork	47	39	2	6
Chimera	65	59	5	1
Blue Mockingbird	23	22	1	0
Whitefly	10	9	1	0
APT41	83	59	11	13
FIN6	52	40	4	8
TEMP.Veles	22	19	2	1
PittyTiger	7	2	2	3
Thrip	7	4	2	1
DarkHydrus	10	7	1	2
APT32	92	78	5	9
BRONZE BUTLER	54	40	7	7
Carbanak	13	9	3	1
Cleaver	9	5	2	2
Inception	25	22	1	2
Leafminer	21	17	4	0
Ferocious Kitten	8	6	1	1
IndigoZebra	10	7	0	3
BackdoorDiplomacy	20	15	3	2
FIN8	37	31	4	2
APT37	44	31	0	13
TA551	18	14	0	4
Andariel	14	12	0	2
Leviathan	61	44	7	10
Naikon	29	14	7	8
Wizard Spider	62	46	8	8
menuPass	71	47	12	12
APT-C-36	10	9	1	0
Frankenstein	27	26	1	0
Silence	31	28	3	0
APT33	47	32	9	6
APT19	23	21	1	1
FIN10	12	11	1	0
CopyKittens	11	7	1	3
APT39	63	52	7	4
APT1	40	23	11	6
APT3	51	45	2	4
MuddyWater	65	54	9	2
Darkhotel	25	25	0	0
Fox Kitten	44	40	1	3
ZIRCONIUM	25	25	0	0
Silent Librarian	13	13	0	0
HAFNIUM	24	21	1	2
Windshift	20	19	0	1
Ajax Security Team	8	6	2	0
Sidewinder	30	29	1	0
Higaisa	29	26	1	2
GALLIUM	46	31	11	4
Tropic Trooper	45	39	1	5

Table 7 (continued)

Group	SOP	AP	T	M
UNC2452	63	54	2	7
GOLD SOUTHFIELD	11	9	1	1
Machete	13	12	0	1
Windigo	8	7	0	1
Evilnum	14	11	1	2
Molerats	22	16	0	6
Operation Wocao	66	60	6	0
Stealth Falcon	16	16	0	0
PROMETHIUM	13	11	0	2
APT17	3	2	0	1
APT16	2	1	0	1
APT30	7	2	0	5
Strider	4	3	0	1
Equation	5	5	0	0
Rocke	36	36	0	0
SilverTerrier	8	3	0	5
Sharpshooter	9	8	0	1
Mofang	7	5	1	1
RTM	8	7	0	1
APT12	8	5	1	2
Honeybee	25	21	4	0
Poseidon Group	8	8	0	0
Rancor	12	8	2	2
GCMAN	2	2	0	0
FIN4	12	12	0	0
Elderwood	18	9	0	9
APT18	18	13	1	4
Deep Panda	17	10	3	4
admin@338	19	12	4	3
Stolen Pencil	11	9	2	0
Group5	7	5	0	2
The White Company	9	7	0	2
PLATINUM	15	12	0	3
Scarlet Mimic	5	1	0	4
Gallmaker	6	6	0	0
Dark Caracal	15	12	0	3
Orangeworm	10	2	7	1
TA459	9	5	0	4
BlackOasis	1	1	0	0
Sowbug	11	9	0	2
NEODYMIUM	1	0	0	1
Suckfly	6	5	0	1
Dust Storm	7	3	0	4
Threat Group-1314	6	4	2	0
Lotus Blossom	2	0	0	2
PutterPanda	8	4	0	4
DragonOK	2	0	0	2
Taidoor	1	1	0	0
Moafee	2	1	0	1

Table 8 Summarised classification performance by MITRE ATT &CK group

Group	SVM	DTC	Logit
APT29	0.879	0.865	0.858
WIRTE	0.065	0.049	0.036
Axiom	0.037	0.061	0.066
Aquatic Panda	0.045	0.035	0.063
Threat Group-3390	0.447	0.438	0.443
Gamaredon Group	0.529	0.589	0.636
APT28	0.601	0.619	0.610
Lazarus Group	0.796	0.794	0.796
Kimsuky	0.523	0.553	0.551
Ke3chang	0.625	0.662	0.561
Mustang Panda	0.546	0.534	0.582
Magic Hound	0.532	0.501	0.608
Confucius	0.034	0.020	0.038
LazyScripter	0.109	0.029	0.048
BlackTech	0.038	0.082	0.079
Indrik Spider	0.060	0.079	0.087
Sandworm Team	0.458	0.473	0.463
Volatile Cedar	0.026	0.024	0.009
TeamTNT	0.546	0.565	0.588
FIN7	0.610	0.621	0.708
APT38	0.547	0.574	0.608
Winnti Group	0.026	0.029	0.021
Dragonfly	0.751	0.737	0.792
Turla	0.481	0.507	0.491
Tonto Team	0.027	0.077	0.100
Cobalt Group	0.438	0.458	0.419
Transparent Tribe	0.078	0.014	0.072
CostaRicto	0.055	0.058	0.013
TA505	0.071	0.120	0.049
Nomadic Octopus	0.040	0.021	0.034
OilRig	0.426	0.435	0.417
Night Dragon	0.049	0.025	0.103
DarkVishnya	0.032	0.010	0.054
FIN5	0.050	0.083	0.013
Gorgon Group	0.070	0.081	0.109
Patchwork	0.466	0.491	0.473
Chimera	0.693	0.742	0.665
Blue Mockingbird	0.111	0.119	0.090
Whitefly	0.046	0.036	0.014
APT41	0.438	0.461	0.437
FIN6	0.516	0.553	0.600
TEMP.Veles	0.100	0.103	0.037
PittyTiger	0.023	0.030	0.036
Thrip	0.007	0.017	0.036
DarkHydrus	0.019	0.041	0.024
APT32	0.507	0.509	0.486
BRONZE BUTLER	0.566	0.643	0.597
Carbanak	0.055	0.059	0.038
Cleaver	0.024	0.032	0.037

Table 8 (continued)

Group	SVM	DTC	Logit
Inception	0.130	0.080	0.065
Leafminer	0.054	0.092	0.081
Ferocious Kitten	0.031	0.041	0.016
IndigoZebra	0.038	0.029	0.035
BackdoorDiplomacy	0.069	0.082	0.018
FIN8	0.167	0.054	0.110
APT37	0.457	0.487	0.474
TA551	0.054	0.049	0.058
Andariel	0.047	0.050	0.053
Leviathan	0.602	0.621	0.619
Naikon	0.123	0.028	0.080
Wizard Spider	0.686	0.707	0.616
menuPass	0.796	0.751	0.856
APT-C-36	0.045	0.025	0.021
Frankenstein	0.079	0.048	0.034
Silence	0.079	0.126	0.099
APT33	0.543	0.539	0.537
APT19	0.091	0.018	0.029
FIN10	0.032	0.019	0.044
CopyKittens	0.053	0.030	0.032
APT39	0.629	0.649	0.763
APT1	0.417	0.409	0.451
APT3	0.578	0.601	0.637
MuddyWater	0.645	0.771	0.658
Darkhotel	0.090	0.113	0.120
Fox Kitten	0.519	0.491	0.518
ZIRCONIUM	0.114	0.116	0.109
Silent Librarian	0.014	0.020	0.063
HAFNIUM	0.072	0.118	0.121
Windshift	0.035	0.082	0.083
Ajax Security Team	0.012	0.020	0.013
Sidewinder	0.053	0.068	0.103
Higaisa	0.143	0.065	0.072
GALLIUM	0.512	0.557	0.521
Tropic Trooper	0.503	0.448	0.519
UNC2452	0.681	0.750	0.652
GOLD SOUTHFIELD	0.010	0.035	0.011
Machete	0.013	0.010	0.056
Windigo	0.006	0.023	0.037
Evilnum	0.041	0.015	0.039
Molerats	0.040	0.057	0.079
Operation Wocao	0.713	0.751	0.788
Stealth Falcon	0.038	0.050	0.071
PROMETHIUM	0.036	0.055	0.059
APT17	0.011	0.006	0.005
APT16	0.007	0.004	0.006
APT30	0.035	0.007	0.019
Strider	0.009	0.009	0.021
Equation	0.018	0.023	0.013
Rocke	0.098	0.082	0.063

Table 8 (continued)

Group	SVM	DTC	Logit
SilverTerrier	0.031	0.041	0.019
Sharpshooter	0.026	0.013	0.039
Mofang	0.024	0.036	0.027
RTM	0.035	0.007	0.036
APT12	0.011	0.036	0.010
Honeybee	0.083	0.027	0.039
Poseidon Group	0.030	0.027	0.015
Rancor	0.047	0.037	0.058
GCMAN	0.010	0.010	0.010
FIN4	0.047	0.030	0.052
Elderwood	0.045	0.059	0.082
APT18	0.066	0.063	0.058
Deep Panda	0.069	0.030	0.028
admin@338	0.090	0.027	0.090
Stolen Pencil	0.035	0.045	0.042
Group5	0.032	0.014	0.036
The White Company	0.033	0.008	0.047
PLATINUM	0.040	0.065	0.077
Scarlet Mimic	0.014	0.013	0.011
Gallmaker	0.008	0.014	0.019
Dark Caracal	0.020	0.057	0.023
Orangeworm	0.011	0.013	0.024
TA459	0.034	0.007	0.018
BlackOasis	0.001	0.004	0.002
Sowbug	0.038	0.040	0.033
NEODYMIUM	0.004	0.001	0.002
Suckfly	0.011	0.015	0.007
Dust Storm	0.015	0.017	0.028
Threat Group-1314	0.026	0.006	0.012
Lotus Blossom	0.009	0.006	0.006
Putter Panda	0.035	0.040	0.013
DragonOK	0.010	0.003	0.008
Taidoor	0.004	0.003	0.004
Moafee	0.002	0.004	0.008

Acknowledgements This work is an extended version of a paper bearing the same name by the same authors, originally presented at the International Conference on Web Information Systems and Technologies (WEBIST) 2021. This longer-form presentation provides additional discussion of the LeWiS method, together with the results of further experimentation and development efforts (including the techniques being applied to a later version of the MITRE ATT &CK data, as documented herein).

Funding The authors declare that this work was self-funded and did not include any form of funding from an external party or body.

Data availability STIX versions of the MITRE ATT &CK data can be generated, as can the SOP objects described in the technique (which are themselves generated as a function of the MITRE data).

Declarations

Conflict of Interest The authors declare that they have no conflicts of interest.

References

1. Alghamdi MI. Survey on applications of deep learning and machine learning techniques for cyber security. Int J Interact Mob Technol. 2020;14:210–24.
2. Amit I, Matherly J, Hewlett W, Xu Z, Meshi Y, Weinberger Y. Machine learning in cyber-security - problems, challenges and data sets; 2018.
3. BBC. Cyber attack 'most significant on irish state'; 2021. https://www.bbc.co.uk/news/world-europe-57111615
4. Bellman R. Dynamic programming. Princeton: Princeton University Press; 1957.
5. Ben-Asher N, Hutchinson S, Oltramari A. Characterizing network behavior features using a cyber-security ontology. In: MILCOM 2016—2016 IEEE military communications conference, military communications conference; 2016. pp. 758 – 763.
6. Blackwell C. A security ontology for incident analysis. In: CSIIRW 10; 2010.
7. Darktrace. Darktrace industrial uses machine learning to identify cyber campaigns targeting critical infrastructure. https://www.darktrace.com/en/press/2017/204/ 2017
8. Das R, Morris TH. Machine learning and cyber security. In: 2017 International conference on computer, electrical and communication engineering (ICCECE); 2017. pp. 1 – 7.
9. DCMS. Cyber security breaches survey 2021. Technical report, UK Government; 2021. https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2021/cyber-security-breaches-survey-2021
10. Elmrabit N, Zhou F, Li F, Zhou H. Evaluation of machine learning algorithms for anomaly detection. In: 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security); 2020. pp. 1– 8.
11. Fransen F, Kerkdijk R, Smulders A. Cyber security information exchange to gain insight into the effects of cyber threats and incidents. Elektrotechnik und Informationstechnik. 2015;132(2):106–12.
12. Gupta B, Sheng M. Machine learning for computer and cyber security : principles, algorithms, and practices. Boca Raton: CRC Press; 2019.
13. Johnson C, B. L. W. D. S. J. S. C. NIST Special Publication 800-150: Guide to Cyber Threat Information Sharing. National Institute for Standards and Technology; 2019.
14. Jungsoo P, Long Nguyen V, Bencivengo G, Souhwan J. Automatic generation of MAEC and STIX standards for android malware threat intelligence. KSII Trans Internet Inf Syst. 2020;14(8):3420–36.
15. Kaloudi N, Jingyue L. The AI-based cyber threat landscape: a survey. ACM Comput Surv. 2020;53(1):1–34.
16. Kumar SR, Yadav SA, Sharma S, Singh A. Recommendations for effective cyber security execution. In: 2016 International conference on innovation and challenges in cyber security (ICICCS-INBUSH), 2016; pp. 342–346.
17. MITRE. Mitre att &ck; 2021. https://cti-taxii.mitre.org/stix/collections/95ecc380-afe9-11e4-9b6c-751b66dd541e/objects/
18. MITRE. Mitre att &ck; 2022. https://cti-taxii.mitre.org/stix/collections/95ecc380-afe9-11e4-9b6c-751b66dd541e/objects/
19. Mittal S, J. A. F. T. Thinking, fast and slow: Combining vector spaces and knowledge graphs. CoRR; 2017. arXiv:abs/1708.03310
20. Mugan J. A developmental approach to learning causal models for cyber security. In: Proceedings of SPIE - The International Society for Optical Engineering, 2013, pp. 87510A.
21. NCSC. Cyber information sharing partnership; 2021. https://www.ncsc.gov.uk/section/keep-up-to-date/cisp
22. OASIS. Stix 2.1 bundle specification;2021a. https://docs.oasis-open.org/cti/stix/v2.1/cs02/stix-v2.1-cs02.html#_gms872kuzdmg
23. OASIS. STIX Version 2.1;2021b. https://docs.oasis-open.org/cti/stix/v2.1/cs02/stix-v2.1-cs02.html
24. PaloAlto. Expanse - attack surface reduction; 2021. https://expanse.co/attack-surface-reduction/
25. Riesco R, Larriva-Novo X, Villagra VA. Cybersecurity threat intelligence knowledge exchange based on blockchain: proposal of a new incentive model based on blockchain and smart contracts to foster the cyber threat and risk intelligence exchange of information. Telecommun Syst. 2020;73(2):259–88.
26. Riesco R, Villagrá VA. Leveraging cyber threat intelligence for a dynamic risk framework: automation by using a semantic reasoner and a new combination of standards (stix™, swrl and owl). Int J Inf Secur. 2019;18(6):715–39.
27. Scheau M, Arsene A-L, Popescu G. Artificial intelligence/machine learning challenges and evolution. Int J Inf Secur Cybercrime. 2018;7:11–22.
28. Shaukat K, Luo S, Varadharajan V, Hameed I, Xu M. A survey on machine learning techniques for cyber security in the last decade. IEEE Access. 2020;8:222310–54.
29. Smart W. Lessons learned review of the wannacry ransomware cyber attack. In: Technical report, department for health and social care; 2018. https://www.england.nhs.uk/wp-content/uploads/2018/02/lessons-learned-review-wannacry-ransomware-cyber-attack-cio-review.pdf
30. Vectra.ai. Vectra.ai - how we do it; 2021. https://www.vectra.ai/products/how-we-do-it
31. Wali A, Soon Ae C, Geller J. A bootstrapping approach for developing a cyber-security ontology using textbook index terms. In: 2013 International conference on availability, reliability and security, availability, reliability and security (ARES); 2013. pp. 569 – 576.
32. Xu J, Wen Y, Yang C, Meng D. An approach for poisoning attacks against rnn-based cyber anomaly detection. In: 2020 IEEE 19th international conference on trust, security and privacy in computing and communications (TrustCom); 2020. pp. 1680 – 1687.
33. Zheng H, Wang Y, Han C, Le F, He R, Lu J. Learning and applying ontology for machine learning in cyber attack detection. In: 2018 17th IEEE international conference on trust, security and privacy in computing and communications/ 12th IEEE international conference on big data science and engineering (TrustCom/BigDataSE), trust, security and privacy in computing and communications; 2018. pp. 1309 – 1315.
34. Zhou L, Shu J, Jia X. Collaborative anomaly detection in distributed sdn. IN: GLOBECOM 2020 - 2020 IEEE Global Communications Conference; 2020. pp. 1 – 6.
35. Zhou Y, Zhu C, Tang L, Zhang W, Wang P. Cyber security inference based on a two-level Bayesian network framework. In: 2018 IEEE international conference on systems, man, and cybernetics (SMC); 2018. pp. 3932 – 3937.