Robust Analysis of XXE Attack Produced by Malware

Surbhi Prakash[1], Prof. AK Mohapatra[2]

[1] PhD Scholar, Indira Gandhi Delhi Technical University for Women, New Delhi, India.

[1] Professor, Indira Gandhi Delhi Technical University for Women, New Delhi, India.

Abstract: Malware Analysis is one of the major growing sections in the cyber security area. Various trends and types have been introduced in the industry for example static malware analysis, Dynamic malware analysis, hybrid malware analysis and machine learning-based malware analysis techniques. There is various malware introduced for example virus, Worms, ransomware, spyware, botnets, etc.

Security threats have increased drastically over the period. From viruses, spyware, worms, trojans, and ransomware to many zero-day Malware is reported and exploited in different platforms. Platforms like Windows, Android, and Cloud (Iaas or Paas). The Phenomenon is like attackers always making targets to humans via social engineering methodology or Phishing. When we talk about humans, the first thing that comes to mind of an attacker is the platform from which they will be able to concentrate on the target. The basic approach used mainly in detecting Malware in any platform is signature-based detection, which is quite beneficial. Still, as Malware is designed to be more obfuscated, detecting those malicious activities using a signature-based approach takes a lot of work. After the signature-based method, the behavior-based process is used to detect Malware. As some drawbacks appeared in both approaches, then, researchers found methodologies that can use Machine Learning Algorithms, for example, KNN, Random Forest, Nearest Neighbor, etc.

Keywords: Malware, vulnerability, Algorithm, XML, Detection.

1. INTRODUCTION

The purpose of malicious content detection is usually to provide the information we need to respond to a network intrusion. Our goals will typically be to determine exactly what happened, and to ensure that we've located all infected machines and files. When analyzing suspected malware, our goal will typically be to determine exactly what a particular suspect binary can do, how to detect it on our network, and how to measure and contain its damage. Once we identify which files require full analysis, it's time to develop signatures to detect malware infections on our network. As we'll learn throughout this project, malware analysis can be used to develop host-based and network signatures. This review includes different methods for feature extraction static analysis, Dynamic analysis, and hybrid analysis **[1].**

In this paper mainly we concentrate on the XML-based attack produced in any application where we have any upload functionality, so we designed one framework using machine learning language and Python to detect XXE vulnerability in we greater percentage. As in Owasp 2017 standard XXE is reported as a vulnerability at the fourth number and in updated version 2021 it is considered under the Injection section.

1.1 Type of Malware Analysis Techniques

On the basis of the literature review we have come up we have analysis of how many types of malware are there and the types of tools used for analysis.

Malware analysis can be defined as malicious file analysis techniques including mostly executable files which are not human-readable **[2].**

So, we need some different techniques like static analysis and dynamic analysis.

Basic Static Analysis

Basic malware analysis includes observing files whether the file is malicious or not. Files with their signature identification can be included in this. Sometimes it provides sufficient information but mostly it does not provide fruitful information **[1].**

Basic Dynamic Analysis

Basic malware analysis involves running the executables and observing the behavior it provides. quite more efficient results as compared to basic static analysis. Sometimes some people analyze. The program without having programming knowledge.

Advanced Static Analysis

Advanced static analysis consists of reverse- engineering, knowledge of disassembly, code constructs, and Windows operating system concepts. In Advanced static analysis, the code is executed in CPU and provide a better result **[3]**.

Advanced Dynamic Analysis

Advanced dynamic analysis consists of a debugger which is used to get the internal information of any malicious program. It is the best technique that is used to extract meaningful feature sets to analyze them **[3].**

Types of Malware

There are different families of malware that come under this category. Different features of different families provide new characteristics of each sample and risk rate. Some of the malware types are:

Backdoor: Malicious code which installs itself on the computer without the permission of an authenticated user. They steal the information without permission **[6].**

Botnet: Like a backdoor, it also gets unauthorized access to the computer but all the systems infected by the same botnet receive the same infection from the same command **[6]**

Downloader: Downloaders can be defined as malicious code which used to download more malicious software. They spread in the system. rigorously and affect the system [**6**].

Information-stealing malware: Malware included in this category is malicious code that is used to steal information such as passwords, online banking, and email information [**3][1].**

Rootkit: Malicious code designed to conceal the existence of other code. Rootkits are paired up with different malware codes and give remote access to computers to infect the computer **[3]**.

Spam-sending malware: Malware that infects an authenticated user's system and then uses that system only to send spam to earn some income from unfair means **[5].**

Worms and viruses: These malicious codes infect the system by replicating the files within the different files like exe, and pdf. Worms and viruses affect the boot sector and many important software of the system [**5][6].**

1.2 SPECIFICATION OF DATASET

Dataset is an important parameter to analyze the different features of any file. This leads to the definition of more than thirty thousand features, which is a large feature set that covers a wide range of sample characteristics. The collection of datasets is from Virus Total private API **[1].** Another analysis was on opcodes by using different machine learning algorithms **[4].** DLL files for sandbox analysis is done in some analysis process [**6].** Total Samples were collected from **Virusign, Malware Bazar, and InQuestLabs.**

2. DESCRIPTION OF XML ISSUE

In this paper, we have concentrated on the detection of XML-based vulnerabilities basically as XXE and Billion laugh attacks designed one Framework that can detect XXE in several applications and compared the previously designed parser with the newly designed Framework.

XML External Entity attack is one of the major vulnerabilities in those applications that parse the XML inputs, this vulnerability occurs when an XML entity is being passed through the weakly configured Parser.

This attack may lead to the disclosure of confidential data, a denial-of-service attack which is also known as a Billion Laugh attack, or server-side request forgery.

XML 1.0 standard defines the structure of XML document. There is one concept exit which is used for storage and known as entity. There are different types of entity exits for example external and Parameter often Known as External entity.

How Malware Can Lead to XML External Entity Attack

As malware is malicious software it will be injected in the DTD (external DTD), Document type definition, and when the parser is not configured properly, its vulnerable XML will be uploaded and parsed by the parser and uploaded to the website and exploit the application which result in Local File inclusion, Remote code execution or DOS.

Table 1. Type of XML attacks

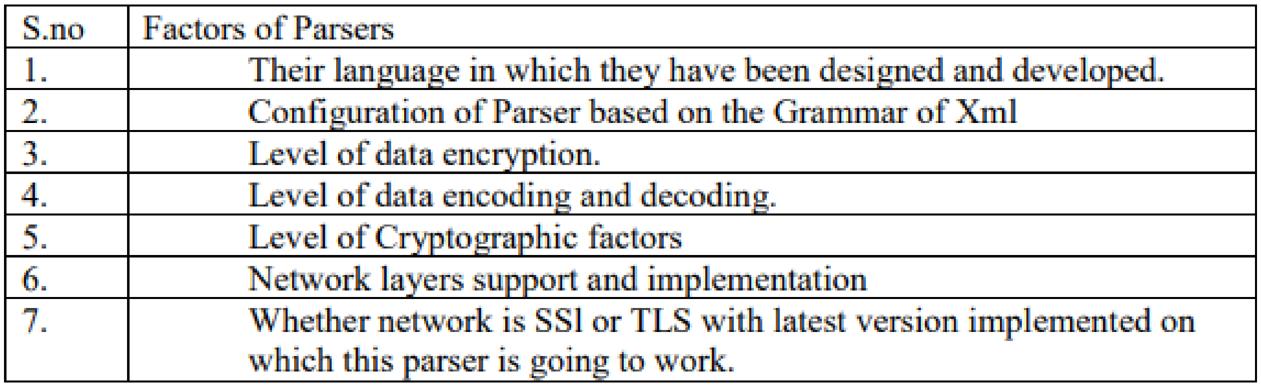
XML Injection
XML XSS
XML Billion Laugh
Server-Side Request Forgery
XML External Entity
XML Schema Poisoning
XML Path
Blind XXE
XML Tautology
XML Rpc File Access

2.1Testing for configuration of Parsers based on Modern Language

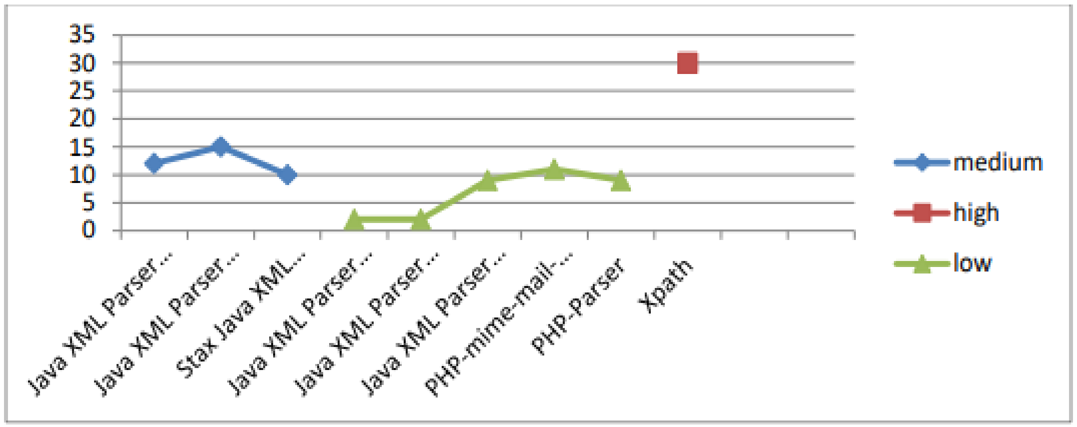
XML Parser is being designed for Parsing. Parsing is a process of refining components from a document to be uploaded on the server for the client services. Important data should be parsed with the proper security implementation. There should be a firewall in the network so that it will allow for parsing specific data of users. In this research work, we have collected 12 modern parsers which are mostly used in the industry for the purpose of parsing data which have a high rate of confidentiality.

Testing of Parsers has been done on the bases:

Table 2. Factors of Parsers



S.no	Factors of Parsers
1.	Their language in which they have been designed and developed.
2.	Configuration of Parser based on the Grammar of Xml
3.	Level of data encryption.
4.	Level of data encoding and decoding.
5.	Level of Cryptographic factors
6.	Network layers support and implementation
7.	Whether network is SSl or TLS with latest version implemented on which this parser is going to work.



Graph 1. Result for Severity Levels of Different Parser

2.2 Methodology for static testing

In Static testing Methodology we include only steps for manual testing and analyze those results. Execution of Payload and command resides under Advanced static testing or dynamic testing.

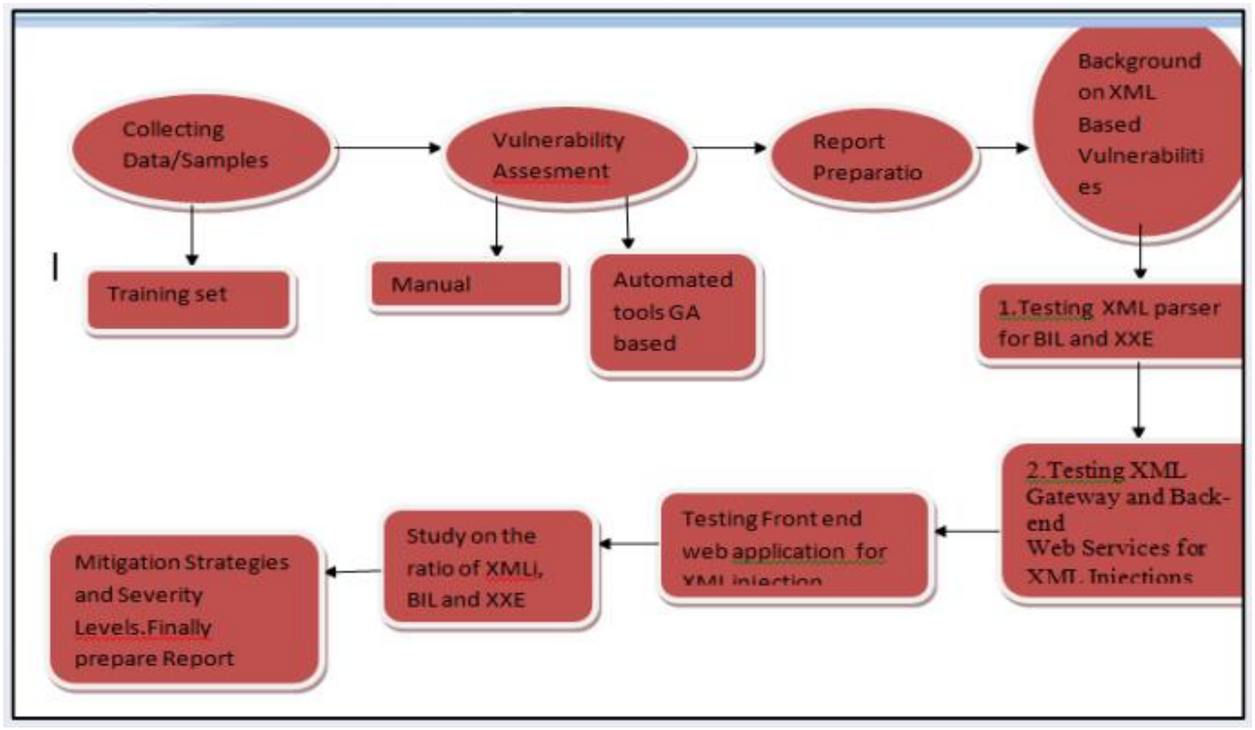

Collecting Data/Samples
Vulnerability Assesment
Report Preparatio
Background on XML Based Vulnerabiliti es
Training set
Manual
Automated tools GA based
1.Testing XML parser for BIL and XXE
2.Testing XML Gateway and Back-end Web Services for XML Injections
Testing Front end web application for XML injection
Study on the ratio of XMLi, BIL and XXE
Mitigation Strategies and Severity Levels.Finally prepare Report

Fig 1. Flow Diagram for analysis

Creation of Framework on mutation-based test generation for XML Injection. A FRAMEWORK FOR XMLi TESTING We introduce a framework on Mutation-based test generation for XML Injection), a framework for testing web services against XMLi attacks. This is equipped with a set of mutation operators that can manipulate XML in order to generate all four types of XMLi attacks. For Type 3 (Replicating) and Type 4 (Replacing) in which XMLi attacks carry nested attacks in the form of XML content, Technique relies on a constraint solver and attack grammars to generate the nested attacks (also called as malicious content), making them more effective in circumventing the validation mechanisms of web services. First, need to work on mutation operators for generating each type of XMLi attack. Then, we describe in detail how malicious content (nested attacks) are generated for XMLi of types 3 and 4. Finally, we define the general test generation strategy implemented in this technique for the detection of successful XMLi attacks.

1. Need to study Mutation Operators which will be used for attack generation. 2. Producing Nested Attacks 3. Domain Constraints 4. Attack Grammer 5. using Constraint Solving Produce malicious constraints 6. Mutations based test generation 7. Study on Json.

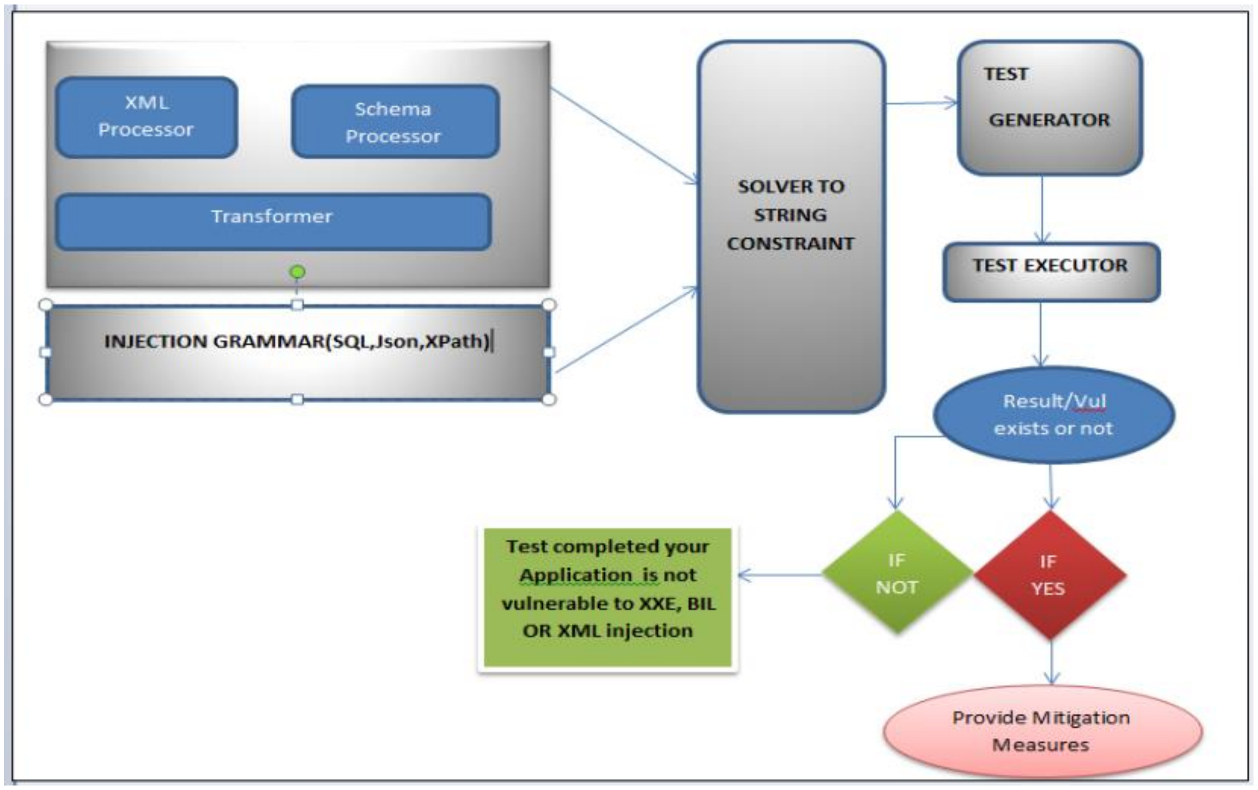

XML Processor
Schema Processor
Transformer
INJECTION GRAMMAR(SQL,Json,XPath)
SOLVER TO STRING CONSTRAINT
TEST GENERATOR
TEST EXECUTOR
Result/Vul exists or not
IF NOT
IF YES
Test completed your Application is not vulnerable to XXE, BIL OR XML injection
Provide Mitigation Measures

Fig 2. Framework Designed for Final Results

The above diagram depicts the working of the Framework in the final phase here we have code for Parser as well, where we have designed parsers with the help of a modern parser that can able to parse malware as well.

```
<?xml version=”1.0”  ?>
<address>
    <friend />
    <name>John <nickname>Spike</nickname> Smith</name>
    <streetAddress>123 Maple Ave. </streetAddress>
</address>
my  $text;
             my  $tag
            m{^(.*)< (.*?)>$}s;
            $text = $1;
            $tag  = $2;
(
   [  “address” , “” ],
   [   “name” , “” , “John” ]
```

```
)
Our   @tree = (“xml”,  “”);              #Parsed XML Tree
Our   $context =   \@tree;                #Context stack
Our    @context = ();                   #Current context
$/  =   “>”;
While  (<>)
{
                Last  if   $_ eq  “”;        #End of File
                # There might be text in front of the XML element.
                 my  $text;
                 my  $tag;
                 m{^(.*)<(. *?)>$}s;
                 $text  =  $1
                 $tag   =  $2
                  If    ($text   ne  “”)
                 {
                           Push  @{$context},  $text;
                  }
}
If     ($tag =~ m{
                   ^(/?)                # closing  tag
                    (\S+)              # tag name
                    #  Optional attributes
                    (
                      #  <space>  <attr-name> = “<attr-value>”
                        (?;\s+
                             [\w:]+    #Attribute name
                             =
                             \”[^\”]”\”    # Attribute value
```

Fig 3. Creating XML Parser

3. OUTPUTS OF FRAMEWORK

It has been implemented in two languages Python and PHP. The database used is MySQL and the server used is Xampp.

Dashboard functionality itself defines the work done by the framework for testing and providing outputs. For the execution of this work, we have taken 20 link URLs as our dataset is different from Parsers. We have done testing of 10 modern parsers and obtained output of their vulnerabilities in the above figure and graph in the section of testing results for Parsers. Now we have taken website data for testing and finding the vulnerabilities based on XML in those web applications. They provided us with response codes as per their functionality and response length to verify what amount of data was present in the web application during testing. As in the proof of work figures below, we can observe that in the first step executable tool asks for the link to the target website, After entering the target URL it will provide the output of the header or we can say the information of Banner Grabbing. After getting information of Banner grabbing then again it will ask for a target URL with a parameter then it will define the existence of the vulnerability in the web Dashboard of Framework Uploading Files Sending files for analysis Test cases execution Details of interface Summary Results: 1. Vulnerability 2. Response code 3. Response Length application after getting a vulnerability true positive result it will show the response code of the website and the current status of the website

Different algorithms used for malware analysis: Nave baye, Decision tree, Knn, Boosted J48, SVM, Random Forest, Gradient Boosting Decision Tree, Artificial Neural Network. In this work we have implemented framework based on Genetic Algorithm.

We use the following standard metrics for measuring the detection effectiveness. Let TP (True Positive) be the number of malicious files that were correctly classified as malicious, FP (False Positive) be the number of benign files that were misclassified as malicious, FN (False Negative) be the number of malicious files that were incorrectly classified as benign, and TN (True Negative) be the number of correctly classified benign files. Detection accuracy is defined as [1][6]:

Accuracy = $\underline{TP+TN}$

TP+TN +FP+FN

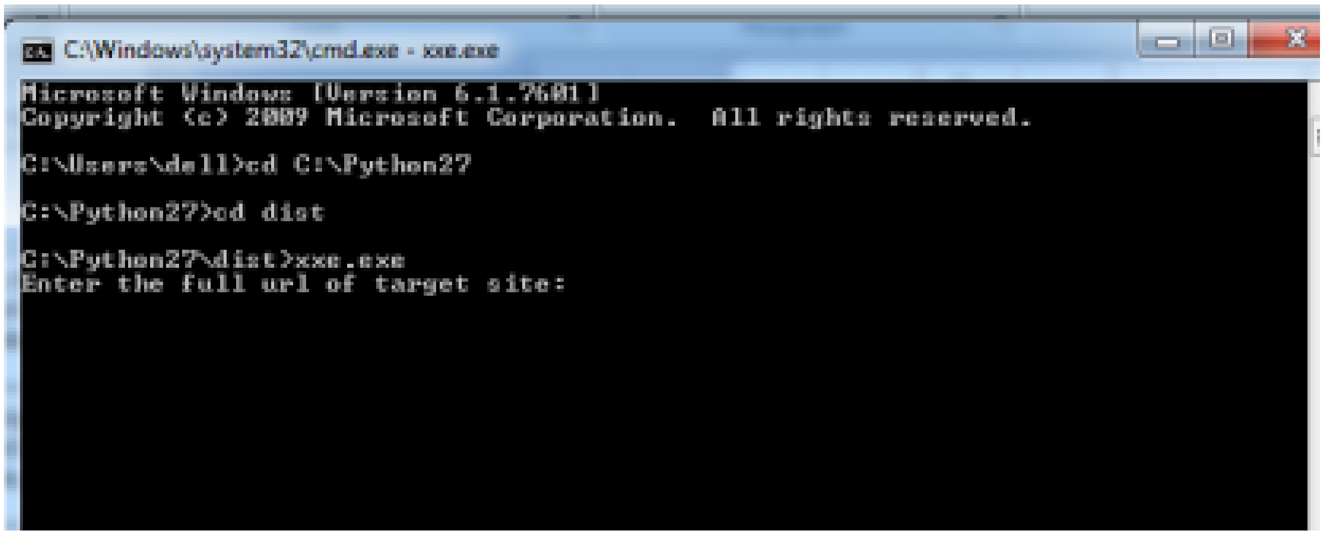
They presented basically a malware family classification approach with limited complexity in feature design and mechanism employed. They also used some machine learning algorithms one as SVM, n-gram, and XGBoost. Evaluation measure is performed by two methods accuracy and logarithmic [2][7].

n m

Logloss = $\underline{-1\sum}$ $\sum$ yij log(Pij) N i=1 j=1

3.1 Proof of Work

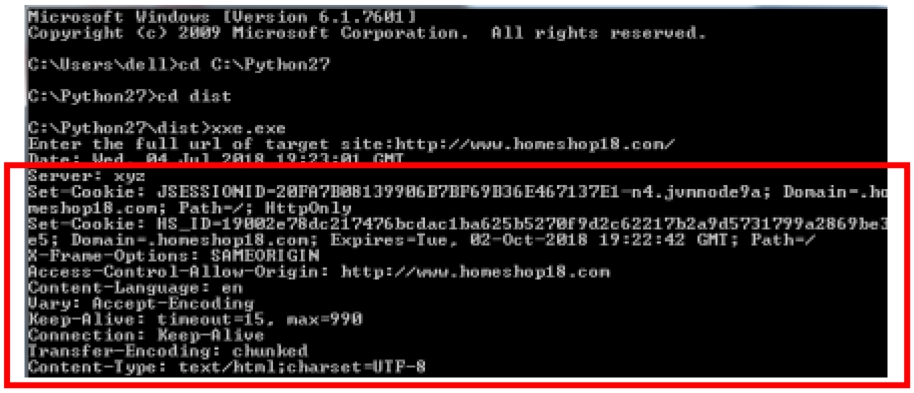
Observed the header information in the output



C:\Windows\system32\cmd.exe - xxe.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\dell>cd C:\Python27

C:\Python27>cd dist

C:\Python27\dist>xxe.exe
Enter the full url of target site:
```

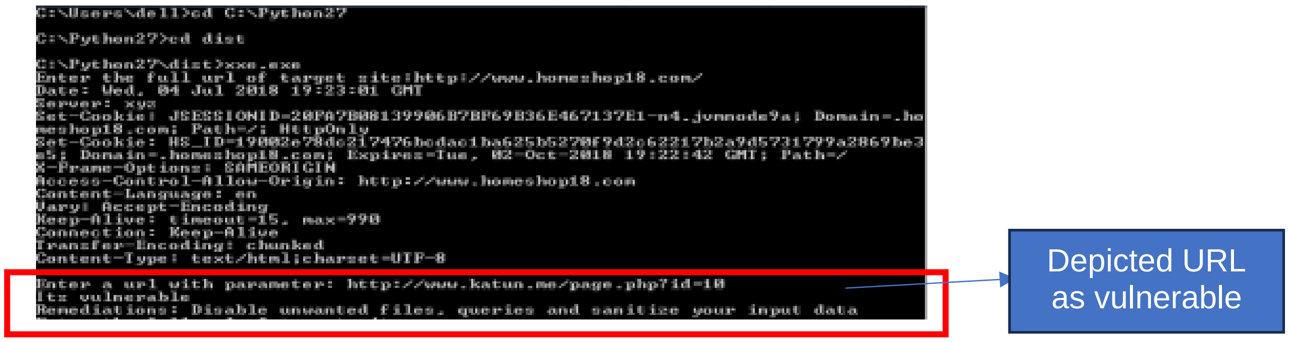
Figure 4. Output of executable file



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\dell>cd C:\Python27

C:\Python27>cd dist

C:\Python27\dist>xxe.exe
Enter the full url of target site:http://www.homeshop18.com/
Date: Wed, 04 Jul 2018 19:23:01 GMT
Server: xyz
Set-Cookie: JSESSIONID=20FA7B08139906B7BF69B36E467137E1-n4.jvmnode9a; Domain=.ho
meshop18.com; Path=/; HttpOnly
Set-Cookie: HS_ID=19002e78dc217476bcdac1ba625b5270f9d2c62217b2a9d5731799a2869be3
e5; Domain=.homeshop18.com; Expires=Tue, 02-Oct-2018 19:22:42 GMT; Path=/
X-Frame-Options: SAMEORIGIN
Access-Control-Allow-Origin: http://www.homeshop18.com
Content-Language: en
Vary: Accept-Encoding
Keep-Alive: timeout=15, max=990
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html;charset=UTF-8
```

Figure 5. Output of Banner Grabbing

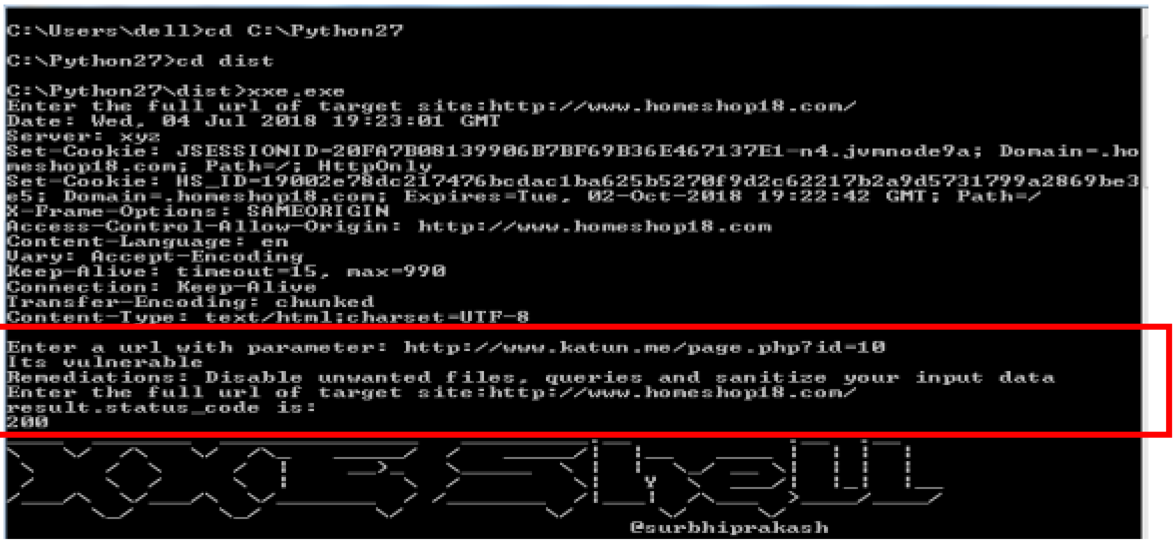
(ii) Enter the target URL with the parameter id and observe whether the URL is vulnerable or not finally by checking all the services and data of XML.



```
C:\Users\dell>cd C:\Python27

C:\Python27>cd dist

C:\Python27\dist>xxe.exe
Enter the full url of target site:http://www.homeshop18.com/
Date: Wed, 04 Jul 2018 19:23:01 GMT
Server: xyz
Set-Cookie: JSESSIONID=20FA7B08139906B7BF69B36E467137E1-n4.jvmnode9a; Domain=.ho
meshop18.com; Path=/; HttpOnly
Set-Cookie: HS_ID=19002e78dc217476bcdac1ba625b5270f9d2c62217b2a9d5731799a2869be3
e5; Domain=.homeshop18.com; Expires=Tue, 02-Oct-2018 19:22:42 GMT; Path=/
X-Frame-Options: SAMEORIGIN
Access-Control-Allow-Origin: http://www.homeshop18.com
Content-Language: en
Vary: Accept-Encoding
Keep-Alive: timeout=15, max=990
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html;charset=UTF-8
Enter a url with parameter: http://www.katun.me/page.php?id=10
Its vulnerable
Remediations: Disable unwanted files, queries and sanitize your input data
```

Figure 6. XXE Detection

(iii) Now check the response code. We can observe the 200 response code of www.homeshop18.com/.



```
C:\Users\dell>cd C:\Python27

C:\Python27>cd dist

C:\Python27\dist>xxe.exe
Enter the full url of target site:http://www.homeshop18.com/
Date: Wed, 04 Jul 2018 19:23:01 GMT
Server: xyz
Set-Cookie: JSESSIONID=20FA7B08139906B7BF69B36E467137E1-n4.jvmnode9a; Domain=.ho
meshop18.com; Path=/; HttpOnly
Set-Cookie: HS_ID=19002e78dc217476bcdac1ba625b5270f9d2c62217b2a9d5731799a2869be3
e5; Domain=.homeshop18.com; Expires=Tue, 02-Oct-2018 19:22:42 GMT; Path=/
X-Frame-Options: SAMEORIGIN
Access-Control-Allow-Origin: http://www.homeshop18.com
Content-Language: en
Vary: Accept-Encoding
Keep-Alive: timeout=15, max=990
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html;charset=UTF-8

Enter a url with parameter: http://www.katun.me/page.php?id=10
Its vulnerable
Remediations: Disable unwanted files, queries and sanitize your input data
Enter the full url of target site:http://www.homeshop18.com/
result.status_code is:
200

@surbhiprakash
```

Figure 7. Response code of target

4. COMPARISON OF TOOLS

There are so many vulnerability assessment tools available in the industry which are used for vulnerability Assessment only. There is a list of tools and compared list which can detect XML-based vulnerabilities. XML-based vulnerabilities assessments being detected by only a small amount of tools.

Comparative Analysis

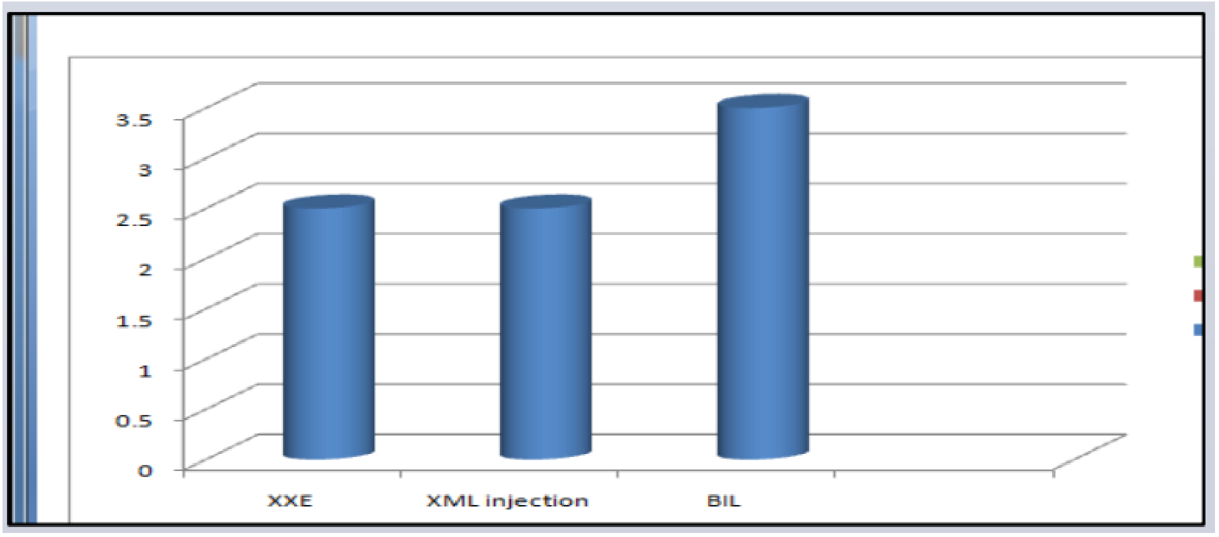
Table 3. Comparative Analysis of Tools

S.no	Assesment tools	Vulnerabilities able to detect	Count of vulnerabilities
1.	Acunentix	XXE	1
2.	Burpsuite	XML RPC, XXE	2
3.	Vega	None	0
4.	WPscan	XML RPC	1
5.	Framework designed	XML RPC, WSDL Access, XXE, Billion Laugh Attack, XML injection	5

4.1 Results and Analysis

In the theoretical study of vulnerabilities based on XML, we have explored many vulnerabilities but when it comes to the point of industrial context, we have explored that there are three major vulnerabilities based on XML. A list of parameter tempering attacks are:

- Tautology attack
- Meta Character injection
- Comment injection attack
- CDATA section injection attack
- Tag injection attack
- External entity injection attack
- Alternate encoding attack
- Injection via evaluation function



Graph 1. Results on the basis of classification

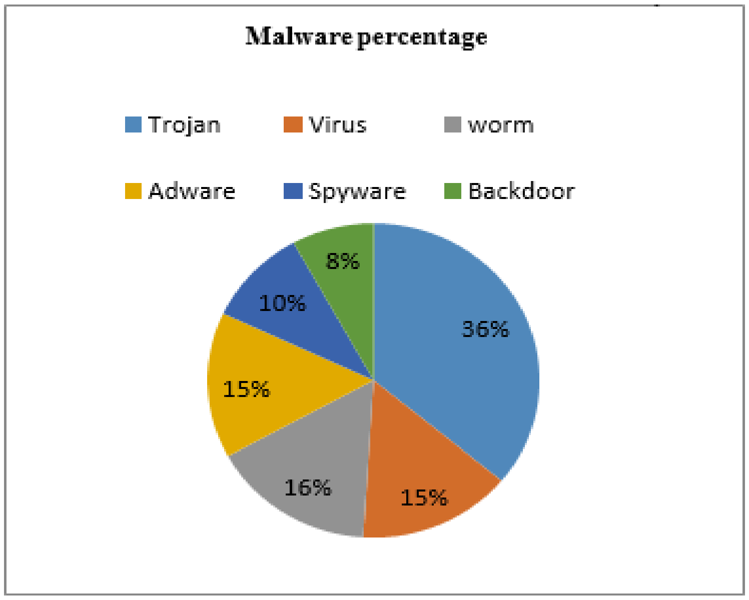


Fig 5. Pie Chart for classification of malware.

5. OBSERVATION AND CONCLUSION

XML Based vulnerabilities are one of the important issues in upcoming services like SOAP and Rest. All the form-based applications are vulnerable to injection attacks. In this work I have shown the implementation of Automatic and Effective Framework for testing XML based vulnerabilities. I have explored three main vulnerabilities like XXE , BIL and XML Injection. Statistics have been given that XML injection have higher percentage among three famous vulnerabilities found and explored.

ACKNOWLEDGEMENT

I acknowledge that the data I have presented in this review is not taken by unauthorized means if data is considered for study purposes, then it has been mentioned in the references. I have given correct information with correct usage of sources.

This work has been performed in the Ph.D. study of Indira Gandhi Delhi Technical University Women. This work has been verified in the software company **"Kadam Technologies Pvt Ltd." (A web and mobile design & and development company).**

6. REFERENCES

[1] R.Mosli, R. Li, B. Yuan, and Y. Pan, "Automated malware detection using artifacts in forensic memory images," in 2016 IEEE Symposium on Technologies for Homeland Security, HST 2016, 2016, pp. 1–6.

[2] M.Karresand, "Separating Trojan horses, viruses, and worms - A proposed taxonomy of software weapons," in IEEE Systems, Man and Cybernetics Society Information Assurance Workshop, 2003, pp. 127–134.

[3] Smartphone Market Share. Accessed: Apr. 30, 2020. [Online]. Available: https://www.idc.com/promo/smartphone-market-share/os

[4] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, ''Significant permission identification for Machine-Learning-Based Android malware detection,'' IEEE Trans. Ind. Informat., vol. 14, no. 7, pp. 3216–3225, Jul. 2018

[5] M. Taleby, Q. Li, M. Rabbani, and A. Raza, ''A survey on smartphones security: Software vulnerabilities, malware, and attacks,'' Int. J. Adv. Comput. Sci. Appl., vol. 8, no. 10, pp. 30–45, 2017

[6] A. A. A. Samra, H. N. Qunoo, F. Al-Rubaie, and H. El-Talli, ''A survey of static Android malware detection techniques,'' in Proc. IEEE 7th Palestinian Int. Conf. Electr. Comput. Eng. (PICECE), Mar. 2019, pp. 1–6

[7] . K. Gyamfi and E. Owusu, ''Survey of mobile malware analysis, detection techniques and tool,'' in Proc. IEEE 9th Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON), Vancouver, BC, Canada, Nov. 2018, pp. 1101–1107.

[8] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. S. Gaur, M. Conti, and M. Rajarajan, ‘‘Android security: A survey of issues, malware penetration, and defenses,’’ IEEE Commun. Surveys Tuts., vol. 17, no. 2, pp. 998–1022, 2nd Quart., 2015

[9] M. L. Polla, F. Martinelli, and D. Sgandurra, ‘‘A survey on security for mobile devices,’’ IEEE Commun. Surveys Tuts., vol. 15, no. 1, pp. 446–471, 1st Quart., 2012.

[10] Web of Science. Accessed: Apr. 30, 2020. [Online]. Available: https:// webofknowledge.com

[11] Accessed: Apr. 30, 2020. [Online]. Available: https://ieeexplore.ieee.

[12] SpringerLink. Accessed: Apr. 30, 2020. [Online]. Available: https://link. springer.com

[13] H. Zhang, S. Luo, Y. Zhang, and L. Pan, ‘‘An efficient Android malware detection system based on method-level behavioral semantic analysis,’’ IEEE Access, vol. 7, pp. 69246–69256, 2019

[14] S. Lou, S. Cheng, J. Huang, and F. Jiang, ‘‘TFDroid: Android malware detection by topics and sensitive data flows using machine learning techniques,’’ in Proc. IEEE 2nd Int. Conf. Inf. Comput. Technol. (ICICT), Kahului, HI, USA, Mar. 2019, pp. 30–36.

[15] M. Lindorfer, M. Neugschwandtner, and C. Platzer, ‘‘MARVIN: Efficient and comprehensive mobile app classification through static and dynamic analysis,’’ in Proc. IEEE 39th Annu. Comput. Softw. Appl. Conf., Jul. 2015, pp. 422–433.

[16] Z. Ma, H. Ge, Y. Liu, M. Zhao, and J. Ma, ‘‘A combination method for Android malware detection based on control flow graphs and machine learning algorithms,’’ IEEE Access, vol. 7, pp. 21235–21245, 2019.

[17] T. Gao, W. Peng, D. Sisodia, T. K. Saha, F. Li, and M. Al Hasan, ‘‘Android malware detection via graphlet sampling,’’ IEEE Trans. Mobile Comput., vol. 18, no. 12, pp. 2754–2767, Dec. 2019

[18] A. N. Mucciardi and E. E. Gose, ‘‘A comparison of seven techniques for choosing subsets of pattern recognition properties,’’ IEEE Trans. Comput., vol. C-20, no. 9, pp. 1023–1031, Sep. 1971.

[19] W. Han, J. Xue, Y. Wang, L. Huang, Z. Kong, MalDAE : Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics, Comput. Secur. 83 (2019)208–233, http://dx.doi.org/10.1016/j.cose.2019.02. 007.

[20] Y. Gao, Z. Lu, Y. Luo, Survey on malware anti-analysis, in: 5th International Conference on Intelligent Control and Information Processing, ICICIP 2014 - Proceedings, 2015, pp. 270–275, http://dx.doi.org/10.1109/ICICIP. 2014.7010353.

[21] J. Singh, J. Singh, Ransomware: an illustration of malicious cryptography (2) (2019), 1608–1611, http://dx.doi.org/10.35940/ijrte.B2327.078219.

[22] W. Zhang, H. Wang, H. He, P. Liu, DAMBA: Detecting android malware by ORGB analysis, IEEE Trans. Reliab. 69 (1) (2020) 55–69, http://dx.doi.org/10. 1109/TR.2019.2924677.

[23] J. Singh, J. Singh, J. Singh, Assessment of supervised machine learning algorithms using dynamic API calls for malware detection assessment of supervised machine learning algorithms using dynamic API calls for malware detection, Int. J. Comput. Appl. (2020) 1–8, http://dx.doi.org/10.1080/1206212X.2020. 1732641.

[24] M. Rabbani, Y.L. Wang, R. Khoshkangini, H. Jelodar, R. Zhao, P. Hu, A hybrid machine learning approach for malicious behaviour detection and recognition in cloud computing, J. Netw. Comput. Appl. 151 (2020) 102507, http://dx.doi. org/10.1016/j.jnca.2019.102507

[25] Wikipedia, Malware, 2020, https://en.wikipedia.org/wiki/Malware.

[26] Omer Aslan and Refik Samet, “A comprehensive review on malware detection approaches,” IEEE Access 8, 6249–6271, 2020.

[27] Deepti Gupta, Smriti Bhatt, Maanak Gupta, Olumide Kayode, and Ali Saman Tosun, “Access control model for google cloud iot. In 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity),” IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS). IEEE, 198–208, 2020.

[28] Wei Yan, “CAS: A framework of online detecting advance malware families for cloud-based security,” In 2012 1st IEEE International Conference on Communications in China (ICCC). IEEE, 220–225, 2012.

[29] Qublai K Ali Mirza, Irfan Awan, and Muhammad Younas, “A CloudBased Energy Efficient Hosting Model for Malware Detection Framework,” In 2018 IEEE Global Communications Conference (GLOBECOM). IEEE, 1–6, 2018.

[30] Deepti Gupta, Olumide Kayode, Smriti Bhatt, Maanak Gupta, and Ali Saman Tosun, “Learner’s Dilemma: IoT Devices Training Strategies in Collaborative Deep Learning,” In 2020 IEEE 6th World Forum on Internet of Things (WF-IoT). IEEE, 1–6, 2020

[31] Jagsir Singh, Jaswinder Singh “ A survey on machine learning-based malware detection in executable files” Journal of Systems Architecture(Elsevier).

DOI: https://doi.org/10.15379/ijmst.v10i1.2630