XML Firewall Anomaly Detection of Web Services Attacks

Shahriar Mohammadi
mohammadi@kntu.ac.ir
Mojtaba Khaghani Milani
mojtaba.khaghani@gmail.com
Vahid Allahvakil
vahidallahvakily@gmail.com
Department of Information Technology,
Khajeh Nasir University, Tehran, Iran

Abstract

Organizations in e-commerce and business transactions are using web services more today, so the security of web services has become a challenging issue. Security mechanism like traditional firewalls even lots of web application firewalls can't protect service oriented architecture environment against attacks. Firewalls such as XML firewalls are responsible for providing security of this environment. These firewalls in the first step should be aware of content of SOAP message and XML and can protect web services against attacks. In this paper, we discuss adding anomaly detection to XML Firewall which supports the detection of new web service attacks.

Keywords: Anomaly Detection, Attack, Network Security, Web Services, XML Firewall

1. Introduction

Web service as an implementation of the service oriented architecture (SOA) has brought new capabilities and benefits. Web service has been designed to support interoperable machine to machine interaction over a network. A web service consists of three components: WSDL, SOAP and UDDI.

WSDL is a XML-based language which provides an interface to describe functionality of web service. SOAP is the protocol describing the actual request and response which is transferred between web services. UDDI is the platform independent registry and a directory for businesses to locate, publish and find their web services. All of three components of web service are based on XML. XML is platform independent language. It means that an application is written with java can communicate with an application which is written with C++.

While providing advanced business functionality, web services introduce significant security considerations and challenges [6]. For example new kind of threats created which we address some of these threats and attacks in separate section.

The Organization for the Advancement of Structured Information Standards (OASIS) and the World Wide Web Consortium (W3C) have standardized several specifications related to security in Web services and XML [7]. As an example of these standards we can mention XML-Encryption or WS-Trust. These standards provide confidentiality and Integrity of web service invocations.

In addition to these specifications, for protecting web services against threats and attacks; we need a new kind of firewalls. Traditional firewalls are not content-aware; furthermore, lots of web application firewalls are not XML-aware. We need a firewall that can scan deep into packets and not only prevent web service from general web application threats, but also parse XML/SOAP messages and prevent XML/web service specific attacks. These firewalls are referred to as XML firewalls.

XML firewalls can detect attacks base on signature and anomaly. In the former, detection is occurred base on comparing the packet to known pattern attacks, while in the latter, detection is occurred base on normal profile which defines normal activity and anything that doesn't have that behavior considered as a threat. We can compare XML firewalls to intrusion detection and prevention system; this is, XML firewalls are generally an application layer and specially SOAP layer intrusion detection and prevention system. In this article, we decide to design anomaly detection for XML firewall.

The remainder of this article organized as follows: in section 2 an overview of related works is mentioned. Section 3 presents attacks on XML and web services. Section 4 is an overview on anomaly detection. Section 5 presents anomaly detection of web service attacks. Section 6 is the validation of our work. In section 7 future works and enhancement is proposed. Finally, in the section 8 we give a conclusion of this article.

2.Related Works

Many works has been done to introduce, detect and prevent web services attacks. Meiko Jensen, Nils Gruschka andRalph Herkenhoner gave a survey of vulnerabilities in the context of web services. As a proof of the practical relevance of the threats, exemplary they performed implementation of attacks on widespread web service. Further, they discuss about general countermeasure for prevention and mitigation of such attacks [1].

Ghasem Esfahani and abdollahi azgomi in [5] addressed some of different methods suggested to overcome web service shortcoming and improved the situation by proposing a new anomaly detection method which can identify embedded attacks in SOAP structured files. Their proposed method is using kernel method among machine learning techniques.

Rahnavard, Najjar and Taherifar in [4] provided a method for the assessment of using Hidden Markov Model in web services anomaly detection (WSAD).

Krugel and Vigna in [2] represent an anomaly detection system that detects web-based attacks using a number of different techniques. Their anomaly detection system takes as input the web server log files which conform to the Common Log Format and produces an anomaly score for each web request. More precisely, the analysis techniques took advantage of the particular structure of HTTP queries that contain parameters. The parameters of the queries are compared with established profiles that are specific to the program or active document being referenced. This approach supports a more focused analysis with respect to generic anomaly detection techniques that do not take into account the specific program being invoked.

3. XML AND WEB SERVICE ATTACKS

The typical requirements for a secure system are integrity, confidentiality and availability. Any action targeting at violation of one of these properties is called an attack, the possibility for occurring an attack is called vulnerability. It is essential to understand different types of attacks on Web Services for developing a reliable and concrete XML based firewall. We categorized the attacks which we've studied in this paper as follow:

3.1XML Based Attacks

The first step has taken by a Web Service after receiving a SOAP message request is to read through, or parse the elements to extract parameters and methods to invoke. These processes allows for special kinds of attacks such as Coercive parsing, oversized payload attack.

3.1.1.Coercive parsing

Xml supports rich and complex document structures, including recursion, which can potentially cause infinite loop during processing input. Coercive parsing uses XML complexity for exhaustion CPU and memory web Server. While denial of service attacks usually requires a large numbers of massages, coercive parsing attack can be launched on a web service with a single 2KB malformed XML massage as shown below:

```
<x>
    <x>
        <x>
            …
```

3.1.2.Oversized Payload

XML parsing is directly affected by the size of the SOAP message. The total memory usage caused by processing one SOAP message is much higher than just the message size. This is due to the fact that most Web Service frameworks implement a tree-based XML processing model like the Document Order Model [1]. Using this model, an XML document – like a SOAP message – is completely read, parsed and transformed into an in-memory object representation, which occupies much more memory space than the original XML document. As a result, large amounts of CPU cycles and memory space are consumed when presented with large documents to process. A hacker can send a payload that is excessively large to deplete systems resources as shown in below.

```
<Envelope>
    <Body>
        <getArrayLength>
            <item>x</item>
            <item>x</item>
            <item>x</item>
            ...
        </getArrayLength>
    </Body>
</Envelope>
```

Oversized payload attack which results in CPU and memory exhaustion [1]

3.2Brute Force Attack

These attacks are characterized by ‘brute force’ which crack using combinations of letters, numbers and special characters in an attempt to determine which service requests and operations with what parameters result in a security Breach. Examples of these attacks are WSDL scanning and parameter tempting.

3.2.1. WSDL Scanning

Since the WSDL document includes all of the operations that are available to the consumer, it is straightforward for a hacker to run through all of the operations with different message request patterns until a breach is identified. This “knocking on every door until one opens” approach is usually effective when poor Programming practices are employed or simply the result of operations that were excluded from published WSDL documents yet are still up and running for some reason.

3.2.2. Parameter Tempting

In a WSDL document, the parameters for the operations are defined. If the attacker tries to send requests with different parameter patterns, the Web Service has the possibility to crash. For example, by submitting special characters (or other unexpected content) the back-end implementation of the Web Service can be crashed, regardless of data validation.

3.3Injection Attack

In these attacks, an attacker inserts, removes or modifies information within a message to deceive the web server which can lead to undesired effects. Examples of these attacks are XML Injection, SQL Injection and SOAP action spoofing.

3.3.1.XML Injection

An XML Injection attack tries to modify the XML structure of SOAP by inserting content – e.g. operation parameters – containing XML tags. The modified XML may allow an attacker to alter the structure of xml document which result in change of behavior of web server as shown below.

```
<Envelope>
    <Body>
        <HelloWorld>
            <a> <b>1</b> </a>
            <b> 2 </b>
        </HelloWorld>
    </Body>
</Envelope>
```

Xml Inject of <b>1</b> result in change of value of a and b in web server [1] .

3.3.2.SQL Injection

Structured Query Language (SQL) is used to manipulate database’s records. SQL statements can be inserted into input of a SOAP message by an attacker. If the statements are not sanitized, the attacker may gain access to the databases. This attack uses SQL commands such as SELECT, CREATE, UPDATE, DELETE, DROP, 1=1, ALTER and INSERT.

3.3.3.SOAP Action Spoofing

The actual Web service operation addressed by a SOAP request is identified by the first child element of the SOAP body element. Additionally, the optional http header field "SOAP Action" can be used for operation identification. Although this value only represents a hint to the actual operation, the SOAP Action field value is often used as the only qualifier for the requested operation. This operation identification enables attacker to invoke an operation different from the one specified inside the SOAP body. It is based on modification of the HTTP header as shown in below.

```
POST /Service.asmx HTTP/1.1

...

SOAPAction: "op2"

<Envelope>

    <Body>

        <op1>

            <s>Hello</s>

        </op1>

    </Body>

</Envelope>
```

The method call that was triggered by this message was op2 instead of op1 [1].

3.4DOS Attack

DOS attack attempts to slow down or completely shut down the web server so as to disrupt the service and deny the legitimate and authorized users to access it, an example of DOS attack is Replay Attacks.

3.4.1.Reply Attack

An attacker attempts to resend SOAP requests to repeat sensitive transaction and overload the web service, similar to network ping of death [6].

4.ANOMALY DETECTION OF WEB SERVICE ATTACKS

Intrusion detection systems use two main techniques to protect against attacks:

- Misuse Detection
- Anomaly Detection

Misuse detection techniques use patterns of well-known attacks or weak spots of the system to match and identify known intrusions. For example, a signature rule for the “guessing password attack” can be defined as: “there are more than four failed login attempts within two minutes.”

Misuse detection techniques are not effective against novel attacks that have no matched rules or patterns. Developing ad hoc signatures to identify these attacks is a time-intensive and error-prone activity which requires substantial security expertise. In addition, it is hard to keep signature sets updated with respect to the large numbers of vulnerabilities discovered daily. To overcome these issues, misuse detection systems should be composed with anomaly detection systems, which support the detection of new attacks. Anomaly detection techniques can be effective against unknown or novel attacks since no a priori knowledge about specific intrusions is required. Anomaly detection systems can be trained to spot attacks against custom-developed web-based applications; However, anomaly detection systems tend to generate more false alarms than misuse detection systems because an anomaly can just be a new normal behavior [2][3].

As mentioned in the related works section, Krugel and Vigna in [2] represent an anomaly detection system that detects web-based attacks using a number of different techniques. Their anomaly detection system takes a web server log files as an input, which conform to the Common Log Format and produces an anomaly score for each web request. More precisely, the analysis techniques used by the tool take advantage of the particular structure of HTTP queries that contain parameters. The parameters of the queries are compared with established profiles that are specific to the program or active document being referenced. This approach supports a more focused analysis with respect to generic anomaly detection techniques that do not take into account the specific program being invoked.

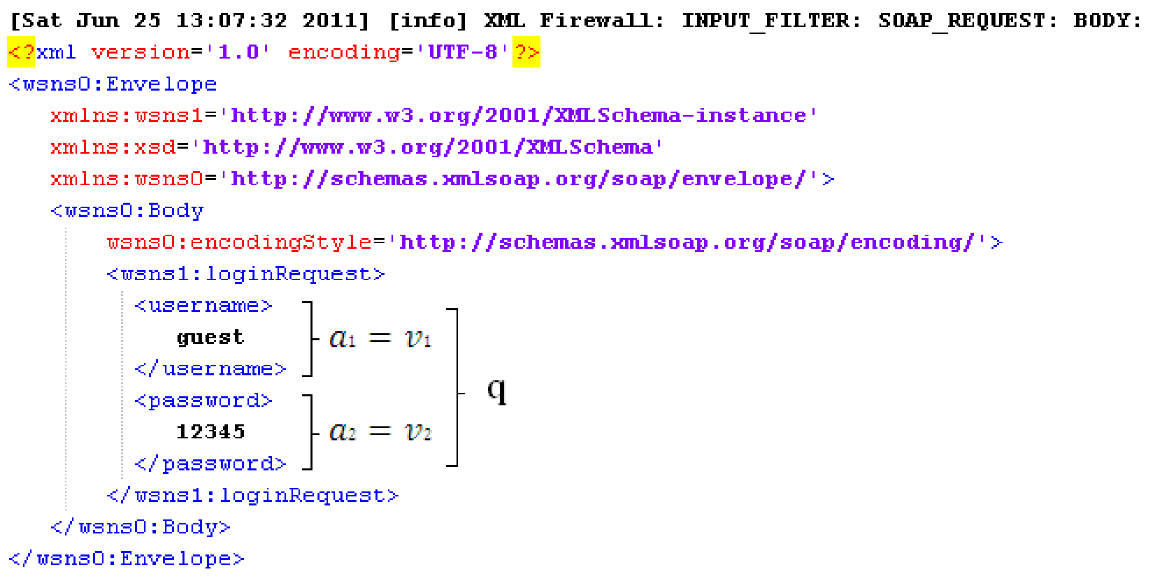
The model of anomaly detection of web service is the extended form from the model of anomaly detection of web based attacks. In the next part, we explain the data model and detection model proposed for web based attacks in [2] which are extended to anomaly detection of web services attacks.

4.1 Data Model

Our anomaly detection analyzes SOAP requests as logged by XML Firewall. SOAP requests use parameters to pass value to the web service. Formally, the input to the detection process consists of an ordered set S = {s1, s2... sm} of successful SOAP requests.

A SOAP request si composed of a header and a body. The body itself consists of a list of attributes and values. These attributes and values are used for passing the parameters to the web service and can be represented as a ordered list of n pair of attributes with their corresponding values namely q. That is, q = (a1, v1), (a2, v2). . . (an, vn) where ai "∈" A, the set of all attributes, and vi is a the value for attribute ai. The set Sq is defined as the subset {aj . . . ak} of attributes of q. **Figure 1** shows an example of an entry from a XML Firewall log and the corresponding attributes and values that are used in the analysis. For this example, attribute and value represent as q and Sq = {a1, a2}.

Anomaly detection algorithm runs on each attribute and its values on S and if attribute and its values are detected as an attack; they will be removed from set S. In fact, the set S represent the functional requests to the web service and each entry in set S represents a unique request and in our model, the detection of attacks for each web request is performed separately.



```
[Sat Jun 25 13:07:32 2011] [info] XML Firewall: INPUT_FILTER: SOAP_REQUEST: BODY:
<?xml version='1.0' encoding='UTF-8'?>
<wsns0:Envelope
   xmlns:wsns1='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:wsns0='http://schemas.xmlsoap.org/soap/envelope/'>
   <wsns0:Body
       wsns0:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
       <wsns1:loginRequest>
         <username>
            guest
         </username>
         <password>
            12345
         </password>
       </wsns1:loginRequest>
   </wsns0:Body>
</wsns0:Envelope>
```

Figure 1- Sample XML Firewall Log Entry

4.2. Detection Model

The anomaly detection process uses a number of different models to identify anomalous entries within a set of input requests Sp associated with service p. A model is a set of procedures used to evaluate a certain feature of an element and corresponding values(e.g., the string length of an attribute value) or a certain feature of them as a whole (e.g., the presence and absence of a particular attribute). Each model is associated with an attribute (or a set of attributes) of a web service by means of a profile. Consider, for example, the string length model for the credit card attribute. In this case, the profile for the string length model captures the 'normal' string length of credit card attribute which is 16-19. The task of a model is to assign a probability value to the query's attributes. This probability value shows the probability of the happening of the given value within the profile. As the possibility of happening of value becomes lower, the probability of attack grows higher.

Based on the probability of a value for specified attribute, a decision is made; that is, the attribute is either reported as a potential attack or as normal. This decision is made by calculating an anomaly score individually for each element attribute. When one or more anomaly scores exceed the detection threshold, the whole query is marked as anomalous.

The anomaly score value is calculated using a weighted sum for each attribute m in the model as shown in Equation 1. In this equation, Wm represents the weight associated for the importance of the attribute, while P_m is its returned probability value. The probability P_m is subtracted from 1 because a value close to zero indicates an anomalous event that should yield a high anomaly score.

$$Anomaly\ Score = \sum_{m \in Models} \mathrm{wm} * (1 - \mathrm{pm})$$

Equation 1

Every anomaly model needs a training time to evaluate a normal behavior; similarly, our model can operate in one of two modes, training or detection. The training phase is required to determine the characteristics of normal events (that is, the profile of a feature according to a specific model) and to establish anomaly score thresholds to distinguish between regular and anomalous inputs. This phase is divided into two steps. During the first step, the system creates profiles for each server-side program and its attributes. During the second step, suitable thresholds are established. This is done by evaluating queries and their attributes using the profiles created during the previous step. For each program and its attributes, the highest anomaly score is stored and then, the threshold is set to a value that is a certain, adjustable percentage higher than this maximum. Once the profiles have been created; that is, the models have learned the characteristics of normal events and suitable thresholds have been derived – the system switches to detection mode. In this mode, anomaly scores are calculated and anomalous queries are reported.

The following sections describe one of the algorithms that analyze attribute length for detecting malicious activity. An explanation of the model creation process (i.e., the learning phase) is included.

4.3.Attribute Length

In many cases, the length of a query attribute can be used to detect anomalous requests. Usually, parameters are either fixed-size tokens (such as session identifiers) or short strings derived from human input (such as fields in an HTML form). Therefore, the length of the parameter values does not vary much between requests associated with a certain program. The situation may look different when malicious input is passed to the program. For example, to overflow a buffer in a target application, it is necessary to ship the shell code and additional padding, depending on the length of the target buffer. As a consequence, the attribute contains up to several hundred bytes.

- Learning

We approximate the mean $\dot{\mu}$ and the variance $\dot{\sigma}^2$ of the real attribute length distribution by calculating the sample mean μ and the sample variance σ^2 for the lengths l_1, l_2 ,. . . , l_n of the parameters processed during the learning phase (assuming that n queries with this attribute were processed).

- Detection

Given the estimated query attribute length distribution with parameters μ and σ^2 as determined by the previous learning phase, it is the task of the detection phase to assess the regularity of a parameter with length l.The probability of l can be calculated using the Chebyshev inequality shown below.

$$p(|x - \mu| > t) < \frac{\sigma^2}{t^2}$$

Equation 2

The Chebyshev inequality puts an upper bound on the probability that the difference between the value of a random variable x and μ exceeds a certain threshold t, for an arbitrary distribution with variance $\sigma^2$and mean μ. This upper bound is strict and has the advantage that is does not assume a certain underlying distribution. We substitute the threshold t with the distance between the attribute length l and the mean μ of the attribute length distribution (i.e., |$l - \mu$ |). This allows us to obtain an upper bound on the probability that the length of the parameter deviates more from the mean than the current instance. The resulting probability value p(l) for an attribute with length l is calculated as shown below.

$$p(|x - \mu| > |l - \mu|) < p(l) = \frac{\sigma^2}{(l - \mu)^2}$$

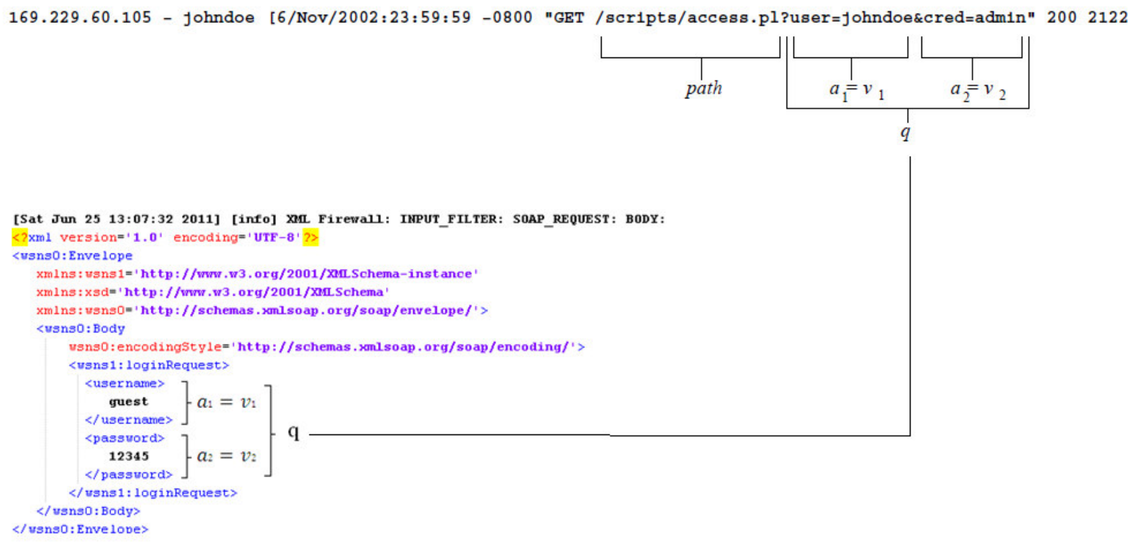
Equation 3

This is the value returned by the model when operating in detection mode. The Chebyshev inequality is independent of the underlying distribution and its computed bound is, in general, very weak. Applied to our model, this weak bound results in a high degree of tolerance to deviations of attribute lengths given an empirical mean and variance. Although such a property is undesirable in many situations, by using this technique only obvious outliers are flagged as suspicious, leading to a reduced number of false alarms.

5. VALIDATION

In this section, we want to show that we can extend proposed anomaly detection system for web based attacks in [2] to anomaly detection of web services attacks. The model as mentioned in previous section has two parts of data model and detection model.

The different between two data models is that anomaly detection of web uses URL but our model use SOAP requests. SOAP requests is a tree of elements and leaves of this tree is elements value. If we put this elements and related value in the ordered list we get the S set presented data model in previous section. This list consists of pair of attributes and their values. In anomaly detection of web, this list is made by URLs and query of URL. This correspondence of two data models is shown at **Error! Reference source not found.**.



```
169.229.60.105 - johndoe [6/Nov/2002:23:59:59 -0800 "GET /scripts/access.pl?user=johndoe&cred=admin" 200 2122
```

```
[Sat Jun 25 13:07:32 2011] [info] XML Firewall: INPUT_FILTER: SOAP_REQUEST: BODY:
<?xml version='1.0' encoding='UTF-8'?>
<wsns0:Envelope
   xmlns:wsns1='http://www.w3.org/2001/XMLSchema-instance'
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:wsns0='http://schemas.xmlsoap.org/soap/envelope/'>
   <wsns0:Body
      wsns0:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
      <wsns1:loginRequest>
        <username>
           guest
        </username>
        <password>
           12345
        </password>
      </wsns1:loginRequest>
   </wsns0:Body>
</wsns0:Envelope>
```

Figure 2- Different Between Web Service Model and Web Model

Detection model and algorithm which is used in anomaly detection is completely the same for web and web service. In this article, we just considered attribute length for anomaly detection model. We can extend other models proposed for anomaly detection of web based attacks to web service based attacks.

Consider XML Firewall log file consists of these ten elements with corresponding value and related length:

Table 1- Sample XML Firewall log file

attribute and Value	Length
<password> 75344598 </password>	8
<password> 348870923 </password>	9
<password> 2829306 </password>	7

<password> 06298684 </password>	8
<password> 207765694 </password>	9
<password> 453639 </password>	6
<password> 89683564 </password>	8
<password> 6463349 </password>	7
<password> 06585335 </password>	8
<password> 0129584087 </password>	10

After calculating we get μ = 8 and σ^2 = 1.2. Putting the values in the equation 3 yiels:

$$p(l) = \frac{1.2}{(l - 8)^2}$$

Equation 4

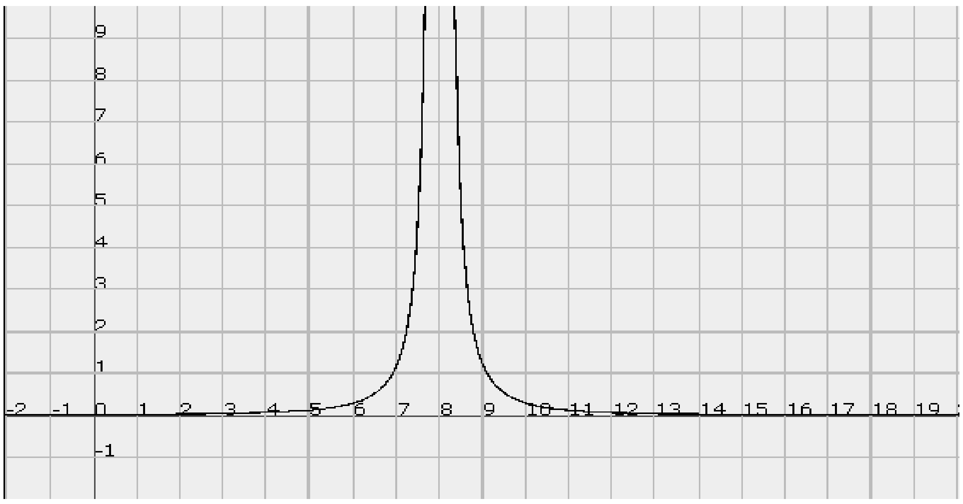


Figure 3 – Visual Representation of Equation 4

Error! Reference source not found.shows the plot of function equation 4. As an example with l1 = 20, p(l1) = 0.0084 and l2=9 , p(l2) = 1.2. If the treshold determi
ned 10% from mean it yield SOAP request with length l1is anomalous and request with length l2 normal.
Error! Reference source not found. shows the general view of anomaly detection process

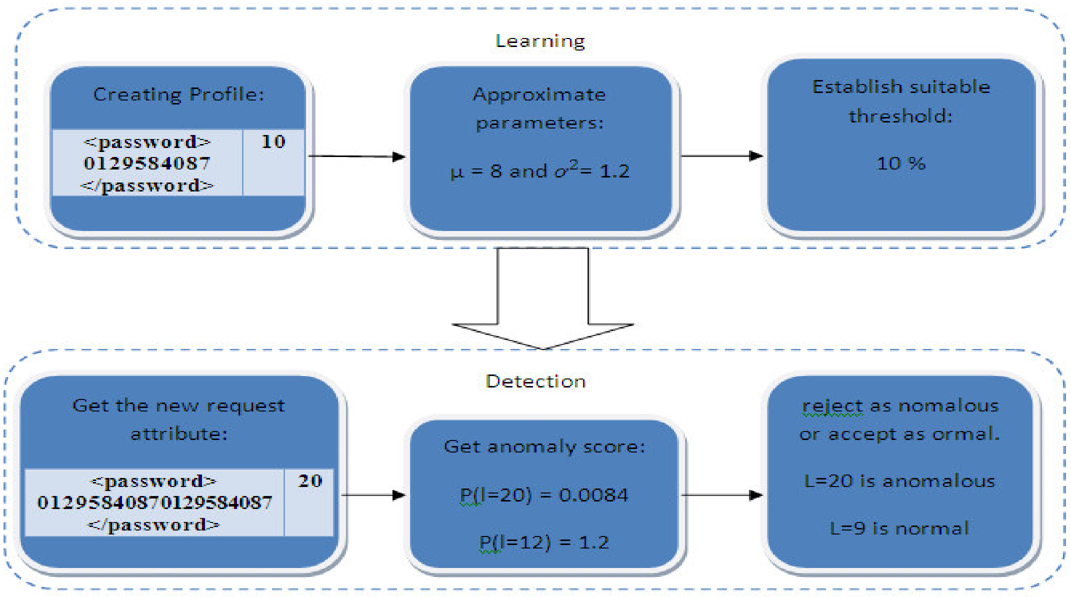

Learning
Creating Profile:
<password> 0129584087 </password>
10
Approximate parameters:
μ = 8 and σ2= 1.2
Establish suitable threshold:
10 %
Detection
Get the new request attribute:
<password> 01295840870129584087 </password>
20
Get anomaly score:
P(l=20) = 0.0084
P(l=12) = 1.2
reject as nomalous or accept as ormal.
L=20 is anomalous
L=9 is normal

Figure 4- General view of anomaly detection process

6.Conclusion

In this article, we've discussed about the results of adding anomaly detection of web services attacks to XML firewall. This model of anomaly detection of web service attacks is the extension of the anomaly detection of the web based attacks proposed in [2]. The model consists of data model and detection model which introduced for web services and SOAP requests. Anomaly detection process uses XML Firewall log files of SOAP requests as an input and has two parts of training and detection. In the training phase, a profile is made for each element of SOAP request for each service; then anomaly detection system obtains calculate anomaly score based on the attribute and the values of the request. In the training phase, anomaly detection system obtains mean μ and variance σ^2 of the length of all elements value length of a service.

In the detection phase, anomaly detection system checks the probability of happening an attack and obtains anomaly score of the request and based on the policy and threshold which is determined in the training phase; tags the request as an attack and omits it or considers it as normal traffic and passes it through. Finally, we've validated the model by showing the process of anomaly detection on the length of SOAP requests in the case of site authentication.

In the future works, we will evaluate the scalability of this approach and see the effect of this approach on the performance and throughput of XML firewall. Also, we will consider anomaly detection of other models of attributes of SOAP requests. For example attribute character distribution, structural inference, presence or absence of attributes and attributes order.

References

[1] Ralph Herkenhoner Meiko Jensen Nils Gruschka, "A Survey of Attacks on WebServices," 2009.

[2] C. and Vigna, G Kruegel, "Anomaly detection of web-based attacks. In Proceedings of," *10th ACM conference on Computer and communications security*, pp. 251-261, 2003.

[3] W. Lee and S. Stolfo., "A Framework for Constructing Features and Models for Intrusion Detection Systems," *ACM Transactions on Information and System Security*.

[4] G. Rahnavard, M.S.A. Najjar, and S. Taherifar, "A method to evaluate Web Services Anomaly Detection using Hidden Markov Models," , Dec. 2010.

[5] R.G. Esfahani and M.A. Azgomi, "Towards an anomaly detection technique for web services based on kernel methods," in *Innovations in Information Technology*, Dec. 2009.

[6] Don Patterson, "XML Firewall Architecture and Best Practices for Configuring and Auditing," 2007.

[7] Nils Agne Norbotten, "XML and Web Services Security Standards," *IEEE COMMUNICATIONS SURVEYS & TUTORIALS*, vol. 11, 2009.