Abstract—Intrusion Detection Systems (IDS) warn of suspicious or malicious network activity and are a fundamental, yet passive, defense-in-depth layer for modern networks. Prior research has applied information fusion techniques to correlate the alerts of multiple IDSs and group those belonging to the same multi-stage attack into attack tracks. Projecting the next likely step in these tracks potentially enhances an analyst’s situation awareness; however, the reliance on attack plans, complicated algorithms, or expert knowledge of the respective network is prohibitive and prone to obsolescence with the continual deployment of new technology and evolution of hacker tradecraft. This paper presents a real-time continually learning system capable of projecting attack tracks that does not require a priori knowledge about network architecture or rely on static attack templates. The intrusion projection system is framed as part of a larger information fusion and impact assessment architecture for cyber security.

Keywords: intrusion projection, VLMM, cyber fusion

I. INTRODUCTION

New IT models, business requirements, and operational considerations continue to drive the trend toward highly interconnected and technologically converged networks [1]. Proprietary information processing solutions and stovepiped databases are giving way to unified Enterprise Resource Planning (ERP) systems [2], for example, raising the potential impact of just a single well planned and executed network intrusion, data theft, or denial of service (DoS). While trained analysts are responsible for day-to-day security and operation of government agency or corporate enterprise networks, the monitoring and intrusion detection tools they rely on must evolve to provide the higher-level situation awareness needed when faced with large multifaceted networks and motivated adversaries.

Unaided, a human cannot efficiently comprehend the high volume of raw IDS alerts from a large network [3], [4], assess impact, and accurately project virtual intrusion trajectories. Multiple internet gateways, wireless devices, and VPN remote connectivity offer the promise of redundancy and increased productivity, but exponentially increase the number of possible attack vectors [5], [6]. An attack waged using IP address spoofing, distributed botnets, and/or automated exploitation tools, capable of targeting multiple layers of the protocol stack [7]–[11], further complicates the correlation of alerts generated from sensors at disparate perimeter gateways.

Information fusion, conversely, holds the promise of correlating those alerts belonging to the same multi-stage attack [12]; automatically assembling pieces of the analyst’s puzzle and building situation awareness, obfuscated intentionally by the attacker, or indirectly by the complicated network architecture. It is not the intention of this overall research effort to remove the analyst from the loop by taking automated defensive actions or reconfiguring firewalls. Rather, our system sifts through the overwhelming amount of data in the form of intrusion alerts, recognizes and correlates important events, projects likely future actions in real-time, assesses the impact of current and projected attacker actions, and presents this information to the analyst.

The attack projection system presented as the focus of this paper is predicated on the sequential definition of alerts comprising attack tracks as correlated by the information fusion engine, and not pre-defined attack plans. The proposed approach shall reveal plausible futures not necessarily obvious, or not previously considered by the analyst. Finally, our projection relies on relatively non-computationally-intensive algorithms, facilitating real-time performance as the system scales to meet the demands of these enterprise networks.

II. OVERALL ARCHITECTURE

This section provides a brief illustration of the overall cyber security architecture (Figure 1), giving perspective to the proposed projection system.

A. Intrusion Detection

Tracking and projecting cyber attacks relies on the accurate and timely reporting of suspicious activity. Host and network based IDSs generate alerts when they observe suspicious, although not necessarily malicious, activity [13]–[15]. Most IDSs are classified as knowledge-based (e.g., pattern inspection, deep packet inspection) or behavioral-based (e.g., statistical anomaly). It is not in the scope of this research to design a better IDS, but to derive understanding from the observables of attacker actions.

While the breadth of included information and formatting of an IDS alert varies, fields indicating the destination IP address, description of the alert cause, network protocol, and time are common and naturally important. Figure 2 shows a
sample alert in XML format. The format of other IDS alerts are similar [13]–[15].

```
<Alert>
  <Track_ID>1149005317059</Track_ID>
  <Description>Multistage Background Recon</Description>
  <Certainty>0.54545454</Certainty>
  <Alert>
    ... 
    <Sensor_ID>snort212</Sensor_ID>
    <Category>Recon_Scanning</Category>
  </Alert>
```

**Figure 2.** IDS Alert XML Example

**B. Alert Correlation**

Several methods for alert aggregation [16], [17] and alert correlation [12], [18]–[21] have been proposed to solve the problem of overwhelming alert volume. Alert correlation essentially finds IDS alerts that are related and organizes them into ordered collections. These collections can be considered to represent the virtual trajectories of cyber attacks.

The alert fusion engine, Information Fusion Engine for Real-time Decision Making (INFERD) [12], is a keystone to our overall architecture. Based on sensor data and a priori models, INFERD dynamically generates, evolves, and evaluates hypothesis on the current state of the environment, performing information fusion at levels zero, one, and two to provide real-time situation assessment. At L2, INFERD uses guidance templates to instantiate acyclic directed graphs used for understanding the current state of the network.

In practice, the follow-on systems, including projection, consider INFERD’s acyclic directed graphs as ordered sequences of alerts called attack tracks. Each track represents an unfolding multi-stage attack sequence. Figure 3 shows the additional information appended to each alert by INFERD. Specifically, note the `track_ID` field that associates an alert with an INFERD track number. The `category` field is also assigned by INFERD, indicating which of several general categories best matches this alert (reconnaissance, intrusion, DoS, Data Exfiltration, and so on).

```
<Alert>
  <Track_ID>1149005317059</Track_ID>
  <Description>Multistage Background Recon</Description>
  <Certainty>0.54545454</Certainty>
  <Alert>
    ... 
    <Sensor_ID>snort212</Sensor_ID>
    <Category>Recon_Scanning</Category>
  </Alert>
```

**Figure 3.** INFERD Track XML Example

**C. Impact Assessment**

Threat and Impact Assessment (TIA) differentiates those attacks with cause for immediate concern from benign or lower priority threats. The Virtual Terrain Assisted Impact Assessment for Cyber Attacks (VTAC) is used in our architecture to analyze the correlated alerts and determine the impact of progressing attacks to network services and users [22].

The VTAC impact assessment algorithm uses a graph-based virtual terrain model and combines impact assessments of damage caused by the attacks. Impact scores are defined for hosts, services, users, and the subnet or network as a whole. For example, the damage done to a host with respect to its services, importance, and asserted exposures is defined as the host impact. These assessments naturally focus attention on those attacks with the most deleterious impact.

**D. Projection**

While the INFERD architecture continues to improve its performance at the L0, L1, and L2 fusion levels, the projection of cyber attacks could be considered an L3 challenge [23]–[25]. Past work in the area of prediction or projection has relied on manually defining the significance of network services and servers, or matching ongoing attacks against predefined attack templates.

For example, Qin and Lee [26] proposed one of the first high level attack projection schemes, which investigated the use of a combined alert correlation and attack prediction system. Their work identified attack sequences by applying Bayesian networks to IDS alerts. Performing plan recognition at this high level required the creation of attack plans by domain experts, thus relying on matching of ongoing attacks against static a priori models. Generally, it is challenging to create attack plans that are broad enough to capture the possible scope of attacker behavior, while specific enough to provide useful or previously unconsidered futures. In addition, a more recent approach by Mehta et. al. [27] employs pre-constructed probabilistic attack graphs.
Holsopple et al. [28] used additional a priori information to project plausible futures, relying on knowledge of the network topology, a mapping between services and host computers, and observed attack sequences. The resulting threat score from this work, and similarly from work by Arnes et al. [29], indicated which entities of the network were most likely to become the next targets of an attack.

The work done by Fava et al. [22] introduces the idea of making predictions based on the correlated alerts as individual attack tracks and not a priori information, decoupling the tasks of network modeling and attack behavior modeling. This context-based model derives from the work originally applied to compression and prediction [30], and from attack graph prediction by Li et al. [31]. It is pioneering in that the projection is based on an attacker’s behavior as learned by the system.

This paper extends Fava’s work [22] by implementing the training and prediction algorithms simultaneously in real-time for an arbitrary number of alert field definitions. The system is also continuously learning, instead of pre-trained, confirming or rejecting previous projections as new attack steps are observed. By projecting these plausible future actions from attack patterns, the system reveals potential network vulnerabilities that may be imminently exploited, complimenting the threat scores by Holsopple et al. [28].

### III. Real-time Projection

The fusion process of extracting behavior patterns starts by transforming ordered collection of IDS alerts into simple sequences of symbols. Incoming alerts are parsed according to their track-based XML tags, mapping the values of selected tags to alphabet symbols, and building the corresponding attack tracks and suffix trees. The suffix tree model is a fused representation of the observed tracks, integrating attacker behaviors into a single model. Separate suffix trees, symbol mappings, and track histories are used internally for each uniquely defined IDS alert field. Variable Length Markov Models (VLMM) are then used to create projections for each track based on their unique histories and the suffix tree. This section explains the algorithms used for these steps.

#### A. Alphabet

Our system does not require pre-filtering alerts or pre-defining the list and meaning of alphabet symbols. Instead, user-defined alert fields are independently mapped to a symbol in their respective alphabet, \( \Omega \), forming a symbol space. Integers are used as symbols, a computationally efficient approach allowing very large alphabet sizes. The observation of a new alert field value automatically creates a new symbol with its corresponding meaning in a hash map.

Attack tracks are built in parallel for each alphabet definition in the symbol space, representing the same multi-stage attack from the perspective of attack description, destination subnet, etc. When an alert arrives, the system looks up the symbol for each alert field and appends the symbol to the sequence of \( n \) alerts \( s = \{ x_1, x_2, ..., x_n \} \) where \( x_i \) belongs to the respective alert field alphabet \( \Omega \) and is not equal to \( x_{i-1} \). Creating an attack track with repetitions has the potential to pollute the suffix tree and a projection result equal to the event that just occurred is of questionable usefulness.

The alert field type intrinsically determines the size of the resulting alphabet. For example, the INFERD generated category is a mapping of the detailed alert description field to a general category, and therefore the category alphabet generally has fewer symbols than the description alphabet.

#### B. Suffix Tree Training

Suffix trees for each alphabet definition in the symbol space are trained in parallel whenever a new alert arrives, integrating the latest behavior of the attack with the tree. The suffix tree training algorithm was originally motivated by the work of Begleiter, El-Yaniv, and Yona [32], and modified by Fava [22] to take a set of finite length sequences instead of a single long sequence of observations. Fava’s finite length sequences have defined start and end of sequence characters, however the real-time arrival of alerts requires a new algorithm, training the suffix tree with partial sequences instead of a finite completed sequence.

For example, a tree built on a single sequence ‘1,2,2,1,2,1,0’, where ‘0’ defines the end of sequence character, is shown in Figure 4. Consider the symbols ‘1’ and ‘2’ as representing alert descriptions WEB-IIS nsiislog.dll access and WEB-MISC Invalid HTTP Version String, respectively. Edges are weighted with the number of times the suffix tree is traversed through that branch. For example, ‘1,2,1,0’ happened only once and ‘1,2’ happened twice in the sequence.

![Figure 4. The suffix tree for a finite sequence ‘1, 2, 2, 1, 2, 1, 0’.](image)
Now consider the construction of the tree one symbol at a time, where the end of the track is unknown. The sequence ‘1, 2, 2, 1, 2, 1’ will be trained as ‘1’, ‘1, 2’, ‘1, 2, 2’, ‘1, 2, 2, 1’, and so on, as the track grows. First, the modified algorithm must properly increment edge weights, considering only the last symbol as a new addition, instead of the entire sequence; otherwise, it would unfairly weight prior suffixes. Second, with no end of sequence characters, child edge weights do not necessarily sum to the parent edge weight, effectively creating an unbalanced tree.

Figure 5 shows the real-time constructed version of the sequence in Figure 4, but without the end of sequence character (‘0’). Note that the path ‘ROOT’ to ‘1’ has an edge weight of three, while the summation of its children node edge weights is only two. An end of sequence character would have created another child node with edge weight of one, balancing the tree.

The probabilities of each prediction up to $n^{th}$ order are then blended using escape probabilities discussed in further detail by Begleiter et al. [32]. The blended probability is the weighted sum of $P^n(x)$:

$$P(x) = \sum_{o=-1}^{m} w_o \times P^o(x) \quad (3)$$

where $m$ is the longest match for the observed sequence $s$ in the suffix tree. In other words, the algorithm continues calculating and blending Markov model predictions until the length grows such that the sequence under consideration cannot be found in the tree.

Probabilities for a $0^{th}$ order model ($P^0$) are the normalized counts of all possible characters in the alphabet, representing the relative probability of one character occurring over another based on the child edge weights directly from the root node on the suffix tree. A minus one order model is used such that $P^{-1}(x) = 1/|\Omega|$ for all $x \in \Omega$, giving equal weight to each symbol in the alphabet, and is explored in depth by Bell et al. [30].

Fava’s work empirically tested several methods of computing escape probabilities and was not able to identify any particular method as superior [33]. This work implements four methods of calculating a context’s escape probability, however, $1/(C + 1)$ is used as default, where $C$ is the accumulated edge frequency of the current context’s children.

Finally, the blended probability for each of the symbols in the symbol space are sorted in descending order. A projection is considered correct if the actual event’s symbol ranks high in the sorted set of predictions prior to its occurrence. From a security analyst’s perspective, high ranking may be in terms of the actual rank or the percentile, for example. We consider the top three predictions pertinent and reasonable, and thus use it to report the performance of the proposed system.

D. Prediction of New Symbols

Without pre-training, this self-learning system will inevitably observe new symbols - a symbol not previously in the universe of known symbols. The proposed system attempts to predict the occurrence of these new symbols by adding a special symbol representing observations of new events in each alphabet space. When a symbol is first observed by the model, the suffix tree will be updated not only with a newly created symbol, but also with the special symbol indicating a new event has occurred in the corresponding alphabet space.

Consider the example sequence ‘1, 2, 2, 1, 2, 1’ used previously. The suffix tree would be updated by sequences ‘1’, ‘1, 2’, ‘1, 2, 2’, ‘1, 2, 2, 1’, and so on, as the events occur. Let ‘0’ be the special symbol representing observations of new events. When ‘1’ arrives, the symbol ‘1’ has not previously been observed, so the tree will be trained with the sequence ‘1’ and also the sequence ‘0’. When the next alert arrives, ‘2’ is a new symbol so the tree will be updated with the sequence ‘1, 2’ and also with the sequence ‘1, 0.’ When ‘2’ arrives again as the next alert, it is not a new symbol and therefore the tree is trained only with the sequence ‘1, 2, 2.’ Once new symbol
behavior is integrated into the suffix tree model, the VLMM predictor is able to predict that a new symbol is about to occur just as it would any other symbol. The resulting suffix tree that accounts for the special symbol ‘0’ is shown in Figure 6.

Figure 6. The suffix tree for a real-time sequence ‘1, 2, 2, 1, 2, 1’, training with the special symbol ‘0’ representing new events.

IV. EXPERIMENTS AND RESULTS

A. Experiment Design and Data set

Typical work on detection and classification uses a separate pair of training and test data sets. Recognizing that cyber attacks could evolve in a much faster pace than the training of systems, this work continuously updates the suffix tree model and performs predictions as events unfold for simultaneous ongoing attack tracks. Note that the relative computational simplicity of the suffix tree training and VLMM prediction is critical to achieve real-time performance at high alert volumes. Correlated IDS alerts in an experiment data set will be processed by the real-time system in chronological order, one alert at a time. As each alert arrives, the suffix tree is updated, new predictions are made, and performance measures are collected. This approach facilitates investigation into the adaptive qualities of the system to new attack scenarios and more closely resembles a system that may be deployed in the real world.

In terms of choosing the data for testing, other research in cyber security intrusion detection has relied on data sets such as those from MIT Lincoln Lab [34] [35], KDD Cup 99 [36], and Defcon [37]. Although in a real-life system the collected alerts could be correlated by the fusion system, these data sets do not include a ground truth that specifies which malicious activities are executed as part of the same multi-stage attack, affecting confidence in the projection results if used for testing. Instead, this work utilizes several new experimental data sets crafted in a virtualized or simulated environment taking into account the need for ground truth.

Results from one data set are reported in this paper. Attack data is generated by scripted multi-stage attacks performed on a virtualized network, modeled after a real-life enterprise architecture. The virtual network contains seven internal subnets (each having a number of user address spaces), 22 external servers, and 24 internal servers. Example servers include IIS Web servers, MS Exchange Servers, FTP and VPN servers, running on Linux and various Windows OS’s. The data set consists of five sets of attack scenarios, producing a total of 19,908 alerts and 2,559 attack tracks. The attack scenarios range from CGI Overflow, Data Exfiltration, Phishing, to Denial-of-Service, and differ in the attack targets. Alert messages were produced by Snort, Dragon, Apache, and IIS. A real-world operational system should have a data alignment pre-processing component that homogenizes alert messages produced by different types of IDSs [21], [38]. A homogenization step was not performed on the data set used for this research. Instead, only Snort alerts were used, reducing the data set to a total of 1482 attack tracks comprising 10425 alerts.

Correlated alerts are sent to the system in the order of their time stamps, simulating the arrival of real-time intrusion events. Alerts are converted to symbols, the model constructed, and per track predictions made in real-time on this per-alert basis. No pre-filtering of alerts was performed. Alerts are naturally filtered per alert field definition as they are converted to symbols and only added to tracks if the symbol is not a repetition of the previous.

Table I shows part of an attack track in this data set, contrasting the observed attack description with the predicted description at each step. Note the prediction of an attack goal step with description WEB-MISC apache directory disclosure attempt in the fourth step. Although this prediction, which is the one with the highest probability prior to Step 4, is incorrect when compared to the observed ground truth at Step 4, it occurs shortly thereafter as observed in Step 6, affirming our plausible futures assertion. Note that steps four and six are both preceded by WEB-MISC Invalid HTTP Version String, yet the predictions for each are different, showing that the VLMM indeed considers the unique histories beyond the immediately previous step.

B. Projection Accuracy

Table II shows the projection accuracy averaged over the entire data set with attack description, attack category, network protocol, and destination subnet used to define the symbol space. The projection accuracy is the percentage of symbols occurring next that fall within a projection set, comprised of symbols with the highest probabilities according to the blended VLMM model. Therefore, the Top-3 row considers a projection correct if the observed event was one
of the top three predictions with the highest probability. The choice of ‘three’ is arbitrary and can be changed to any reasonable number to reflect the number of plausible futures an analyst can evaluate.

Numbers shown in the Ideal row represent the best real-time theoretical prediction accuracy possible for each alphabet, given the universe of known symbols at the time of each prediction. In other words, the incorrect predictions for this ideal case are due to a new symbol appearing as the next event, and therefore could not be predicted using this real-time algorithm with no a priori information or training sets. The numbers shown in the Ideal row serve to contrast the performance achieved by the proposed system when using different symbol space definitions. The Predictions row in Table II gives the total number of predictions that the system made for each alphabet definition. Since only transitions are reflected in attack tracks, and not repetitions, these numbers are less than the total 10,425 original alerts. The row Symbols shows the unique number of symbols under each alphabet definition.

Table II

<table>
<thead>
<tr>
<th>Time</th>
<th>Category</th>
<th>Description (Observed)</th>
<th>Description (Predicted)</th>
</tr>
</thead>
<tbody>
<tr>
<td>14:22:35</td>
<td>Goal_Dos</td>
<td>WEB-MISC apache directory disclosure attempt</td>
<td>WEB-MISC apache directory disclosure attempt</td>
</tr>
<tr>
<td>14:23:59</td>
<td>Intrusion_Other</td>
<td>(http_inspect) BARE BYTE UNICODE ENCODING</td>
<td>(http_inspect) BARE BYTE UNICODE ENCODING</td>
</tr>
<tr>
<td>14:24:42</td>
<td>Intrusion_Root</td>
<td>WEB-MISC Invalid HTTP Version String</td>
<td>WEB-MISC Invalid HTTP Version String</td>
</tr>
<tr>
<td>14:25:01</td>
<td>Intrusion_Root</td>
<td>WEB-MISC apache directory disclosure attempt</td>
<td>WEB-MISC apache directory disclosure attempt</td>
</tr>
<tr>
<td>14:26:29</td>
<td>Intrusion_Other</td>
<td>(http_inspect) BARE BYTE UNICODE ENCODING</td>
<td>(http_inspect) BARE BYTE UNICODE ENCODING</td>
</tr>
</tbody>
</table>

Numbers shown in the Ideal row serve to contrast the performance achieved by the proposed system when using different symbol space definitions. The Predictions row in Table II gives the total number of predictions that the system made for each alphabet definition. Since only transitions are reflected in attack tracks, and not repetitions, these numbers are less than the total 10,425 original alerts. The row Symbols shows the unique number of symbols under each alphabet definition.

Table II

<table>
<thead>
<tr>
<th></th>
<th>DESC.</th>
<th>PROTOCOL</th>
<th>DEST.</th>
<th>CATEGORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ideal</td>
<td>75.4%</td>
<td>100%</td>
<td>71.1%</td>
<td>94.6%</td>
</tr>
<tr>
<td>Predictions</td>
<td>1596</td>
<td>321</td>
<td>121</td>
<td>1319</td>
</tr>
<tr>
<td>Symbols</td>
<td>51</td>
<td>4</td>
<td>33</td>
<td>9</td>
</tr>
</tbody>
</table>

Note the exceptional accuracy achieved with the uses of category (94.6% with respect to the ideal 99.2%) and protocol (rounded to 100%) symbol definitions. The accuracy achieved with the network protocol definition is expected as the symbol space is small - only four in the experiment data set. The nearly perfect projection using the category definition is promising, especially since it gives the analyst a reasonable scope of projection without worrying about the specific vulnerability the attacker will attempt to exploit.

The accuracies achieved with the uses of description and destination, though lower than the other definitions, are also encouraging given the relatively large number of symbols in the corresponding symbol spaces. Consider the case of description, a blind guess of three out of 51 (from the Symbols row in Table II) will give a prediction rate of $\frac{3}{51} \approx 6\%$, which is much smaller than 75.4%, not to mention that the ideal real-time case can only achieve 95.0%.

Figures 7 and 8 show the moving average prediction accuracies achieved by the system with respect to the total number of injected alerts. This set of results reveals how the real-time system performs as the model is trained with more and more alert data and how it fluctuates in the steady state. First, observe the initial transient periods where the system has not yet observed enough alerts to build an accurate model. After approximately 2000 alerts, most symbols in each symbol space have occurred in the data set and prediction accuracies rise toward the values shown in Table II.

The removal of repetitions and time correlation of the prediction accuracy charts across the alphabet definitions leads to what appears to be a longer transient region for protocol and destination subnet compared to description and destination. The moving window average starts with an assumption of zero percent accuracy, while the 100% protocol accuracy in Table II is a rounded result of taking the overall number of correct predictions divided by total number of predictions for that alphabet definition.

As expected, Figure 7 shows better accuracy for the generic category alphabet definition than its more specific description counterpart. Similarly, Figure 8 shows near perfect protocol
prediction after the transient period. The destination subnet prediction accuracy is also noteworthy, showing that the next targeted subnet is relatively harder to predict. This result reiterates the lower ideal prediction rate result for destination subnet shown in Table II, meaning that the overall attacker behavior exhibits a proclivity for targeting or progressing toward new subnets.

Figure 9 shows the percentage of the total number of symbols known to the projection software as the alerts are injected. The number of symbols generally levels off during the middle several scenarios, but the last scenario contributes new symbols to the destination subnet and description alphabets. This last scenario starts after ∼8,000 alerts have been observed and it introduces new specific attack methods, i.e., attack descriptions. As can be seen in Figure 7, prediction accuracy under description alphabet definition drops temporarily. After 1,000 additional alerts have been observed, the system performance rises again to around 85%-90% accuracy. This phenomenon suggests the adaptiveness of the proposed approach in the presence of new attacks. Note that the new attacks can be introduced by a combination of changes in network configuration and the discovery of new exploits.

It is likely, however, that the less impressive prediction rates shown for destination subnet in Figure 8 would continue to rise with additional data, and thus the results shown for this definition may still represent an early transient period. Note the dips in destination subnet accuracy shown in Figure 8 whenever a new symbol appears in Figure 9. New symbols continue to appear late in the data set.

Figure 10 shows the accumulated blended VLMM probabilities for the top three predictions using the destination subnet definition, contrasting this accumulated probability for the correctly versus incorrectly made predictions. The higher accumulative top-3 probability for correct predictions suggests that incorrect predictions occur when the plausible futures do not center on a relatively few number of symbols. This is especially true when using the destination subnet symbol space, as shown in Figure 10.

C. Prediction of New Symbols

The real-time implementation also attempts to predict that a never-before-seen attack is about to happen. The system does not predict exactly what new attack will happen; instead, it predicts that a new symbol will occur. In this experiment, the system is able to predict 8 out of 51 occurrences of new attack methods (description), increasing overall description prediction accuracy from 75.4% to 76.6%. While 8 out of 51 symbols does not sound as impressive as the prediction accuracies presented earlier, these results show promise that the system is able to not only adaptively train and predict attacks that have been observed, but also provide warnings of new attacks while not degrading prediction accuracy with false positives.

V. Conclusions

This paper proposes a method for the real-time projection of network intrusions based on information fusion and behavioral modeling of correlated intrusion observations. The projection of specific attack traits such as the vulnerability exploited or destination address provides an analyst with plausible futures, enabling manual intervention or processing by follow-on automated defensive systems.

The current projection assessment method, considering only the next step in an attack track to determine correctness, may also require revision. In keeping with the theme of
projecting plausible futures, a prediction could be considered correct if it occurs within some reasonable time window in the future, not just as the next event for a given track. The determination of this window, whether constant or dynamic, and its relation to the usefulness of the data presented to the analyst will need to be investigated.

Finally, it is expected that to derive maximum value from these predictions they will need to be interpreted at a much higher level, probably combined with results from other systems. The challenge lies in assessing the combination compatibility of multiple types of predictions, choosing the proper combination algorithm, and interpreting the results.

ACKNOWLEDGMENTS

This work is funded through the National Center for Multisource Information Fusion (NCMIF) grant under the technical supervision of AFRL/RFEA.

REFERENCES


