Extending Case-Based Reasoning to Network Alert Reporting

Robert F. Erbacher
U.S. Army Research Laboratory
2800 Powder Mill Road
Adelphi, MD 20783
301-394-1674
Robert.F.Erbacher.civ@mail.mil

Steve E. Hutchinson
ICF International Inc., for
U.S. Army Research Laboratory
2800 Powder Mill Road
Adelphi, MD 20783
301-394-1237
Steve.E.Hutchinson.ctr@mail.mil

***Abstract*—A substantial amount of cyber security analyst time is spent handling well-known and naïve threats and policy violations on the local network. This includes both the time spent actually identifying and analyzing the activity as well as generating and filing reports associated with the activity. With increasing concern over advanced persistent threats, there is an interest in the development of techniques to automatically handle well-known threats and policy violations. We propose extensions to existing case-based reasoning approaches to support the unique requirements of cybersecurity report generation. Specifically, we consider the fact that we are reporting on hostile actors that will attempt to game the system or manipulate the system to actually aid the actors in obfuscating their activity. In this paper, we describe the need for automated reporting, the applicability of case-based reasoning, our proposed extension to the standard case-based reasoning system model, and provide examples of the modified case-based reasoning system as applied to example cybersecurity scenarios.**

Keywords—Case-Based Reasoning, Incident Reporting, CyberSecurity.

I. INTRODUCTION

Computer security typically consists of four primary stages: comprehension, protection, identification, and remediation. Comprehension amounts to the range of research working to identify vulnerabilities, types of attack, sources of attack, etc. Protection is the most known component since by protecting the network and its associated systems we alleviate the potential for compromise. The third component, identification, amounts to the identification that some form of compromise or violation has occurred and must be remediated. In this paper, we are concerned with the third component, identification. More specifically, we are concerned with the most common form of identification, namely intrusion detection systems.

The most well discussed problem associated with intrusion detection systems (IDS) [3][4] is the need to handle large numbers of false positives and false negatives. A less obvious problem arises from the sheer number of alerts being generated by intrusion detection systems, i.e., true positives. These alerts *must* be remediated by the analyst; they cannot simply be ignored. This is true even for the most unsophisticated of attacks identified by the IDS. This requires that a report be generated for each of these alerts. The process of acquiring and validating the necessary evidence along with filling in the required report is time consuming. While critical for many attacks, this usage of time can feel like a waste for naïve and well known attacks. The need to improve analyst efficiency in report generation is in line with other approaches specifically designed to improve analyst efficiency in other steps of the process [7], such as the application of visualization to the intrusion detection process itself [6].

The goal with this research is to identify mechanisms by which these naïve and well known attack alerts can automatically be handled by a case-based reasoning system [1]. Our proposed case-based reasoning system provides multiple stages of analysis but ultimately removes the tedious work of handling the majority of unsophisticated attacks. The proposed system will use case-based reasoning (CBR) to match the current alert to the well-known alerts in the knowledge base and use an associated template to fill in the report form. Typically, report generation requires that the analyst enter a wide array of data identifying:

- Affected hosts
- Analyst doing the review
- Attack classification or categorization
- Attack severity
- Evidence of the attack
- Identification of the individual responsible for the affected machine

The incident reporting form for the state of Texas [21] provides an example of a more substantial report format. It requires the entry of 34-42 text boxes, depending on the amount of additional information needing to be entered, and three check box groups. While most incident response forms are not so extensive, the Texas document exemplifies the amount of effort that must be incurred to handle incident reporting. It also exemplifies the information the case-based reasoning system must be able to acquire to properly fill in the form. If the form cannot be filled in then it will fall to the analyst to complete the process, again placing time demands on them. More representative examples are provided by the Request Tracker for Incident Response (RTIR) form by Best Practical Solutions LLC [22][23].

II. Previous Work

In this research, we explored the ability to automatically generate incident reports using CBR based on a central knowledge-base. While no prior work has examined report generation for network incident reports, there is existing research from the data mining field by Vesanto et al. [20]. The work by Vesanto et al. focuses on generating reports documenting the characteristics of the data after data mining algorithms have executed to provide a starting point for understanding the data and the results of the data mining operations.

Antonides et al. [2] looked at streamlining incident reporting for network security incidents. Antonides et al. focused on the issues of non-standardized formats, mediums, and timeframes. While of relevance, this work did not deal with the issue of automating the report generation.

A. *The Network Alerts Domain*

Most IDSs will generate alerts of one form or another; it is actually unusual for an IDS to not generate alerts. The difficulty arises from the fact that IDSs typically use different formats for their alerts, leading to correlation difficulties. For general purpose automation of alert generation, these correlation issues will need to be resolved. For our needs we will be able to focus on a single format. The most well-known alert format is that of Snort [13]; Examples of standard snort rules can be found at [26] and alternative rules from bleeding snort at [27]. While fully explaining the full rule configuration capability of snort is well beyond the scope of this paper, there are several arguments that are absolutely critical to our needs, namely: classtype, sid, msg, and reference.

Of note is the fact that most alert generating tools, as with snort, will provide some method of specifying similar types of information; though often not as distinctly. For instance, the Bro Network Security Monitor [18] provides very limited support but can be adapted through scripting to provide equivalent support. In essence, Bro provides a single message when an alert is raised, namely: `"event <string>"`; though more extensive capabilities are provided to create alert rules. This string could essentially embed the same information as snort.

Both bleedingsnort.com [24] and snort.org [25] provide multiple levels of documentation on generated alerts. The generated alerts themselves are rather cryptic [28]. Both environments provided extended messages through message maps [29], though these are still simple sentence fragments attempting to document the message in a limited format. The epitome of the goal is the extended documentation provided online for snort [30]. In essence, the unique sid numbers intrinsic to the snort rules are associated with extensive documentation. However, this additional information being online requires manual searching and downloading (copy and paste) by the analyst. In addition, most snort installations would likely not even have any such additional information available for the local rule sets; maintaining provenance and correct documentation management with rules as they are being updated in an operational environment is a challenge in and of itself.

The lack of documentation is exemplified by snort preprocessor generated alerts that will typically not have a sid number or even a classification label; identification of which preprocessor rule fired is the extent of classification provided. With general rules, even if the classification and sid are not included, the message associated with the alert provides some level of discernment; the hope being that these message are actually *unique* and descriptive. This cannot be guaranteed with the thousands of rules in a typical IDS configuration.

This deviation in documentation exemplifies the goals of this research. Intrinsically, the goal is to reduce the workload on the analyst when dealing with naïve and well known attack alerts. Fundamentally, this means filling in the report forms, identified in the introduction, automatically. However, a secondary goal will be to avoid requiring analysts to create and maintain the extensive set of documentation demonstrated online by snort at [30]. The desired solution will support all scenarios from the extremely detailed information being provided as is demonstrated by snort to the scenario in which an alert generation tool, such as Bro, is used and very little additional information has been provided in the alert message or associated documentation. Initially, a site will have very few templates in the knowledge-base, but as events are identified the knowledge-base will grow; clearly, the most common events will get templates first showing a very rapid decline in analyst time expenditure on naïve events for both the creating of the supporting documentation as well as the generation of report templates.

It is for this reason that we examine case-based reasoning systems. Such CBR systems are intrinsically designed to handle this variation in documentation automatically and will avoid the need for analysts to independently expand and maintain a documented database of information. This greatly resolves the need for analyst to waste time on naïve and well-known attacks.

B. *Case-Based Reasoning (CBR)*

Case-based reasoning originally derived from a cognitive science based model of dynamic memory by Schank et al. [14]. Schank's work resulted in the first two computer-based CBR systems in 1983 [9][11]. The fundamental process model, Figure 1, was identified by Aamodt et al. [1] in 1994. It is this process model, termed the 4REs, which has driven the majority of research and application development in CBR research since. The term 4REs derives from the four action model represented by retrieve, reuse, revise, and retain. Fundamentally, the model is based on the incorporation of a knowledge base [17], or case base, which is a historical set of scenarios for which solutions are known. A new scenario is then matched against the existing scenarios to find the most relevant match that is then mapped to the new scenario providing an updated solution.

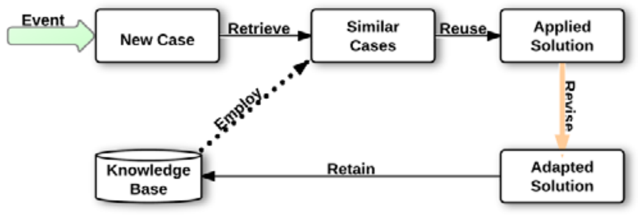

Figure 1. Traditional case-based reasoning process model.

There have been many surveys and reviews of the existing literature related to CBR [12], an indication of the extent of research performed in this domain. Each of the four main actions in CBR have an extensive research following, though the research in retrieval is likely the most extensive due to the critical importance in the returned results on the case-based reasoning process success.

III. Application of CBR to Network Alerts

With the goal of reducing analyst effort required to handle naïve and well-known attacks we examine the applicability of CBR-based systems to the task of correlating alerts and automating report generation. In the simplest case, we can simply use the sid numbers associated with the snort alert to acquire the detailed description from a database. However, such a scenario will not be effective in the general case, in which such information is not available. Relying on sid numbers will also not support the case of slight variations in an attack that may cause the rule to fire but the existing data is inadequate in a fixed form. Finally, since the goal is to reduce analyst effort, requiring analysts to manually maintain such databases of attacks and associated report templates is an unacceptable time sink. As discussed in the previous work session these are tasks that CBR-based systems are specifically designed for. The unique nature of network analyst work requires adaptations of the typical 4RE model associated with CBR-based systems, Figure 2.

Retrieve – the retrieve function is much more complex than a traditional CBR-based retrieve function due to the fact that we may need to retrieve cases based on multiple alerts (correlation) and the complexity of the information being searched. The amount of detail we have from the alert will aid in determining what similarity metric should be employed; if we can use a simple overlay metric then we do not want to use a more complex error-prone metric.

Reuse – the application of the retrieved template from the database will not change from the traditional function.

Revise – since alerts did fire on the specified event we know that the attack has been seen before and rules generated. However, this could be a slight variation of known malicious activity and will most likely involve different IP addresses. The revise function should be able to automatically fill in the template with both the static information from the template and the dynamic information from the current alerts. This will then need to be revised further by the analyst (if needed) for accuracy, to add additional validation for the new variant, etc.

At this point, the analyst must determine whether to distribute the adapted solution, and generated report, or not.

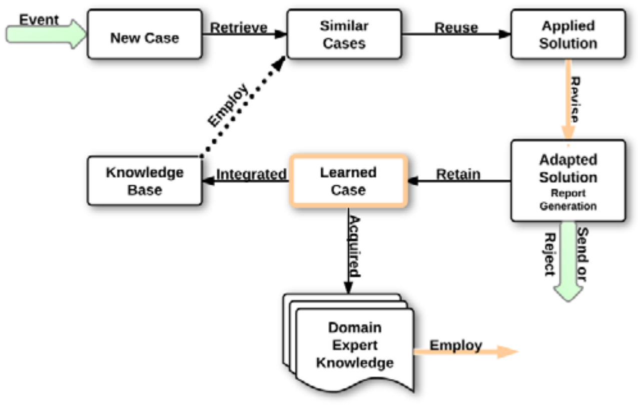

Figure 2. CBR process model extended to the unique characteristics of network security incident report generation. System IO is emphasized as green. Analyst interaction and decision making is emphasized in orange.

Retain – if the new case has important modifications over existing cases, i.e., it's a new variant of a known malicious attack, then the case should be learned by the system and injected into the knowledge base to simplify report generation of the next occurrence of this variant. This will likely only be required for substantial deviations from the original variant and must be approved by the analyst. This is due to the potential for an attacker to game the system if it were fully automated. Thus, the Learned Case in Figure 2 allows for the exemplification of the actions for the specific case; which are abstracted out in Figure 1. Specifically, the analyst must determine if the new case should be integrated into the knowledge base for use in the future and whether this specific case should be acquired as new domain expert knowledge, in scenarios where the domain experts can learn from the scenario and improve their ability to more efficiently resolve known threats and handle new evolutions of known threats. Once acquired, this new domain knowledge can be employed by the domain experts in future scenarios.

When considering whether to integrate the learned case, the analyst must determine if injecting the new variant into the knowledge base will impact future interpretation of events by missing critical activity, i.e., not seeing a particular alert could impact an analyst's world view and impact detection of advanced persistent threats.

Whether the new event is integrated into the knowledge base or not, the knowledge of the new attack characteristics will be added to the analyst's domain expertise that can be further employed by that analyst in future alert analysis scenarios. For instance, identifying trends in how variants of known malicious events are being generated. This type of trending analysis cannot currently be integrated into CBR-based systems and is the reason the analyst must remain firmly integrated into the analysis and report generation process, even if the tedious aspects of this process are handled automatically.

Existing research has examined the issue of removing harmful cases from the case base, such as duplicate entries [13]. However, the need for the analyst to manually identify whether a case should be added to the knowledge base is unique. These cases may not be harmful per se but may simply be alerts that the analyst must be aware of and handle directly.

There may be cases for instance that can too easily obfuscate non-naïve attacks that should not be handled automatically. Similarly, we must not exhaust analyst time by forcing them to perform more labor by validating the proposed report. Doyle [5] proposed a mechanism by which explanatory text can be generated within CBR systems. Similarly, Silva et al. [15] discuss the integration of arguments into the knowledge base.

IV. Validation/Case Studies

Given that we currently only have a theoretical model, we performed case studies to validate the model. The validation based on the model will help to ensure the likelihood of success from a full implementation. The scenarios exemplified in the below case studies are alert messages taken from the bleeding snort database of snort alerts [25].

Scenario 1: Alert exists in knowledge-base

1. Alert generated with message M1="BLEEDING-EDGE MALWARE 180solutions Update Engine". The rule from bleeding snort is as follows:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET
$HTTP_PORTS (msg: "BLEEDING-EDGE MALWARE
180solutions Update Engine"; flow:
to_server,established; content:"GET"; depth: 3;
content:"Host|3a|"; within: 300;
content:"ping.180solutions.com"; within: 40;
reference:url,www.safer-
networking.org/index.php?page=threats&detail=212
; classtype: trojan-activity; sid: 2000930;
rev:4; )
```

(*Begin Retrieve*)

2. Perform search using overlay metric for message M_1
   a. Binary similarity metric
   b. Only return exact matches
   c. Returns a single exact match
3. Calculate performance metrics to identify confidence to the analyst. Returning a complete set of metrics allows individual analysts to identify what metric they find most relevant, especially under different scenarios.
   a. tp (true positives) = 1
   b. fp (false positives) = 0
   c. fn (false negatives) = 0
   d. tn (true negatives) = 3058
   e. precision [8] $= \frac{tp}{tp+fp} = 1$
   f. recall [8] $= \frac{tp}{tp+fn} = 1$
   g. accuracy [8] $= \frac{tp+tn}{tp+fp+fn+tn} = 1$
   h. F_1 (F-measure) [19] $= 2 \cdot \frac{precision \cdot recall}{precision + recall} = 1$
   i. Conciseness[6][7][10] $= \frac{tp+fp}{tp+fp+fn+tn} \times 100\% = 0.0327\%$

The goal with conciseness is to provide an indication as to what percent of the total documents were returned in the results; i.e., how selective or unique are the returned documents. While conciseness is used in the literature, we find *selectivity* a more representative term; though selectivity itself has a different defined meaning. False positives are included in the formula to provide an indication of the true *selectivity* of the results returned to the user, since the false positives will be additional results the user needs to parse to find the best result.

4. A single match is returned

(End Retrieve)

(Begin Reuse)

5. Extract report template from knowledge base
6. Fill in template report with details from this event
   a. IP Addresses
   b. Data/Time
   c. Analyst performing review

(*End Reuse*)

(*Begin Revise*)

7. Analyst examines form for validity, acceptance, applicability, etc.
   a. Does the particular policy affect the designated location
   b. Is the alert still a reportable incident
   c. Identify the level of the incident to determine extent of notification required
8. Modify report and disseminate or reject the report

(*End Revise*)

(*Begin Retain*)

9. Analyst determines if this event should be specifically stored in knowledge base
10. Store in knowledge base

(*End Retain*)

11. Analyst's domain expertise enhanced
12. Analyst's enhanced domain expertise employed in future incidents

Scenario 2: Alert does not exist in knowledge-base for one level

1. Alert generated with message M_1="BLEEDING-EDGE MALWARE 180solutions Spyware (action url reported)". The rule from bleeding snort is as follows:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET
$HTTP_PORTS (msg: "BLEEDING-EDGE MALWARE
180solutions Spyware (action url reported)";
flow: to_server,established;
uricontent:"/actionurls/ActionUrl"; nocase;
reference:url,securityresponse.symantec.com/avce
nter/venc/data/pf/adware.180search.html;
classtype: trojan-activity; sid: 2001399; rev:6;
)
```

(*Begin Retrieve*)

2. Perform search using overlay metric for message M_1
   a. Binary similarity metric
   b. Only return exact matches
   c. No match found. This indicates that while the rule exists no report-template has specifically been integrated into the knowledge-base related to this alert. We must locate a related template to use as a starting point.
3. Remove one word of specification and repeat overlay metric
   a. Assume each word further to the right increases specificity
   b. Assume a parenthetical set segregates one "word"
   c. M_2="BLEEDING-EDGE MALWARE 180solutions Spyware"
   d. Matches:
      i. P_1=BLEEDING-EDGE MALWARE 180solutions Spyware (tracked event reported)
      ii. P_2=BLEEDING-EDGE MALWARE 180solutions Spyware Reporting
      iii. P_3=BLEEDING-EDGE MALWARE 180solutions Spyware Keywords Download
      iv. P_4=BLEEDING-EDGE MALWARE 180solutions Spyware Install
      v. P_5=BLEEDING-EDGE MALWARE 180solutions Spyware Defs Download
      vi. P_6=BLEEDING-EDGE MALWARE 180solutions Spyware config Download
      vii. P_7=BLEEDING-EDGE MALWARE 180solutions Spyware versionconfig POST
4. Apply overlay metric to classtype for sanity check validation and generate reduced subset
   a. All potential matches P_i match classtype
5. Traverse snort alert rule argument list
   b. Accumulate similarity metrics for each potential match
      i. +1 for exact match
      ii. -1 for arguments that don't exist or extra arguments in potential match
      iii. +SMV. Otherwise add the value from the similarity metric
      iv. Return the potential matches in the order from highest score to lowest

   a. E_x=the xth argument for event E.
   b. $E_{1,1}$=tcp; $E_{1,2}$=$HOME_NET; $E_{1,3}$=any; $E_{1,4}$=$EXTERNAL_NET; $E_{1,5}$=$HTTP_PORTS; $E_{1,6}$=flow: to_server,established; $E_{1,7}$=uricontext:"/actionurls/ActionUrl"; $E_{1,8}$=nocase; $E_{1,9}$=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;
   c. $A_{x,y}$=argument. Where x is the potential match number and y is the argument number.
   d. We must search for the existence of each argument; naively this becomes n×m in the number of arguments in the event and potential match. Remove duplicate and grammatical arguments. This loses context but becomes much simpler. While many of the parameters shown below seem fairly generic, keep in mind the goal is to demonstrate a completely generic and broadly applicable approach.
   e. $A_{1,1}$=alert; $A_{1,2}$=tcp; $A_{1,3}$=$HOME_NET; $A_{1,4}$=any; $A_{1,5}$=->; $A_{1,6}$=$EXTERNAL_NET; $A_{1,7}$=$HTTP_PORTS; $A_{1,8}$=msg: BLEEDING-EDGE MALWARE 180solutions Spyware (tracked event reported) $A_{1,9}$=flow: to_server,established; $A_{1,10}$=uricontext:"TrackedEvent.aspx?"; $A_{1,11}$=nocase; $A_{1,12}$=uricontent:"eid="; $A_{1,13}$=nocase; $A_{1,14}$=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html; $A_{1,15}$=classtype: trojan-activity; $A_{1,16}$=sid: 2001397; $A_{1,17}$=rev:6;
      i. Remove $A_{1,1}$; $A_{1,5}$; $A_{1,8}$; $A_{1,13}$; $A_{1,15}$; $A_{1,16}$; $A_{1,17}$; as being duplicates, already checked, and non-discriminable
      ii. Note that each of these arguments is essentially textual, i.e., ordinal in nature.
      iii. An overlay-based textual similarity metric will be used
   f. $A_{1,2}$=tcp; $A_{1,3}$=$HOME_NET; $A_{1,4}$=any; $A_{1,6}$=$EXTERNAL_NET; $A_{1,7}$=$HTTP_PORTS; $A_{1,9}$=flow: to_server,established; $A_{1,10}$=uricontext:"TrackedEvent.aspx?"; $A_{1,11}$=nocase; $A_{1,12}$=uricontent:"eid="; $A_{1,14}$=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;
   g. Renumber
   h. $A_{1,1}$=tcp; $A_{1,2}$=$HOME_NET; $A_{1,3}$=any; $A_{1,4}$=$EXTERNAL_NET; $A_{1,5}$=$HTTP_PORTS; $A_{1,6}$=flow: to_server,established; $A_{1,7}$=uricontext:"TrackedEvent.aspx?"; $A_{1,8}$=nocase; $A_{1,9}$=uricontent:"eid="; $A_{1,10}$=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;

i. $A_{2,1}$=tcp; $A_{2,2}$=$HOME_NET; $A_{2,3}$=any; $A_{2,4}$=$EXTERNAL_NET; $A_{2,5}$=$HTTP_PORTS; $A_{2,6}$=flow: to_server,established; $A_{2,7}$=uricontext:"/showme.aspx?keyword="; $A_{2,8}$=nocase; $A_{2,9}$=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;

j. $A_{3,1}$=tcp; $A_{2,2}$=$HOME_NET; $A_{3,3}$=any; $A_{3,4}$=$EXTERNAL_NET; $A_{3,5}$=$HTTP_PORTS; $A_{3,6}$=flow: to_server,established; $A_{3,7}$=uricontext:"keywords/kyf"; $A_{3,8}$=nocase; $A_{3,9}$=content:"partner_id="; $A_{3,10}$=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;

k. $A_{4,1}$=tcp; $A_{4,2}$=$HOME_NET; $A_{4,3}$=any; $A_{4,4}$=$EXTERNAL_NET; $A_{4,5}$=$HTTP_PORTS; $A_{4,6}$=flow: to_server,established; $A_{4,7}$=uricontext:"/downloads/installers"; $A_{4,8}$=nocase; $A_{4,9}$=content:"simpleinternet/180sainstaller.exe"; $A_{4,10}$=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;

l. $A_{5,1}$=tcp; $A_{5,2}$=$HOME_NET; $A_{5,3}$=any; $A_{5,4}$=$EXTERNAL_NET; $A_{5,5}$=$HTTP_PORTS; $A_{5,6}$=flow: to_server,established; $A_{5,7}$=uricontext:"/geodefs/gdf"; $A_{5,8}$=nocase; $A_{5,9}$=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;

m. $A_{6,1}$=tcp; $A_{6,2}$=$HOME_NET; $A_{6,3}$=any; $A_{6,4}$=$EXTERNAL_NET; $A_{6,5}$=$HTTP_PORTS; $A_{6,6}$=flow: to_server,established; $A_{6,7}$=uricontext:"/config.aspx?did="; $A_{6,8}$=nocase; $A_{6,9}$=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;

n. $A_{7,1}$=tcp; $A_{7,2}$=$HOME_NET; $A_{7,3}$=any; $A_{7,4}$=$EXTERNAL_NET; $A_{7,5}$=$HTTP_PORTS; $A_{7,6}$=flow: to_server,established; $A_{7,7}$=uricontext:"/versionconfig.aspx?"; $A_{7,8}$=uricontext:"&ver="; $A_{7,9}$=nocase; $A_{7,10}$=reference:url, securityresponse.symantec.com/avcenter/venc/data/pf/adware.180search.html;

o. For brevity, consider the following table of results in terms of the similarity metric results for each of the arguments.

	$A_{x,1}$	$A_{x,2}$	$A_{x,3}$	$A_{x,4}$	$A_{x,5}$	$A_{x,6}$	$A_{x,7}$	$A_{x,8}$	$A_{x,9}$	$A_{x,10}$	**SUM**	**Rev**	**Priority Order**
P_1	1	1	1	1	1	1	0	1	-1	1	7	6	4
P_2	1	1	1	1	1	1	0	1	1	0	8	5	1
P_3	1	1	1	1	1	1	0	1	-1	1	7	3	6
P_4	1	1	1	1	1	1	0	1	-1	1	7	4	5
P_5	1	1	1	1	1	1	0	1	1	0	8	3	2
P_6	1	1	1	1	1	1	0	1	1	0	8	2	3
P_7	1	1	1	1	1	1	0	-1	1	1	7	1	7

p. Given the number of duplicate scores, we can use two additional metrics for ordering. First, the revision number associated with the snort rule will provide an indication of the currency and accuracy of the rule. Second, the old standby of using the ordering of the rules can be used.

6. A list of matches is returned in priority order

    a. Cycle through the matches in priority order

    b. Identify the highest priority match that has an associated template in the knowledge base

    c. Return this highest priority match

    d. In the case that no matches have a template in the knowledge base return to step 3 and repeat. Efficiency note: in an efficient implementation the matches without a template would obviously be pre-filtered.

7. Calculate performance metrics to identify confidence to analyst. There were three documents that had equivalent similarity metrics of 8. The differentiation between these, using order and revision, is essentially artificial and a direct indication of uncertainty; i.e., we could just have easily chosen randomly. The four documents with similarity metrics of 7 can be construed as false positives. Thus,

a.	tp	=	3
b.	fp	=	4
c.	fn	=	0
d.	tn	=	3052
e.	precision	=	0.4286
f.	recall	=	1
g.	accuracy	=	0.9987
h.	F_1	=	0.6
i.	conciseness	=	0.2288%

8. A single match is returned

(*End Retrieve*)

(*Begin Reuse*)

9. Extract report template from knowledge base

10. Fill in template report with details from this event

    a. IP Addresses

    b. Data/Time

    c. Analyst performing review

(*End Reuse*)

(*Begin Revise*)

11. Analyst examines form for validity, acceptance, applicability, etc.

    a. Does the particular policy affect the designated location

b. Is the alert still a reportable incident

c. Identify the level of the incident to determine extent of notification required

12. Modify report and disseminate or reject the report

(*End Revise*)

(*Begin Retain*)

13. Analyst determines if this event should be specifically stored in knowledge base
14. Store in knowledge base

(*End Retain*)

15. Analyst's domain expertise enhanced
16. Analyst's enhanced domain expertise employed in future incidents

For comparison, consider the results of the conciseness metric for two further iterations of the algorithm:

Metric		**Iteration 3**	**Iteration 4**
tp	=	9	3059
fp	=	N/A	N/A
fn	=	0	0
tn	=	3050	0
conciseness	=	0.2942%	100.00%

This exemplifies the impact and value of the conciseness metric. While extreme in this case, the issue is to provide a representation to the analysts of when the returned results may not be of value. Clearly, there will be jumps in the conciseness metric that clearly indicate a lack of selectivity that makes the returned results ineffective even when such jumps are not as extreme as this scenario; the need is to provide the fact that such a jump did in fact occur to the analyst. Since the total number of potential results may vary, the percentage associated with the conciseness value is more informative. For instance, the percentage will be informative whether there are 20 total potential results or 100 thousand.

Scenario 3: Why Semi-Automated, Event Priority Obfuscation

```
alert tcp $HOME_NET any -> $EXTERNAL_NET
!6661:6668 (msg: "BLEEDING-EDGE ATTACK RESPONSE
IRC - DCC file transfer request on non-std
port"; flow: to_server,established;
content:"PRIVMSG "; nocase; offset: 0; depth: 8;
content:" \:.DCC SEND"; nocase; tag:
session,300,seconds; classtype: policy-
violation; sid: 2000349; rev:5; )

alert tcp $HOME_NET any -> $EXTERNAL_NET
!6661:6668 (msg: "BLEEDING-EDGE ATTACK RESPONSE
IRC - Private message on non-std port"; flow:
to_server,established; dsize: <128;
content:"PRIVMSG "; nocase; offset: 0; depth: 8;
tag: session,300,seconds; classtype: trojan-
activity; sid: 2000347; rev:5; )
```

Note the limited differences in the above two alert rules; ignoring differences unrelated to the rule process. The first alert has `content:" \:.DCC SEND";` while the second has `dsize: <128;` the result of the rules are the same. Now consider the difference in class type; a `policy-violation` versus `trojan-activity`. The difference in interpretation by the analyst will be enormous. The trojan activity will get very high priority while the policy violation will get very low priority. In general it is the after effects of a successful compromise that an analyst can identify and not the attack itself. Thus, the attack response of the installed trojan is how the presence of the trojan will be identified.

1. Alert generated with message M_1= "BLEEDING-EDGE ATTACK RESPONSE IRC - DCC file transfer request on non-std port,"
   a. First time this alert is seen
2. Perform search using overlay metric for message M_1
   a. No acceptable matches
   b. Conciseness too high
3. Perform repeated searches using range of similarity metric for message M_1
   a. No acceptable matches
   b. F-Measure too low
4. Analyst creates report and injects into the knowledge base
5. New, zero-day, trojan injected into network
   a. Modification of existing Trojan that would normally be identified by rule with sid 2000347
   b. Challenge-response packets modified to contain `" \:.DCC SEND"` as the first sequence of bytes
6. Alert generated with message M_1= "BLEEDING-EDGE ATTACK RESPONSE IRC - DCC file transfer request on non-std port,"
7. Perform search using overlay metric for message M_1
8. Exact match found on first pass
9. Report automatically generated and distributed with `classtype: policy-violation` and associated priority.

This rather simple example exemplified the impact of a fully automated system; note that these are actual rules in the bleeding snort rule base; given the tens of thousands or perhaps hundreds of thousands of events in a typical rule-set it is likely quite common for such similar rules to exist. An analyst reviewing the report before letting it be sent out will identify potential inconsistencies and will raise the priority when multiple alerts start to be generated. The automated system will not notice anything unusual. This example exemplifies the ease with which an attacker can de-escalate the priority of their compromise and evade detection. Given the extent to which attackers go through to compromise hosts we have to assume they will modify compromises in a similar fashion to maintain control of the host for as long as possible. Keep in mind the bleeding snort rule base is public domain and the attackers will have access to it. While internal rules may differ, attackers will also put a lot more time into evasion than we did, especially nation states.

V. Conclusions

This research developed a model for the automated generation of reports based on a knowledge base created by analysts over time. This automation is targeted at known threats and will directly automate slight deviations in the attack. Sophisticated attacks, i.e., true zero-day attacks, will sufficiently differ from existing attacks in the knowledge base to go completely unseen by the automated system. While the creation of the reports is automated, we do incorporate the analyst into the process to validate the report before disseminating the report or incorporating the results into the knowledge base. This will prevent attackers from gaming the system. As attackers will analyze the code of an application to identify vulnerabilities i.e., buffer overflows, they will also attempt to identify vulnerabilities in the analyst process.

The resultant model was validated with case studies. These case studies provide validation as to the viability of the model.

Of note is the reliance on open source rule-systems for snort, i.e., bleeding snort. As we rely solely on open source capabilities, we are limiting the potential sources of attack; i.e., we only assumed snort alerts were available. In a real environment multiple tools will likely be available all generating alerts. The similarity metric process, exemplified in Figure 4, assumes multiple alert types. This is why we cannot always just assume the overlay metric will be sufficient. We must support all alert types in the automated report generation and correlation capabilities. The current model supports this. The limitation of such generic support is that it provides many more avenues for an attacker to compromise or attempt to game the system.

VI. References

[1] Agnar Aamodt and Enric Plaza. 1994. “Case-based reasoning: foundational issues, methodological variations, and system approaches.” *AI Commun.* 7, 1 (March 1994), 39-59.

[2] Antonides, J.R.; Benjamin, D.N.; Feldpausch, D.P.; Salem, J.S.; , “Streamlining the US Army network incident reporting system,” *Systems and Information Engineering Design Symposium, 2008. SIEDS 2008. IEEE* , vol., no., pp.17-21, 25-25 April 2008.

[3] J. P. Anderson. “Computer security threat monitoring and surveillance.” Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, April 1980.

[4] D. E. Denning. “An intrusion-detection model.” *IEEE Transactions on Software Engineering*, 13(2), February 1987.

[5] Donal Doyle, “A Knowledge-Light Mechanism for Explanation in Case-Based Reasoning,” University of Dublin, Trinity College. Department of Computer Science, Doctoral Thesis TCD-CS-2005-71, 2005.

[6] Robert F. Erbacher, Kim Christensen, and Amanda Sundberg, "Designing Visualization Capabilities for IDS Challenges," *Proceedings of the IEEE Workshop on Visualization for Computer Security*, Minneapolis, MN, October 2005, pp. 121-127.

[7] Robert F. Erbacher and Karl Sobylak, "Improving Intrusion Analysis Effectiveness," *2002 Workshop on Computer Forensics*, Moscow, ID, September, 2002.

[8] Gafni, R., “Framework for Quality Metrics in Mobile-Wireless,” (K. Lynch, Ed.) *Interdisciplinary Journal of Information, Knowledge, and Management*, Vol. 3, pp. 23-38, 2008.

[9] Gafni, R., “Quality Metrics for PDA-based M-learning Information Systems,” (A. Koohang, Ed.) *Interdisciplinary Journal of E-Learning and Learning Objects*, Vol. 5, pp. 359-378, 2009.

[10] Junker, M.; Hoch, R.; Dengel, A., “On the evaluation of document analysis components by recall, precision, and accuracy,” *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, pp.713-716, 20-22 Sep 1999.

[11] Janet Kolodner, “Reconstructive Memory: A Computer Model,” *Cognitive Science* 7 (1983): 4, pp. 281-328.

[12] Dawn Lawrie, Henry Feild, and David Binkley. 2006, “Syntactic Identifier Conciseness and Consistency,” In *Proceedings of the Sixth IEEE International Workshop on Source Code Analysis and Manipulation* (SCAM '06). IEEE Computer Society, Washington, DC, USA, pp. 139-148.

[13] Michael Lebowitz. 1983. “Memory-based parsing,” *Artificial Intelligence* 21, 4 (November 1983), pp. 363-404.

[14] Ramon Lopez De Mantaras, David McSherry, Derek Bridge, David Leake, Barry Smyth, Susan Craw, Boi Faltings, Mary Lou Maher, Michael T. Cox, Kenneth Forbus, Mark Keane, Agnar Aamodt, and Ian Watson. 2005. Retrieval, reuse, revision and retention in case-based reasoning. *Knowl. Eng. Rev.* 20, 3 (September 2005), 215-240.

[15] Stewart Massie, Susan Craw, and Nirmalie Wiratunga, “When Similar Problems Don't Have Similar Solutions,” In *Proceedings of the 7th international conference on Case-Based Reasoning: Case-Based Reasoning Research and Development* (ICCBR '07), Rosina O. Weber and Michael M. Richter (Eds.). Springer-Verlag, Berlin, Heidelberg, 2007, pp. 92-106.

[16] Roger Schank, *Dynamic Memory: A Theory of Learning in Computers and People* (New York: Cambridge University Press, 1982).

[17] Silva, Luís A. L.; Buxton, Bernard F.; Campbell, John A. “Enhanced Case-Based Reasoning through Use of Argumentation and Numerical Taxonomy.” *The 20th International Florida Artificial Intelligence Research Society Conference* (FLAIRS-20), Special Track on Case-Based Reasoning. AAAI Press, Key West, Florida (2007) 423-428.

[18] L. K. Soh and T. Blank, “Integrating Case-Based Reasoning and Meta-Learning for a Self-Improving Intelligent Tutoring System. *Int. J. Artif. Intell. Ed.* 18, 1 (January 2008), 27-58.

[19] Stahl, A. Learning of Knowledge-Intensive Similarity Measures in Case-Based Reasoning. PHD-Thesis, dissertation.de, Technische Universität Kaiserslautern, 2004.

[20] Vern Paxson, “Bro: A System for Detecting Network Intruders in Real-Time,” *Computer Networks*, 31(23—24), pp. 2435-2463, 1999.

[21] van Rijsbergen, C. 1979. *Information Retrieval*. London: Butterworths.

[22] J. Vesanto and J. Hollmén, A. Abraham and M. Koppen, “An automated report generation tool for the data understanding phase,” *Hybrid Information Systems*, pp. 611 - 626, 2002.

[23] http://www2.dir.state.tx.us/SiteCollectionDocuments/Security/Incident%20Management/securityincident_reportingform.doc. (Accessed on 06/06/2011)

[24] http://bestpractical.com/static/rtir/rtir-presentation.pdf. (Accessed on 06/06/2011)

[25] http://www.ja.net/services/csirt/wp-content/uploads/rtir-incident-handling.pdf. (Accessed on 06/06/2011)

[26] http://www.bleedingsnort.com/downloads/ (Accessed on 06/21/2011)

[27] https://www.snort.org/snort-rules/#rules (Accessed on 06/21/2011)

[28] http://www.bleedingsnort.com/downloads/bleeding-all.rules (Accessed on 06/21/2011)

[29] http://hintsforums.macworld.com/archive/index.php/t-36315.html (Accessed on 06/21/2011)