

Toward interface customization in intrusion detection systems

Lloyd Williams, Sean P. McBride, Robert St. Amant, and Peng Ning

Department of Computer Science

North Carolina State University

Raleigh, NC 27695

stamant@csc.ncsu.edu, ning@csc.ncsu.edu

ABSTRACT

Intrusion detection systems offer a new challenge for intelligent user interfaces research. An IDSs might identify tens of thousands of alerts that may indicate intrusions over a computer network in a single day. In a system that we are currently building, the user constructs hypotheses concerning actual attacks and attempts to verify these hypotheses via knowledge embedded in the system and information external to the system. Behavior-based customization holds some promise to alleviate the difficulty users face in this task.

Author Keywords

Intrusion detection

INTRODUCTION

It is commonly known that intrusion detection systems (IDSs) produce a high volume of alerts. Not only does the large number of alerts pose a complex environment for uncovering individual attacks, but the vast number of false positives and missed alerts makes the endeavor extremely challenging. Much research has examined facilitating the analysis of IDS data through correlation techniques [Ning et al., 2004]. These techniques implement approaches based on similarities between separate alerts or the relationships between prerequisites and consequences of known attack patterns.

The former approach is regularly used to determine a possible missed alert or a common connection between attacks, while the latter techniques can help users detect possible multi-staged attacks through the myriad of IDS alerts. These methods require specific predetermined attack scenarios to be encoded within the system prior to analyzing the alert data. Our current research works toward intelligent assistance in creating generalized attack

scenarios based on historical IDS alerts, real-time information sources, and users' expert knowledge.

In our system, hypothetical attacks are interactively generated by users in the form of interconnected data objects. Figure 1 gives a high-level overview of the system. Each data object signifies both specific and generalized features about a possible attack. The user specifies connections between these data objects and creates relationships between the features within the data objects. Additionally, users can add weights to data objects (signifying their importance) and connections (describing their strength). The creation of these weighted data object graphs are then used to create Bayes nets and regular expressions that can be used to detect future attacks.

Our work on interface customization is currently in the planning stages. We are pursuing two paths to behavior-based customization, one that deals with models of the process of constructing hypothetical attacks, the other with the process of querying external data sources for information relevant to given hypotheses. In the body of this paper we describe the system and its capabilities. We end with a discussion of how the system's effectiveness may be enhanced by customization based on observations of the user's behavior.

DATA OBJECTS

Intrusion detection is traditionally treated as an interpretation problem, in which the system generates and integrates information about potential attacks, with the user making queries to refine the information in the process of extracting a correct or plausible interpretation. Drawing on the lessons learned from direct manipulation, we take a more constructive view of the intrusion detection process. Our goal is to open up the system's internal information processing, so that the user can construct explicit hypotheses based on information the system provides, which both the user and system can then refine collaboratively.

In abstract terms, the user constructs data objects by identifying pieces of information from internal knowledge representations and external data sources that are relevant to the incremental construction of a hypothesis concerning a possible attack. When more than one data object has been

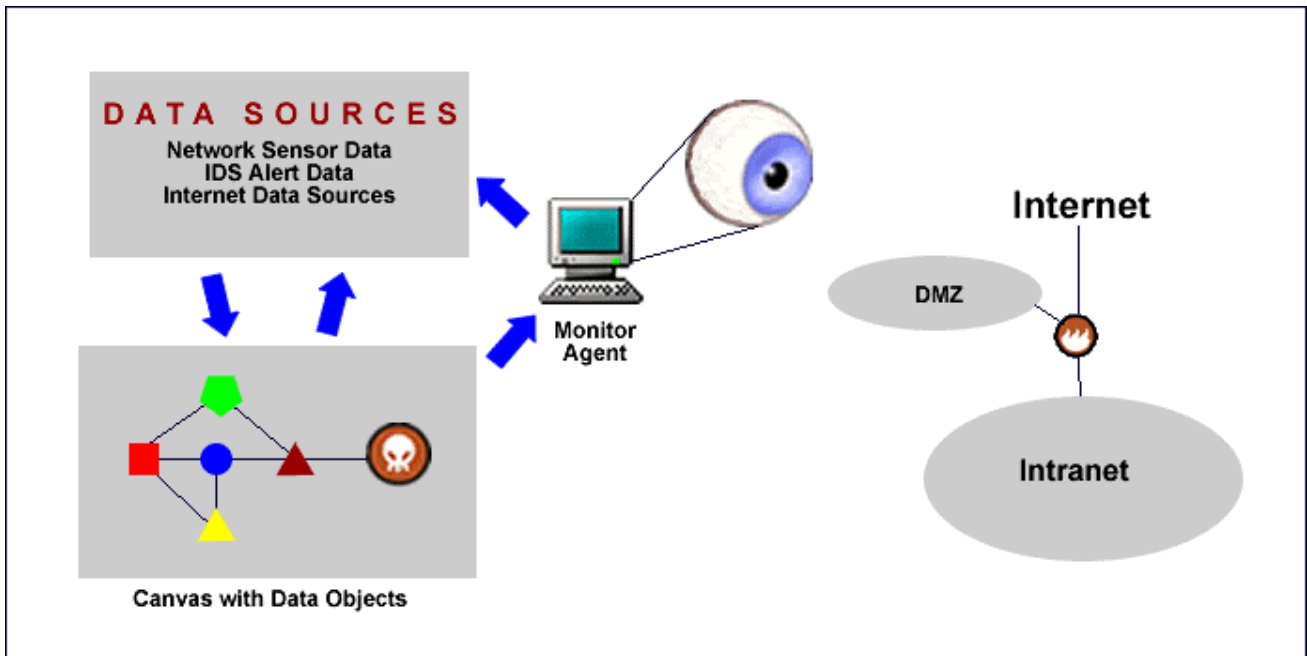


Figure 1. System overview. Data objects are selected from among the data sources. These data objects are then grouped and interconnected on the Canvas. The resulting graph can be used to create a Bayes net. Monitoring functions can then use this structure to identify future network activity that matches the attack in the graph.

created, the user can then specify hierarchical and relational connections between the data objects. The system acts in the role of an observer in this process, adding more information as the opportunity arises. The user can limit or expand the inferences that the system makes by adding constraints to the types of processing that the system is allowed to carry out.

Data objects are created by encompassing existing data objects in a bounding box or by dragging an entity from an information source to a canvas. These information sources include a list of user-filtered alerts, a list of system-filtered alerts, *whois* information, *tracert* data, and current Internet Storm Center information. As for their informational content, a data object can represent one or more “information nuggets”: alert features, specific alerts, hyper-alert facts and types, Internet Storm Center information, *whois* or *tracert* information. The canvas-captured data objects will appear as color-coded two-dimensional shapes depending on the type of data they contain. A data object can be created that combines information from more than one information source, and the system is designed to allow the addition of other information sources.

Alert features

Alert features are simply the individual data attributes associated with an alert, the most common of which are: Source IP Address, Source Port Number, Destination IP Address, Destination Port Number, and Time Stamp. When piecing together data objects that represent these different attributes, users are creating a definition for an alert that may not yet exist in any of the historical data.

Specific alerts

Given the situation where a system has been previously attacked and the system did not have a broad enough attack scenario knowledge base, users will build attack scenarios to catch similar recurring threats. The base IDS may have signaled most of the alerts associated with this attack, but the correct correlation of these alerts was the missing element. With this wealth of data, users can filter through the alert history and pull out a set of specific alerts. It is then possible to create data objects, which will encompass the features of the chosen alerts.

Hyper-alert facts and types

During causal correlation of alerts, prerequisites and consequences are used to chain what are called hyper-alert facts and types to form larger attack scenarios and identify possible attacks [Ning et al., 2004]. A hyper-alert type is defined as triple (*prerequisite*, *fact*, *consequence*) where *fact* is a set of alert attributes, *prerequisite* is a logical combination of predicates with all free variables in *fact*, and *consequence* is a set of predicates with all free variables in *fact* [Ning et al., 2002]. To construct hyper-alert types within our system users can encompass alert attributes within a data objects as representations of hyper-alert facts. These hyper-alert fact data objects can then be used, as an associated collection, to construct hyper-alert type data objects.

Internet Storm Center information

Internet Storm Center (ISC) information is current attack data measured on a global scale. The source for all this

information is publicly available at <http://isc.sans.org/> and is broken into three major components: current threat level, “Top 10” attacked ports and source IP Addresses, and current daily attack trends. Data objects derived from this information take on Boolean features representing statements like “the port in question appears in the top 5 attacked ports” or “the current threat level is *orange*”. During analysis, this data is gathered in real time directly from the ISC allowing the system to dynamically respond to new threats as they emerge.

Whois

When examining an individual alert, there is usually very little information that any user, expert or novice, can determine from the source IP address alone. However, when coupled with data stored in a *whois* database the IP address now takes on the properties of an individual, company, address, etc. Users can capture this information within a data object to provide this extra dimension to an attack scenario. For example, an added *whois* data object could represent the statements like, “this is less likely to be an attack if the source IP address is registered to *mycompany*”.

Tracert

There are many *tracert* tools, which complement the *whois* database. Just as *whois* can provide registration information for an IP address, current *tracert* tools can provide a fairly accurate estimation of the geographical location for a specific IP address. Using geographical information when analyzing possible attacks offers a basis for signaling anomalies within individual alerts and attack scenarios. For example, a data object containing geographic information can describe statements like “if an alert has a source IP address outside the United States, this is more likely to be an attack”. Such a statement can be very powerful when relating to a US based company’s intranet—why would any non-employee outside the country be accessing the intranet?

Handles

Incorporated into each of these data objects is a common set of handles. Every data object has a Boolean tag that represents whether the object’s requirements are satisfied. Each object also has a probability field, which gives a numeric representation of the extent to which an object’s requirements have been satisfied. These values are determined by heuristics that are custom designed for each data type. The common handles on each data object make it easy for them to interact with each other and be incorporated into reasoning processes.

CONNECTIONS

Data objects alone cannot describe potential attacks. Relationships between data objects are necessary to create a cohesive structure amongst the data objects before translating then into Bayes nets and regular expressions. Graphically, a relationship is nothing more than a line

between two or more data objects. By connecting the data objects the user is describing an ambiguous relationship between the features described by each of the data objects. This ambiguity must be resolved for the system to detect any future alerts. To aid in clarification, we have determined there to be three classifications of connections between data objects: And, Or, and Congruent; Maybe And, Maybe Or, and Maybe Congruent; and Prerequisite/Consequence.

And, Or, and Congruent

The *And*, *Or*, and *Congruent* connections are very straight forward. They quite simply perform conjunction, disjunction, and congruency operations between the data objects they connect. Once the connection is in place, the connected data objects can be grouped together as a single data object for analysis. Due to the Boolean nature of these functions, *And*, *Or*, and *Congruent* connections cannot have any associated weights. They can only be used to combine data objects in the form of a regular expression.

Maybe And, Maybe Or, and Maybe Congruent

The *Maybe* connections are descendants from the *And*, *Or*, and *Congruent* connections. Like their parent connections the *Maybe* connections are used to group data objects into a larger, collective data object. However, unlike the *And*, *Or*, and *Congruent* operations, they can be weighted and, thereby, introduce a level of uncertainty into the relationship. By introducing uncertainty, the connected data objects produce a Bayes net used to evaluate values within the encompassing data object.

Prerequisite/Consequence

In order to fully utilize casual and clustering correlation [Ning et al., 2004], data objects can be linked together as prerequisites and consequences. The most obvious technique to create such a link is to simply create connections between hyper-alert fact data objects with one data object as the prerequisite or consequence of the other, thereby creating a representation of a hyper-alert type. Due to the encompassing nature of data objects, it is possible for one prerequisite or consequence link to apply to multiple hyper-alert facts if they are contained within a single data object.

GENERALIZATION

When placed on the canvas, a data object is created signifying specific and/or generalized aspects of a possible attack. Examples of specific aspects could include the general threat level according to the Internet Storm Center, an explicit IP Address, or textual information within a *whois* query. Generalized features are more difficult to frame, but are necessary to extend the graph and detect similar, but not identical attacks. Therefore, we provide some techniques to aid and constrain this information. The first step of which is to classify the various types of generalization. These types can then be described on the canvas via color-coding. This will allow the user to see

what type of generalization the system is applying to a specific object. If that generalization type is incorrect, the user can manually specify the type of generalization for that specific data object.

- *No generalization.* No generalization means that all values within the data object are explicit (i.e., specified as specific values by the user) and fixed (i.e., may not be changed by the system). The system cannot use the values as a template for programming by example techniques, but rather as specific facts during analysis.
- *Template generalization.* Template generalization is a classification where values within a data object are explicit but not fixed. Rather, the values are predicates to be used with programming by example techniques. These data objects essentially serve as a template for alert analysis.
- *Inferred generalization.* Inferred generalization occurs when values within a data object are not explicit but are fixed. This occurs when a data object is created from an alert and rather than representing the alert or any values within the alert, the data object represents the technique by which the alert was obtained. For example, if a user creates a data object from a specific ranking within a hierarchy of filtered list of alerts, not only are the specifics of the alert contained within the data object but also the parameters of the search used to find the alert. Similarly, if an alert was added from the system filtered alert list, the data object graph used to filter that alert would be stored within the new data object. This is an ideal method to represent ranges of information.
- *Full generalization.* Full generalization occurs when values within a data object are neither explicit nor fixed. Essentially, there are no bounds on the data object. A data object falling within this classification could represent any data object. This could be a useful technique for influencing attack scenarios en masse, for example decrease the likelihood of all attacks originating from an IP addresses belonging to a specific *whois* registrant.
- *Mixed generalization.* The mixed generalization classification is necessary due to the encompassing abilities of data objects. Since a single data object can contain other data objects, it is possible for internal data objects to be in different generalization classifications. In this situation, mixed generalization can classify the encompassing data object.

Our generalization framework is influenced by work in machine learning, in particular the view of generalization as search [1]. Rather than relying on a completely automated search process, however, our representation allows the user to specify the types of generalization that can be carried out

locally on specific data objects. If successful, the generalization process will lead to more abstract representations that will be able to capture more attacks than individual, more specialized representations.

CUSTOMIZATION

We are planning two areas for interface customization. The first addresses the process of users constructing graphs to represent possible attack scenarios. As a user adds information to the canvas, the system monitors the types of objects that are created, the connections between them, and any weights that the user adds. Sequences of these actions constitute user behaviors. Our goal is for the system to build generalizations of patterns in the user's actions, and tie these patterns to specific types of hypotheses, in order to create larger building blocks that could drive the construction process forward in larger steps. This is an interesting but difficult problem, in that it requires combining observations of the user's actions with information not accessible to the user (i.e. from external data sources), as is common in knowledge acquisition.

The second area for customization follows the more traditional path of programming by demonstration. The system will offer suggestions based on its queries to external sources, and the user will also have the capability of making such queries. In contrast to the activity described above, this is a more exploratory than constructive process. Based on the context of a given hypothesis, the system will search for patterns in the precedence and content of user queries, in order to generate macros that can be activated in similar contexts.

ACKNOWLEDGMENTS

This effort was supported by the National Science Foundation under award ANI-0219315. The information in this paper does not necessarily reflect the position or policies of the U.S. government, and no official endorsement should be inferred.

REFERENCES

1. T. Mitchell, *Machine Learning*, McGraw Hill, 1997.
2. P. Ning, Y. Cui, D. S. Reeves, Constructing Attack Scenarios through Correlation of Intrusion Alerts, in *Proceedings of the 9th ACM Conference on Computer & Communications Security*, pp. 245-254, Washington D.C., November 2002.
3. P. Ning, D. Xu, C. G. Healey, and R. St. Amant, Building Attack Scenarios through Integration of Complementary Alert Correlation Methods, in *the 11th Annual Network and Distributed System Security Symposium (NDSS '04)*, February, 2004.