Pattern Analysis and Signature Extraction for Intrusion Attacks on Web Services

Urjita Thakar[1], Nirmal Dagdee[2], Sudarshan Varma[3]

[1] Reader, Computer Engineering Department,
Shri G.S. Institute of Technology and Science,
23, Visweswaraiya Road, Indore,
MP, 452 003 India
urjita@rediffmail.com,uthakar@sgsits.ac.in

[2]Director and Professor, Computer Science and Information Technology,
S.D. Bansal College of Technology,
A.B. Road, Umaria,
Indore, MP, India
nirmal_dagdee@rediffmail.com

[3] Project Manager, Ideavate Solutions,
2101 Highway 516, Suite F Old Bridge, NJ 08857, USA
sudarshan.varma@ideavate.com

ABSTRACT

The increasing popularity of web service technology is attracting hackers and attackers to hack the web services and the servers on which they run. Organizations are therefore facing the challenge of implementing adequate security for Web Services. A major threat is that of intruders which may maliciously try to access the data or services. The automated methods of signature extraction extract the binary pattern blindly resulting in more false positives. In this paper a semi automated approach is proposed to analyze the attacks and generate signatures for web services. For data collection, apart from the conventional SOAP data loggers, honeypots are also used that collect small data which is of high value. To filter out the most suspicious part of the data, SVM based classifier is employed to aid the system administrator. By applying an attack signature algorithm on the filtered data, a more balanced attack signature is extracted that results in fewer false positives and negatives. It helps the Security Administrator to identify the web services that are vulnerable or are attacked more frequently.

KEYWORDS

Web services, Intruders, Signatures, Honeypots, Classifier

1. INTRODUCTION

With the increasing popularity and growth of the Internet, more and more web applications and web services are being deployed. Web services are software components that are meant to be used by other users over the Internet. They are widely used by businesses for their business transactions. Web services are fundamentally based on Web Services Description Language (WSDL), Universal Directory and Description Interface (UDDI) and Simple Object Access Protocol (SOAP) technologies. WSDL documents are used to publish the service descriptions. UDDI directories can be used by service requestors to find the available services. SOAP is a messaging protocol, which is used for communicating messages between two parties. SOAP messages are transported using protocols like HTTP, SMTP etc. The SOAP server usually runs on a web server, therefore the threats existing for a web server also exist for a SOAP server [1]. An attacker can send a specially-formulated SOAP request to cause a denial of service condition on a SOAP server.

Even though the SOAP message body can be encrypted, network eavesdropping may lead to disclosure of confidential information. Some existing standards and specifications such as WS-Security [2], WS-SecureConversation [3], WS-SecurityPolicy [4] etc. are meant for applying security during communication of the SOAP messages and usage of web services.

As web services are being popularly used by businesses, attackers are getting attracted to hack the web services and the servers on which they run. A major threat is that of intruders which may maliciously try to access the data or services. Thus, the servers on which web services run need to be protected from intruders. Therefore, Intrusion Detection systems need to be employed. Signature based IDS are popularly being used. Honeypots are a highly flexible security tool with a variety of applications for security [5]. They are a security resource that does not have any production or authorized activity but have an important use in intrusion prevention, detection and information gathering. Honeypot collects very little data and that is normally of high value. This information can be used in extraction of intrusion detection signature.

Many common intrusion detection systems often work as misuse detectors, where the packets in the monitored network are compared against a repository of signatures that define characteristics of an intrusion. Successful matching causes alerts to be fired. The signature often consists of one or more specific binary patterns found in a given network packet. The signature can be described as a boolean relation called rule [6]. Continuous updation of the signature database is essential since as intrusion detection system is able to recognize an attack only when the signature for it is known. Also, continuous efforts are required to detect new attacks and determine appropriate signatures from them. Moreover, a slight change in the attack scenario may be enough to alter the attack signature and thus fool a signature filter. They are consequently vulnerable to polymorphic attacks and other evasion techniques which are expected to grow in the near future. The creation of these signatures is a tedious process that requires detailed knowledge of each software exploit that is to be captured and a large pool of ASCII-log data to analyze. The automated extraction of the signatures like longest common substring (LCS) algorithm to the database of attack log data as presented in [7] extracts the binary pattern blindly, resulting in more false positives. Signature extraction discussed in [8] does not consider attacks at application layer and needs to be improved for extraction of signatures for attacks on web services. Also, better data analysis methods need to be employed on the traffic log data.

In this paper, we discuss various common attacks on web services and present a method that can be used in analysis of the intrusion patterns and signature extraction for them. These extracted signatures can be used in signature based IDS. We present a system to facilitate extraction of balanced attack signatures to avoid too many false positives or negatives. The presented system architecture is generalized and allows the Security Administrator to determine which web services are being attacked more frequently and need more protection. A prototype of the proposed system is implemented using an SVM based semi-supervised classifier, honeypot framework and SOAP traffic loggers.

Rest of the paper is organized as follows – Related work has been discussed in section 2. In section 3 common attacks on web services are discussed. In Section 4 a brief overview of Intrusion detection systems and Honeypots is presented. Overview of the classification technique is discussed in section 5. The proposed system is presented in Section 6. The prototype development details are given in section 7. Features of the proposed system are discussed in section 8 and conclusions are drawn in section 9. The future work is also presented in this section.

2. RELATED WORK

In the literature, two basic techniques used to detect intruders have been discussed; namely, anomaly detection and misuse detection (signature detection). Anomaly Detection is designed to uncover abnormal patterns of behavior, the IDS establish a baseline of normal usage patterns, and anything that widely deviates from it is flagged as a possible intrusion. Misuse Detection, commonly called signature detection, uses specifically known patterns of unauthorized behavior to predict and detect subsequent similar attempts. These specific patterns are called signatures [7]. A tool discussed for signature extraction as discussed in [8] does not consider attacks at application layer. The data analysis method presented in [9] needs to be improved to filter log data to obtain higher concentration of attacks for precise signature extraction.

In [10], Honeypots have been discussed as a means of simple and cost-effective intrusion detection. These have been successfully applied for network intrusion detection widely. However, these have not been used for protection at application layer so far. An application layer intrusion detection method for SQL Injection attack technique has been discussed in [11].

Some researchers have proposed intrusion detection systems for distributed environments. A. Abraham et al. have proposed a distributed intrusion detection system based on fuzzy-rule based classifiers. The intrusion detection and monitoring in this work is performed at network layer [12]. The distributed intrusion detection system is based on network and transport layer data. The system requires the audit data collected from different places to be sent to a central location for analysis [13]. A method that uses Fuzz-Trust model for authentication and Honey tokens for detection of intrusion in web services has been discussed in [14]. It is useful for defense against DoS and DDoS attacks.

Machine learning classifiers use object characteristics to identify the class or group it belongs to. A linear classifier achieves classification based on the value of a linear combination of characteristics that are presented in the form of a vector 'feature vector' to the machine for classification. Several classifiers exist such as Neural network based [17], Genetic Algorithm based [18], Decision tree based [19] etc. These have been used for classification of intrusion attacks by various researchers [15] [16]. The data sets used in these works contain intrusion attacks at network and transport layer.

3. Web Services and Possible Attacks on them

3.1 Web Services

A web service is a software application identified by a Uniform Resource Identifier (URI) whose interfaces and bindings are capable of being defined, described, and discovered as XML artifact [20]. A Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols. Some of the key features of Web services are that they are self-contained, self-describing and modular, they can be published, located, and invoked across the Web, and they are language independent and interoperable. Web services are a relatively new technology that have received wide acceptance as an important implementation of service-oriented architecture. Web services provide a distributed computing approach for integrating extremely heterogeneous applications over the Internet. The Web service specifications are independent of programming language, operating system, and hardware to promote loose coupling between the service consumer and provider. The technology is based upon open technologies such as Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Universal Description Discovery and Integration (UDDI), Web Services Description Language (WSDL).

SOAP is a protocol for exchanging XML-based messages over computer networks, normally using HTTP. SOAP forms the foundation layer of the Web services stack, providing a basic messaging framework. There are several different types of messaging patterns in SOAP, but by far the most common is the Remote Procedure Call (RPC) pattern, in which one network node (the client) sends a request message to another node (the server), and the server immediately sends a response message to the client.

3.2 Attacks against Web Services

Vulnerabilities present in messaging protocols like SOAP and the markup language XML expose Web Services to potential attack by intruders. Some common attacks targeting the Web services [21] [22] are discussed below.

(1) *Denial of Service Attacks*
A direct attack on availability, a DoS attack prevents the service provider from receiving or responding to messages from a requester. Because Web service interfaces are heterogeneous, it takes knowledge about the underlying Web service applications to protect them against DoS attacks. In Denial of Service attack the attacker's goals are to reveal information that he can use for crashing the Web application process. DoS may disable the users' computer or network where the attack can, for example, prevent data exchanging between two sites. Performing DoS the attacker may attack a router, firewall, or proxy server with the goal of making them unusable. The attacker may issue repetitive SOAP/XML messages in an attempt to overload the Web service. The effect of the Denial of Service attack is to prevent the service-providing computer from being able to provide the service.

(2) *Command Injection*
In a command injection, executable logic is inserted in non-executable text strings submitted to a provider/provider Web service. The main types of command injection are SQL injection targeting Web service-enabled database applications, and XML injection targeting Web services. SQL Injection attack occurs when malicious SQL statements are inserted into XML in order to disrupt the back-end system. Trying to force a SOAP endpoint, i.e., server to do something it wasn't meant to do. For example retrieval of data it is not authorized to access, destruction of data through SQL Injection and manipulation of content within a SOAP message. This results in receiving endpoint and consumes excessive resources, i.e., buffer overflow, and crashes or becomes unresponsive.

(3) *Oversized Payloads Sent to XML Parsers*
XML is verbose by design in its markup of existing data and information, so file size must always be considered. A huge payload could be from an attacker again exercising the parser to execute a DoS attack. Parsers based on the document object model, which represent the entire XML document in memory, are especially susceptible to this attack, given their need to model the entire document in memory prior to parsing. Coercive parsing, discussed above, is an example of sending an oversized payload.

(4) *Principal Spoofing*
In this attack, a false message is sent which appears to be from a valid requester. For example, the attacker sends a message that appears as though it is from a valid requester service. When performing IP-spoofing attack an attacker fakes IP address to deceive receiver to believe it is sent from a location that it is not actually from. If attacker gains access to the network with a valid IP address, he/she can modify, reroute, or delete data. The attacker can gain access to sensitive information or take control of the "victim" computer.

(5) *Buffer Overflow Exploits*
Buffer overflow exploits are targeted at Web service components (most often those written in C or C++) that accept data as input and store it in memory (rather than on disk) for later use or manipulation. An overflow of a memory buffer results when the Web service component fails to adequately check the size of the input data to ensure that it is not larger than the memory buffer allocated to receive it, and instead passes the too-large data into the too-small buffer. The result is that the excess data is written into other areas of memory that are not prepared to receive it. Buffer overflows are particularly dangerous when those other areas of memory are allocated to store executable code rather than passive data.

(6) *Schema Poisoning*

XML schemas provide formatting instructions for parsers when interpreting XML documents. Schemas are used for all of the major XML standard grammars coming out of OASIS. Because these schemas describe necessary preprocessing instructions, they are susceptible to poisoning. An attacker may attempt to compromise the schema in its stored location and replace it with a similar but modified one that will either cause valid XML documents to be rejected, or cause invalid or malicious XML documents to be accepted by the application.

(7) *Registry Disclosure Attacks*
Attackers can use mis-configured registries (LDAP, X.500, etc.) to obtain information about the Web service being attacked. In particular, these registries can contain authentication information that an attacker may be able to use. The registries can be compromised or corrupted, which may allow an attacker to gain information about the Web service's host or even gain access to that host.

(8) Dictionary Attack
XML Web service interfaces are heterogeneous in nature with each system having its own authentication system and methods for deterring undesired behavior. Dictionary attacks are common where an attacker may either manually or programmatically attempt common passwords to gain entry into a system or multiple systems. Most password-based authentication algorithms are vulnerable to dictionary attacks.

(9) Format String Attacks
To exploit format string vulnerability, the attacker sends unexpected inputs to the program in the form of strings specifically crafted to cause a privileged program to enable privilege escalation by a normal user. For example, each HTTP command associated with a method that tells the server about the type of action to be performed. Get and Post methods are most commonly used methods. The Get method is designed for getting information and the Post method is designed for posting information. When a URL directory is typed in a browser the Get method is used. The Get method can include some of its own information, which passed as a sequence of characters appended to the request URL in what‘s called a query string. Users can manipulate the query string values, since they are displayed in the browser's URL address field.

4. INTRUSION DETECTION SYSTEMS

Intrusion detection systems, or IDSs, have become an important component in the Security Administrator's toolbox. In a nutshell, intrusion detection systems do exactly as the name suggests: they detect possible intrusions. More specifically, IDS tools aim to detect computer attacks and/or computer misuse, and to alert the proper individuals upon detection. Intrusion detection systems serve three essential security functions: they monitor, detect, and respond to unauthorized activity by organization insiders and outsider intrusion. Intrusion detection systems use policies to define certain events that, if detected will issue an alert in the form of a sound or email. Intrusion detection systems are an integral and necessary element of a complete information security infrastructure performing as "the logical complement to network firewalls". Simply putting, IDS tools allow for complete supervision of networks, regardless of the action being taken, such that information will always exist to determine the nature of the security incident and its source.

4.1. Types of IDS

Intrusion Detection Systems are of following basic types-

(i) Host-based systems were the first type of IDS to be developed and implemented. These systems collect and analyze data that originate on a computer that hosts a service, such as a Web server. Once this data is aggregated for a given computer, it can either be analyzed locally or sent to a separate/central analysis machine.

(ii)Network-based intrusion detection analyzes data packets that travel over the network. These packets are examined and sometimes compared with empirical data to verify their nature: malicious or benign. Because they are responsible for monitoring a network, rather than a single host, Network-based intrusion detection systems (NIDS) tend to be more distributed than host-based IDS.

The two types of intrusion detection systems differ significantly from each other, but complement one another well. In a proper IDS implementation, it would be advantageous to fully integrate the network intrusion detection system, such that it would filter alerts and notifications in an identical manner to the host-based portion of the system, controlled from the same central location. In doing so, this provides a convenient means of managing and reacting to misuse using both types of intrusion detection.

4.2. Techniques

For each of the two types as described above, there are two basic techniques used to detect intruders: anomaly detection and misuse detection (signature detection).

(i) Anomaly Detection is designed to uncover abnormal patterns of behavior. The IDS establish a baseline of normal usage patterns, and anything that widely deviates from it is flagged as a possible intrusion [7].

(ii) Misuse Detection, commonly called signature detection, uses specifically known patterns of unauthorized behavior to predict and detect subsequent similar attempts. These specific patterns are called signatures. Therefore in case of Misuse Detection at the heart of IDS is the attack signature. The signatures can be generated through approaches like Network Grapping / Pattern Matching, Protocol Decode/Analysis, Heuristic and Honeypot.

4.3. Attack Signatures

The purpose of attack signatures is to describe the characteristic elements of attacks. A signature can be a portion of code, a pattern of behavior, a sequence of system calls, etc. There is currently no common standard for defining these signatures. As a consequence, different systems provide signature languages of varying expressiveness. A good signature must be narrow enough to capture precisely the characteristic aspects of exploit it attempts to address; at the same time, it should be flexible enough to capture variations of the attack. Failure in generating good signatures leads to either large amounts of false positives or false negatives.

Content Based Signature Generation [23] is process of extracting the attack signatures based on selection of the most frequently occurring byte sequences across the flows in the suspicious flow pool. To do so various algorithms like LCS are applied to extract the common patterns in it since malicious payload appears with increasing frequency as the malicious activity spreads. A method for generating semantics-aware signatures has been discussed in [24].

4.4. Honeypots

The honeypot has emerged as an effective tool for observing and understanding intruder's toolkits, tactics, and motivations [10]. A honeypot suspects every packet transmitted to/from it, giving it the ability to collect highly concentrated and less noisy datasets for network attack analysis.

Honeypots are decoy computer resources set up for the purpose of monitoring and logging the activities of entities that probe, attack or compromise them [25]. Activities on honeypots can be considered suspicious by definition, as there is no point for benign users to interact with these systems. Honeypots come in many shapes and sizes; examples include dummy items in a database, low-interaction network components like preconfigured traffic sinks, or full-interaction hosts with real operating systems and services [26].

Honeypots are highly useful at detection, addressing many of the problems of traditional detection. Honeypots reduce false positives by capturing small data sets of high value, capture unknown attacks such as new exploits or polymorphic shell-code, and work in encrypted and IPv6 environments [6]. In general, low-interaction honeypots make the best solutions for detection as they are easier to deploy and maintain.

5. SVM BASED SEMI-SUPERVISED CLASSIFIER

Machine Learning Classifiers use object characteristics to identify the class or group it belongs to. A linear classifier achieves this based on the value of a linear combination of characteristics that are presented in the form of a vector 'feature vector' to the machine for classification. Several classifiers exist such as Neural network based, Genetic Algorithm based, Decision tree based, etc. [17][18][19].

In this work, a classifier to classify the semi-supervised data samples using the concept of support vector machine as discussed in [27] has been used. In this approach formulation of spherical decision boundaries and the exploitation of the dynamical system associated with support function is done to obtain the number of clusters. To be able to apply the classifier, data set may be preprocessed.

6. PROPOSED APPROACH

General system architecture as shown in figure 1 is proposed to allow attack pattern analysis and signature extraction.The proposed system consists of following components–

i) Data Logging Component: To log the activities of an attacker, traffic logging is important. This component includes various traffic capturing mechanisms like Hoeypots and other traffic monitoring tools for data collection. This component also uses some tools to intercept the requests for web services simulated on honeypots. Honeypot can be used to create connection logs that report attempted and completed connections for all protocols. To analyze the complete attack scenario, the system needs full payload of the packets entering and leaving the honeypot. This task is performed by a tool for network monitoring. The web service requests are in the form of SOAP messages which use HTTP, SMTP as transport protocols. To filter the SOAP messages from these requests a filter may be used.

ii) Data Analysis Component: This component contains data analysis part of signature extraction mechanism for extracting precise attack signature. The data analysis module analyses the traffic data to select the most suspicious part. This part is realized though a semi automatic method in which the data is classified into benign and malign classes corresponding to various categories of attack on web services. The security manager is then allowed to select the suspicious data based on his experience. For this purpose some support such as graphical interface shall be very useful.

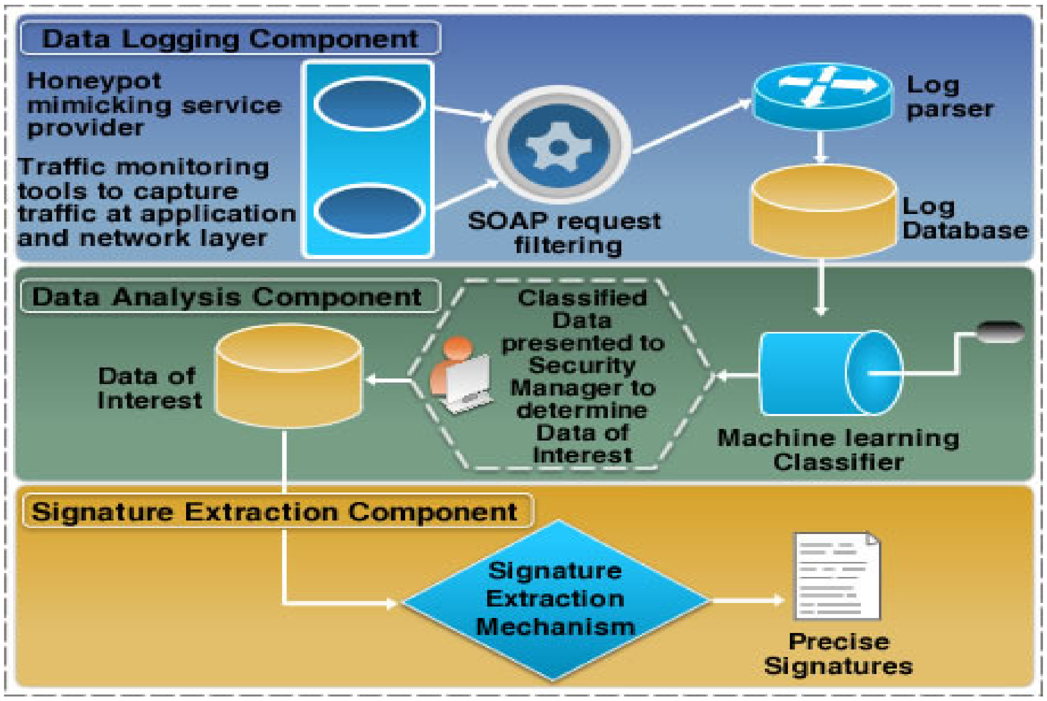
Data Logging Component
Honeypot mimicking service provider
Traffic monitoring tools to capture traffic at application and network layer
SOAP request filtering
Log parser
Log Database
Data Analysis Component
Classified Data presented to Security Manager to determine Data of Interest
Data of Interest
Machine learning Classifier
Signature Extraction Component
Signature Extraction Mechanism
Precise Signatures

Figure 1: Proposed System Architecture

iii) Signature Extraction Component: This component contains signature extraction mechanism for good quality attack signatures. A number of mechanisms can be employed for this. These include artificial intelligence based techniques such as Fuzzy Logic, Neural Network for pattern matching or application of algorithms like LCS.

7. PROTOTYPE DEVELOPMENT

A prototype of the proposed system has been implemented. The honeypots are deployed using the open source Honeyd system [28] [29]. Details of the system are as given in the following subsections.

7.1. Data Logging Component

Data logging is mainly done by using Honeypot log, tcpdump log and the log generated by SOAP traffic filter tools. The Honeyd creates connection logs that report attempted and completed connections for all protocols. To analyze the complete attack scenario, the system needs full payload of the packets entering and leaving the honeypot. This task is performed by Tcpdump which captures every packet's full payload. Tcpdump is a tool for network monitoring and one of the most well known sniffers for Linux. Built with the libpcap (packet capture library) interface, it collects information from packets on the network including those intended for other host machines. The web service requests are in the form of SOAP messages, which mainly use HTTP and SMTP as transport protocols. The HTTP and SMTP requests have been filtered to extract SOAP messages from them. Some tools to log and filter the SOAP traffic such as UtilSnoop [30], SOAPui [31] and Fiddler [32] etc. exist for this purpose. These are used to intercept and monitor the requests coming to the web services simulated on the honeypot. The intercepted requests are logged in a single file for further analysis. Following are few sample examples that show the SOAP requests captured using the data logging component -

(a) Example # 1 Intercepted SOAP Request Showing SQL Injection Attack

```
POST /soap/servlet/rpcrouter HTTP/1.0
Host: localhost:9000
Content-Type: text/xml; charset=utf-8
Content-Length: 389
SOAPAction: ""
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
    <fn:PerformFunction xmlns:fn=" ">
        <fn:uid>8123</fn:uid>
            <fn:password>
                ’or 1=1 or password=‘
            </fn:password>
        </fn:PerformFunction>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

(b) Example # 2 Intercepted SOAP Request Showing DoS Attack

```
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle=
"http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <SOAP-ENV:Body>
  <calculate>
        <principal xsi:type="xsd:double">10000.0</principal>
        <months
        xsi:type="xsd:int">28484c3593275ab5451a8729hbCBDZXJ0aWZpY2F0a&$#~!^&$#~!^&$#
        ~!^W9uIEF1dGhvMDMxODU4MzRaMFwxCzAJBgNVBAYT&$#~!^&$#~!^&$#~!^1cml0
        eSwgSW5jLj&$#~!^&$#~!^ErMCkGA1sdfbdbbgfb##$#$^%43UECxMi&$#~!^&$#~!^1d
        Ghvcml0eTCBmzANBgkq&$#~!^&$#~!^&$#~!^&$#~!^hkiG9w0BsbsZwmdu41QUDaSiC
        &$#~!^&$#~!^nHJ/lj+O7Kw </months>
   <rate xsi:type="xsd:float">0.08</rate>
  </calculate>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Sample of the contents of honeypot log files related to the above examples are shown below:

Honeypot Log

```
 honeyd log started ------
2007-03-08-15:58:19.8679 honeyd log started ------
2007-03-08-16:48:02.8939 tcp(6) S 10.1.20.197 1781 10.1.20.198 220 [Windows XP SP1]
2007-03-08-16:48:07.7747 tcp(6) S 10.1.20.197 1782 10.1.20.198 220 [Windows XP SP1]
2007-03-08-16:48:09.4611 tcp(6) S 10.1.20.197 1783 10.1.20.198 220 [Windows XP SP1]
2007-03-08-16:48:10.4445 tcp(6) S 10.1.20.197 1784 10.1.20.198 220 [Windows XP SP1]
2007-03-08-16:48:43.3277 tcp(6) E 10.1.20.197 1773 10.1.20.198 220: 393 486
2007-03-08-16:49:01.3137 tcp(6) E 10.1.20.197 1780 10.1.20.198 220: 709 710
2007-03-08-16:49:02.9353 tcp(6) E 10.1.20.197 1781
2007-03-08-16:49:07.8303 tcp(6) E 10.1.20.197 1782 10.1.20.198 220: 772 711
2007-04-18-14:11:45.0116 honeyd log stopped ------
```

7.2. Data Analysis Component

This component contains data analysis part of signature extraction mechanism. The data analysis module analyses the traffic data to select the most suspicious part. The filtered SOAP data log is classified into benign and malicious classes corresponding to various categories of attacks on web services.

A SVM based classifier discussed in [33] was used for this purpose. A SOAP attack log dataset was created from the SOAP server log containing requests and responses. In the dataset some samples were labeled by the attack name. The types of attacks considered in this work are Denial of Service attack and SQL injection attack. This dataset includes attributes like client (Service requestor) IP address, client (Service requestor) port number, server (Service provider) IP address, server (Service provider) port address and the message string containing the request/response. Depending on the content pattern observed in the request/response message, the message was classified as normal access or attack to the system. In this study, the data is divided into four different classes. The grouping is discussed as follows –

- If the request/response contains a ‘’ character in the message string then they are grouped to form class1.
- If the request/response contains a ‘%’ character in the message string then they are grouped to form class 2.
- If the message contains meaningless string pattern then they are classified as Denial of Service (DoS) attack and grouped to form class 3.
- If the access is normal to system then they are classified as belonging to class 4.

The number of samples selected for training and testing is shown in the table below:

Table 1: Description of the dataset

Data set	Dimension, Number of classes	Training	Testing
Attack	2,4	150	50

Table 2: Performance in terms of accuracy

Training		Accuracy
Labeled	Unlabeled	After pre-processing
60	90	72%

Using the classified data, the web interface that gives graphical output using which security administrator can easily find out the most attacked port, the IP address making most attacks on various ports on different machines of the organizations network in the form of pie chart as shown in figures 3 and 4 respectively. The figure 5 shows the links that were being most frequently attacked on a particular server. It also shows the number of attempts to access those links. The links correspond to the URIs of the web services implemented on the server. Figure 6 and 7 show ports that have been most active in the last three hours and one month respectively. Such summary can be obtained for any time duration as desired by the Security Administrator.

The data analysis component can be realized by following the steps given below-

i) Configure honeyd to simulate network.
ii) Run Tcpdump for traffic analysis. The intercepted traffic is filtered to determine the protocols and filter the SOAP requests.

iii) Run SOAP requester tools Utilsnoop, SOAPui, Fiddler on different simulated machines to capture the SOAP requests and generate logs for these requests. Combine the logs (i.e. the log of SOAP requests obtained from step (ii) and step (iii) to give a single log of SOAP requests).
iv) Invoke the auto-run shell script that will run in a particular time interval and execute the parser utility that will parse the data from the log files and insert it into the database, as shown in figure 1. The realization of parser utility can be done in any language that has strong string tokenization capability like Java.
v) Different logs for the transport protocols for SOAP are generated that help in determining the popular transport protocol being used for attack.
vi) Preprocess the data so that it can be applied to the classifier.
vii) Classify the data to segregate benign and malign SOAP requests. Classify the malign data into various attack categories.
vi) Login to the web interface to view the attack patterns and analyze the data for extraction of good quality signature.

To enable the Security Administrator to select the suspicious data, the web GUI has the following features:

i) Ability to display packet information from the database.
ii) Ability to display real time network traffic from data stored in database as well as historical traffic statistics.
iii) Ability to display the frequency of attacks on various servers like FTP, SMTP etc.
iv) Ability to display the frequency of attacks on various web services deployed.
v) Display the ports, which were attacked within a certain time range.
vi) A timeline based hit statistics showing number of hits per second Honeypot got in a certain time range.
vii) Show which remote IP-addresses "visited" the Honeypot in a certain time range using a pie chart. Here it is also possible to specify a port number to show activity on a specific port.

In the proposed method, database module is useful due to the reason that it is easier to search for a particular data in the database as it only requires to construct a proper query. The database also facilitates graphical representation of generated data. The graphical interface can be run independent of the Honeypot. Since past events are all recorded in a database, the web GUI can analyze events without having to interfere with normal operations of the Honeypot. Thus, the proposed system allows for better selection of data for extracting the attack signatures as against the existing methods that blindly apply the content-based signature extraction algorithm on entire data captured by the honeypot.

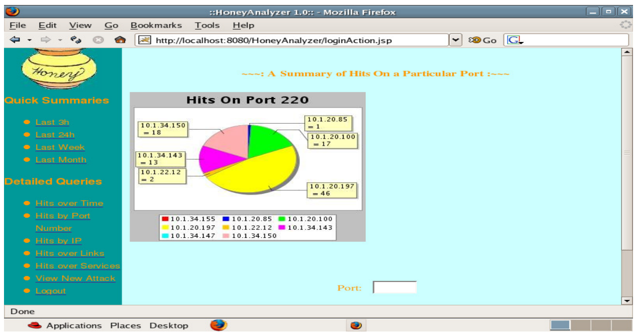

Figure 3: This is a quick summary of hits on a particular port of different machines. In this case it is port number 220.

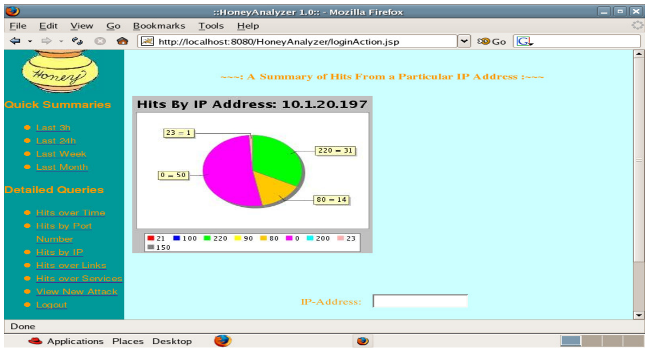

Figure 4: This is a quick summary of hits by a particular IP address e.g. the machine 10.1.20.197 has tried to access port number 220, 31 times.

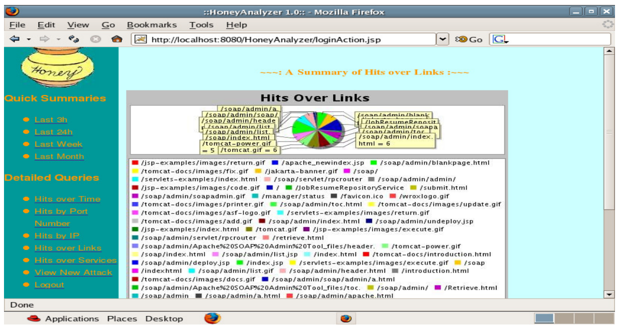
::HoneyAnalyzer 1.0:: - Mozilla Firefox
http://localhost:8080/HoneyAnalyzer/loginAction.jsp
~~~: A Summary of Hits over Links :~~~
Hits Over Links
/soap/admin/a.
/soap/admin/soap/
/soap/admin/heade
/soap/admin/list.
/soap/index.html
/tomcat-power.gif = 5
/tomcat.gif = 6
/soap/admin/blank
/JobResumeReposit
/soap/admin/soapa
/soap/admin/toc
/soap/admin/index.html = 6
/jsp-examples/images/return.gif
/apache_newindex.jsp
/soap/admin/blankpage.html
/tomcat-docs/images/fix.gif
/jakarta-banner.gif
/soap/
/servlets-examples/index.html
/soap/servlet/rpcrouter
/soap/admin/admin/
/jsp-examples/images/code.gif
/
/JobResumeRepositoryService
/submit.html
/soap/admin/soapadmin.gif
/manager/status
/favicon.ico
/wroxlogo.gif
/tomcat-docs/images/printer.gif
/soap/admin/toc.html
/tomcat-docs/images/update.gif
/tomcat-docs/images/asf-logo.gif
/servlets-examples/images/return.gif
/tomcat-docs/images/add.gif
/soap/admin/index.html
/soap/admin/undeploy.jsp
/jsp-examples/index.html
/tomcat.gif
/jsp-examples/images/execute.gif
/soap/admin/servlet/rpcrouter
/retrieve.html
/soap/admin/Apache%20SOAP%20Admin%20Tool_files/header.
/tomcat-power.gif
/soap/index.html
/soap/admin/list.jsp
/index.html
/tomcat-docs/introduction.html
/soap/admin/deploy.jsp
/index.jsp
/servlets-examples/images/execute.gif
/soap
/indexhtml
/soap/admin/list.gif
/soap/admin/header.html
/introduction.html
/tomcat-docs/images/docs.gif
/soap/admin/soap/admin/a.html
/soap/admin/Apache%20SOAP%20Admin%20Tool_files/toc.
/soap/admin/
/Retrieve.html
/soap/admin
/soap/admin/a.html
/soap/admin/apache.html

Figure 5: This is a quick summary of Hits over links of a particular server. The links correspond to services implemented on the server.

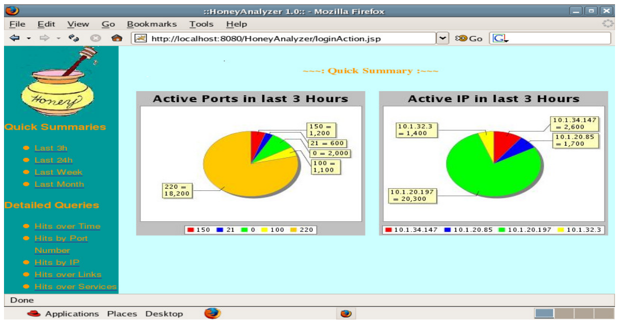

Figure 6: This is a quick summary of ports that have been active in last three hours

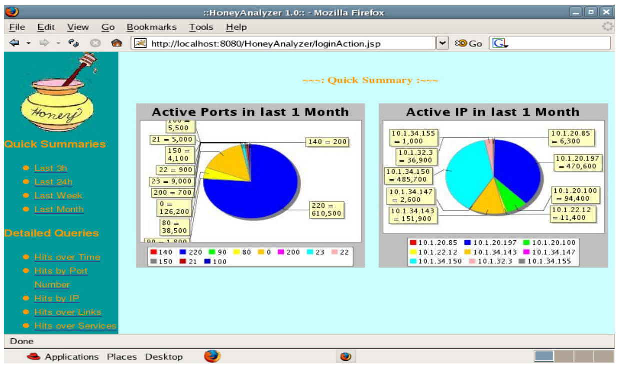

Figure 7: This is a quick summary of ports that have been active in last one month
~~~

7.3. Signature Extraction Mechanism

The graphical interface allows the Security Administrator (SA) to determine the traffic to be used to extract the signature. In this module the data filtered by the aid of graphical interface is processed for signature extraction. The signature extraction can be accomplished by application of Rule based AI technique like Fuzzy Logic and Neural Network. In this prototype the LCS algorithm is applied on the traffic of interest as against the present systems that apply LCS algorithm on entire data.

The steps followed for finding the good quality attack signature are as follows: -

i) Identify data of interest (i.e. of significance) from the database by looking at the web GUI.

ii) Analyze combined data from different data sources i.e. honeypot, filtered data from Tcpdump and Utilsnoop. For each received packet initiate the following sequence of activities:

a) If there is any existing connection state for the new packet, that state is updated otherwise new state is created.
b) If the packet is outbound, don't process the packet.
c) Perform protocol analysis [6] at the network and transport layer.
d) For each stored connection, perform header comparison in order to detect matching IP networks, initial TCP sequence numbers, etc.
e) For SOAP requests detected from different IP address, report them to the SA in a separate display area.

iii) Apply content-based string matching algorithm on the payload of interest by applying following sequence of activities:

a) If the connections have the same destination port, perform pattern detection on the exchanged messages with the help of Longest Common Substring algorithm. A description about string-based pattern detection is given in the [13].
b) If a new signature is created in the process use the signature to augment the signature pool otherwise stop the process

Given below are the sample attack signatures extracted from the captured logs –

Sample Attack Signature for DoS and Oversized Payloads:

```
ws-dos-2008-0912::
ip-proto == tcp
dst-ip != 10.10.0.0/32
dst-port = 80
http /.*&$#~!^.*\?.* Content-Length >100000/
type "Denial of Service"
::
```

Sample Attack Signature for SQL Injection:

```
ws-sql-ij-2008-0912::
ip-proto == tcp
dst-ip != 10.10.0.0/32
dst-port = 80
http SOAPSDK4:name>'</ SOAPSDK4:name
type "SQL Injection"
::
```

8. FEATURES OF THE PROPOSED APPROACH

The testing results have been shown in section 6. The semi supervised classifier is able to classify the data with 72% accuracy. The data pertaining to various attack categories when shown through graphical interface is shown through some screen shots (figures 3, 4, 5, 6 and 7) useful for analysis have been presented. The extracted signatures have also been shown. Here we list salient features of the system –

I. It is a generalized approach that is useful for generation of attacks for variety of intrusion attempts that can be made.
II. Precision of signatures is more as the data is first classified into different attack categories. This classified data is presented to the system administrator.
III. The links most frequently accessed on a web server and number of attempts to access those links can be obtained.
IV. The transport protocols being used for accessing the web services can be determined.
V. Through the pie charts, frequency of attacks made on these services can be known and the administrator can find out which services are more vulnerable to attack.
VI. This information can be used to generate precise signatures of attacks and obtain information about the attack technique.
VII. The frequency of attacks on the ports within a certain time range can be seen using pie charts. The various time ranges supported are three hours, one week, and one month.
VIII. The remote ip-addresses that visited the honeypot in a certain time range on a particular day can be viewed.
IX. The ports that are targeted by a particular IP address can be viewed.

9. Conclusion and Discussion

In this paper a generalized system architecture has been presented to allow generation of attack signatures for different types of attacks on web services. The presented system has the advantage that it can be used for generation of more balanced attack signatures for web services. The developed system is able to alert the system administrator about the attack patterns on the web services. It allows the administrator to determine the number of attacks made on different services using different transport protocols. The presented system is helpful in analyzing the attacks and extracting good quality signatures from the data logs of honeypot, traffic analyzer and SOAP interceptor tools for web services.

Administrators watch intruder's activities and then perform a tedious forensic analysis. The proposed system simplifies this process by including a separate data analysis module. The data analysis method is semi-automatic; the data is first classified into various attack categories using a SVM based classifier. The system administrator is presented with the classified data containing the attacks/accesses made on different machines graphically. Thus the data presented contains more concentrated attack information. This is done on hourly, weekly and monthly basis, therefore, detection and analysis of signatures is easy and quicker. In addition, the system helps detecting and analyzing new attack signatures. This detection is made possible through the implementation of Longest Common Substring algorithm. This system also performs an early detection of attacks and their analysis. Generation of hourly and weekly attack patterns provides this functionality. The most vulnerable services of the system can be found and the attack techniques can be observed through various log files being maintained. Frequency of attacks on links of vulnerable web servers can also be detected.

In future, attempts shall be made to perform the fully automatic analysis using some AI based tools. For instance, neural network based modules can be employed that have been trained to learn the data selection strategies adopted by the Security administrators. The signatures can be made precise by modifying the extraction algorithms that make use of a rule base. Learning capabilities can be imbibed in these algorithms which can learn and evolve and produce improved signatures.

10. References

[1] S.J. Basha, S. Cable, Galbraith,M. Hendricks, Romin Irani, James Milbery, T. Modi, Andre Tost, Alex Toussaint, "*Professional Java Web Services*", Apress, Shroff Publishers and Distributors Pvt. Ltd.

[2] WS-Security**,** OASIS identifier wss-v1.1-spec-os-SOAPMessageSecurity, [http://docs.oasis-open.org/wss/v1.1/]

[3]WS-SecureConversation,[http:/docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.html]

[4]WS-SecurityPolicy, OASIS WS-Security Policy specification, [specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf]

[5] Christian Plattner, Reto Baumann, "*White Paper: Honeypots*",http://www.inf.ethz.ch/personal/plattner/pdf/whitepaper.pdf.

[6] Erwan Lemonnier, Defcom, "*Protocol Anomaly Detection in Network-based IDSs*", http://erwan.lemonnier.free.fr/.

[7] Christian Kreibich, Jon Crowcroft, "*Honeycomb-Creating Intrusion Detection Signatures*" Using Honeypot, ACM SIGCOMM Computer Communication Review archive Volume 34,Issue1 (January 2004), Pp. 51 – 56.

[8] Urjita Thakar, Sudarshan Varma, A.K. Ramani, *"HoneyAnalyzer – Analysis and Extraction of Intrusion Detection Patterns & Signatures Using Honeypot"*, The Second International Conference on Innovations in Information Technology, Dubai, UAE September 26-28, 2005

[9] Nirmal Dagdee, Urjita Thakar,' *Intrusion Attack Pattern Analysis and Signature Extraction for Web Services Using Honeypots',* IEEE Proceedings of the First International Conference on Emerging Trends in Engineering and Technology, ICETET'08, 16-18 July 2008, Nagpur, India Pp.1232 – 1237 .

[10] Lance Spitzner, "*Honeypots: Simple, Cost-Effective Detection*", http://www.securityfocus.com/infocus/1690.

[11] Application Layer Intrusion Detection for SQL Injection, Frank S. Rietta, ACM SE'06, March 10-12, 2006, Melbourne, Florida, USA].

[12] D-SCIDS: Distributed Soft Computing Intrusion Detection System, Ajith Abraham, Ravi Jain, Johnson Thomas, Sang Yong Han, Journal of Network and Computer Applications 30 82 (2007) 81–98

[13] Snapp SR, Bretano J, Diaz GV, Goan TL, Heberlain LT, Ho C, Levitt KN, Mukherjee B, Smaha SE, Grance T, Teal DM, Mansur D. DIDS (Distributed Intrusion Detection System)—motivation architecture and an early prototype. In: Proceedings of the 14th national computer security conference, Washington, DC, October, 1999. Pp. 167–76.

[14] An e-intelligence approach to e-cmmerce intrusion detection, Symon S. Chang, Min S. Chang, 2005, IEEE, Pp 401-404

[15].Elkan, Charles, "Results of the KDD'99 classifier learning", *SIGKDD Explorating*, 2000, pp.63-64.

[16] Lee, W.and Stolfo, S., "A frame work for constructing features and models for Intrusion Detection system", *ACM Transactions on Information and system security*, 3 (4), 2000, pp.227-261.

[17] Zhihua Zhou, Shifu Chen, Zhaoqian Chen , "*FANNC: A Fast Adaptive Neural Network Classifier*" Knowledge and Information Systems (2000) 2: 115{129

[18] Eugen Barbu, Romain Raveaux, Herve Locteau, Sebastien Adam, Pierre Heroux, Eric Trupin, "*Graph Classification Using Genetic Algorithm and Graph Probing Application to Symbol Recognition*," ICPR, vol. 3, pp.296-299, 18th International Conference on Pattern Recognition (ICPR'06) Volume 3, 2006

[19] Anurag Srivastava , Eui-Hong Han, Vipin Kumar and Vineet Singh "*Parallel Formulations of Decision-Tree Classification Algorithms",* Data Mining and Knowledge Discovery, 3,237-261 (1999)

[20] Web Services Architecture, W3c Working Group Note 11 February 2004, [http://www.w3.org/ tr/2004/note-ws-arch-20040211/]

[21] Esmiralda Moradian , and Anne Håkansson, "*Possible attacks on XML Web Services*", International Journal of Computer Science and Network Security, Vol.6 No.1B, January 2006, pp154-170.

[22] Meiko Jensen · Nils Gruschka · Ralph Herkenh¨oner, "*A Survey of Attacks on Web Services - Classification and countermeasures*", Springer Verlag, CSRD (2009) 24: 185–197

[23] Hyang-Ah Kim, Brad Karp, "*Autograph: Toward Automated, Distributed Worm Signature Detection*", In Proceedings of the 13th Usenix Security Symposium (Security 2004), San Diego, CA, August, 2004. Pp. 271–286.

[24] Vinod Yegneswaran, Jonathon T. Giffin, Paul Barford, Somesh Jha, "*An Architecture for Generating Semantics-Aware Signatures*", Proceedings of the 14th conference on USENIX Security Symposium–Volume, Baltimore, MD, 2005 pp7-7

[25] Martin Roesch, "*Snort – Lightweight Intrusion Detection for Networks*", Proceedings of USENIX 13th System Administration Conference, November 1999, pp.229-238.

[26] Yuqing Mai, Radhika Upadrashta and Xiao Su, "*J-Honeypot: A Java-Based Network Deception Tool with Monitoring and Intrusion Detection*", Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 1 April 05 - 07, 2004, Pp. 804-808.

[27] Narendra S. Chaudhari, Aruna Tiwari, Jaya Thomas, "*Performance Evaluation of SVM based Semi-supervised Classification Algorithm*", IEEE proceedings of ICARCV 2008, Hanoi, Vietnam, Pp 1942-1947

[28] Niels Provos, "*A Virtual Honeypot Framework*", In Proceedings of the 13th Usenix Security Symposium (Security 2004), San Diego, CA, August, 2004, Pp. 1–14.

[29] Honeyd System, [http://www.Honeyd.org/]

[30] UtilSnoop [http://www.lanw.com/books/javasoap/default.htm],

[31] SOAPui [www.soapui.org]

[32] Fiddler [www.fiddlertool.com]

[33] Narendra S. Chaudhari, Aruna Tiwari, Urjita Thakar, Jaya Thomas, "*Semi-supervised Classification for Intrusion Detection System in Networks*", (Accepted) CIS-RAM 2010

Authors

Urjita Thakar earned her under graduate and graduate degrees in Computer Engineering in the year 1990 and 1997 from Shri G.S. Institute of Technology and Science, Indore, India. She is presently pursuing her doctoral research and working as Reader in the Department of Computer Engineering at Shri G.S. Institute of Technology and Science, Indore, India. Her major areas of interest include Information Security, Web Engineering and Computational Theory.

Nirmal Dagdee earned his undergraduate and Graduate Degrees in Computer Engineering in the year 1986 and 1990 respectively from a premier institute of central India, Shri G.S. Institute of Technology and Science, Indore, India. He earned his Doctoral degree from Rajiv Gandhi Technological University of Madhya Pradesh, India in 2003. His major fields of study include Web Engineering, Soft Computing and Computer Security.

Sudarshan Varma completed his Bachelor of Engineering in Production engineering in 1995 and completed his Master of Engineering in Computer Engineering in 2005 from Shri G.S. Institute of Technology and Science, Indore, India. His areas of interest are Information Security, Web Sciences and Database Technologies. He is presently working as Project Manager in Ideavate Solutions, NJ, USA.