PREPRINT: Taneeya Satyapanich, Tim Finin and Francis Ferraro, CASIE: Extracting Cybersecurity Event Information from Text, 34th AAAI Confference on Artificial Intelligence, New York, Feb. 2020.

CASIE: Extracting Cybersecurity Event Information from Text

Taneeya Satyapanich, Francis Ferraro, Tim Finin
Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, MD 21250 USA
{taneeya1,ferraro,finin}@umbc.edu

Abstract

We present CASIE, a system that extracts information about cybersecurity events from text and populates a semantic model, with the ultimate goal of integration into a knowledge graph of cybersecurity data. It was trained on a new corpus of 1,000 English news articles from 2017–2019 that are labeled with rich, event-based annotations and that covers both cyberattack and vulnerability-related events. Our model defines five event subtypes along with their semantic roles and 20 event-relevant argument types (e.g., file, device, software, money). CASIE uses different deep neural networks approaches with attention and can incorporate rich linguistic features and word embeddings. We have conducted experiments on each component in the event detection pipeline and the results show that each subsystem performs well.

Introduction

Cyberattacks and cybercrimes are common today and their frequency and severity are only expected to increase. They are also evolving to exploit new vulnerabilities and environments, such as the *Internet of Things* and *cyber-physical systems*. People and machines will be better able to defend against attacks if we can keep abreast of current trends and vulnerabilities. We have developed CASIE (CyberAttack Sensing and Information Extraction) as a tool to help do this by extracting information about cybersecurity events from news articles. The results can be used to keep people informed and also integrated into knowledge graphs of cybersecurity data to help automated systems.

The task of identifying instances of specific events in text and extracting relevant information from them has been studied for a long time but remains a challenging task. The task was introduced in the second Message Understanding Conference in 1989 and has been part of many information extraction challenges since; see Grishman and Sundheim (1996) for a history of MUC. Efforts to improve event detection performance have tended to focus on adding new features or improving pattern matching algorithms (Ji and Grishman 2008; Li, Ji, and Huang 2013), or the development of neural networks to better capture information, such as dependency tree-based CNNs (Nguyen and Grishman 2018). To overcome the data scarcity of labeled data, multiple sources of external knowledge, such as semantic frame analyses (Liu et al. 2016; Li et al. 2019; Chen et al. 2017), consistent and complement information to disambiguation from multilingual data (Liu et al. 2018), and expert-level patterns from an open-source pattern-based event extraction system called TABARI (Cao et al. 2018), have been leveraged.

Most previous event detection research has focused on common events in a person's life, such as those defined by ACE (Walker et al. 2006) or the TAC Knowledge Base Population (Mitamura, Liu, and Hovy 2015). These life events include things like "being born," "getting married," "starting a job' and "being charged with a crime." One core difference between extracting life events and cybersecurity events is the required domain-relevant expertise. We share this difficulty with other types of domain-specific information extraction, such as extracting information about biomedical events. The development of these rich event extraction datasets and tasks has helped to spur innovation in the BioNLP area (Kim et al. 2009). One of our contributions is the analogous creation for the cybersecurity domain.

A second difference between extracting life events and cybersecurity events is the inherent complexity of cybersecurity events. A cyberattack event can consist of an *attack pattern* with multiple actions, attempted or completed. Each mention of one of these actions can be considered as a separate cybersecurity event description, which multiplies the possible choices for a cybersecurity event reference. Comparing to life events, the challenge is among the homonym and synonym sets of an event mention. Our work makes three main contributions.

- We define and specify five cybersecurity events along with their semantic roles and 20 types of arguments that are candidates for role-fillers.
- We present a novel, challenging corpus of newswire articles annotated with the cybersecurity events in them. These annotations will be available upon publication.
- We present CASIE, a neural cybersecurity event extraction system that combines modern deep learning with lin-

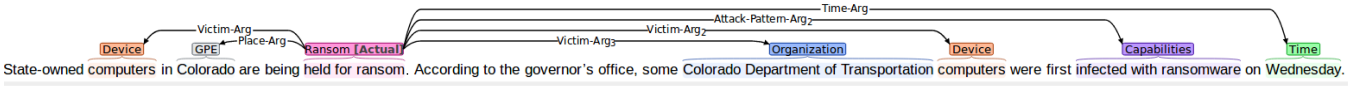

Figure 1: This example shows text describing an **Attack.Ransom** event triggered by the event nugget phrase "held for ransom" in the first sentence. Also annotated are entities (e.g., "computers," 'infected with ransomware"), their types ("Device," "Capabilities") and roles (the labeled edges, e.g, "Victim:Arg" connecting the event nugget to the "computers" entity). Notice that event annotations span sentences.

guistic features, providing a suite of competitive information extraction models and tools for developing cybersecurity features from background knowledge graphs. We will also make the code available upon publication.

While there has been previous work in cyberattack event analysis (Qiu, Lin, and Qiu 2016; Khandpur et al. 2017), to the best of our knowledge, our research covers the largest range and complexity of cybersecurity events. We first provide the cybersecurity event definitions and common terminology used in our event detection task. Then we present CASIE's overall architecture along with each of its components, and provide details of our corpus and annotations. We then discuss our evaluation and experimental results and outline ongoing and future work.

Cybersecurity Event Extraction

We borrow common event detection terminology from TAC, including the following.

- An **event nugget** is a word or phrase that most clearly expresses the event occurrence. These differ from event triggers in that they can be multi-word phrases.
- An **event argument** is an event participant or property value. They can be taggable entities involved in the event, such as person or organization, or attributes that specify important information, such as time or amount.
- A **role** is a semantic relation between an event nugget and an argument. Each event type specifies the roles it can have and constraints on the arguments that can fill them.
- A **realis** value specifies whether or not an event occurred and can be one of the three values: *Actual* (event actually happened), *Other* (failed event, future event), or *Generic* (an undetermined/non-specific event, such as referring to the concept of phishing attacks).

To develop our approach, we chose an initial set of frequently occurring, high impact, and distinct events from cybersecurity news that involve cyberattacks or vulnerability-related events. The basic roles and arguments that can fill them of each event type are defined in Table 1. We further subcategorized these two event types into five event subtypes: Attack.Databreach, Attack.Phishing, Attack.Ransom, Discover.Vulnerability, and Patch.Vulnerability. Each event subtype comes with specific roles in addition to the basic roles. Full details can be found in (Satyapanich 2019b). Specifically we define the following events:

- **Attack.Databreach**, an attacker compromises a system and removes data, e.g., to sell or publish it. The specific roles for the Attack.Databreach are Compromised-Data and Number-Of-Data.
- **Attack.Phishing**, an attacker imitates another entity, in an attempt to get a victim to access malicious materials, such as a website or attachments. The additional role for the Attack.Phishing is Trusted-Entity.
- **Attack.Ransom**, an attacker breaks into a system and encrypts data, and will only decrypt the data for a ransom payment. The specific role for the Attack.Ransom are Ransom-Price and Payment-Method.
- **Discover.Vulnerability**, a security expert or other entity, like a company, finds a software vulnerability. The additional roles of Discover.Vunerability consist of Discoverer, Capabilities, and VS-Owner.
- **Patch.Vulnerability**, a software company addresses a known vulnerability by releasing or describing an appropriate update. The extra roles of Patch.Vulnerability are Releaser, Issue-Addressed, Patch, Patch-Number, and Supported-Platform.

An example of an event is shown in Figure 1. The event nugget and arguments are shown in shaded text, with the types indicated by the colored labels above the text and roles indicated by the labeled arcs. The phrase '*held for ransom*' is identified as an event nugget for an **Attack.Ransom** event. Event arguments consist of *'computers'* which is a type of Device and fills the Victim role; *'Colorado'* which is a GPE and fills the Place role; *'Colorado Department of Transportation'* which is an Organization filling the Victim role; and *'infected with ransomware'* which has type Capabilities filling the Attack Pattern role; and *'Wednesday'* which is a Time expression that fills the Time role.

CASIE's Design and Architecture

CASIE's architecture, shown in Figure 2, includes six steps: event nugget detection, event argument detection, event argument and role linking, event realis identification, event coreference, and mapping the results to a knowledge graph. In this targeted work, we focus on the first four steps, formalizing each as a multi-class classification task, leaving the complex tasks of event coreference and full knowledge graph integration to future work.

Event Nugget and Event Argument Detection

Both the event nugget detection system and event argument detection system employ a hybrid Bidirectional LSTM neural network. Each event nugget classifier and event argument classifier uses its own features, which include both

Types	Roles	Arguments
Cyber Attack	Attacker	Organization; Person
	Victim	Device; Organization; Person; Product; Website
	Attack-Pattern	Capabilities
	Tool	File; Malware; Website
	Damage-Amount	Money
	Number-of-Victim	Number
	Purpose	Purpose
	Place	GPE
	Time	Time
Vulnerability-related	CVE	CVE
	Vulnerability	Vulnerability
	VS	Device; Product; Website
	VS-Version	Version
	Time	Time

Table 1: Basic roles and arguments that can filled them for each event types. VS means Vulnerable-System.

linguistically-inspired features and a continuous word embedding for every word in an input sentence. We use a BIO scheme to tag each token.

Event nugget features. The feature extraction starts by applying CoreNLP (Manning et al. 2014) to the raw text for tokenization, lemmatization, part of speech tagging, and named entity recognition. After that all stopwords were removed. We also introduced two external knowledge bases—DBpedia Spotlight (Mendes et al. 2011) and Wikidata (Vrandečić and Krötzsch 2014)—to find additional named entities, such as instances of software and malware. We used these external sources to overcome the lack of appropriate types in CoreNLP. Using these resources followed by our post-processing, we create the following linguistic features sets.

1. Part of speech labels for each word (e.g., NN, VBD, etc.);
2. Entity types from CoreNLP (e.g., Person, Organization, etc.), DBpedia (e.g., Network, System, etc.);
3. Relevant Wikidata classes for an entity (e.g., software company, computer system, etc.);
4. The set of syntactic dependency relations extracted from the sentences (e.g., nsubj, dobj, etc.); and
5. Shallow syntactic chunking type (e.g., S, NP, VP, etc.) and its depth (the highest level in a constituency parse tree) for every word.

Event argument features. We combine the following features with the feature (1)-(4) from the event nugget features

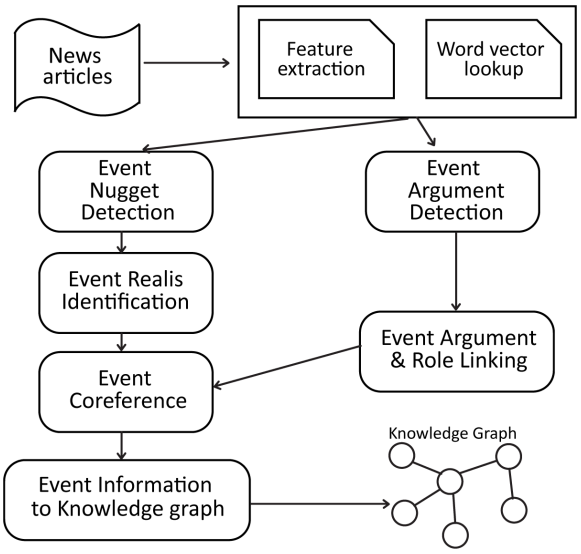

Figure 2: CASIE can populate a knowledge graph with data about cybersecurity events extracted from text.

defined above. For each word in a sentence we record:

6. the shallow syntactic chunking type (e.g., S, NP, VP, etc.) and its depth, selecting the lowest level in parse tree;
7. the type of the nearest event (i.e., our five event types);
8. the distance in the dependency tree (number of relation hops) to the nearest event nugget head;
9. the relative position with the nearest event nugget head (i.e., after-same-sentence, before-same-sentence, after-differ-sentence, before-differ-sentence);
10. the dependency parse path to the nearest event nugget head (e.g., conj-nmod-nsubj, etc.); and
11. the common constituency parse tree node with the nearest event nugget head (e.g., NP, PP, VP, etc.).

The nearest event nugget is from comparing the distance between the target token to every event nugget in the dependency tree. The event nugget head is the word with the highest level in the dependency tree compared to the other words in the same nugget. We find additional named entities by linking nominal mentions to the nearby named entity of the same type. For example, the word 'company' is labeled as *Organization* if we can find a nearby *Organization* named entity.

Word Embeddings. There are currently many word embeddings techniques that are built from different sized collections and genres. We experimented with four different embedding types: Transfer-Word2vec, Domain-Word2vec, Cyber-Word2vec, and Pre-built BERT. We consider these four approaches to be either context-free (e.g., Word2vec (Mikolov et al. 2013)) or context-dependent embeddings. For the context-free embeddings, we trained with two different Word2vec embeddings: Transfer-Word2vec and Domain-Word2vec.

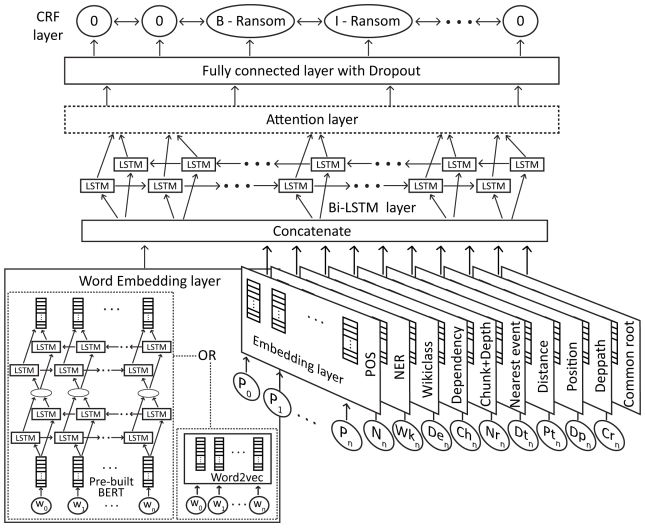

Figure 3: A bidirectional LSTM network is used to classify event nugget and event argument types. The attention layer is applied only to the event argument detection system. The word embedding layer is either a Pre-Built BERT network or the Word2vec embedding.

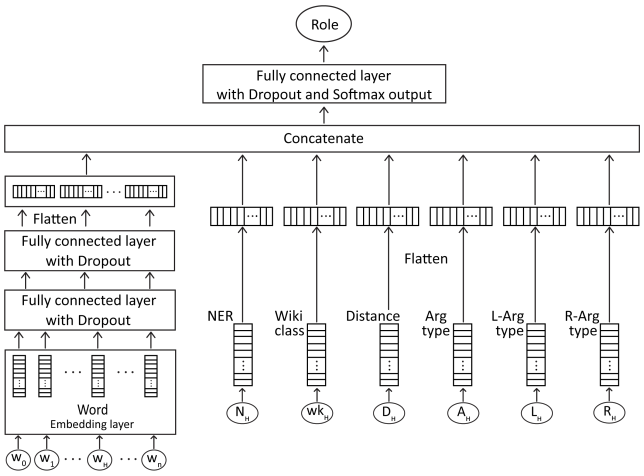

Figure 4: Our neural network for assigning roles to arguments.

The non-contextual embedding types were trained on a collection of 5,000 cybersecurity news articles, and differed in their initialization and dimensions. The 5,000 articles contain 2,531,577 tokens (separated by spaces). Transfer-Word2vec was initialized with the publicly available 300-dimension Google-News-vectors-negative300 model (Mikolov et al. 2013). After training, 14,960 terms were added. Domain-Word2vec used 100-dimension randomly initialized vectors and required words to occur at least twice in the corpus. The vocabulary size of the Domain-word2vec embedding was 28,283 words. The Cyber-Word2vec embeddings were produced by Padia et al. (2018). It has 100-dimension which trained on a large corpus of approximately one million cybersecurity-related webpages. The Cyber-Word2vec contains 6,417,554 vocabularies.

For the context dependent embeddings, we used the pre-trained "BERT-Base Uncased" model (Devlin et al. 2018), doing fine-tuning in our neural networks. This model is a 12-layer, 12-attention head network that produces a context-dependent 768 dimension embedding for each word *token*. Because BERT is context-dependent, the same word in different sentences produces different vectors. BERT used WordPiece tokenization, breaking word into pieces and generates embeddings for them. We computed the average of embeddings for a word's pieces as its embedding; we call this *Pre-built BERT*. We kept all of the word embeddings as input and experimentally found that using the fourth-to-last hidden layer gave the best development performance.

Approach. We used a Bi-Directional LSTM network with CRF loss to classify event nugget types and event argument types. This architecture was selected both because it can use information from both the left context and right context in predicting a target word, and because full sequence decoding can be done. We use five types of layers: embedding layers, a Bi-LSTM layer, an attention layer, a fully-connected layer, and a CRF layer. The number of nodes of every layer is equal to a half of the number in the previous layer, except for the attention layer (where the output and input sizes are equal) and the CRF layer (where the output size is equal to the number of output classes). The network architecture is shown in Figure 3.

We form embedding layers by concatenating embedding layers of each linguistic feature, including the word embedding layer. We treat each linguistic feature as a categorical variable. Each category of a feature is used as the index into a table containing randomly initialized weight vectors for all categories. Each feature embedding layer's output size is equal to half of the total number of its categories. For the word embedding layer, unique word types are categories and we select initial weights using the word vector obtained from each embedding (Transfer-Word2vec, Domain-Word2vec, Cyber-Word2vec). The embedding layers are then concatenated to form the first layers of the Bi-LSTM-CRF networks. When we use BERT embeddings, we add two additional Bi-LSTM layers before concatenating them with the feature embedding layers. The first layer has input from Pre-built BERT. The size of each Bi-LSTM additional layer is equal to half its input size. Concatenating the embedding layers yields a single Bi-LSTM hidden layer.

Attention mechanisms have been used with great success in neural machine translation, where they learn to highlight important parts of the text. We applied a location-based attention mechanism (Zheng et al. 2018; Luong, Pham, and Manning 2015) as an attention layer following the Bi-LSTM layer. Early experiments showed that an attention size of five gave the best performance, which was further improved by using a $\tanh$ activation function. We found that adding an attention layer improved event argument detection but it did not improve event nugget detection. After the attention layer, we have a dense layer. Finally, we applied a CRF to predict the output sequence labels.

Event Argument and Role Linking

A role will be assigned to an event argument that is consistent with the type constraints defined by our event frames. These requirements are listed in Table 1. For example, if a mention of a *Person* argument was found in the context of an **Attack.Phishing** event, its possible role could be *Attacker*, *Victim*, or *Trusted-Entity*.

Features. The following features are computed after the event nuggets and event arguments are predicted.

- Word vectors of the argument surface words
- Features (2), (3), and (8) from the event nugget and event argument features defined above;
- The target event argument type and the types of the immediate left and right event argument of the target event argument.

Approach. We have developed a fully-connected neural network to assign a role to an argument. The neural networks consist of an embedding layer and three fully-connected layers. The embedding layer was built for each feature (including word embeddings). The word embedding layer was passed through two fully-connected layers and then concatenated with the embedding layers of the other features. The last layer is another fully-connected layer with the number of output node specified by the event type. The neural network architecture is shown in Figure 4. Each layer's dimension is half of the previous layer's dimension.

The neural networks will be used to predict the arguments which can be filled by more than one role, otherwise, no prediction is needed. We built a neural network for each event type. The role that links an event argument to an event nugget is determined by its event type. Since the event type is known before predicting the argument role, it can be used to eliminate irrelevant roles. The number of output layer nodes is reduced to the number of roles of each event instead of the total number of roles in the system. For example, in the **Patch.Vulnerability** event, the *Attacker* and *Victim* are not included in the output classes. We found that this improved performance significantly. The performances of two system settings: specific and non-specific event type, are compared in the Table 7.

Event Realis Identification

We identified the realis value for event nuggets. Each event nugget is annotated with a realis value to give information whether an instance of an event actually happened (Actual), did not occur (Other), or the event is referenced as a concept (Generic). Examples of events with the three possible realis values are:

- Actual: Facebook have admitted they ***were conned***$_{Actual}$ out of an alleged $100 million in ***a phishing scam***$_{Actual}$.
- Other: Apple ***has not yet clarified***$_{Other}$ whether the latest versions of macOS and iOS ***are vulnerable***$_{Other}$.
- Generic: ***Phishing***$_{Generic}$ has grown to be one of the most serious threats to healthcare organizations.

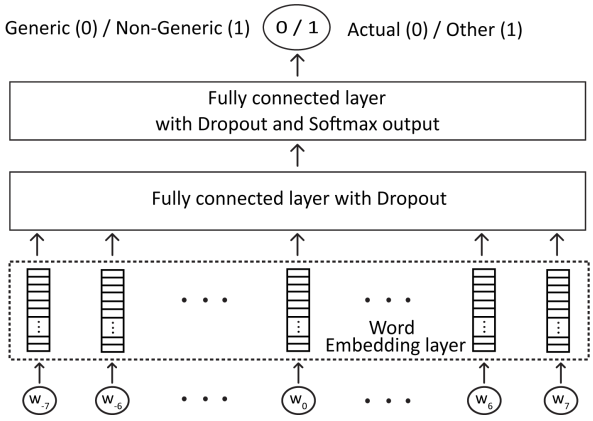

Figure 5: Our network for event realis property classification. We use this for predicting both steps with different output tags, i.e., Generic vs. Non-generic, and Actual vs. Other.

Event realis identification features. When the event nugget was found, the feature vector for realis is the concatenation of the word vectors of the event nugget and its surrounding lexical context. We found through initial development that a context window of seven words resulted in the best performance. All stopwords are included in the realis identification component as they contain important information. For example, modal verbs (may, can) and negatives (not, no) are evidence of a failed event.

Approach. Realis identification has two steps: (i) Classify the event nugget as Generic or Non-generic; and (ii) if Non-generic, further classify as Actual or Other. Both classifiers are binary and use the same neural network architecture (Figure 5) with three layers: an embedding layer and two fully-connected layers. The embedding layer is the size of word vector, the numbers of nodes of the fully-connected layers are half the size of their previous layers.

Related Work on Cybersecurity Event Extraction

Related work falls into two categories: event definitions and its data collection, and methods for event detection. We draw inspiration from ACE 2005 (Walker et al. 2006), a widely-used set of event definitions annotated on 599 newswire documents, and the Rich ERE-based TAC KBP event track. Whereas 70% of ACE's event types have fewer than 100 instances, all of ours have more than 1,000. Lim et al. (2017) built an annotated malware database based on MAEC vocabularies from 39 APT reports but that effort did not annotate rich event annotations as done here. While there are a number of research, community, and commercial efforts to create repositories of reports and timelines of cybersecurity events (Hackmageddon 2019; PrivacyRight 2019, i.a.), there is no, to our knowledge, prior work that labels cybersecurity events with the rich annotation scheme as our work.

Techniques used to detect events, in general, include hand-crafted rich features (Liao and Grishman 2010; Huang and Riloff 2012; Li, Ji, and Huang 2013), applied deep learning such as GRU (Chen et al. 2015; Ghaeini et al. 2018;

Event Type	Events	Nuggets	Roles/Event
Attack.Databreach	916	1780	2.90
Attack.Phishing	955	1,564	2.34
Attack.Ransom	944	1,585	2.23
Discover.Vulnerability	560	2,122	2.93
Patch.Vulnerability	528	1,419	2.79

Table 2: Number of cyber event instances in our corpus, and the average number of annotated roles per event.

Orr, Tadepalli, and Fern 2018), LSTM (Sha et al. 2018), CNN (Nguyen and Grishman 2016; 2018), RNN (Ghaeini et al. 2018), and LSTM-CNN (Feng, Qin, and Liu 2018). These event detection efforts are based on the ACE 2005 corpus and task. More recent work on event detection has used social media data, as Twitter or Flickr (Mittal et al. 2016; Saeed et al. 2019; Hasan, Orgun, and Schwitter 2019). Most relevant to CASIE are Khandpur et al. (2017), who detect ongoing cyberattacks on Twitter but without considering the event's participants as we do, and Qiu, Lin, and Qiu (2016), who detect events in Chinese newswire.

Data and Annotations

We collected about 5,000 cybersecurity news articles (Cyberwire 2019). These news articles were published in 2017-2019. About 1,000 of them which mention our five events were annotated by three experienced computer scientists, using majority vote to select the final annotations. Multiple cybersecurity events were almost always present in an article, and multiple event types were typically found as well. Our annotation guidelines required that an event argument could only be annotated if an event nugget had been identified in its context, which was set to be the sentence along with its preceding and following sentences. No such constraint was applied to annotation of event nuggets, which could be annotated whenever they appeared, whether or not any event arguments were nearby.

We did not limit the number of words in each annotation label—every event nugget and event argument could be a phrase of any length. Long phrase annotation is important in the cybersecurity domain to properly represent some arguments that cannot be captured in a few words. For example, the phrase *"injected into memory using PowerShell commands"* should be labeled as an argument of type Capabilities. See Table 2 for high-level corpus statistics.

The event annotation process was complicated by many challenges, which we illustrate by examples involving annotating nuggets, arguments and event coreference relations. A description of a cyber attack can include several distinct actions that may be thought of as part of the same event. For example, in

> The hackers have stolen the important files. These were posted for sale on the dark web.

both have stolen and posted would be labelled as Attack.Databreach events, even though they have different meanings and represent different constituent actions making up a single Attack.Databreach event. An event argument can sometimes be interpreted as another, separate event. In the sentence

> Facebook discovered a security issue that allowed hackers to access sensitive information

mentions a Discover.Vulnerability event with a Capabilities argument of allowed hackers to access sensitive information. This Capabilities argument by itself can be misunderstood as an Attack.Databreach event with the word access as its event nugget. A final example involves involves deciding when a set of atomic events co-refer and be part of the same event hopper. The opportunity for confusion arises when an attack event can consist of several steps, each of which is expressed as an action. For example, in the sentence

> Media Prima did not, however, confirm the attack, though sources indicated that the publicly listed company paid the ransom.

the same Attack.Ransom is mentioned twice. The first nugget is "the attack" and the second is "paying the ransom". These two event nuggets appear to have different realis values that conflict about whether the event happened or not. However, they should be annotated as being coreferential. Additional information on the annotation process can be found in Satyapanich (2019b).

Labels	Confused as	% of confusion
Purpose	None	16.82%
	PII	7.66%
	Attack.Databreach	3.83%
Capabilities	None	22.83%
	Attack.Databreach	2.91%
	Data	2.74%
PII	None	13.31%
	Person	5.45%
	Data	5.41%

Table 3: Percentages of the highest confusion between labels in our corpus, with 'None' for no label.

Inter-annotator Agreement

Before starting the annotation, each annotator was given a group of annotation exercises to help them understand rules and follow the annotation guidelines. We measured the inter-annotator agreement using Cohen's Kappa score (Cohen 1968). The score is 0.81 which is considered to be at the low end of the *near-perfect agreement* range, which is typically defined as between 0.81 and 0.99. We also reviewed the annotation disagreement at the level of individual labels by constructing a label confusion matrix, including the five event types and the twenty event arguments. We found that the least confused labels were CVE, Time, and Money. The most highly confused labels are shown in Table 3.

We found that the highest disagreement is the decision of whether to label a token or not (None). The disagree-

System		P	R	F_1
Nugget	w/o (1-5)	74.56	54.36	62.88
	w/ (1-5)	70.79	68.50	69.63
Argument	w/o (1-4, 6-11)	59.13	43.61	50.20
	w/ (1-4, 6), w/o (7-11)	60.51	51.80	55.82
	w/o (1-4, 6), w/ (7-11)	67.56	65.16	66.34
	w/ (1-4, 6-11)	69.08	68.38	68.73

Table 4: Ablation study on the feature sets of the event nugget and argument detection systems.

System	Methods	P	R	F_1
Nugget without Attention	Pre-built BERT	79.60	**80.28**	**79.94**
	Transfer Word2vec	80.43	70.12	74.93
	Domain Word2vec	**85.00**	70.78	77.24
	Cyber-Word2vec	78.19	68.17	72.84
Argument with Attention	Pre-built BERT	72.88	**76.74**	**74.76**
	Transfer Word2vec	67.78	69.51	68.64
	Domain Word2vec	**73.82**	67.45	70.50
	Cyber-Word2vec	73.35	64.17	68.45

Table 5: CASIE event detection scores for different methods

ment mostly happened at the boundaries of the annotation phrase. For example, *"an attempt to sell these"*, annotator A chose to annotate only *"an attempt to sell"* as Purpose and excluded *"these"*, while annotator B chose to annotate the entire phrase as Purpose. We saw this type of disagreement occurring with every label. Another frequent annotation disagreement involved the Purpose argument, which was easily confused with PII and Attack.Databreach. For example, in *"harvest the sensitive information"*, one annotator chose to tag *"harvest"* as an event nugget of type Attack.Databreach and *"the sensitive information"* as an event argument of type PII, while another chose to label the entire phrase.

Experimental Results

Our evaluation used metrics developed for the TAC Event Task (NIST 2015) to find the best mapping between CASIE's output and the correct labels. We used the `types` metric, which computes the score using the overlap in an event nugget or argument mention-span and the ground truth span. We developed CASIE using 8-fold cross validation of 900 articles of training data set, using 100 articles for testing. We tested our sub-systems by averaging scores for five runs.

Event Nugget and Event Argument Detection. To evaluate the effectiveness of features for nugget and argument detection (features (1)-(11) defined above), we performed ablation experiments by grouping features to three feature sets. For nuggets, we included or excluded all features (1)-(5) (features 6-11 were not applicable). For argument detection, we created two sets of grouped features; the first was features (1)-(4) and (6), and the second features (7)-(11).

Arg	P	R	F_1	Arg	P	R	F_1
Money	93.9	92.8	93.3	GPE	76.0	31.7	44.7
CVE	86.8	89.3	88.0	Capa.	45.2	59.1	51.2
Patch	81.4	81.2	81.3	Pur.	65.2	53.7	58.8

Table 6: High (left) and low (right) scores for event arguments; Capa. means Capabilities and Pur. means Purpose.

This experiment was performed on the 8-fold cross validation of training set with the Domain-Word2vec word embeddings. Experimental results are shown in Table 4. Nugget detection scores with the feature set (1)-(5) are higher than without, suggesting that even with expressive neural methods, the feature set provides benefits in detecting nuggets. From the large differences on arguments, it is clear that the features defined for argument detection are also beneficial.

We evaluated CASIE's performance on the test set by including all features (as defined previously). We experimented with the four different word embeddings described previously. The resulting scores, shown in Table 5, are the best performance from development hyperparameter tuning.

Nugget detection scores show that Pre-built BERT gave the best recall and F_1. However, it is not significantly different from the second-ranked method, Domain-Word2Vec, which had the best precision for detecting event nuggets even though its vector size was the smallest. This suggests that cybersecurity-related text has terms that are uncommon or have different senses than in general text. Training with Transfer-Word2vec did not give much benefit. For argument detection, the pre-trained BERT models gave the best performance, likely due to their larger vector size and context-sensitivity.

Table 6 shows the event argument performance. Due to space constraints and the large number of types, we show three with high performance and three with low. We note that identifying *Capabilities* and *Purpose* arguments is difficult because they are often long phrases with technical terms. Another difficult task involved the *GPE* argument type, which had relatively few instances with high variety, with half of them used with the sense of a government organization rather than a place.

Event Argument and Role Linking. Table 7 shows metrics for assigning arguments to roles by comparing two system settings—if the event type that the arguments participated in was specified or not. The scores show the higher performance when specifying the event type. The largest improvement is for the Vulnerable-System-Owner role which can be filled by the same set of arguments (Person and Organization) for the Attacker, Discoverer, Releaser, Trusted-Entity, and Victim. The argument and role linking system when specifying the event type performs very well, especially in predicting a role for Person or Organization arguments. The argument with lowest linking scores is Version, due to difficulties in detecting this argument and the similarity of the Patch-Number and Vulnerable-System-Version roles, as in:

- The new vulnerability affects Xen 4.8.x, 4.7.x,4.6.x,

Role	Event Specific	Not Event Specific
Attacker	0.79	0.80
Discoverer	0.75	0.85
Number-of-Data	0.89	1.00
Number-of-Victim	0.91	1.00
Patch-Number	0.74	0.55
Ransom-Price	0.98	0.99
Releaser	0.69	0.84
Tool	0.87	0.80
Trusted-Entity	0.42	0.76
Victim	0.77	0.88
VS	0.98	0.94
VS-Version	0.69	0.62
VS-Owner	0.09	0.75

Table 7: Argument and role linking scores of two system settings (specify and non-specify) for the associated event type. VS means Vulnerable-System

Realis	P	R	F_1
Micro Avg	77.59	77.70	77.65

Table 8: Scores for identifying event realis attributes.

$\underline{4.5.x}_{VS\text{-}Version}$.

- Users of Struts $\underline{2.3}_{VS\text{-}Version}$ are advised to upgrade to $\underline{2.3.35}_{Patch\text{-}Number}$.

The first contains only *Vulnerable-System-Version* whereas the second also has *Patch-Number*. A heuristic like "Vulnerable-System-Version should be lower than Patch-Number" may help for the second, but not for the first.

Event Realis Identification. Table 8 shows scores of realis identification using embedding features from Domain-Word2vec. The scores are from averaging among three realis types. It shows that the realis identification system performs well.

Conclusion and Future Work

We defined a new cybersecurity event extraction task and provided definitions of five event types, their semantic roles, and argument types that can fill them. Our focused work targets the key tasks in an event detection system: detecting event nuggets and arguments, predicting event realis and linking arguments and nuggets with roles. We collected a corpus of 5,000 news articles discussing cybersecurity events and annotated 1,000 of them with our rich event schema. We developed CASIE as an information extraction system trained using the annotated data and evaluated its performance. We show that neural methods with linguistic and word embedding features can extract cybersecurity event information with high accuracy. Additional details can be found in Satyapanich (2019b) and Satyapanich, Finin, and Ferraro (2019). We make our corpus, annotations and code publicly available (Satyapanich 2019a).

Ongoing work includes improving event argument detection, especially for the *Capabilities* and *Purpose* roles, and mapping arguments to event roles. We are developing components to link event arguments to Wikidata entities and for computing coreference and sequence relations between events. Future work will support additional event types (e.g., DDoS attacks), will align and map the events, arguments, and roles to the Unified Cybersecurity Ontology (Syed et al. 2016) and STIX CTI schema (OASIS 2019), and export extracted information as an integrated knowledge graph.

Acknowledgement. Partial support for this research was provided by a gift from the IBM AI Horizons Network.

References

Cao, K.; Li, X.; Ma, W.; and Grishman, R. 2018. Including new patterns to improve event extraction systems. In *Florida AI Conference*. AAAI.

Chen, Y.; Xu, L.; Liu, K.; Zeng, D.; and Zhao, J. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, 167–176.

Chen, Y.; Liu, S.; Zhang, X.; Liu, K.; and Zhao, J. 2017. Automatically labeled data generation for large scale event extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 409–419.

Cohen, J. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin* 70(4):213.

2019. The Cyberwire. https://thecyberwire.com/.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR* abs/1810.04805.

Feng, X.; Qin, B.; and Liu, T. 2018. A language-independent neural network for event detection. *Science China Information Sciences* 61(9):092106.

Ghaeini, R.; Fern, X. Z.; Huang, L.; and Tadepalli, P. 2018. Event nugget detection with forward-backward recurrent neural networks. *arXiv preprint arXiv:1802.05672*.

Grishman, R., and Sundheim, B. 1996. Message understanding conference-6: A brief history. In *16th International Conference on Computational Linguistics*.

Hackmageddon. 2019. Hackmageddon. http://hackmageddon.com.

Hasan, M.; Orgun, M. A.; and Schwitter, R. 2019. Real-time event detection from the twitter data stream using the twitternews+ framework. *Information Processing & Management* 56(3):1146–1165.

Huang, R., and Riloff, E. 2012. Modeling textual cohesion for event extraction. In *26th AAAI Conference on Artificial Intelligence*.

Ji, H., and Grishman, R. 2008. Refining event extraction through cross-document inference. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, 254–262.

Khandpur, R. P.; Ji, T.; Jan, S.; Wang, G.; Lu, C.-T.; and Ramakrishnan, N. 2017. Crowdsourcing cybersecurity: Cyber attack detection using social media. In *Conference on Information and Knowledge Management*, 1049–1057. ACM.

Kim, J.-D.; Ohta, T.; Pyysalo, S.; Kano, Y.; and Tsujii, J. 2009. Overview of BioNLP'09 shared task on event extraction. In *Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, 1–9. ACL.

Li, W.; Cheng, D.; He, L.; Wang, Y.; and Jin, X. 2019. Joint event extraction based on hierarchical event schemas from framenet. *IEEE Access* 7:25001–25015.

Li, Q.; Ji, H.; and Huang, L. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 73–82.

Liao, S., and Grishman, R. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 789–797.

Lim, S. K.; Muis, A. O.; Lu, W.; and Ong, C. H. 2017. Malwaretextdb: A database for annotated malware articles. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1557–1567.

Liu, S.; Chen, Y.; He, S.; Liu, K.; and Zhao, J. 2016. Leveraging Framenet to improve automatic event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 2134–2143.

Liu, J.; Chen, Y.; Liu, K.; and Zhao, J. 2018. Event detection via gated multilingual attention mechanism. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Luong, M.; Pham, H.; and Manning, C. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; and McClosky, D. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, 55–60.

Mendes, P.; Jakob, M.; Garcia-Silva, A.; and Bizer, C. 2011. Dbpedia spotlight: Shedding light on the web of documents. In *International Conference on Semantic Systems*, 1–8.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mitamura, T.; Liu, Z.; and Hovy, E. H. 2015. Overview of TAC KBP 2015 event nugget track. In *Text Analysis Conference*. National Institute of Standards and Technology.

Mittal, S.; Das, P. K.; Mulwad, V.; Joshi, A.; and Finin, T. 2016. Cybertwitter: Using twitter to generate alerts for cybersecurity threats and vulnerabilities. In *Conf. on Advances in Social Networks Analysis and Mining*, 860–867.

Nguyen, T. H., and Grishman, R. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Conference on Empirical Methods in Natural Language Processing*, 886–891.

Nguyen, T. H., and Grishman, R. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

NIST. 2015. TAC KBP event track. http://tac.nist.gov/-2015/KBP/Event/.

OASIS. 2019. STIX version 2.1. https://docs.oasis-open.org/cti/stix/v2.1/csprd01/stix-v2.1-csprd01.pdf.

Orr, J. W.; Tadepalli, P.; and Fern, X. 2018. Event detection with neural networks: A rigorous empirical evaluation. *arXiv preprint arXiv:1808.08504*.

Padia, A.; Roy, A.; Satyapanich, T.; Ferraro, F.; Pan, S.; Park, Y.; Joshi, A.; Finin, T.; et al. 2018. UMBC at SemEval-2018 Task 8: Understanding text about malware. In *Proc. International Workshop on Semantic Evaluation*.

PrivacyRight. 2019. Privacyright clearing house. http://privacyrights.org.

Qiu, X.; Lin, X.; and Qiu, L. 2016. Feature representation models for cyber attack event extraction. In *International Conference on Web Intelligence Workshops*, 29–32. IEEE.

Saeed, Z.; Abbasi, R. A.; Razzak, M. I.; and Xu, G. 2019. Event detection in twitter stream using weighted dynamic heartbeat graph approach. *arXiv preprint arXiv:1902.08522*.

Satyapanich, T.; Finin, T.; and Ferraro, F. 2019. Extracting rich semantic information about cybersecurity events. In *Second Workshop on Big Data for CyberSecurity*. (held in conjunction with the IEEE International Conference on Big Data).

Satyapanich, T. 2019a. CASIE Repository. https://github.-com/Ebiquity/CASIE.

Satyapanich, T. 2019b. *Modeling and extracting information about cybersecurity events from text*. Ph.D. Dissertation, University of Maryland, Baltimore County.

Sha, L.; Qian, F.; Chang, B.; and Sui, Z. 2018. Jointly extracting event triggers and arguments by dependency-bridge RNN and tensor-based argument interaction. In *AAAI*.

Syed, Z.; Padia, A.; Finin, T.; Mathews, L.; and Joshi, A. 2016. UCO: A unified cybersecurity ontology. In *Workshop on AI for Cyber Security*, 195–202. AAAI.

Vrandečić, D., and Krötzsch, M. 2014. Wikidata: A free collaborative knowledgebase. *CACM* 57(10):78–85.

Walker, C.; Strassel, S.; Medero, J.; and Maeda, K. 2006. ACE 2005 multilingual training corpus. Technical report, Linguistic Data Consortium.

Zheng, G.; Mukherjee, S.; Dong, X.; and Li, F. 2018. Opentag: Open attribute value extraction from product profiles. In *Conf. on Knowledge Discovery & Data Mining*.