Internet Applications Security Testing: Manual VS Automatic Approach

Md. Shafiul Alam Chowdhury, Md. Emrul Hasan, Md. Shahadot Hossain

Abstract— Over the last few years there has been a significant increase in the use of Web Applications that deal with private information like social security numbers, account numbers, address, credit card numbers and passwords. Due to this increase, malicious hackers are making these web applications their target of attacks. Two approaches are used by different organizations to evaluate the security of their web applications. Some organizations use automated tools for finding vulnerabilities in their web applications while others do it manually by security professionals.

In this case study a comprehensive research has been carried out to investigate the results of both manual and automated approaches used for evaluation of web application security. The advantages and disadvantages of each approach are discussed and an attempt is made to suggest the best solution. Our suggested best solution combines the automated tools with the expertise of security professionals. Further we evaluate our suggested solution by developing a state of the art tool as a part of this study. The tool is named "Proxy Security Evaluation Tool" and the results obtained from the tool has been presented and analyzed. Conclusion and suggestions for future studies are also presented at the end.

Index Terms— Internet and worldwide Web, Open Web Application Security Project (OWASP), Web Haking Methodology, Proxy security evaluation tool, SQL, Manual Testing, Denial of Service Attack (DoS)

1 Introduction

Millions of people throughout the world use internet for various activities including financial transactions, purchasing/selling variety of goods and services, and performing research. These activities are carried out through web applications and with each transaction private information including names, social security numbers, phone numbers, account numbers, addresses, credit card numbers and passwords are transferred from one place to another and are also stored in various locations. This information must be protected from unauthorized access. As companies are rushing towards web applications for selling items to users, malicious hackers are likewise rushing to find out vulnerabilities in these web applications. The increase in the number of attacks on web sites is negatively impacting the companies trying to do business over the web. The target of attacks on web applications is to steal the sensitive information (business secrets, credit card numbers, passwords etc) that are stored in applications and the associated databases with them. Today network security has matured and has more protection against web hackers. Thus hackers are looking for an easy way to penetrate into systems and hence their target is web applications through port 80. Thus the use of firewalls or SSL can not protect against the attacks on web applications.

According to Gartner Inc. "Over 75% of attacks are occurring due to web applications". One of the major reasons for this tremendous increase in web application attacks is the vulnerabilities which exist in these applications. This tremendous increase in web applications vulnerabilities and their exploitation is forcing many organizations to evaluate the security of their existing web applications. The area of web application security has gained more focus in the recent years. Two of the well-known groups formed for this purpose are:

1.1 Web Application Security Consortium:

It is an international group of experts, industry practitioners and organizational representatives that produces open source and best practices security standards for World Wide Web.

1.2 Open Web Application Security Project (OWASP)

This group is dedicated towards finding and fighting the causes of insecure software. It produces free, unbiased, opensource documentation, tools and standards in the area of web application security. Most organizations use penetration testing for evaluating the security of their web applications. Some organizations use manual approach while others rely on automated tools. Both of these approaches have advantages and disadvantages. Manual testing is time consuming and requires

- *Md. Shafiul Alam Chowdhury, Assistant professor & Chairman, Department of Computer Science & Engineering, Uttara University, Bangladesh, Email : shafiul.a.chowdhury@gmail.com*
- *Md. Emrul Hasan, Senior Lecturer, Department of Computer Science & Engineering, Uttara University, Bangladesh, Email:emrul87@gmail.com*
- *Md. Shahadot Hossain, Assistant Professor, Department of Computer Science & Engineering, Uttara University, Bangladesh. Email:shahadot_bdm@yahoo.com*

a lot of effort from security personnel performing the tests.

In contrast automated tools are very fast, efficient and cost effective. But most of these tools are efficient only in finding some specific set of technical vulnerabilities like SQL Injection or cross site scripting. Another major problem with these tools is their inability to maintain a session with web server/application while performing the tests, resulting in a different (sometime invalid) state of the application for the following tests. There has been a significant development in the automated tools for testing web applications security. But still the area is emerging and no single tool can address all problems related to web application security. It has been historically proven that neither automated tools nor manual testing alone can spot all kinds of vulnerabilities in web applications.

In order to provide a comprehensive solution for detecting web application vulnerabilities, a combination of manual and automated testing approach is needed. In this way security personnel are equipped with a tool that can reduce much of their work load. Using this combined approach can address most of the problems that exist when automatic tools are used alone. Also it can facilitate personnel in testing the security of a vast array of web applications in significantly reduced amount of time.

2 PROBLEM

There is lack of a comprehensive solution that can spot all kinds of vulnerabilities in a web application in a quick, efficient and cost effective manner. Automatic tools for assessment of web application security alone cannot efficiently find out all potential vulnerabilities and often results in false positives. Also most of the available tools (whether commercial or free) cannot maintain a session with the web server, while the tests are in progress. Manual approach that requires a human to traverse the entire site is time consuming and introduces human errors.

3 GOAL

The goal of this case study is to investigate the results of combining this manual and automated approach by developing a semi automated proxy security evaluation tool that automates the security testing of web applications and at the same time give control of the testing process to the test performer. The tool is also expected to maintain a session with the webserver/web application while the tests are in progress. This semi-automated proxy security evaluation tool with the help of a security analyst is expected to eliminate the problems that can result by using automated or manual approach alone.

4 AUDIENCE

The proxy security evaluation tool is primarily intended for security consultants/analysts working with web application security; however web application developers can also benefit from it by improving their code to eliminate the vulnerabilities that the tool finds in their web applications.

5 LIMITATIONS

The limitations Listed:

- Proxy security evaluation tool will work only with the Internet Protocol HTTP/HTTPS.
- The platform for the proxy security evaluation tool will be Windows.
- Automatic Web Crawling function will not be implemented from scratch. In case it is needed, some open source web crawler will be embedded in the tool.
- The Proxy Security Evaluation Tool (PSET) will not have any security functions itself.

6 RESEARCHMETHODOLOGY

The overall approach followed for carrying out the research is divided into the three phases; Research type, Literature Review, Development of Proxy Security Evaluation Tool (PSET).

6.1 Type of Research

A research method can be divided in to two categories:

- Inductive
- Deductive

In deductive approach the researcher has an idea or a guess (hypothesis) of the solution to the problem. A hypothesis is in the form of "If X then Y. X is the independent variable that is manipulated to see how Y the dependent variable reacts." i.e the hypothesis is empirically tested using experiments and the solution is analyzed. While an inductive approach starts with the collection of empirical data as the researcher has no idea/guess of the solution to the problem. From empirical data conclusions are drawn based on the methodological and systematic analysis.

A research can also be classified as:

- Qualitative
- Quantitative

Qualitative research is descriptive and inductive where the researcher is concerned about the process primarily rather than the outcomes or products. In this case the researcher makes abstractions, concepts, hypothesis and theories from the details. On the other hand a quantitative approach begins with hypothesis and theories. It deals with prediction, collection and presentation of data, experimentation and component analysis. Our research is based on deductive and quantitative

approach because we have a hypothesis that the combination of manual and automatic tools for web application security testing will solve the problem.

6.2 Literature Review

The first phase of the study is carrying out an intensive literature review in order to understand the common techniques for hacking web applications as well as the assessment tools for the security of these applications. The study begins with the understanding of the general approach taken by web hackers for attacking a web application. The study moves from general web hacking methodology to the in depth analysis of specific techniques like SQL Injection, Cross site scripting etc used for attacking web applications. Following the analysis of common web hacking methods, the manual and automated testing approach for web applications is compared. Next some well known available tools (both free and commercial) for web application assessments are studied and tested. Following the analysis of well known tools for web applications security, the two approaches (manual and automatic) for testing web application security are evaluated and the best of both approaches is presented. The sources of study consist of latest books, magazines, articles, publications in the area of web application security.

6.3 Development of Proxy Security Evaluation Tool

The important and crucial phase of the study is the actual implementation of proxy security evaluation tool that can fit in the combined approach for web application security testing. The proxy security evaluation tool will be developed using the rapid prototyping strategy. Prototyping is a technique in which the system is developed in small iterations. Each iteration adds a set of features to the over all system. For developing the proxy security evaluation tool, first a prototype is developed with the basic features that can work as a simple HTTP proxy.

Next the prototype is tested for its functionality based on the chosen features. If any feature of the prototype is not functional, the prototype is modified until it works properly. An iteration is completed once the developed prototype has all the features defined at the start of the iteration. The next iteration begins by defining additional features for the tool. The features are selected on the basis of the combining the advantages of automatic and manual approach for web application security. While selecting the features for the proxy security evaluation tool, the input is also taken from the analysis of well known available tools. This process continues until the tool is enhanced with all the features that can help the security analyst to successfully use it for web application security testing.

6.4 Analyses of Results

The last and the most important step of the study is to analyze the results achieved during the testing process of the proxy security evaluation tool. The tests are analyzed thoroughly to draw any conclusions about the achievement of the study goal and solution of the problem. In this phase of the study, the result of combining manual and automated testing approaches for web application security is evaluated. Any problems or discrepancies found during the analysis are discussed here and the recommended future research is described here.

7 Web Hacking Methodology

I Hackers use variety of approaches for attacking a web application. In General, Web hacking methodology includes the following steps:

7.1 Profile the Infrastructure

The first step in web hacking is to get information about the target web infrastructure. It is helpful prior to the attack to get Information about the transport used, types and number of web servers, ports used for running web servers, load balancer etc.

7.2 Attack the Web Server

After getting knowledge about infrastructure, next step is to find out vulnerabilities in web server and try to exploit them. Once the type of web server is known, it is easy to find out the vulnerabilities that may exist in the web server. The ongoing research on web server's vulnerabilities helps the attackers to exploit them and take control over the web server.

7.3 Survey the Application

If no vulnerabilities are found in the Web server, hackers then shift towards the Web application. Careful examination of the web application is necessary for hackers to proceed.

They look for information about application technologies deployed (Java, ASP, CGI etc), directory structure, types of authentication, restricted contents in the website, nature of databases (backends) etc. This helps the attacker to make a complete picture of the contents, components and flow of the website. Manual inspection along with some automated tools like "lynx" and "Wget" for gathering this information and documenting the application structure can be used.

7.4 Authentication Mechanism

Once the type of authentication mechanisms and restricted contents are found, next step is to get access to those restricted contents through bypassing or breaking the authentication mechanism. Attackers can use different techniques including password guessing, stealing session IDs etc to bypass or break authentication mechanisms. Unvalidated inputs can be a great advantage for applying attacks like SQL Injection to break the authentication mechanism.

7.5 Authorization Scheme

Authorization comes after a user is authenticated. Attackers use various techniques to get unauthorized access to files and other objects. Some of the common methods used to bypass authorization scheme are directory traversal, changing user principle, requesting hidden objects, escalating privileges and executing SQL commands etc. Typically input fields related to userid, username, group access, file names, cost etc in HTTP requests are modified to perform this attack. The attacker needs to know how session management is handled by the application. The components responsible for tracking the users identify and roles should be identified. Important areas to look for are Profiles, Shopping Carts, Checkout and change password forms.

7.6 Functional Analysis

To apply a successful attack, it is necessary to carefully analyze each function of the web application. For example components responsible for customer's order input or confirmation should be identified and checked for fault injection (input validation). By understanding each component of the web application, an attacker can apply more targeted attacks and sometime can uncover very easy ways to hack the application.

7.7 Exploit Data connectivity

Almost every web application use one or more databases on backend with which the web application interact. Most web applications accept user inputs and use them in the commands for retrieving data from the database. If these inputs are not properly validated, an attacker can insert commands instead of normal data in the input fields and can execute those commands on the database server. This type of attack is known as SQL Injection that not only enables the attacker to get unauthorized access to information but also execute dangerous operating system commands through database server.

7.8 Attack Management Interface

Attacking management interface is a different method than the above discussed methods, as in this case the entry point of an attacker is different. Most web application has support for remote web application administration. This support for remote web application administration allows the administrators to maintain servers, contents and back-end databases remotely. For this purpose a port is kept open on the server which is a point of interest for attackers.

7.9 Attack the Client

Web applications are different from self-contained applications where all the code is stored locally and execution is performed in a closed environment. In web applications one part of source code can reside on client side while other part can reside on server side. Attackers in most cases target the client side, as it is easy to attack, especially when the input validation is performed only on client side and there is no cross checking at the server side. Attacks like Cross site scripting, session hijacking etc are most commonly used by attackers.

7.10 Denial of Service Attack

When the attacker fails to compromise the system in any of the above mentioned ways, he tries to launch a Denial of Service (DoS) in order to affect the availability of the system. Attackers think that if they cannot get into the system, they will not let others to use the web application. In DoS, large set of requests are sent to the web site by the attacker which denies the legitimate user's requests.

7.11 Remove Logs and leave

If the attacker succeeds to enter the system, he does his job (copying/deleting/overwriting directories and files etc) and then removes all the system logs from which he can be traced. After removing the logs he leaves the system.

8 Web Application Attack Methods

Most companies are now moving towards the web for delivering their services, this has caused the attackers to pay most attention to identify and exploit vulnerabilities in web applications.

8.1 Unvalidated Input and Client Side Validation

Attacking client is considered an easy attack by web hackers. There are two reasons: one is that the application may accept input from users without validation, and the other is that input validation is usually performed only on client side. Web applications are different from the ordinary self contained applications where all code is stored on the client side. In web application all inputs/interactions are stored either on server side or on client side. Storing the source code (especially that portion of code which is related to validation of user inputs) or other inputs on client side can make the life of the attackers easy. Attackers can review the client side code and find vulnerabilities in it.

Many web applications allow users to make choices about different things. For example choosing between Jobs, Gender, types of Credit Cards, Country names etc. This facility is usually provided in the form of user interface controls which consist of textboxes, combo boxes, radio buttons, list boxes etc. Appropriate choice of the interface control element is important from the developer's perspective. These control elements vary depending on the restrictions that they can impose. For example a text box control is less restrictive as compared to a combo box control. A user can enter anything in the text box control while the combo box control presents some limited set of options to the user from which the selection is to be made. The use of these controls varies depending on the situation, for example if the user is asked to enter their credit card number then a text box must be provided for this purpose not a combo box. In case of the textbox if the application directly accepts user input without checking for its validity then it is called Unvalidated input and can be used by attackers to launch various attacks. Lets say

that the developer wants to put some validation mechanism to assure that the user does not enter any illegal values in the textbox, for example in case of credit card numbers the user must be restricted to enter digits only. Further the number of digits entered can be restricted too. But how to do it? This can be done by providing validation functions that check the values entered by the user for their validity. Now the question is that where these validation functions should be placed? They can be placed either on the client side or server side or on both sides. The biggest mistake most developers make is that they place the validation functions only on the client side and think that they have validated the inputs properly. But they are wrong because client side validation can be easily bypassed using a proxy like Paros or Webscarab, leaving the web application without any protection against malicious parameters.

First step in this attack is to identify all input vectors in the web application and carefully analyze input restrictions that each user interface control implements. Next an attempt is made to bypass the restrictions and see if it can be used to exploit the web application or not. This can be done by two ways: either modifying the source code of the page or intercepting/modifying the request sent by user for the page.

Consider a simple example where the user is presented with a Request for Loan form on a website. There is a text box on the form for entering the Loan Amount. The policy is that a user can request not more than 99999$. The developer has implemented this policy by validating input on the client side which allows the user to enter only 5 digits in the text box.

The HTML code for this restriction is

<input name= "LoanAmount" maxlength=5>

The developer has not put any check on the server side for the amount entered. Now users cannot submit a request for more than 99999$ loan. But what if the user just check the source code, find the max length attribute, changes the value from 5 to 7, save the changes and then resubmits the form with a request for 1000000$. It will be accepted immediately as there is no check on the server side for the amount and the user will be granted 1000000$ loan. The same thing can be done by intercepting the HTTP requests on the fly, using a proxy and modifying the maxlength of the parameter and then submitting the modified request to the server. This flaw can be corrected by putting some validation mechanism at server side.

Similarly other validation mechanisms can be enforced using client-side scripting languages for example JavaScript, VBScript etc. These languages can be used to perform a variety of useful things but their most common use is validating the input as the user enters data. Client side scripts are based on an event driven model, i.e. the scripts are run when the user clicks on submit button or move the curser over some control etc. Client side validation is useful because the user come to know about the error messages very quickly and can correct it without waiting for sending the requests to server and then waiting for the response. Thus client side validation is used for performance and usability but it has no security benefit. Therefore server side checking is required for security purposes. Once server side checks are in place, the client side checks can be provided for immediate response to the legitimate users and reducing the amount of invalid traffic to the server. An important thing to remember is "Never trust client side data".

Huge number of attacks can be avoided by validating input parameters at appropriate place. The developer must validate input parameters against a "positive" specification that defines:

- Data type (string, integer etc)
- Allowed character set
- Min and max length
- Whether null is allowed
- Whether the parameter is required or not
- Whether duplicates are allowed
- Numeric range
- Specific legal values (enumeration)
- Specific patterns (regular expressions)

8.2 State based Attacks

Web is stateless; it does not have any mechanism to remember which page a user browsed previously. Each page is presented to the user without any prior knowledge of previous pages or restrictions where they can go next. The property of statelessness does not effect static applications but when it comes to dynamic applications; it can lead to enormous errors and security violations. Consider a web application for online shopping which maintains no state information and you can go to the previous page where the credit card number was entered. If you can jump directly to the page where the receipt was displayed, you can shop extra items without paying for them.

Web application developer is responsible for maintaining state in the application and enforcing restrictions where page access is important. The growing trend of dynamic websites and shopping carts, purchase history, shipment tracking features require some state to be made available to the web application.

8.3 SQL Injection

When a web application accept inputs from users and pass that input to the database or CGI processor without ensuring that the data is valid, the attacker can use SQL commands in the input to extract unauthorized information, read/write or delete files on the system and execute arbitrary SQL commands by the backend database. This vulnerability is called SQL Injection. Programmers often chain together SQL queries with input parameters provided by user, which give chance to attackers to embed SQL commands inside these parameters. As a result an attacker is able to execute arbitrary SQL queries/commands on the backend database via web application.

8.4 Directory Traversal

All web applications are stored at some central location (server). Static and dynamic web pages that are stored at the server are presented to clients by web servers (along with Web applications). The users visiting the server should be restricted to only those pages for which they are allowed to view. In directory traversal attack a malicious user determine the location of restricted files and tries to view/execute them. For example an attempt may be made to view the local password file or modify the contents of the website by executing some files.

8.5 Buffer Overflow

In buffer overflow attack an attempt is made to manipulate the execution stack by sending a large amount of data to the application. An attacker can cause the application to execute arbitrary code and may take over the machine.

According to CERT, "Buffer overflow accounted for more than 50% of all major security bugs". Some of the well-known internet worms Code Red, Slapper and Slammer also used buffer overflow for their execution.

9 Web Application Security Testing

The Web application vulnerabilities are increasing day by day and hackers are trying to put more effort in this field. Therefore it is necessary to check your web applications for security issues in order to find out any holes in them which can be exploited. This requires testing your web application from a hacker's perspective. It can be accomplished through Black box testing. In black box testing the internal structure (source code) of the application is not known to the tester and the application is considered as a black box accepting inputs, processing them and producing outputs. Generally speaking two approaches are used for Penetration testing of web applications to evaluate their security. One is manual and other is automatic. The following section discusses the advantages and disadvantages of each approach.

9.1 Manual Testing

In manual testing, security of a web application is tested by a security professional(s). Initially manual approach may look cheaper but when the amount of work increases, the associated cost also increases. Manual approach may be good for small web applications but not for dynamic web applications that consists of more than 500 pages. This lack of scaling property makes these tests inefficient. Beside this, manual approach often takes longer time. Imagine a security analyst who is analyzing a web application which consists of more than 400 pages and each page contains approximately 100 parameters (input fields). Checking for only single vulnerability like SQL Injection on all of these input fields will take a huge amount of time. Manual approach is also labour intensive which means that it requires a lot of effort from security professional performing the test. Another disadvantage of manual testing is that it is highly dependent on the security experts who perform the test, thus when they leave the organization for whatever reason, the organization losses the expertise in this area. If the security tester is an expert in the area and possesses the abilities and expertise to deal with a variety of circumstances, the manual approach can be very advantageous in that case. Security experts can uncover the logical vulnerabilities which an automated tool might miss. For example an unauthorized funds transfer might not be detected by automated tool but a highly qualified security analyst can identify it. This property of manual testing approach minimizes the number of false positive results that are often generated by automatic tools. As web applications are unique. This uniqueness and evolving nature of web applications can be carefully analyzed only by human, not an automated tool. A tool may be good for one web application but might not achieve best result for another web application's structure.

9.2 Automatic Testing

An alternative of manual testing is to use automated testing tool for evaluating the security of a web application. Many organizations are now following this approach in order to minimize the dependency on humans. These automated tools are sometime called attack simulators since they replicate the actions of an attacker. The main aim of these tools is to find out vulnerabilities in the web application and report them. Automatic testing tools are very fast as compared to manual testing approach because they significantly reduce the time of assessment. More than one web application can be tested with them in a very short time. Generally they are cheaper than the price paid to the security professionals for their time and effort. Thus an organization acquires a security tool and can run it from time to time against a target web site to find vulnerabilities. Besides the advantages, automated tools also have some shortcomings;

- These tools can find out only those vulnerabilities for which they are coded. It always takes sometime before a new exploit is discovered and added to the list of automated probes. It works like an antivirus program that relies on signature .data files.
- Automated cannot uncover logical flaws in a web application. For example an unauthorized funds transfer from a bank account or user impersonation might not be detected by automated tool.
- Automated tools may create many false positives because they are not as flexible and efficient as a security professional.
- To test full functionality of a web application it is necessary to log in to the application. It becomes a challenge for an automated tool to log in to the web application automatically because the log in process is different for different application, for instance some require SSL, multiple pieces of authentication infor-

mation beyond the simple username and password and multiple re-directs.

- An automated tool may not be able to detect when it is logged out from the application due to some reason, for instance timing out, application errors or session expiration etc.

- Creating an accurate structural map for a complex web application in a reasonable amount of time is not an easy task to be performed automatically.

- Almost all automated tools rely on errors and response codes for finding vulnerabilities but application developers often customize or even remove these error messages. Therefore a tool may not be able to determine the meaning of an error message which results in a false positive.

- Various strange URL structure exists these days. It is difficult for an automated tool to find the web application name, parameter names and their associated values when there is no question mark and no delimiter like "&" signs in a URL or there is some strange file extension. For example; /hue/sbc/b4in345/rfe=234_3/07~3454/owen/pid=234454/s=home/search/view/main_id/23435/

- Traversing a website for an automated tool extremely tough when the website has client side generated links which are created at run time by menus and style sheets.

9.3 The best of both (Combination of software and security personnel)

An automated tool for web application security testing can alleviate a huge work load but the tool alone cannot be expected to give 100% accurate results. Automated tools can identify most of the technical vulnerabilities but they fail to spot logical vulnerabilities while manual approach being good in identifying logical vulnerabilities is not efficient to find out the technical ones or even the logical. It has been historically proven that neither automated testing approach nor manual testing alone can spot all kinds of vulnerabilities in web applications. One of the research conducted by Watchfire on 100 websites, showed that in 17% of the websites humans identified those vulnerabilities that were left by scanners and in 36% of websites human identified zero vulnerabilities beyond the scanner while in 47% both produced complementary results. According to James Spooner, managing director of security consultancy, Lodoga: " if you take some of the best of breed commercial tools and ran them blindly, pressed all the quick assessment buttons, you would actually end up with some pretty bad results because what's important is that the exploration of a website is done properly so that you haven't created false results about what the website contains, so when the tool runs its assessment against those results it will believe that it is inducing errors which you assume are vulnerabilities. This is why the human bit is so important it involves a combination of manual skills and tools."

According to Michael Gavin Senior analyst, Forrester Research Inc: "Fancy tools aren't enough. Automated testing tools can't replace smart QA people. Just as attackers use tools and their own expertise, you need to combine tools and expertise to fight them" . On another occasion he said that: "If you just use a tool, you're only as secure or as good as the tool is". He also said that "Alot of the automated tools do fuzzing, which is basically throwing random junk into input fields and seeing what comes back. But what we haven't seen from most is the ability to customize the fuzzingm, and even if the tool can be customized, testers need some knowledge about what the tool is doing and how the software is reacting to it. That enables you to perform a more complete test." And, he added, "If the tool isn't doing a good job fuzzing, or it just stock tests that don't break the application, then you will get false sense of security. Automation is great, but you need to apply extra knowledge."

Therefore the best approach towards the assessment of web application security is to use a combination of software and security. In this approach, a highly qualified security professional tests the security of a web application with the help of software tools. Using a combination of software and security personnel can result in the following benefits.

- All technical and logical security issues can be identified. Technical vulnerabilities can be efficiently identified by automated tool while the logical vulnerabilities which could be missed by the software, can be identified by security professional by careful analysis.

- Volume of false positives can be reduced. Software can apply various tests on a web application and customized error messages and response codes generated can better be analyzed by security professionals. This will reduce the number of false positive results which could occur if the software is used alone.

- The unique and evolving nature of websites require a human to select suitable tests for a particular web site, to be applied by the software. In this way large websites can be tested quickly and efficiently.

- A logged in state can be maintained using a combination of software and security personnel. If the software is logged out at some point, security personnel can detect it and log in again before the software proceed to next test.

- Software tools can remotely scan without source code accessibility. They can quickly crawl through a website and find out all the links associated with that domain. Human interaction will enhance this process to carefully map the website and remove the bad links.

- Security personnel can also apply the tests to find out recently discovered vulnerabilities which may not be hard coded in the software version released.

- Security personnel can help the software to recognize the strange URL structure which may not be coded in the software.

10 Analysis of Results

In this research work we analyze the results obtained from our developed software Proxy Security Evaluation Tool. Therefore once the system was successfully developed, we tested it on different web sites and collected the data. For finding vulnerabilities, we have used a sample Web Server (and web application) called WebGoat. WebGoat is a project of Open Web Applicatoin Security Project (OWASP) that aims at the learning of Web applications vulnerabilities.

11 Conclution

A Web application security has become an important issue. New vulnerabilities are being discovered in these applications which are threatening the companies doing business over the internet. There has been a significant development in the testing of web applications for finding vulnerabilities in them. Various tools have been developed for this purpose and companies are using these tools. In this study we looked at the latest vulnerabilities that may exist in today's web applications and the methods to identify them. We have thoroughly analyzed most of the freely available tools for testing web application security and also some commercial ones.

The analysis of these tools showed that they cannot make a comprehensive evaluation of web applications security and often produce false positives. These tools are only good for a limited number of tests and also most of these tools cannot withhold a session while the tests are in progress. We also looked at the manual approach in which a security professional thoroughly analyzes a web application for security measures without the help of any software tool. This manual method also cannot uncover all the vulnerabilities and is time consuming as well as labour intensive.

We state that the best approach towards the web application security is combination of both manual and automated approach. Therefore we developed such a tool that has the flexibility to be adapted to the unique nature of a web application by someone who has expertise in the area of web application security. The developed tool was named Proxy Security Evaluation Tool. The developed tool was tested against some sample web sites and the results were analyzed. We were able to solve the session breaking problem by replaying the previous HTTP requests automatically before applying the next test patterns or next tests. The results obtained from the tool showed that this combined approach of using a tool with human expert is a better solution than using any one of them separately.

12 Future Study

There are some issues that can be solved by further investigation and improvement in the developed Proxy Security Evaluation Tool.

13 Acknowledgement

Thanks to Dr. Christer Magnusson, Mikeal Simovits and Anders Knuttsson, for their valuable supports throughout this research work.

14 References

[1] U5 Security Myths
http://www.varbusiness.com/showArticle.jhtml?articleID=22104030&flatPage=true; April 2006
http://www.gartner.com/; April 2006
http://www.webappsec.org/projects/whid/statistics.shtml; April 2006

[2] 9 Ways to hack a web application by Martin Nystrom
http://developers.sun.com/learning/javaoneonline/2005/webtier/TS-5935.html; April 2006
http://www.sans.org/; April 2006
http://www.owasp.org/; April 2006
http://www.webappsec.org/; April 2006

[3] Web application security: Automated scanning or Manual Penetration testing? By Danny Allan, Strategic Research Analyst Watchfire, available at
http://www.watchfire.com/news/whitepapers.aspx ; May 2006

[4] Vulnerability assessment tools, Network Security, July 2003 by Elspeth Wales
http://www.compseconline.com/hottopics/hottopic_Nov03/hottopic_Nov_03.html. May 2006
http://www.watchfire.com/products/appscan/powertools.aspx

[5] Trends 2006: Application Security by Micheal Gavin Forestor Inc.Tting
http://www.expresscomputeronline.com/20060306/management02.shtml, May 2006

[6] Want secure software? Think like an attacker By Colleen Frye
http://searchappsecurity.techtarget.com/originalContent/0,289142,sid92_gci1167677,00.html; May 2006

[7] Technology alone cannot defeat Web application attacks: Understanding technical vs.logical vulnerabilities by Jeremiah Grossman 05.23.2006, Available at
http://searchappsecurity.techtarget.com/tip/1,289483,sid92_gci1189767,00.html?bucket=ETA; May 2006

[8] Web Applications (Hacking Exposed) (Paperback) by Joel Scambray, Mike Shema

http://lynx.browser.org; April 2006
http://www.gnu.org/software/wget/wget.html; April 2006

[9] OWASP Top Ten Most Critical Web Application Security Vulnerabilities available at http://www.owasp.org/documentation/topten.html; April 2006

[10] How to Break Web Software by Mike Andrews, James A.Whittaker
http://www.imperva.com/application_defense_center/glossary/cookie_poisoning.html;April 2006
http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf; April 2006

[11] Testing Your Web Applications for Cross-Site Scripting Vulnerabilities by Chris Weber, Casaba Security, LLC Published: May 6, 2005
http://www.cert.org/; April 2006
http://www.imperva.com/application_defense_center/glossary/buffer_overflow.html;April 2006
http://www.ntobjectives.com/freeware/index.php; April 2006
http://www.ruby-lang.org/en/; date: 2006-04-20
http://www.owasp.org/software/webscarab.html; April 2006
http://www.systegra.com/; April 2006

[12] Web Hacking Attacks and Defense by Stuart McClure, Saumil Shah and Shreeraj Shah
http://www.mentalis.org/soft/projects/seclib/docs/; Dec 2006
http://www.owasp.org/index.php/OWASP_WebGoat_Project; Dec 2006.