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-
Specifically we define the following events: Attack.Databreach, Attack.Phishing, Attack.Ransom, and Discover.Vulnerability. Each event type specifies the roles it can have and constraints on the arguments that can fill them.

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.

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.Vulnerability 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...
<table>
<thead>
<tr>
<th>Types</th>
<th>Roles</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyber Attack</td>
<td>Attacker</td>
<td>Organization; Person</td>
</tr>
<tr>
<td></td>
<td>Victim</td>
<td>Device; Organization; Person; Product; Website</td>
</tr>
<tr>
<td></td>
<td>Attack-Pattern</td>
<td>Capabilities</td>
</tr>
<tr>
<td></td>
<td>Tool</td>
<td>File; Malware; Website</td>
</tr>
<tr>
<td></td>
<td>Damage-Amount</td>
<td>Money</td>
</tr>
<tr>
<td></td>
<td>Number-of-Victim</td>
<td>Number</td>
</tr>
<tr>
<td></td>
<td>Purpose</td>
<td>Purpose</td>
</tr>
<tr>
<td></td>
<td>Place</td>
<td>GPE</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>Time</td>
</tr>
<tr>
<td></td>
<td>CVE</td>
<td>CVE</td>
</tr>
<tr>
<td></td>
<td>Vulnerability</td>
<td>Vulnerability</td>
</tr>
<tr>
<td></td>
<td>VS</td>
<td>Device; Product; Website</td>
</tr>
<tr>
<td></td>
<td>VS-Version</td>
<td>Version</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>Time</td>
</tr>
</tbody>
</table>

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 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.
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 performance 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 connedActual out of an alleged $100 million in a phishing scamActual.
- Other: Apple has not yet clarifiedOther whether the latest versions of macOS and iOS are vulnerableOther.
- Generic: PhishingGeneric has grown to be one of the most serious threats to healthcare organizations.

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

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Events</th>
<th>Nuggets</th>
<th>Roles/Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack.Databreach</td>
<td>916</td>
<td>1,780</td>
<td>2.90</td>
</tr>
<tr>
<td>Attack.Phishing</td>
<td>955</td>
<td>1,564</td>
<td>2.34</td>
</tr>
<tr>
<td>Attack.Ransom</td>
<td>944</td>
<td>1,585</td>
<td>2.23</td>
</tr>
<tr>
<td>Discover.Vulnerability</td>
<td>560</td>
<td>2,122</td>
<td>2.93</td>
</tr>
<tr>
<td>Patch.Vulnerability</td>
<td>528</td>
<td>1,419</td>
<td>2.79</td>
</tr>
</tbody>
</table>

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 action. For example, in the sentence

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).

Inter-annotator Agreement

Before starting the annotation, each annotator was given a group of annotation exercises to help them understand the 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.

Table 3: Percentages of the highest confusion between labels in our corpus, with 'None' for no label.
Table 4: Ablation study on the feature sets of the event nugget and argument detection systems.

<table>
<thead>
<tr>
<th>System</th>
<th>Methods</th>
<th>P</th>
<th>R</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nugget</td>
<td>w/o (1-5)</td>
<td>74.56</td>
<td>54.36</td>
<td>62.88</td>
</tr>
<tr>
<td></td>
<td>w/ (1-5)</td>
<td>70.79</td>
<td>68.50</td>
<td>69.63</td>
</tr>
<tr>
<td>Argument</td>
<td>w/o (1-4, 6-11)</td>
<td>59.13</td>
<td>43.61</td>
<td>50.20</td>
</tr>
<tr>
<td></td>
<td>w/ (1-4, 6), w/o (7-11)</td>
<td>60.51</td>
<td>51.80</td>
<td>55.82</td>
</tr>
<tr>
<td></td>
<td>w/ (1-4, 6), w/ (7-11)</td>
<td>67.56</td>
<td>65.16</td>
<td>66.34</td>
</tr>
<tr>
<td></td>
<td>w/ (1-4, 6-11)</td>
<td>69.08</td>
<td>68.38</td>
<td>68.73</td>
</tr>
</tbody>
</table>

Table 5: CASIE event detection scores for different methods

<table>
<thead>
<tr>
<th>System</th>
<th>Methods</th>
<th>P</th>
<th>R</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nugget</td>
<td>Pre-built BERT</td>
<td>79.60</td>
<td>80.28</td>
<td>79.94</td>
</tr>
<tr>
<td></td>
<td>Transfer Word2vec</td>
<td>80.43</td>
<td>70.12</td>
<td>74.93</td>
</tr>
<tr>
<td></td>
<td>Domain Word2vec</td>
<td>85.00</td>
<td>70.78</td>
<td>77.24</td>
</tr>
<tr>
<td></td>
<td>Cyber-Word2vec</td>
<td>78.19</td>
<td>68.17</td>
<td>72.84</td>
</tr>
<tr>
<td>Argument</td>
<td>Pre-built BERT</td>
<td>72.88</td>
<td>76.74</td>
<td>74.76</td>
</tr>
<tr>
<td></td>
<td>Transfer Word2vec</td>
<td>67.78</td>
<td>69.51</td>
<td>68.64</td>
</tr>
<tr>
<td></td>
<td>Domain Word2vec</td>
<td>73.82</td>
<td>67.45</td>
<td>70.50</td>
</tr>
<tr>
<td></td>
<td>Cyber-Word2vec</td>
<td>73.35</td>
<td>64.17</td>
<td>68.45</td>
</tr>
</tbody>
</table>

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 2. Nugget detection scores with 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 F₁. 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.

**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).

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

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

Table 8: Scores for identifying event realis attributes.

<table>
<thead>
<tr>
<th>Realis</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro Avg</td>
<td>77.59</td>
<td>77.70</td>
<td>77.65</td>
</tr>
</tbody>
</table>

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.

### References

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


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.


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.


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.