Statistical Analysis of Threat Detection Framework for Securing Semantic Web Amenities

[1]Nagendra Kumar Singh, *[2]Sandeep Kumar Nayak, [3]Mohammad Faisal

Abstract

The emergence of semantic web amenities has momentously streamlined accustomed life. Semantic web amenities have emerged as a medium through which conformist people can access various types of amenities quite easily. Digital trade is an emerging semantic web service that millions of users use in daily life. Through these amenities, the user acquires various types of information, as well as shares his confidential information. These amenities use assortments of database programs that preserve disparate types of data. That is why it is necessary that assorted types of database programs are used instead of the one type of database program to preserve disparate types of data available on semantic Web amenities. In this research paper, the utility of various types of database programs to preserve different types of data is explained through statistical analysis. In this paper, the chi-square test is used for statistical analysis. This paper shows how a single database program is not relevant to preserving different types of data and also shows how different types of database programs are capable of unprecedented increases in the functionality of semantic web amenities.

Keywords: *Access Control, Dispersed database ploy, Digital-trade, Semantic Web, Chi-square test*

I. Introduction

With the progressive development of semantic Web amenities, the availability of various types of amenities has been easily achieved. Semantic web amenities have an invaluable contribution if the user is able to avail various types of commercial and public amenities. Many new and modern applications of semantic web amenities such as digital trade, banking applications, online reservation, Thesauri, concept-based searches, etc. have emerged [1]. Meanwhile, a novel concept known as semantic Web of things (SWoT) have emerged that enables integration of new kind of resources known as things such as AC, refrigerator, microwave oven etc. with Web protocols [2]. Semantic web is not a neoteric type of web, although it is an augmentation of the prevalent web in which information is depicted in depth so that web amenity users can easily use these features.

A variety of amenities can be availed through the Semantic Web, of which digital trade has emerged as a core service. With the help of digital trade semantic web service, the user can get information of various types of products online at home and can also order the products of his choice. These types of amenities store peculiar typefaces of data and it becomes exceedingly indispensible for the service provider to choose the appropriate database programs to store the peculiar typefaces of data attainable on these amenities. Digital trade semantic web services mainly consist of two web servers. One web server conserves application logic and the other web server, known as database server, stores the data of users and digital trade service. Digital trade web service's confidential and important data remains on the database server. That is why it becomes necessary to protect these database servers from insider and outsider attacks.

Keeping these requirements in mind, a Framework for Integrated Threat Detection on Semantic Web Services was introduced to secure digital trade semantic web services [3]. The framework presents a dispersed database ploy for digital-trade Web amenities. In framework, a set of database programs is insinuated to handle the assorted type of data. Securing Clandestine data available on database and application servers is also an important issue. To solve this problem, it is insinuated to use a combination of IP-MAC addresses on both the client and server end. The insinuated ploy stores the IP and corresponding MAC address of both client and server during first interaction with both ends.

This paper presents the statistical analysis of the threat detection framework for securing the digital trade semantic Web amenities [3]. Chi-square test has been used for statistical estimation of the presented framework. The chi-square test is performed with the help of the R Studio Tool. The results obtained during the implementation phase of the framework, are validated using the chi-square test. The empirical validation has been followed to validate the efficiency of the framework in securing the digital trade semantic Web amenities against internal threats as well as improving the overall performance of the framework for facilitating the better services to its potential users.

The paper is framed as follows. Section 2 describes the related work done in the field of semantic Web services. Section 3 elaborates the outcome of chi-square test on resultant data obtained by implementing the proposed framework for digital trade Web amenity. Finally, conclusion and future work is given in section 4.

II. Related Work

The ratification technique is the utmost arrogate practice to scrutinize the accomplishment and efficacy of the proposed scaffold. In [4], two ceremonial frameworks for verification are fathomed; Bryand's delineate and Zeus's delineate. Moreover, decorous corroboration is typified with the help of Bryand's scaffold and Zeus's scaffold. The authors [5] are familiarizing the ratification scaffold of an indoor navigation routine for blind and visually impaired (BVI) users. The guidelines and procedures for verification and the validation process of an occupational likelihood ratio calculation tool is discussed [6]. The partiality and discrepancy of prototypical ratification practices in the field of gaffe prophecy are reconnoitered in [7]. The authors [8] piloted and devised a pragmatic exploration to gauge technology in fabrication milieus.

III. Statistical Analysis of Integrated Framework for Securing Semantic Web Services

The integrated threat detection framework [3] has been implemented on Badhaliya Gems digital-trade semantic Web service [9]. The framework has been implemented on both types of database design i.e. well-structured and messy database design. The framework secures the semantic Web services by introducing the concept of IP-MAC trussing. The Web server stores the IP and MAC address of the registered client. The client computer stores the IP and MAC address of the Web server. During communication, the Web server first obtains the IP and MAC address of the client device and evaluates with previously registered client device. If current IP and MAC address are matched with previously registered client device, the communication is established.

The same process is followed at client side. Whenever a server sends response to the client, the client first obtains the IP and MAC address of the Web server and evaluates with previously registered IP and MAC addresses of the Web servers. If current IP and MAC address of the Web server matched with previously registered IP and MAC address of the Web server, the response is received by the client. Otherwise, the client will not accept response from the unknown server. By introducing the concept of IP-MAC trussing, the client and Web server both are able to secure against the internal threats.

The results obtained from the SOAPSonar simulator must be validated empirically to determine the empirical efficiency of the framework implemented [9]. A statistical validation has been performed to evaluate the performance of the framework. A chi-square test through the R language using the RStudio [10] tool is performed for statistical validation of the proposed framework. The following steps are executed to perform the chi-square test on data collected by the SOAPSonar simulator. The chi-square test has been performed on well-structured and messy database design. The chi-square test of independence is used to scrutinize the frequency table (i.e. contingency table), which is formed by two cataloging variables.

3.1 Chi-square test on Well-structured database design

First, chi-square testing has been performed on well-structured database design. R [11] is a programming language commonly used in statistical computing, data analytics, and scientific research. It is one of the most ubiquitous languages used by statisticians, data analysts, and researchers to scrutinize, envisage, and acquire input data. It has read.csv() method to read the csv (comma separated values) file in r language. It also has chisq.test() method to perform chi-square test on input data. Input data has been collected while implementing the threat detection framework on badhaliya gems digital trade semantic Web service. The following steps are followed to perform chi-square test on well-structured design.

Step 1: Define Hypothesis

The first stage of conducting chi-square test is to define the hypothesis. There are two hypothesis: one is null hypothesis and other is alternate hypothesis. Null hypothesis is believed to be true and alternate hypothesis to be untrue. One of these hypothesis is always be true.

Null Hypothesis:

H_O: Attacks are controlled by the proposed framework.

Alternative Hypothesis:

H_A: Attacks are not controlled by the proposed framework

Step 2: Input Sample Data in Rstudio

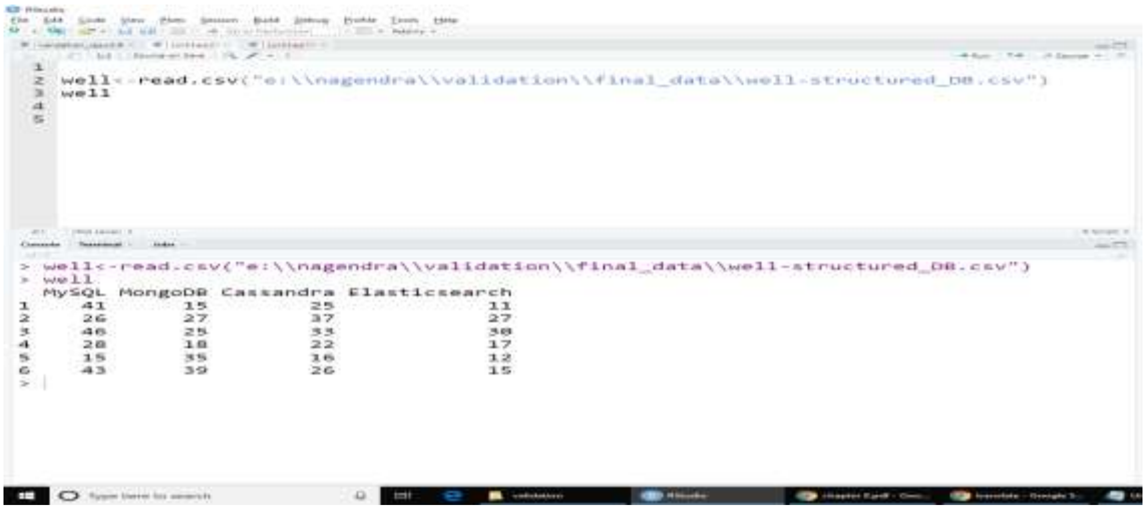
A test database from badhaliya gems has been used. The sample data has been collected using SOAPSonar simulator and put in a comma separated value (.csv) file. The read.csv() function takes .csv file and input and produces a dataframe from .csv file as output. This dataframe can be used for further processing in Rstudio and can be passed inside another function such as chisq.test().

Step 3: Chi-square Test using Rstudio

Chi-square test has been performed by using chisq.test() in-built function. The chisq.test() function takes .csv file as input and returns x-squared, df and p-value of input data. The chi-square test has been performed on both well-structured and messy data base design.

A) Reading well-structured_DB.csv file in Rstudio

The well-structured_DB.csv file is read in the Rstudio using read.csv() as shown in figure 3.1:



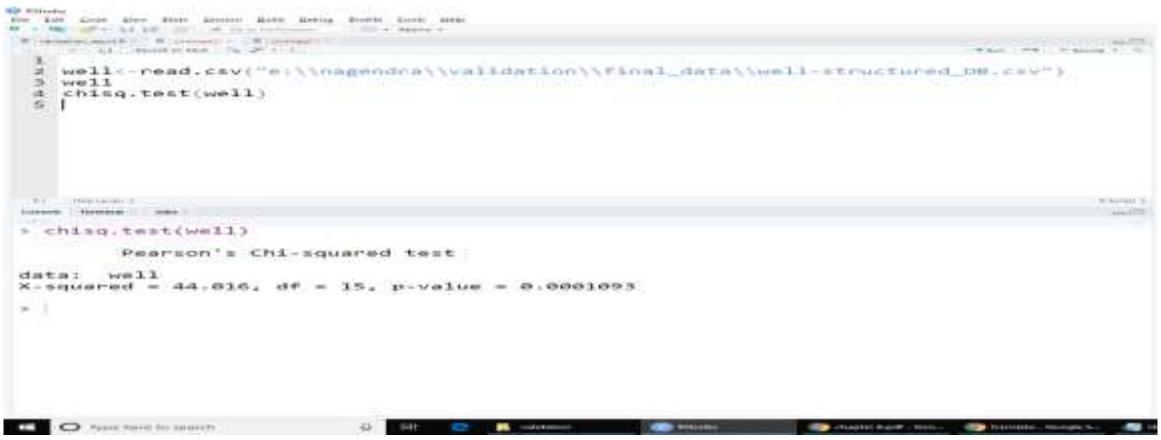
```
well<-read.csv("e:\\nagendra\\validation\\final_data\\well-structured_DB.csv")
well
```

```
> well<-read.csv("e:\\nagendra\\validation\\final_data\\well-structured_DB.csv")
> well
  MySQL MongoDB Cassandra Elasticsearch
1    41      15        25            11
2    26      27        37            27
3    46      25        33            38
4    28      18        22            17
5    15      35        16            12
6    43      39        26            15
> 
```

Figure 3.1: Reading well-structured_DB.csv file

B) Chi-square test on well-structured database design without attack.

Figure 3.2 shows the result after applying chi-square test on well-structured_DB.csv file using chisq.test(). The p-value is 0.0001093



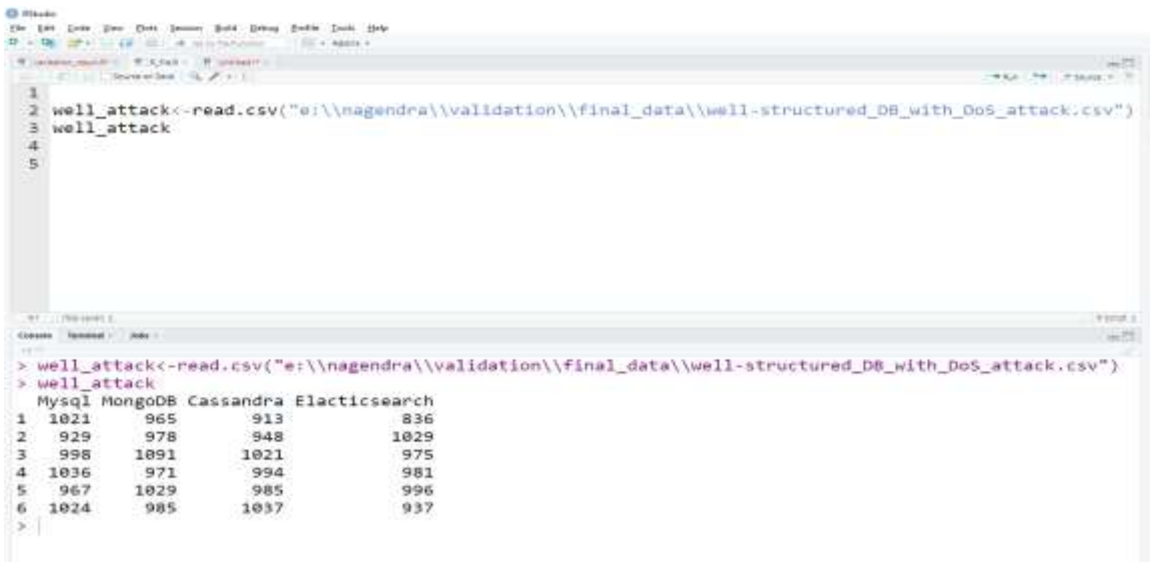
```
well<-read.csv("e:\\nagendra\\validation\\final_data\\well-structured_DB.csv")
well
chisq.test(well)
```

```
> chisq.test(well)

        Pearson's Chi-squared test

data:  well
X-squared = 44.816, df = 15, p-value = 0.0001093

> 
```

Figure 3.2: Chi-square test on well_structured_DB.csv

C) **Reading Well-structured database csv file after injecting DoS attack**

The well-structured database has been used after injecting DoS attack [14]. The well-structured_DB_with_DoS_attack.csv file is read in the Rstudio tool as shown in figure 3.3:



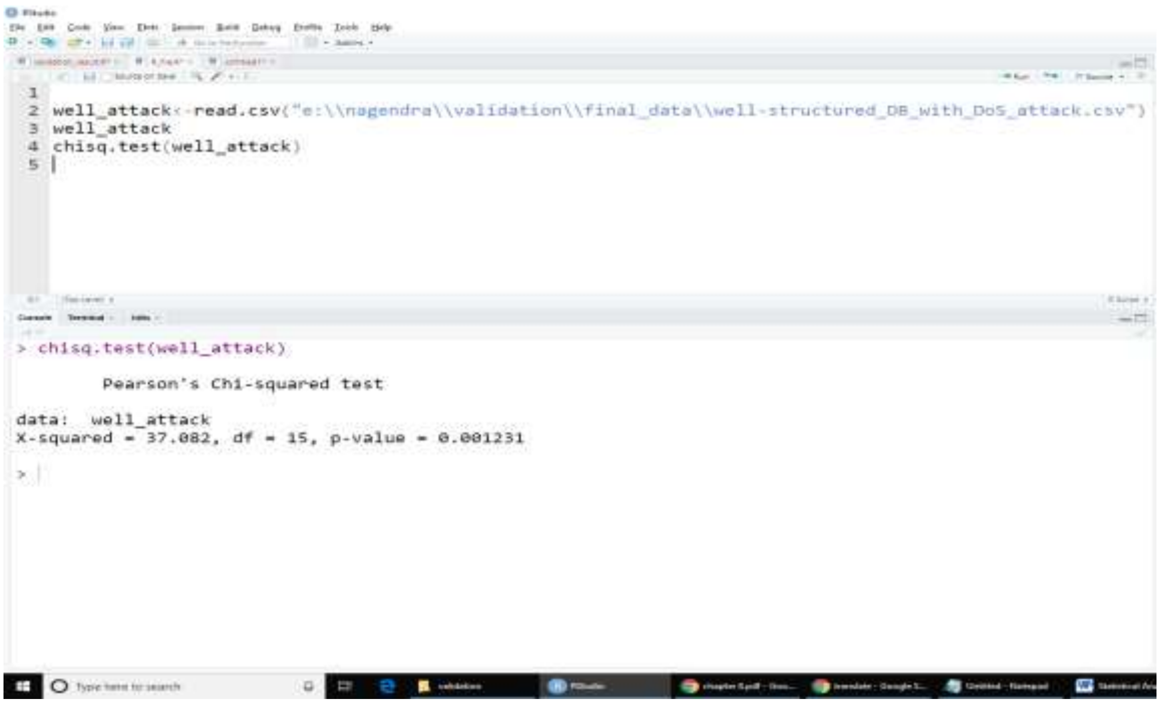
```
well_attack<-read.csv("e:\\nagendra\\validation\\final_data\\well-structured_DB_with_DoS_attack.csv")
well_attack
```

```
> well_attack<-read.csv("e:\\nagendra\\validation\\final_data\\well-structured_DB_with_DoS_attack.csv")
> well_attack
  Mysql MongoDB Cassandra Elacticsearch
1  1021     965       913           836
2   929     978       948          1029
3   998    1091      1021           975
4  1036     971       994           981
5   967    1029       985           996
6  1024     985      1037           937
> 
```

Figure 3.3: Reading well_structured_DB_with_attack.csv file in Rstudio

D) **Chi-square test on well-structured database design with DoS attack**.

Figure 3.4 shows the chi-square test using chisq.test() on well-structured database design. The p-value of chi-square test is 0.001231



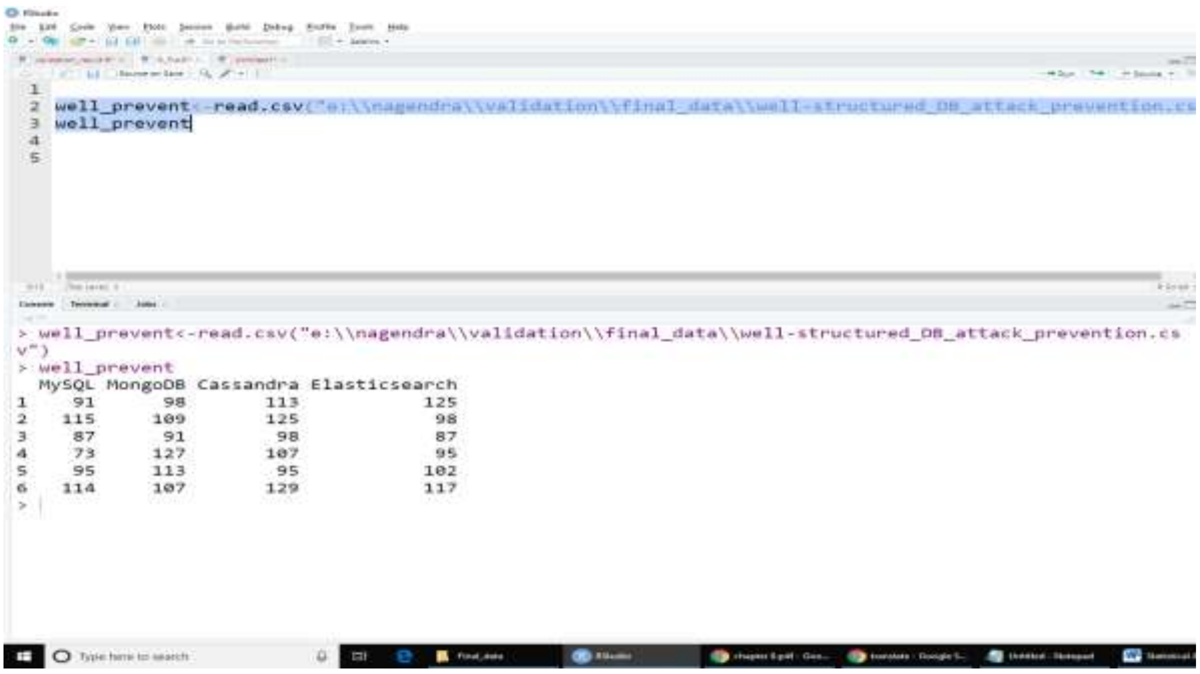
```
well_attack<-read.csv("e:\\nagendra\\validation\\final_data\\well-structured_DB_with_DoS_attack.csv")
well_attack
chisq.test(well_attack)
```

```
> chisq.test(well_attack)

        Pearson's Chi-squared test

data:  well_attack
X-squared = 37.082, df = 15, p-value = 0.001231

> 
```

Figure 3.4: Chi-square test on well-structured_DB_with_attack.csv

E) Reading well-structured database csv file after implementing the proposed algorithm

The well-structured database has been used after implementing the proposed algorithm [15]. The well-structured_DB_with_DoS_attack.csv file is read in the Rstudio tool as shown in figure 3.5:



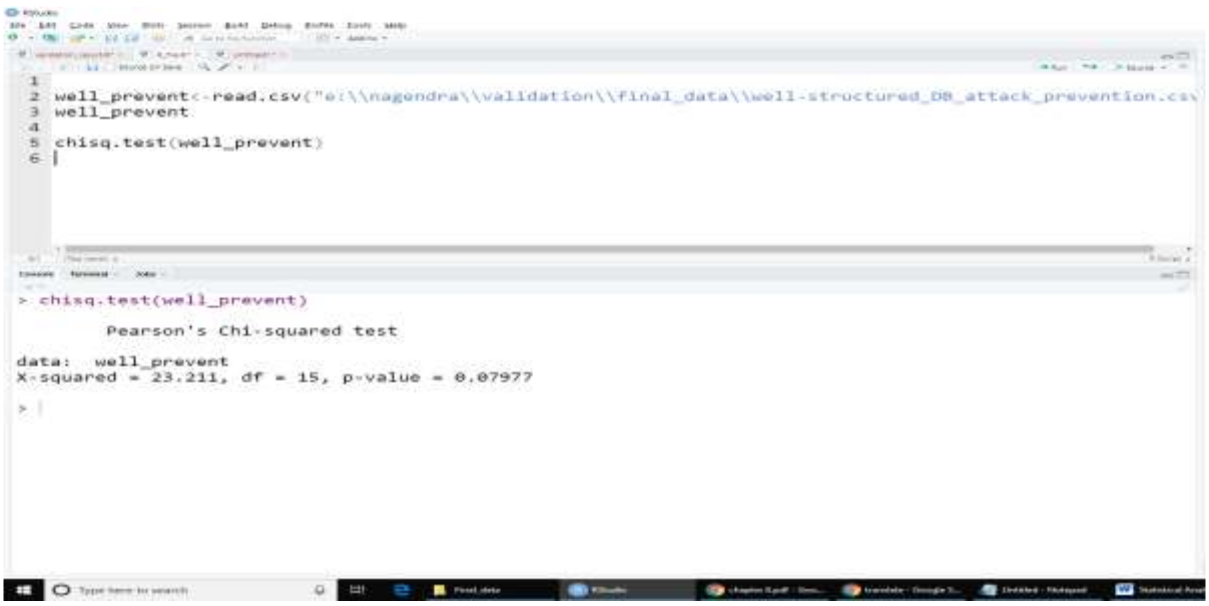
```
well_prevent<-read.csv("e:\\nagendra\\validation\\final_data\\well-structured_DB_attack_prevention.cs
well_prevent
```

```
> well_prevent<-read.csv("e:\\nagendra\\validation\\final_data\\well-structured_DB_attack_prevention.cs
v")
> well_prevent
  MySQL MongoDB Cassandra Elasticsearch
1    91      98       113           125
2   115     109       125            98
3    87      91        98            87
4    73     127       107            95
5    95     113        95           102
6   114     107       129           117
>
```

Figure 3.5: Reading well_structured_DB_attack_prevention.csv

F) Chi-square test after implementing proposed algorithm

Figure 3.6 shows the result of chi-square Test using Rstudio on well-structured database after implementing the proposed algorithm. The p-values is 0.07977



```
well_prevent<-read.csv("e:\\nagendra\\validation\\final_data\\well-structured_DB_attack_prevention.csv
well_prevent

chisq.test(well_prevent)
```

```
> chisq.test(well_prevent)

        Pearson's Chi-squared test

data:  well_prevent
X-squared = 23.211, df = 15, p-value = 0.07977

>
```

Figure 3.6: chi-square test on well_structured_DB_attack_prevention

3.1.1 Results for Well-structured database design

After successfully applying the chi-square test on well-structured database design, the consolidated results are shown in table 3.1. The table shows the p-value of each chi-square test performed on well-structured database design.

Table 3.1: Consolidated result of chi-square test on Well-structured Database Design

Database	p-value
Well-structured Database	0.0001093
Well-structured Database with DoS Attack	0.001638
Well-structured Database attack prevention	0.07977

A) Result for well-structured database without DoS attack

Significance value p = 0.0001093, That is α > p, (α = .05). The result illustrates the value of p < α then H0 is rejected and the alternative hypothesis H1 is accepted. Since, any attack is not present within the request; the framework is able to work well.

B) Result for well-structured database with DoS attack

Significance value p = 0.001638, That is α > p, (α = .05). The result illustrates the value of p < α then H0 is rejected and the alternative hypothesis H1 is accepted. Now, DoS attack is injected within the request. The DoS attack significantly reduces the performances of the proposed framework.

C) Result for well-structured database after preventing DoS attack

Significance value p = 0.07977, That is α > p, (α = .05). The result illustrates the value of p > α then H1 is rejected and the null hypothesis H0 is accepted. After applying the proposed algorithm, the framework is able to work well even when DoS attack is present within the request.

On the basis of the above results, it can be concluded that the proposed algorithm effectively controls the effect of DoS attack on well-structured database design.

3.2 Chi-square test on Well-structured database design

Step 1: Define Hypothesis

Null Hypothesis: H_O: Attacks are controlled by the proposed framework.

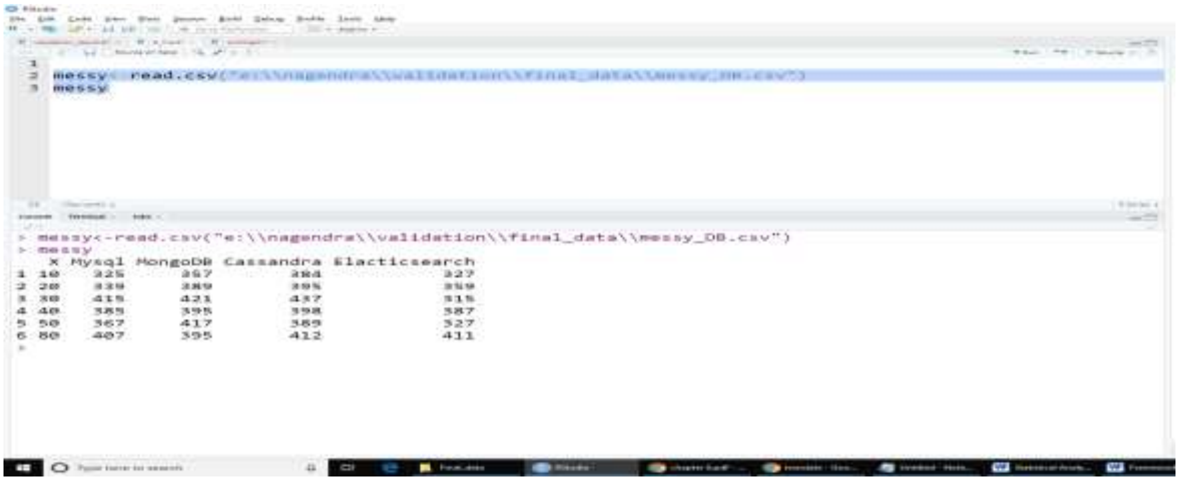
Alternative Hypothesis: H_A: Attacks are not controlled by the proposed framework

Step 2: Input Sample Data in Rstudio

A test database from badhaliya gems has been used. The sample data has been collected using SOAPSonar simulator and put in a comma separated value (.csv) file.

A) Reading messy database file without attack

The messy database has been used without attack. The messy_DB.csv file is read in the Rstudio using the read.csv() as shown in figure 3.7:



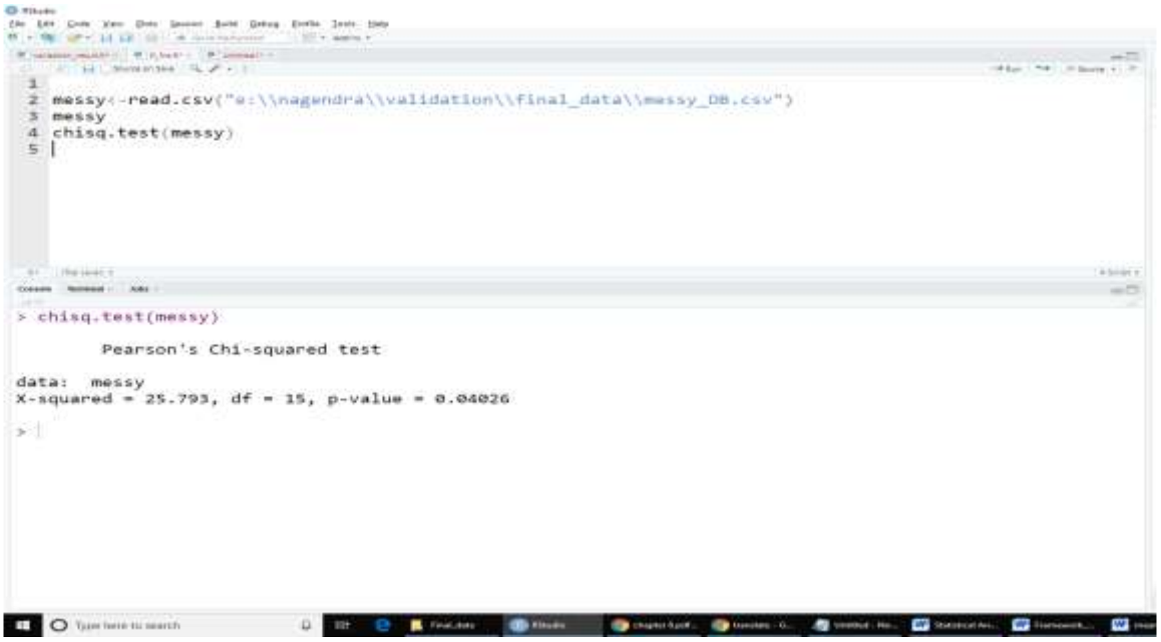
```
messy<-read.csv("e:\\nagendra\\validation\\final_data\\messy_DB.csv")
messy
```

```
> messy<-read.csv("e:\\nagendra\\validation\\final_data\\messy_DB.csv")
> messy
   X Mysql MongoDB Cassandra Elacticsearch
1 10   325     357       384           327
2 20   339     389       395           359
3 30   415     421       437           315
4 40   385     395       398           387
5 50   367     417       389           327
6 60   407     395       412           411
>
```

Figure 3.7: Reading messy_DB.csv file in Rstudioth

B) Chi-square Test using Rstudio on Messy database without DoS attack

Figure 3.8 shows the p-value after applying chi-square test on messy database without any attack. The p-value is 0.04026



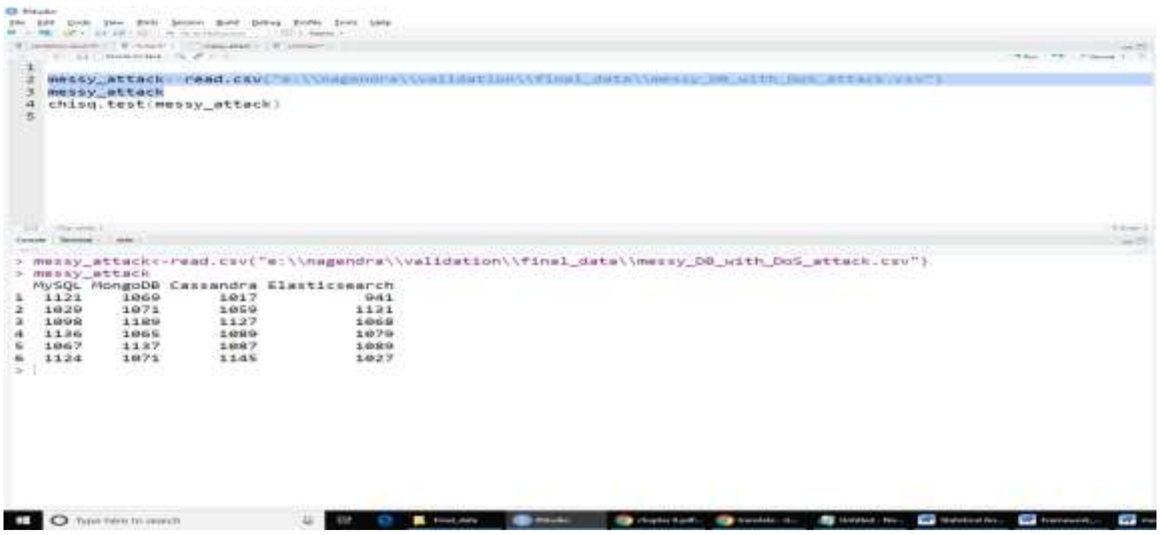
```
messy<-read.csv("e:\\nagendra\\validation\\final_data\\messy_DB.csv")
messy
chisq.test(messy)
```

```
> chisq.test(messy)

        Pearson's Chi-squared test

data:  messy
X-squared = 25.793, df = 15, p-value = 0.04026

>
```

Figure 3.8: Chi-square test on messy database without attack

C) Reading messy database csv file after injecting DoS attack

The messy database has been used with DoS attack. The messy_DB_attack_prevention.csv file is read in the Rstudio as shown in figure 3.9:



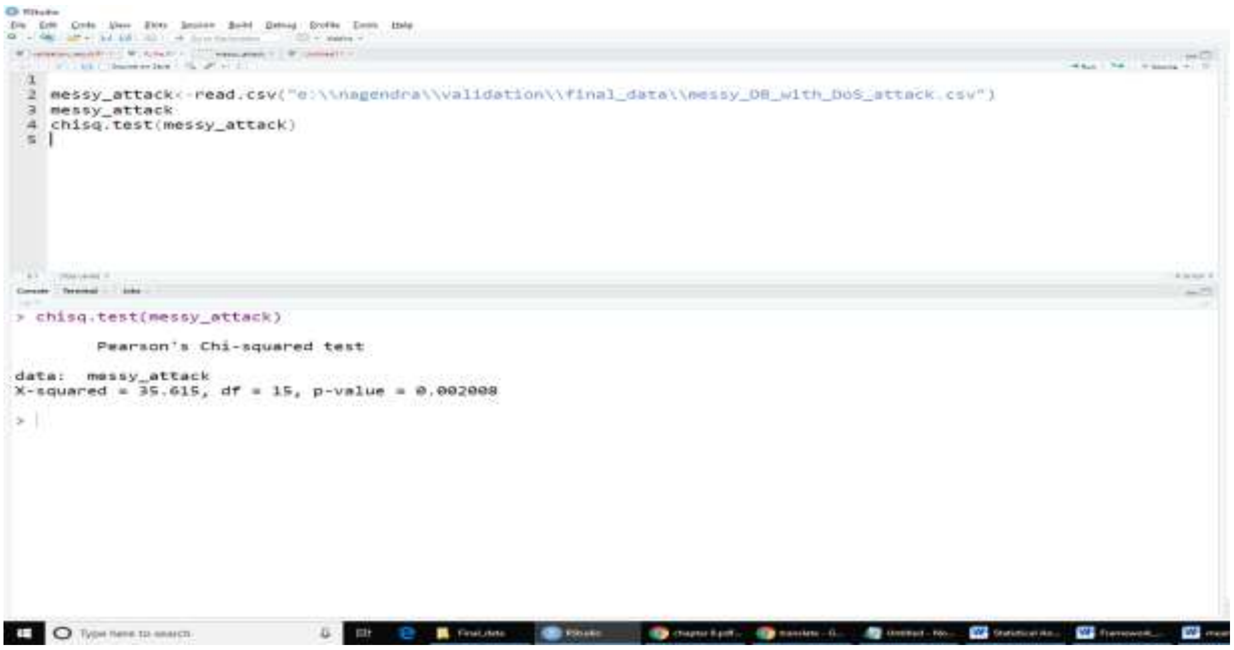
```
messy_attack<-read.csv("e:\\nagendra\\validation\\final_data\\messy_DB_with_DoS_attack.csv")
messy_attack
chisq.test(messy_attack)
```

```
> messy_attack<-read.csv("e:\\nagendra\\validation\\final_data\\messy_DB_with_DoS_attack.csv")
> messy_attack
  MySQL MongoDB Cassandra Elasticsearch
1  1121    1069      1017           941
2  1020    1071      1059          1131
3  1098    1109      1127          1068
4  1136    1065      1089          1079
5  1067    1137      1087          1089
6  1124    1071      1145          1027
> |
```

Figure 3.9: reading messy_DB_with_DoS_attack.csv

D) Chi-square test on messy database design with DoS attack

Figure 3.10 shows the Chi-square Test using Rstudio on Messy database with DoS attack. The p-value after applying the chi-square test is 0.002008.



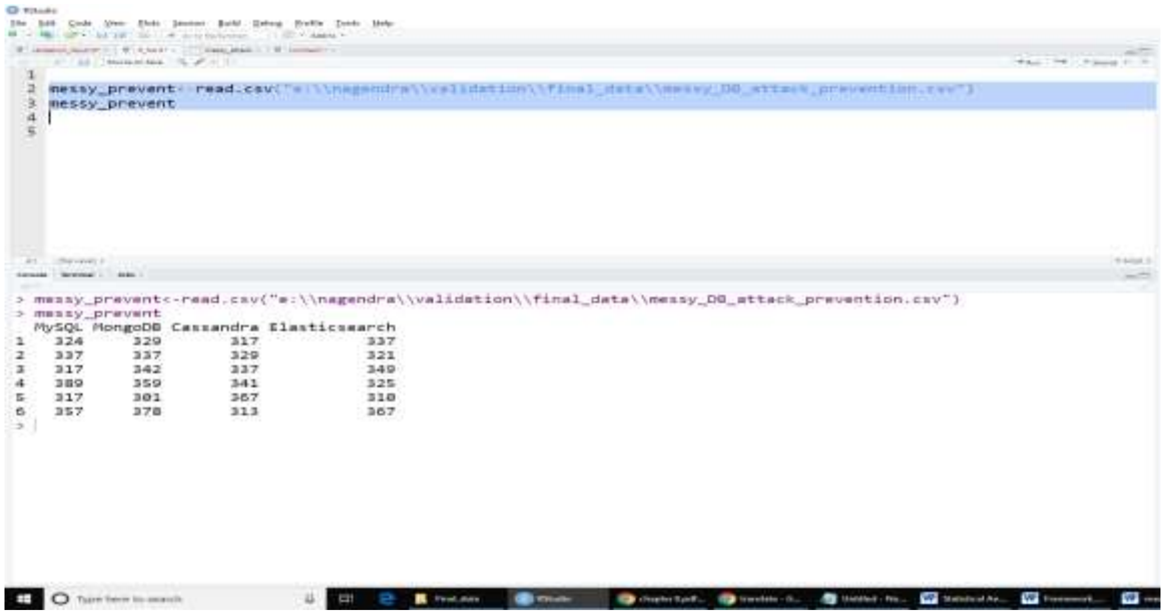
```
messy_attack<-read.csv("e:\\nagendra\\validation\\final_data\\messy_DB_with_DoS_attack.csv")
messy_attack
chisq.test(messy_attack)
```

```
> chisq.test(messy_attack)

        Pearson's Chi-squared test

data:  messy_attack
X-squared = 35.615, df = 15, p-value = 0.002008

> |
```

Figure 3.10: chi-square test on messy_DB_with_DoS_attack.csv

E) Reading messy database csv file after implementing proposed algorithm

The messy database has been used after implementing the proposed algorithms [15]. The messy_DB_attack_prevention.csv file is read in the Rstudio tool as shown in figure 3.11:



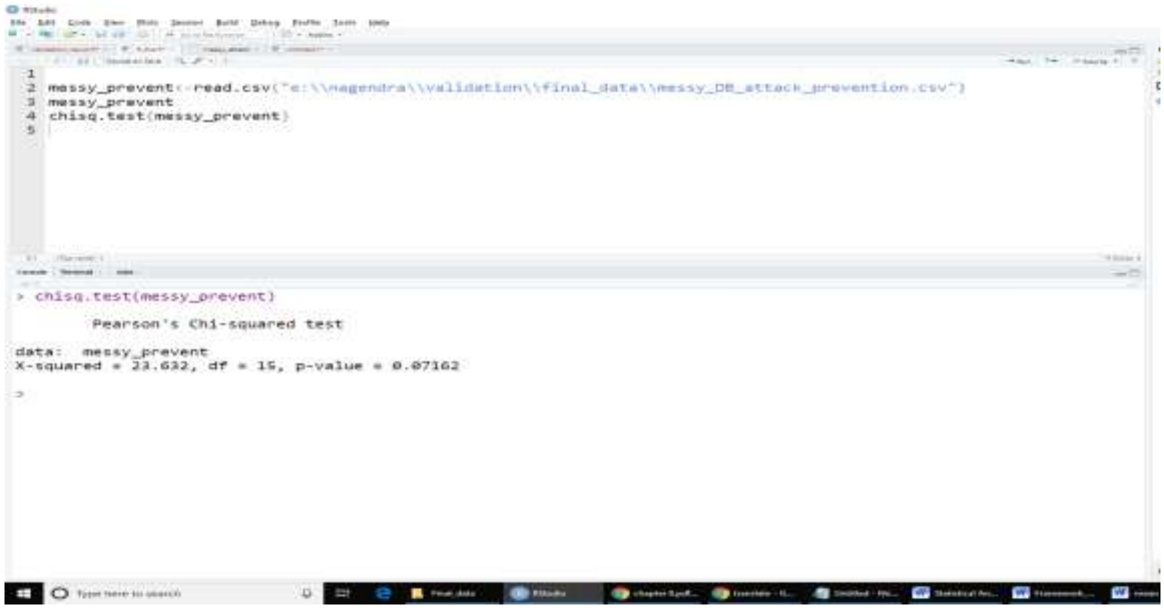
```
messy_prevent<-read.csv("e:\\nagendra\\validation\\final_data\\messy_DB_attack_prevention.csv")
messy_prevent
```

```
> messy_prevent<-read.csv("e:\\nagendra\\validation\\final_data\\messy_DB_attack_prevention.csv")
> messy_prevent
  MySQL MongoDB Cassandra Elasticsearch
1   324     329       317           337
2   337     337       329           321
3   317     342       337           349
4   389     359       341           325
5   317     301       367           318
6   357     378       313           367
>
```

Figure 3.11: reading messy_DB_attack_prevention.csv in Rstudio

F) Chi-square test on messy database after implementing the proposed Algorithm

Figure 3.12 shows the Chi-square Test using Rstudio on Messy database after applying proposed algorithm. The p-values is 0.07162.



```
messy_prevent<-read.csv("e:\\nagendra\\validation\\final_data\\messy_DB_attack_prevention.csv")
messy_prevent
chisq.test(messy_prevent)
```

```
> chisq.test(messy_prevent)

        Pearson's Chi-squared test

data:  messy_prevent
X-squared = 23.632, df = 15, p-value = 0.07162

>
```

Figure 3.12: Chi-square test on messy database after implementing proposed algorithm

3.2.1 Result

The results have been obtained from chi-square test for messy database design are consolidated and presented in table 3.2. The consolidated results are compared against different p-values obtained after applying the chi-square test on messy database design. The result clearly indicates that the p-value for each of the messy database is different to each other.

Table 3.2: Consolidated result of chi-square test on Messy Database Design

Database	p-value
Messy Database	0.04026
Messy Database with Attack	0.002008
Messy Database attack prevention	0.07162

A) For Messy database without DoS attack

Significance value p = 0.04026, That is α > p, (α = .05). The result illustrates the value of p < α then H0 is rejected and the alternative hypothesis H1 is accepted.

B) For messy database with DoS attack

Significance value p = 0.002008, That is α > p, (α = .05). The result illustrates the value of p < α then H0 is rejected and the alternative hypothesis H1 is accepted.

C) For messy database after preventing DoS attack

Significance value p = 0.07162, That is α > p, (α = .05). The result illustrates the value of p > α then H1 is rejected and the null hypothesis H0 is accepted. So it can be concluded that the proposed algorithm effectively controls the effect of DoS attack on messy database design.

IV. Conclusion

In this paper, a statistical analysis of the framework to prevent attacks on digital-trade semantic Web amenities based on XACML is carried out. Statistical analysis has been performed for both well-structured and messy database design for digital-trade semantic Web amenities. The chi-square test has been used to perform statistical analysis of the proposed framework. Chi-square test is performed on well-structured databases without attack and with DoS attack. Then, after applying the proposed algorithm, chi-square test is performed on a well-structured database. The proposed algorithm is able to protect the digital-trade semantic Web amenities from internal threats such as DoS, after looking at the results of the Chi-square test for well-structured database design.

Chi-square test is also performed on messy databases such as without attack and with DoS attack. Then, after applying the proposed algorithm, chi-square test is performed on a messy database. The p-value of each type of messy database such as without attack, with DoS attack and after implementing the proposed algorithm, clearly indicates that the proposed algorithm protects digital-trade semantic web amenities against internal threats such as DoS threat.

Chi-square results on the Threat Detection Framework indicate that the framework is capable of further strengthening security in digital-trade semantic web amenities. The results also indicate that well-structured database design leads to unprecedented increase in the functionality of digital trade semantic web services compared to messy database design. It can be concluded that the presented framework is capable of playing an important role in protecting digital-trade semantic Web amenities from inside threats.

The goal of the future will be to further improve the performance of the framework. Efforts will be made to implement some such amendments in future so that the functionality of the proposed framework can be broadening rallied.

References

1. Martínez-González, M.M. and Alvite-Díez, M.L., 2019. Thesauri and Semantic Web: Discussion of the evolution of thesauri toward their integration with the Semantic Web. IEEE Access, 7, pp.153151-153170.
2. Antoniazzi, F. and Viola, F., 2019. Building the Semantic Web of Things Through a Dynamic Ontology. IEEE Internet of Things Journal, 6(6), pp.10560-10579.
3. Singh, N.K. and Nayak, S.K., 2019. A Framework for Integrated Threat Detection on Semantic Web Services. International Journal of International Journal of Control and Automation, Vol. 12, No. 4, pp. 157-169.
4. Sharma, R. and Kumar, M., 2014, July. Validation of the data warehouse metrics using formal frameworks. In 2014 International Conference on Signal Propagation and Computer Technology (ICSPCT 2014) (pp. 239-243). IEEE.
5. Tao, Y., Ding, L. and Ganz, A., 2017. Indoor navigation validation framework for visually impaired users. IEEE Access, 5, pp.21763-21773.
6. Guapo, F., Correia, P., Meuwly, D. and van der Vloed, D., 2016, March. Empirical validation of likelihood ratio methods–A case study in forensic speaker recognition. In 2016 4th International Conference on Biometrics and Forensics (IWBF) (pp. 1-5). IEEE.
7. Tantithamthavorn, C., McIntosh, S., Hassan, A.E. and Matsumoto, K., 2016. An empirical comparison of model validation techniques for defect prediction models. IEEE Transactions on Software Engineering, 43(1), pp.1-18.
8. Cunha, J., Fernandes, J.P., Mendes, J. and Saraiva, J., 2014. Embedding, evolution, and validation of model-driven spreadsheets. IEEE Transactions on software Engineering, 41(3), pp.241-263.
9. Badhalia Gems [Online], Available: http://jaipur.dkinfosolutions.com/badhaliagems/
10. Singh, N.K. and Nayak, S.K., 2019. The Threat Detection Framework for Securing Semantic Web Services. Journal of Computational and Theoretical Nanoscience, 16(12), pp.5099-5104.
11. RStudio, https://www.rstudio.com/about
12. R Language, https://www.r-project.org/about.html
13. Chisq.test(), https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/chisq.test
14. Singh, N.K. and Nayak, S.K., 2019. Semantic Web Services: An Empirical Methodology to Security. International Journal of Engineering and Advanced Technology (IJEAT), 8(6), pp. (3100-3104)
15. Singh, N.K. and Nayak, S.K., 2019. A Pragmatic Algorithm for Attack Prevention on Semantic Web Services. International Journal of Engineering and Advanced Technology (IJEAT), 9(1), pp. (1945-1950).