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, cyber-attacks

ACM Reference Format:

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;
while minimizing false positives. Differentiated predictions identify
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 clas-
sification 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
every 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 ob-
jects 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 gen-
erate 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).

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

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

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.
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 $D_{KL}(p||p')$ 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).

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
We define the models in the following four sections. We refer to this model as which uses a single sensor as input and predicts a single role of a incompatibility score for any assignment to .

Given between a predicted value of , technique commonly used in probabilistic models [9, 18]. Our definition problem. We capture the statistical dependency between the value for and the sensor output and .

Event-relational (h) with three time steps (dotted). Time-propositional rewrites (g).

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 with each function, and find optimal . See Section 5.4.2 for an extended example.

Figure 7: Selected role values from sets (L_1, L_2, 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 and with partitions for additional sensor outputs
- Vector replaces the single role
- Set replaces one for each role – replace
- Set \( \Phi = \{\phi_1, \ldots, \phi_p\} \) of feature functions replaces \( \phi \)

This event-propositional model fuses sensors and predicts 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 \( \phi_1, \phi_2, \phi_3 \) is \( r = (r_1, r_2, r_3) = (22, UDP, DOS) \). In Figure 7, we illustrate the three sets (\( L_1, L_2, L_3 \)), 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 \( \phi_4 : L_1 \times L_2 \times S \rightarrow \mathbb{R}_{\geq 0} \). 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 \( r \) in Example 4.2 is optimal with respect to functions \( \phi_1, \phi_2, \phi_3 \), that combination is improbable in this domain. Assignment \( r = (22, TCP, UAA) \) 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 (\( \phi_1, \phi_2, \phi_3 \)) and functions encoding compatible event composition (\( \phi_4, \phi_5, \phi_6 \)). The event-propositional model optimizes all functions in \( \Phi \) 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 \( \Phi \) may need to be weighted differently, e.g., to tune the balance between sensors and compatibility. We train a weight vector \( w \in \mathbb{R}^p_0 \) (see Section 5.2.1), associate one weight from the vector \( w \) with each function, and find optimal assignment \( r \in L_1 \times \ldots \times L_K \) as \( \arg \min_w w^T \Phi (r, s) \).

### 4.3 Extension: Time-Propositional

Although the event-propositional model gives the most likely composition of an event, represented by \( r \), it is limited to a single event. It also doesn’t indicate whether an event will occur. We will replace
the single event \( r \) with a more general binary encoding. In the encoding, an element of vector \( e = (e_1, \ldots, e_u) \) exists for each of \( u \) possible configurations of roles in \( L_1 \times \cdots \times L_K \) 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 \( \Phi : S \times \{0,1\}^u \to \mathbb{R}_{\geq 0} \) and defined over projections of events. We illustrate rewriting in Figure 6d-f, in which an edge annotated with a set \( y \subseteq \{1, \ldots, K\} \) refers to a feature function defined over an event \( e \), but using only a projection of \( e \)'s roles, i.e., \( \pi_y(e) \) using relational algebra.

Example 4.4. Logical atoms are a convenient representation for \( e \), 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 \((r_1, r_2, r_3) = (22, \text{TCP}, \text{UA})\) as \( e_1 = \text{event}(22, \text{TCP}, \text{UA}) \). Rule \( \phi_4 \) in Figure 6c places dependencies on \( r_1 \) and \( r_2 \); the equivalent function in Figure 6 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 \( e_2 = \text{event}(\text{gmail}, \text{MD}) \) (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 \( \phi_2 \) could add a dependency linking our confidence in \( e_2 \) with confidence in an additional similar event \( e_3 = \text{event}(\text{gmail}, \text{timesheet}, \text{VA}) \).

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

4.4 Extension: Event-Relational

To capture dependencies across time steps \( t_1, \ldots, t_n \), we replace \( s \) with vectors \( s = (s_1, \ldots, s_n) \). 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 event\((t_1, \text{gmail1}, \text{timesheet}, \text{VA})\). Function \( \phi_2 \) 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:

- \( K \) roles in each event
- \( u = |e| = |L_1 \times \cdots \times L_K| \) possible events in each time step
- \( n \) time steps
- \( \Phi \) feature functions

With the event-relational model, we find an optimal \( e \) as follows:

$$\arg \min_{\hat{e}} w \top \Phi(e, s)$$

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 \( e \) (see Section 4.3). We use the following mapping to produce the model in PSL:

- Roles \( \Rightarrow \) Logical terms (logical variables or constants)
- Events \( \Rightarrow \) Logical atoms
- Sensor output \( \Rightarrow \) 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 \( \Phi \) 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 \( x \) and \( y \) represent observed and inferred atoms, respectively: \( \arg \min_{y \in \{0,1\}^w} w \top \Phi(y, x) \). A high quality binary solution \( y \in \{0,1\}^v \) is possible by applying conditional probabilities rounding to the soft-valued output [1]. In CERF, \( x \equiv y \equiv e \), and \( v \equiv nu \), establishing equivalence with our objective in Equation 1. We learn \( w \) with maximum likelihood estimation [2].

5.2.2 Soft Truth Values. In addition to the binary solution, the soft-valued \( e \in \{0,1\}^{uw} \) 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:

- \( \text{event} \): 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 \( R \in L_i \) and \( R' \in L_i \)
- \( s_i(T, R) \): 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 \( \sigma_i \) 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 \( R_i \in L_\ell \) is a member of category \( \ell \in \chi_i \), where set \( \chi_i \) 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)
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 \( \text{event}(T, R_1, \ldots, R_K) \), i.e., any attack is individually improbable. The remaining CERF rules are as follows, where rules with parameterized weights, e.g., \( w_{TP}(i) \), are templates:

\[
\begin{align*}
\text{w}_{\text{RP}} & : \sum_{i \in \{1, \ldots, K\}} s_i(T, R_i) \to \text{event}(T, R_1, \ldots, R_K) \quad \text{(RP)} \\
\text{w}_{\text{EP}}(i, j, \ell, \ell') & : -\text{event}(T, R_1, \ldots, R_i, \ldots, R_j, \ldots, R_K) \land (R_i > \ell) \land (R_j > \ell') \quad \text{(EP)} \\
\text{w}_{\text{TP}}(i) & : \text{event}(T, R_1, \ldots, R_i, \ldots, R_K) \land \sigma_i(R_i, R_j) \land R_i \neq R_j \to \text{event}(T, R_1, \ldots, R_j', \ldots, R_K) \quad \text{(TP)} \\
\text{w}_{\text{ER}} & : \text{event}(T, R_1, \ldots, R_K) \land \sigma_i(T, T') \land T \neq T' \to \text{event}(T', R_1, \ldots, R_K) \quad \text{(ER)}
\end{align*}
\]

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

#### 5.4.1 Partial-Grounding

Rules \( \text{TP} \) and \( \text{EP} \) 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 \( \text{TP} \) 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 \( \text{EP} \) for every 4-tuple in the following set:

\[
\{(i, j, \ell, \ell') \mid (i, j) \in [1, \ldots, K]^2, i \neq j, (\ell, \ell') \in \chi_1 \times \chi_2\}
\]

#### 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 \( \phi_1 \) allows sensor output in \( s_1 \) for the 2nd role (i.e., \( r_2 \)) to influence the 2nd role of \( e_1 \). EP rule \( \phi_2 \) also influences role 2 by preferring compatibility between it and the 4th role. TP rule \( \phi_3 \) is satisfied if \( e_1 \) clusters with another similar event \( e_2 \) that varies with respect to role 4. ER rule \( \phi_4 \) 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.

### 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, 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.

<table>
<thead>
<tr>
<th>#</th>
<th>Role (Variable)</th>
<th>Label Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Class of attack (C)</td>
<td>( L_1 = {\text{Phish, Malware}} )</td>
</tr>
<tr>
<td>2</td>
<td>Job category (J)</td>
<td>( L_2 = {\text{Acquisitions, Training, Sales, IT, Other}} )</td>
</tr>
<tr>
<td>3</td>
<td>Location (L)</td>
<td>( L_3 = {\text{CA, NY, DC, FL, TX}} )</td>
</tr>
<tr>
<td>4</td>
<td>Grade/level (G)</td>
<td>( L_4 = {1, 2, 3, 4, 5, 6, 7, 8} )</td>
</tr>
</tbody>
</table>

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: \( g_i = \{m_j \in M \mid j \leq i\} \).

---

We calculate AuROC by comparing soft truth values from CERF and the baseline with true events in \( g_i \). 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 \( nu \) (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.

### 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 (ELLIPE) project. The authors thank the many ELLIPE team members that contributed to this work. ELLIPE 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


[21] Satoshi Watanabe. 1960. Information theoretical analysis of multivariate correla-

