Evaluation of tools for assessing Web applications

V. Suhina, M. Kozina and S. Groš
Department of Electronics, Microelectronics, Computer and Intelligent Systems
Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
Phone: (01) 612-99-35 Fax: (01) 617-00-07 E-mail: vanja.suhina@fer.hr

Summary – In this paper, we provide a selected list of tools that can be used for assessing Web application security. Intention is to present available tools that can help in search of vulnerabilities in Web applications and thus make them more secure. The paper is written for Web application developers and testers, and of course, students who have interest in web development. So, it is concentrated on free and/or open source tools, and thereby, commercial products are not mentioned. First part of paper provides overview of vulnerabilities that can be detected by using mentioned tools. In the second part, the tools are described and it is shown for which actions these tools can be used.

I. INTRODUCTION

Web applications have become common way for companies to conduct business with the outside world. The static web pages are gone and now, almost every company has its own dynamic, interactive Web application that communicates with their clients. Sometimes, these applications are written by programmers not familiarized with security problems behind their Web application front-end or their work is guided by hard to get deadlines so application security is not very high on priority list. Thus these Web applications become main threat to secrecy and integrity of company's sensitive data.

Web applications security incidents have significantly increased in past two years [1]. Most incidents were possible to happen due to very well known design flaws in applications. As it is shown in [2], some of the most common flaws are Cross Site Scripting (XSS), Injection flaws, Malicious File Execution, Insecure Direct Object Reference and Cross Site Request Forgery (CSRF). Last year, top four types of vulnerabilities, XSS, SQL Injection, Remote File Inclusion and Buffer Overflow, were responsible for more than 50% of all Common Vulnerabilities and Exposures (CVE's) [3].

There are various tools for Web application security assessment, both open and closed source. In this paper are listed and evaluated those that are found valuable and mature while in the same time being freely available, i.e. open source. Description of their features and architecture is also given.

The paper is structured as follows. In the second section overview of the most common vulnerabilities is given. Then, in section 3, description of each tool is given. The paper finishes with conclusion in section 4, and list of references in section 5.

II. OVERVIEW OF VULNERABILITIES

To aid understanding of the description of tools given in the following section, as well as to emphasize what should be taken care of during design and development of Web applications, a short description of most common techniques and vulnerabilities used to compromise Web applications follows.

Session Prediction or Session Hijacking is a method of impersonating a Web application user by deducting or guessing the unique value used to identify a user session. As a consequence of this vulnerability, attacker gains the ability to issue web site request with hijacked user's privileges [4]. This method is directed to objects used to bind user actions into a single stateful session by web applications. Most common objects are cookies and URL parameters.

Cross Site Scripting (XSS) vulnerability occurs when an attacker uses a Web application to send malicious executable code to another Web application user. Web application simply takes attacker supplied data and sends it to user's web browser without validating data. Executable code is often used to compromise confidential information or even hijack user sessions [4].

Directory indexing and information leakages are vulnerabilities that reveal sensitive data, such as developer comments, error messages, secret files and other contents. As a result of this vulnerabilities, an attacker gains additional information and guidance in exploitation of a Web application [4].

Insufficient Process Validation vulnerability occurs when a logical flow of a Web application is bypassed or circumvented by user - attacker. If the user state through a session is not verified, the Web application could be exploited [4].

SQL injection attacks are used to compromise Web applications that construct SQL statements from user input. Before a user supplied input is used in a Web application processes, it must be adequately validated and checked for special characters that can be used to construct a custom SQL statement. This attack can lead to information leakage or taking full control of a database by an attacker [4].

HTTP response splitting attack occur when the data enters a Web application through an untrusted source, most frequently an HTTP request, and then, it is included in an HTTP response header sent to a web user without being validated for malicious characters. It can lead to web cache poisoning, cross user defacement, hijacking pages and Cross Site Scripting [5].

A Brute Force attack is an automated process of trial and error used to guess user credentials [4].

III. TOOLS

In this section description and evaluation of several free Web application assessment tools is given. These tools are:

- WebScarab
- Pantera
- ATK

- Gamja
- Burp Suite
- Wapiti

While examining tools, focus has been on their architecture, functionality, analyzing and testing abilities. In the case when a tool is modularized, most important modules and their functionalities are described.

1. WebScarab

The aim of WebScarab is to provide a free tool for Web application developers and reviewers that will help them in better understanding of Web applications functionality and in identification of the possible vulnerabilities that could compromise or bring down those Web applications.

OWASP (Open Web Application Security Project) WebScarab is an open-source framework written in Java used for analyzing and testing Web applications.

Common use of WebScarab is as an intercepting proxy. Proxy allows operator to review and modify requests created by a user using web browser before those requests are sent to a server. It can also review and modify responses sent from a server, before they are received by user's browser. Intercepting proxy applies both to HTTP and HTTPS communication. Every conversation which passes through WebScarab is added to a communication list, so it can be reviewed and additionally analyzed.

WebScarab is based on plugin architecture. There are two types of plugins: (i) plugins that create conversation, and (ii) plugins that analyze conversations. Plugins that create conversations use specific methods to decide which resources to request from the server, how to parameterize the request headers, and then submits the request to the server. After receiving the response from a server, they can perform some computation on the responses, and based on computation result decide whether or not to submit that conversation to the framework. When conversation is submitted to the framework, it is distributed to other plugins that can do further analysis. Current version of WebScarab offers these plugins:

- Proxy
- Manual Request
- Spider
- SessionID Analysis
- Scripted
- Fragments
- Compare
- Fuzzer
- Search

Proxy plugins observe traffic between the browser and the web server. The WebScarab proxy is able to observe both HTTP, and encrypted HTTPS traffic by negotiating an SSL connection between WebScarab and the browser instead of simply connecting the browser to the server and allowing an encrypted stream to pass through it. Various additional proxy plugins have been developed to allow the operator control over requests and responses that pass through the proxy [6]. The *Manual Intercept* is a special proxy plugin which allows the user to intercept requests from the browser and responses from the server, inspect them, and optionally make modifications before transmission. This is particularly useful when one wants to submit a form to a web server, but JavaScript validation rejects the values entered into the form. The request can be altered after the validation has been performed.

The *Manual Request* plugin allows the user to handcraft a request to be sent to the server. It is also possible to replay a previous request by selecting it from the drop down selection box. Previous requests which are loaded into the editor can also be edited before being sent to the server. When the "Fetch Response" button is selected, WebScarab sends the request to the appropriate server, and saves the conversation for analysis by the other WebScarab plugins. It can also be used for getting cookies relevant to the requested URL from the "Shared Cookies" list and use them in future requests [7].

The *Spider* plugin analyses responses to identify any links in the response body or the "Location" header. If the URL represented has not been seen, the URL is added to a tree, and can be automatically downloaded when desired. WebScarab has two modes of fetching unseen links. "Fetch Tree" enumerates all currently unknown links below the selected node, and queues them for retrieval. "Fetch Selection" queues only the selected nodes for retrieval [7].

SessionID Analysis plugin collects and analyses a number of cookies or URL-based parameters to visually determine the degree of randomness and unpredictability. WebScarab tags all sessionids with the date and time when they were collected, and then, after performing calculations on the string value of the sessionid in order to convert it into a number, plots the value against time on a graph. The human eye is a lot more efficient at identifying patterns than a computer, so by plotting the values it makes it easy for humans to visualize the sequence [7].

Scripted plugin is intended to give users the ability to create test scripts by using special BSF (Bean Scripting Framework) supported scripting language. BSF is Java-like language used to write test scripts. Scripts use WebScarab Java objects in order to create requests and send them to the server, and then to perform analysis on the responses [7].

The *Fragments* plugin parses HTML responses, and looks for scripts and comments. This plugin can be used for searching any hidden links, other debugging code.

The *Compare* plugin allows the user to judge the degree of difference between a number of responses. This is useful when client issued a number of requests for a particular URL, possibly via the Scripted plugin, and we would like to evaluate the results. Obviously, if it is possible to eliminate groups of identical responses at once, it can save some time.

The *Fuzzer* plugin allows users to use different combinations of parameters. The idea is to configure the request method, the basic URL, the request version, and list of parameters to expose incomplete parameter validation, leading to vulnerabilities like Cross Site Scripting (XSS) and SQL Injection.

The *Search* plugin uses BSF scripts to identify and filter conversations that user would like to analyze.

Use case of WebScarab can be divided into two phases. In the first phase analysis is performed, and we'll call it analysis phase. In the analysis phase it is recommended to use Proxy, Fragments, Scripted and Spider plugins. Proxy is very helpful in analyzing communication, especially in inspecting structure of requests and responses. Fragments plugin can be used to detect possible information leakage caused by comments or scripts. Spider plugin is used to crawl through application to gather all possible links and even find links to hidden files.

Second phase is a testing phase. WebScarab is an excellent tool for manual testing. Main plugins used in this phase are: Manual Request, Session ID analyzer and Fuzzer plugin. Manual Request plugin is used to make customized requests and for cookie manipulations. It can be used to detect security vulnerabilities like HTTP splitting. Session ID analyzer collects

and examines a reasonably large sample of session identifiers, to determine if they could be vulnerable to prediction, or brute force attacks. Fuzzer is very useful in detection of SQL or LDAP injection, XSS vulnerabilities and insufficient process validation vulnerability.

2. Pantera

OWASP Pantera is a web assessment project written in Python [8] currently in beta development phase. Pantera itself is a Web application that runs inside the browser and can be customized by the user. Although Pantera is still under development it offers tools like penetration testing proxy, an application scanner and an intelligence analysis framework. Pantera's analysis engine can store each web page that Pantera sees and analyze it for comments, scripts, vulnerabilities, hidden tags and more. All this is done in background and transparent to the user while testing the Web application manually. All the information gathered during analysis is stored in the database [9]. Pantera has many utilities like Data Miner, Interceptor, Session Trace, Fuzzer, Web spider. All those utilities make Pantera powerful web assessment studio in searching of all kinds of vulnerabilities.

In future, the primary goal of Pantera would be to combine automated capabilities with complete manual testing to get the best penetration testing results.

3. ATK

Attack Tool Kit (ATK) is tool developed in Visual Basic, written for Windows operating system. It is licensed under the GNU General Public License (GPL) and is free to use and distribute. It can be used to perform fast checks for dedicated vulnerabilities [10].

The tool is also based on plugin architecture, but plugins have a little bit different meaning than usual. Each plugin is actually one check, i.e. one specific test designed to detect or exploit certain vulnerability or flaw. The tool is shipped with 340 preinstalled plugins (checks), but it is very simple to download and add new one or even write custom ones. The plugins are written in simple scripting language, and there exists detailed documentation on how to use it. The biggest advantage of this tool is possibility of easy modification of each plugin [10]. It can even be done during run-time.

The plugins can be searched by several parameters: name, ID, port number, family, severity or class (of vulnerability). All this makes it very simple to find required plugin. On Visualizing form, it can be monitored on dynamic sketch what's going on between the attacker and the target. On Response form, the last response from the server is shown. Everything is logged and can be nicely combined into one report and statistics sheet.

This tool is great for verifying existence of a specific vulnerability and exploitation of a specific weakness. This makes it useful for scanning systems for vulnerable applications or server flaws and it comes handy while doing penetration tests. However, the tool is not designed for automatic testing of custom made applications and finding new security flaws. For automatic scanning, it relies on knowledge base of known server and application flaws that it tries to find. To test custom made applications, new tests need to be written.

4. Gamja

Gamja is another project written under GPL license. It is command-line tool written in Perl [11], portable to most popular platforms and at this moment it is in early beta phase. This tool is able to find XSS vulnerabilities, SQL Injection flaws and does URL perimeter validation check [11].

The scanning of web site is fixed process that cannot be altered. Only parameters that can be configured for scanning are port number and start page. As this project is in early phase of development, it is assumed that new configuration options and features will be added. The tool crawls through a web site for all accessible links, finds entry points to the application and tries to exploit them. As there is no available documentation for the tool, the methods of how this tool works can only be deduced by examining the source code.

5. Burp Suite

Burp Suite is a tool for attacking web applications. It consists of several, so called, "burp tools" (Proxy, Spider, Intruder and Repeater) and the robust framework that lies beneath them. The framework is designed to handle HTTP requests, authentication, downstream proxy, logging and alerting, and the tools are designed to help in a process of assessing applications. All tools are well matched and information gathered from one tool can easily be shared with all the other tools. Each tool can be used as a single application, but the strength of Burp Suite comes when combining all of them together.

Burp suite is written in Java, so it is portable on all platforms that have Java Runtime Environment available. It is copyrighted to PortSwigger.net [13]. The code is closed, but the tool can freely be used for personal and commercial purposes.

Burp Repeater is tool for manually sending HTTP or HTTPS requests to web application. The request sent can easily be altered before sending new one. The Repeater remembers all sent requests and responses and allows simple navigation through them. It's search engine can be used for tracking values that match regular expressions through given responses. It is well connected with other Burp tools such as Burp Proxy.

Burp suite has its own proxy. It allows user to intercept, inspect and modify the raw traffic passing in both directions. It can also display traffic in three different ways: raw text, table of parameters and values, and hexadecimal view. The incoming and outgoing traffic can be filtered using very efficient filtering rules that can be combined using simple filter statements. One more addition to this tool is header manipulation area, which allows configuring replacement for some strings (matched by regular expressions) found in HTTP headers. All traffic is saved to history where it can select some packet and send it to other tool such as Burp Repeater, Intruder or Spider.

Burp Spider is tool for enumerating application's content and functionality. It follows hyperlinks found within HTML, JavaScript, and submitting forms, but it also uses other clues such as directory listings, source code comments and configuration files. The results can be displayed in tree view or table format and requests can be again sent to burp intruder or repeater. When one item is selected, the tool also displays information regarding where this resource is linked from, and what other hyperlinks does it have.

Burp Intruder is central tool for assessing web applications. It is used to automate a wide range of attacks against applications, including testing for common web application vulnerabilities.

Some of them are SQL injection, cross-site scripting, buffer overflows, directory traversal, etc. Before using the Intruder, the user must gain some knowledge of functionality and structure of the target application. This information can be gathered with other burp tools. First, places in HTTP requests where payloads will be injected have to be defined. There are many types of payloads that can be used: preset list of strings, character and case substitutions, number and date iterators, bruteforcer character sets, illegal Unicode and null values, etc. One example where injecting payloads can be used is password guessing. Besides performing the simple brute force attack, more complex types can be achieved using character substitution (common substitution of letters with numbers that look alike), case substitutions, etc. All payloads can be URL-encoded or base-64 encoded and its hash can be calculated. All this makes Intruder highly configurable and capable for all types of attacks.

Burp Suite is excellent tool for checking and attacking web applications. It is not simple point-and-click tool as it requires some knowledge of the target application. But, it can easily automate further investigation and exploitation of vulnerabilities and flaws.

At the time of this writing, it is announced that another tool will soon be released, Burp Scanner. Burp Scanner will perform automated vulnerability assessments of web-enabled applications [13].

6. Wapiti

Wapiti is a web application security auditor written in Python. It works as a black box vulnerability scanner and it can find following vulnerabilities: Database injection, Cross Site Scripting, Bad File Handling detection, Command Execution detection, Lightweight Directory Access Protocol (LDAP) and Carriage Return Line Feed (CRLF) injection. [14]

For optimal performance, it should be used with Tidy library [15] whose purpose is to clean up the HTML and thus, helps scanner to do the scan more efficiently. The code is broken in several files. Wapiti.py acts as a fuzzer, lswww.py is webspider and cookie.py and getcookie.py are tools for getting cookie from server.

It is command line tool so all configuration is done through parameters to the application. The configuration is very simple and consists of configuring start URL, excluding some URLs from checking, setting proxy, cookie or authentication credentials.

The tool is released under GPL.

IV. CONCLUSION

Developing a secure web application has become an important, but difficult task to many web developers. Difficulty lies in growing number of vulnerabilities which cause web applications to malfunction. While developing applications, focus is mostly on functionality, scalability and user-friendly features which leaves less time for development of security measures. Many Web applications go through rapid development phases with extremely short turnaround time, making it difficult to eliminate vulnerabilities. Much vulnerabilities can be avoided by better coding, secure strategies and education of developers, but these solutions are very expensive.

Today's tendency of solving security issues and removing vulnerabilities from web applications is to use web application security assessment tools. Open source web application security tools are free so they don't increase cost of developing web application. Every tool analyzed in this paper has it own field of usage where it shows the best results.

OWASP WebScarab is one of the most comprehensive tools in the field used for analyzing request and responses from both HTTP and HTTPS communication. It is also very powerful tool in doing manual tests of web applications. Users of this tool should know basics of HTTP and of today's web application vulnerabilities to fully use every feature of WebScarab. On the other hand, WebScarab's limitation is an absence of a database with known prearranged list of vulnerabilities, so WebScarab can't be used for automatic web application security tests. As a result of this shortcoming, OWASP took one step further, and started development of a new project – Pantera. Idea of Pantera is to provide a user with a full security assessment of a web application. Pantera will have more features than WebScarab, especially in the field of automatic test and data mining. Also, knowledge needed to use Pantera would be minimal.

For those looking for a database-oriented tool that checks server and application for known flaws, ATK is a suitable tool. It is very simple to use, but it depends on a knowledge base, so if not updated regularly, it may not find newly discovered vulnerabilities. It can be used as first step in assessing the target: defining on which server application run, finding some server misconfigurations, gathering sensitive data, etc.

For more aggressive approach, there is Burp Suite. Beside finding vulnerabilities, this tool can also be used for exploiting them. One of the examples is automatization of password guessing. However, to use this tool, the user should be familiarized with concepts of why vulnerabilities exist and how exploits work.

Gamja and Wapiti projects have not come so far in richness of features, but if there's a need for command line tools that are portable via all kind of platforms, they should be tried.

In Table 1., it is given overview of each tool's features and field of use.

TABLE I.
OVERVIEW OF TOOL'S FEATURES

	Tests for known vulnerabilities	Automatic tests	Manual tests	Fetching and using Cookies	HTTP packet and Url parameters manipulation	Proxy	Link crawler
WebScarab			x	x	x	x	x
Pantera			x		x	x	x
ATK	x	x	x				
Gamja		x					x
BurpSuite			x		x	x	x
Wapiti		x		x			x

REFERENCES

[1] Web Application Security Consortium – Web Hacking Statistics (10.1.2007.) http://www.webappsec.org/projects/whid/statistics.shtml

[2] The Open Web Application Security Project, “The Ten Most Critical Web Application Security Vulnerabilities: 2007 Update”, Release Candidate 1

[3] Steven M. Christey, “Vulnerability Type Distribution in CVE”

[4] Web Application Security Consortium – Threat classification (15.1.2007) http://www.webappsec.org/projects/threat/

[5] A. Klein, “’Divide and Conquer” HTTP Response Splitting, Web Cache Poisoning Attacks, and Related Topics”, Sanctum Inc., March 2004

[6] OWASP – OWASP WebScarab Project (20.1.2007) http://www.owasp.org/index.php/OWASP_WebScarab_Project

[7] WebScarab - User documentation (21.1.2007) http://dawes.za.net/rogan/webscarab/docs/

[8] OWASP – OWASP Pantera Project (28.1.2007) http://www.owasp.org/index.php/OWASP_Pantera_project

[9] Python Programming Language http://www.python.org/

[10] The Attack Tool Kit Web Site, “Marc Ruef” (27.1.2007) http://www.computec.ch/projekte/atk/

[11] Perl.com: The Source for Perl http://www.perl.com

[12] Gamja: Web vulnerability scanner (28.1.2007) http://sourceforge.net/projects/gamja

[13] PortSwigger.net (25.1.2007) http://portswigger.net/

[14] Wapiti (26.1.2007) http://wapiti.sourceforge.net/

[15] Tidy library (26.1.2007) http://tidy.sourceforge.net/