ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 6, June 2015*

# Susceptibilities Detection Approach in Network Connected Servers using the Composite Fault Prototypical

**Sivakumar Kalimuthu[1], Premylla Jeremiah[2], Kannan Ponkoodanlingam[3], Anatte Schaffer Anak Juslen[4]**

Research Scholar, Faculty of Computer Science and Engineering, Sathyabama University, India [1]

Research Scholar, University Technology Malaysia (UTM), Malaysia[2]

Sr. Lecturer, School of Computing and IT, INTI International University, Malaysia[3]

Bachelor in Computing (Hons) Student, University of Greenwich UK (SEGI College Kualalumpur), Malaysia[4]

**Abstract**: The enormous issue is the discovery of newer threats and different forms of attack, there by compromising the security of the system. In this paper, we first analysis the most relevant concepts underlying the view of network vulnerabilities and summarize the most known technique called Attack Injection Tool. We propose an approach to identification of potential vulnerabilities and evaluate software components security mechanisms using the composite fault prototypical. The approach and tool is based on the awareness that inserting potentially dangerous susceptibilities in a network server and web application by analysing the assessment of existing security mechanisms and tools in convention situations. This methodology is designed for the detection of vulnerability in the software components in a proposed approach design called the Susceptibilities Detection Approach (SDA) that behaves like hackers and security analyst for the discovery of susceptibility in the network connected servers. The analysis and design methodology thereby showed that this approach can be used to identify the vulnerability.

**Keywords**: Network Susceptibilities, Injection Attack, Communication Protocol Servers, Susceptibilities Detection Approach, Composite Fault Prototype.

## I. INTRODUCTION

Nowadays, Vulnerabilities gotten distinctly more intense and the tremendous self-trust set on computer network systems burdens advanced levels of reliability. The enormous growth in the development of software filed has issued with an extensive number of helpful applications with a constant enhancing functionality. Though, this kind of change will be accomplished in the bigger and more complicated project tasks, which require the coordination among a few groups such poor documentation and backings inside of the group.

Out of sight, the endless exchange off between consistent time and testing for utilization influences the nature of the software product. These elements, identified with the present development and testing procedure, have turned out to be deficient and inadequate to develop tried and true software.

Consistently, new susceptibilities will be discovered in what was already accepted to be secure applications, opening new risk and security dangers can be exploited by malicious challengers [1].

This paper exposes an attack injection methodology [1, 2] that can be utilized for susceptibilities discovery and removal. It replicates the behaviour of an opponent by infusing the attacks against a target system and reviewing the execution to figure out whether any of the attacks has caused failure [1, 2, 4]. The perception of some diverse behaviour demonstrates that attack was successful in creating a current defect.

After the Identification of the issue, a portion of the investigating strategies can be utilized, for case, by looking, at the applications control while processing the offending attacks, to locate the origin of the vulnerability and to proceed with its elimination [4, 1, 2].

According to Adelsbach A, Alessandri D, et al (2002), several tactics can be employed to improve the dependability of a system with respect to malicious faults [8]. Obviously, invasions would never occur if all susceptibilities could be eliminated. Susceptibility removal can be performed both during the development and operational phases [4].

In fact, there is a tool name called Attack in-JECtion Tool (AJECT) has been introduced, that can be used for vulnerability detection and removal [1, 4]. AJECT mimics the behavior of an enemy by inserting attacks against a target system.

At that point, it detects the execution of the system to figure out whether the attacks have caused a failure. In the favorable case, this demonstrates that the assault was effective, which uncovers the presence of susceptibility. After the recognizable proof of defect, one can utilize conventional debugging procedures to look at the application code and running environment, to figure out the root the susceptibility and let its consequent elimination.

The proposed approach and designed methodology is called Susceptibilities Detection Approach (SDA). It was

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 6, June 2015*

intended to search for susceptibilities in system server applications, in spite of the fact that it can likewise be used with nearby spirits. We have select servers because, from a security perspective, they are most likely the most applicable parts that need protection since they have the essential contact purposes of a system facility.

SDA does not require the source code of the server to perform the attacks, that is, it regards the server as a black box. Nonetheless, have the capacity to create clever attacks; SDA needs to get a detail of the convention used in the correspondence with the server.

To exhibit the convenience of our methodology, we have analysed three different network protocol namely SMTP, IKE, and POP3 convention [4, 7] for assault infusion tests. The principle target was to examine if SDA could naturally find already obscure susceptibilities in completely created and progressive server applications. Despite the fact that the number and kind of target applications was not comprehensive, they are all things considered a delegate test of the universe of the system servers.

Our assessment acknowledged that SDA could discover various classes of vulnerabilities in three of the servers such as POP3, IKE, and SMTP and it will help the designers in their removal by giving the experiments, that is, the susceptibility syndromes.

These investigations additionally prompt other absorbing conclusions. For example, we admitted the desire that complex protocols are significantly more inclined to vulnerabilities than easier ones since every single identified vulnerability were identified with the SMTP, POP3 protocol [1], and IMAP protocol [2].

Additionally, based on the three e-mail servers, we found that closed source applications appear to have a higher predisposition to contain vulnerabilities According to Antunes J et al (2010) none of the open source servers was found vulnerable whereas forty two percentages of the closed source servers had problems [1].

As per the literature available these tools are very effective in locating such bugs or problems in the software systems. However, they are having limitations as they produce many false positives in the process of detecting vulnerabilities. We also discuss security issues for web application systems, and then discuss current challenges for network securities and some preliminary approaches that address some of these challenges.

A. *Identify the Susceptibility by Using Various Attacks*

Vulnerability are caused by any sudden differences that are not found during the testing phase in many cases vulnerability are found by certain unusual behaviour finding the vulnerability manually is very difficult, in order to do this we must analyse the code very carefully or subject it to different levels of testing. The small boxes represent various levels of the software which are able to generate malicious program (Fig1). A channel or an interface access allows external access for the component input data validation phase must protect the complex system. These layers are used to validate the arriving data. The attack looks for vulnerability by weakening the component with abnormal interaction like sending wrong message. A trusted system is the one that must continue to execute even in the presence of any faults. But if there are any abnormal behaviour it means that there is some vulnerability. Vulnerability is any abnormal behaviour occurring due to some illegal behaviour. If the component is not properly protected attacker can access it in an unpredicted manner causing an intrusion. If it is not properly handle, the system will not work properly.

B. *The Attack Injection methodology*

According to Antunes J, Nuno Neves et al (2010), the attack injection methodology adapts and used fault injection method to find the vulnerability using AJECT tool [1].

The methodology can be a benefit in increasing the reliability of system. Here they used injection tool to find any abnormal behaviour that attacks the component which is known as target system. The source of malicious software generates potential attacks to the functionality of the target system and it is to be a confident of the absence of the vulnerability they attack has to be able to find as many flaws. Some attacks will not be allowed by the input data validation mechanism but many attacks will allow passing.

Test cases are built not only reach all computer system but also used with every possible input. But there is a disadvantage in existing tool that it needs different types of input data and the methods to execute them. This disadvantage can be reduced by analysing the source code and creating many test cases in the injection of the attack methodology [1, 2].

However, test designer required a worthy experience to deal with testing, even though much vulnerability can be ignored. In some cases the source code will not be available as it might be developed by unknown persons. To overcome this required to generate test cases from the component interface. The tool that use should be able to generate unexpected behaviour depending on the capable of the tool it should able to examine the systems output and the system calls it executed. If there is any abnormal behaviour it should intimate that vulnerability has been found.

A vulnerability exciting the system if it crashes or if there is any abnormal behaviour the information that we collect should enable as to find the vulnerability and also its removal. The response from the component along with the offending attack enables us to know the protocol use and the execution track to know the fault. If it is unable to locate or remove the vulnerability immediate action has to be taken.

For example if the target system is business related one. The description of the attack can be used to take some kind of preventive action. Thus by blocking this kind of attack no vulnerability can be exploited there by increasing the reliability of the system.


DOI 10.17148/IJARCCE.2015.4615

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 6, June 2015*

## II. RELATED MODULES

### A. *Attack Creation Stage*

The Target System is the whole hardware and software modules that include the operating system, objective application, product libraries, framework resources and its execution environment. The Computer Server is commonly an administration that can be inquired remotely from customer projects [3].

The objective application utilizes a recognized protocol to correspond with the customers, and these customers can do attacks by transferring using suspicious packets. Suppose the packets are not properly treated, the target can hurt several varieties of faults with discrete significances, extending, for example, from a slowdown to a crack [7]. The Network Server Protocol Specification is a Graphical User Interface (GUI) part that helps the specification of the communication protocol used by the server. These certain specification is misused by the Attack to yield an expansive number of experiments.

To create an enormous number of attacks, it uses a well-defined algorithm for the value test case generation, and the server's communication protocol.

These attacks are thus injected into the network and the performance of the server is monitored in the target system while the response is resumed to the respective client.

The presence of any unwanted behaviour suggests the occurrence of susceptibility which was generated by some unknown attacks. These attacks can then be used to reduce the irregularity and to aid in the removal of error and know the correctness of these approaches it was performed on well-known network server protocols like SMTP, IPsec IKE and POP3.

### B. *Algorithm Used*

The value test definition algorithm used to verify whether the server can handle wrong data or not. A mechanism is used to develop unauthorized information or data from the message specification that is from each field's authorized data. Experimenting with all the possible values is difficult when the message is very large and also with fields with arbitrary textual content. Traversing the message types of the protocol is done; test case is created based on one type of message.

This algorithm fills each field with incorrect values instead of using the correct values. When a particular field holds a numerical values deriving illegal value is very simple as they corresponds to the complementary values, which means that the number does not belong to any legal data set.

To decrease the number of values to be tried and also to enhance the injection process this algorithm employs the boundary values and the complementary values.

To choose the complementary values we custom illegitimate coverage proportion and irregular scope proportion. Illegitimate scope proportion chooses just as inaccessible qualities based on the aggregate scope of illicit qualities. Random coverage ratio uses illegal numbers to select illegal values.

### C. *Value-Test generation*

```
Test Value:
Input: Protocol ←specification of the network protocol used by
server
Output: Attacks
S ← set of all states of the Protocol specification
foreach State s← ϵ S do
   M ← set of message specification of s
   Transitions ← set of ordered network packets necessary to reach s
   P ← ϕ
foreach MessageSpecification m ← ϵ M do
   foreach FieldSpecification f← ϵ m do
     if f is type Numbers then
        f' all boundary values, plus some intermediary illegal values,
        from f specification

     elseif f is type Words then
        f' ← combination of predefined malicious tokens
        m' ← copy of m replacing f with f'
        P ←P ⋃ {set of all network packets based on m'
              specification}
 foreach attack_packet ϵ P do
    attack ← Transitions ⋃ {attack_packet}
    Attacks ← Attacks ⋃ {attack}
return Attacks
```

Pseudocode 1:The Value Test generation Algorithm [1]

### D. *Malicious string Generation*

```
Generate Illegal Words
Input: Words ←specification on the field
Input: Tokens ←predefine list of malicious eg: taken from
hacker exploits
Input: Payload← predefine 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 ϵ Tokens do
if t includes keyword $ (PAYLOAD) then
   foreach p ϵ Payoad do
      t' ← copy of t replacing $ (PAYLOAD) with p
     Tokens← Tokens ∪ {t}
    Tokens← Tokens \ {t}

// Step 2: 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 ∪ k-combinations
   k ← k+1
return IllegalWords
```

Pseudo code 2: The generation of malicious strings algorithm [1]

Malicious string Generation algorithm uses malicious token or the expression obtained from the Hackers as an input. Attack Injection Tool enlarges the keyword PAYLOAD with the lines taken from the some other file with payload data. The payload can be occupied with long strings, the username that we know. The results that are obtained from the combination of both files can be used to outline the illegal word fields.


DOI 10.17148/IJARCCE.2015.4615

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 6, June 2015*

## III. NETWORK SERVERS

A network server is a computer system that offers numerous shared resources to workstations and other servers on a computer network. The shared resources can include email services, hardware access, and disk space. It's commonly the case that nearly any computer can be a "network server." What separates a server from a workstation is not the hardware, but rather the function performed by the computer [10]. There are three basic network communication servers such as SMTP, IPsec, and POP3 taking into consideration of this analysis section part, which will aid in the removal of error and know the correctness of these approaches it was performed on well-known network server protocols like SMTP, IPsec IKE and POP3. The result thereby showed that this approach can be used to identify the vulnerability and would enable us to forming SDA algorithm. We also discuss security issues for web application systems, and then discuss current challenges for network securities and some preliminary approaches that address some of these challenges. This part provides a brief overview of the SMTP, IPsec, and POP3 communication protocol that is used by the servers under test. Moreover it defines the finite state machines of the network server protocols.

### A. *Post Office Protocol3 (POP3) General Operation, Session states and Client/Server Communication*

POP3 is defined in terms of a finite state machine with a session transitioning through three states namely authorization state, transaction state and update state during the course of its "lifetime", as illustrated in below figure 2 [11].

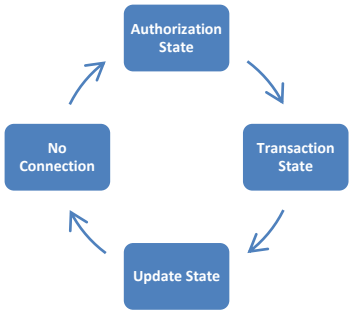


Fig 1: Post Office Protocol (POP3) Finite State Machine [11]

*step1: Establish a TCP Connection [No Connection state to Authorization state]*
*step2: Upon successful Authentication [Authorization state to Transaction state]*
*step 3: Done with Transaction; issue QUIT Command [Transaction state to Update state]*
*step4: Delete Marked Message, Release Resources, Terminate Connection [Update state to No Connection state]*

The session goes through every state once and just once, in the subsequent classification:

***Authorization* State:** The server gives a welcome to the customer to show that it is prepared for commands. The customer then gives validation data to permit access to the user's mail box.

***Transaction* State:** The customer is permitted to perform different operations on the mail box. These incorporate posting and recovering messages, and stamping recovered messages for deletion.

***Update* State:** When the customer is finished with every last bit of its tasks and issues the *QUIT* command [11], the session enters this state naturally, where the server really erases the messages checked for deletion in the Transaction state. The session is then finished up and the TCP association between the two terminated [11].

POP uses a finite state machine to depict its operation, yet it is exceptionally straightforward because it is linear. When a TCP connection is established between a POP3 customer or client and POP3 server, the session continues through three states in succession, after which the connection is ended. POP3 is composed so that just certain commands may be sent in each of these states. POP3 is a client/server protocol that is showed using a simple linear sequence of states. A POP3 session begins with a client creating a TCP connection with a POP3 server, and soon thereafter the session is in the Authorization state. After successful validation or authentication, the session transfers to the Transaction state, where the client possible to accomplish mail access transaction. When it is done, the client closures the session and the Update state is entered naturally, where clean-up functions are performed and the POP3 session finished [11].

### B. *The Simple Mail Transport Protocol (SMTP)*

The SMTP mail protocol was intended to be an easily implemented, trustworthy mechanism for moving messages starting with one trusted host then onto the next. This paper incorporates a diagram of the protocol; however the conclusive specification is (RFC821). SMTP is indicated autonomous of a transport service. This paper interprets a SMTP execution using TCP/IP, which is the most well-known transport medium being used today for SMTP on microcomputers. SMTP is doled out to the lasting TCP/IP port 25 [12]. The SMTP specification describes a lock-step protocol in which the sender and the recipient transmit particularly arranged ASCII messages to each other, anticipating a reaction before proceeding. At an abnormal state, the SMTP structural planning can be depicted by a basic limited state machine which contains three principle states (see Figure 3). The apparatus is checking reply codes, sending new charges, or sending the message substance [12].

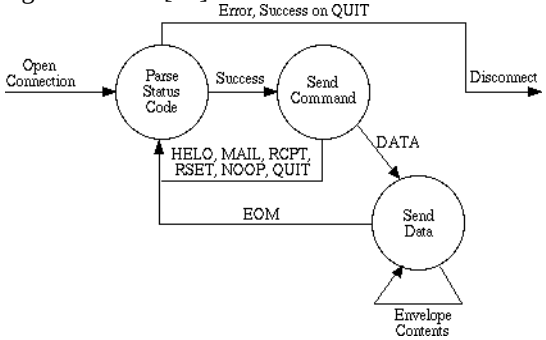


Fig 2: SMTP Finite State Machine (FSM) [12]


DOI 10.17148/IJARCCE.2015.4615

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 6, June 2015*

SMTP states a small essential command set, with numerous possible commands comprised for accessibility purposes.
The following Table 2 illustrations the minimal set essential for an SMTP sending client.

| Commands | Description |
|---|---|
| HELO | Initial State Identification |
| MAIL | Mail Sender Reverse Path |
| RCPT | One Recipient's Forward Path |
| DATA | Mail Message Text State |
| RSET | Abort Transaction and Reset all buffers |
| NOOP | No Operation |
| QUIT | Commit Message and Close Channel |

Table 3: A Minimum SMTP Command Set [12]

SMIT Commands may have either more parameters or zero parameter and will be issued as ASCII plain content strings.
A command does not span lines and is ended with a line-food pair, carriage-return [13]. The end pair finishes the command line when it is experienced. The following snip code is indicates a using SMTP Authentication in conjunction with MicroSoft Exchange.

```
package require mime
package require smtp
proc send_email {from to subject body} {
  set opts {}
  lappend opts -servers [list 10.0.0.1]
  lappend opts -ports [list 25]
  lappend opts -username user
  lappend opts -password pass
  lappend opts -header [list "Subject" $subject]
  lappend opts -header [list "From" $from]
  lappend opts -header [list "To" $to]
  set mime_msg [mime::initialize -canonical "text/plain" -encoding "7bit" -
string $body]

  smtp::sendmessage $mime_msg {*}$opts -queue false -atleastone false -
usetls false
  mime::finalize $mime_msg
                                } // end of proc send_email
send_email "Twylite <twylite@mydomain>" "Person <someone@somewhere>"
"Test" {this is my mail message.}
```

Fig 3: Using SMTP Authentication in conjunction with MicroSoft Exchange [13]

However, the SMTP convention does not handle any of the fields you would take up with a standard mail message. These fields, which make up a message that fits in with (RFC822), are fabricated and parsed by the mail taking care of specialists on either end of the SMTP exchange. SMTP treats the mail message in an obscure way, sending the headers and message body at the same time amid the Send Data state.
The SMTP code just looks at the message to learn EOM straightforwardness conditions. SMTP places the way data created in an exchange at the front of the message substance. This zone is frequently called the envelope.

C. Internet Key Exchange (IKE) protocol
This IKE protocol is used to create and maintain Internet Protocol Security (IPSec) associations and secure tunnels in the IP layer [20].

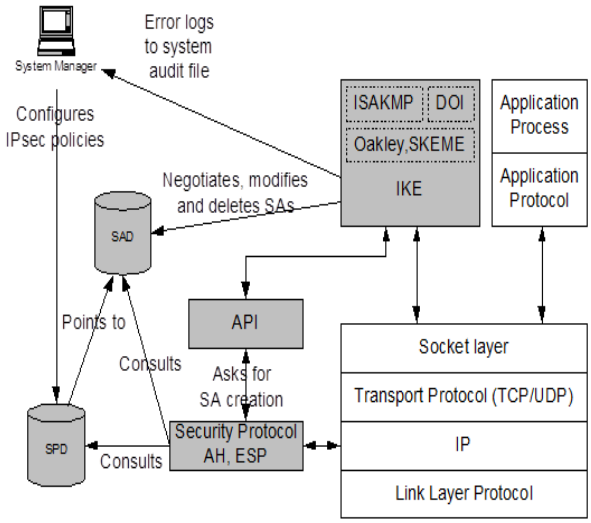


Fig 4: IPsec protocol suite - Internal structure [21]

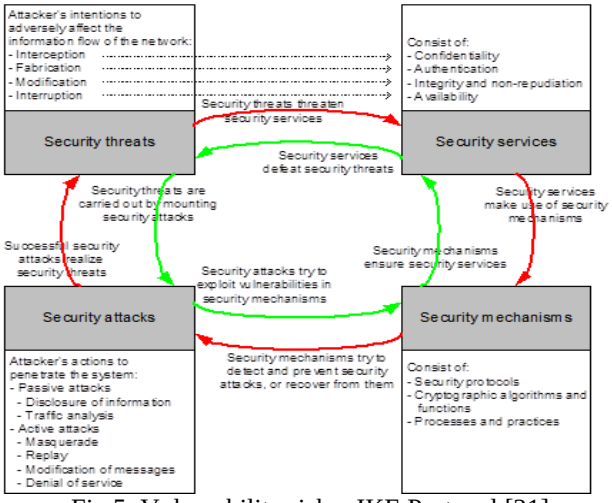


Fig 5: Vulnerability risk – IKE Protocol [21]

Green circle & Red circle indicates that, Security is engaged in spite of the mounted attacks and Security threats are understood by successful attacks respectively [21]. IKE was made from a few other key management protocols and is the default for IPSec, yet other key management protocols can be utilized. In all actuality, no key management is needed for IPSec-functions and the keys can be physically overseen. Nonetheless, manual key management is not fascinating for all executions because of the managerial overhead and the way that keys never terminate [23].

## IV. PROPOSED MODEL

A. *Susceptibility Detection Approach (SDA)*
Susceptibility is a powerless spot in the system framework that may be abused by a security hazard. Risks are the possible results and effects of unidentified susceptibilities. In a manner of speaking, fail to do Windows Updates on your Web server is inadequateness. A rate of the risks related with that deficiency consolidate on data loss, staff time, hours or days of web site downtime anticipated that would remake a server after its been exchanged off. Attacks on and inside the network system comes in

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 6, June 2015*

various mixed bags. Ordinarily the attackers don't even know who they are attacking, however there are occasions of systems or associations that are particularly focused on. Taking in the distinctive techniques used to trade off PCs and systems will give you the important viewpoint to continue. The several susceptibilities on network represent prospective costs such as assets, money and time [9]. Several tools currently available to check the current safekeeping network states such as open ports, unpatched software, and other weaknesses.

However, quite few of these software emphases on a specific machine, while others can scan your entire network [9]. Susceptibility Detection approach can reduce the impact of malicious attacks by identifying the possible vulnerability through an organization's network servers. The perception of some unusual behaviour demonstrates that an attack was effective in setting off a current defect. After the recognizable proof of the issue, traditional investigating procedures can be utilized, by analyzing the application's control stream while handling the aberrant attacks, to find the unusual of the susceptibility and to continue with its end.

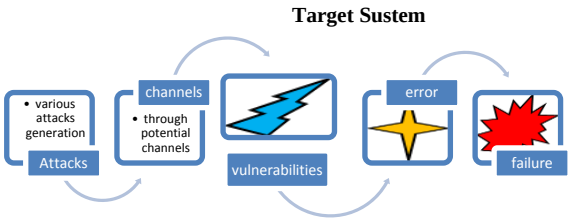


Fig 6: Architecture of proposed SDA model

The proposed approach in SDA basically targets system server applications, despite the fact that it can likewise be used with most local spirits. We picked servers as, from a security point of view; they are presumably the most significant parts that need protection. The proposed approach and methodology does not require the source code of the server to perform the assaults, i.e., it regards the server as black box. Nonetheless, in request to have the capacity to produce keen attacks, SDA needs to acquire a determination of the protocol executed by the target server. To show the value of our methodology, we have directed various investigations with a few email servers such are, IKE, POP3, and IMAP servers. The fundamental goal was to demonstrate that SDA could consequently find various diverse vulnerabilities, which were depicted in bug following destinations by different individuals. The tests figured out how to affirm that SDA could be utilized to distinguish numerous vulnerabilities. Additionally, AJECT was likewise ready to find another vulnerability that was formerly unclear to the security group [2].

B. *Proposed SDA Algorithm*

The following sequence shows that, the series of steps applying in the SDA Model Algorithm.

Step 1: Create various attacks from outside the target system.
Step 2: Apply the attacks through potential channel.
Step 3: Apply vulnerability through intruders.
Step 4: Look for failures or target system error
Step 5: Prevent the attacks or eliminate the susceptibilities once it's been occurred.
Step 6: Go to step (3).
Step 7: End.

**Generate Susceptibility Words**
**Input:** *field specification* ←*Words*
**Input:** predefine list of malicious ← *Tokens*
**Input:** *Payload* ← predefine list of special tokens
**Output:** *Susceptibility Words*
*// Step 1: Create various attacks from outside the target system*
**foreach** t $\epsilon$ Tokens **do**
**if** t includes keyword Si (Suspicious ) **then**
**foreach** p $\epsilon$ **Si** Suspicious **do**
**t'** ← copy of t replacing $ (Suspicious) with p
*// Step 2: Apply the attacks through potential channel*
// combinations of n elements from tokens
**while** c ≤ *potential_channels {c ← channels}*
*Sj - combinations*← (*tokens* j)
*Illegalwords* ← *Illegalwords* ∪ j-*combinations*
Sj ← Sj+1 // 'Sj' is incremented by 1
**return** *IllegalWords*
*// Step 3: Apply* vulnerability through intruders
**foreach** Sj $\epsilon$ Tokens **do**
**t'** ← " t" replacing Si (Suspicious) with p
*// Step 4: Look* for failures or target system error
n- failure identification in the target system
*// Step 5: a*vert the attacks or remove the susceptibilities once it's been occurred
**foreach** p $\epsilon$ $\mathbf{S_i}$ Suspicious **do**
**t'** ← "t" replacing Si (Suspicious removed) with p
until end;

Pseudo Code 3: SDA Pseudo-code

The Attack Injection Tool act like an attacker by generating attacks. These attacks are injected into the network. The attacks are idenfied in the network server with the help of proposed SDA Model. The SDA Model is used as it produces more accurate identifation of the unexpected behaviour. In the following, we describe the attack from against the SDA model or protocol. Assume that $S_i$ is a susceptibility or malicious program. Then $S_i$ apply the attacks through potential channel, on j at random from {1, . . . ,i-1} and a new Sj+1 (Apply vulnerability through intruders) of its choice and $S_i$ uses its potential channel $\prod_{i=1}^{n} S_i$ Formally, Si collects a suggestion from the channel Sj as follows:

$$\mathbf{S_i} \rightarrow \mathbf{S_j} \quad : \prod \mathbf{s_i}, \{a_0, \ldots . a_{j-1}\}$$
$$\mathbf{S_j} \rightarrow \mathbf{S_{j+1}} : \prod \mathbf{s_i}, \{a_0, \ldots . a_j\}$$
$$\mathbf{S_{j+1}} \rightarrow \mathbf{S_i} \quad : \prod \mathbf{s_i}, \{a_0, \ldots . a_{j+1}\}$$

However, there are various devices accessible that can check a system for known susceptibilities. In any case, such tools consider vulnerabilities in confinement, free of each other. Regrettably, the interdependency of susceptibilities and the network of systems make such investigation restricted. While a single susceptibility may not seem to represent a noteworthy risk, a blend of such

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 6, June 2015*

susceptibilities may permit attackers to reach discriminating network resources [14].

## V. CONCLUSION

This paper exhibits a model that makes attacks on network-servers and decides security susceptibilities. To exhibit this model application is developed that proceeds protocol specification certainties from server and performs different attacks on the server and finds susceptibilities. The found susceptibilities are then preceded into database for more strides to alter the bugs in the product in which susceptibilities are found. Despite the fact that few points of interest are accessible about the susceptibilities subsequently they were originate in closed source programs, it was conceivable to conclude that three of the blemishes were identified with resource management. Analysis and methodology shows that the injection of susceptibilities and attacks is undeniably an effective way to evaluate security mechanisms and to point out not only their weaknesses but also ways for their improvement.

## REFERENCES

[1] Antunes J, Nuno Neves, Miguel Correia, Paulo Verissimo, and Rui Neves (2010)
"Vulnerability Discovery with Attack Injection" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 36.
[2] P. Verissimo, N. Neves, C. Cachin, J. Poritz, D. Powell, Y.Deswarte, R. Stroud, and I. Welch, "Intrusion-Tolerant Middleware: The Road to Automatic Security," IEEE Security and Privacy,vol. 4, no. 4, pp. 54-62, July/Aug. 1996.
[3] B. Beizer, Software Testing Techniques, second ed. Van Nostrand Reinhold, 1990.
[4] N. Neves, J. Antunes, M. Correia, P. Verissimo, and R. Neves,"Using Attack Injection to Discover New Vulnerabilities," Proc.Int'l Conf. Dependable Systems and Networks, June 2006.
[5] J. Myers and M. Rose, "Post Office Protocol—Version 3," RFC 1939 (Standard), updated by RFCs 1957, 2449, http://www.ietf.org/rfc/rfc1939.txt, May 1996.
[6] M. Crispin, "Internet Message Access Protocol—Version 4rev1,"Internet Eng. Task Force, RFC 3501, Mar. 2003.
[7] Charles M. Kozierok (2005) The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference, No Starch Press; 1st edition, ISBN-10: 159327047X, ISBN-13:978-1593270476.
[8] A. Adelsbach, D. Alessandri, C. Cachin, S. Creese, Y. Deswarte, K. Kursawe, J. C. Laprie, D. Powell, B. Randell, J. Riordan, P. Ryan, W. Simmonds, R. Stroud, P. Ver´_ssimo, M. Waidner, and A. Wespi. *Conceptual Model and Architecture of MAFTIA.Project MAFTIA deliverable D21*. Jan. 2002. http://www.research.ec.org/maftia/deliverables/D21.pdf.
[9] "Identifying Vulnerabilities and Risks on Your Network" [Online] Available: https://techsoupforlibraries.org/cookbook-3/networking-and-security/identifying-vulnerabilities-and-risks-on-your-network [Last retrieved 20 May 2015]
[10] J. Stanley (2015) "Computer Network Server**"** [Online] Available: http://www.wisegeek.com/what-is-a-network-server.htm [Last retrieved 20 May 2015]
[11] [Online] Available: http://www.tcpipguide.com/free/t_POP3 Gener alOperationClientSer verCommunicatio nandSe-2.htm#Figure_306 [Last retrieved 22 May 2015]
[12] [Online] Available: http://www.mactech.com/articles/mactech/ Vol.12/12.11/PPSMTP Client/index.html_[Last retrieved 26 May 2015]
[13] Dbohdan (2015) "Simple Mail Transfer Protocol" [Online] Available: http://wiki.tcl.tk/1256 [Last retrieved 26 May 2015]
[14] Sushil Jajodia, Steven Noel, Brian O'Berry (2004) "Topological analysis of network attack vulnerability".
[15] Found Stone Inc. Found Stone Enterprise, 2005. [Online] Available: http://www.foundstone.com.
[16] K.Goswami, R.Iyer, and L.Young. Depend: A simulation based environment for system level dependability analysis. IEEE Trans. on Computers,46(1):60–74, Jan.1997.
[17] Saint Corp., "SAINT Network Vulnerability Scanner," [Online] Available: http://www.saintcorporation.com.
[18] Qualys, Inc., "QualysGuard Enterprise," [Online] Available: http://www.qualys.com.
[19] D. Wagner, J.S. Foster, E.A. Brewer, and A. Aiken (2000), "A First StepTowards Automated Detection of Buffer Overrun Vulnerabilities," Proc. Network and Distributed System Security Symp.
[20] P.C.Cheng (2001) "An architecture for the Internet Key Exchange Protocol" IBM SYSTEMS JOURNAL, VOL 40, NO 3, 2001
[21] Ari Muittari (2004) "Internet Key Exchange (IKE) protocol vulnerability risks"
[22] E. Haugh and M. Bishop (2003), "Testing C Programs for Buffer Overflow Vulnerabilities," Proc. Symp. Networked and Distributed System Security, pp. 123-130.
[23] Byeong-Ho Kang, Maricel O. Balitanas (2009) "Vulnerabilities of VPN using IPSec and Defensive Measures" International Journal of Advanced Science and Technology Volume 8, July, 2009
[23] J. Dura˜es, H. Madeira (2005), "A Methodology for the Automated Identification of Buffer Overflow Vulnerabilities in Executable Software without Source-Code," Proc. Second Latin-Am. Symp. Dependable Computing.