Sensor Fusion and Structured Prediction for Cyberattack Event Networks

Alex Memory
Leidos, Inc.
Arlington, VA, USA
University of Maryland
College Park, MD, USA
memoryac@leidos.com

W. Graham Mueller
Leidos, Inc.
Arlington, VA, USA
muellerwg@leidos.com

ABSTRACT

Early detection of cyberattacks – such as data breaches or ransomware – is critical to mitigate their effects. Despite advances in automated cyberattack sensors, many attacks are still detected days or months after they occur. We propose a new approach using statistical relational learning to fuse cyberattack sensor outputs and generate attack predictions. Leveraging the graphical structures of both sensor outputs and cyberattack events themselves, we achieve higher accuracy than individual sensors by reasoning collectively over both sensors and attacks. In addition to improved accuracy, our predictions also are more useful to analysts because they are structured objects containing details of the predicted attacks. We measure accuracy and scalability in an extensive empirical evaluation of our approach using a database of real cyberattacks against a large corporation. We show that, relative to a sensors-only baseline, our approach increases accuracy by up to seven percent and doubles the lift of high-confidence predictions.

CCS CONCEPTS

• **Computing methodologies** → *Statistical relational learning*; *Structured outputs*; *Probabilistic reasoning*; • **Information systems** → *Information integration*; • **Security and privacy**;

KEYWORDS

Probabilistic inference, structured prediction, sensor fusion, cyberattacks

ACM Reference Format:
Alex Memory and W. Graham Mueller. 2019. Sensor Fusion and Structured Prediction for Cyberattack Event Networks . In *Proceedings of Mining and Learning with Graphs (MLG'19).* ACM, New York, NY, USA, 8 pages.

1 INTRODUCTION

Cyberattacks are a growing concern for cybersecurity analysts and administrators of networks in governments, commercial companies, and educational institutions. For example, the average data breach exposes tens of thousands of records, and costs the victim organization millions of dollars to mitigate [17]. Early detection, or prediction, of attack events can reduce these costs, but the average breach still takes nearly 200 days to detect [17]. There has been progress to develop automated sensors for detecting attack events [3, 14, 23]. To improve the utility of the sensors, *sensor fusion* has the goal of linking and combining the outputs, and *structured prediction* has the goal of producing a set of accurate and detailed event predictions, combining the strengths of the sensors (Section 2).

We propose a new approach to *solve the sensor fusion and structured prediction problems jointly*. Our approach is based on three insights about real cyberattack events' timing and details – which we refer to as *roles*: (a) roles of events are interdependent, (b) events occur in clusters, and (c) events evolve over time (Section 3).

Based on these insights about the structure of attacks, we refer to a timeline of attacks as a cyberattack event network (CEN). We introduce the *event-relational* model using statistical dependencies capturing the three insights, enabling fusion and prediction via collective reasoning over all sensors and events (Section 4).

We apply the general event-relational model to CENs; we refer to our implementation using probabilistic soft logic [1] as Cyber Event Relational Fusion (CERF) (Section 5). We conduct an extensive empirical evaluation of CERF using a database with nine months of real cyberattacks against a large corporation in the United States (US). We show how CERF fuses sensors to increase the level of detail in predictions. We show that CERF increases accuracy of predicted events by three percent, as measured by area under the receiver operator characteristic (AuROC) curve, and more than doubles maximum lift for high-confidence predictions. We show that, with as little as 10% partially observed events, we increase AuROC by an additional four percent (Section 6).

2 BACKGROUND

In this section, we review current cyberattack sensors, the sensor fusion problem, and techniques for structured prediction.

2.1 Cyberattack Sensors and Sensor Graphs

Cybersecurity analysts and network administrators have long used a variety of cyberattack sensors to detect, prevent or mitigate attacks. In Table 1, we list example sensor outputs, sources used by sensor software to produce those outputs, and typical uses by analysts. There has been recent progress to use a wider variety of input sources, and produce a wider variety of output types [3, 14, 23]. For this work, we assume that sensors produce an output collection every time step, and provide *confidence* levels with each output. Many sensor outputs are indicators that analysts then need to interpret;

MLG'19, August 2019, Anchorage, Alaska USA

Table 1: Example sensor output types, possible sources, and typical uses by cybersecurity analysts or network administrators.

Category	Output Type	Source	Uses
Network	IP address	Recent network-based attacks	Block network communications with IP address
Network	Network port or protocol	Recent attacks	Monitor network activity using port or protocol
Host	Hash code or binary signature	Detected malicious files	Scan systems for files with hash code or signature
Host	Host name	Computer targeted in recent attack	Monitor host for further attacks
Email	Words in email subject	Recent malicious emails	Monitor emails for similar words
Email	External address domain	Senders of recent malicious emails	Block emails from sender domain

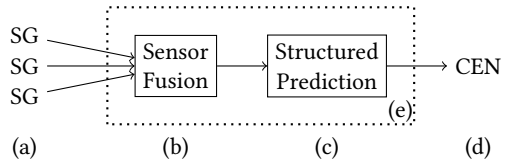

Figure 1: Sensor fusion combines sensor graphs (SG). Structured prediction produces a cyberattack event network (CEN). Our proposed approach (dotted) combines both.

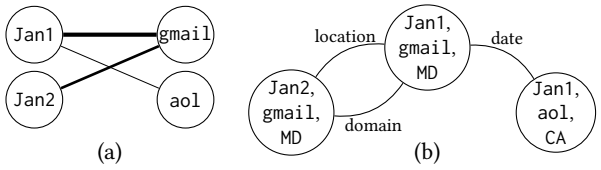

Figure 2: Examples of (a) a SG for email domain role; and (b) a CEN with three events, $K = 2$ roles, and three similarities.

we instead explicitly *predict* events. Ideally, predictions are both *accurate* and *differentiated*. Accurate predictions reveal true attacks while minimizing false positives. Differentiated predictions identify multiple characteristics of an attack.

Example 2.1. Predictions of phishing emails, a common type of attack, are *differentiated* if they combine predicted time with details such as the sender address domain, a word appearing in the subject line, and the work location of the recipient. Using differentiated phishing predictions, network administrators can make precise email filters, and search efficiently for related attacks.

While some sensor outputs are simple scalar time series, we focus on structured outputs containing details that can be combined into differentiated predictions. We refer to these dynamically-changing structures as *sensor graphs* (SG) (Figure 1a). Even SGs typically contain only a single type of output, e.g., only IP addresses or only hash codes, because each sensor uses just one or a few types of input data. Therefore we assume a SG is a bipartite graph comprising nodes for each time step and *label* (e.g., a specific IP address) for some output type. Edges between time steps and labels represent sensor outputs, weighted by confidence level (Figure 2a).

2.2 Sensor Fusion

Sensor fusion (Figure 1b) is the task of linking and combining sensor graphs. A challenge is that SGs from separate sensors may have no nodes in common with the exception of time steps. Linking shared dates across SGs is a starting point, but does not reveal how to link label nodes across SGs to form differentiated predictions. Learning cross-sensor dependencies is one way to link labels (Section 4.2).

2.3 Structured Prediction

We represent events and their details as *structured objects*; each is a K-ary tuple $(r_1, \ldots, r_K)$. We refer to details of an event as its *roles*, as they identify participants and objects involved. The value for each role is some constant from a predefined set of labels.

Structured prediction is the task of predicting structured objects; in our case, we predict cyberattack events from the results of sensor fusion (Figure 1c). Typically, we make sets of predictions covering some time period. Furthermore, we use the roles of events to form a network of related events, which we call *cyberattack event networks* (CEN) (Figures 1d and 2b). Techniques for structured prediction are well studied [9, 18, 20], and applied to a number of tasks, which we review briefly here.

2.3.1 Predict a Single Role. Predicting even a single role of an event requires sophistication. For example, recent multi-label classification approaches aim to leverage relationships between labels when assigning them to a role. Relationships between labels in a single role include pairwise similarity, organization into categories, or logical constraints [13]. For cyberattack events, we use similar relationships between labels (see Sections 5.3.2 and 5.3.3).

2.3.2 Predict Multiple Roles. Other tasks involve multiple roles. For example, the goal of event sequence label learning is to predict the composition of labels over a collection of roles. Riedel et al. do this with statistical relational learning [19]; we use a similar technique to predict composition of cyberattack roles (see Section 4.2).

2.3.3 Predict Multiple Events. Predicting multiple structured objects at once – each with multiple roles – can have advantages such as improved accuracy. For example, collective inference [4, 8, 12] and hierarchical tensor representations [11] have been used to generate recommendations with two or more roles. Like Kouki et al., we use collective inference to predict clusters of events within time steps (see Section 4.3) and over time (see Section 4.4).

Although sensor fusion and structured prediction have distinct goals, we will solve them *jointly* (Figure 1e), leveraging the graphical structures of SGs and CENs. We describe CENs in detail in Section 3.

3 CYBERATTACK EVENT NETWORKS

In this section, we describe CENs, our motivating problem setting. We will illustrate three key insights about real attacks using a dataset of attacks events, which will be described fully in Section 6.1.1. We refer to this dataset as the ground truth (GT) CEN.

3.1 Roles of Events are Interdependent

We see in real cyberattack events that their roles are interdependent. For example, in Example 2.1 the email subject may have been crafted to target victims at that location to appear more legitimate. To confirm this intuition empirically in the GT CEN, we calculate the Kullback-Leibler divergence $D_{\mathrm{KL}}(p||p')$ where $p(r_1, r_2, \ldots, r_K)$ is the empirical joint distribution of roles in the CEN, and p' assumes independence: $p'(r_1, r_2, \ldots, r_K) = p(r_1)p(r_2)\cdots p(r_K)$. This measure (total correlation [21]) for the GT CEN is 2.1, which is over 70% higher than if role values were shuffled across events in the training set, averaged over five trials. Although unsurprising, this confirms the need for dependencies between roles when linking SGs to form predicted events. In Section 4.2 we define a model with these dependencies (event-propositional).

3.2 Events Occur in Clusters

We also find that clusters of similar events tend to occur together, even in the same time step, e.g., on the same day. We confirm this using a sample of the GT CEN, which has over one thousand events. To produce the sample, we define an initial set of edges between events representing similarities such as events having the same location (see Figure 2b). To focus on individual time steps, we remove edges relating events across multiple days. We combine all parallel edges, resulting in edge weights between one and four. A weight of one means two events occur on the same day; additional similarity results in higher weights. As shown in Figure 4, most events on the same day are similar with respect to one or more measures. In Figure 3, we plot a five-name snowball sample [5, 6] on 500 seed events with a force-directed layout. Removing cross-day edges causes separate connected components for different days.

We arbitrarily chose one role – location – to color the nodes, which reveals that locations are distributed non-uniformly over days. For example, the component labeled (a) in Figure 3 has a high proportion of attacks in California (green), while on other days (b) attacks in California are rare. This pattern, known as homophily [22], is frequently seen in social networks. In Section 4.3, we define a model with this dependency (time-propositional).

3.3 Events Evolve Over Time

Cybersecurity analysts have observed that attack events tend to occur in clusters over time, progressing through stages. For example, according to the Cyber Kill Chain® framework, exploitation attacks follow delivery attacks, which follow reconnaissance attacks [7]. We consider a related but simpler form of evolution over time: whether events tend to occur in clusters of consecutive time steps. We confirm this in the GT CEN by counting events occurring in fixed intervals; we arbitrarily chose seven days. We compare that to a Poisson distribution resulting from assuming independent arrival times, i.e., not clustered. We plot both in Figure 5, which shows that events tend to cluster in large numbers in some intervals (e.g., Figure 5a), leaving few in other intervals (b). That is, if we have high confidence in some attack occurring in a time step, it is likely that other attacks occur in surrounding time steps. In Section 4.4, we define a model with this dependency (event-relational).

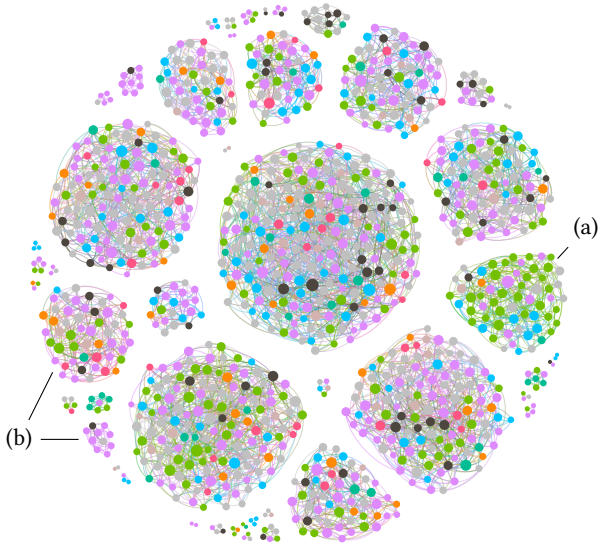

Figure 3: Sample of a real CEN, colored by victim location.

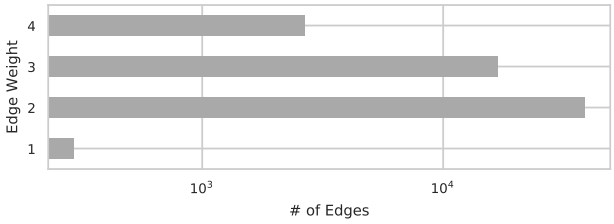

Figure 4: Edge weight distribution for graph in Section 3.2.

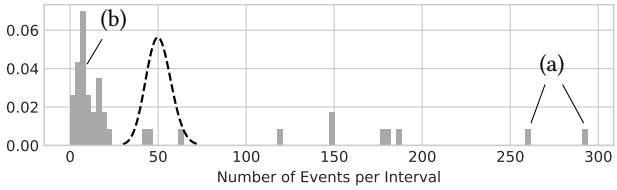

Figure 5: Distribution of actual events per interval (grey), and assuming independent arrivals (dashed).

4 EVENT-RELATIONAL MODEL

In this section, we introduce the event-relational structured prediction model, which will be a framework to encode the insights from Section 3. For ease of understanding, we lead up to the event-relational model in four steps; in each step we define a model that

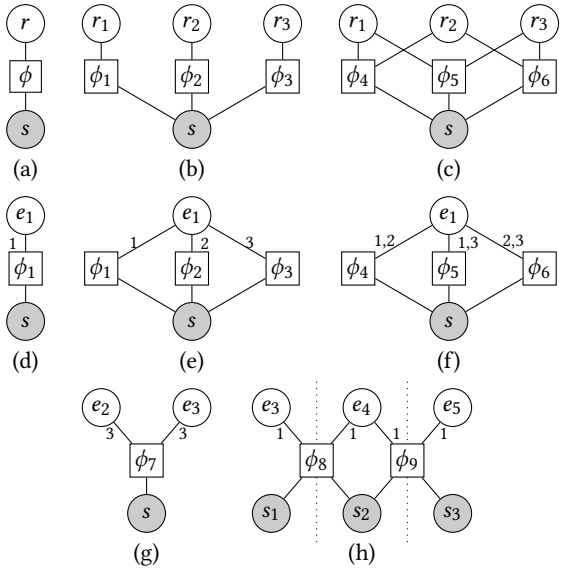

Figure 6: Examples of four models: Role-propositional (a). Event-propositional (b and c). Time-propositional rewrites of previous (d-f). Time-propositional with two events (g). Event-relational (h) with three time steps (dotted).

extends the previous. The most basic of the four is *role-propositional*, which uses a single sensor as input and predicts a single role of a single event. The remaining three models extend the first, and each captures one of the three insights:

- Roles of events are interdependent ⇒ *Event-Propositional*
- Events occur in clusters ⇒ *Time-Propositional*
- Events evolve over time ⇒ *Event-Relational*

We define the models in the following four sections.

4.1 Baseline: Role-Propositional

A sensor produces an output vector $s \in \mathcal{S}$ that corresponds to some *role* of an event represented by the random variable (RV) r, where the value for r is a label from set $\mathcal{L}$. Predicting r is a multiclass classification problem. We capture the statistical dependency between the sensor output and r using feature function $\phi : \mathcal{L} \times \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$, a technique commonly used in probabilistic models [9, 18]. Our definition for ϕ is problem-specific; generally, it measures *incompatibility* between a predicted value of r and what we observe in sensor data. Given $\mathcal{L}$ and s, we pick an optimal value for r as $\arg\min_{r \in \mathcal{L}} \phi(r, s)$. We refer to this model as *role-propositional*, and it represents a baseline approach. We illustrate the model in Figure 6a, with shading indicating the observed variable.

Example 4.1. Suppose role r represents the port targeted in a network-based attack event, and the set of possible labels contains commonly used ports, e.g., $\mathcal{L} = \{22, 53, 80\}$. Given a sensor output reporting a high confidence of .9 for an attack on port 22 at time step t, feature functions used in our approach would produce a high incompatibility score for any assignment to r other than 22, which has zero incompatibility. The optimum prediction is $r = 22$.

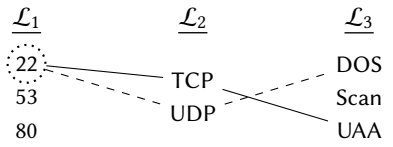

Figure 7: Selected role values from sets $(\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3)$ in predictions from examples 4.1 (circle), 4.2 (dashed), and 4.3 (solid).

4.2 Extension: Event-Propositional

A limitation of the role-propositional model is it uses a single input sensor and predicts events with a signal role, which are not well-differentiated. We will change the model as follows:

- Extend s and $\mathcal{S}$ with partitions for additional sensor outputs
- Vector $\mathbf{r} = (r_1, \ldots, r_K)$ replaces the single role r
- K sets $(\mathcal{L}_1, \ldots, \mathcal{L}_K)$ – one for each role – replace $\mathcal{L}$
- Set $\mathbf{\Phi} = \{\phi_1, \ldots, \phi_p\}$ of feature functions replaces ϕ

This *event-propositional* model fuses K sensors and predicts K roles. In Figure 6b, we illustrate an example with $K = 3$.

Example 4.2. We extend Example 4.1, adding sensors for network protocol and attack class producing high-confidence outputs `UDP` (User Datagram Protocol) and `DOS` (denial of service), respectively. The optimum with respect to ϕ_1, ϕ_2, and ϕ_3 is $\mathbf{r} = (r_1, r_2, r_3) = (22, \texttt{UDP}, \texttt{DOS})$. In Figure 7, we illustrate the three sets $(\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3)$, and the selected roles from our prediction (dashed line).

Feature functions are flexible in the event-propositional model, and can capture relationships *between* roles. For example, in Figure 6c, we use function $\phi_4 : \mathcal{L}_1 \times \mathcal{L}_2 \times \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$. This allows us to predict the most likely *composition* of an event, which may differ from the optimal assignment according to individual sensors.

Example 4.3. Although assignment $\mathbf{r}$ in Example 4.2 is optimal with respect to functions ϕ_1, ϕ_2, and ϕ_3, that combination is improbable in this domain. Assignment $\mathbf{r} = (22, \texttt{TCP}, \texttt{UAA})$ is more probable because the Secure Shell (SSH) service running on port 22 uses TCP (Transmission Control Protocol), and – as a remote access service – unauthorized access attempts (UAA) against SSH are more likely than DOS attacks. In Figure 7, the solid line marks our updated prediction with the more compatible assignment.

As shown in the example, the best assignment may require balancing functions encoding sensor outputs (ϕ_1, ϕ_2, ϕ_3) and functions encoding compatible event composition (ϕ_4, ϕ_5, ϕ_6). The event-propositional model optimizes all functions in $\mathbf{\Phi}$ simultaneously, forcing assignments to consider all sensors and all types of compatibility *collectively*. See Section 5.4.2 for an extended example.

The feature functions in $\mathbf{\Phi}$ may need to be weighted differently, e.g., to tune the balance between sensors and compatibility. We train a weight vector $\mathbf{w} \in \mathbb{R}^p_{>0}$ (see Section 5.2.1), associate one weight from the vector $\mathbf{w}$ with each function, and find optimal assignment $\mathbf{r} \in \mathcal{L}_1 \times \ldots \times \mathcal{L}_K$ as $\arg\min_{\mathbf{r}} \mathbf{w}^\intercal \mathbf{\Phi}(\mathbf{r}, s)$.

4.3 Extension: Time-Propositional

Although the event-propositional model gives the most likely composition of an event, represented by $\mathbf{r}$, it is limited to a single event. It also doesn't indicate *whether* an event will occur. We will replace

the single event $\mathbf{r}$ with a more general binary encoding. In the encoding, an element of vector $\mathbf{e} = (e_1, \ldots, e_u)$ exists for each of u possible configurations of roles in $\mathcal{L}_1 \times \ldots \times \mathcal{L}_K$ and e_i is true iff its corresponding configuration of roles occurs in an event. Because this models all events in a time step, we refer to this model as time-propositional. We rewrite the event-propositional feature functions for the binary representation; they become $\mathbf{\Phi} : \mathcal{S} \times \{0,1\}^u \rightarrow \mathbb{R}_{\geq 0}$ and defined over projections of events. We illustrate rewriting in Figure 6d-f, in which an edge annotated with a set $\gamma \subseteq \{1, \ldots, K\}$ refers to a feature function defined over an event e, but using only a projection of e's roles, i.e., $\pi_\gamma(e)$ using relational algebra.

Example 4.4. Logical atoms are a convenient representation for $\mathbf{e}$, so we use the K-ary logical predicate *event* to represent all events, where the number K of logical terms representing roles varies by the type of attack. CERF also uses this representation (Section 5.2). Extending Example 4.3, we rewrite assignment $(r_1, r_2, r_3) = (22, \texttt{TCP}, \texttt{UAA})$ as $e_1 = \text{event}(22, \texttt{TCP}, \texttt{UAA})$. Rule ϕ_4 in Figure 6c places dependencies on r_1 and r_2; the equivalent function in Figure 6f uses projection $\pi_{1,2}(e_1)$ to do the same with the 1st two roles of the event, e.g., assigning low incompatibility between 22 and TCP.

Handling multiple events is the goal of time-propositional models; we illustrate a multi-event example in Figure 6g.

Example 4.5. As shown in Section 3.2, similar events tend to cluster. Suppose we predict a phishing event $e_2 = \text{event}(\texttt{gmail}, \texttt{timesheet}, \texttt{MD})$ (see Example 2.1). Also, suppose we have similarities across location labels, e.g., between Maryland (MD) and Virginia (VA) (see Section 5.3.2). Function ϕ_7 could add a dependency linking our confidence in e_2 with confidence in an additional similar event $e_3 = \text{event}(\texttt{gmail}, \texttt{timesheet}, \texttt{VA})$.

Given s and $\mathbf{w}$, we find optimal $\mathbf{e} \in \{0,1\}^u$ as $\arg\min_{\mathbf{e}} \mathbf{w}^\intercal \mathbf{\Phi}(\mathbf{e}, s)$.

4.4 Extension: Event-Relational

To capture dependencies across time steps $t_1, \ldots, t_n$, we replace s with vectors $\mathbf{s} = (s_1, \ldots, s_n)$. We refer to this model as event-relational and illustrate an example for $n = 3$ in Figure 6h.

Example 4.6. Clusters of attacks tend to continue over multiple time steps (Section 3.3). We extend Example 4.4 and assume that in step t_1 event e_3 occurs, which we now represent as $\text{event}(t_1, \texttt{gmail}, \texttt{timesheet}, \texttt{VA})$. Function ϕ_8 adds a dependency between e_3 and e_4, where e_4 is identical except it occurs in t_2.

In summary, the following four parameters (counts) characterize the size and complexity of an event-relational model:

- $\underline{K}$ roles in each event
- $\underline{u} = |\mathbf{e}| = |\mathcal{L}_1 \times \ldots \times \mathcal{L}_K|$ possible events in each time step
- $\underline{n}$ time steps
- $\underline{p} = |\mathbf{\Phi}|$ feature functions

With the event-relational model, we find an optimal $\mathbf{e}$ as follows:

$$\arg\min_{\mathbf{e} \in \{0,1\}^{nu}} \mathbf{w}^\intercal \mathbf{\Phi}(\mathbf{e}, \mathbf{s}) \tag{1}$$

Although solving Equation 1 exactly is NP-hard, in Section 5 we describe how CERF finds an approximate solution efficiently. We have shown how the event-relational model enables statistical dependencies including all types shown in Figure 6d-h, which capture each of the insights described in Section 3.

5 CYBER EVENT RELATIONAL FUSION

In this section, we describe CERF, our system applying the event-relational model from Section 4 to CENs (Section 3).

5.1 Probabilistic Soft Logic

We use probabilistic soft logic (PSL) [1] to implement the event-relational model. Using PSL has several advantages: (a) PSL uses logic, a natural representation for roles, events and feature functions. (b) Solving Equation 1 exactly is NP-hard, but PSL provides a high quality approximation. (c) PSL scales well to large CENs.

5.2 Mapping to Event-Relational Model

We use logical atoms to represent event set $\mathbf{e}$ (see Section 4.3). We use the following mapping to produce the model in PSL:

- Roles ⇒ Logical terms (logical variables or constants)
- Events ⇒ Logical atoms
- Sensor output ⇒ Ground logical atoms

5.2.1 Inference and Learning. A PSL program defines a hinge-loss Markov random field (HL-MRF), where weighted logical rules are feature functions $\mathbf{\Phi}$ and atoms with soft truth values are RVs. Maximum a posteriori (MAP) inference in the HL-MRF allows a highly efficient approximation [1]. This is done in PSL by approximating the following, where $\mathbf{x}$ and $\mathbf{y}$ represent observed and inferred atoms, respectively: $\arg\min_{\mathbf{y} \in [0,1]^v} \mathbf{w}^\intercal \mathbf{\Phi}(\mathbf{y}, \mathbf{x})$. A high quality binary solution $\mathbf{y} \in \{0,1\}^v$ is possible by applying conditional probabilities rounding to the soft-valued output [1]. In CERF, $\mathbf{x} \equiv \mathbf{s}$, $\mathbf{y} \equiv \mathbf{e}$, and $v \equiv nu$, establishing equivalence with our objective in Equation 1. We learn $\mathbf{w}$ with maximum likelihood estimation [2].

5.2.2 Soft Truth Values. In addition to the binary solution, the soft-valued $\mathbf{e} \in [0,1]^{nu}$ from PSL has advantages: We can present high-value events to analysts, and measure AuROC and lift (Section 6).

5.3 Predicates

Logical predicates define the inputs, outputs, and latent variables of a PSL program. We use the following three predicates:

- *event*: a CEN node; its first term is time step T; the remaining K terms vary by event type
- $\sigma_i(R, R')$: a CEN edge – similarity of $R \in \mathcal{L}_i$ and $R' \in \mathcal{L}_i$
- $\mathrm{s}_i(T, R)$: an edge of a SG for role i

MAP inference assigns truth values of inferred event atoms. Truth values of all observed event atoms (see Section 6.4) are set to 1.0.

5.3.1 Sensors. Truth values of s_i atoms are based on the confidence of the sensor for those outputs (Section 6.1.3).

5.3.2 Similarities. All similarity σ_i atoms are observed; their truth values are set by external similarity functions.

5.3.3 Categories. We use sets of categories over labels within roles. Binary predicate $\succ$ represents membership in a category; $R_i \succ \ell$ is true iff label $R_i \in \mathcal{L}_i$ is a member of category $\ell \in \chi_i$, where set χ_i contains all categories for role i.

Categories have several benefits: (a) They add sensor information to labels. (b) They control the number of parameters (weights) and over-fitting for template-based rules (see Section 5.4.1). (c)

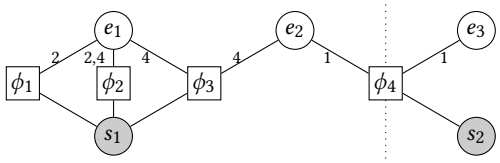

Figure 8: Fragment of a CERF model with $n = 2$ and examples of Rule RP (ϕ_1), Rule EP (ϕ_2), Rule TP (ϕ_3), and Rule ER (ϕ_4).

They help CERF scale to larger CENs with higher values for K by controlling the number of groundings of template-based rules.

5.4 CERF PSL Rules

PSL rules implement feature functions of the event-relational model. The simplest rule is the prior $\neg\text{event}(T, R_1, \ldots, R_K)$, i.e., any attack is individually improbable. The remaining CERF rules are as follows, where rules with parameterized weights, e.g., $w_{\text{TP}}(i)$, are *templates*:

$$w_{\text{RP}} : \bigwedge_{i \in 1,\ldots,K} \text{s}_i(T, R_i) \rightarrow \text{event}(T, R_1, \ldots, R_K) \quad \text{(RP)}$$

$$w_{\text{EP}}(i, j, \ell, \ell') : \neg\Big(\text{event}(T, R_1, \ldots, R_i, \ldots, R_j, \ldots, R_K) \wedge (R_i \succ \ell) \wedge \big(R_j \succ \ell'\big)\Big) \quad \text{(EP)}$$

$$w_{\text{TP}}(i) : \text{event}(T, R_1, \ldots, R_i, \ldots, R_K) \wedge \sigma_i(R_i, R_i') \wedge R_i \neq R_i' \rightarrow \text{event}(T, R_1, \ldots, R_i', \ldots, R_K) \quad \text{(TP)}$$

$$w_{\text{ER}} : \text{event}(T, R_1, \ldots, R_K) \wedge \sigma_t(T, T') \wedge T \neq T' \rightarrow \text{event}(T', R_1, \ldots, R_K) \quad \text{(ER)}$$

Rule RP is role-propositional and assumes roles are independent, combining confidence levels from sensors for each role with a simple conjunction. Rule EP is event-propositional; it penalizes incompatible combinations of categories (Section 5.3.3). Rule TP is time-propositional, propagating confidence across similar events in the same time step. Rule ER is event-relational; it is a variant of Rule TP that crosses time steps.

5.4.1 Partial-Grounding. Rules TP and EP are templates; partial-grounding is the process of setting constant values for each template parameter, e.g., i in $w_{\text{TP}}(i)$. We do partial-grounding as follows: Create a copy of Rule TP for every role $i \in 1, \ldots, K$ to learn separate weights for each role. Using category sets $\chi_1, \ldots, \chi_K$ (Section 5.3.3), create a copy of Rule EP for every 4-tuple in the following set:

$$\left\{(i, j, \ell, \ell') \text{ with } (i, j) \in [1, \ldots, K]^2,\ i \neq j,\ (\ell, \ell') \in \chi_i \times \chi_j\right\}$$

5.4.2 Collective Inference. To illustrate how rules combine for collective inference, in Figure 8 we present a small, notional fragment of a CERF model. RP rule ϕ_1 allows sensor output in s_1 for the 2nd role (i.e., r_2) to influence the 2nd role of e_1. EP rule ϕ_2 also influences role 2 by preferring compatibility between it and the 4th role. TP rule ϕ_3 is satisfied if e_1 clusters with another similar event e_2 that varies with respect to role 4. ER rule ϕ_4 is satisfied if event e_2 continues in time step 2. Sensor outputs used in any rule can have a global effect on the predicted CEN via this chain of dependencies.

Table 2: Event roles in the GT CEN

#	Role (Variable)	Label Set
1	Class of attack (C)	$\mathcal{L}_1 = \{\texttt{Phish}, \texttt{Malware}\}$
2	Job category (J)	$\mathcal{L}_2 = \{\texttt{Acquisitions}, \ldots, \texttt{Training}\}$
3	Location (L)	$\mathcal{L}_3 = \{\texttt{AK}, \texttt{AL}, \ldots, \texttt{WI}\}$
4	Grade/level (G)	$\mathcal{L}_4 = \{1, 2, 3, 4, 5, 6, 7, \texttt{C}\}$

For example, if event e_3 is unlikely according to sensor data, that can cause CERF to change its predicted roles for event e_1.

6 EVALUATION

We evaluate CERF on a real corporate database of cyberattacks. Our goals are to evaluate the following:

- *Accuracy*: measured using AuROC and lift (Section 6.3)
- *Partially-observed events*: their effect on accuracy (Section 6.4)
- *Rules*: their relative contributions to accuracy (Section 6.5)
- *Scalability*: running time of CERF (Section 6.6)

Predicting well-differentiated events is another goal. Sensors produce one role each, so CERF's output is K times more detailed. This factor of improvement impacts the cost of solving Equation 1, but we show CERF scales well to $K = 4$ (Section 6.6).

6.1 Data

We describe the GT CEN collected from internal organization records, and the input SGs collected from a variety of external data sources.

6.1.1 A Real Cyberattack Event Network. To evaluate CERF, we use a GT CEN with nine months of actual attack events against a large US corporation. The GT is provided by the Cyberattack Automated Unconventional Sensor Environment (CAUSE) project,[1] and is available through agreement with the US Intelligence Advanced Research Projects Activity (IARPA). The GT focuses on significant attacks that current network defenses do not stop; it excludes spam email, routine network scans and other low-impact events. We filter GT further to events in which each label appears at least twenty times in the nine months. Although this GT is not openly available, it is representative of attacks against similar organizations in the same time period. It would be reasonable to extend our results using a data set for a similar organization.

As described in Section 5, the *event* predicate has $K + 1$ terms. The first is time step T identifying the date. We list the remaining K roles, their logical variable names, and their label sets in Table 2. Role 1 identifies the class of attack: *phishing* events are emails containing malicious attachments or links, and *malware* events are malicious applications discovered on computer hosts. Roles 2-4 give details about the victim of the attack. We represent each GT CEN event as a grounding of $\text{event}(T, C, J, L, G)$ with truth value 1. Similarities over labels are simple predefined measures of similarity (see Section 5.3.2), e.g., $\sigma_3(\texttt{MD}, \texttt{VA}) = .75$.

6.1.2 Cross Validation. We split the nine months of GT $M = m_1, \ldots, m_9$ into three sets (g_1, g_2, g_3) by month: $g_i = \{m_j \in M \mid$

[1] https://www.iarpa.gov/index.php/research-programs/cause

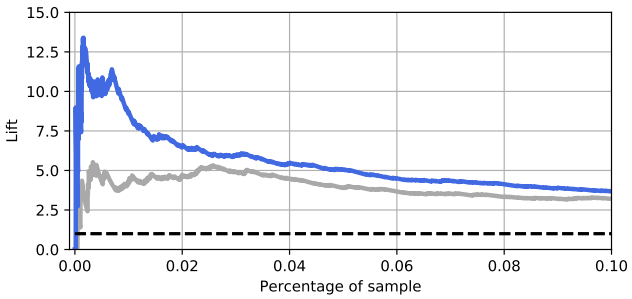

Figure 9: Lift of CERF (blue, top) and the baseline (grey).

$(j-1) \bmod 3 + 1 = i\}$. We use the three sets for training sensors, training CERF, and testing CERF, respectively. We define the sets this way so each is representative of the entire period, to limit over-fitting caused by correlations across sets or caused by training CERF on the same events used to train its inputs (the sensors), and to have contiguous periods to train and test Rule ER.

6.1.3 Sensor Graphs. We use one SG for each of the $K = 4$ roles. For each SG, we represent an edge between label R and time T with $s_i(T, R)$. For example, the truth value of ground atom $s_2(\texttt{Jan1}, \texttt{Acquisitions})$ represents our confidence that employees with job category `Acquisitions` will be attacked on `Jan1`. To generate the SGs, we train discriminative, multi-output classifiers on projections of g_1, conditioned on features extracted from the following original data sources: Twitter; the Global Database of Events, Language, and Tone (GDELT)[2] [10]; Open Threat Exchange; Wikidata; and Global Vectors for Word Representation (GloVe) [16]. The sensors produce SG outputs for day $T = t$ using evidence from $t - 7$, i.e., seven-day forecasts. CERF uses any time step given in a SG, whether past, present, or future. We refer to Okutan et al. for further details about how we use these sources [14, 15].

6.2 Systems

Our final CERF implementation used in the evaluation uses a subset of the possible partially-ground rules: We ground Rule EP on the following pairs of terms: (T, L), (C, L), (J, G), and (L, G), which we found balances accuracy and efficiency. We use day of week as categories for dates and treat all other labels as singleton categories. The final model has 518 rules with weights trained on g_2.

We compare CERF with a baseline assuming roles are distributed independently within events. It uses Rule RP, plus the simple prior, and we train it on g_2. Due to the sparseness of the t-norm used in PSL conjunctions, we actually do this join outside PSL as a normal product, then load the results into PSL.

6.3 AuROC and Lift

We calculate AuROC by comparing soft truth values from CERF and the baseline with true events in g_3. AuROC is .76 for the baseline, and .78 for CERF, confirming CERF's over-all improvement to accuracy, but we are especially interested in the rate of false positives among high-confidence predictions, as analysts are only able to review a small number of predictions every day. Lift is a better measure than AuROC for this. In Figure 9 we show the lift of CERF (blue, top) compared to the baseline (grey). Applying a threshold on truth value predicts a sample of the whole population of some size $\leq nu$ (horizontal axis of Figure 9). The lift of a detector is the ratio of its precision in that sample to expected precision (percent of the total population that is positive). For high-confidence predictions, CERF has up to double the lift of the baseline, indicating that analysts using CERF recommendations will have fewer false positives, compared to using sensors alone.

[2]https://www.gdeltproject.org

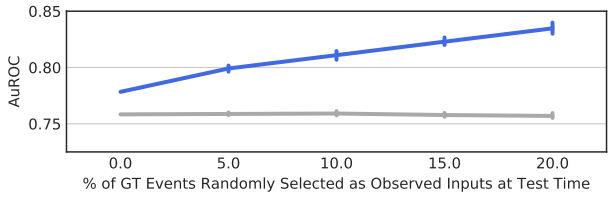

Figure 10: AuROC (95% CI) of CERF (blue, top) and the baseline (grey) as partial observation varies from zero to 20%.

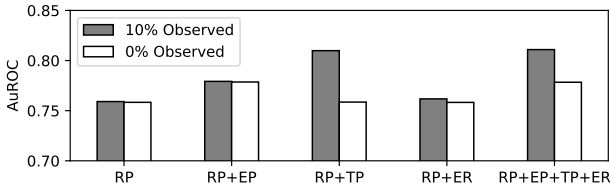

Figure 11: AuROC of CERF (RP+EP+TP+ER); the baseline Rule RP; and rules EP, TP, and ER.

6.4 Partially-Observed Events

Analysts learn of some attacks as they occur. To measure our ability to leverage these *partially-observed* events, we remove a random sample of events from test set g_3 (so later accuracy measurements ignore them) and add them to the inputs of each model with truth value 1.0 (see Section 5.3), without retraining the models. In Figure 10, we measure AuROC for both systems as we vary the size (as a percentage of g_3) of the random sample. For each setting, we repeat with five different random sets of observed events. For zero observed events, we do not measure repeatedly as results are constant. The baseline lacks rules crossing events, so its accuracy changes little with observed events. CERF uses its event-crossing rules (TP, ER) to increase accuracy with the size of the observed set. This suggests that when some attacks are known, CERF can rapidly alert analysts to other likely attacks.

6.5 Rule Contributions

Rule RP is the baseline and necessary to make predictions; without it, the rules could be trivially satisfied by predicting no events. To understand the contributions of the remaining three rules, we enable each in combination with RP. We use two settings for partially-observed events: one at 10% (results averaged over five random

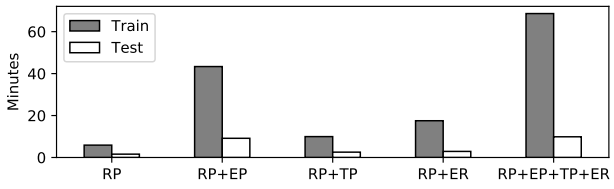

Figure 12: Running time of CERF (RP+EP+TP+ER), the baseline Rule RP, and the three remaining rules.

sets), and another with 0%. We plot AuROC in Figure 11. Compared to the baseline, Rule EP is more accurate, but it is insensitive to observed events – as expected, because EP focuses on event composition. For both cross-event rules (TP and ER), without observed events their accuracy is similar to the baseline. This suggests that the benefits of the current similarity measures are small when no events are observed. However, with observed events their accuracy increases substantially – especially with TP. CERF (RP+EP+TP+ER) combines all four types of rules and inherits their strengths.

6.6 Scalability

Increasing K, u, or n increases the cost of solving Equation 1 (Section 4.4), but for the GT CEN with $K = 4$, $u = 5600$, and $n = 269$ (over 1.5M RVs), CERF learns the model and infers events in just 69 and 10 minutes, respectively. Cost also varies by which of the $p = 518$ rules are enabled. In Figure 12 we plot running times with different subsets of rules. Rule EP contributes the longest running time because it acts as a template with more partial-groundings than the other rules (Section 5.4.1). We ran all experiments on a machine with 16 2.67GHz Xeon cores and 256GB RAM.

7 CONCLUSION

We introduce the new event-relational structured prediction model to solve the sensor fusion and structured prediction problems jointly. We apply the model to the problem of predicting cyberattack event networks. We evaluate CERF, the resulting system, on a CEN with nine months of real cyberattacks against a US corporation. We confirm CERF predicts events more accurately than sensors alone. Results suggest high utility for cybersecurity analysts, especially if some actual attacks are already known. We anticipate expanding or refining the sensors, similarity functions, and categories used in the model would increase accuracy further.

ACKNOWLEDGMENTS

This work was done as part of the Exploiting Leading Latent Indicators in Predictive Sensor Environments (ELLIPSE) project. The authors thank the many ELLIPSE team members that contributed to this work. ELLIPSE is supported by the Office of the Director of National Intelligence (ODNI) and the Intelligence Advanced Research Projects Activity (IARPA) via the Air Force Research Laboratory (AFRL) contract number FA8750-16-C-0114. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, AFRL, or the U.S. Government.

REFERENCES

[1] Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2017. Hinge-Loss Markov Random Fields and Probabilistic Soft Logic. *Journal of Machine Learning Research* 18 (2017), 1–67.

[2] Stephen H. Bach, Bert Huang, Ben London, and Lise Getoor. 2013. Hinge-loss Markov Random Fields: Convex Inference for Structured Prediction. In *Uncertainty in Artificial Intelligence*. 32.

[3] Adam Dalton, Bonnie Dorr, Leon Liang, and Kristy Hollingshead. 2017. Improving cyber-attack predictions through information foraging. In *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 4642–4647.

[4] Shobeir Fakhraei, Bert Huang, Louiqa Raschid, and Lise Getoor. 2014. Network-based drug-target interaction prediction with probabilistic soft logic. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 11, 5 (2014), 775–787.

[5] Leo A. Goodman. 1961. Snowball sampling. *The annals of mathematical statistics* 32, 1 (1961), 148–170.

[6] Pili Hu and Wing Cheong Lau. 2013. A survey and taxonomy of graph sampling. *arXiv preprint arXiv:1308.5865* (2013).

[7] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. 2011. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research* 1, 1 (2011), 80.

[8] Pigi Kouki, Shobeir Fakhraei, James Foulds, Magdalini Eirinaki, and Lise Getoor. 2015. Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 99–106.

[9] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 282–289.

[10] Kalev Leetaru and Philip A. Schrodt. 2013. GDELT: Global data on events, location, and tone. *ISA Annual Convention* (2013).

[11] Qiang Liu, Shu Wu, and Liang Wang. 2015. Collaborative prediction for multi-entity interaction with hierarchical representation. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 613–622.

[12] Alex Memory, Angelika Kimmig, Stephen H Bach, Louiqa Raschid, and Lise Getoor. 2012. Graph summarization in annotated data using probabilistic soft logic. In *Proceedings of the 8th International Conference on Uncertainty Reasoning for the Semantic Web-Volume 900*. CEUR-WS, 75–86.

[13] Farzaneh Mirzazadeh, Siamak Ravanbakhsh, Nan Ding, and Dale Schuurmans. 2015. Embedding inference for structured multilabel prediction. In *Advances in Neural Information Processing Systems*. 3555–3563.

[14] Ahmet Okutan, Shanchieh Jay Yang, and Katie McConky. 2017. Predicting cyber attacks with bayesian networks using unconventional signals. In *Proceedings of the 12th Annual Conference on Cyber and Information Security Research*. ACM, 13.

[15] Ahmet Okutan, Shanchieh Jay Yang, and Katie McConky. 2018. Forecasting cyber attacks with imbalanced data sets and different time granularities. *arXiv preprint arXiv:1803.09560* (2018).

[16] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.

[17] Ponemon Institute. 2018. *2018 Cost of a Data Breach Study: Global Overview*. Technical Report. Ponemon Institute LLC.

[18] Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning* 62, 1-2 (2006), 107–136.

[19] Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A markov logic approach to bio-molecular event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*. Association for Computational Linguistics, 41–49.

[20] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of machine learning research* 6, Sep (2005), 1453–1484.

[21] Satosi Watanabe. 1960. Information theoretical analysis of multivariate correlation. *IBM Journal of research and development* 4, 1 (1960), 66–82.

[22] Ke Zhang and Konstantinos Pelechrinis. 2014. Understanding Spatial Homophily: The Case of Peer Influence and Social Selection. In *Proceedings of the 23rd International Conference on World Wide Web (WWW '14)*. ACM, New York, NY, USA, 271–282. https://doi.org/10.1145/2566486.2567990 event-place: Seoul, Korea.

[23] Shi Zong, Alan Ritter, Graham Mueller, and Evan Wright. 2019. Analyzing the Perceived Severity of Cybersecurity Threats Reported on Social Media. *arXiv preprint arXiv:1902.10680* (2019).