ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 1, Issue 7, September 2012

Vulnerability Discovery with Attack Injection

B.sandhya, S.Vasundra

***Abstract*—Software systems are prone to security vulnerabilities. Security is very important aspect of any software. In spite of security mechanisms available, there is increasing security threats revealed every year. This is a continuous problem with software systems as new threats are tried out by attackers. This paper solves this problem by developing a framework with an attack injection methodology which continuously discovers vulnerabilities in software systems and allows security administrators to resolve the issues. A prototype application is built to demonstrate the methodology. The application gets protocol details from networked server and makes attacks to discover vulnerabilities and the vulnerabilities thus discovered are persisted to a database. This helps in rectifying problems or fixing bugs in the software that causes security vulnerabilities.**

I. INTRODUCTION

Computer usage has become part of human beings in all walks of life. It has become an essential part of any work. This does mean that reliance on computer systems is increasing dramatically each and every year. The emerging technologies, advancements in storage and security, the Internet and all such things paved way for this reliable usage of computers in all businesses and personal pursuits. The software development processes have been around ever since man started using systems to develop software. However, every software built by humans based on some systematic approach known as software process model, may have bugs hidden. With respect to security, the bugs are known as vulnerabilities. These vulnerabilities are exploited by hackers and adversaries to perform security attacks for monetary or other gains. This is the reason for this research paper. This paper aims at developing a new methodology that helps in discovering security vulnerabilities of networked servers. This paper also presents a Vulnerability Prediction Tool that is meant for generating attacks on target server based on the protocol specifications of target server. Once the vulnerabilities are discovered, they are handed over to the team involved in the development of target server. The software development team follows traditional debugging and other techniques to spot the vulnerabilities and rectify them. This is an experimental approach for the verification of fault handling mechanisms (fault removal) and for the estimation of various parameters that characterize an operational system (fault forecasting), such as fault coverage and error latency [6], [7].
The proposed methodology and vulnerability detection tool are meant for proactively making attacks on target server, in this case it is Red Hat Linux, with a good intention to make a list of vulnerabilities and let the development and security personnel of the target server to identify and fix the problems so as to avoid the security threats from real adversaries who make attacks for monetary or other gains. While making attacks the expected behavior and abnormal behaviors of server against various attacks are observed. The expected behavior does mean that there is no security vulnerability exists with respect to the attack made based on the test definition which is driven by protocol specification. Any behavior of server that deviates from expected one confirms the presence of vulnerability. These results are valuable to the team that takes care of security of the target server. The nice thing about the proposed vulnerability detection tool is that, it does not need the source code of the target server in order to make attacks on it. Instead it treats server OS as block box and perform attacks. The tool simply needs protocol specifications of target server that give enough information to derive test definitions, generate attacks and inject them to complete the experiments and discover vulnerabilities in the target server. The tool follows certain phases such as generate attacks, inject attacks and look for errors or failures. The generate attacks phase takes protocol specifications as input and generates a set of test definitions. In turn these test definitions are used to generate attacks. The inject attacks phase is actually injecting attacks generated by generate attacks phase. Once injecting attacks is completed, it results in discovery of vulnerabilities. The following sections throw light into review of literature, proposed methodology and tool creation details.

PROPOSED METHODOLOGY

We propose an attack injection methodology that discovers vulnerabilities in select networked servers. Networked servers have inherent and latent vulnerabilities that are not unearthed unless some attacks are made. Exception [8] and FTAPE [9] are examples of tools that can inject hardware or software faults in a target system under evaluation. The emulation of other types of faults has also been accomplished with fault injection techniques, for example, software and operator faults [10], [11]. Robustness testing mechanisms study the behavior of a system in the presence of erroneous input conditions. Their origin comes both from the software testing and fault injection communities, and they have been applied to various areas, for instance, POSIX APIs and device driver interfaces [12], [13].

When attacks are made by adversaries, the network servers fail to perform their functionalities. The security attacks made by hackers cause the IT systems to lose important sensitive information. It is essential to test the servers to find whether there are security vulnerabilities.

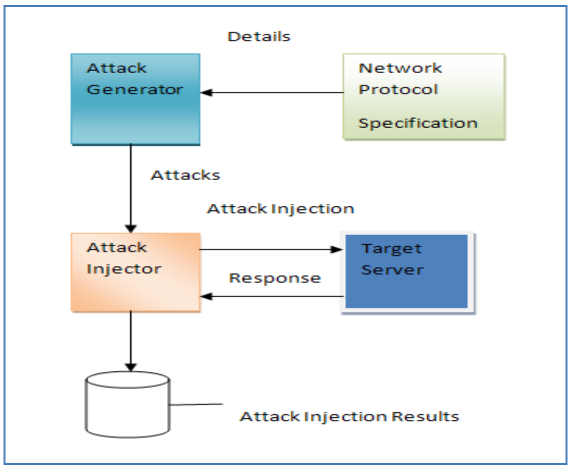


Fig. 1 – Architecture of Proposed Methodology

The fig. 1 shows architecture of proposed methodology.

As can be seen in fig. 1, the architecture shows the methodology used in discovering vulnerabilities in the target servers. Networked servers are of many types such as Windows, Linux, and Apple Mac and so on. Before making attacks it is required to know the protocols used by the target server. For this reason network server protocol specifications are used. From the protocol specifications, the proposed system takes details of a particular protocol. The extracted protocol specification details are kept in a database for further usage. Based on the specification, various kinds of tests are defined. The tests thus defined are used in the proposed framework. The attack generator component of the proposed architecture makes use of the defined tests and generates various attack instances. Such attacks are saved to a database for further use. These attacks are used by attack injector which is responsible for actually injecting attacks into target server and get the responses. The attack injection result may be positive or negative. Positive does mean that attack is made successfully and certain vulnerability has been discovered. Negative does mean that attack was not successful and that indicates robustness of target server in that area. The attack injector component various sub components for the purpose of processing attacks, injecting packets, and collecting response and execution data. The results of the attack injection process are saved to database known as attack injection results database. Target server is the component that existed in the real world. It is a networked server with security in place. However, there might be unknown security vulnerabilities that are discovered by the proposed framework.

ALGORITHMS USED

The algorithms proposed in [1] are used for the experiments. They are given in this paper. The first algorithm is meant for generating attacks while the second algorithm is meant for generating of malicious strings.

```
Test value
Input:protocol ← specification of the network  protocol
used                    by the server
Output:Attacks
S  ← Set of all states of the rotocol specification
Foreach State s Є S do
M ← set of message specification of s
Transaction  ← set of ordered network packets necessary
to reach s
P ← Φ
Foreach MessageSpecifiation m є VM do
Foreach FieldSpecification f є m do
If f is type Numbers then
f‘ ← all boundaries values,plus some intermediary illegal
values, from specification
elseif   f  is type words then
f’ ← combination.of predefined malicious tokens
m’ ← copy of m replacing f with f’
P ← P υ  (set of all network packets based on m‘
specification)
Foreach attack_packet є P do
Attack ← Transitions  υ [attack_packet]
Attacks ←  Attacks  υ (attack)
return:Attacks
```

Fig. 2 – Algorithm for generating of attacks

This algorithm traverses all message types and states of protocol in order to maximize the attack space. Based on the message type each and every test case is generated. This algorithm is different from others as it populates each field with wrong values systematically instead of using only legal values. Overall, this algorithm takes specification of network server protocol and generates various security attacks to discover vulnerabilities in the specified server.

```
Generate illegalWords
Input:Words ← specification of the field
Input:Tokens ← predefined list of malicious tokens, e.g., taken from hacker exploits
Input:payload ← predefined list of special tokens to fill in the malicious tokens
Input:max_combinations ← maximum number of token combinations
Output:IllegalWords
// step 1:expand list of tokens
Foreach t є Tokens do
If t includes keywords S(PAYLOAD) then
Foreach p є Payload do
t' ← copy of t replacing S(PAYLOAD)with p
Tokens ← Tokens U (t')
Tokens ← Tokens \ (t)
//step2 : generate and append all k-combinations of tokens
k ← 1
While k <= max_combinations
k-combinations ← (tokens k)//combinations of k elements from Tokens
IllegalWords ← IllegalWords U k-combinations k ← k+1
Return:IllegalWords
```

Fig. 3 – Algorithm for generating malicious strings

As can be viewed in fig. 3, this algorithm takes specification of the field, list of malicious tokens, predefined list of special tokens and maximum number of token combinations as input and generates malicious strings as output. This output is used in discovering vulnerabilities in the target server.

EXPERIMENTS

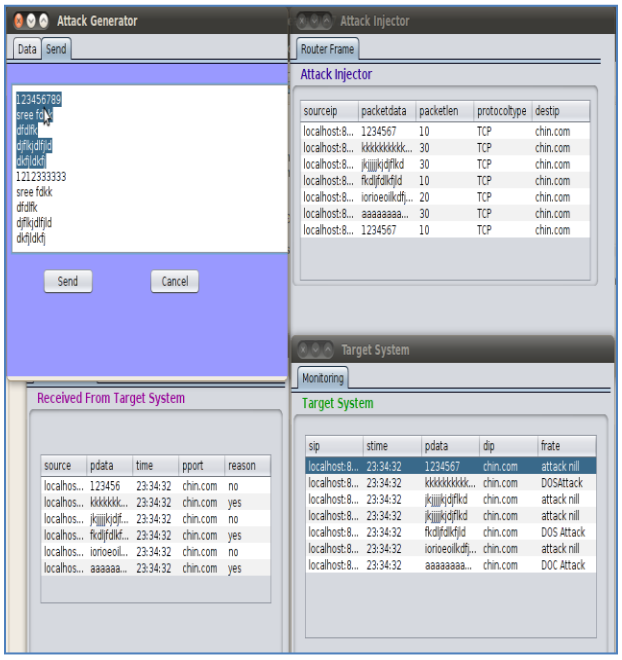
This section describes the environment used for conducting experiments and the way experiments are conducted.

Environment

The environment used to build a prototype application for the purpose of making experiments include JDK 1.6 (JSE 6.0), Net Beans IDE, Oracle 10G Express Edition, a PC with Linux Server running. No special hardware or software is required apart from this.

Prototype Application

An application is built with GUI (Graphical User Interface) that facilitates to achieve the goal of this paper. The aim of this paper is to make attacks on networked server with the intention to discover vulnerabilities and let security personnel to fix the discovered vulnerabilities. General IDS (Intrusion Detection System) applications are meant for detecting intrusions made by adversaries. However, the proposed tool is to make attacks proactively and discover vulnerabilities. The attacks are devised based on the knowledge gained from the protocol specifications of target server. The vulnerability detection tool or the application developed in this paper ensures that attacks are made based on test definitions that are driven by the facts given in the protocol specifications of target server.



```
123456789
sree fdkk
dfdlfk
djflkjdlfjld
dkfjldkfj
1212333333
sree fdkk
dfdlfk
djflkjdlfjld
dkfjldkfj
```

sourceip	packetdata	packetlen	protocoltype	destip
localhost:8...	1234567	10	TCP	chin.com
localhost:8...	kkkkkkkkkk...	30	TCP	chin.com
localhost:8...	jkjjjjkjdjflkd	30	TCP	chin.com
localhost:8...	fkdljfdlkfjld	10	TCP	chin.com
localhost:8...	iorioeoilkdfj...	20	TCP	chin.com
localhost:8...	aaaaaaaa...	30	TCP	chin.com
localhost:8...	1234567	10	TCP	chin.com

source	pdata	time	pport	reason
localhos...	123456	23:34:32	chin.com	no
localhos...	kkkkkk...	23:34:32	chin.com	yes
localhos...	jkjjjjkjdjf...	23:34:32	chin.com	no
localhos...	fkdljfdlkf...	23:34:32	chin.com	yes
localhos...	iorioeoil...	23:34:32	chin.com	no
localhos...	aaaaaa...	23:34:32	chin.com	yes

sip	stime	pdata	dip	frate
localhost:8...	23:34:32	1234567	chin.com	attack nill
localhost:8...	23:34:32	kkkkkkkkkk...	chin.com	DOSAttack
localhost:8...	23:34:32	jkjjjjkjdjflkd	chin.com	attack nill
localhost:8...	23:34:32	jkjjjjkjdjflkd	chin.com	attack nill
localhost:8...	23:34:32	fkdljfdlkfjld	chin.com	DOS Attack
localhost:8...	23:34:32	iorioeoilkdfj...	chin.com	attack nill
localhost:8...	23:34:32	aaaaaaaa...	chin.com	DOC Attack

Fig. 4 – Proposed Vulnerability Detection Tool

As can be seen in fig. 4, the tool is built in Linux environment using Java programming language. Java's SWING API is used to build graphical interface while the network API is used for communication with target server. IO is used to perform read and write operations on local file system while JDBC (Java Database Connectivity) is used to perform database operations. The application has two important components or modules. They are known as attack generator and attack injector. The attack generator module is used to define various tests meant for generating attacks. These tests are based on the protocol specifications collected from target server. In our experiments we used Red Hot Linux server as target server. The vulnerability discovery tool also tested in Linux environment. The procedure followed by the tool is very much similar to the description given about the architecture of proposed methodology as shown in fig. 1.

The attack generator module is responsible to make use of tests defined and generate possible attacks. It does mean that an attack is generated for each test definition. The collection of attacks that generated is persisted to database. These attacks are used by the attack injector module. The attack injector module is responsible to actually inject attacks into target server. It monitors the attacking process and also results there of. The attack results are saved to database as attack injection results. The results of attacks are then used by security personnel of target server to focus on the vulnerabilities and take steps to ensure that those vulnerabilities are rectified and they no longer give chance to security threats in future. As can be seen in the results shown in fig. 4, it is evident that there are some vulnerabilities discovered in case of Red Hot Linux server which is freely available for download over Internet.

CONCLUSION

This paper presents a methodology that makes attacks on networked servers and discovers security vulnerabilities. To demonstrate this prototype application is built that takes protocol specification details from server and performs various attacks on the server and discovers vulnerabilities. The discovered vulnerabilities are then persisted into database for further steps to fix the bugs in the software in which vulnerabilities are found. The experiments are made on Linux server and the empirical results revealed that the proposed tool is capable of discovering vulnerabilities effectively.

REFERENCES

[1] Joa˜o Antunes, Student Member, IEEE, Nuno Neves, Member, IEEE,Miguel Correia, Member, IEEE, Paulo Verissimo, Fellow, IEEE, and Rui Neves. Vulnerability Discovery with Attack Injection. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 36, NO. XX, XXXXXXX 2010

[2] J. Arlat, A. Costes, Y. Crouzet, J.-C. Laprie, and D. Powell, "Fault Injection and Dependability Evaluation of Fault-Tolerant Systems," IEEE Trans. Computers, vol. 42, no. 8, pp. 913-923, Aug. 1993.

[3] M.-C. Hsueh and T.K. Tsai, "Fault Injection Techniques and Tools," Computer, vol. 30, no. 4, pp. 75-82, Apr. 1997.

[4] P. Koopman and J. DeVale, "Comparing the Robustness of POSIX Operating Systems," Proc. Int'l Symp. Fault-Tolerant Computing, pp. 30-37, June 1999.

[5] B.P. Miller, L. Fredriksen, and B. So, "An Empirical Study of the Reliability of UNIX Utilities," Comm. ACM, vol. 33, no. 12, pp. 32- 44, 1990.

[6] P. Oehlert, "Violating Assumptions with Fuzzing," IEEE Security and Privacy, vol. 3, no. 2, pp. 58-62, xpls/abs_all.jsp?arnumber=1423963, Mar./Apr. 2005.

[7] Qualys, Inc., "QualysGuard Enterprise,". com, 2008.

[8] J. Carreira, H. Madeira, and J.G. Silva, "Xception: Software Fault Injection and Monitoring in Processor Functional Units," Proc. Int'l Working Conf. Dependable Computing for Critical Applications, pp. 135-149, http://citeseer.ist.psu.edu/54044.html; http://dsg.dei.uc.pt/Papers/dcca95.ps.Z, Jan. 1995.

[9] T.K. Tsai and R.K. Iyer, "Measuring Fault Tolerance with the FTAPE Fault Injection Tool," Proc. Int'l Conf. Modeling Techniques and Tools for Computer Performance Evaluation, pp. 26-40, http://portal.acm.org/citation.cfm?id=746851&dl=ACM&coll=&CFID=15151515&CFTOKEN=6184618, Sept. 1995.

[10] D. Wagner, J.S. Foster, E.A. Brewer, and A. Aiken, "A First Step Towards Automated Detection of Buffer Overrun Vulnerabilities," Proc. Network and Distributed System Security Symp., Feb. 2000.

[11] J . Dura˜es and H. Madeira, "Definition of Software Fault Emulation Operators: A Field Data Study," Proc. Int'l Conf. Dependable Systems and Networks, pp. 105-114, June 2003.

[12] P. Koopman and J. DeVale, "Comparing the Robustness of POSIX Operating Systems," Proc. Int'l Symp. Fault-Tolerant Computing, pp. 30-37, June 1999.

[13] M. Mendonc¸a and N. Neves, "Robustness Testing of the Windows DDK," Proc. Int'l Conf. Dependable Systems and Networks, pp. 554- 564, June 2007.

[14] Microsoft, Corp., "A Detailed Description of the Data Execution Prevention (DEP) Feature in Windows XP Service Pack 2, Windows XP Tablet PC Edition 2005, and Windows Server 2003," http://support.microsoft.com/kb/875352, Sept. 2006.

B.Sandhya M.Tech Student, CSE Department, JNTUACEA, Anantapur, Andhra Pradesh, India.

S.Vasundra M.Tech,(Ph.D.),Associate Professor, CSE Department, JNTUACEA, Anantapur, Andhra Pradesh, India.