Situation Aware Intrusion Detection Model

by

Sumit S. More

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfilment
of the requirements for the degree of
MS
2012

Dedicated to Aai

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Dr. Anupam Joshi. His constant support and guidance have been instrumental in completion of this thesis. Constructive discussions and innovative suggestions by him during those discussions have been elementary during the course of my work at the Ebiquity Lab. He has always allowed me to explore new concepts and pursue newer research problems. I am thankful for his words of advice and many skills I have gained by working with him. I would like to thank to Dr. Tim Finin for all his valuable inputs throughout my work at Ebiquity lab. Collectively, I would also like to thank my thesis committee members Dr. Anupam Joshi, Dr. Tim Finin, and Dr. Yelena Yesha for their time, suggestions, and for graciously agreeing to be on my committee, and always making themselves available. All the Ebiquity group members have been extremely supportive in building an atmosphere conducive to research. It has been a great joy working with them throughout the year. I extend my sincere thanks to National Science Foundation, the Air Force Office of Scientific Research, and Northrop Grumman Corporation for funding this research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Cyber-crimes are being used to assist in activities like espionage, politically motivated attacks and credit card fraud. For instance, the 'Tracking GhostNet' investigation uncovered a network of over 1,295 infected hosts in 103 countries, where high-value targets were compromised at ministries of foreign affairs, embassies, international organizations, news media, and NGOs [1]. According to infowar-monitor, in March 2011 forty South Korean government websites (including the Presidential Office, National Assembly, and the U.S. Forces Korea) and corporate institutions (five major banks and securities companies) came under a large scale DDoS attack. [2]. In September 2011 CIA had reported that the tally of digital certificates stolen from a Dutch company in July had exploded to more than 500, including ones for intelligence services like the CIA, the U.K.'s MI6 and Israel's Mossad, a Mozilla developer said Sunday [3]. the official Linux Kernel Organization had confirmed that multiple servers used to maintain and distribute the Linux operating system were infected with malware that gained root access, modified system software, and logged passwords and transactions of the people who used them [4]. Flash security bulletin reported critical vulnerabilities in Adobe Flash Player 10.2, which is vulnerable to unauthorized remote access [5]. Adobe security bulletin had warned users of their nearly ubiquitous Adobe Reader software of a new zero-day vulnerability being exploited in the wild [6]. 'Track-

ing GhostNet' report described how Tibetan computer systems were compromised giving attackers access to potentially sensitive information [1], news streams had reported how an attacker defaced the web-site of Turkey's embassy in China [7] and CNN had reported how hackers stole 40 million credit card numbers [8].

Most of the above mentioned high risk cyber-attacks are zero-day attacks. A zero-day attack is cyber-attack that exploits a security vulnerability as soon as the vulnerability gets exposed [9]. In a normal scenario, a software or hardware containing a potential security threat is reported to the manufacturer. The manufacturer then fixes the issue and provides a patch, software update or replacement product to the user. Also, a signature of the attack is developed and is added to the corresponding defence mechanism for preventing the system from being attacked and exploited using the same strategy. If the potential attacker comes to know about the vulnerability after the manufacturer releases a patch for it, then he/she can't exploit the vulnerability easily. Sometimes however, the attacker may be the first to detect such a vulnerability, and exploit it to cause harm to the systems using the software/hardware. Since the vulnerability is not known in advance, there is no way to guard the system against the exploit before an attack exploiting it occurs.

Intrusion detection and prevention systems are software-hardware mechanisms which monitor the computer-infrastructure for malicious activities or rule-violations, and flag an alert if such violations are detected [10]. Intrusion detection systems are broadly classified into two types namely: (i) Network intrusion detection system (NIDS) and (ii) Host-based intrusion detection system (HIDS). Network intrusion detection systems are independent platforms that identify intrusions by examining the network traffic and monitors multiple hosts. Network intrusion detection systems are generally stationed at network hub, network switch etc., which are the choke points in the network. The NIDS sensors capture all the network traffic and analyse the content of individual packets for malicious traffic. An example of a NIDS is Snort [11]. Host-based intrusion detection systems consist of an agent

on a host that identify intrusions by analysing system calls, application logs, file-system modifications (binaries, password files, capability databases, Access control lists, etc.) and other host activities and state. In a HIDS, sensors usually consist of a software agent. Examples of HIDS are Tripwire [12] and OSSEC [13]. Intrusion detection systems use one of two detection techniques: (i) Statistical anomaly-based IDS determines normal network activity like what sort of bandwidth is generally used, what protocols are used, what ports and devices generally connect to each other and alert the administrator or user when traffic is detected which is anomalous(not normal). (ii) Signature based IDS monitors packets in the Network and compares with pre-configured and pre-determined attack patterns known as signatures.

State of the art intrusion detection and prevention systems (IDPS) available today are rule or signature based systems, i.e. they make sure that the computer-systems conform to a certain specification of a security policy. A security policy is a set of rules that specify the authorizations, prohibitions and obligations of agents (including both users and applications) that can access to the computer system. The IDPS mechanisms monitor the cyber-infrastructure, and identify any malicious activities based on the signature of the 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 foundation to very useful spatial and temporal information of the system activities, which can be useful in tracking threats and attacks. Furthermore, even if the signature of the attack becomes available, the intrusion detection system needs to be updated with this new signature, which requires patching the existing system with a newer version of the software which consists of the new signatures.

Information about a system can be inferred by analysing and reasoning over the information gained from different data sources such as the network activity behavior, host behavior, and malicious activities. Moreover, a semantic integration of this information can help in better inferencing of means and consequences of a cyber-attack or threat. These possibilities motivate us to propose a semantically rich approach to intrusion detection, where gathering of data from different sources, integrating them, and reasoning over an information-rich knowledge-base result in detection of attack or threat whose signature is not available. We present a framework for this ontological approach and demonstrate its working.

Chapter 2

BACKGROUND AND RELATED WORK

In this chapter we discuss the motivation behind this thesis work and some related work that forms the basis of this research.

2.1 Motivation

Intrusion detection and prevention systems like Snort [11] and IBM X-Force [14] 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 threat in the infrastructure if the signature of the threat is not present in the system database. Many host and network based activity monitors such as Wireshark [15], Nagios [16] and Cacti [17] 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 event satisfies certain rule. Unless the

behavior of the attack is known, these systems cannot detect it. Thus, these systems have a limited threat detection ability as they lack clear description of relations between intrusion behavior.

Security information and event (SIEM) solutions available today do a reasonable job in data aggregation, correlation, dash-boarding, and alerting. But they still have limited capability in correlating the events that are spatially and temporally apart in-order to infer vulnerability in the system and detect threat if any. SIEMs also miss out on information-rich source of the web. Inclusion of the web data channel can help in building better and stronger SIEM correlation models. This work demonstrates how aggregation of web-data along with data channels from host and network activity monitors can lead to better intrusion detection model with the help of semantic technologies.

2.2 Related Work

Raskin et al. introduce an ontological semantic approach to information security. The approach pursues the dual goals of inclusion of natural language data sources as an integral part of the overall data sources in information security applications, and formal specification of the information security community know-how for the support of routine and time-efficient measures to prevent and counteract computer attacks [18].

Wei et al. present an ontology-based intrusion detection model with advantages of hierarchical and cooperative models. The paper mentions the limitation of current intrusion detection systems in describing relations between intrusion behavior, and describes 'ontology' as an emerging knowledge expression technology that helps determine precise meaning of the concept through strict definition and the concept relations, which express mutually accepted and shared knowledge. The paper further emphasizes on how ontology can make a the representation of intrusion systematized and organised, which can sig-

nificantly reduce the characteristics needed for describing and improve the efficiency of intrusion detection systems [19] .

Undercoffer et al. emphasize 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 information at hand [20], [21]. They also present 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 [21], and state the benefits of transitioning from taxonomies to ontologies and ontology specification languages, which are able to simultaneously serve as recognition, reporting and correlation languages [22].

Many data sources on the web today are comprised of information related to intrusions and attacks in different levels of verbosity. Varish et al. describe a framework for extracting information about vulnerabilities and cyber attacks from different unstructured data sources like vulnerability description feeds (CVE, CCE, CPE, CVSS, XCCDF, OVAL) [23], hacker forums, chat rooms, blogs, etc. and informing the expert about it. The framework was found to be useful in adding structure to already existing vulnerability descriptions as well as detecting new ones [24]. The framework used semantic web technologies RDF/OWL for generating machine understandable assertions with the help of appropriate entities and concepts in the linked data cloud.

Khadilkar et al. explain the inadequacy of relational data models in syntactic interoperability, and highlight the advantages of semantic web technologies to facilitate easy data sharing and interoperability [25]. The report explains the importance of semantic model, presents an ontology for a real world application based on National Vulnerability Database, and demonstrate building of a knowledge-base from a structured data set.

Integration of conventional signature-based intrusion detection 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.

The understanding of a few background concepts of Semantic Web and Intrusion Detection should make this thesis easier to understand.

2.3 Semantic Web and Ontology

The Semantic Web refers to both a vision and a set of technologies. The vision was first articulated by Tim Berners-Lee as an extension to the existing web in which knowledge and data could be published in a form easy for computers to understand and reason with. Doing so would support more sophisticated software systems that share knowledge, information and data on the Web just as people do by publishing text and multimedia. Under the stewardship of the W3C, a set of languages, protocols and technologies have been developed to partially realize this vision, to enable exploration and experimentation and to support the evolution of the concepts and technology [26].

In a semantic model information is represented as a set of assertions known as statements. A statement (also known as triple) is made up of three parts namely: subject, predicate and object. A subject of a statement is an entity that the statement describes. A predicate describes a relationship between a subject and an object. The object being the value that the subject takes for that particular predicate. Statements of this type naturally form a directed graph, with subjects and objects of each statement as nodes and predicates as edges. This is the data model used in the Semantic Web [27] and it is formalized in the

language called Resource Description Framework (RDF) [28].

An ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an ontology is systematic account of existence [29]. For knowledge-based systems, what 'exists' is exactly that which can be represented. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. This set of objects, and the describable relationships among them, are reflected in the representational vocabulary with which a knowledge-based program represents knowledge. Thus, we can describe the ontology of a program by defining a set of representational terms. In such an ontology, definitions associate the names of entities in the universe of discourse (e.g., classes, relations, functions, or other objects) with human-readable text describing what the names are meant to denote, and formal axioms that constrain the interpretation and well-formed use of these terms.

An Ontology is a structure capturing semantic knowledge about a certain domain by describing relevant concepts and relations between them [30]. In general, an ontology describes formally a domain of discourse. Typically, an ontology consists of a finite list of terms and the relationships between these terms. The terms denote important concepts (classes of objects) of the domain. The relationships typically include hierarchies of classes. A hierarchy specifies a class C to be a subclass of another class C' if every object in C is also included in C'. Apart from subclass relationships, ontologies may include information such as:

- properties (X teaches Y),
- value restrictions (only faculty members may teach courses),
- disjointness statements (faculty and general staff are disjoint),
- specifications of logical relationships between objects (every department must in-

clude at least ten faculty members).

At present, the most important ontology languages for the Web are the following:

- RDF is a data model for objects (resources) and relations between them; it provides a simple semantics for this data model; and these data models can be represented in an XML syntax.

- RDF Schema is a vocabulary description language for describing properties and classes of RDF resources, with a semantics for generalization hierarchies of such properties and classes.

- OWL is a richer vocabulary description language for describing properties and classes, such as relations between classes (e.g., disjointness), cardinality (e.g., exactly one), equality, richer typing of properties, characteristics of properties (e.g., symmetry), and enumerated classes.

2.4 Intrusion and Intrusion-Detection

Cyber attacks/intrusions take advantages of vulnerabilities in the system and exploit them to propagate and inflict damage to the system. Such vulnerabilities include:

- User apathy: Even though every computer user is aware of the threat of cyber attacks/intrusions, she feels that ”it’s not going to happen to me”. Computer users share software without checking for infection, they ignore suspicious behavior that may indicate the presence of a threat, and they don’t spend the time to learn and apply basic security measures.

- Insufficient security control: Many computers, especially personal computers, are not equipped with hardware and software features that help in detecting and isolating threats.

- Weakness in system software: Modern operating systems are extremely complex and are implemented and maintained by large teams of programmer. Vulnerabilities and weak points are discovered all the time in this type of software. Quite often a discovery is made by a security expert or a clever user who then notifies the manufacturer of the operating system. Sometimes, a weakness is discovered by a user with a malicious intent, who immediately sets up to exploit it.

- Unauthorized use: There are those who regard breaking into a computer as a challenge, and the more secure and secret the computer, the greater the challenge. Once a hacker manages to break into a computer, the temptation to cause havoc is great, often too great.

- Anonymity of networks: Before the era of computer networks, a person with malicious intent had to actually walk into a computer center in order to do damage. Nowadays, with the prevalence of networks, attackers have the advantage of anonymity.

A few ways to strengthen the defences against the cyber attacks are:

- Training seminars: From time to time, the user must be trained in the basics of security which include understanding of security policies and procedures. This might include topics such as:

  - A background on latest viruses, how they are planted, how they propagated, the types of damage the inflict, and how to detect their presence.
  - Software vulnerabilities exploited by viruses in the past.
  - Company security policies and contingency procedures.

- Any decisions pertaining to the acquisition of new software and hardware should involve security experts.

- Monitoring user and network activity: Special software can monitor the activities of the various user computers and report any suspicious activity to a central monitoring facility.

- Limited sharing: An organisation should try to limit the sharing of computing resources among its members, and the sharing of data between itself and the outside world.

Intrusion detection and prevention systems are an important component of defensive measures protecting computer systems and networks from abuse. They play a significant role in security architecture of a cyber-system. Intrusion detection has been an active field of research for about two decades, starting in 1980 with the publication of John Andersons [31] Computer Security Threat Monitoring and Surveillance which was one of the earliest papers in the field. Dorothy Dennings seminal paper, [32] published in 1987, provided a methodological framework that inspired many researchers and laid the groundwork for many commercial products.

Chapter 3

SYSTEM ARCHITECTURE

The proposed system architecture of the framework is shown in Figure 3.1. The architecture is broadly divided into following sections:

1. Data streams.
2. Ontology and knowledge-base.
3. Reasoning logic/rules.

The data-streams include web-text coming from on-line standard vulnerability description repositories, online help-forums, online hacker blogs and forums, chat rooms, social media streams, system monitoring mechanisms like Wireshark, Snort, corporate network scanners to name a few. The information extracted from these channels is collected in a generic format. The information collected from different data streams is asserted into the knowledge-base as per the ontology rules. The reasoning logic then integrates the information built in the knowledge-base to infer possibility of a cyber threat/attack.

The following sections explain the individual modules of the system in depth.

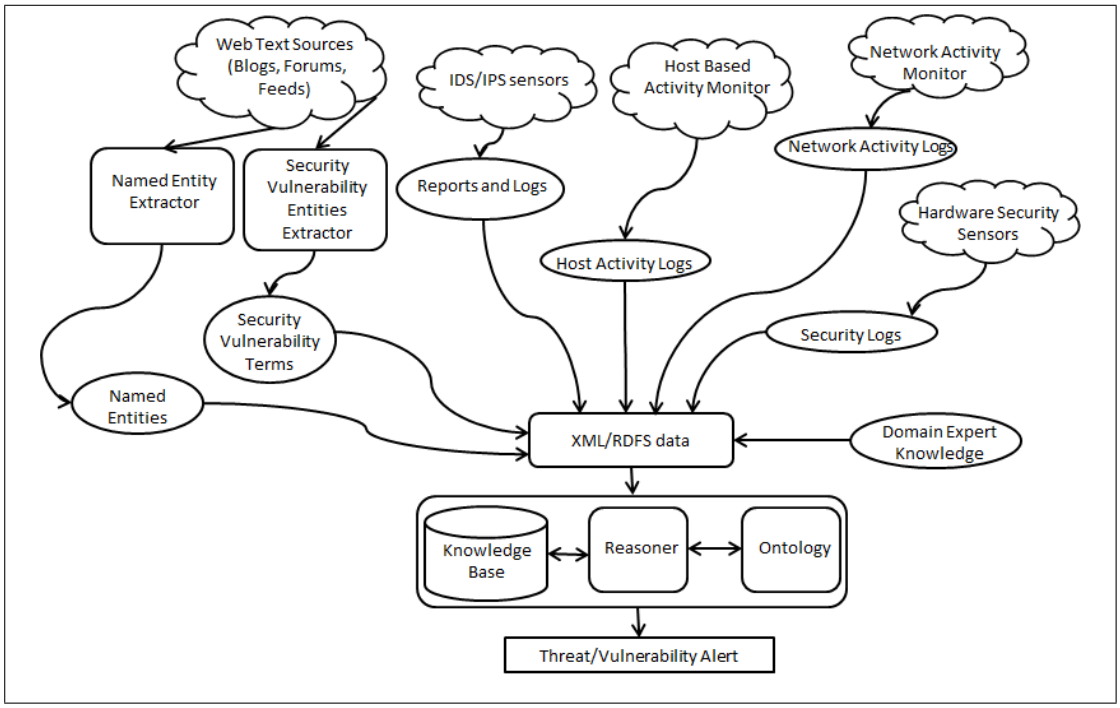

FIG. 3.1. System Architecture.

3.1 Data Streams

The data streams include different channels that provide useful information related to an attack. These data channels may include:

- Streams from host system which provide information related to the activities/processes that are executing at the host like logs from top [33] and monit [34].
- Streams from network based activity monitors like Wireshark [15], Nagios [16], Cacti [17] 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 [35].
- 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/unstructured text data like threat/attack descriptions from vulnerability reporting web-portals [36].

The items of interest are parsed out from these data streams, and are asserted on facts in our knowledge-base.

As representative examples, for this work we have used web-text streams from [36], [23], data logs from IBM ES750 Network Scanner [37], [11], and [15].

3.1.1 Web-Text Analysis Module

Structured and unstructured data from different web sources is parsed through information extraction modules to get the entities of importance from them. Figure 3.2 shows the web-text analysis module of the framework. Following steps are performed during this phase:

- The web-text comprising of potential vulnerability information is given to the 'Named Entity Extractor' module. This module parses the text, classifies and returns Resource Description Framework formatted output identifying the entities, facts and events within the text.
- The UMBC Ebiquity lab has been working on extracting information from unstructured web-text, and we use the previous work done in the lab for this research work. The 'Vulnerability Terms Extractor' module is one such module. It uses Wikitology, a general purpose knowledge base derived from Wikipedia, to extract concepts

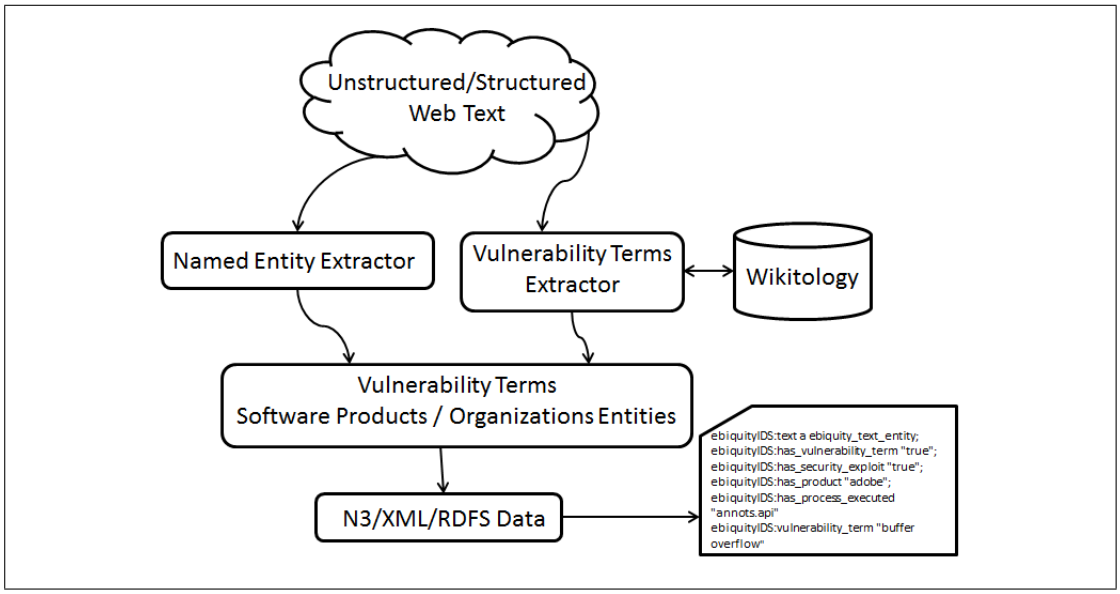

FIG. 3.2. Web-Text Analysis Module.

that describe specific vulnerabilities and attacks, maps them to related concepts from DBpedia and generate machine understandable assertions. This is a 3 step process:

- Identifying potential vulnerability descriptions.
- Identifying potential terms from the text-descriptions which are either of relevant concepts, entities, relations, and events.
- Generating machine understandable assertions encoded in Semantic Web language OWL.

The vulnerability terms and entities extracted by the above modules are used as inputs to the reasoning logic.

Named Entity Extractor Module: 'Named Entity Extractor' module aims at extracting and automatically tagging entities, facts and events from the text input. This module deals with unstructured text and processes it using natural language processing (NLP)

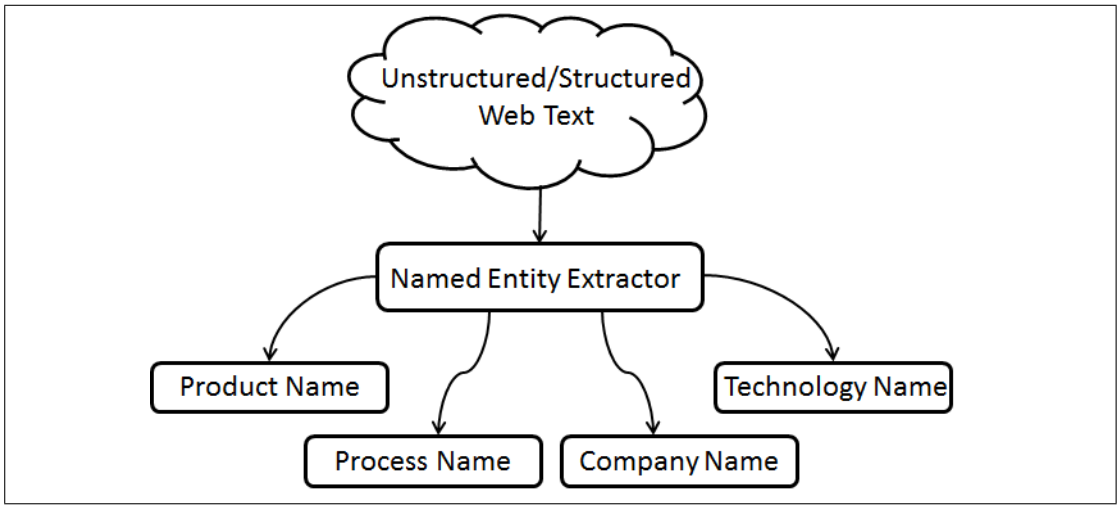

FIG. 3.3. Named Entity Extractor Module.

and machine learning (ML) algorithms, to returns RDF-formatted entities, facts and events. Figure 3.3 shows the working of the 'Named Entity Extractor' module.

The web-text given to this module is a combination of Common Vulnerabilities and Exposures (CVEs) chosen from National Vulnerability Database [36] [23], and some other web-links describing cyber attacks.

Common Vulnerabilities and Exposures (CVE) is a dictionary of common names (i.e., CVE Identifiers) for publicly known information security vulnerabilities. CVE's common identifiers make it easier to share data across separate network security databases and tools, and provide a baseline for evaluating the coverage of an organization's security tools. Figure 3.4 shows a sample CVE structure [38]. CVE basically has the following properties:

- One name for one vulnerability or exposure.
- One standardized description for each vulnerability or exposure.
- A dictionary rather than a database.
- How disparate databases and tools can "speak" the same language.

CVE ID: CVE-2009-0927
Status: Candidate
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.
References:

- BUGTRAQ:20090324 ZDI-09-014: Adobe Acrobat getIcon() Stack Overflow
- URL:http://www.securityfocus.com/archive/1/archive/1/502116/100/0/threaded
- MISC:http://www.zerodayinitiative.com/advisories/ZDI-09-014
- CONFIRM:http://www.adobe.com/support/security/bulletins/apsb09-04.html
- XF:adobe-unspecified-javascript-code-execution(49312)
- URL:http://xforce.iss.net/xforce/xfdb/49312

Phase: Assigned (20090317)

FIG. 3.4. Sample CVE description

- The way to interoperability and better security coverage.
- A basis for evaluation among tools and databases.
- Free for public download and use.
- Industry-endorsed via the CVE Editorial Board and CVE-Compatible Products.

The entities extracted from the web-text help to get information such as software/hardware name, product name, company name, process name etc. Figure 3.5 shows a sample web-text and the entities extracted by the 'Named Entity Extractor'

Vulnerability Terms Extractor Module: 'Vulnerability Terms Extractor' module ingests text snippets found on the web and generate assertions about vulnerabilities, threats and attacks found in those texts. The algorithm for vulnerability terms' extraction uses the

CVE [38] 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.

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

FIG. 3.5. A sample text description and the named entities extracted by the 'Named Entity Extractor' module.

knowledge from Wikitology along with a computer security exploit taxonomy extracted from Wikipedia to identify vulnerabilities, threats and attacks. The algorithm queries the text description against the 'contents' field of Wikitology. The knowledge-base returns a ranked list of Wikipedia concepts matching the query. These results include the concepts of interest to us like vulnerabilities, threats and attacks. The vulnerability terms' extraction is done in following steps:

1. The unstructured web-text is classified into relevant and irrelevant classes depending on the content of the text. If the content contains some cyber attack or threat related it is classified as a relevant text snippet.

2. Using the knowledge from Wikitology, Wikipedia, and a computer security taxonomy, the reasoning mechanism detects the terms denoting the vulnerabilities in the text.

3. The vulnerability terms are collected in form of RDF triples encoded in N3 and stored in the knowledge-base.

Figure 3.6 shows the working of this module.

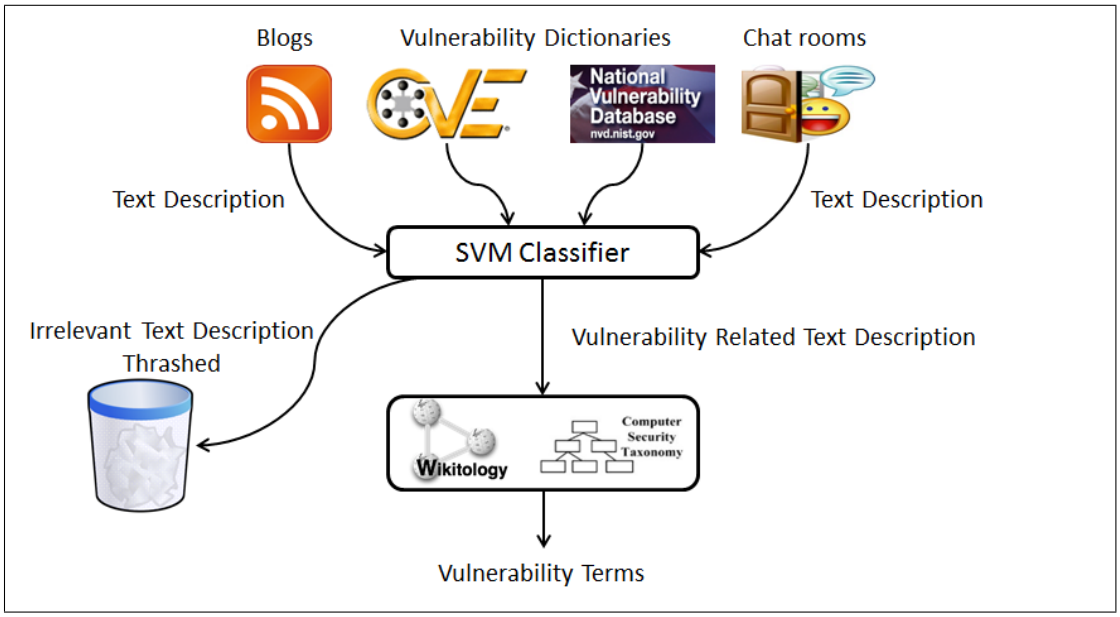

FIG. 3.6. Vulnerability Terms Extractor Module

3.1.2 Sensor Log Collection Module

The host system activities like processes running on the system, network activity (network protocols, packets sent, ports/IPs in use), etc. is monitored by tools like Snort [11], Wireshark [15], and network scanners like IBM ES750 [37]. Figure 3.7 shows the 'Sensor Log Collection Module' and its working.

Following systems were used for this work:

1. IBM ES750 Network Scanner: IBM ES750 network scanner has following properties:

    - Detects over 1,500 device types and over 3,400 service types to accurately identify the components of the network.
    - Provides multi-source discovery by using different asset identification techniques that can detect newly connected devices and previously undiscovered assets.

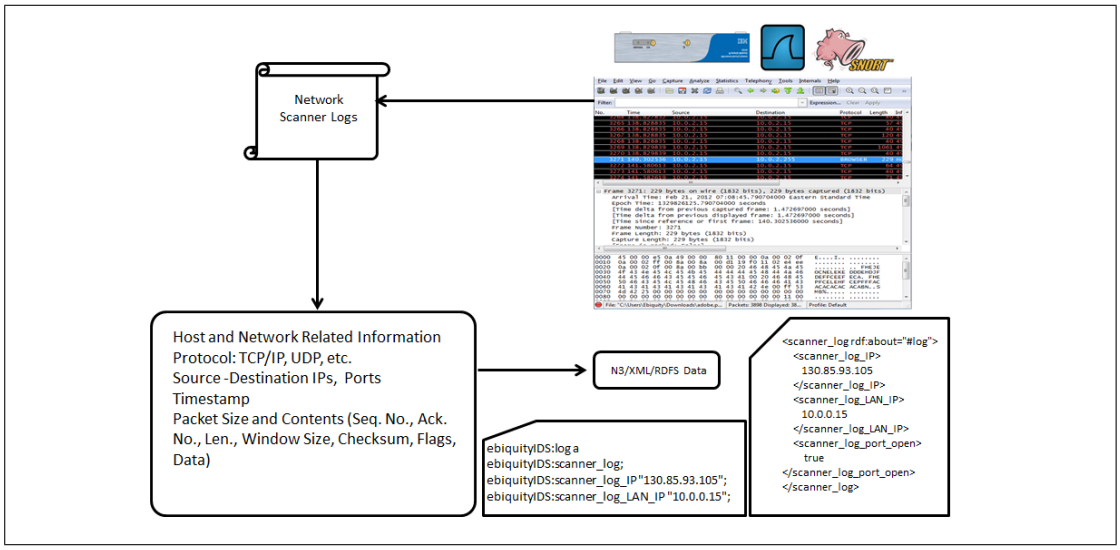

FIG. 3.7. Sensor Log Collection Module.

- Capable of conducting emergency scans to provide quick, ad-hoc scan results of your network when the need arises.
- Can group and report by geography, network layout, business system or any other useful grouping of assets
- Performs specific tests in a non-impacting manner to analyse the effects of a real attack without exposing your network to real threats

2. Snort [11]: Snort is an open source network intrusion prevention system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching, and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more.

3. Wireshark [15]: Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, and analysis. Wireshark is software that ”understands” the structure of different networking protocols. Thus, it is able to display the encap-

sulation and the fields along with their meanings of different packets specified by different networking protocols. Properties of Wireshark are as follows:

- Data can be captured ”from the wire” from a live network connection or read from a file that recorded already-captured packets.
- Live data can be read from a number of types of network, including Ethernet, IEEE 802.11, PPP, and loopback.
- Captured network data can be browsed via a GUI, or via the terminal (command line) version of the utility, TShark.
- Captured files can be programmatically edited or converted via command-line.
- Plug-ins can be created for dissecting new protocols.
- Raw USB traffic can be captured.

3.2 Ontology and Knowledge-Base

This module comprises of the ontology, and the knowledge-base of the semantic analysis part of the system. We have customised the ontology and knowledge-base proposed by UMBC Ebiquity alumni Dr. Undercoffer [21] [39], as per the requirements of this thesis work. We added properties related to the ‘system’, ‘attack’, ‘means’ and ‘consequences’ classes of this ontology in a way which could help capture the information coming in from the different data-channels. Example: Host system related information like names of processes executing on the system, names of application products installed on the system, process execution timestamps, network port opening timestamps etc.

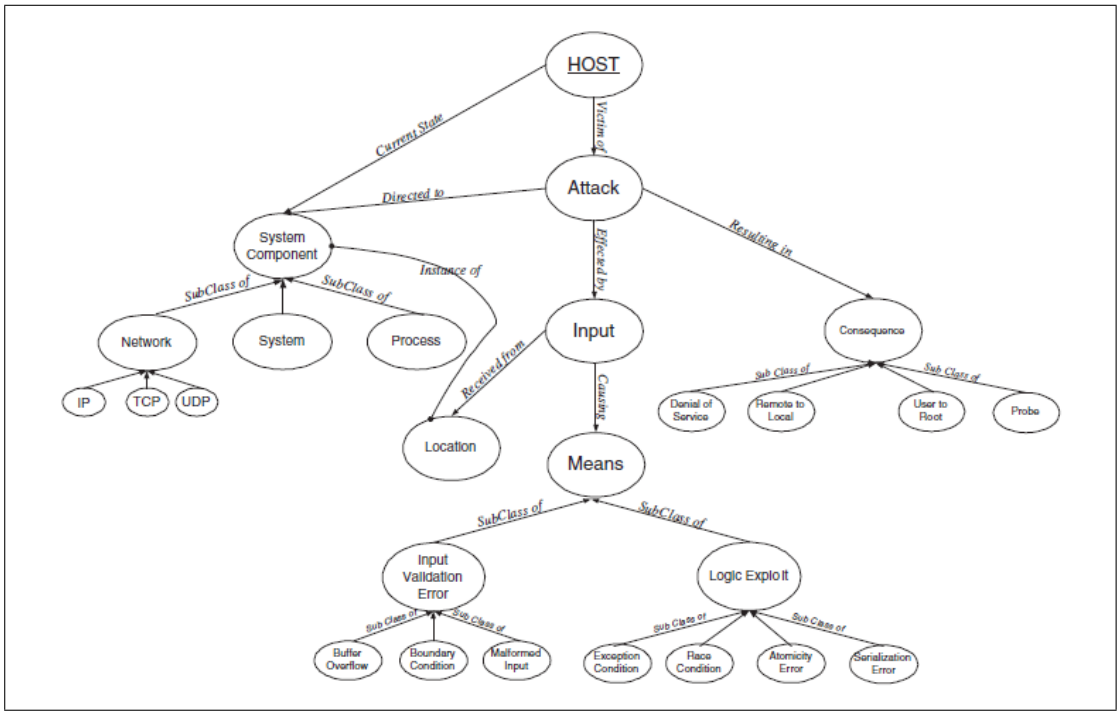

FIG. 3.8. Intrusion Detection Ontology [22]

3.2.1 Ontology

The ontology comprises of 3 fundamental classes: means', 'consequences', and 'host'. The 'means' class encapsulates the ways and methods used to perform an attack, the 'consequences' class encapsulates the outcomes of the attack, and the 'host' 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 'host' class consists of sub-classes like 'SystemUnderDoSAttack', 'SystemUnderProbe', 'SystemUnderSynFloodAttack', etc. Figure 3.8 shows the baseline ontology that forms the backbone of the framework.

Class *Host* has the predicates *Current State* and *Victim of*. *Current State* ranges over

System Component and *Victim of* ranges over the class *Attack*. As earlier stated, the predicate defines the relationship between a subject and an object. The System Component class is comprised of the following subclasses:

1. Network. This class comprises of network related information like IP addresses, ports, protocols in use etc. For example, we can have TCP/IP protocol stack being used for communication between the host system with IP address 10.0.1.25 on port 80, and the external world.

2. System. This includes attributes representing the host. It includes attributes representing operating system, memory, disk capacity, processes being executed at the host.

The class *Attack* has the properties *Directed to*, *Effected by*, and *Resulting in*. This construction is predicated upon the notion that an attack consists of some input which is directed to some system component and results in some consequence. Accordingly, the classes *Host*, *Input*, and *Consequence* are the corresponding objects.

The class *Consequence* is comprised of several subclasses which include:

1. Denial of Service: The attack results in a Denial of Service to the users of the system. The denial of service may be because the system was placed into an unstable state or all of the system resources may be consumed by meaningless functions.

2. User Access: The attack results in the attacker having access to services on the target system at an unprivileged level.

3. Root Access: The attack results in the attacker being granted privileged access to the system, consequently having complete control of the system.

4. Probe: This type of an attack is the result of scanning or other activity wherein a profile of the system is disclosed.

Finally, the class *Input* has the predicates *Received from* and *Causing* where *Causing* defines the relationship between the *Means* of attack and some input and *Received from* defines the relationship between *Input* and *Location*. The class *Location* is an instance of *System Component* and is restricted to instances of the Network and Process classes.

We define the following subclasses for *Means* of attack:

1. Input Validation Error: An input validation error exists if some malformed input is received by a hardware or software component and is not properly bounded or checked. This class is further sub-classed as:

   (a) Buffer Overflow: The classic buffer overflow results from an overflow of a static-sized data structure.

   (b) Boundary Condition Error. A process attempts to read or write beyond a valid address boundary or a system resource is exhausted.

   (c) Malformed Input. A process accepts syntactically incorrect input, extraneous input fields, or the process lacks the ability to handle field-value correlation errors.

2. Logic Exploits: Logic exploits are exploited software and hardware vulnerabilities such as race conditions or undefined states that lead to performance degradation and/or system compromise. Logic exploits are further sub-classed as follows:

   (a) Exception Condition: An error resulting from the failure to handle an exception condition generated by a functional module or device.

   (b) Race Condition: An error occurring during a timing window between two operations.

   (c) Serialization Error: An error that results from the improper serialization of operations.

(d) Atomicity Error: An error occurring when a partially-modified data structure is used by another process. An error occurring because some process terminated with partially modified data where the modification should have been atomic.

3.2.2 Knowledge-base

The entities that are collected from different data streams are asserted into one of the classes on the basis of the properties of the class and meaning of the entity. For instance, 'annots.api executable' is an object of a class 'process under stack overflow', which is a subclass of 'buffer overflow', which in turn is a subclass '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 collected in RDF format and encoded in Notation-3 format. Notation-3 or N3 is a shorthand non-XML serialization of Resource Description Framework models, designed with human-readability in mind [40]. It is a language which is a compact and readable alternative to RDF's XML syntax, and also is extended to allow greater expressiveness. It has subsets, one of which is RDF 1.0 equivalent, and one of which is RDF plus a form of RDF rules. An N3 document encodes a set of statements, and its meaning is the conjunction of 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 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 knowledge-base is built up by collecting the information in RDF and encoding it in N3 triples. The entities collected from different data sources can be properties of a

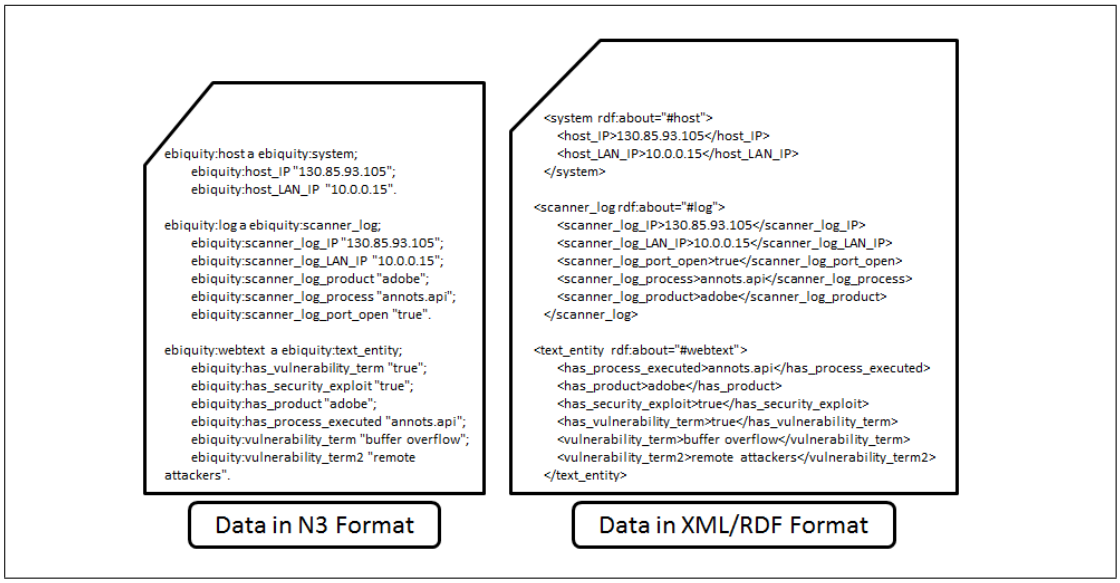

FIG. 3.9. Extracted RDF information encoded in N3 format.

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 can be object of 'process under buffer overflow' class in the ontology). Figure 3.9 shows the N3 triples of the sample information extracted, and the corresponding XML/RDF structure.

3.3 Reasoning Logic

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. Figure 3.10 shows the workflow of the reasoning logic module.

The extracted RDF information encoded in N3 triples format is used by the reasoning logic 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 analyser are

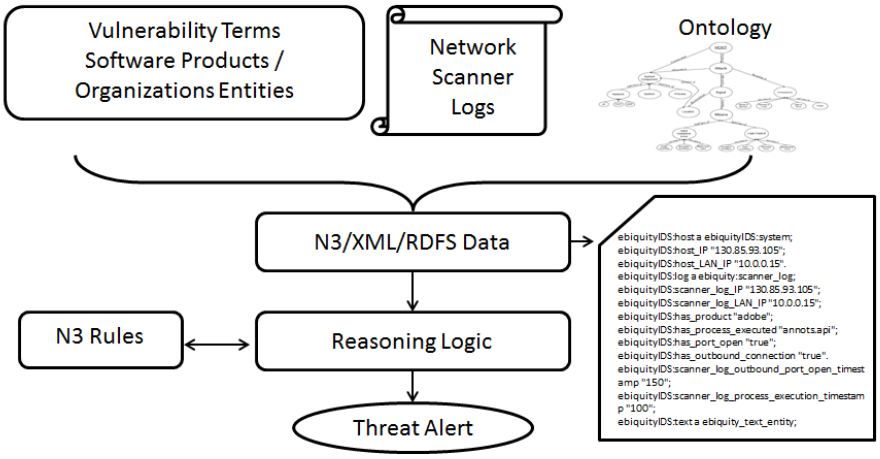

FIG. 3.10. Reasoning Logic WorkFlow.

used for continuous evolution of the knowledge-base.

The reasoning logic is a set of RDF rules defined in N3 format. 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 be 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 derived rules resulting from the chain of implication, the value of those triples will be returned.

Table 3.1 shows sample predicates used in this work.

The rules are constructed using these N3 triples and conditional operators. Figure 3.11 shows a sample rule that infers an attack based on these triple values. The rule states that if the text description consists of some ‘vulnerability terms’, and mentions some ‘security exploit’, and has a text mentioning a certain product (with some specific version) and some

Subject Name	Predicate Name	Data Type
Host	hostIP	String
Host	hostUnderAttack	Boolean
Scanner	scannerLogHostIP	String
Scanner	scannerLogProduct	String
Scanner	scannerLogProcess	String
Scanner	scannerLogPortOpen	Boolean
Webtext	hasVulnerabilityTerm	Boolean
Webtext	hasSecurityExploit	Boolean
Webtext	entity	String

Table 3.1. Reasoning Logic Predicates.

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.

```
{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. }
```

FIG. 3.11. Sample Reasoning Logic Rule

Chapter 4

SYSTEM IMPLEMENTATION

4.1 Named Entity Extractor

We used OpenCalais for entity extraction from the web text. Opencalais identifies and tags entities, facts and events in the text. At its core lays the OpenCalais web service. The web service is an API that accepts unstructured text, processes it using Natural Language Processing (NLP) and Machine Learning (ML) algorithms, returns RDF-formatted entities, facts and events.

The basic working of the OpenCalais API is as follows:

1. An API key a string that uniquely identifies the specific instances of the application that uses the service is requested for using the API.

2. After obtaining the key, we can submit content to the Calais Web service. The methods supported by the OpenCalais Web Service API are: SOAP, REST and HTTP Traffic Compression. The API default usage quotas are 50,000 transactions per day, 4 transactions per second.

3. Calais tags each person, place, fact and event in the content, making it machine-readable and interoperable on the Web.

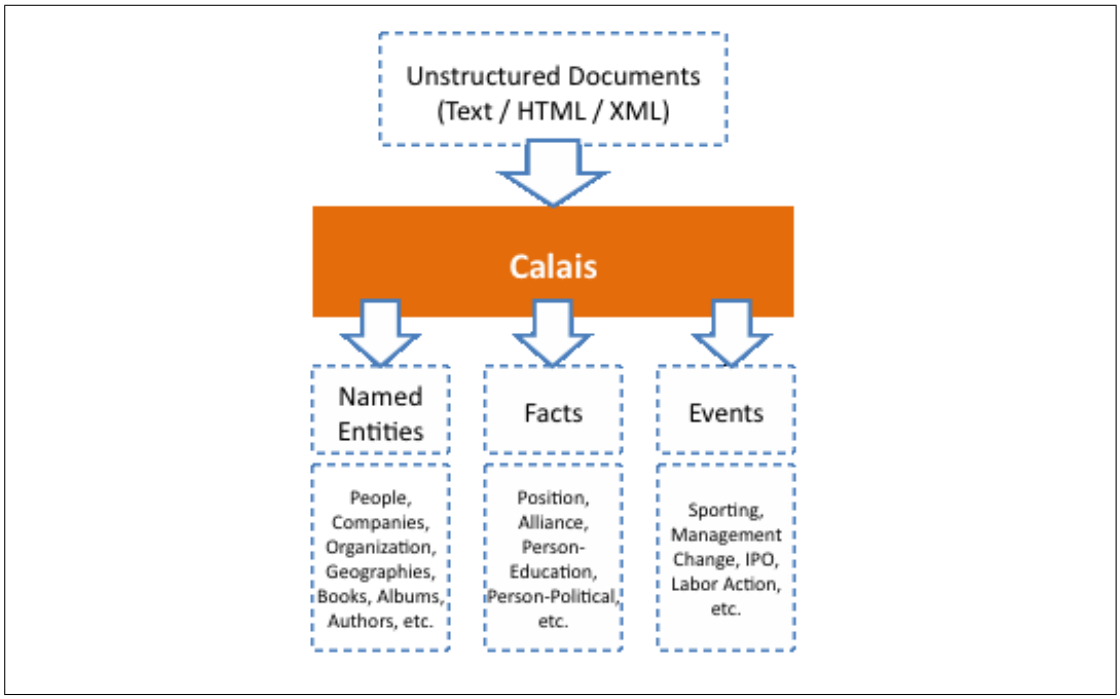

FIG. 4.1. OpenCalais Web Service.

4. Each piece of content and each entity or event in that content is assigned a unique identifier (a document ID and many URIs) that ties back to the Linked Data Cloud.

5. The metadata returned by Calais (tags, document IDs and URIs) can be used to build a local RDF store for querying and display.

We invoke the OpenCalais API via a SOAP call. The OpenCalais Web Service currently provides one web method - Enlighten. Figure 4.2 shows the signature of the 'enlighten' web method for the OpenCalais web service.

4.2 Sensor Log Collection Module

The logs received from the monitoring tools like Snort and Wireshark are in 'pcap' (packet capture) format. We used jNetPcap libarary, a java wrapper for pcap file parsing.

String Enlighten(String licenseID, String content, String paramsXML)			
Field Name	Type	Definition	Notes
licenseID	String	API access key	Obtained through registration
content	String	Content to be annotated	Max input length is 100,000 characters
paramsXML	String	Processing and user directives and external metadata	Max parameters length is 16,000 characters

FIG. 4.2. OpenCalais Web Service SOAP Call.

jNetPcap is an open-source java library. It contains:

1. A Java wrapper for most of the libpcap library native calls.
2. Can decode captured packets in real-time.
3. Provides a large library of network protocols (core protocols).
4. Can be customised to add new protocol definitions using java SDK.
5. jNetPcap uses a mixture of native and java implementation for optimum packet decoding performance.

The network packet contents like source-destination IP addresses, ports, protocols, processes etc. were extracted from the packet captures.

4.3 Reasoning Logic

CWM (pronounced coom) is a general-purpose data processor for the semantic web, somewhat like sed, awk, etc. for text files or XSLT for XML. It is a forward chaining reasoner which can be used for querying, checking, transforming and filtering information. Its core language is RDF, extended to include rules, and it uses RDF/XML or RDF/N3 serializations as required.

Closed World Machine (CWM) is a query machine for Notation3 that recognizes a few logical primitives. CWM is a popular Semantic Web program that can do the following tasks:-

- Parse and pretty-print the following RDF formats: XML RDF, Notation3, and NTriples.
- Store triples in a query-able triples database.
- Perform inferences as a forward chaining FOPL inference engine.
- Perform builtin functions such as comparing strings, retrieving resources, all using an extensible builtins suite

The data received from text descriptions and the network scanners was parsed into RDF format serialized into N3 triples using simple Java parsing modules. We used jython, a wrapper for python in java for reasoning using CWM, as CWM data processor is in pyton. These triples were then processed by CWM for inferencing the possibility of the attack.

Chapter 5

SYSTEM EVALUATION

The system evaluation is broadly divided into two parts:

1. Text extraction evaluation.

2. Reasoning logic evaluation.

The following sections elaborate each of the goals and the evaluation results.

5.1 Text Extraction Evaluation

The primary task of this module was to extract text-entities of importance from cyber-security point of view, and if possible help in determining if the text inherently talks about some cyber threat/vulnerability. We tested our two modules namely 'Named Entity Extractor' and 'Vulnerability Extractor' by passing the vulnerability descriptions in the Common Vulnerabilities and Exposures dataset for testing the effectiveness of the named entity extraction. The dataset comprised of 48933 vulnerability descriptions [36]. The named entities were classified and tagged into various groups. Table 5.1 shows the primes entities-of-interest extracted from the vulnerability texts.

The 'Vulnerability Extractor/Detector' module was also supplied with the Common Vulnerabilities and Exposures dataset, and it was tested whether or not the vulnerability

Total CVE descriptions	48933	
Technology Terms	31844	Eg: MIME, DNS, IRIX
Operating System Terms	9656	Eg: Javascript, ActiveX control
Company Names	11840	Eg: AOL, Ethereal
Industry Terms	15113	Eg: SQL injection, CRLF injection
Programming Language Terms	22653	Eg: Microsoft, Qualcomm
Vulnerability Terms	11560	Eg: buffer overflow, malformed DHCP packets
Programming Bugs Terms	5835	Eg: iOS, AIX
Arbitrary Code Execution Terms	3558	Eg: rpc.statd bug
< Code > Injection Terms	13135	Eg: delete_bug.php

Table 5.1. Named Entity Extraction Results.

extractor module flagged the presence of vulnerability in the text.

5.2 Reasoning Logic Evaluation

We tested the reasoning logic on multiple vulnerabilities that roughly fell in a similar category. 8090 CVE 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 5.4, 5.5, 5.6, and 5.7 were used to infer possibility of an attack. The network scanner logs were simulated i.e. the logs were built up so as to reflect the data mentioned in the text to be true. For example: For the text in figure 5.8, the scanner log was assumed to have information regarding the 'Horde' software, the corresponding version numbers, and the malicious file name (../Image/Image.php). It was observed that the vulnerability detection module on analysis of the text could detect 2610 text descriptions as computer attacks, while the reasoning framework which used conjunction of the text, network monitor logs, and the reasoning rules mentioned above was successful in inferring 7120 attacks. Since the IBM ES750 Intrusion Detection System was assumed not to have any signature of these attacks, it could detect none of these attacks.

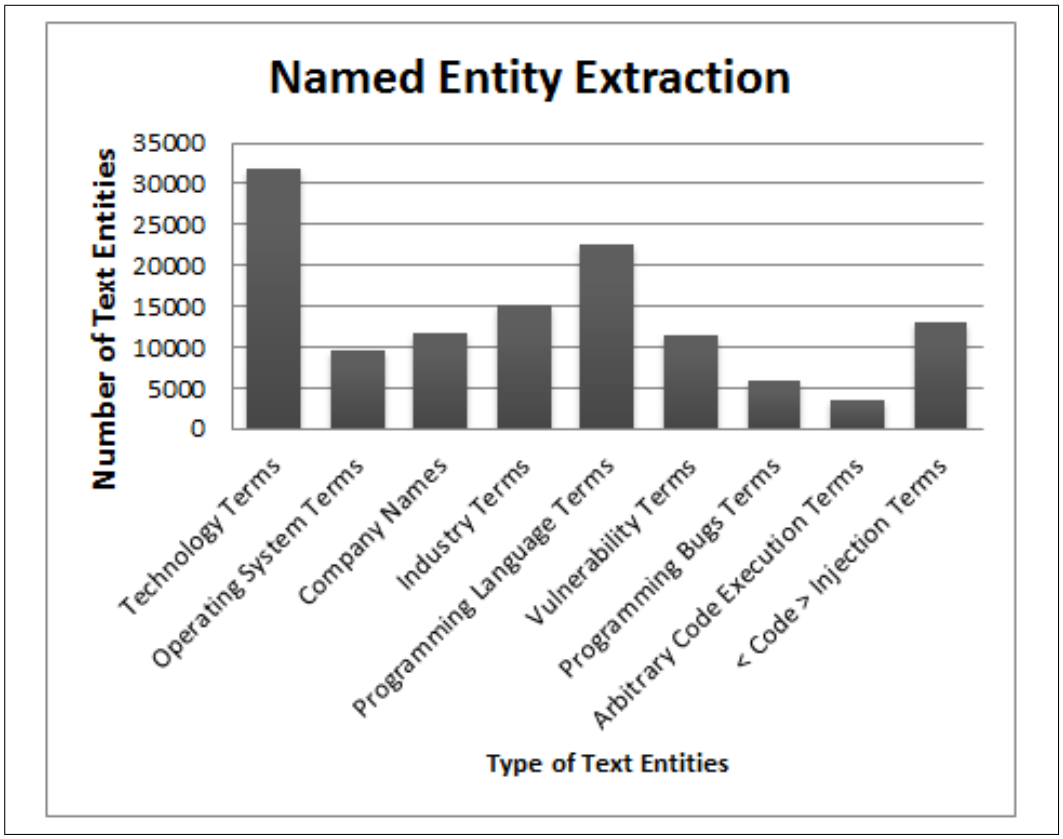

FIG. 5.1. Named Entity Extractor Statistics

Figures 5.8, 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16 and 5.17 are few of the vulnerability examples that got detected using the reasoning rules defined in 5.4, 5.5, 5.6, and 5.7.

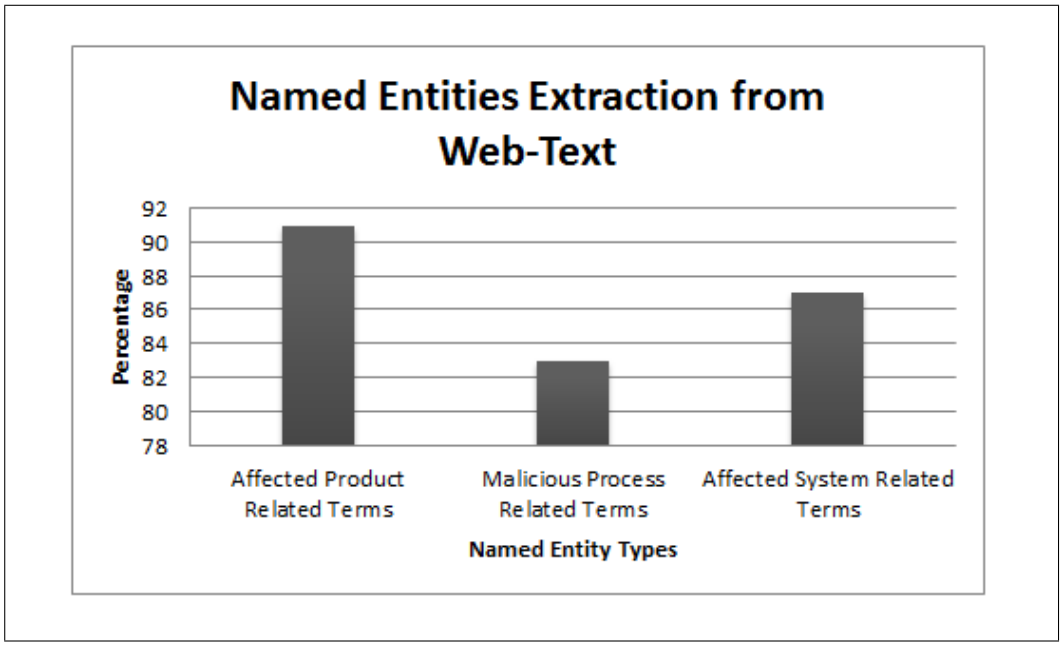

FIG. 5.2. Percentage-wise breakup of Named Entities Extracted

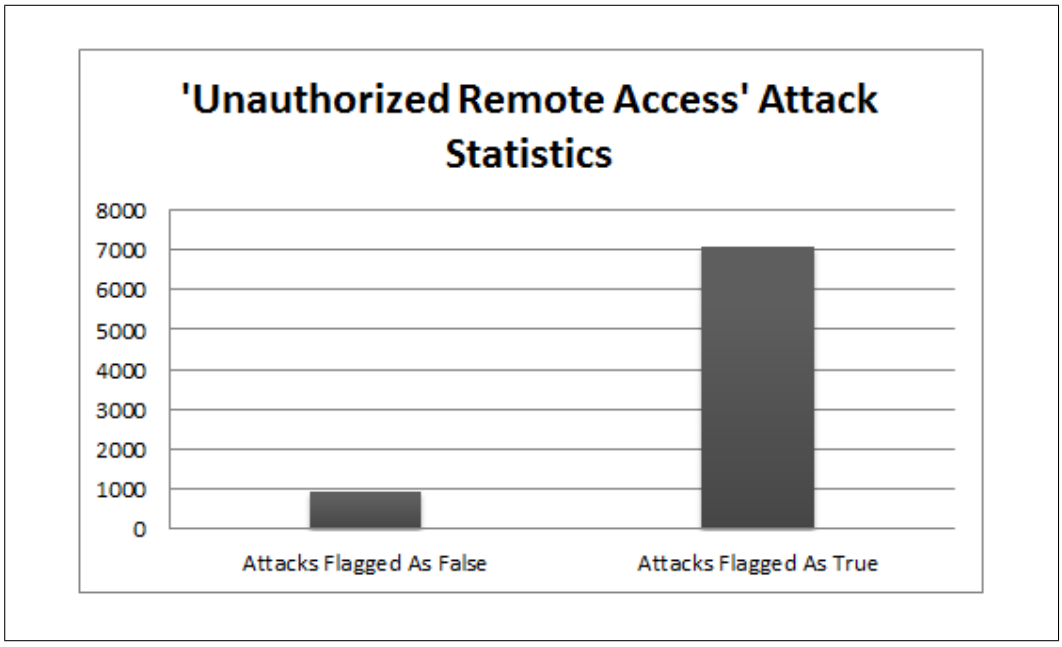

FIG. 5.3. Results for 'Unauthorized Remote Access' detection

We performed an actual attack on a host in a controlled environment and tested the effectiveness of the system. Figure 5.18 outlines the experiment performed. Following were the steps involved in the experiment:

1. Initial System Configuration:
    - Healthy host system is connected to the network.
    - Host system has the corrupted pdf file embedded with the malicious code.
    - Remote attacker is also connected to the same network.
2. Vulnerability Exposure:

```
{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”}
```

FIG. 5.4. Reasoning Logic Rule 1: Outbound Access (Unauthorized Remote Access) via Malicious Process Execution

- The user at host system opens the corrupt pdf.
- The malicious code in annots.api of Acrobat results opening up of outbound port.
- Remote attacker can now directly connect to the compromised host system via the port opened up at the host system.

3. Data Extraction and Collection:

- The host system is connected to a network scanner which collects network activity logs.
- This log information is converted into a format understandable to the reasoner.
- The reasoner uses the information in the knowledge-base and applies the rules to detect and determine the occurrence of an attack.

```
{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”}
```

FIG. 5.5. Reasoning Logic Rule 2: Unauthorized Remote Access/Monitoring via Malicious Command Execution

4. Reasoning and Alert Generation:

    - N3 serialized data, knowledge-base rules that give implications for ‘Unauthorized Remote Access’ to the host system on execution of malicious program/command, and ontology are used to infer facts and generate an alert if it is asserted.

```
{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”}
```

FIG. 5.6. 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”}
```

FIG. 5.7. Reasoning Logic Rule 4: Unauthorized Remote Access/Monitoring via Malicious Object

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

FIG. 5.8. Sample 1 CVE-2009-0932: Description, extracted entities, and result

CVE-2009-0953 text description: Heap-based buffer overflow in Apple QuickTime before 7.6.2 allows remote attackers to execute arbitrary code or cause a denial of service (application crash) via a crafted PICT image.

Entities extracted: Technology Internet, Disaster Accident, QuickTime, Buffer overflow, Programming bugs, Heap, PICT, Denial-of-service attack.
Vulnerability Terms: Computer_security_exploits, Programming_bugs, Computer_errors, Software_anomalies, WikiProject_Computer_Security_articles, Internet_Relay_Chat, Denial-of-service_attacks.
Scanner Logs Terms: Apple QuickTime, sample.pict
Inference Rule Applied: Rule4
Result: Attack Flagged

FIG. 5.9. Sample 2 CVE-2009-0953: Description, extracted entities, and result

CVE-2009-0965 text description: SQL injection vulnerability in functions/browse.php in Ganesha Digital Library (GDL) 4.0 and 4.2 allows remote attackers to execute arbitrary SQL commands via the node parameter in a browse action to gdl.php.

Entities extracted: SQL, Data management, Computing, Vulnerability, SQL injection, Databases, Cross-platform software, Computer science.
Vulnerability Terms: Articles_with_example_SQL_code, SQL, Injection_exploits, Computer_security_exploits, Data_management, Machine_code, Injection_exploits, Malware, Computer_security, Software_testing
Scanner Logs Terms: functions/browse.php, gdl.php, GDL
Inference Rule Applied: Rule1
Result: Attack Flagged

FIG. 5.10. Sample 3 CVE-2009-0965: Description, extracted entities, and result

CVE-2009-0968 text description: SQL injection vulnerability in fmoblog.php in the fMoblog plugin 2.1 for WordPress allows remote attackers to execute arbitrary SQL commands via the id parameter to index.php. NOTE: some of these details are obtained from third party information.

Entities extracted: Law Crime, Software testing, Databases, SQL, Code injection, Cross-site scripting, Vulnerability, SQL injection, WordPress, Blog software.
Vulnerability Terms: Articles_with_example_SQL_code, SQL, Injection_exploits, Computer_security_exploits, Data_management, Computer_security, Software_testing
Scanner Logs Terms: fmoblog.php, SQL
Inference Rule Applied: Rule1
Result: Attack Flagged

FIG. 5.11. Sample 4 CVE-2009-0968: Description, extracted entities, and result

CVE-2009-1022 text description: Heap-based buffer overflow in the Preview/ Set Segment function in Gretech GOMlab GOM Encoder 1.0.0.11 and earlier allows user-assisted remote attackers to cause a denial of service (memory corruption and application crash) or execute arbitrary code via a long text field in a subtitle (.srt) file.

Entities extracted: Disaster Accident, Technology Internet, Buffer overflow, Memory corruption, Programming bugs, PaX, Stack buffer overflow, C, Crash, Denial-of-service attack, GOM Player. Vulnerability Terms: Computer_security_exploits, Programming_bugs, C_standard_library, Computer_errors, Software_anomalies, Programming_bugs, WikiProject_Computer_Security_articles, Internet_Relay_Chat, Denial-of-service_attacks
Scanner Logs Terms: GOM, sample.srt
Inference Rule Applied: Rule4
Result: Attack Flagged

FIG. 5.12. Sample 5 CVE-2009-1022: Description, extracted entities, and result

CVE-2009-1028 text description: Stack-based buffer overflow in ediSys eZip Wizard 3.0 allows remote attackers to execute arbitrary code via a crafted .zip file.

Entities extracted: Law Crime, Environment, Stack, Buffer overflow, Programming bugs, Technology Internet, Stack buffer overflow, Return-to-libc attack, Arbitrary code execution.
Vulnerability Terms: Computer_security_exploits, Programming_bugs, C_standard_library, Computer_errors, Software_anomalies, Programming_bugs, Web_security_exploits, Injection_exploits, Articles_lacking_sources_(Erik9bot)
Scanner Logs Terms: ediSys, sampleEzip.zip
Inference Rule Applied: Rule4
Result: Attack Flagged

FIG. 5.13. Sample 6 CVE-2009-1028: Description, extracted entities, and result

CVE-2009-1029 text description: Stack-based buffer overflow in POP Peeper 3.4.0.0 and earlier allows remote POP3 servers to execute arbitrary code via a long Date header, related to Imap.dll.

Entities extracted: Post Office Protocol, Buffer overflow, Programming bugs, Technology Internet, Stack buffer overflow, Stack. Vulnerability Terms: Computer_security_exploits, Programming_bugs, Computer_errors, Software_anomalies, Exploit-based_worms, C_standard_library
Scanner Logs Terms: Imap.dll, POP Peeper
Inference Rule Applied: Rule1
Result: Attack Flagged

FIG. 5.14. Sample 7 CVE-2009-1029: Description, extracted entities, and result

CVE-2001-0260 text description: Buffer overflow in Lotus Domino Mail Server 5.0.5 and earlier allows a remote attacker to crash the server or execute arbitrary code via a long "RCPT TO" command.

Entities extracted: Disaster Accident, Mail transfer agents, Computing, Programming bugs, Technology Internet, Stack buffer overflow, Arbitrary code execution, IBM Lotus Domino, Buffer overflow, Groupware. Vulnerability Terms: Computer_security_exploits, Programming_bugs, Computer_errors, Software_anomalies, Exploit-based_worms
Scanner Logs Terms: Lotus Domino 5.0.5
Inference Rule Applied: Rule2
Result: Attack Flagged

FIG. 5.15. Sample 8 CVE-2001-0260: Description, extracted entities, and result

CVE-2001-0678 text description: A buffer overflow in reggo.dll file used by Trend Micro InterScan VirusWall prior to 3.51 build 1349 for Windows NT 3.5 and InterScan WebManager 1.2 allows a local attacker to execute arbitrary code.

Entities extracted: Programming bugs, Microsoft Windows, Windows NT, Technology Internet, Arbitrary code execution, Buffer overflow, Trend Micro. Vulnerability Terms: Privilege_escalation_exploits, Operating_system_security, Computer_security_exploits, Programming_bugs, C_standard_library
Scanner Logs Terms: InterScan WebManager, reggo.dll
Inference Rule Applied: Rule1
Result: Attack Flagged

FIG. 5.16. Sample 9 CVE-2001-0678: Description, extracted entities, and result

CVE-2009-1029 text description: CVE-2002-2061 text description: Heap-based buffer overflow in Netscape 6.2.3 and Mozilla 1.0 and earlier allows remote attackers to crash client browsers and execute arbitrary code via a PNG image with large width and height values and an 8-bit or 16-bit alpha channel.

Entities extracted: Buffer overflow, Computing, Netscape, Technology Internet, Mozilla, Portable Network Graphics
Vulnerability Terms: Computer_security_exploits, Programming_bugs
Scanner Logs Terms: netscape, mozilla, .png
Inference Rule Applied: Rule3
Result: Attack Flagged

FIG. 5.17. Sample 10 CVE-2002-2061: Description, extracted entities, and result

Experiment Exploit: Adobe vulnerability CVE-2009-0927
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: Rule1
Result: Threat alert flagged

FIG. 5.18. Adobe Vulnerability CVE-2009-0927: Description, extracted entities, and result

Chapter 6

CONCLUSION AND FUTURE 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 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.

It was further observed that the framework can be improved on following fronts:

- Better expert knowledge: The reasoning logic primarily comprises of N3 rules defined based on the knowledge of the attack behavior. The effectiveness of the framework will significantly increase with more expert knowledge being reflected into the rules. The generalisation of rules as much as possible will also improve the system and its flexibility. These rules can be formulated by the domain experts who understand the intrusion behavior.

- Better and uptodate data-sources: Integration of more data-streams can lead to a rich pool of entities, which can be used to assert facts into the knowledge-base, which can be mapped into the ontology, for better inferencing of the threats/attacks. There is huge potential for making the web-text analyser stronger by enabling real-time

crawling of the web to harvest latest information regarding potential attacks, which in turn can help detect and prevent zero-day attacks. False negatives and false positives can occur due to data sources like blogs and forums which cannot be trusted. Setting up of trust policies can help in dealing with this issue.

- Dealing with possible false negatives and false positives: Trusted data-sources and generalization of reasoning logic rules which capture the intrusion behavior in a more accurate sense can help reduce the false positives and false negatives generated by the system.

We continue to experiment with integration of newer data sources in the framework, and observe the effectiveness of the additions. Modelling the spatial and temporal aspects of the cyber-attacks into the framework is one of the ambitious future work. Linking spatially and temporally separated host and network activities, finding relationship among them, and inferring possibility of an attack could lead to a very effective situation-aware intrusion detection system.

Appendix A

APPENDIX

A.1 OpenCalais Web Service

Sample OpenCalais web service invocation: **Enlighten Method**

String licenseID = "3rrkn34pgpmjr72qqrxw2nzp";

String content = "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.";

String paramsXML = "< c:params xmlns:c = http://s.opencalais.com/1/pred/ xmlns:rdf = http://www.w3.org/1999/02/22-rdf-syntax-ns >< c:processingDirectives c:contentType = TEXT/RAW c:enableMetadataType = GenericRelations, SocialTags c:outputFormat = Text/Simple ></ c:processingDirectives >< c:userDirectives c:allowDistribution=true c:allowSearch=true c:externalID = 17cabs901 c:submitter=ebiquitySumit >< /c:userDirectives >< c:externalMetadata >< / c:externalMetadata></ c:params>";

result = new CalaisLocator().getcalaisSoap().enlighten(licenseID,content,paramsXML);

OpenCalais web service json output snippet:

{
"doc":
{
"info":
{
"allowDistribution":"true",
"allowSearch":"true",
"calaisRequestID" : "bf4cd0ed-73de-9f42-1366-093efbf775de",
"externalID" : "17cabs901",
*"id":"http://id.opencalais.com/SyzTy6*BmD2TYE0QcBDCIA",*
"docId":"http://d.opencalais.com/dochash-1/64089ea2-e8e4-3030-ade5-83b9e79b2741",
"document":"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.",
"docTitle":"",
"docDate":"2012-03-29 17:29:29.850",
"externalMetadata":"",
"submitter":"ebiquitySumit"
},
"meta":
{
"contentType":"TEXT/RAW",
"emVer":"7.1.1103.5",
"langIdVer":"DefaultLangId",
"processingVer":"CalaisJob01",

”submitionDate”:”2012-03-29 17:29:29.679”,

”submitterCode”:”2d60c278-82ef-fdb5-a9de-5c9f55b6f657”,

”signature”:”digestalg-1—ny3OfJcmKqGcCpNzDRSYj3ILcFU=— Cr/jRR4w1pK6RKpXoMWZafYUps

”language”:”English”,

”messages”:[]

}

},

”http://d.opencalais.com/dochash-1/64089ea2-e8e4-3030-ade5-83b9e79b2741/cat/1”:

{

”typeGroup”:”topics”,

”category”:”http://d.opencalais.com/cat/Calais/TechnologyInternet”,

”classifierName”:”Calais”,

”categoryName”:”Technology Internet”,

”score”:0.592

},

”http://d.opencalais.com/dochash-1/64089ea2-e8e4-3030-ade5-83b9e79b2741/SocialTag/2”:

{

”typeGroup”:”socialTag”,

”id”:”http://d.opencalais.com/dochash-1/64089ea2-e8e4-3030-ade5-83b9e79b2741/ SocialTag/2”,

”socialTag”:”http://d.opencalais.com/genericHasher-1/5805427c-8b1f-3142-9877- 2d2422cfb989”,

”name”:”Programming bugs”,

”importance”:”1”,

”originalValue”:”Programming bugs”

},

”http://d.opencalais.com/dochash-1/64089ea2-e8e4-3030-ade5-83b9e79b2741/SocialTag/3”:

{
”typeGroup”:”socialTag”,
”id”:”http://d.opencalais.com/dochash-1/64089ea2-e8e4-3030-ade5-83b9e79b2741/ SocialTag/3”,
”socialTag”:”http://d.opencalais.com/genericHasher-1/09e9e48c-83f0-32e5-aa34- a89a592a5698”,
”name”:”Adobe software”,
”importance”:”1”,
”originalValue”:”Adobe software”
},
}

A.2 Network Activity Packet Capture Reading

Pcap pcap = Pcap.openOffline(file, errbuf); PcapPacketHandler$< String >$ jpacketHandler = new PcapPacketHandler$< String >$()

REFERENCES

[1] "Tracking ghostnet: Investigating a cyber espionage network." http://www.infowar-monitor.net/2009/09/tracking-ghostnet-investigating-a-cyber-espionage-network.

[2] "Incidents of ddos attacks: South korea, wordpress and boxun." http://www.infowar-monitor.net/2011/03/incidents-of-ddos-attacks-south-korea-wordpress-and-boxun/.

[3] "Hackers steal ssl certificates for cia, mi6, mossad." http://anteyekon4myst.visibli.com/share/d9Qe5X.

[4] "Kernel.org linux repository rooted in hack attack." http://www.theregister.co.uk/2011/08/31/linux_kernel_security_breach/.

[5] "Critical vulnerability in flash and acrobat." http://venturebeat.com/2011/03/16/critical-vulnerability-in-flash-and-acrobat/.

[6] "Targeted emails exploit new acrobat reader vulnerability." http://nakedsecurity.sophos.com/2011/12/10/targeted-emails-exploit-new-acrobat-reader-vulnerability/.

[7] "Turkish government site hacked amid spat with china." http://www.pcworld.com/businesscenter/article/168359/turkish_government_site_hacked_amid_spat_with_china.html.

[8] "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.

[9] "Zero-day exploit definition." http://searchsecurity.techtarget.com/definition/zero-day-exploit.

[10] M. E. Whitman and H. J. Mattord, *Principles of Information Security*. Boston, MA, United States: Course Technology Press, 3rd ed., 2007.

[11] "Snort." http://www.snort.org/.

[12] "Tripwire." http://www.tripwire.com/.

[13] "Ossec." http://www.ossec.net/.

[14] "Internet security systems x-force security threats." http://xforce.iss.net.

[15] "Wireshark." http://www.wireshark.org/.

[16] "Nagios." http://www.nagios.org/.

[17] "Cacti." http://www.cacti.net/.

[18] V. Raskin, C. F. Hempelmann, K. E. Triezenberg, and S. Nirenburg, "Ontology in information security: a useful theoretical foundation and methodological tool," in *Proceedings of the 2001 workshop on New security paradigms*, NSPW '01, (New York, NY, USA), pp. 53–59, ACM, 2001.

[19] M. Wei, G. Xu, X. Chen, and C. Xu, "Study on ontology-based intrusion detection," in *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, vol. 10, pp. V10–357 –V10–359, oct. 2010.

[20] 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.

[21] J. Undercoffer, T. Finin, A. Joshi, and J. Pinkston, "A target-centric ontology for intrusion detection," in *Proc. 18th Int. Joint Conf. on Artificial Intelligence*, 2004.

[22] 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.

[23] "National vulnerability database." `http://nvd.nist.gov`.

[24] 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.

[25] V. Khadilkar, J. Rachapalli, and B. Thuraisingham, "Semantic web implementation scheme for national vulnerability database," Tech. Rep. UTDCS-01-10, Univ. of Texas at Dallas, 2010.

[26] C. Y. C. F. Haibo Hu, Dan Yang and R. Li, "Detecting interactions between behavioral requirements with owl and swrl," *WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY*, vol. 48, pp. 330–336, Dec. 2010.

[27] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, vol. 284, pp. 34–43, May 2001.

[28] O. Lassila, R. R. Swick, W. Wide, and W. Consortium, "Resource description framework (rdf) model and syntax specification," 1998.

[29] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowl. Acquis.*, vol. 5, pp. 199–220, June 1993.

[30] G. Antoniou and F. van Harmelen, *A Semantic Web Primer*. Cambridge, MA: MIT Press, 2. ed., 2008.

[31] J. P. Anderson, "Computer security threat monitoring and surveillance," tech. rep., Fort Washington, Pennsylvania, 1980.

[32] D. E. Denning, "An intrusion-detection model," in *IEEE Symposium on Security and Privacy*, pp. 118–133, 1986.

[33] "Top command (linux)." `http://linux.die.net/man/1/top`.

[34] "Monit." `http://mmonit.com/monit/`.

[35] "Cisco hardware sensor." `http://www.cisco.com/en/US/products/hw/vpndevc/ps4077/index.html`.

[36] "Common vulnerabilities and exposures." `http://cve.mitre.org/`.

[37] "Ibm proventia network enterprise scanner 750." `http://www.proventiaworks.com/ES750.asp`.

[38] "Adobe acrobat vulnerability cve-2009-0927." `http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-0927`.

[39] J. Undercofer, *Intrusion Detection: Modeling System State to Detect and Classify Aberrant Behavior*. PhD thesis, University of Maryland, Baltimore County, February 2004.

[40] "N3 definition." `http://www.w3.org/TeamSubmission/n3/`.

[41] "Cuppens." `http://www.lsv.ens-cachan.fr/~goubault/SECI-02/Final/Cuppens/cuppens.pdf`.

[42] S. Smith, "How to write a dissertation," *Fake Journal*, vol. 1, pp. 1–5, 2006.

[43] J. Jones, "Creating beautiful graphs," in *Proceedings of the Conference on Drawing Graphs*, (Some Location, Some Country), pp. 4–6, Some Publisher, 1998.

[44] "Netflow." http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html.

[45] "Aurora." http://www.zurich.ibm.com/aurora/.

[46] "Juniper website text description of cve-2009-0927." http://www.juniper.net/security/auto/vulnerabilities/vuln34169.html.

[47] US-CERT, "Vulnerability notes database." http://www.kb.cert.org/vuls/.

[48] T. Finin and Z. Syed, "Creating and Exploiting a Web of Semantic Data," in *Proc. 2nd Int. Conf. on Agents and Artificial Intelligence*, Springer, January 2010.

[49] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 1 ed., July 2008.

[50] "Ebiquity ids ontology." http://ebiquity.umbc.edu/ontologies/cybersecurity/ids/.

[51] S. Huang, H. Tang, M. Zhang, and J. Tian, "Text clustering on national vulnerability database," in *Proc. 2nd Int. Conf. on Computer Engineering and Applications - Volume 02*, ICCEA '10, (Washington, DC), pp. 295–299, IEEE Computer Society, 2010.

[52] "Opencalais." http://opencalais.com/.