A Knowledge-Based Approach To Intrusion Detection Modeling

Sumit More, Mary Matthews, Anupam Joshi, Tim Finin
Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, MD, USA
{*sumitm1, math1, joshi, finin*}*@umbc.edu*

***Abstract*—Current state of the art intrusion detection and prevention systems (IDPS) are signature-based systems that detect threats and vulnerabilities by cross-referencing the threat or vulnerability signatures in their databases. These systems are incapable of taking advantage of heterogeneous data sources for analysis of system activities for threat detection. This work presents a situation-aware intrusion detection model that integrates these heterogeneous data sources and build a semantically rich knowledge-base to detect cyber threats/vulnerabilities.**

***Keywords*-security, vulnerability, intrusion detection, information extraction, ontology**

I. Introduction

Cyber crimes are being used to assist activities like espionage, politically motivated attacks and credit card fraud at an alarming rate. For example, [1] describes how Tibetan computer systems were compromised giving attackers access to potentially sensitive information, [2] reports how an attacker defaced the web-site of Turkey's embassy in China and [3] reports how hackers stole 40 million credit card numbers.

State of the art intrusion detection and prevention systems (IDPS) perform signature-based monitoring of the cyber-infrastructure to identify any malicious activities and generate an alert when such activity is detected. These systems share a limitation that if the signature of an attack is not available in their database, they cannot detect it. These systems are also point-based solutions which are currently incapable of integrating information coming from heterogeneous sources. The heterogeneous data sources can provide very useful spatial and temporal information of the system activities, which can be useful in tracking threats and attacks. We argue for a semantically rich approach to intrusion detection, where gathering data from different sources, integrating them, and reasoning over such information-rich knowledge-base will improve the detection of detect an attack or threat. We present a framework for this ontological approach and demonstrate its working.

II. Background and Previous Work

Intrusion detection and prevention systems like Snort [4] and IBM X-Force [5] are signature-based systems that monitor a system's behavior and compares it with a predefined notion of acceptable behavior. If the system deviates from the predefined and fixed description of acceptable behavior, an associated set of anomalous activities is checked, and an alert is raised if the current activity is found in that set. Though most of these IDS/IPS systems have well defined attack update mechanisms that keep them current with information on new attacks, they face certain limitations. These systems cannot detect threats in the infrastructure if the signature of the threat is not present in the system database. Apart from the traditional IDS/IPS systems, there are many other host and network based activity monitors such as Wireshark [6], Nagios [7] and Cacti [8] that provide elaborate data logs of the activities being performed at the host/network level. These monitoring tools also have a rule-based alerting mechanism, where the activities in the infrastructure are monitored and checked against a pre-defined set of rules, and corresponding actions are taken when certain events satisfy certain rules. Unless the behavior of the attack is known, these systems cannot detect it.

Many data sources on the Web today are comprised of information related to intrusions and attacks in different levels of verbosity. Mulwad et al. [9] described a system for extracting information about vulnerabilities and cyber attacks from different unstructured data sources like vulnerability description feeds (CVE, CCE, CPE, CVSS, XCCDF, OVAL) [10], hacker forums, chat rooms, blogs, etc., and informing the expert about it. Khadilkar et al. [11] explain the importance of a semantic model for information representation and present an ontology for National Vulnerability Database. They demonstrate the building of a knowledge-base from structured a data set and its usage.

Undercoffer et. al [12], [13] emphasized the importance of ontological linking of the vulnerabilities for a more effective IDS mechanism. Ontological specification of attacks and intrusions presents superior machine interpretable definitions of the concepts and relations between them, as compared to the conventional taxonomic representation. This makes way for knowledge sharing and reuse, as well as better reasoning and analysis of the information at hand.

Undercoffer [14] presents a host based intrusion detection system, making use of ontological representation of the intrusions and attacks, which performs better than the conventional signature-based intrusion detection system.

Integration of conventional signature-based intrusion de-

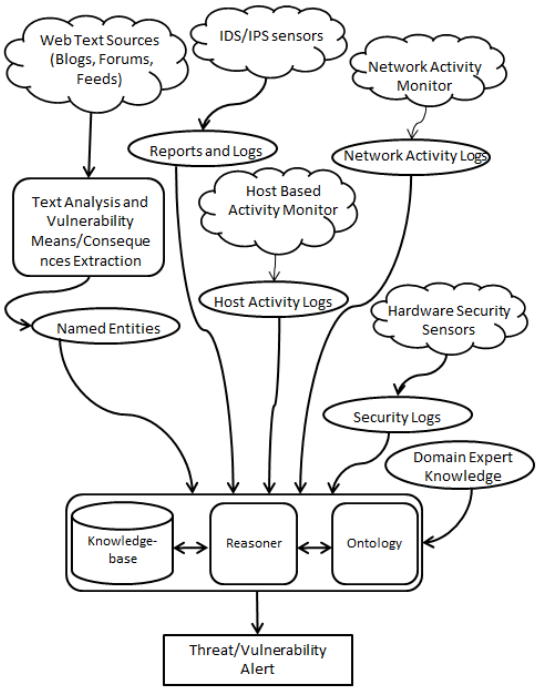

Figure 1. Situation Aware Intrusion Detection System: System Architecture

tection with other data sources along with ontological reasoning can lead to an intrusion detection system that has the ability to potentially link and infer means and consequences of cyber threats and vulnerabilities whose signatures are not yet available. The key components of this system would be the IDS/IPS sensor information module, data from different sensor streams, text-data from web, domain expert knowledge, the ontology knowledge base and the reasoner.

III. System Architecture and Implementation

Figure 1 shows the system architecture of our framework. The architecture can be broadly divided into two sections: data streams and ontology knowledge-base and reasoner.

A. Data Streams

The data streams include different channels that provide useful information related to an attack. Examples of some these channels are:

- Streams from host system which provide information related to the activities/processes that are executing at the host like logs from top [15] and monit [16]
- Streams from a network based activity monitors like Wireshark [6], Nagios [7], Cacti [8] etc.
- Sensor data coming from the IDS/IPS modules provide us with verbose information related to the system and network traffic, the data-packets sent and received by the system, the source and destination ports/IPs, the type of hardware at the source and destination, protocols of communication, and time-stamp related information.
- Logs from hardware sensor monitors like Cisco IPS 4200 [17].
- Web data stream which contains text related to the attacks/vulnerabilities. This unstructured text data can be parsed into named entities which can be used to enrich the knowledge-base.
- Structured text data like well-defined threat/attack descriptions. [10]

The items of interest from these data streams are asserted on facts in our knowledge-base. The integration and inferencing over this aggregated data can enable better detection of complex attacks.

B. Knowledge-Base and Reasoner

This module comprises of the ontology, knowledge-base, and the reasoning logic. We extended the ontology and knowledge-base proposed by Undercoffer [12] [13] and added rules to the reasoning logic. The ontology comprises of 3 fundamental classes: 'means', 'consequences', and 'targets'. The 'means' class encapsulates the ways and methods used to perform an attack, the 'consequences' class encapsulates the outcomes of the attack, and the 'target' class encapsulates the information of the system under attack. For instance, the 'means' class consists of sub-classes like 'BufferOverFlow', 'synFlood', 'LogicExploit', 'tcpPortScan', etc., which can further consist of their own sub-classes; the 'consequences' class consists of sub-classes like 'DenialOfService', 'LossOfConfiguration', 'PrivilegeEscalation', 'UnauthUser', etc.; and the 'targets' class consists of sub-classes like 'SystemUnderDoSAttack', 'SystemUnderProbe', 'SystemUnderSynFloodAttack', etc.

The entities that are collected from different data streams are asserted into one of the classes based on the properties of the class and the meaning of the entity. For instance, 'annots.api executible' is an object of a class 'process under stack overflow', which is a subclass of 'buffer overflow', which in turn is a subclass of 'means' class. Similarly, 'remote execution' is a subclass of 'remote to local' class, which in turn is a subclass of 'unauthorized user access' class, which in turn is a subclass of 'consequence' class. Likewise, system being monitored is an object of 'system under remote attack', which is a subclass of 'system under unauthorized user access', which in turn is a subclass of 'targets' class.

The information from different data channels is encoded in Notation-3 format. The knowledge-base is built up by encoding the information as OWL assertions which are converted to N3 triples. Notation 3 encodes a set of statements and its meaning in conjunction to the meaning of the statements. The statement of the form (x p y). asserts that the relation p holds between x and y. When p is identified

```
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix ids: <http://ebiquity.org/ontologies/cybersecurity/
ids/v2.0/ids♯ > .
[a ids:Vulnerability;
ids:hasMeans dbpedia:Buffer_overflow;
ids:hasConsequence dbpedia:Denial-of-service_attack].
```

Figure 2. This example shows the extracted information encoded as OWL assertions serialized using N3.

by a URI, and that URI when dereferenced in the web gives some information, that information may in practice be used to determine more information about the semantics of the statement. The entities collected from different data sources can be properties of a class in the ontology (e.g., port number, IP address, OS version, hardware platform etc.) or objects of the classes defined in the ontology (e.g., stack overflow in annots.api [18] can be object of 'process under buffer overflow' class in the ontology). The knowledge-base thus consists of N3 triples of the form *(subject predicate object)*.

The reasoning logic module takes inputs from different data streams, the knowledge asserted into the knowledge-base, and the ontology to infer the possibility of a threat/attack. The N3 triples are used by the reasoning algorithm to deduce an occurrence of a threat/attack. The reasoner logic infers from the knowledge-base and information gathered to flag an alert, giving the means, consequences, and targets of the attack, if there exists any. The sensor information coming from the IDS/IPS modules, and the web-text information coming from the web-text analyzer are used for continuous evolution of the knowledge-base. The reasoning logic is a set of rules defined in N3. The knowledge base that is built up by asserting the ontology is used by these rules to derive chains of implications. Instances are asserted and de-asserted into/from the knowledge base as temporal events occur. The query language is of the form ((predicate)(subject) (object)), where at least one of the three elements of the triple must contain a value. The other one or two elements may be left uninstantiated (signified by prefacing them with a ?). If there are any triples in the knowledge base that match the query either as the result of an assertion of a fact or a derivation of rules resulting from the chain of implication, the value of those triples will be returned. Figure 2 shows an example of an OWL assertion represented in the N3 notation.

Figure 4 shows the ontology backbone of the framework [13] [19]. It gives a high-level overview of the reasoning mechanism being used by the framework for analysis and result deduction. Each of the classes of the ontology have properties which give important information regarding that class. For example, the 'system' class has properties like 'hasMaliciousProcess', 'maliciousProcessDetails', 'hasAffectedProduct', 'affectedProductDetails', 'outboundAccess', 'portDetails' etc. which map information coming from the

```
{ebIDS:webtext ebIDS:hasVulnerabilityTerm "true" .
ebIDS:webtext ebIDS:hasSecurityExploit "true" .
ebIDS:webtext ebIDS:hasText ?Product .
ebIDS:scannerLog ebIDS:hasProduct ?Product .
ebIDS:webtext ebIDS:hasText ?Process .
ebIDS:scannerLog ebIDS:hasProcessExecuted ?Process .
ebIDS:scannerLog ebIDS:hasPortOpened "true" .
ebIDS:scannerLog ebIDS:hasOutboundConnection "true" .
ge(ebIDS:scannerLogOutboundPortOpenTimestamp
ebIDS:scannerLogProcessExecutionTimestamp)
ebIDS:attack ebIDS:hasMeans "Arbitrary Code Execution".
ebIDS:attack ebIDS:hasConsequence "Unauthorized Remote
Access"}
=>
{ebIDS:system ebIDS:hostUnderAttack "true".
ebIDS:webText ebIDS:hasProduct ?Product.
ebIDS:webText ebIDS:hasProcessExecute ?Process.
ebIDS:means ebIDS:maliciousProcess ?Process.
ebIDS:means ebIDS:affectedProduct ?Product. }
```

Figure 3. Sample Reasoning Logic Rule

network scanner and the web-text. In the Adobe attack, these properties had corresponding values of 'true', 'annots.api', 'true', 'Adobe Reader 8.1.1', 'true', and 'port 80'.

The rules are constructed using these N3 triples and conditional operators. Figure 3 shows a sample rule that infers an attack based on the N3 triple values. The rule states that if the text description consists of some 'vulnerability terms', mentions some 'security exploit', has a text mentioning a certain product (with some specific version) and some process which is being executed, which in turn is also logged by the scanner, and there is an opening up of an out-bound port; then there is a possibility of an attack on the host system with 'means' and 'consequences' mentioned in the ontology.

IV. Prototype and Validation

In order to test the working of our framework, we simulated an attack in a controlled environment on a local network (a private Ethernet based network consisting of 2 desktop machines and IBM ES750 Network Scanner) and observed the results of the system. We simulated a vulnerability present in Adobe Acrobat Reader, CVE-2009-0927 [18], as it was reproducible in a small controlled environment and has the of characteristics necessary for this validation. The vulnerability was a stack based overflow in Adobe Acrobat Reader which allowed remote executors to execute arbitrary code. The attack resided in the Annots.api plug-in of Adobe Acrobat Reader. The vulnerability database of the IBM Proventia Network Scanner was set to the level where it could not detect the attack CVE-2009-0927 directly. The attack payload was embedded in a PDF file and was configured to open up a TCP port for a remote machine on execution. On attack simulation the Network Scanner logged the execution of Annots.api, and thereafter port 80 was opened for a remote machine. However, since the IDS

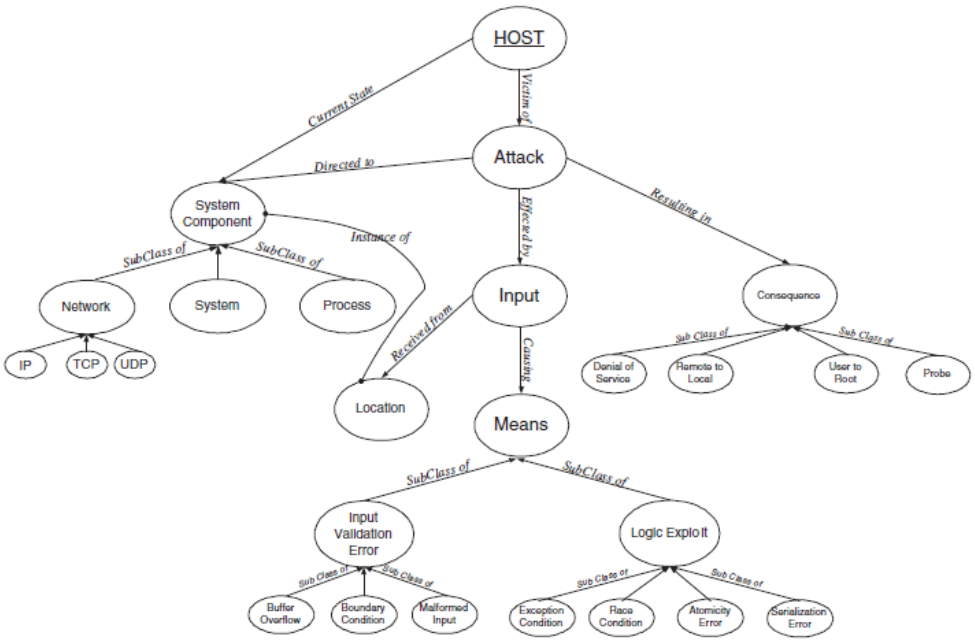

Figure 4. This figure shows a high-level sketch of the IDS OWL ontology from [19].

vulnerability database did not have the signature for the exploit, the attack was not flagged.

We used IBM Proventia ES750 Network Scanner and Snort as the IDS mechanisms. The logs from these systems were also used as packet captures where threats/attacks were not detected. The logs gave us timestamped host and network information like port/protocol of communication, IPs of source and destination, processes/system-calls invoked at the host, etc.

Web sources like vulnerability description feeds (CVE, CCE, CPE, CVSS, XCCDF, OVAL) [10], hacker forums, chat rooms, blogs, etc. were traversed to get a set of named entities out of the unstructured text. The CVE description [18] and a technology blog post [20] were chosen as text from which the named entities were to be extracted. The named entities were then mapped onto the classes in the ontology knowledge-base and were used by the reasoner for decision making.

OpenCalais [21], an open source semantic analysis tool, was used as a web-text analyzer. OpenCalais took unstructured text as input and gave a set of named entities as output. It also tried to group the entities in certain classes. OpenCalais was given text from two web links [18], [20]. Figure 5 shows the texts given to the web-text analyzer. OpenCalais takes the text and attaches semantically rich metadata like the topic being discussed, entities that pop up in the text, events and facts that occur, etc. to the content. Figure 6 shows the named entities extracted by the text-analyzer. The named entities were mapped to the

CVE [18] text description: Stack-based buffer overflow in Adobe Reader and Adobe Acrobat 9 before 9.1, 8 before 8.1.3, and 7 before 7.1.1 allows remote attackers to execute arbitrary code via a crafted argument to the getIcon method of a Collab object, a different vulnerability than CVE-2009-0658.

Juniper Networks Link [20] text description: Acrobat and Reader are prone to a remote code-execution vulnerability because they fail to sufficiently sanitize user-supplied input before using it in the 'strncpy' function in the 'Annots.api' file. This issue affects the 'getIcon()' JavaScript method of a Collab object. Specially crafted arguments can cause a stack-based buffer overflow.

Figure 5. The two boxes are the text descriptions given to the web-text analyzer. The first text box gives an official description of the vulnerability, while the second gives description from some tech-blog web-link

corresponding means, consequences, and targets classes of the ontology.

The reasoner logic found the annots.api dll being executed at the host via the logs received from the network scanner. The log also pointed out the product using this service i.e. Adobe Acrobat Reader. The text data from the web-link [20] also comprised of 'annots.api' in the text. The packet dump showed the opening up of port 80 for clear outbound access after execution of annots.api. The CVE description [18] mentioned 'remote execution' in the text. The rule repository could comprise of a rule which would flag an alert if there is an opening of outbound network port if the application requesting it inherently does

Company: Adobe Systems Incorporated
Technology: Adobe
Named Tags: Computing
Adobe Software
Buffer Overflow
Vulnerability
Arbitrary Code Execution

Company: Adobe Systems Incorporated
Technology: Adobe
api
PDF
Named Tags: Computing
Javascript
Buffer Overflow
Vulnerability
Arbitrary Code Execution

Figure 6. These two examples show the named entities extracted from the text-descriptions by the web-text analyzer.

CVE-2009-0932 text description: Directory traversal vulnerability in framework/Image/Image.php in Horde before 3.2.4 and 3.3.3 and Horde Groupware before 1.1.5 allows remote attackers to include and execute arbitrary local files via directory traversal sequences in the Horde_Image driver name.

Entities Extracted: Technology Internet, Horde, Directory traversal, Computing, Vulnerability, Tree traversal.
Vulnerability Terms: Computer_security_exploits, Web_security_exploits, Machine_code, Injection_exploits, Malware
Scanner Logs Terms: image.php, horde groupware
Inference Rule Applied: Rule1
Result: Attack Flagged

Figure 7. Sample 1 CVE-2009-0932: Description, extracted entities, and result

not require a network access for its execution. The reasoner linked the occurrence of Annots.api in the packet dump from IDS, the opening up of port 80, and the text-analyzer output to conclude that it is a probable attack on the system.

Figure 12 shows the summary of the Adobe attack and the steps executed by the framework to conclude the occurrence of an attack. The named entities extracted from the web-text and the network scanner are asserted into the knowledge-base in the form of N3-triples, and the reasoning logic shown in Figure 3 is used to determine the occurrence of the attack.

The reasoning logic was tested on multiple vulnerabilities that roughly fell in a similar category. 8070 CVE [22] vulnerability text descriptions were chosen, which mentioned vulnerabilities in different products/platforms/applications that resulted in giving the attacker an unauthorized remote access to the host. The rules mentioned in figures 8, 9, 10, and 11 were used to infer the possibility of an attack. The network scanner logs were simulated, i.e. the logs were built up so as to reflect that the data mentioned in the

{ebIDS:webtext ebIDS:hasVulnerabilityTerm “true” .
ebIDS:webtext ebIDS:hasSecurityExploit “true” .
ebIDS:webtext ebIDS:hasText ?Product .
ebIDS:scannerLog ebIDS:hasProduct ?Product .
ebIDS:webtext ebIDS:hasText ?Process .
ebIDS:scannerLog ebIDS:hasProcessExecuted ?Process .
ebIDS:scannerLog ebIDS:hasPortOpened “true” .
ebIDS:scannerLog ebIDS:hasOutboundConnection “true” .
ge(ebIDS:scannerLogOutboundPortOpenTimestamp
ebIDS:scannerLogProcessExecutionTimestamp) .
ebIDS:means ebIDS:maliciousProcess ?Process .
ebIDS:means ebIDS:affectedProduct ?Product}
=>
{ebIDS:system ebIDS:hostUnderAttack “true”.
ebIDS:webText ebIDS:hasProduct ?Product.
ebIDS:webText ebIDS:hasProcessExecute ?Process.
ebIDS:attack ebIDS:hasMeans “Arbitrary Code Execution”.
ebIDS:attack ebIDS:hasConsequence “Unauthorized Remote Access”}

Figure 8. Reasoning Logic Rule 1: Outbound Access (Unauthorized Remote Access) via Malicious Process Execution

{ebIDS:webtext ebIDS:hasVulnerabilityTerm “true” .
ebIDS:webtext ebIDS:hasSecurityExploit “true” .
ebIDS:scannerLog ebIDS:hasProduct ?Product .
ebIDS:webtext ebIDS:hasText ?Product .
ebIDS:scannerLog ebIDS:hasCommandExecuted ?Command .
ebIDS:webtext ebIDS:hasText ?Command .
ebIDS:scannerLog ebIDS:hasPortOpened “true” .
ebIDS:scannerLog ebIDS:hasOutboundConnection “true” .
ge(ebIDS:scannerLogOutboundPortOpenTimestamp
ebIDS:scannerLogProcessExecutionTimestamp) .
ebIDS:means ebIDS:maliciousCommand ?Command.
ebIDS:means ebIDS:affectedProduct ?Product}
=>
{ebIDS:system ebIDS:hostUnderAttack “true”.
ebIDS:webText ebIDS:hasProduct ?Product.
ebIDS:webText ebIDS:hasCommandExecute ?Command.
ebIDS:attack ebIDS:hasMeans “Arbitrary Code Execution”.
ebIDS:attack ebIDS:hasConsequence “Unauthorized Remote Access”}

Figure 9. Reasoning Logic Rule 2: Unauthorized Remote Access/Monitoring via Malicious Command Execution

text to be true. For example: For the text in figure 7, the scanner log was assumed to have information regarding the Horde' software, the corresponding version numbers, and the malicious file name (../Image/Image.php). The reasoning framework which used conjunction of the text, network monitor logs, and the reasoning rules mentioned above was successful in inferring 7120 of the 8070 attacks.

V. Conclusion and Ongoing Work

We have described a semantically rich framework for a situation aware intrusion detection system which can harvest the advantages of heterogeneous data sources to detect the threats, if any. We tested and found the semantic integration of web-text and IDS/IPS sensor information to be effective in detecting threats. The framework performs well, provided

{ebIDS:webtext ebIDS:hasVulnerabilityTerm “true” .
ebIDS:webtext ebIDS:hasSecurityExploit “true” .
ebIDS:scannerLog ebIDS:hasBrowser ?Browser.
ebIDS:webtext ebIDS:hasText ?Browser .
ebIDS:scannerLog ebIDS:hasObject ?Object.
ebIDS:webtext ebIDS:hasText ?Object .
ebIDS:scannerLog ebIDS:hasPortOpened “true” .
ebIDS:scannerLog ebIDS:hasOutboundConnection “true” .
ge(ebIDS:scannerLogOutboundPortOpenTimestamp
ebIDS:scannerLogProcessExecutionTimestamp) .
ebIDS:means ebIDS:maliciousBrowser ?Browser .
ebIDS:means ebIDS:maliciousObject ?Object}
=>
{ebIDS:system ebIDS:hostUnderAttack “true”.
ebIDS:webText ebIDS:hasBrowser ?Browser .
ebIDS:attack ebIDS:hasMeans “Arbitrary Code Execution” .
ebIDS:attack ebIDS:hasConsequence “Unauthorized Remote Access”}

Figure 10. Reasoning Logic Rule 3: Remote Access via Browser

{ebIDS:webtext ebIDS:hasVulnerabilityTerm “true” .
ebIDS:webtext ebIDS:hasSecurityExploit “true” .
ebIDS:scannerLog ebIDS:hasProduct ?Product .
ebIDS:webtext ebIDS:hasText ?Product .
ebIDS:scannerLog ebIDS:hasObject ?Object.
ebIDS:webtext ebIDS:hasText ?Object .
ebIDS:scannerLog ebIDS:hasPortOpened “true” .
ebIDS:scannerLog ebIDS:hasOutboundConnection “true” .
ge(ebIDS:scannerLogOutboundPortOpenTimestamp
ebIDS:scannerLogProcessExecutionTimestamp)}
=>
{ebIDS:system ebIDS:hostUnderAttack “true”.
ebIDS:webText ebIDS:hasProduct ?Product .
ebIDS:webText ebIDS:hasObject ?Object .
ebIDS:attack ebIDS:hasMeans “Arbitrary Code Execution” .
ebIDS:means ebIDS:maliciousProduct ?Product .
ebIDS:means ebIDS:maliciousObject ?Object .
ebIDS:attack ebIDS:hasConsequence “Unauthorized Remote Access”}

Figure 11. Reasoning Logic Rule 4: Unauthorized Remote Access/Monitoring via Malicious Object

that we have a good set of web links which provide some meaningful information regarding the threat/attack and data sources which give entities that map well into the ontology. There is huge potential for making the web-text analyzer stronger by enabling it to crawl the web to harvest new information regarding potential attacks, which in turn can help detect and prevent zero-day attacks. We continue to experiment with integration of newer data sources in the framework and observe the effectiveness of the addition.

ACKNOWLEDGMENT

This work was partially supported by a gift from Northrop Grumman Corporation and grant from Air Force Office of Scientific Research.

Experiment Exploit: Adobe vulnerability CVE-2009-0927 [18]
Vulnerability Overview: Stack-based buffer overflow in Adobe Reader and Adobe Acrobat 9 before 9.1, 8 before 8.1.3 , and 7 before 7.1.1 allows remote attackers to execute arbitrary.
Experiment:
Step 1: Test the exploit on a mock setup.
Step 2: Network traffic collected from the IDS/IPS system for the exploit.
Step 3: Taking a web source mentioning the vulnerability and parsing the data from that source to detect the vulnerabilities.
Step 4: Detecting the product/software-application under threat.
Step 5: Integrating the results of Steps 2, 3 and 4 to give an appropriate alert.

CVE-2009-0927 text description: Stack-based buffer overflow in Adobe Reader and Adobe Acrobat 9 before 9.1, 8 before 8.1.3, and 7 before 7.1.1 allows remote attackers to execute arbitrary code via a crafted argument to the getIcon method of a Collab object, a different vulnerability than CVE-2009-0658. Acrobat and Reader are prone to a remote code-execution vulnerability because they fail to sufficiently sanitize user-supplied input before using it in the ‘strncpy’ function in the ‘Annots.api’ file. This issue affects the ‘getIcon()’ JavaScript method of a Collab object. Specially crafted arguments can cause a stack-based buffer overflow.

Entities Extracted: Technical communication tools, Adobe software, Programming bugs, Arbitrary code execution, Vulnerability, Stack, Buffer overflow, Software testing, Adobe Creative Suite, Adobe Acrobat, Annots.api, Technology Internet.
Vulnerability Terms: Computer_security_exploits, Programming_bugs, Computer_errors, Software_anomalies,
Scanner Logs Terms: acrobat reader, annots.api
Inference Rule Applied: Rule in Figure 3
Result: Threat alert flagged true

Figure 12. Adobe Vulnerability CVE-2009-0927: Description, extracted entities, and result

REFERENCES

[1] “Tracking ghostnet: Investigating a cyber espionage network,” http://www.infowar-monitor.net/2009/09/tracking-ghostnet-investigating-a-cyber-espionage-network.
[2] “Turkish government site hacked amid spat with china,” http://www.pcworld.com/businesscenter/article/168359/turkish_government_site_hacked_amid_spat_with_china.html.
[3] “Justice: Hackers steal 40 million credit card numbers,” http://articles.cnn.com/2008-08-05/justice/card.fraud.charges_1_card-numbers-debit-magnetic-strips?_s=PM:CRIME.
[4] “Snort,” http://www.snort.org/.
[5] “Internet security systems x-force security threats,” http://xforce.iss.net.
[6] “Wireshark,” http://www.wireshark.org/.
[7] “Nagios,” http://www.nagios.org/.

[8] "Cacti," http://www.cacti.net/.
[9] V. Mulwad, W. Li, A. Joshi, T. Finin, and K. Viswanathan, "Extracting Information about Security Vulnerabilities from Web Text," in *Proceedings of the Web Intelligence for Information Security Workshop*. IEEE Computer Society Press, August 2011.
[10] "National vulnerability database," http://nvd.nist.gov.
[11] V. Khadilkar, J. Rachapalli, and B. Thuraisingham, "Semantic web implementation scheme for national vulnerability database," Univ. of Texas at Dallas, Tech. Rep. UTDCS-01-10, 2010.
[12] J. Undercoffer, A. Joshi, T. Finin, and J. Pinkston, "Using DAML+OIL to classify intrusive behaviours," *The Knowledge Engineering Review*, vol. 18, pp. 221–241, 2003.
[13] J. Undercoffer, A. Joshi, and J. Pinkston, "Modeling Computer Attacks: An Ontology for Intrusion Detection," in *Proc. 6th Int. Symposium on Recent Advances in Intrusion Detection*. Springer, September 2003.
[14] J. Undercofer, "Intrusion Detection: Modeling System State to Detect and Classify Aberrant Behavior," Ph.D. dissertation, University of Maryland, Baltimore County, February 2004.
[15] "Top command (linux)," http://linux.die.net/man/1/top.
[16] "Monit," http://mmonit.com/monit/.
[17] "Cisco hardware sensor," http://www.cisco.com/en/US/products/hw/vpndevc/ps4077/index.html.
[18] "Adobe acrobat vulnerability cve-2009-0927," http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-0927.
[19] http://ebiquity.umbc.edu/ontologies/cybersecurity/ids/.
[20] "Juniper website text description of cve-2009-0927," http://www.juniper.net/security/auto/vulnerabilities/vuln34169.html.
[21] "Opencalais," http://opencalais.com/.
[22] "Common vulnerabilities and exposures," http://cve.mitre.org/.