TALLINN UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Department of Computer Science

ITV70LT

Martin Kiigemaa 111602IVCMM

AUTOMATED SECURITY TESTING SOLUTION FOR WEB APPLICATIONS IN SKYPE

Master thesis

Elar Lang

M.Sc

Penetration tester/Lecturer

Rain Ottis

Ph.D

Associate Professor

Tallinn 2014

Declaration

I hereby declare that I am the sole author of this thesis. The work is original and has not been submitted for any degree or diploma at any other University. I further declare that material obtained from other sources has been dully acknowledged in the thesis.

…….. …….

(Date) (Authors signature)

List of Acronyms and Abbreviations

XSS	Cross-Site Scripting
SQLi	SQL injection
API	Application Programming Interface
UI	User Interface
CI	Continuous Integration
URL	Uniform Resource Locator
OWASP	Open Web Application Security Project
GUI	Graphical User Interface
ZAP	OWASP Zed Attack Proxy
FP	False Positives
JSON	JavaScript Object Notation
PoC	Proof of Concept
CSRF	Cross-Site Request Forgery
DDoS	Distributed Denial of Service
REST	REpresentational State Transfer
AJAX	Asynchronous Javascript And XML

Abstract

Attacks on the web pages are happening every second – some of them with serious consequences and others with no damage. Attacks what did not do any damage can be called an attack attempts, what failed due to the fact that web application was secure enough. To get a web application to be secure enough, the owner of the web page has to perform necessary security tests.

Current thesis analyzes how to create and implement automatic security testing solution for Skype Web Development team´s web applications so that possible attack attempts would stay as attempts. It is researched, which open source security scanner would be wise to implement, considering there is already existing testing automation framework in place, with what the security scanner should be integrated.

Resümee

Ründeid veebilehtedele toimub igal sekundil – mõned neist tõsiste tagajärgedega ning teised ilma kahju tegemata. Kahjuta lõppenud rünnakuid veebilehtedele võib nimetada ründekatseks, mis on ebaõnnestunud tõenäoliselt tänu piisavale turvalisusele veebirakenduses. Veebilehe piisavalt turvaliseks muutmiseks on rakenduse omanik pidanud läbi viima vajalikud turvatestid.

Käesolev lõputöö uurib, kuidas luua automaatne turvatestimise lahendus Skype Web Development tiimi veebirakendustele, et kõik ründed meeskonna rakenduste vastu jääksidki vaid katseteks. Uuritakse, millise vabavaralise turvaskänneri installeerimine oleks mõttekas, võttes arvesse, et meeskonnas juba eksisteerib automaatsete testide raamistik, millega uus turvaskänner tuleb integreerida.

Table of Contents

List of figures

List of tables

1. Introduction

Majority of today´s sites on the web are applications, what are highly functional and depend on flow of information between the user and server. It is possible to register and login, do financial transactions and search. The outcome (often happens to be highly sensitive data) presented to users is generated dynamically and is normally specific to each user. Security therefore, is a big issue. [37]

To be able to comprehend possible threats in today's cyber world, every company who is active on the Internet, dealing with customers' sensitive data and providing them service needs to think about security of their systems.

Multibillion companies are being compromised every year [17, 18, 19]. In 2013 the biggest intrusion was Adobe's case – breach impacted 38 million users and Adobe also suffered software source code theft of multiple products [1].

Another example, of major company being hacked, is Twitter. With 250 000 users' sensitive data like email addresses, session tokens and encrypted passwords stolen [3].

Therefore companies need to invest in security. In authors option the easiest and cheapest way to invest to that field is to perform regular security audits or in other words - perform security testing. Since majority of software development companies are all moving to agile methodology [6], continuous delivery and continuous integration then manual security testing is not enough to verify if security regression has occurred with new development.

Company which is actively promoting security testing for their products and actively blogs about it is Mozilla [4]. They started moving to CI (Continuous Integration) at 2011 and during that time they were also automating their security test cases to reduce the risk of introducing security flaws to production [5].

1.1 Current problematic situation in Skype

Currently Skype has a test automation framework written in Python – it supports executing API tests (tests operate at a certain REST API level by verifying the input and output of the API) and Webdriver UI tests (user interface tests which operates at the browser level and actually verifying the output of the page by checking the page from the browser). Although there exists a good and working solution for test automation, the situation is poor regarding security testing – there is no automation framework in place for web applications, to locate

and handle security vulnerabilities. Therefore majority of security related threats are found during manual penetration testing.

While moving towards continuous delivery, there is a definite need of security related test automation to be in place, other ways there will be a great risk of being compromised by a security flaw what reached to production environment due to lack of tests.

That being said, since it is decided to start executing automated security tests there is a need to answer to few questions first:

- What are the ways to automate security testing in Skype?
- What do the current, on the market, solutions offer and how to select one that suits all the company needs?
- How to implement it with existing testing solution?

1.2 Importance of the problem

Problematic situation in Skype, regarding security test automation, is actual and in high priority. If system´s security gets compromised, it could seriously reflect company´s revenue.

Skype is being used by 40 million customers at each moment of time [36], which means that in the case of malicious use of security flaw the number of affected computers would be tremendous.

1.3 Purpose of the thesis

There are a lot of security testing tools on the market – commercial versions and open source ones. Current thesis is attempting to select the best one for automating security testing by relying on open source tools only.

Skype Web Development team web application varies from client usable applications from only internal web applications.

The aim of this thesis is to select and implement security testing solution for web applications in Skype Web Development team, based on the company needs (specific needs described in Chapter 3). Implementation of the new solution should be aligned with already working test

automation framework. Final outcome of current thesis should be a solution that could locate different permutations of vulnerabilities.

Expected outcome of the solution is to provide instant feedback about new security related threats in the application to help company to drive closer towards continuous delivery.

1.4 Outline of the thesis

The thesis is organized as follows. Chapter one gives introduction to thesis what contains description of the current problematic situation, overview how important the problem is to Skype and outline of thesis purpose. Chapter two gives introduction to the analyses part of the work by describing how it is planned to improve current situation. In chapter three will be described the company needs for the security scanner to be selected – bearing in mind the simplification of implementation and management aspects. Chapter four will cover the related work of various security researchers in the Internet. In chapter five will be the analytical aspect of the thesis – what is the most suitable tool based on comparative analyses. Chapter six contains all the aspects of implementing selected scanner to our existing process – including faced problems on the path and the final outcome. In chapter seven is covered future tasks what will include all nice to have features of implemented solution. Chapter eight is the conclusion of the thesis, covering the work what has been done.

2. Recommendation of improvement

To improve the current unacceptable situation, a new automated security testing solution should be created and implemented to prevent critical security flaws reaching to production environments.

This platform should be able to discover and report located security vulnerabilities, including as less false positives as possible.

Solution should be implemented in a way that written security tests will be executed after application has been built and deployed to testing environment. For the execution should be used already existing test management tool – can handle tests written in Python. Besides the automated execution there should also remain a possibility to execute tests manually though test management system. Final test report should also be managed by existing test management tool.

For security auditing should be used an external security scanner tool what would meet all the possible company needs (described in Chapter 3).

2.1 Advantages of automated external application scanning

External application scanning will introduce many advantages compared to manual penetration testing:

- Possibility to quickly identify misconfigurations and error conditions resulting from invalid input;
- Ability to run on automatically on regular basis to provide reports and ongoing vulnerability statistics to support CI;
- Will speed up process to audit large applications;
- Lower cost to operate and execute compared to excessive and expensive manual regression testing. [7]

2.2 Disadvantages of automated external application scanning

Besides advantages there will be disadvantages introduced with the implementation of external scanner – manageable and unmanageable.

2.2.1 Manageable disadvantages

- Requires appropriate skills to configure and use correctly.

- Requires efforts to emulate user session (session cookies or through form). [7]

2.2.2 Unmanageable disadvantages

- Cannot locate vulnerabilities what will appear only after specific steps are performed beforehand.
- Tools are not intelligent enough for some aspects of the application and therefore will generate false positive results. [7]

2.3 Concept of the proposal

The main working flow of application to be developed and implemented is to gather all visited URL from application logs (URL-s what are visited with GET and POST HTTP requests). The reason why the security scanner should work with previously collected URL-s and not to crawl through the web application by itself, is AJAX forms and request in Skype web applications – initial review with popular scanners proved that they cannot handle AJAX based web pages.

When the security tests are triggered for certain application, then it is possible to scan URL-s related to this application. Full workflow of the new solution is visible in Figure 1.

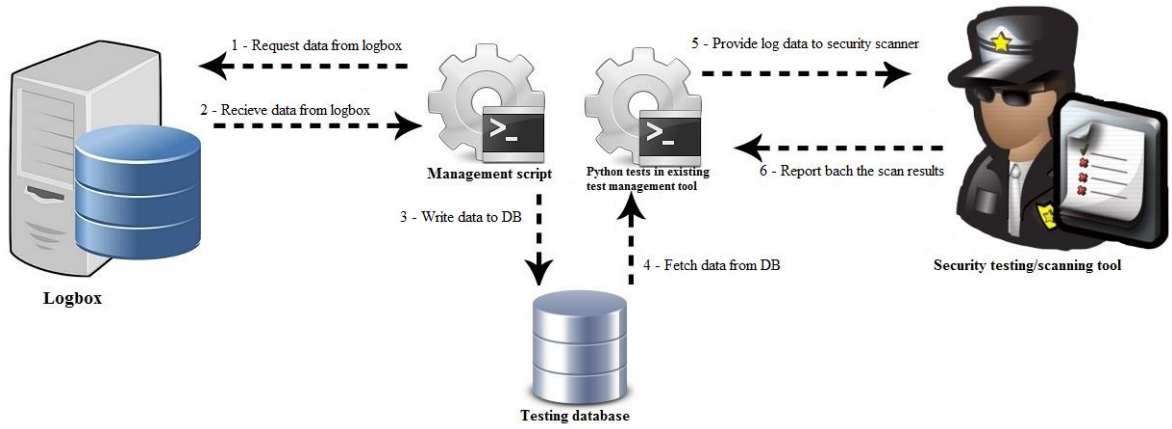

Figure 1 Concept of solutions workflow

2.3.1 Tool selection

The whole concept of new security testing solution proposal is to select most suitable tool for the company needs. The list of usable and applicable tools will be gathered from "Related work" chapter.

2.3.2 Providing necessary data to the tool

To be able to provide necessary data to the scanning tool there should be developed a script what will collect all the GET and POST request URL-s from our log server (defined as logbox in Figure 1) and insert gained data to testing database.

When records are inserted to database and certain application security tests are executed, then it is possible to query URL-s related with this application from testing database and provide them to security scanner.

2.3.3 Handling the results

Since there is a plan to integrate the new security testing solution to Skype´s testing automation framework, then the result handling would already be solved by the existing solution.

There should be created new security tests next to regular Python tests into test management system. The execution of the security related tests would work as with regular tests and therefore also the test reports would be handled the same way.

2.3.4 Is it solving the problem?

Described solution should help to solve the problem related with lack of security testing for Skype web applications. The solution seems to be suitable for this problem due to the fact that it will support moving to CI and help to prevent security regression reaching to production systems.

Collecting the URL data from the actual logs should ensure that the scanner will receive the latest and actually visited pages from where to check vulnerabilities from.

Using selected scanner tool is supposed to help to make sure the number of security holes, introduced with new development cycle, are as minimal as possible.

Reporting scanner results back to the test management system helps to generate overview of security related status of current release.

2.4 What should be detected?

Based on the OWASP 2013 TOP 10, three most critical web application security flaws are [25]:

- Injection:

    - Injection flaws - SQL, OS and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query.
- Broken authentication and session management:
    - Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, session tokens or to exploit implementation flaws to assume other users' identity.
- Cross-Site Scripting:
    - Allows attacker to execute scripts in the victim's browser which can hijack user sessions, deface web sites or redirect user to malicious sites.

Since broken authentication and session management testing does not meet the current proposal of improvement solution, then it is required that the selected scanner should be able to detect at least Cross-Site Scripting (XSS) and SQL injection (SQLi) vulnerabilities (if OS and LDAP injection modules are also supported by the tool, then those tests would be included also in the future).

2.4.1 Cross-Site Scripting

Cross-Site Scripting attacks are based on malicious scripts what are injected into otherwise legitimate and trusted web sites. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user and generates the output from it without validating or encoding it. [26]

XSS can be categorized into three categories:

- Stored:
    - Attacks where the injected script is permanently stored on the target servers, such as in a database - in a message forum, visitor log or comment field. [23]
- Reflected:
    - Attacks where the injected script is reflected off the web server, such as in an error message, search result or any other response that includes some or all of the input sent to the server as part of the request. [23]
- DOM based:
    - Attack where in the attack payload is executed as a result of modifying the DOM "environment" in the victim's browser used by the original client side script, so that the client side code runs in an "unexpected" manner. [22]

Cross-Site Scripting attacks are dangerous due to the fact that if found on a website it is possible to attack with multiple vectors, starting with defacing a website (that happened with CIA website [27]) and finishing with creating a DDoS (Distributed denial of service) – happened with video-sharing site Incapsula [28, 29]. XSS attacks are widely spread - even two of the biggest social network sites like Facebook and Twitter [30] have been compromised multiple times by the XSS flaw [31, 32, 33].

2.4.2 SQL injection

A SQL injection attack consists of injection of a SQL query via the input data from the client to the application. A successful SQL injection can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database, output the contents of a file on the system and in some cases issue commands to the operating system. [24]

There is a special type of SQLi called Blind SQL Injection. It is a type of SQL Injection attack that asks the database true or false questions and determines the answer based on the applications response not based on the web servers error responses (in the case where error reporting might have been turned off or a generic error message is shown by the web application). [20]

SQL injection attacks are dangerous due to the fact that when web page is compromised through SQLi attack, then it is reasonable to believe that sensitive customer data has been compromised also – as happened with Yahoo when password hashes of 450 000 users were posted to Internet [34]. Sony Pictures even suffered a 171 000 000 dollar loss from a single SQL injection attack what resulted loss of customer’s private data, Sony´s administrator details including password, 75 000 music codes and 3.5 million music coupons [35].

3. Company needs for future security testing application

Skype already has built a fully functional test automation framework including test runners, test report and statistics tool and test libraries. When developing new security testing solution, then already created functionality should be taken into account for the sake of manageable implementation and integration with old system. The following chapter describes all the needs and requirements Skype has for the scanning tool, to be able to build usable security testing solution on top of these expectations.

3.1 License

The tool what will be used to build the solution, should be open source software [15], mainly for the following reasons:

- Customizability:
  - Since the code is open, it is a matter of coding and adding the functionality needed;
  - Due to the open code it is possible to work through the code and understand how it works and why it works the way it works – what makes it possible to alter it if needed.
- Cost:
  - Since majority of open source software is free of charge it will not generate any negativity along managers;
  - Helps to speed up the implementation process due to the fact that no financial decisions are required.

3.2 Platform

Already developed test management tool and test runners are developed on Linux operating system, what means that scanner tool should be able to run on Linux. There is no need to have support for Windows or Macintosh.

There is no GUI support in existing test runner machines and there are no future plans for the implementation of it, which means there should be a possibility to execute and configure the scanner only through command line.

Test runners are built only to execute tests written in Python – considering this, then for the concern of simplified implementation (all required Python packages are already installed) it would be reasonable to pick a scanner what is also written in Python.

3.3 Features

Based on the new solution improvement proposal and existing web applications Skype has, it is required that selected security scanner should include following features:

- Ability to detect reflected [23] and stored [23] XSS vulnerabilities:
    - Scanner should be able to detect XSS vulnerabilities related with GET requests – by making requests to possible parameters with malicious payloads;
    - Scanner should be able to detect XSS vulnerabilities related with POST requests – by inserting malicious payload data to requests and validating the output.
- Ability to detect different type of SQL injection [24] vulnerabilities:
    - Scanner should be able to detect regular SQL injection vulnerabilities by manipulating input parameters and verifying output on the webpage;
    - Scanner should be able to detect blind SQL injection vulnerabilities by manipulating input parameters and verifying output on the webpage.
- Support for authenticated scans:
    - All Skype web applications are behind user login page – which means the new scanner should support session management:
        - Create cookie based sessions;
        - Create form login sessions;
        - Save existing sessions for other scans.
- Possibility to scan only one page at the time:
    - Since the visited URL-s with different parameters are collected individually then it is required for a scanner to have possibility to only scan that one URL what is presented to it and not to start crawling on the web page.
- Possibility to easily develop new modules and insert new payloads:
    - The fact that the new tool should be open source ensures that it is possible to inspect source code and add new needed modules;

    - The module adding solution should be as straight forward as possible;
    - New custom payload adding should be as straight forward as possible.
- As few “false positives” as possible:
    - Based on “Security Tools Benchmarking” blog it is possible to compare the amount of “false positives” the tools reported in special testing environments with different attack methods [12];

Two of those demands (license and operating system) are taken into account in the “Related work” chapter to combine an initial list of suitable scanners. Other expectations are taken into account when analyzing that initial list more deeply in the “Comparative analysis of suitable security testing scanners” chapter.

4. Related work

Security testing automation challenge is not new to software development companies who care about security and periodic automated test execution. Due to that there are articles and writings, on different security related sites, what are already covering majority of web application scanning tools. In this chapter of the thesis it is analyzed the existing work and theories done by security researchers in the Internet. The tools found from their suggestions are brought out, analyzed and selected or ruled out based on the company needs described in previous chapter.

The most prominent site of them is OWASP (Open Web Application Security Project) [8, 9]. A list of web application vulnerability scanners are brought out on their site [10]. The lineup of scanners, what OWASP has promoted in their website, contains a lot of applications. The table below only brings out applications that are open source:

Name	Platforms
Grabber	Windows, Unix/Linux and Macintosh
Grendel-Scan	Windows, Linux and Macintosh
Nikto	Web servers testing tool
Vega	Windows, Linux and Macintosh
Wapiti	Windows, Unix/Linux and Macintosh
Wikto	Windows
Xenotix XSS Exploit Framework	Windows
Zed Attack Proxy	Windows, Unix/Linux and Macintosh

Table 1 Security scanners in OWASP site

Based on the presented data in the Table 1 and actual company needs it is possible to rule out tools which are suitable to Windows only, what makes the usable tool list from the OWASP site: Grabber, Grendel-Scan, Nikto, Vega, Wapiti and Zed Attack Proxy.

Another site where the author of this thesis found usable analyses of security scanners is a blog of security researcher Mária Jurčovičová [11]. List of scanners found from this site is:

Name	Platform
WebSecurify	Firefox plugin
SkipFish	Windows, Unix/Linux and Macintosh
W3af	Windows, Linux

Zed Attack Proxy	Windows, Linux and Macintosh
WebScarab	Proxy mode only
WebScarab Next Generation	Proxy mode only
Ratproxy	Proxy mode only
Arachni	Linux
Exploit Me	Firefox plugin
Nikto	Web server testing tool
ProxMon	Proxy mode only

Table 2 Security scanners in "This is stuff" security blog

As previously done then based on the presented data in the Table 2 and company needs it is possible to rule out not suitable tools, like the ones for Windows only, proxies and plugins, and that makes the usable tool list following: SkipFish, W3af, Zed Attack Proxy and Arachni.

One more security blog has accomplished to put together a good analyzing comparison between various security scanners – Security Tools Benchmarking blog [12]. The list there includes large amount of commercial and open source tools – since thesis in concentrated on open source tools, then only these will be mentioned in Table 3.

Name	**Platform**
Zed Attack Proxy	Windows, Unix/Linux and Macintosh
IronWASP	Windows, Linux, Macintosh
W3af	Windows, Linux
Arachni	Linux
Skipfish	Windows, Unix/Linux and Macintosh
WATOBO	Proxy mode only
VEGA	Windows, Linux
Wapiti	Windows, Unix/Linux and Macintosh
XSSer	Windows, Unix/Linux and Macintosh
N-Stalker 2012 Free Edition	Windows
Syhunt Mini	Windows

Table 3 Security scanners in "Security Tools Benchmarking" security blog

Suitable tools from the Security Tool Benchmarking blog are: Zed Attack Proxy, IronWASP, W3af, Arachni, Skipfish, Vega, Wapiti and XSSer.

Thesis will cover findings from one more security related project – Web Application Security Consortium [13]. The list of open source scanners mentioned in their page is following [14]:

Name	Platform
Arachni	Linux
Grabber	Windows, Unix/Linux and Macintosh
Grendel-Scan	Windows, Linux and Macintosh
Paros	Proxy mode only
Andiparos	Proxy mode only
Zed Attack Proxy	Windows, Unix/Linux and Macintosh
Powerfuzzer	Windows, Linux
Skipfish	Windows, Unix/Linux and Macintosh
W3af	Windows, Linux
Wapiti	Windows, Unix/Linux and Macintosh
Watcher	Windows
WATOBO	Proxy mode only
Websecurify	Firefox plugin

Table 4 Security scanners in "The Web Application Security Consortium" page

Based on the presented data on Table 4 and actual company needs it is possible to rule out the tools from this list as well, so the passing tool list from "Security Consortium" page is following: Arachni, Grabber, Grendel-Scan, Zed Attack Proxy, Powerfuzzer, Skipfish, W3af and Wapiti.

The main idea of this chapter was to analyze the work others has done and with that information put together a list of usable tools for Skype. By comparing the data inside the tables with two of the main criteria's of Skype (usage platform should be Linux compatible and tool should be open source) it was possible to pick the initial list of suitable scanners to continue with.

Suitable security scanners and the number of times they were mentioned:

Tool	Times mentioned
Grabber	2
Grendel-Scan	2

Nikto	1
Vega	2
Wapiti	3
Zed Attack Proxy	4
SkipFish	3
W3af	3
Arachni	3
IronWASP	1
XSSer	1
Powerfuzzer	1

Table 5 Security tool list and number of times mentioned

Based on the times the tool was mentioned (statistics brought out in Table 5), by other researchers, it is possible to select only the most popular ones for deeper analyses in the next chapter. Tools mentioned 1-2 times and the ones what did not get any sensational feedback in the security blogs are left out from deeper analyses – Nikto, IronWASP, XSSer, Powerfuzzer, Grabber, Grendel-Scan and Vega.

Scanner tools what are selected, based on mention count, for further analyses are – Zed Attack Proxy, Wapiti, SkipFish, W3af and Arachni.

5. Comparative analysis of suitable security testing scanners

Based on the previous chapter it is selected five suitable security testing scanners - Zed Attack Proxy, Wapiti, SkipFish, W3af and Arachni. In this chapter of the thesis the features of the tools are compared to the Skype requirements, what are brought out in the "Company needs for future security testing application" chapter. The comparison will be made with the help of comparative table and deeper individual analyses. In the end of this chapter, a tool, what will be used to develop a new security testing solution, will be chosen.

5.1 Comparative table of security scanners

To be able to select out one testing scanner, it is needed to compare tools features to previously mentioned requirements. Table 6 will bring out Skype´s expectations for new tool and will help to understand which tools are able to fulfill those requirements. Information for the field of false positives has been acquired from "Security Tools Benchmarking" blog post [12].

Requirement	ZAP	Wapiti	SkipFish	W3af	Arachni
Console mode only support	**X**	**X**	**X**	**X**	**X**
Written in Python		**X**		**X**	
Reflected XSS	**X**	**X**	**X**	**X**	**X**
Stored XSS	**X**	**X**	**X**	**X**	**X**
DOM based XSS				**X***	
SQL injection	**X**	**X**	**X**	**X**	**X**
Blind SQL injection	**X**	**X**	**X**	**X**	**X**
Authenticated scans (session cookies)	**X**	**X**	**X**	**X**	**X**
Scan only one page	**X**	**X**	**X**	**X**	**X**
Easy implementation of new modules and payloads		**X**		**X**	**X**
False positives	**XSS – 0% SQLi – 30%**	**XSS – 42% SQLi – 20%**	**XSS – 0% SQLi – 0%**	**XSS – 0% SQLi – 30%**	**XSS – 0% SQLi – 20%**

Table 6 Comparative table of security scanners

*W3af has a plugin for DOM based XSS vulnerability, but it works as a simple *grep* over the page source [21].

From the Table 6 can be seen that the most requirements are covered by Wapiti, Arachni and W3af, but even those could not provide a suitable module for DOM based XSS vulnerability – what initiates to pay more attention to the possibility to add new modules with minimal effort.

5.2 More detailed breakdown of the tools

5.2.1 OWASP Zed Attack Proxy

Although ZAP is mainly a tool with GUI, there is a way to start it in daemon mode, which automatically enables an API for more dynamic use. It is possible to configure scans also through given API so the tool would be really useful for security regression tests.

ZAP has a multiple modules for different attack vectors, including the ones Skype needs, for XSS and SQL injection.

Given tool also supports authenticated scans either through the UI or console with the help of "formauth" module – including session generating request (needs to be compiled manually or acquire them from UI) to ZAP scan context makes the tool to send this request every time the webpage requires user login.

ZAP supports a chance to scan only one page at the time, without the crawling spiders also working on the background and feeding new URL-s to scanner.

Based on benchmarking blog post it had 0% of FP in reflected XSS test and 30% of FP in SQL injection test [12].

ZAP is written in Java and in a way, that there is no easy method to add custom modules to it. The fact that it is developed in Java, unfortunately, rules this tool out for Skype since Java is not installed to our tests runners.

Local test was made with ZAP to scan one of Skype´s application, what contained "[" and "]" symbols in the URL-s the tool had to scan. Existence of those symbols created a problem and that is also one of the reasons implementation of OWASP Zed Attack Proxy was dropped.

Error example in ZAP log:

```
ERROR org.parosproxy.paros.core.scanner.VariantURLQuery  - escaped
query not valid
F[currency]=EUR&F[items][0][name]=Skype+Credit&F[items][0][price]=1
0.00000&state=setup&F[items][0][id]=skype-credit-
10&atu=1&F[atu]=1&skypename=username& F[items][0][group]=skype-
credit&F[integration]=basket-1
org.apache.commons.httpclient.URIException: escaped query not valid
```

That result was gained when testing ZAP through GUI. So there was no way to escape the query automatically in the testing code.

5.2.2 Wapiti

Wapiti is only console based application, there is no UI available, which suits the Skype needs perfectly. Configuration of the tool is made through command line by providing configuration keys as application input parameters.

Scanner contains multiple modules for XSS and SQL injection scanning. When starting the scan it is possible to add input parameters at startup to either enable or disable certain modules.

Regarding authenticated scans, Wapiti contains a separate module to gain session cookies from websites – "wapiti-cookie" module. Upon executing of this module with certain input keys it gets user password and username (forms presented on the actual webpage), logs the user in and saves gained cookie to JSON format into cookies.json file for later use of the upcoming scans.

```
./wapiti-cookie cookies.json https://
login.skype.net/login?method=Skype username=test password=test
```

Scanning only one page with Wapiti has been made as simple as possible – by passing on input key "-b page" at program startup will make the scanner only operate within the scope of given URL.

Based on benchmarking blog post it had 42% of FP in reflected XSS test and 20% of FP in SQL injection test, which is not very impressing [12].

Wapiti is completely written in Python which makes the implementation part relatively easy and painless. There is a possibility to add new modules and new custom built payloads for different module types. All payloads are located under certain attack module in a configuration file.

Local test with one of our application URL-s confirmed the correct behavior of Wapiti even with special characters. The same address was given to Wapiti for a XSS and SQL injection scan and it successfully scanned though all parameters in the URL and replaced them with malicious payload one by one.

Executing Wapiti (scanning only one page and XSS GET vulnerabilities):

```
$wapiti
"https://secure.skype.net/wallet/checkout?F[currency]=EUR&F[items][
0][name]=Skype+Credit&F[items][0][price]=10.00000&state=setup&F[ite
ms][0][id]=skype-credit-10&atu=1&F[atu]=1&skypename=username&
F[items][0][group]=skype-credit&F[integration]=basket-1" -c
cookies.json -b page -m "-all,xss:get" -v 2
```

PoC output from the tool work:

```
+GET
https://secure.skype.net/wallet/checkout?F%5Bcurrency%5D=%22%3E%3C%
2Fform%3E%3Cscript%3Ealert%28%27wy2w48dekx%27%29%3C%2Fscript%3E&F%5
Bitems%5D%5B0%5D%5Bname%5D=Skype%2BCredit&F%5Bitems%5D%5B0%5D%5Bpri
ce%5D=10.00000&state=setup&F%5Bitems%5D%5B0%5D%5Bid%5D=skype-
credit-
10&atu=1&F%5Batu%5D=1&skypename=username&%20F%5Bitems%5D%5B0%5D%5Bg
roup%5D=skype-credit&F%5Bintegration%5D=basket-1

+GET
https://secure.skype.net/wallet/checkout?F%5Bcurrency%5D=EUR&F%5Bit
ems%5D%5B0%5D%5Bname%5D=%22%3E%3C%2Fform%3E%3Cimg%09src%3D.%09onerr
or%3DString.fromCharCode%280%2Cwj1qsgyyl5%2C1%29%3E&F%5Bitems%5D%5B
0%5D%5Bprice%5D=10.00000&state=setup&F%5Bitems%5D%5B0%5D%5Bid%5D=sk
ype-credit-
10&atu=1&F%5Batu%5D=1&skypename=username&%20F%5Bitems%5D%5B0%5D%5Bg
roup%5D=skype-credit&F%5Bintegration%5D=basket-1
```

```
+GET
https://secure.skype.net/wallet/checkout?F%5Bcurrency%5D=EUR&F%5Bit
ems%5D%5B0%5D%5Bname%5D=Skype%2BCredit&F%5Bitems%5D%5B0%5D%5Bprice%
5D=10.00000&state=setup&F%5Bitems%5D%5B0%5D%5Bid%5D=skype-credit-
10&atu=1&F%5Batu%5D=1&skypename=%22%3E%3C%2Fform%3E%3Cobject%09data
%3Djavascript%3AString.fromCharCode%280%2Cwr50s6qe0r%2C1%29%3E&%20F
%5Bitems%5D%5B0%5D%5Bgroup%5D=skype-
credit&F%5Bintegration%5D=basket-1
```

5.2.3 W3af

W3af can be used either through graphical user interface or console. There are multiple configure options available through GUI and via command line. Studies show that it is possible to create configuration script were the entire needed configuration will be held and during execution of W3af core program it is possible to read all the configuration keys from that script file: [16]

```
$./w3af_console -s MyScript.w3af
```

Scanner includes multiple modules for XSS and SQL injection scanning. When configuring the scan it is possible to enable or disable either one or multiple scanning vectors from "audit" module.

Authenticated scans are also supported in W3af. There are two ways to make user authenticated – specify user login credentials in form based authentication module or pass a valid cookie file to scanner.

Scanning only one page at the time will work in W3af if there is scanner module disabled in the "crawl" section. That way the scanner only performs vulnerability checks for the target user specified.

Based on benchmarking blog post it had 0% of FP in reflected XSS test and 30% of FP in SQL injection test, which is acceptable result [12].

Local test with one of our application URL-s confirmed the correct behavior of W3af even with special characters. The same address what was given to ZAP and Wapiti, was also given

for W3af for XSS and SQL injection scan and it successfully scanned though all parameters in the URL and replaced them with malicious payload one by one.

PoC output from the tool work:

```
GET
https://secure.skype.net/wallet/checkout?F[currency]=EUR&F[items][0
][name]=Skype+Credit&F[items][0][price]=peGc1/*&state=setup&F[items
][0][id]=skype-credit-10&atu=1&F[atu]=1&skypename=username&
F[items][0][group]=skype-credit&F[integration]=basket-1

GET
https://secure.skype.net/wallet/checkout?F[currency]=EUR&F[items][0
][name]=Skype+Credit&F[items][0][price]=10.00000&state=XCKnp`&F[ite
ms][0][id]=skype-credit-10&atu=1&F[atu]=1&skypename=username&
F[items][0][group]=skype-credit&F[integration]=basket-1

GET
https://secure.skype.net/wallet/checkout?F[currency]=EUR&F[items][0
][name]=Skype+Credit&F[items][0][price]=10.00000&state=setup&F[item
s][0][id]=skype-credit-10&atu=1&F[atu]=1&skypename=YA7fG'YA7fG&
F[items][0][group]=skype-credit&F[integration]=basket-1
```

With W3af all features seemed perfect and potentially it would have been a main candidate for the choice, but there was one thing that came out when testing the scanners work locally – if the URL had multiple parameters (like the one in the example) then the tool never finished its work. Number of requests just started dropping and no results or debug lines were printed out to the console.

5.2.4 Arachni

Arachni is usable either through web interface or though command line. Web interface of the application offers multiple configuration options while command line only takes in specific configuration keys as startup arguments.

Scanner includes multiple modules for XSS and SQL injection scanning. When starting the application it is possible to enable or disable different attack vectors with startup keys (--modules=xss* key will include all XSS modules.

Authenticated scans are also supported in Arachni. There is a module called “autologin” what is supposed to take in login page, username and password field and automatically log the user in. Unfortunately that solution did not work for Skype login page – presumably due to CSRF (Cross-site Request Forgery) protection hashes.

Scanning only one page at the time will work in Arachni also. It needs “--link-count” parameter value is set as 1 and “--redirect-limit” value set as 0 – that way the crawler will not start and only given target is analyzed.

Based on benchmarking blog post it had 0% of FP in reflected XSS test and 20% of FP in SQL injection test, which is a good result [12].

Local test with one of our application URL-s confirmed the correct behavior of Arachni even with special characters. The same address what was given to ZAP, Wapiti and W3af was also given for Arachni for XSS and SQL injection scan and it successfully scanned though all parameters in the URL and replaced them with malicious payload one by one.

Executing Arachni (scanning only XSS vulnerabilities):

```
$arachni --depth=0 --link-count=1 --redirect-limit=0 --modules=xss
"https://secure.skype.net/wallet/checkout?F[currency]=EUR&F[items][
0][name]=Skype+Credit&F[items][0][price]=10.00000&state=setup&F[ite
ms][0][id]=skype-credit-10&atu=1&F[atu]=1&skypename=username&
F[items][0][group]=skype-credit&F[integration]=basket-1"
```

PoC output from the tool work:

```
[!] HTTP: URL:
https://secure.skype.net/wallet/checkout?%2520F%255Bitems%255D%255B
0%255D%255Bgroup%255D=skype-
credit&F%255Batu%255D=1&F%255Bcurrency%255D=EUR&F%255Bintegration%2
55D=basket-1%28%29%22%26%251%27-
%3B%3Csome_dangerous_input_363745c5f4329ea2d325575a33d92a900fd5ad86
88a64fc0279f5580ec82c03d%2F%3E%27&F%255Bitems%255D%255B0%255D%255Bi
d%255D=skype-credit-
10&F%255Bitems%255D%255B0%255D%255Bname%255D=Skype%2BCredit&F%255Bi
tems%255D%255B0%255D%255Bprice%255D=10.00000&atu=1&skypename=userna
me&state=setup
```

```
[!] HTTP: URL:
https://secure.skype.net/wallet/checkout?%2520F%255Bitems%255D%255B
0%255D%255Bgroup%255D=skype-
credit&F%255Batu%255D=1&F%255Bcurrency%255D=EUR&F%255Bintegration%2
55D=basket-1&F%255Bitems%255D%255B0%255D%255Bid%255D=skype-credit-
10&F%255Bitems%255D%255B0%255D%255Bname%255D=Skype%2BCredit&F%255Bi
tems%255D%255B0%255D%255Bprice%255D=10.00000&atu=1&skypename=userna
me--
%3E%3Csome_dangerous_input_363745c5f4329ea2d325575a33d92a900fd5ad86
88a64fc0279f5580ec82c03d%2F%3E%3C%21--&state=setup
```

5.2.5 Skipfish

SkipFish is a command line tool only, written in C – therefore not easy to modify and no implemented possibility to add new modules or payloads easily. Configuration parameters are given as startup arguments or loaded in from custom configuration file.

There is no way to select the modules what the scan should use – this means, that once the program has been started it will start every attack known to it against the target. That kind of approach does not suit with Skype needs, since it makes the scan significantly slower.

Authenticated scans are supported by two methods – form based login (login parameters from command line or configuration file) and cookie based authentication.

Theoretically it is possible to make the tool scan only one webpage presented. Configuration key “-I” will take in regular expression pattern and will scan only pages that will match this pattern. Also there is a chance to specify maximum depth of the crawler. The author of this thesis was not able to make SkipFish scan only the given page.

Based on benchmarking blog post it had 0% of FP in reflected XSS test and 0% of FP in SQL injection test, which is a very good result.

There is no need for local testing due to the fact that there is no possibility to make scanner only attack XSS and SQLi vulnerabilities.

5.3 Selected tool for implementation

Based on the data got from analyzing five tools it is possible to make the best selection from them and start to implement the whole solution.

Taken into account a possibility to configure through command line or with configuration file, to select XSS or SQLi modules for the scan, to make authenticated scans, to scan only one presented page at the time, considering the rate of false positives and the fact that initial local testing was successful and the tool did not collapsed it is confirmed that the most suitable solution, for Skype, would be to take into use Wapiti scanner tool.

5.4 Proof of concept of tools work against vulnerable Skype´s application

Before the full implementation of the new solution the author of the thesis would like to be confident that selected tool will be able to locate security related threats from scanned applications. For that the author set up one of Skype Web Development team application to work in local machine and introduced a reflected Cross-Site Scripting bug by removing sanitization from one of GET parameters called orderId:

```
wapiti 'http://localhost/skype-
wallet/php/tal/Paymentflow/Web/Checkout_1_2.html?state=pending&orde
rId=1' -m '-all,xss:get' -f json
```

Executing this from local machine gave back result file with XSS vulnerabilities array filled:

```
"vulnerabilities": {
    "Cross Site Scripting": [
      {
        "info": "XSS vulnerability found via injection in the
parameter orderId",
        "http_request": "GET /skype-
wallet/php/tal/Paymentflow/Web/Checkout_1_2.html?
state=pending&orderId
=%3Cscript%3Ealert%28%27weftdl4um3%27%29%3C%2Fscript%3E
HTTP/1.1\nHost: localhost\n",
        "level": "1",
```

```
"curl_command": "curl \"http://localhost/skype-
wallet/php/tal/Paymentflow/Web/Checkout_1_2.html?state=pending&orde
rId =%3Cscript%3Ealert%28%27weftdl4um3%27%29%3C%2Fscript%3E\"",
        "path": "/skype-
wallet/php/tal/Paymentflow/Web/Checkout_1_2.html",
        "parameter": "orderId",
        "method": "GET"
      }    ],
    "Htaccess Bypass": [],
    "Backup file": [],
    "SQL Injection": [],
    "Blind SQL Injection": [],
    "File Handling": [],
    "Potentially dangerous file": [],
    "CRLF Injection": [],
    "Commands execution": []
  }
```


This result gives confidence that the selected tool is able to handle the threats inside our application and it is reasonable to make efforts for the implementation of the solution.

6. Implementation of the solution

Given chapter will cover all the topics related with the implementation of the selected tool, helper scripts and integration with the existing test automation framework.

The whole implementation of the new solution is integrated with the existing test management tool in a following way:

- The scanner tool is installed to test runners;
- New security testing module is created, in Python, into existing regression test suites of applications:
    - That way they are included to application tests suite in test management tool and are automatically executed when application is built.
- New test cases are created dynamically based on the URL-s from the logs:
    - This means new test case will be created upon every unique URL (query strings are excluded when creating test case names);
    - During the test case execution it will get full URL-s, with query strings, from the database and scans all of them.
- Upon test case execution the tests will create a valid user session;
- When test case finishes the results will be accessible through the test management tool.

6.1 URL-s data inside database

All the URL-s, what has got either GET or POST requests in the application, are stored to the database for the scanner to consume.

At the moment there is an existing schema in test environment database. Inside that testing schema is a table where the collector script will insert the application name, URL and the method used to access the URL. The output from that table can be seen from Figure 2.

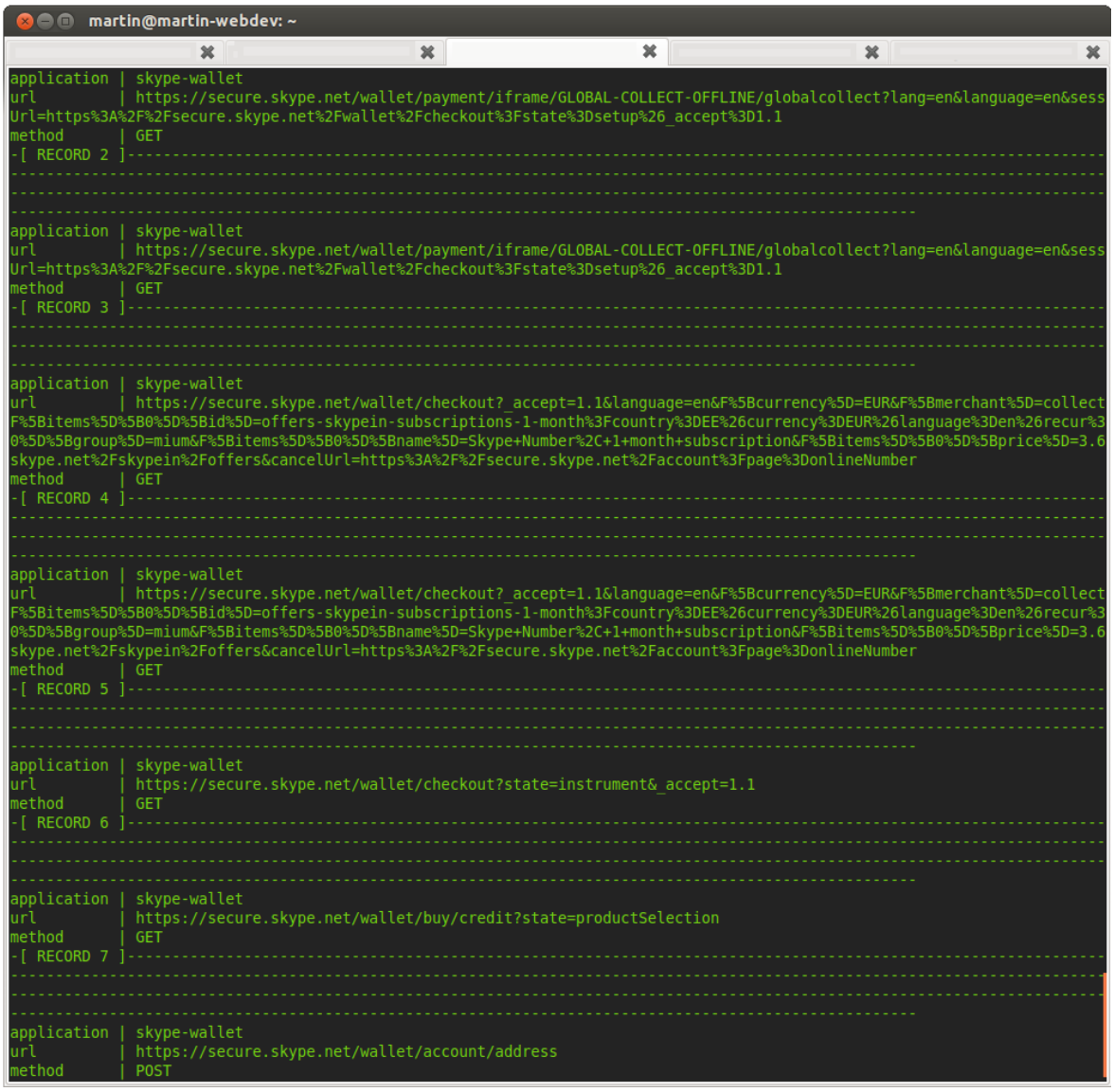

Figure 2 Database structure and data

6.2 URL collector script

The URL collector script will take application names and their log directories from configuration file and when executed, then searches all URL patterns for a certain application and inserts them to testing database with matching method (GET or POST).

Plan is that the script is executed automatically twice a day – after the test environment synchronization and at 14:00 to get the new data from the logs (incase new URL-s have been introduced in the application). At the moment the script execution is done manually to verify the correct work and performance of the script. The scheme how that script is working can be seen from Figure 3.

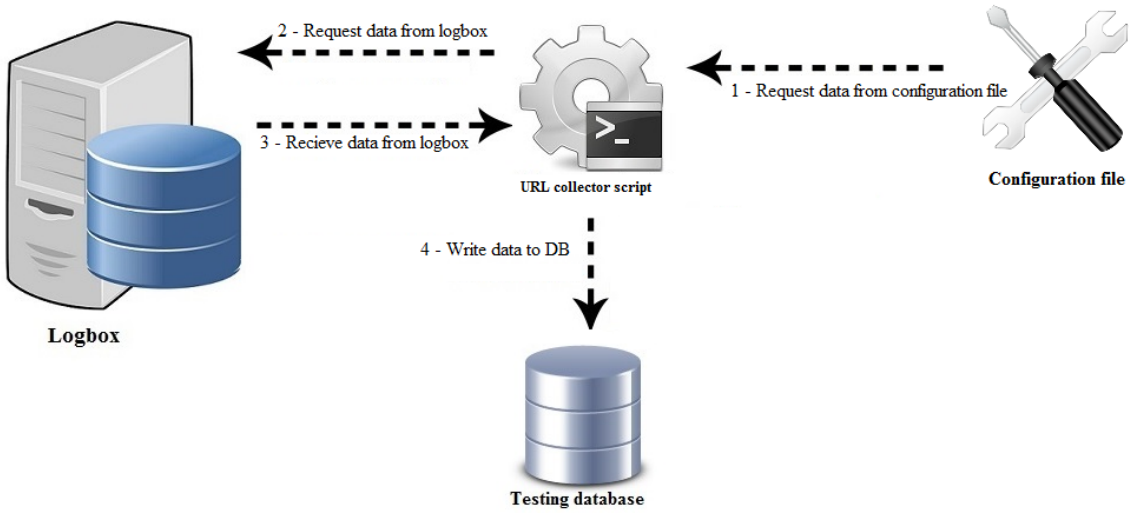

Figure 3 URL collector script in action

6.3 Security tests creation and execution

6.3.1 Dynamic test creation

Test case names are gained by querying unique URL-s (without query strings) and methods from the testing database where URL collector script has inserted the data. Since the query asks for the unique URL-s and method, then with this data it is possible to put together not repeatable test case names.

Example test case name:

```
test_walletcheckout_get
test_walletaccountaddress_post
```

For example the first test case includes the scanning of following URL-s with HTTP GET payloads:

- https://secure.skype.com/wallet/checkout?state=setup&_accept=1.1;
- https://secure.skype.com/wallet/checkout?_accept=1.1&language=et&F[currency]=EUR&state=setup.

The second example includes the scanning of following URL with HTTP POST payloads:

- https://secure.skype.com/wallet/account/address.

6.3.2 Creating a valid session for scanner

Before the security tests can be executed, there had to be created a way to generate user sessions, so that scanner would get access to user restricted pages. Session generation was established with a helper application what was included to the Wapiti scanner suite – wapiti-cookie.

```
subprocess.call('wapiti-cookie cookies.json
https://login.skype.net/login?method=skype username=test
password=test', shell=True)
```

The previous code inside our Python test case set up method will ensure that each executed test can use cookies.json file and will have a valid user session to use.

6.3.3 Scanner execution via test cases

When the legitimate session has been created, then it is possible for test case to execute the scanner. For that the Python code asks all URL-s from database *like* the URL in test case name (without query string, but slashes added):

```
cursor.execute("SELECT url, method FROM testingdb.security_test_urls
WHERE application=CURRENT_APP AND url LIKE '%TESTCASE_NAME%'")
urls = cursor.fetchall()
```

That way the test case will get all the URL-s with different query strings and passes it on to the scanner.

The main execution flow for the scanner is:

- Pass the gained URL to the scanner as input parameter:
  - wapiti/bin/wapiti URL
- Specify output file type:
  - -f json
- Make the scanner to use previously created cookies file:
  - –c cookies.json
- Make the scanner to scan only the given page:
  - –b page

- Choose the attack vector – BlindSQL, SQLi and XSS (either GET or POST based on the method field from database):
    - –m '-all,xss,sql,blindsql,permanentxss'
- Get the output result from the scan, so it could be assigned to variable for later analyses is also executed:
    - json.loads(subprocess.check_output(["cat", "vulnerabilities.xml"]))["vulnerabilities"]

When those options put into sequel commands in Python then the scanner execution commands will be:

```
scan_result = subprocess.check_output(["wapiti/bin/wapiti", URL, "-
f", "json", "-c", "cookies.json", "-b", "page", "-m", "-
all,xss,sql,blindsql,permanentxss", "-v", "2"])

found_vulnerabilities = json.loads(subprocess.check_output(["cat",
"vulnerabilities.xml"]))["vulnerabilities"]
```

Previous command will give back a result of the scanned vulnerabilities in JSON format, what will be assigned to *found_vulnerabilities* variable. Example:

```
{
    "Cross Site Scripting": [],
    "Htaccess Bypass": [],
    "Backup file": [],
    "SQL Injection": [],
    "Blind SQL Injection": [],
    "File Handling": [],
    "Potentially dangerous file": [],
    "CRLF Injection": [],
    "Commands execution": []
  }
```

That result is analyzed further inside the test code and if one of the attack vector arrays is not empty, the test is marked as red in the test management tool.

6.4 Outcome

At the moment the solution works as described in the chapter. There are separate security testing branches inside our applications testing suites in test management tool and under those branches are dynamically created security test cases based on the URL-s got from the log server. The tests are run automatically when new version of application is deployed to testing environment, but there is a possibility to execute the tests manually also.

New security tests can be created by any of team quality engineer. Actions needed to get a new application to be covered with security tests are:

1. Add new application to log collector script configuration;
2. Create python security module to existing test suite of the application;
3. Dynamically create test cases for all URL-s (without query strings);
4. Pass fetched URL-s from database to security scanner.

7. Future development

Although, created security testing solution is implemented and working as expected, there still is some development needed, to improve its work.

The most important part is to create a way to get only unique URL-s from log server – at the moment we are scanning one URL multiple times even if only the username parameter is different. This is the reason why it is needed to develop an API to get all applications URL-s with predefined parameters that are changing and should not be considered when putting together the URL-s what are about to be scanned – for example if only username is different for two URL-s then this URL should only be scanned once.

Due to the fact that it is relatively easy to create new modules and payloads for Wapiti, there are quality engineers who are monitoring what should be the new modules and payloads what are needed and will add them continuously. That includes activating other scanning modules what Wapiti includes – File disclosure, PHP injections, OS injection, use of known potentially dangerous files and weak *".htaccess"* configurations.

8. Conclusion

Current thesis analyzed the best security testing solution for Skype Web Development team. Analyze was done by selecting existing security scanners on the market and comparing them with different company needs – defined by the team needs and a fact that the whole new solution was to be implemented to the existing test automation framework.

The initial selection of scanners was put together by the work made by other security researchers and by the requirements specified in the company needs. From the initial list was selected 5 of the main tools – the ones what got the best results from other researchers and the ones what were more popular in terms of analyze count from other research blogs. The 5 main tools were: OWASP Zed Attack Proxy, Wapiti, W3af, Arachni and Skipfish. With each of the scanner there was done an initial test against one of Skype application to verify the capability to work correctly. The most suitable of these tools was Wapiti scanner which includes separate module to gain cookies (helps with authenticated scans), supports new payloads and module adding with minimal efforts and was able to scan and was able to locate purposely introduced security bug from one of Skype application.

Integration of the selected security scanner with the existing test automation framework was done by installing Wapiti tool to test runner machines and by creating separate security tests next to existing tests, what ensured that the results of the security tests were also displayed through test management system.

The new solution is usable by every quality engineer of the team and new security tests are automatically created based on the visited URL-s.

References

1. Target and The 10 Biggest Hacks of 2013 [WWW] http://www.pcmag.com/slideshow/story/319071/target-and-the-10-biggest-hacks-of-2013/1 (10.04.2014)
2. Skype Statistics [WWW] http://www.statisticbrain.com/skype-statistics/ (20.05.2014)
3. Keeping our users secure [WWW] https://blog.twitter.com/2013/keeping-our-users-secure (20.05.2014)
4. Introducing Mozilla Winter of Security 2014 [WWW] https://blog.mozilla.org/security/ (20.05.2014)
5. Automating Test Cases [WWW] https://blog.mozilla.org/webappsec/2011/10/26/automating-test-cases/ (20.05.2014)
6. Survey Archives [WWW] http://stateofagile.versionone.com/survey-archives/ (20.04.2014)
7. Automated External Application Scanning [WWW] https://www.owasp.org/index.php/Definition_for_Security_Assessment_Techniques#Automated_External_Application_Scanning (20.05.2014)
8. Winners [WWW] http://awards.scmagazine.com/Winners2014 (20.05.2014)
9. About The Open Web Application Security Project [WWW] https://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project (20.05.2014)
10. Vulnerability Scanning Tools [WWW] https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools (20.05.2014)
11. Testing for XSS Vulnerabilities - Choosing a Scanner [WWW] http://meri-stuff.blogspot.com/2011/07/testing-for-xss-vulnerabilities.html (20.05.2014)
12. The Web Application Vulnerability Scanners Benchmark [WWW] http://sectooladdict.blogspot.com/2014/02/wavsep-web-application-scanner.html (20.05.2014)
13. Web Application Security Consortium [WWW] http://www.webappsec.org/ (20.05.2014)

14. Web Application Security Scanner List [WWW] http://projects.webappsec.org/w/page/13246988/Web%20Application%20Security%20Scanner%20List (20.05.2014)
15. The Open Source Initiative [WWW] http://opensource.org/ (20.05.2014)
16. Automated Audit using W3AF [WWW] https://www.owasp.org/index.php/Automated_Audit_using_W3AF (20.05.2014)
17. A Timeline of Companies That Have Been Hacked In 2013 [WWW] http://www.heavy.com/tech/2013/03/a-timeline-of-companies-that-have-been-hacked-in-2013/ (20.05.2014)
18. The Biggest Hacking Attacks Of 2011 [WWW] http://www.businessinsider.com/imf-cyber-attacked-hackers-sony-rsa-lockheed-martin-epsilon-michaels-2011-6 (20.05.2014)
19. Looking back at the major hacks, leaks and data breaches [WWW] http://www.zdnet.com/2012-looking-back-at-the-major-hacks-leaks-and-data-breaches-7000008854/ (20.05.2014)
20. Blind SQL Injection [WWW] https://www.owasp.org/index.php/Blind_SQL_Injection (20.05.2014)
21. w3af – Plugins [WWW] http://w3af.sourceforge.net/plugin-descriptions.php#domXss (20.05.2014)
22. DOM Based Cross Site Scripting or XSS of the Third Kind [WWW] http://www.webappsec.org/projects/articles/071105.shtml (20.05.2014)
23. Stored and Reflected XSS Attacks [WWW] https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)#Stored_and_Reflected_XSS_Attacks (20.05.2014)
24. SQL Injection [WWW] https://www.owasp.org/index.php/SQL_Injection (20.05.2014)
25. OWASP 2013 TOP 10 list [WWW] https://www.owasp.org/index.php/Top_10_2013-Top_10 (20.05.2014)
26. Cross-site Scripting (XSS) [WWW] https://www.owasp.org/index.php/Cross-site_Scripting_(XSS) (20.05.2014)
27. XSS attack on CIA (Central Itelligence Agency) Website [WWW] http://thehackernews.com/2011/06/xss-attack-on-cia-central-itelligence.html (20.05.2014)

28. XSS flaw in popular video-sharing site allowed DDoS attack through browsers [WWW] http://www.computerworld.com/s/article/9247450/XSS_flaw_in_popular_video_sharing_site_allowed_DDoS_attack_through_browsers (20.05.2014)
29. DDOS Attack Enabled by Persistent XSS Vulnerability on Top Video Content Provider's Site [WWW] http://news.softpedia.com/news/DDOS-Attack-Enabled-by-Persistent-XSS-Vulnerability-on-Top-Video-Content-Provider-s-Site-436029.shtml (20.05.2014)
30. Top 15 Most Popular Social Networking Sites | May 2014 [WWW] http://www.ebizmba.com/articles/social-networking-websites (20.05.2014)
31. Facebook Login Page hacked through XSS by Mauritania Attacker [WWW] http://hackersnewsbulletin.com/2013/06/facebook-login-page-hacked-through-xss-by-mauritania-attacker.html (20.05.2014)
32. Hacking Facebook users just from chat box using multiple vulnerabilities [WWW] http://thehackernews.com/2013/04/hacking-facebook-users-just-from-chat.html (20.05.2014)
33. All about the "onMouseOver" incident [WWW] https://blog.twitter.com/2010/all-about-onmouseover-incident (20.05.2014)
34. Yahoo Hit By SQL Injection Attack [WWW] http://www.internetnews.com/security/yahoo-hit-by-sql-injection-attack.html (20.05.2014)
35. Hackers Attack Sony Pictures with Single SQL Injection [WWW] http://www.thewhir.com/web-hosting-news/hackers-attack-sony-pictures-with-single-sql-injection (20.05.2014)
36. 40 Million People: How Far We´ve Come [WWW] http://blogs.skype.com/2012/04/10/40-million-people-how-far-weve/ (20.05.2014)
37. Stuttard, D., Pinto, M. The Web Applications Hacker´s Handbook. 2nd ed. Indianapolis : John Wiley & Sons, Inc, 2011