Implementation of IDS for Web Application Attack using Evolutionary Algorithm

Saba Khan
Student of Master of Engineering,
Department of Computer Engineering,
Vidyalankar Institute of Technology
Mumbai University, India
Saba.khan@vit.edu.in

Prof. Dilip Motwani
Associate Professor,
Department of Computer Engineering,
Vidyalankar Institute of Technology
Mumbai University, India
Dilip.motwani@vit.edu.in

***Abstract*— Web application security is a threat to the world's information technology infrastructure. The most widely used accepted solution to this threat is to deploy an Intrusion Detection System(IDS). Such systems currently rely on either signature of the attack or changes in the behavior patterns of the system to identify an intruder. These systems, either signature-based or anomaly-based, are readily understood by attackers. Problem occurs when attacks are not detected by the existing IDS because the attack does not fit the pre-defined attack signatures. This work intends to address this problem. In this paper, we implemented two IDS model which includes an anomaly detection technique measured by cross entropy and a signature-based attack detection using genetic algorithm. Both the methods, that is signature based as well as anomaly based IDS are used to detect even more attacks than either approach could detect alone. By compiling each of the approaches, additional false positive and false negative attacks can be discovered as well. Those included in this work are SQL injection, cross-site scripting and remote file inclusion.**

***Index Terms*— Intrusion Detection System, Application layer attack, Anomaly-based IDS, Cross entropy, Signature-based IDS, Genetic algorithm**

I. Introduction

There are two types of Intrusion Detection System: Signature-based IDS and Anomaly-based IDS. Signature based IDS depends upon the signature to identify the attacks. As long as signature are specified ahead of time this approach is accurate. This can lead to several false positive as signature based IDS will not be able to detect new attack. Whereas the anomaly based IDS focuses on behavior pattern of the system. To implement anomaly-based IDS, two phases are used: Training phase and the testing phase. In training phase, normal behavior of the system is observed in the absence of malicious activity. Based on these observation, a profile is build. In testing phase, the current behavior of the system is compared with the stored profile and any deviation from the stored profile is considered as attack on the system. Anomaly based IDS suffers from false negative alarms. Since even a minor deviation from the stored profile will alert the attack. Both the types of IDS have its own benefits and limitation. Hence, we have used both the approaches in our system

A. Overview of Common Web Attack

This paper focuses the detection of three most reported application layer attacks: SQL Injection(SQLI), Cross-Site Scripting(XSS), Remote File Inclusion(RFI). XSS occurs when a malicious script is injected into trusted website. This occurs due to failure of input validation. SQL injection is an attack where attackers inject SQL queries through the input of a web page. The target is to bypass server level and gain access to the backend. Remote File inclusion (RFI) is a type of inclusion attack wherein an attacker can cause the web application to include a remote file by exploiting a web application. The web application may trick to include remote files with malicious code at a time of taking user input.

II. PROPOSED APPROACH

1. We have implemented both anomaly-based IDS and signature-based IDS as a separate module
2. Anomaly-based IDS is developed using cross-entropy measure
3. Whereas Signature-based IDS is developed using Genetic Algorithm

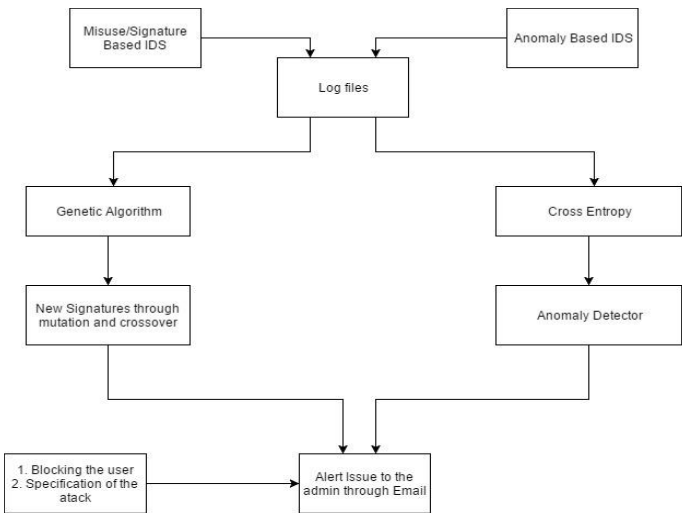
Misuse/Signature Based IDS
Anomaly Based IDS
Log files
Genetic Algorithm
Cross Entropy
New Signatures through mutation and crossover
Anomaly Detector
1. Blocking the user
2. Specification of the atack
Alert Issue to the admin through Email

Figure 1. Overall Architecture of our Proposed approach

III. ANOMALY-BASED IDS DEVELOPMENT

In this work, we are focusing on log files to apply Cross Entropy(CE) metrics for the detection of anomalous request by the attacker. For profiling and attack detection we apply cross entropy for three which are parameter name, value and its type.

```
1./IDS2/GetProduct?manufacturer=Logitech&category= Webcam

2. /IDS2/GetProduct?TV=LG
```

Figure:2 Example of Log Data

In the above example, there are two requests that we gather by deploying a web application. In this request 1 wants to access the resource /IDS2/GetProduct? and has the list of parameters such as manufacturer and category with associated values Logitech and webcam. The second request accesses the same resource as the first one. However, it has one parameter (TV) with the value LG. Therefore, web applications can allow a user to access various sets of parameters and values with a single resource path. Attack cannot be detected by only relying on the parameter sequence where the sequence remains the same with different parameter, value and type (e.g.,SQLI). Our proposed IDS uses three types of measure. For each same path the anomaly detector profiles three types of information they are: parameter set, parameter value set and parameter value data type. The algorithm for the same is given below:

```
Input: F: Input log file Output:
R: set of unique resource path r
Pr: set of parameter for r
Tr: type of parameter
Vr: value set for each parameter
1. For each line I ∈ F
2. r=getResourcePath(I)
3. if r ∉ R
4. R=R ∪ r
5. Pr=getParam(l)
6. for each pi ∈ Pr
7. vi=getValue(pi)
8. ti=getType(vi)
9. updateValueFreq(pi,vi)
10. updateTypeFreq(pi,ti)
11. updateParamFreq(Pr)
```

Algorithm 1 : Processing of URLs from a log file

A. Detection of Web Application Attacks with Cross-Entropy

We apply Cross entropy to profile the randomness of parameter occurrences, parameter values, and value types. Cross-Entropy(CE) measure between two request is calculated as follow:

$$CE (a, b) = - \Sigma i\ a(xi) * \log2(b(xi)) \qquad (1)$$

Here, a(xi) is the probability of xith element occurrence from a set, and b(xi) is the probability of xith element occurrence from b set. CE becomes minimum if both a and b are identical. The computed CE decides whether the request is anomalous or not. We define a threshold level 'd' which, if exceeds, will generate an alert to the admin. If the CE does not exceed threshold, we consider it normal request. For anomaly IDS, we use three measures: cross-entropy of parameter (CEP), cross-entropy of value (CEV), cross entropy of type (CET).

CEP is used to check any injected parameter or missing parameter

$$CEPr (P1,P2) = - \Sigma i\ P1(xi)*\log2(P2(xi)) \qquad (2)$$

We used a smoothing value to avoid the zero occurrence of any parameter by replacing zero with a very small probability value (as the logarithm of zero probability cannot be calculated)

CEV is used to find the deviation between the value of new request and the stored request for a given parameter.

$$CEVp (V1,V2)= - \Sigma iV1(xi)*\log2(V2(xi) \qquad (3)$$

CET is used to find the deviation between data type of new parameter value and stored parameter value. It increase the attack detection.

$$CETr (T1, T2)= - \Sigma iT1(xi)*\log2(T2(xi) \qquad (4)$$

These metrics are used to calculate deviations between the stored and the new request and if threshold increases it raise an alert to the admin. For the training phase, we used our deployed web application to perform various operation with different inputs (normal request with no malicious request) and the logs are generated for the same. These logs will be stored in the database and when a new request comes the cross entropy between them is calculated and the admin is alerted if threshold exceeds its limit

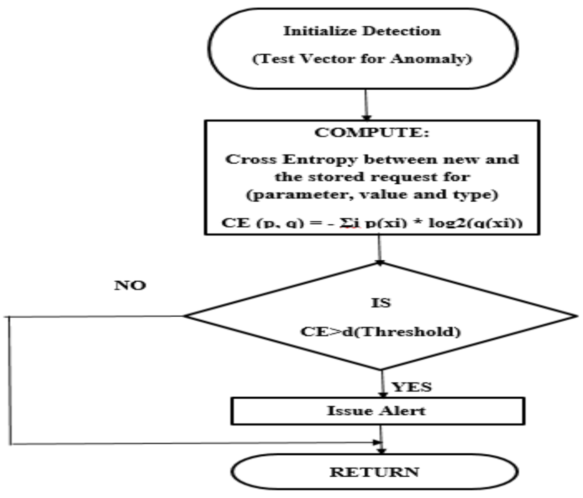
Initialize Detection
(Test Vector for Anomaly)
COMPUTE:
Cross Entropy between new and the stored request for (parameter, value and type)
CE (p, q) = - Σi p(xi) * log2(q(xi))
NO
IS
CE>d(Threshold)
YES
Issue Alert
RETURN

Figure: 3 Testing Phase of Anomaly-based IDS

IV. SIGNATURE-BASED INTRUSION DETECTION SYSTEM DEVELOPMENT

Traditional Signature-Based Intrusion Detection Systems (IDSs) is reliable till the attack type is known in advanced and their signature are available in the database before the attack actually occurred. To address the limitation of existing signature based IDS, in this paper, we propose to develop a Genetic Algorithm (GA) based IDS. In existing signature based attack detection approach, any signatures with even minor deviations from the attack descriptions would not generate alarms, which leads to false positive. However, our GA-based IDS approach can address this limitation by generating new signatures from existing signatures.

A. GA-Based IDS Development

It is implemented using genetic algorithm. In this approach log file is converted into chromosomes in a binary string representation and genetic algorithm is applied to generate signature of the attack. The goal is to generate new signature from the existing signature by applying mutation and crossover.

For the implementation:
1. We deploy a web application
2. The applications were provided with attack inputs and the logs are generated. The input for the attack is taken from OWASP.
3. These attack inputs are applied randomly within web requests from a browser.

B. *Steps of GAIDS*

1. Convert each web request into chromosomes
2. Evaluate the fitness of each chromosomes
3. Provide another set of chromosomes as output followed by certain number of iteration while following crossover and mutation.

Now let us see how these web log data are converted into chromosomes and how new signatures are generated using GA approach:
Below are the example of log data for a SQL injection, XSS and RFI attack

1. LOG DATA FOR SQL ATTACK:

```
"GET/sqlinj/?id=1%27+or+%271%27+%3D
+%271%27%29%29%2F*&Submit=Submit&user_token
=c14e5f424d9f279c19ba507492……
```

2. LOG DATA FOR XSS ATTACK:

```
"GET/xss_r/?name=%3CIMG+SRC%3DJaV
aScRiPt%3Aalert%28%26quot%3BXSS%26      quot%3B%
29%3E&user_token=f37e5a82a994725092fd 3155bb8 ……
```

3. LOG DATA FOR RFI ATTACK:

```
"GET/?FORMAT={${include("http://www.veryba
dwebsite.com/hacker.txt")}}{${exit()}}…
```

Step 1: Genetic Algorithm accepts a set of population as input, which in this case comes from the log data and convert it into binary string representation. This is done differently for all the three types of attack as shown below:

i)For SQL attack, the chromosomes are generated as below:

# of SQL keywords	**Presence of encoded character**	**# of fields with SQL keyword**	**Attack type**
010	1	001	0001

In this chromosome generation of SQL injection, we used three block of information, and then on the basis of what keywords present the last block will take the decision of attack type, Number of SQL keywords is found, based on log data above it is two (OR, =) for SQL, presence of an encoded character (1=Yes, 0=No), number of input fields that have SQL keywords (in this case it is one). There are six common types of SQL injection attacks. Hence, we reserve three bits to express various types of attacks such as tautology (0001), piggybacking(0010),union(0011), storedprocedure (0100),blind (0101), timing (0110).

ii) For XSS attack, the chromosomes are generated as below:

# of script/html keywords	**Presence of encoded character**	**# of fields with XSS keyword**	**Attack type**
011	1	001	0111

In this chromosome generation of XSS attack, similar type of fields are being used except the first field which is number of script/html keywords present in the log data. There are three type of XSS attacks such as stored(0111), reflected(1000) and DOM based XSS(1001)

iii) For RFI attack, the chromosomes are generated as below:

# of URLs	**Encoded**	**# of commands**	**Attack type**
001	0	001	1100

In this chromosome generation of RFI attack, all the fields are same as XSS except first and third. In first field number of URLs are calculated and in third number of commands. On the basis of log data mentioned above we have one URL, not encoded, and one command. There are three types of RFI attack such as URL only(1010), CMD only(1011), both URL and CMD(1100).

Steps 2: We Evaluate fitness functions for the chromosome, the more the fitness is the more the probability that the given chromosome will be selected to undergo crossover. In this work, fitness depends upon how many attack are detected by a chromosome.

Step 3: The chromosomes are crossed over based on fitness level. We apply one point cross over in this. For example: consider two chromosome C1 and C2

Before crossover

C1	010	1	001	0001
C2	011	1	001	0111

After crossover

C1	010	1	001	0111
C2	011	1	001	0001

After crossover, if we consider chromosome C1 for mutation

Before Mutation

010	**1**	001	0111

After Mutation

010	**0**	001	0111

In this we uses a bit flip mutation, in which one or more bit randomly chosen is flipped.

In the above chromosome C1, the fourth bit is flipped from 1 to 0 and hence mutated to generate new signature (01000010111), where attack payload is not encoded

V. SOFTWARE AND HARDWARE USED

1. Language: JDK (1.8.0)
2. Web Technologies: HTML 5, JSP, Servlets
3. Web Server: Apache Tomcat 9.0
4. Backend: MYSQL 5.5
5. Development Environment: NetBeans IDE 8.2
6. Operating System : windows 10
7. Processor : 6th Gen Intel Core i5 (6200U)
8. Hard Disk : 1 TB
9. RAM : 8GB

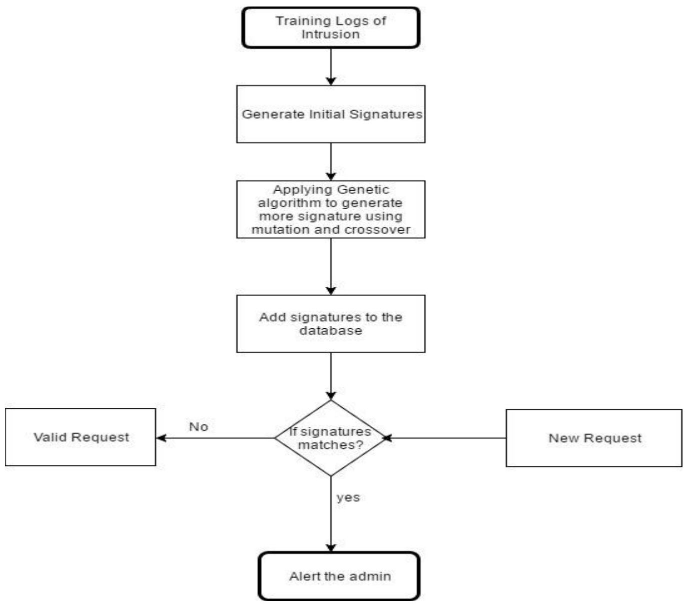
Training Logs of Intrusion
Generate Initial Signatures
Applying Genetic algorithm to generate more signature using mutation and crossover
Add signatures to the database
Valid Request
No
If signatures matches?
New Request
yes
Alert the admin

Figure 4. Flow of GA-Based IDS

VI. CONCLUSION

In this work, we implemented both anomaly based IDS as well as Signature based IDS for three most reported web attacks SQLI, XSS and RFI. For anomaly based IDS, we employ cross entropy measures for parameter, value and data type of a web request. For Signature based IDS, we employ genetic algorithm with a goal of generating new signature from the existing signature using mutation and crossover. This approach can address the limitations of existing signature based IDS. The more the population size of chromosome becomes the more it will be able to detect new attack. The results of this work can benefit and influence decisions of network and system administrators at organizations who use an existing IDS and those who may be looking for a more advanced IDS model. A hybrid model can be implemented by combining the above two approaches. We plan to include more web application attack. In our future work, we will compare our proposed techniques with the other existing technique for web attack detection.

REFERENCES

[1] Park, Y. and Park, J. 2008. Web Application Intrusion Detection System for Input Validation Attack. Proceedings of the 11th International Conference on Computer and Information Technology, pp. 497-504.

[2] Le, M. and Stavrou, A. 2012. DoubleGuard: Detecting Intrusions in Multitier Web Applications. IEEE, pp. 512-525.

[3] Robertson, W., Vigna, G., Kruegel, C. and Kremmer, R. 2006. Using Generalization and Characterization Techniques in the Anomaly-based Detection of Web Attacks (NDSS), San Diego, California, USA.

[4] Lee, W. and Xiang, D. 2001. Information-theoretic measures for anomaly detection. In IEEE, California, USA, pp. 130-143.

[5] Bronte, R., Shahriar H. and Haddad, H. 2016. Information Theoretic Anomaly Detection Framework for Web Application. IEEE (COMPSAC), Atlanta, GA, USA, pp. 394-399.

[6] Avancini, A. and Ceccato, M. 2010. Towards security testing with taint analysis and genetic algorithms.ACM, New York, NY, USA, pp. 6571. Doi: 10.1145/1809100.1809110

[7] The Open Web Application Security Project. Top Ten 2013

[8] Bronte, R., Shahriar H. and Haddad, H. 2016. A Signature-Based Intrusion Detection System for Web Applications based on Genetic Algorithm. ACM, New York, NY, USA, 32-39. Doi:10.1145/2947626.2951964