GRAPHWAVE

An efficiency improvement for automated vulnerability scans

Tijme Gommers
Author/Graduate
Northwave

Martijn Hoogesteger
Company Supervisor
Northwave

Marco Marcellis
University Supervisor
AUAS

Prof. Eric Filiol
University Supervisor
ESIEA

GRAPHWAVE

An efficiency improvement for automated vulnerability scans

Amsterdam University of Applied Sciences
HBO-ICT Software Engineering

École D'ingénieurs du Monde Numérique
MASTER Information Security

Northwave BV
Red-Team Department
Marconibaan 49
3439 MR Nieuwegein

Internship Period
05/02/2018 to 03/08/2018

Tijme Gommers
Author/Graduate (AUAS/ESIEA)
Student^{nr} 500708891
tijme.gommers@hva.nl

Martijn Hoogesteger
Supervisor (Northwave)
Teamlead CERT
martijn.hoogesteger@northwave.nl

Marco Marcellis
Supervisor (AUAS)
Software Engineering Lecturer
m.m.c.m.marcellis@hva.nl

Prof. Eric Filiol
Supervisor (ESIEA)
Head of ESIEA CVO Lab
eric.filiol@esiea.fr

Nieuwegein, the Netherlands - May 31, 2018
Version 1.98.2-b9366ca3

Summary

In a world where desktop applications are shifting more and more towards web applications, web application security is a rising concern. Manually testing these applications for security vulnerabilities, such as cross-site scripting and SQL injection, is infeasible; as security specialists are scarce. Automated Web Application Vulnerability Scanners, hereinafter referred to as scanners, are tools that scan web applications in an automated way, normally from the outside, to look for these security vulnerabilities.

The advantage of scanners is that they can replace parts of the manual security vulnerability scanning. However, a major disadvantage is that they tend to be inefficient depending on the type of web application that is being scanned. Reviews about scanners by Northwave, a security company based in Nieuwegein, the Netherlands, and the online security community confirm that not all scanners are efficient. They state that in some cases scanners may take up to 36 hours to complete. However, it is not clear why they are inefficient or how the efficiency can be improved. It is important to know that efficiency is essentially just the speed of the scanners. Effectiveness is how many vulnerabilities the scanners find.

The goal of this study was to create a generic open-source solution that improves the efficiency of scanners while keeping the effectiveness of the scanners at the same level. Three scanners were used to test possible solutions, namely; Burp Suite, Acunetix & NYAWC. These scanners were used to test possible solutions since they all provide functionalities to modify their own behaviour. Testing more than three scanners would simply take too much time.

Research on four key concepts, originated from the intersection of all of the key concepts of the three scanners, indicate that "target scope reduction" can improve the efficiency the most while maintaining effectiveness. Therefore, "target scope reduction" is chosen as the most promising key concept. Although it is easy to manually reduce the target scope, it is more complicated to automate that process since the effectiveness needs to be maintained and automation does not know how to reduce the scope effectively.

Reducing the target scope can be done by ignoring similar HTTP responses. If two HTTP responses have the same HTML structure it is likely that they have the same code flow and therefore the same security vulnerabilities. By measuring the similarity of HTTP responses it is possible to automatically reduce the target scope by removing the similar (and therefore redundant) HTTP responses.

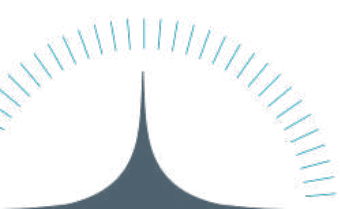
A technology that can be applied to detect similar HTML structures from HTTP responses is graph theory. Graph theory can enable the scanners to detect similar HTTP responses by adding all incoming HTTP responses to the graph except the HTTP responses that have similar aspects as the HTTP responses that are already in the graph. Since scanners do not have access to

the code of the web application they are scanning, the effectiveness can never be maintained with a 100% certainty.

The graph theory technology can be integrated into Burp Suite in a user friendly manner as an extension. After all the similar HTTP responses have been detected, the extension enables the user to either start scanning for vulnerabilities on all the unique HTTP requests or to export the unique HTTP requests to a text file and continue scanning for vulnerabilities in a scanner of choice. This makes it a generic open-source solution.

Target scope reduction using graph theory, in the form of a Burp Suite extension, is a user-friendly solution that improves the efficiency of scanners up to 45% while maintaining the effectiveness rate above 87% on average. Maintaining a 100% effectiveness is not always possible since scanners do not have access to the code of web applications they are scanning.

GraphWave is the name of the final product of this thesis. The name is a mix of the words 'Northwave' and 'graph theory'. It was published on the 14th of May, 2018 and since that day it is being used by security professionals around the globe.



Contents

Terms and acronyms

Glossary

Acunetix Acunetix is the market leader in automated web application security testing, and is the tool of choice for many Fortune 500 customers. Acunetix Vulnerability Scanner detects and reports on a wide array of web application vulnerabilities (Naudi, 2016). 1, 6, 8, 10, 20, 21, 26, 38

Arachni Arachni is a feature-full, modular, high-performance Ruby framework aimed towards helping penetration testers and administrators evaluate the security of web applications (Sarosys LLC, 2018). 6

Burp Suite Burp Suite is an integrated platform for performing security testing of web applications. It is not a point-and-click tool, but is designed to be used by hands-on testers to support the testing process (PortSwigger, 2018). 1, 6, 8, 10, 11, 16, 18–21, 26–30, 32, 38

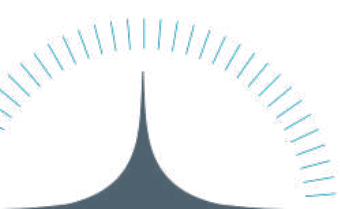
CFG-path A control flow graph (CFG) is a representation of all paths that might be traversed through a program during its execution (Kornblum, 2006). The CFG-path is one of the paths from the control flow graph. 21–25, 28

closed-source Closed-source software is software developed by someone (typically an organisation or company). That user can provide it to the public as a service, but does not release it to the public as source code (Free Software Foundation, Inc, 2018). 10, 17, 30

crawl Crawling or spidering is the act of systematically browsing a web application for the purpose of indexing its URLs or HTTP requests/response pairs (Kobayashi and Takeda, 2000). 24

Northwave Northwave is a company that specialises in an intelligent, integrated approach to security (Northwave, 2018). It is the company where this study was conducted. 1, 6–12, 21, 22, 28, 32, 34

open-source Open-source software is software that comes with permission for anyone to use, copy, and/or distribute, either verbatim or with modifications, either gratis or for a fee. In particular, this means that



the source code must be available (Free Software Foundation, Inc, 2018). 6–8, 10

payload A payload is a text snippet or a large binary object that contains malware such as worms or viruses which performs a malicious action. 10, 11

red-team The Northwave red-team tests security to the full extent to see if they can effect confidentiality, integrity and/or availability. They do this both digitally and through personal approaches (Northwave BV, 2018). 6, 7, 10, 21, 32

RESTful URL RESTful URLs or clean URLs are links to web applications that are meaningful and do not change over time (W3C, 1998). 29, 33

run-time The time during which a computer program is executed (Blumofe et al., 1995). 6, 11, 13

vulnerability A vulnerability is a weakness in an application, which can be a design flaw or an implementation bug, that allows an attacker to cause harm to the confidentiality, integrity and/or availability of the application (OWASP, 2016). 6, 8, 11, 14, 16, 17, 24, 27

Wireshark Wireshark is the world's foremost and widely-used network protocol analyser. It lets you see what is happening on your network at a microscopic level and is the de facto standard across many commercial and non-profit enterprises, government agencies, and educational institutions (Wireshark Foundation, 2018). 20, 38

Acronyms

API Application Programming Interface. 2, 10, 18, 26, 32, 34

HTTP Hypertext Transfer Protocol. 1, 8, 12, 19–21, 24, 25, 27, 30, 32, 38

NYAWC Not Your Average Web Crawler. 1, 6–8, 10, 11, 18, 26

scanner Automated Web Application Vulnerability Scanner. 1, 2, 6–8, 10–14, 16–19, 21, 23, 24, 26–29, 32–34, 40, 44

TTFB Time To First Byte. 1, 11, 12

Conventions

The key words "must", "must not", "required", "shall", "shall not", "should", "should not", "recommended", "may", and "optional" in this document are to be interpreted as described in RFC 2119 (Bradner, 1997).

1. **Must**: this word, or the terms "required" or "shall", mean that the definition is an absolute requirement of the specification.
2. **Must not**: this phrase, or the phrase "shall not", mean that the definition is an absolute prohibition of the specification.
3. **Should**: this word, or the adjective "recommended", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
4. **Should not**: this phrase, or the phrase "not recommended" mean that there may exist valid reasons in particular circumstances when the particular behaviour is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behaviour described with this label.
5. **May**: this word, or the adjective "optional", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

1 Introduction

In a world where desktop applications are shifting more and more towards web applications, web application security is a rising concern. Manually testing these applications for security vulnerabilities such as cross-site scripting and SQL injection is infeasible; as security specialists are scarce.

Automated Web Application Vulnerability Scanners, hereinafter referred to as scanners, are tools that scan web applications in an automated way, normally from the outside, to look for these security vulnerabilities (OWASP, 2018).

Acunetix and Burp Suite are two commercial scanners that Northwave, a security company based in Nieuwegein, the Netherlands, makes use of at the moment. However, using them is not as easy as it sounds. At this moment they are not satisfied with the efficiency of the scanners.

An open-source variant of the two scanners is Not Your Average Web Crawler (NYAWC). Northwave is planning to use NYAWC too. However, their first priority is optimising the efficiency of the scanners while maintaining effectiveness.

It is important to know that efficiency is essentially just the speed (or runtime) of the scanners. Effectiveness is how many vulnerabilities the scanners find. Both of these terms are covered extensively in the theoretical framework (chapter 2.3).

1.1 Problem

Reviews about scanners by Northwave and the online security community show that not all scanners are efficient. However, it is not clear why they are inefficient, or how the efficiency can be improved. The current situation at Northwave is that scanners may take up to 36 hours to complete.

But this is not only a problem in the Northwave red-team. There are many articles available online that show security researchers complaining about the time that it takes to successfully finish a scan on certain web applications. *'The Burp Suite active Automated Web Application Vulnerability Scanner (scanner) is running very slow'* (Vurtis, 2016). *'I left Arachni scanning for more than one day before stopping it myself'* (Ogri, 2017).

Scanners that take too much time are either aborted or performed incompletely. This causes annoyance and delays in the Northwave red-team and the security community.

1.2 Motivation

The red-team security audit is an important part of the services that Northwave provides. During the audit, automated and manual security tests are performed on web applications of the client. Scanners are important because they enable the red-team to focus on complex vulnerabilities during manual tests instead of the easily detectable bugs that the scanners focus on.

By making the results of this research and a possible technology to improve the efficiency open-source, they will be available for everyone who is interested. This means open-source tools like NYAWC can use the results to improve their efficiency.

1.3 Relevance

The final product will not directly yield any revenue and it will not decrease revenue either. However, some improvements may cause indirect increment of the revenue. Besides that, there is a practical usefulness of a solution for Northwave, namely:

1. that it decreases the annoyance in the red-team.
2. that it improves the efficiency of the scanners.
3. that the red-team does not stop a scan because it is inefficient.
4. that it increases customer satisfaction due to faster result delivery.

Point three is a situation where the final product may cause indirect increment of the revenue. Scans that take 36 hours are not stopped, but there are edge cases where scans take weeks or even months to finish. In these cases Northwave stops the scans and starts to manually test the target. This costs time and decreases the revenue. If the final product decreases the time it takes to successfully finish a scan it could prevent Northwave from stopping the scan. In that case, the amount of hours that would normally be spent on manually testing a target can be marked as additional revenue.

1.4 Goal

The ultimate goal is to provide Northwave and the security community with a generic open-source solution that enables them to improve the efficiency of scanners while keeping the effectiveness of the scanners at the same level.

1.5 Questions

Main question:

Which user friendly product can be developed to improve the efficiency of Automated Web Application Vulnerability Scanners while maintaining effectiveness?

Sub-questions:

1. Which key concept of the Automated Web Application Vulnerability Scanners is the most promising to improve efficiency?
2. Which technology can be used to improve the most promising key concept in an automated way?
3. In which user friendly way can the most efficient and effective technology be integrated with Automated Web Application Vulnerability Scanners?

1.6 Scope

The only scanners that are in scope are Automated Web Application Vulnerability Scanners using the Hypertext Transfer Protocol (HTTP) that have the functionality to improve or modify the program in such a way that the efficiency can be improved. A maximum of three scanners will be used to test new technologies. Two of them are the scanners used by Northwave (Burp Suite and Acunetix) and the remaining one is the open-source variant of those two scanners: Not Your Average Web Crawler.

Testing more than three scanners would take too much time and testing less than three scanners would result in less accurate results. The three scanners that will be used are extensively covered in chapter 2.1.

It is beyond the scope of this study to examine how to apply improvements that cannot be applied in a generic manner (meaning they cannot be applied on all of the scanners in scope).

Effectiveness is related to efficiency, however, the technology that will be used should not improve the effectiveness of the scanners (and neither should it worsen it). The reason for this is that scanners are exhaustive and therefore already find a 100% of the vulnerabilities they are programmed to find. Besides that, the effectiveness of the scanners currently used by Northwave is high enough.

1.7 Context

This study was conducted at Northwave, as the final part of my Bachelor study Software Engineering at the AUAS. During my study I did an international Master in Information Security at ESIEA (located in France) because I

love doing security related research and development. Both my Bachelor at AUAS and my Master at ESIEA required me to do a graduation internship and write a thesis to finalise my studies. This is the main reason why my thesis is written in English instead of Dutch.

Northwave is a security company based in Nieuwegein, the Netherlands, and consists of approximately seventy employees. They focus on intelligent security operations, which includes devising and implementing adequate solutions for their customers. Northwave mainly serves medium to large sized companies. These are companies that have to deal with the security problems of large enterprises, but do not have the resources to organise this internally. Northwave fills this gap by providing security services tailored to these types of companies. The organisation can be divided into two business units that both have their own tasks, Northwave Business Security (NBS) and Northwave Cyber Security (NCS).

Northwave Business Security

NBS is responsible for the strategic and tactical security level of their customers. This includes implementing security standards like ISO/IEC 27001 or improving the business continuity and recovery plans. NBS mostly provides tailored services to its customers, based on the outcome of risk assessments that they or the customer did.

Northwave Cyber Security

NCS has the responsibility over the operational security level of their customers. This includes doing penetration tests to advise on physical and software security improvements, but also monitoring the network traffic of their customers and responding to attacks on the network.

Amsterdam University of Applied Sciences

The Amsterdam University of Applied Sciences (AUAS) or Hogeschool van Amsterdam (in Dutch) is based in the Netherlands, in the city of Amsterdam.

Marco Marcellis is a lecturer at the AUAS and he will be one of my university supervisors during the internship.

École Supérieure D'informatique, Électronique, Automatique

École Supérieure D'informatique, Électronique, Automatique is a French university for engineers based in France, in the city of Laval.

Eric Filiol is a professor at ESIEA and he will also be one of my university supervisors during the internship.

2 Theoretical framework

These are the scanners, key concepts and measurements of this study, explained using academic resources or information provided by the creators of the aforementioned.

2.1 Automated Web Application Vulnerability Scanners

The three Automated Web Application Vulnerability Scanners defined in chapter 2.1.1, 2.1.2 and 2.1.3 are used to test a possible new technology since they all provide functionalities to modify their own behaviour. At Northwave, Burp Suite (2.1.1) and Acunetix (2.1.2) are the two most used scanners.

2.1.1 Burp Suite

Burp Suite is a closed-source graphical tool for testing the security of web applications. It has a powerful API that allows extensions to customise Burp Suite's behaviour, integrate with other tools and apply technologies (PortSwigger, 2017b).

2.1.2 Acunetix

Acunetix is a commercial closed-source graphical tool for testing the security of web applications. Northwave uses Acunetix as their primary scanner.

Acunetix has functionalities to modify its behaviour (e.g. changing multi-threading configurations or changing the way it decides on which URLs to scan). This makes it possible to integrate new technologies in Acunetix (Darmanin, 2017).

2.1.3 Not Your Average Web Crawler

NYAWC is an open-source command line variant of the scanners defined in chapter 2.1.1 and 2.1.2. It enables red-teams to execute payloads on attack vectors of a previously defined scope of the web application through the use of callbacks in the crawler. NYAWC can be used for testing since it provides

functionality to modify its behaviour (and apply a technology) through those callbacks (Gommers, 2017b). However, NYAWC does not provide payloads by default, therefore it cannot be used when testing payloads or vulnerabilities.

2.2 Key concepts

The following chapters describe an intersection of all the key concepts from scanners that are used by, and determine the efficiency and/or run-time of the scanners, based on their documentation.

2.2.1 Target scope (reduction)

Before starting a scanner, e.g. Burp Suite, it requires a scope. A scope is essentially what hosts and URLs constitute the target for the current work. The target scope is, roughly, the requests that are currently interesting and that need to be scanned (PortSwigger, 2017d). The bigger the scope, the longer the run-time of the scanner will be.

All the scanners (defined in chapter 2.1) have options to reduce the scope of the scanning run-time. Reducing the scope means less requests will be scanned (which reduces the run-time).

2.2.2 Multi-threading

The three scanners that will be tested all have the functionality to tweak the threading behaviour. Multi-threading enables computers to run multiple tasks at the same time instead of waiting for tasks to finish. This shortens the run-time.

Each thread (task) gets its own time slice, so each thread represents one basic unit of processor utilisation. Multi-threading is simultaneously executing multiple threads (Intel Corporation, 2003, p. 6).

2.2.3 Time To First Byte

The Time To First Byte (TTFB) is the time spent waiting for the initial response of the server after making a request. This time captures the latency of a round trip to the server in addition to the time spent waiting for the server to deliver the response (Garbee, 2018).

Scanners can scan web applications locally or remotely. The average TTFB should be around 200ms according to Google (Google LLC, 2018). This only applies to remote servers though, since locally hosted web applications do not have a round trip to a remote server.

According to Martijn Hoogesteger, Security Specialist at Northwave, the lead time of getting a customer's web application to work locally is ten

hours on average. It takes ten hours because the environment that the web application runs in must be an exact replica of the production environment. Besides that, the customer's web applications have not been developed by Northwave and therefore the environment specific bugs take a long time to fix.

Theorem 2.2.3.1 (TTFB is negligible) *For Northwave, the amount of requests R that scanners have to scan locally before gaining a time benefit compared to scanning remotely is: the amount of milliseconds it takes Northwave to setup the customer's web application divided by the average amount of milliseconds a remote request takes. It can be formulated using discrete mathematics as follows (Rosen, 2012, p. 94).*

$$R(h,\ t) = (\bar{h} \cdot 60 \cdot 60 \cdot 1000) \,/\, \bar{t}$$

In this formula R is the amount of requests that scanners has to scan. The arguments are h (a set of hours it takes to setup the customer's web application) and t (a set of TTFBs in milliseconds).

Using Theorem 2.2.3.1 it can be proven that any performance improvement through TTFB is negligible for Northwave. The average amount of requests that scanners need to scan locally before gaining a time benefit compared to scanning remotely is a 180.000, which is much more than the average amount of requests web applications from Northwave customers have.

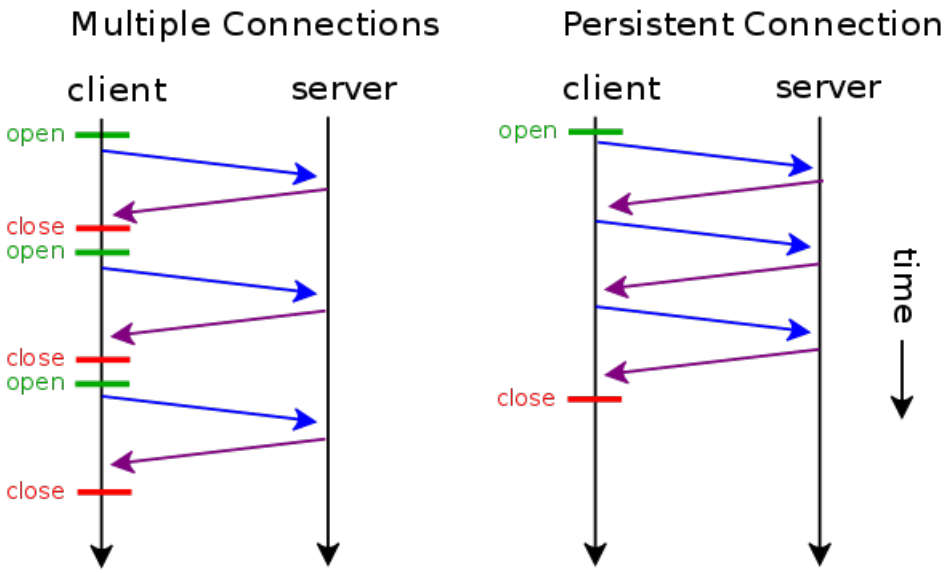
$$R(\{10\ hours\},\ \{200\ milliseconds\}) = 180.000$$

2.2.4 Persistent HTTP connections

HTTP is a request-response (client-server computing model) protocol used by many scanners (W3C, 1999, p. 7). The scanner is the client and the web application being scanned is the server (PortSwigger, 2017c).

As seen in Figure 2.1, prior to persistent connections, a separate TCP connection was established to fetch each URL, causing congestion on the Internet. Persistent HTTP connections have the advantage that they allow requests and responses to be pipelined on a single connection. Pipelining allows a client to make multiple requests without waiting for each response, allowing a single TCP connection to be used much more efficiently, with much lower elapsed time (W3C, 1999, p. 44). This technique can be applied on scanners.

Figure 2.1: The difference between multiple connections and a persistent connection (Helix84, 2006).



2.3 Measurements

The terms 'efficiency', 'effectiveness' and 'promising' are important terms in this research. The goals is to improve the efficiency of a promising key concept, while maintaining effectiveness. Therefore these terms need to be defined and measured.

2.3.1 Efficiency

Efficiency is being able to work well and do what is necessary without wasting time, money or effort (Bloomsbury, 2015, p. 107). This usually means the shorter the run-time of an scanner, the more efficient it is.

Theorem 2.3.1.1 (efficiency measurement) *The efficiency Y of a scanner is the improved run-time compared to normal run-time, measured in percentage. It can be formulated using discrete mathematics as follows (Rosen, 2012, p. 94).*

$$Y(R_n,\ R_i) = \frac{100}{R_i} \cdot R_n$$

In this formula R is the run-time duration and the corresponding R_n is the old technology (normal) and R_i is the new technology (improved). The efficiency is 100% when the new technique has the same run-time duration as the old technique. An example can be found below.

$$R_n = 36\ hours,\ R_i = 12\ hours$$

$$Y(R_n,\ R_i) = \frac{100}{R_i} \cdot R_n$$

$$Y(R_n,\ R_i) = \frac{100}{12\ hours} \cdot 36\ hours$$

$$Y(R_n,\ R_i) = 300\%$$

2.3.2 Effectiveness

Effectiveness is being able to produce the required result (Bloomsbury, 2015, p. 107). Please note that there is a major difference between efficiency and effectiveness. One can be efficient without being effective, and vice versa.

Theorem 2.3.2.1 (effectiveness measurement) *The effectiveness S of a scanner is the amount of vulnerabilities an improved technology finds on average compared to a normal technology, measured in percentage. It can be formulated using discrete mathematics as follows (Rosen, 2012, p. 94).*

$$S(V_n,\ V_i) = \frac{100}{|V_n|} \cdot |V_i|$$

In this formula V is the set of vulnerabilities scanned and the corresponding V_n is the set of vulnerabilities from the old technology (normal) and V_i is the set of vulnerabilities from the new technology (improved). This means when $S(V_n,\ V_i) = 100$, the same amount of vulnerabilities are found. Since the tested scanners are exhaustive, S can never be higher than a hundred percent. An example can be found below.

$$V_n = \{vuln1,\ vuln2,\ vuln3,\ vuln4\},\ V_i = \{vuln1,\ vuln2,\ vuln3\}$$

$$S(V_n,\ V_i) = \frac{100}{|V_n|} \cdot |V_i|$$

$$S(V_n,\ V_i) = \frac{100}{|\{vuln1,\ ...,\ vuln4\}|} \cdot |\{vuln1,\ ...,\ vuln3\}|$$

$$S(V_n,\ V_i) = \frac{100}{4} \cdot 3$$

$$S(V_n,\ V_i) = 75\%$$

2.3.3 Promising

Something is promising when it shows signs that it is going to be successful or enjoyable (Cambridge, 2018). Improving a scanner is successful when the efficiency improves and the effectiveness is maintained. Therefore, the most promising key concept is the key concept that, in relation to other key concepts, improves efficiency the most while maintaining the effectiveness.

Theorem 2.3.3.1 (Promising measurement) *To see which key concept C can improve the efficiency the most while maintaining effectiveness (and which is*

therefore the most promising), the negative average percentage of effectiveness and the average percentage of efficiency are calculated. Since the efficiency is measured in seconds, the less time it takes, the more efficient it is. For effectiveness it is the reverse, the more vulnerabilities are found, the more effective it is. Therefore the negative average percentage of effectiveness is used. This means that the lower the outcome of the formula, the more promising the key concept is. It can be formulated using discrete mathematics as follows (Rosen, 2012, p. 94).

$$C(S,Y) = \frac{200 - (S_l + S_h)}{2} + \frac{Y_l + Y_h}{2}$$

In this formula S is the effectiveness, Y is the efficiency and the corresponding $_l$ and $_h$ are the low and high values. The low and high values arise when multiple scans are performed to measure the effectiveness and efficiency. An example can be found below.

$$S_l = 23,\ S_h = 27,\ Y_l = 371,\ Y_h = 464$$

$$C(S,Y) = \frac{200 - (S_l + S_h)}{2} + \frac{Y_l + Y_h}{2}$$

$$C(S,Y) = \frac{200 - (50)}{2} + \frac{835}{2}$$

$$C(S,Y) = 75 + 417.5$$

$$C(S,Y) = 492.5$$

3 Research design

This is an iterative research design based on the combined use of qualitative and quantitative research methodologies, as described in Niglas, 2004. Each subchapter (3.1, 3.2 & 3.3) describes how to gather and analyse data about the corresponding sub-question.

3.1 A promising key concept of the scanner

Sub-question one: Which key concept of the Automated Web Application Vulnerability Scanners is the most promising to improve efficiency?

3.1.1 Data gathering

Quantitative research, in the form of lab experiments, must be used to gather data about which key concepts are (in)efficient. The lab experiments can be conducted by using the measurement formulas defined in chapter 2.3 while changing the behaviour of the key concepts (by e.g. dividing the scope in two or increasing the amount of threads to use).

Example: For the key concept "target scope reduction" four scans must be performed in Burp Suite. The first one while dramatically reducing the scope, the second one while dividing the scope, the third while while slightly reducing the scope and the last one while not reducing the scope. This must be done for every web application that is listed below. Afterwards the other key concepts and scanners must be tested.

The three different (common) versatile types of web applications listed below will be tested. Testing more than three web applications would take too much time and testing less then three would cause the results to be less accurate.

1. **www.dvwa.co.uk**: a content web application containing forty different kinds of pages with, in total, 35 unique vulnerabilities.
    - This web application will be used for testing purposes since it contains many vulnerabilities and can therefore be used to test if a technology maintains the level of effectiveness.

2. **testasp.vulnweb.com**: a forum web application containing seventeen different kinds of pages with an unknown amount of vulnerabilities (since it is closed-source). The web application is meant to act and look as real as possible.

    - Since this web application contains many similar kinds of pages, scanners can be inefficient while scanning it. It will be used for testing purposes since it could be used to show major differences in efficiency while changing the behaviour of key concepts.

3. **demo.testfire.net**: a banking web application containing 51 different kinds of pages with an unknown amount of vulnerabilities (since it is closed-source). The web application is meant to act and look as real as possible.

    - This web application is a banking web application that contains many different functionalities. It is a common type of web application on the internet, a mixture between the first two web applications. Therefore this application represents many real-life cases and will be used for testing purposes.

The setup for testing the web applications is an identical virtual environment (e.g. VirtualBox). It does not matter what kind of environment (as long as it is identical for every test) since the technologies are measured relative to the results without using that technology, as explained in chapter 2.3.

3.1.2 Analysis method

The data that should be analysed are the effectiveness and efficiency before and after behavioural changes, for each key concept, web application and scanner. It be analysed empirically (by comparisons) using theorem 2.3.3.1 in chapter 2.3.3, which outputs how promising a key concept is. The lowest result is the best and is therefore the most promising key concept. It must be used in sub-question two (a technology to improve the efficiency).

3.2 A technology to improve the efficiency

Sub-question two: Which technology can be used to improve the most promising key concept in an automated way?

3.2.1 Data gathering

A mix between qualitative and quantitative field research must be used for data gathering. The most promising key concept is the key concept which has the most varying efficiency when changing its behaviour, as defined in chapter 3.1. To find out which technologies can be developed and/or applied to improve the most promising key concept, various technologies need to be investigated. For example, using a blocking queue with the producer consumer principle instead of starting threads for every request

(like NYAWC does, Gommers, 2017c). These technologies can come from own experience or the experience of other security or software engineers.

For every technology a prototype must be developed. Using these prototypes the required data, about the effectiveness of the prototype, can be gathered and therefore one knows what the most efficient and effective technology is.

3.2.2 Analysis method

The data that should be analysed is the amount of supported scanners that are compatible with the technology and how effective each scanner is when using the prototypes. This can be analysed empirically (by comparisons). To see which technologies can be used the researcher must select the technologies that are compatible with all the scanners. This can be verified by checking the (API) documentation of the respective scanner. To see which prototype is the most efficient and effective, the theorems 2.3.1.1 and 2.3.2.1 can be used.

3.3 A user friendly product

Sub-question three: In which user friendly way can the most efficient and effective technology be integrated with Automated Web Application Vulnerability Scanners?

3.3.1 Data gathering

Quantitative field research must be used to gather data about how user-friendly a technology is. The user-friendliness is the amount of additional user interactions in combination with the additional seconds it takes for the user to start a scanner using the new prototypes, compared to not using them. User interactions may include e.g. mouse clicks in graphical user interfaces and/or commands in command line interfaces.

Example: Without the prototype, starting a scan in Burp Suite may take around ten seconds and five user interactions. With the prototype, starting a scan in Burp Suite may take around twenty seconds and eight user interactions. This means that the prototype requires the user to spend ten additional seconds and three additional user interactions.

3.3.2 Analysis method

The data that should be analysed are the amount of additional user interactions and elapsed seconds while starting the scanner. It can be analysed empirically (by comparisons). To test in what user-friendly way the technology from sub-question two (a technology to improve the efficiency) can be used, the solution with the least possible user interactions in combination with elapsed seconds should be chosen.

4 Research results

The research results are originated from the research design and therefore they are also iterated per sub-question, meaning every sub-question has its own chapter.

4.1 A promising key concept of the scanner

> Sub-question one: Which key concept of the Automated Web Application Vulnerability Scanners is the most promising to improve efficiency?

The results of the data gathering, defined in the research design (chapter 3.1), can be found in Appendix A. The vertical axis contains the three demo web applications and the type of scans performed on these web applications. The horizontal axis contains the two scanners that can be scanned and their effectiveness and efficiency noted relative (as a percentage) and absolute.

Some columns state N.A. This means that the relevant technology is not available or modifiable in the scanner. For example, it turned out that persistent HTTP connections are not working in Burp Suite. Even if the `Connection: keep-alive` header is added, Burp Suite ignores it (as visible in Figure 4.1 and 4.2).

`Connection: keep-alive` is a response header which tells the application, in this case Burp Suite, to use persistent HTTP connections.

Figure 4.1: A Wireshark capture of request/response pairs, including the source ports which are notably different for every request. Since the source ports are different the TCP connections were not persistent, which proves that persistent HTTP connections do not work in Burp Suite. If it did work, the source ports would have been identical.

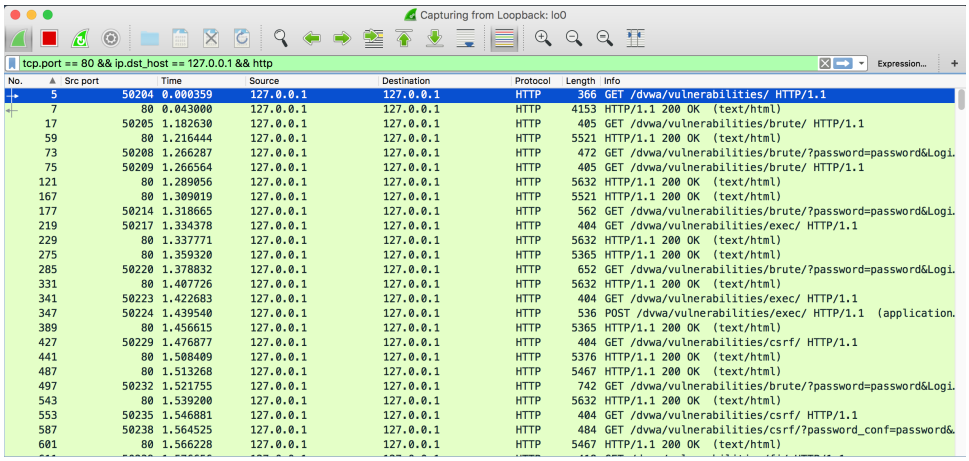
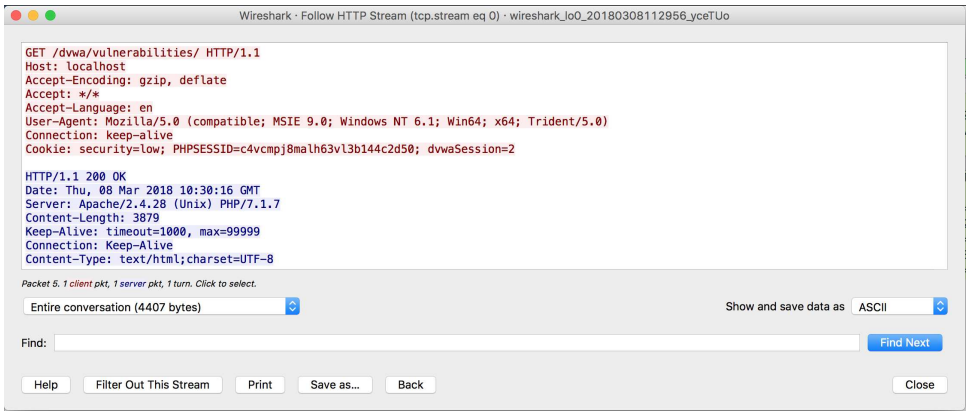
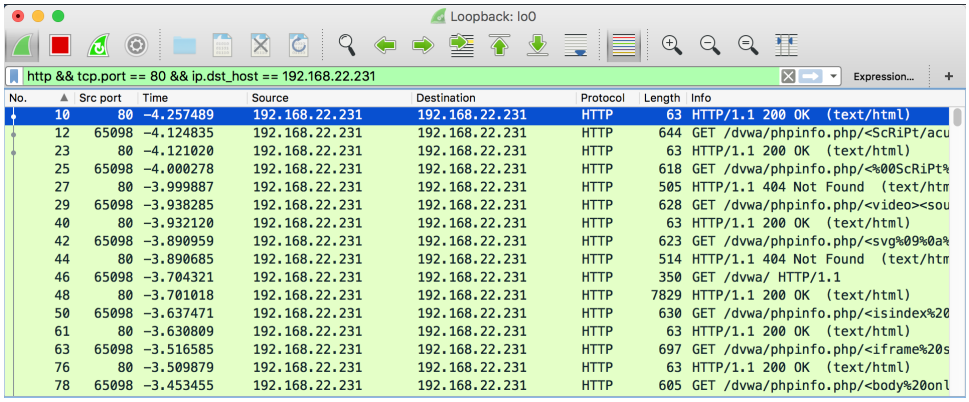


Figure 4.2: An HTTP stream of a request from Figure 4.1 showing that Burp Suite **did** use the correct connection header.



Acunetix does the opposite. The persistent HTTP connections cannot be disabled unless the server does not support it (as visible in Figure 4.3).

Figure 4.3: A Wireshark capture of request/response pairs that show persistent HTTP connections in Acunetix since the source ports are identical.



Different results can be seen in the multi-threading rows from Appendix B. The reason for this is that a scan without modification already uses multi-threading in most scanners. For example, in Acunetix the multi-threading options are set to the highest available option by default (which means modifying the behaviour only makes it slower). For Burp Suite the amount of threads to use is set to ten by default. This means it can not only be adjusted to a lower value but also to a higher value.

Conclusion: The results indicate that "Target scope reduction" can improve the efficiency the most (up to 45%) while maintaining effectiveness when using the formula in chapter 3.1.2. Therefore, "Target scope reduction" is the most promising key concept.

4.2 A technology to improve the efficiency

Sub-question two: Which technology can be used to improve the most promising key concept in an automated way?

The efficiency of scanners can be improved by the use of target scope reduction. However, the results from sub-question one (a promising key concept of the scanner), as visible in Appendix A, also indicate an effectiveness decrease of up to 18% for the high values. By looking at the ignored requests from the tests from sub-question one, one can see that the effectiveness is maintained when similar code flows are ignored, but decreases when code flows that are not similar are ignored.

A code flow can be denoted as a CFG-path (a path in a code flow graph). When performing an HTTP request to a server, the server executes a piece of code and returns the response. Different HTTP requests can cause the server to execute different pieces of code (different code flows). This means when similar HTTP requests are sent to the server, the server could execute the same code flow (Bulkar, 2016).

If two HTTP responses have the same HTML structure it is likely that they have the same CFG-path and therefore the same security vulnerabilities. By measuring the similarity of HTTP responses it is possible to automatically reduce the target scope by removing the similar (and therefore redundant) HTTP responses. Unfortunately the red-team at Northwave does not have access the the code of the web applications they are scanning. Therefore CFG-paths cannot be read and the proposed technologies can **not** be 100% accurate.

The technologies proposed below are technologies that should be able to detect the CFG-path from requests/responses with a high probability so that similar ones can be ignored. This will increase efficiency and keep the effectiveness at a high rate. The technologies are also all compatible with the scanners, as defined in Chapter 3.2 (PortSwigger, 2017a and Acunetix, 2017 and Gommers, 2017a).

4.2.1 Technologies

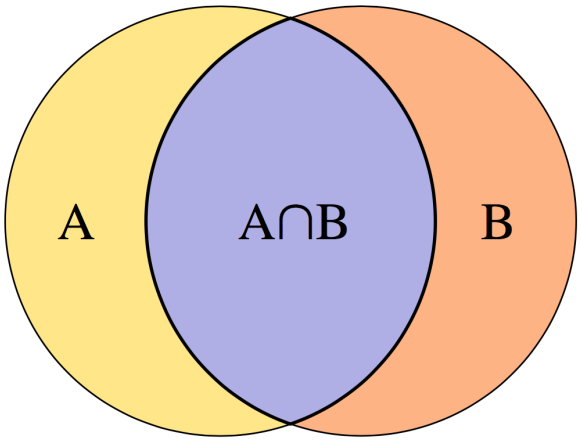
4.2.1.1 HTML tree similarity measure

A technology proposed by the development team of Northwave is measuring the HTML and/or URL tree similarity between responses by the use of the Levenstein or Jaccard algorithm.

Levenshtein distance is a measure of the similarity between two strings, a source string and a target string. The distance is the number of deletions, insertions, or substitutions required to transform the source into the target (Gilleland, 2006).

The Jaccard similarity coefficient is a statistical measure of similarity between sample sets. For two sets, it is defined as the cardinality of their intersection divided by the cardinality of their union (Bank and Cole, 2008a).

Figure 4.4: An intersection of two sets, $A \cap B$, (Kulla, 2015).



Similar CFG-paths output similar HTML, however, the content in the HTML (technically called node values) can be different (it can be different when e.g. reading a different row from a database). Therefore, to detect similar CFG-paths, one should only look at the HTML nodes and not at what text they contain (the node values).

Both of the algorithms described above can be used to detect similarities in (HTML) trees and can therefore detect similar CFG-paths.

4.2.1.2 Piecewise response hashing

Another technology proposed by the development team of Northwave is piecewise hashing on certain parts of the HTML and/or URL tree.

Piecewise hashing is a hashing method commonly used in digital forensics. It uses an arbitrary hashing algorithm to create many checksums for a file instead of just one. Rather than generating a single hash for the entire file, a hash is generated for many discrete fixed-size segments of the file. For

example, one hash is generated for the first 512 bytes of input, another hash for the next 512 bytes, and so on (Bank and Cole, 2008b).

Figure 4.5: An example of similar signatures outputted by piecewise hashing. The two hashed values are 99% similar.

```
signature:          768:ZoLymAAjaHx/4DpIXYSEAdP0Pn0nxqgeFjviVHeFc:KCHOtSin0nrelviNeK
signatureToCompare: 768:asdfmAAjaHx/4DpIXYSEAdP0Pn0nxqgeFjviVHeFc:asdftSin0nrelviNeK
Similarity:         99
```

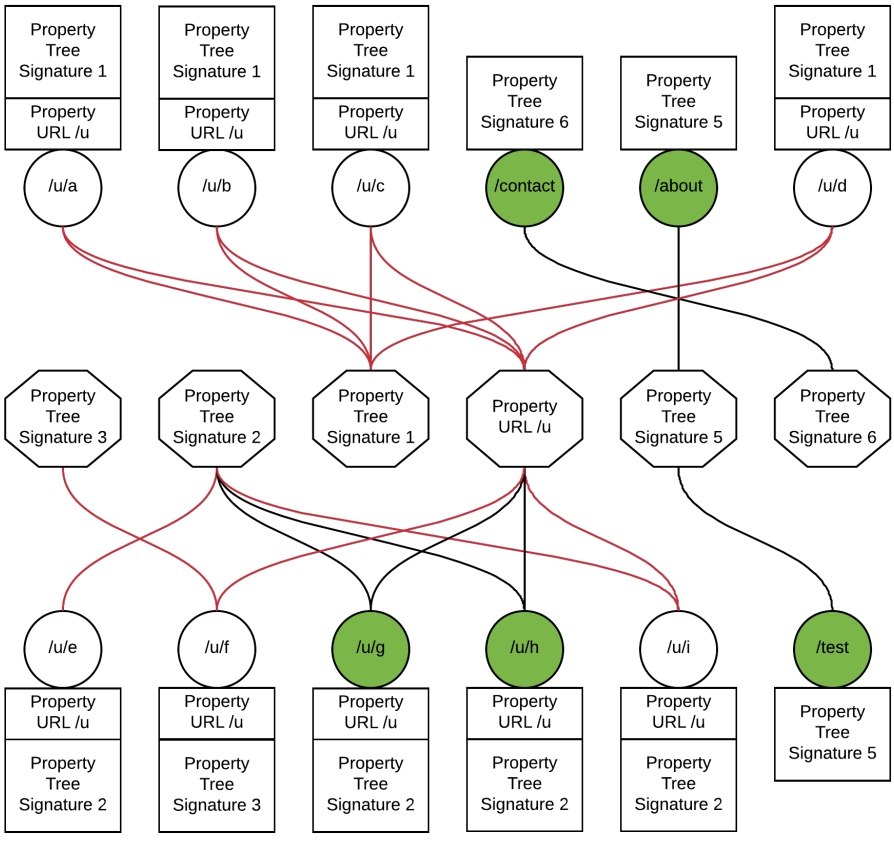
Piecewise hashing can be used to detect similarities in (HTML) trees and can therefore detect similar CFG-paths.

4.2.1.3 A self-developed undirected graph

The last technology is self-developed in the form of an undirected graph, as visualised in Figure 4.6.

- The circles are nodes which represent responses from the scanner.
- The octagons are nodes which represent properties of the responses (e.g. similar URL structures).
- The black and red lines are edges which represent which responses belong to which properties.

Figure 4.6: The self-developed graph cut visualisation.



When a scanner crawls a web application the graph can be built in realtime by looking what kind of properties the response has. Afterwards (or while building the graph), the following algorithm can be used to determine which responses should be further investigated (e.g. detecting if it has vulnerabilities).

1. Parse a response and generate all properties that identify it.
2. Iterate over all those properties;
   (a) If the property does not exist in the graph; `continue`.
   (b) If the property exists in the graph but has less than x edges; `continue`.
   (c) Otherwise; keep track of how much this property identifies the response (this is called the weight).
3. If the sum of all matching property weights is higher than or equal to the threshold y, similar responses already exist in the graph and thus this response can be ignored.
4. If the sum of all matching property weights is lower than the threshold y, the response should be added to the graph. The properties of the response should also be added if they are not in the graph yet. After this the response should be linked to its properties using the edges.

In this case, x is the sum of flows to properties and y is the minimum threshold for scanning similar items. This means that a response should have a minimum weight (the sum of x) to be marked as similar. And it means that a minimum of y similar responses will be scanned and after that further similar responses are ignored.

Using this algorithm only the green nodes from Figure 4.6 will be connected in the end. The white nodes are disconnected (their edges have been marked in red). Therefore, only the green nodes will be scanned and, as visible in Figure 4.6, the white nodes that will not be scanned are all similar.

This algorithm can be used to detect similarities in HTTP responses and can therefore detect similar CFG-paths.

4.2.2 Prototypes

A prototype has been developed for each technology. The results of the effectiveness measurements of these prototypes can be found in Appendix B. The vertical axis contains the three demo web applications and the technologies used to perform the scans on the web applications. The horizontal axis contains the effectiveness and efficiency noted relatively (as a percentage) and absolutely.

In contrast to the results from sub-question one (a promising key concept of the scanner), these results are not per scanner. Since the technology, scope reduction, is measured relative to the results without using that technology, this would result in identical results per scanner.

The results show divergent effectiveness percentages for the first web application (Damn Vulnerable Web Application), in relation to the other web applications. The reason for this is that this web application is designed in such a way that it does not have a lot of similar CFG-paths, therefore the efficiency can barely be improved.

The other two web applications are supposed to act as common web applications and do have some similar CFG-paths. This is reflected in the results.

Listing 4.1 is a code snippet that shows how some of the properties are built (in this case using the query parameters of URLs). For each key value pair in the parameter string properties are added to an array. Properties are e.g. if the parameter value is a number or a word.

Code Listing 4.1: An example of the properties (octagons) from URL queries in the self-developed graph cut technology

```
@staticmethod
def get_url_query_properties(response):
    properties = []

    parts = response.url.query.split(self.QUERY_DELIMITER)
    for index, part in enumerate(parts):

        # Exact Match
        properties.append(Property("u.q.exact[" + part + "][" + str(index) + "]", 0.15,  ↩
                                                    part))

        # Is number
        if PropertyGenerator.pattern_number.match(part):
            properties.append(Property("u.q.number[" + str(index) + "]", 0.15, None))

        # Is word
        elif PropertyGenerator.pattern_word.match(part):
            properties.append(Property("u.q.word[" + str(index) + "]", 0.05, None))

        # Is slug
        elif PropertyGenerator.pattern_slug.match(part):
            properties.append(Property("u.q.slug[" + str(index) + "]", 0.025, None))

    return properties
```

Conclusion: The self-developed graph cut technology from chapter 4.2.1.3 gives the best efficiency/effectiveness results on average. This technology combines multiple aspects of the HTTP responses. The efficiency improved with 48% while the effectiveness only decreased with 12% on average.

4.3 A user friendly product

Sub-question three: In which user friendly way can the most efficient and effective technology be integrated with Automated Web Application Vulnerability Scanners?

The results of the measurements can be found in Appendix C. The vertical axis contains the possible solutions on how to integrate the technology. The horizontal axis contains the three scanners that the solution was tested on.

Some columns state N.A. This means that the relevant solution is not available or capable as integration in the scanner.

4.3.1 An extension for every scanner

A solution is user-friendly if it is integrated with a scanner itself. As visible in Appendix C, a Burp Suite extension only requires two additional user interactions and only eleven additional seconds compared to using Burp Suite without the extension. Not Your Average Web Crawler (NYAWC) provides sufficient callbacks to bring the user interactions down to zero. Acunetix does not provide sufficient opportunities to integrate the prototype as an extension.

Integrating the prototype as a scanner extension may seem like a good solution; however, extensions are scanner specific. For example, an extension for Burp Suite cannot be used in NYAWC. This means it is nearly impossible to provide an extension to everyone who wants to use the prototype in a scanner, since it would take too much time to develop.

4.3.2 Crawler that outputs URLs to a file

Every scanner needs a start request. This request, sometimes referred to as the entry point, can be either a single URL or a list of URLs. A solution to use a prototype with a scanner is to build a separate crawler that crawls all URLs using the smart analysis of the prototype and outputs them to a text file. This text file can be imported to a scanner afterwards.

This means additional user interactions include managing the crawling process from the separate crawler and importing the results into the scanner. The results indicate five additional user interactions on average, which translates to 46 additional elapsed seconds.

4.3.3 Available as API

Another solution would be to expose an API so users can build an extension themselves. The results, as visible in Appendix C, look promising; however, there is one major downside. An extension has to be built for every scanner so it can communicate with the API. Some people may not have enough

knowledge to build an extension themselves, which is why this solution may be inaccessible for some users.

4.3.4 A scanner extension that outputs URLs to a file

To get the best results possible, a combination of two of the solutions above can be used. The prototype can be integrated into Burp Suite as an extension and have the possibility to either continue the scanning process in Burp Suite or to export the URLs to a text file and continue in a scanner of choice.

The vulnerability scanning process in Burp Suite is only available in the paid version; however, the crawling process is also available in the free version. Therefore this solution would be accessible for everyone.

The results indicate that this is the best solution since it is compatible with all scanners, since it does not require any development by the user and since it requires the least user interactions while keeping in mind the aforementioned.

Listing 4.3 and 4.3 are code snippets used in the prototype that demonstrate how the technology was implemented into Burp Suite.

Code Listing 4.2: The registration of an HTTP listener using the Burp Suite extender

```
1  class BurpExtender(IBurpExtender, ITab):
2
3      def registerExtenderCallbacks(self, callbacks):
4          http_listener = NorthwaveHttpListener()
5          http_listener._lock = Lock()
6
7          callbacks.setExtensionName("Prototype")
8          callbacks.registerHttpListener(http_listener)
```

Code Listing 4.3: The HTTP listener excluding requests if similar ones are already in the graph

```
1  class NorthwaveHttpListener(IHttpListener):
2
3      def processHttpMessage(self, toolFlag, messageIsRequest, requestResponse):
4          request = self._helpers.analyzeRequest(requestResponse)
5          html = self._helpers.bytesToString(requestResponse.getResponse())
6
7          self._lock.acquire()
8
9          added = graph.add_response(Response(request.getUrl().toString(), html))
10
11         if added == False:
12             config["target"]["scope"]["exclude"].append(request.getUrl().toString())
13
14         self._lock.release()
```

In Listing 4.3 a lock is used to prevent race conditions in the graph since the method `processHttpMessage` is asynchronous. Race conditions could cause similar request to be added to the graph while they should be ignored instead.

5 GraphWave

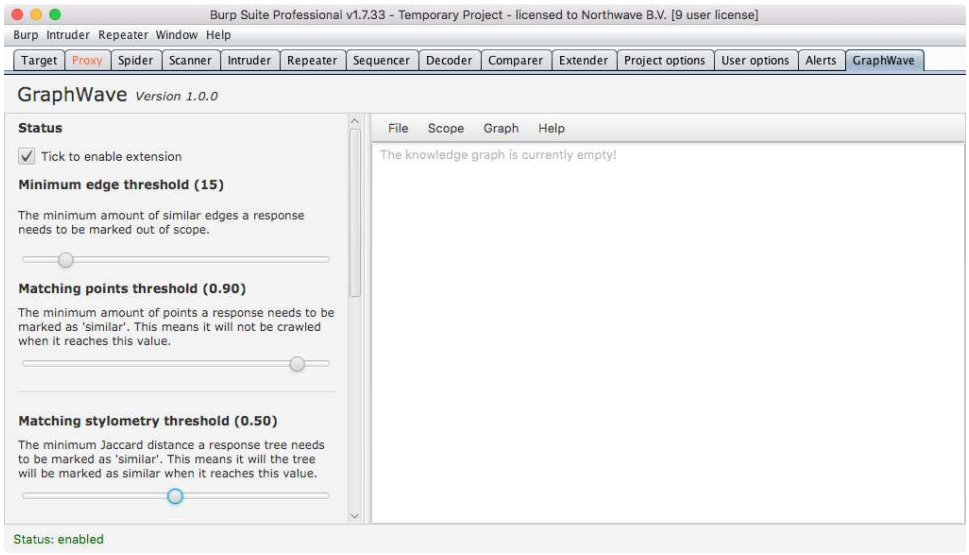
GraphWave is the name of the final product of this thesis. The name is a mix of the words 'Northwave' and 'graph theory'.

The first version of GraphWave was published to GitHub, a code collaboration platform, on the 14th of May, 2018 (`github.com/tijme/graphwave`). Since that day it is being used by security professionals around the globe. The figures and listings below demonstrate the use of GraphWave.

5.1 Workflow

Figure 5.1 is a print screen of the GraphWave tab in Burp Suite. It is integrated as an extension using the self-developed graph cut based on response properties to ignore similar CFG-paths. The integration in Burp Suite is based on the results from sub-question three (a user friendly product).

Figure 5.1: GraphWave overview in the Burp Suite tab.

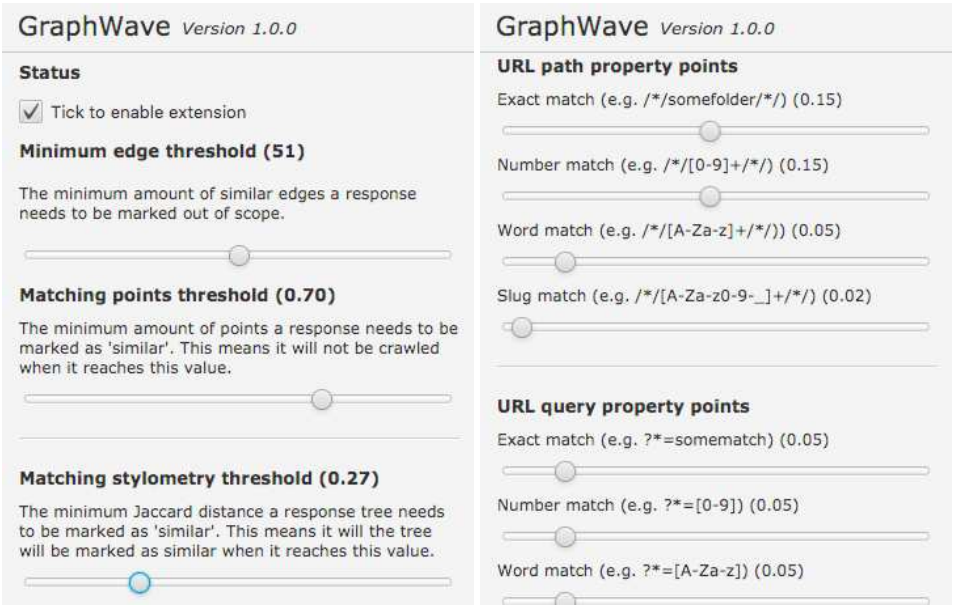


In the left column the settings are displayed. The settings are adjustable by ticking the checkboxes or dragging the sliders. The right side displays a list of responses that look similar. In Figure 5.1 the list is empty since the scanner did not start yet. However, when one starts the scanner the right column will update in realtime.

The default settings are sufficient for the tested web applications. However, a user can adjust the settings to increase the efficiency for specific web applications. For example, if a web application has RESTful URLs the URL path property points can be increased and the URL query properties points can be decreased.

Figure 5.2 shows all the settings that are available. These settings are extensively covered in Appendix D.

Figure 5.2: All the settings from GraphWave.



When the extension is enabled one can start a scan, which introduces the following workflow:

- **Reconnaissance phase:**
  1. Burp Suite scans the web application in scope (this is called spidering).
  2. GraphWave keeps track of every scanned request and does its analysis accordingly.
  3. The Burp Suite scan is completed after several minutes or hours.
  4. GraphWave shows the similar requests in the GraphWave tab.
- **Vulnerability scanning phase:**
  1. The user can mark the similar requests as 'out-of-scope' to continue in Burp Suite or the user can export the requests into another scanner.
  2. The user can start the vulnerability scanning on the requests that are in scope.

5.2 Internals

Burp Suite is a closed-source application written in Java. Extensions in Burp Suite can make use of Jython to enable software engineers to develop extensions for Java in Python. This is useful because Python is high level and enables the software engineers to rapidly develop extensions.

Listing 5.1 is a code snippet which demonstrates the use of Java classes in Python. For example the `FXMLLoader` is a JavaFX class which can load XML files to build GUIs (Oracle, 2015).

Code Listing 5.1: Loading FXML files using JavaFX

```
1  class BurpExtender(burp.ITab, burp.IBurpExtender):
2
3      def registerExtenderCallbacks(self, callbacks):
4          callbacks.setExtensionName(ExtensionDetails.TITLE)
5
6          self.layout = JFXPanel()
7
8          root = FXMLLoader.load(File(self.resourcePath + "extension_tab.fxml").toURL())
9
10         self.resourceScene = Scene(root)
11         self.layout.setScene(self.resourceScene)
12
13         callbacks.addSuiteTab(self)
14         callbacks.registerHttpListener(self.httpListener)
15
16         runnable = ExtensionRunnable(self.initializeElements)
17         Platform.runLater(runnable)
```

At the moment, Java provides two options to develop GUIs for desktop applications. One of them is Swing. Swing enables software engineers to build GUIs programmatically (using a programming language). Another way to build GUIs in Java is to use JavaFX. JavaFX enables software engineers to build GUIs in XML (a markup language). JavaFX is more convenient than Swing because it requires less code and because it is easier to reuse XML for multiple views. Therefore, GraphWave uses JavaFX (as seen in Listing 5.1).

Listing 5.2 is a code snippet that demonstrates how certain properties in the graph are generated based on URL paths. All properties are constructed using a weight that is retrieved from the default settings or the settings that the user inserted. The weights are used to measure if an HTTP response is similar to another HTTP response.

Code Listing 5.2: Properties being generated based on the URL path

```
@staticmethod
def getUrlPathProperties(path, options):
    properties = []

    for index, part in enumerate(path.strip("/").split("/")):

        if GraphWavePropertyGenerator.pattern_number.match(part):
            # Is number
            properties.append(GraphWaveProperty("url.path.number[" + str(index) + "]",
                                                options["upNumberMatch"], None))
        elif GraphWavePropertyGenerator.pattern_word.match(part):
            # Is word
            properties.append(GraphWaveProperty("url.path.word[" + str(index) + "]",
                                                options["upWordMatch"], None))
        elif GraphWavePropertyGenerator.pattern_slug.match(part):
            # Is slug
            properties.append(GraphWaveProperty("url.path.slug[" + str(index) + "]",
                                                options["upSlugMatch"], None))

    return properties
```

The method in Listing 5.2 iterates over all parts of a URL path (e.g. `/news/251/this-is-a-slug`). For each part, e.g. `this-is-a-slug` it generates multiple properties. If these properties also occur in other URL paths, there is the possibility that the corresponding responses are similar.

Listing 5.3 is a piece of markup language that is used to represent the adjustable settings in GraphWave. For example, the slider containing `fx:id="mctSlider"` represents the adjustable setting: 'matching points threshold'. The matching points threshold is the minimum amount of points a response needs to be marked as similar, as defined in Appendix D.

Code Listing 5.3: A piece of an FXML file that is rendered using JavaFX in Jython

```
<Label fx:id="mctLabel" GridPane.rowIndex="5" text="Matching points threshold">
    <font>
        <Font name="Verdana Bold" size="13.0" />
    </font>
</Label>

<Slider fx:id="mctSlider" snapToTicks="true" GridPane.rowIndex="7" />

<Separator prefWidth="200.0" GridPane.rowIndex="22" />

<TextArea fx:id="log" editable="false" GridPane.rowIndex="1" />
```

6 Conclusion

The goal of this research was to look for the answer to the question: 'Which user friendly product can be developed to improve the efficiency of Automated Web Application Vulnerability Scanners while maintaining effectiveness?'. A quantitative and qualitative study has been conducted to measure the effects of behavioural changes in three Automated Web Application Vulnerability Scanners. These measurements have been used to improve the efficiency using a new technology.

Efficiency, effectiveness and promising are important terms since the complete research is based on them. They can be measured using mathematical formulas, which makes it easy to replicate and validate this research.

Two key concept measurement results indicate that 'Target Scope Reduction' can improve the efficiency of scanners the most (up to 45%) while maintaining effectiveness. The key concept 'Persistent HTTP Connections' was not tested since both of the tested scanners do not have functionality to change the behaviour of the key concept. The key concept 'Time To First Byte' has also not been tested since the theory from chapter 2 shows that any improvement of this key concept is negligible since the time to change the behaviour of this key concept is more than it can improve.

Three prototypes based on the key concept 'Target Scope Reduction' have been developed, all of them based on measuring differences between HTTP responses so that similar ones can be ignored. The 'Self-developed Graph Cut' prototype is the most efficient prototype while maintaining a high effectiveness. The efficiency improved with 48% while the effectiveness only decreased with 12% on average.

User experience tests show that making the prototype available using an API is the most user-friendly solution. However, since this would be inaccessible for some consumers the second most user-friendly solution has been developed; A Burp Suite extension in the form of a crawler that can directly scan a web application from within Burp Suite. It also has functionality to output the URLs to a text file so they can be imported in a scanner of choice.

Target scope reduction using graph theory, in the form of a Burp Suite extension, is a user-friendly solution that improves the efficiency of scanners while maintaining the effectiveness rate above 88% on average. Maintaining a 100% effectiveness is not possible since the red-team of Northwave does not have access to the code of web applications they are scanning.

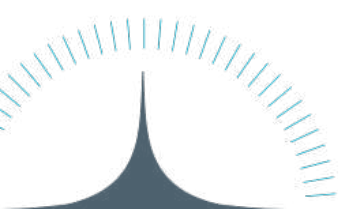
7 Discussion

For this research three different web applications and three different Automated Web Application Vulnerability Scanners have been used to test a new technology to improve efficiency of scanners. In case of a repeat of this research (using the same research methodology and technologies), the results would be the same.

The results of the tests showed that while some technologies improved the efficiency, the effectivity could not be fully maintained, as seen in Appendix B and C. This result is not in line with fully maintaining effectiveness, which was the goal of this research.

A possible explanation for the effectiveness loss is that every type of web application requires specific weight values in the graph properties. For example, an application that uses RESTful URLs should get more points for matching URL paths, in contrast to web applications that do not use RESTful URLs. This causes a major effectiveness drawback in e.g. Damn Vulnerable Web Application, as seen in Appendix C.

The advice for follow-up research is therefore to carry out a similar study to find out whether the technology can be tweaked in such a way that the graph options are based on the web application that is being scanned, so that the effectiveness remains at a higher rate.



Epilogue

Writing this thesis was a great pleasure. It was interesting and instructive. I am proud of the research and the product that I developed for Northwave as well as the security community.

Life at Northwave was great, especially because I was surrounded by professionals at all time. During previous internships I often had the feeling that colleagues did not have enough knowledge to help me. Northwave has been an internship where I was constantly amazed by the ingenious solutions that my colleagues came up with when I had a challenge. In particular, the development team helped me a lot with proof-of-concepts for possible techniques (e.g. piecewise hashing and Levenshtein distance on HTML trees).

In addition, my co-graduate Rik van Brakel made the stressful times bearable with his humour, advice and a good team spirit. For example, he introduced daily happy hours with snacks and drinks for all graduates at Northwave. I will always be grateful for the forward movements he introduced at Northwave.

During the research I thought that developing an algorithm would be difficult and implementing the technology in the scanners would be a piece of cake. Later it turned out to be the reverse. I developed an algorithm within a day that already had an efficiency improvement of 40%. Implementing that algorithm in a scanner of choice turned out to be difficult since there was little to no documentation of the APIs.

With regard to the thesis itself, I have benefited greatly from Martijn Hoogesteger's knowledge and the fact that he wrote his thesis not too long ago. Martijn has a lot of knowledge about this study and supported me well with setting up my research methodologies.

Tijme Gommers

May 31, 2018

Bibliography

Acunetix. (2017). Configuring acunetix to exclude scanning a portion of website. Retrieved February 21, 2018, from https://www.acunetix.com/blog/docs/exclude-directory-file-from-scan/

Bank, J. & Cole, B. (2008a). Calculating the jaccard similarity coefficient with map reduce for entity pairs in wikipedia. Retrieved March 9, 2018, from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.168.5695&rep=rep1&type=pdf

Bank, J. & Cole, B. (2008b). Calculating the jaccard similarity coefficient with map reduce for entity pairs in wikipedia. https://www.dfrws.org/sites/default/files/session-files/paper-identifying_almost_identical_files_using_context_triggered_piecewise_hashing.pdf.

Bloomsbury. (2015). *Basic dictionary pre-intermediate level 2nd edition*.

Blumofe, R., Joerg, C., Kuszmau, B., Leiserson, C., Randall, K., & Zhou, Y. (1995). Cilk: An efficient multithreaded runtime system.

Bradner, S. (1997). Key words for use in rfcs to indicate requirement levels. https://tools.ietf.org/html/rfc2119.

Bulkar, M. (2016). The difference between static and dynamic websites? Retrieved May 3, 2018, from https://www.linkedin.com/pulse/difference-between-static-dynamic-websites-mangesh-bulkar

Cambridge. (2018). Cambridge dictionary. Retrieved May 29, 2018, from https://dictionary.cambridge.org/dictionary/english/promising

Darmanin, G. (2017). Tips on reducing acunetix scan time. Retrieved March 8, 2018, from https://www.acunetix.com/blog/docs/tips-reducing-acunetix-scan-time/

Free Software Foundation, Inc. (2018). Categories of free and nonfree software. Retrieved April 25, 2018, from http://www.gnu.org/philosophy/categories.en.html

Garbee, J. (2018). Understanding resource timing. Retrieved March 9, 2018, from https://developers.google.com/web/tools/chrome-devtools/network-performance/understanding-resource-timing

Gilleland, M. (2006). Levenshtein distance, in three flavors. https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm.

Gommers, T. (2017a). Callbacks - not your average web crawler. Retrieved February 23, 2018, from https://tijme.github.io/not-your-average-web-crawler/latest/options_callbacks.html

Gommers, T. (2017b). Not your average web crawler. Retrieved February 23, 2018, from https://github.com/tijme/not-your-average-web-crawler

Gommers, T. (2017c). Not-your-average-web-crawler/crawler.py. Retrieved March 5, 2018, from https://github.com/tijme/not-your-average-web-crawler/blob/32191cc375f4ec9cde1c23cc388c5ff28f3d573b/nyawc/Crawler.py%5C#L213

Google LLC. (2018). Improve server response time. Retrieved March 9, 2018, from https://developers.google.com/speed/docs/insights/Server

Helix84. (2006). Http persistent connection.svg. Retrieved March 22, 2018, from https://commons.wikimedia.org/wiki/File:HTTP_persistent_connection.svg

Intel Corporation. (2003). *Intel hyper-threading technology, technical user's guide*. https://web.archive.org/web/20100821074918/http://cache-www.intel.com/cd/00/00/01/77/17705_htt_user_guide.pdf.

Kobayashi, M. & Takeda, K. (2000). Information retrieval on the web.

Kornblum, J. (2006). Identifying almost identical files using context triggered piecewise hashing. https://www.cs.ccu.edu.tw/~naiwei/cs5812/st4.pdf.

Kulla, S. (2015). Intersection of sets a and b. Retrieved May 4, 2018, from https://commons.wikimedia.org/wiki/File:Intersection_of_sets_A_and_B.svg

Naudi, T. (2016). Acunetix wins "best of 2016" award with acunetix online vulnerability scanner. Retrieved April 6, 2018, from https://www.acunetix.com/blog/news/acunetix-wins-best-2016-award-acunetix-online-vulnerability-scanner/

Niglas, K. (2004). The combined use of qualitative and quantitative methods in educational research. http://www.tlulib.ee/files/arts/95/nigl9.pdf.

Northwave. (2018). Driven security experts. Retrieved April 11, 2018, from https://www.northwave.nl/over/

Northwave BV. (2018). Samenhang: De meerwaarde van onze services. Retrieved April 2, 2018, from https://www.northwave.nl/diensten/

Ogri, E. (2017). Arachni github issue 843. Retrieved February 19, 2018, from https://github.com/Arachni/arachni/issues/843

Oracle. (2015). Fxmlloader (javafx 8). Retrieved April 19, 2018, from https://docs.oracle.com/javase/8/javafx/api/javafx/fxml/FXMLLoader.html

OWASP. (2016). Category:vulnerability. Retrieved April 16, 2018, from https://www.owasp.org/index.php/Category:Vulnerability

OWASP. (2018). Category:vulnerability scanning tools. Retrieved April 18, 2018, from https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools

PortSwigger. (2017a). Burp suite extensibility. Retrieved February 21, 2018, from https://portswigger.net/burp/extender

PortSwigger. (2017b). Burp suite scanner. Retrieved February 23, 2018, from https://portswigger.net/burp

PortSwigger. (2017c). Options: Http. Retrieved February 26, 2018, from https://portswigger.net/burp/help/options_http

PortSwigger. (2017d). Target scope. Retrieved February 27, 2018, from https://portswigger.net/burp/help/target_scope

PortSwigger. (2018). Getting started with burp suite. Retrieved February 21, 2018, from https://portswigger.net/burp/help/suite_gettingstarted

Rosen, K. H. (2012). *Discrete mathematics and its applications*. https://www.maa.org/press/maa-reviews/discrete-mathematics-and-its-applications.

Sarosys LLC. (2018). Free, simple, distributed, intelligent, powerful and friendly. Retrieved February 20, 2018, from http://www.arachni-scanner.com

Vurtis, T. (2016). Burp suite support center. Retrieved February 19, 2018, from https://support.portswigger.net/customer/portal/questions/16114895-scanner-is-very-slow-running

W3C. (1998). Cool uris don't change. Retrieved April 26, 2018, from https://www.w3.org/Provider/Style/URI.html

W3C. (1999). Hypertext transfer protocol – http/1.1. https://tools.ietf.org/html/rfc2616.

Wireshark Foundation. (2018). Wireshark - go deep. Retrieved May 1, 2018, from https://www.wireshark.org

List of Figures

Appendices

A Results 'a promising key concept of the scanner'

Sub-question one

The next page contains a PDF of results from sub-question one; 'Which key concept of the Automated Web Application Vulnerability Scanners is the most promising to improve efficiency?'. The vertical axis contains the three demo web applications and the type of scans performed on these web applications. The horizontal axis contains the two scanners that can be scanned and their effectiveness and efficiency noted as percentage as well as decimal.

Some columns state N.A. This means that the relevant technology is not available or modifiable in the scanner.

	Acunetix								Burp Suite							
	Effectiveness				*Efficiency*				*Effectiveness*				*Efficiency*			
	Vulns found (decimal & percentage)				*Seconds elapsed (decimal & percentage)*				*Vulns found (decimal & percentage)*				*Seconds elapsed (decimal & percentage)*			
	Low	*High*	*Low*	*High*	*Low*	*High*	*Low*	*High*	*Low*	*High*	*Low*	*High*	*Low*	*High*	*Low*	*High*
Damn Vulnerable Web Application http://www.dvwa.co.uk/																
Basis (without modification)	*27*	*27*	*100%*	*100%*	*702*	*702*	*100%*	*100%*	*22*	*22*	*100%*	*100%*	*1063*	*1063*	*100%*	*100%*
Target scope reduction	23	27	**85%**	**100%**	371	464	**53%**	**66%**	12	18	**55%**	**82%**	358	826	**34%**	**78%**
Multi-threading	27	27	**100%**	**100%**	1191	2638	**170%**	**376%**	22	22	**100%**	**100%**	1053	1418	**99%**	**133%**
Persistent HTTP connections	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
Vulnweb Forum http://testasp.vulnweb.com/																
Basis (without modification)	*16*	*16*	*100%*	*100%*	*302*	*302*	*100%*	*100%*	*15*	*15*	*100%*	*100%*	*4867*	*4867*	*100%*	*100%*
Target scope reduction	12	14	**75%**	**88%**	210	297	**70%**	**98%**	13	15	**87%**	**100%**	2897	4765	**60%**	**98%**
Multi-threading	16	16	**100%**	**100%**	477	1136	**158%**	**376%**	15	15	**100%**	**100%**	3756	4616	**77%**	**95%**
Persistent HTTP connections	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
AltoroMutual http://demo.testfire.net/																
Basis (without modification)	*24*	*24*	*100%*	*100%*	*906*	*906*	*100%*	*100%*	*19*	*19*	*100%*	*100%*	*1571*	*1571*	*100%*	*100%*
Target scope reduction	13	22	**54%**	**92%**	214	536	**24%**	**59%**	17	17	**89%**	**89%**	700	864	**45%**	**55%**
Multi-threading	24	24	**100%**	**100%**	2001	3015	**221%**	**333%**	19	19	**100%**	***100%***	900	1706	**57%**	**109%**
Persistent HTTP connections	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.

B Results 'a technology to improve the efficiency'

Sub-question two

The next page contains a PDF of results from sub-question two; 'Which technology can be used to improve the most promising key concept in an automated way?' The vertical axis contains the three demo web applications and the technologies used to perform the scans on the web applications. The horizontal axis contains the effectiveness and efficiency noted as percentage as well as decimal.

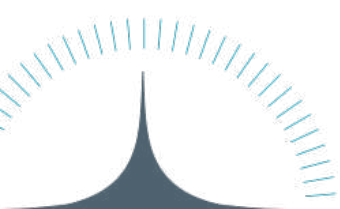
	Effectiveness *Vulnerabilities found*		*Efficiency* *Seconds elapsed*	
	Decimal	*Percentage*	*Decimal*	*Percentage*
Damn Vulnerable Web Application http://www.dvwa.co.uk/				
Basis (without modification)	*22*	*100%*	*1063*	*100%*
HTML tree similarity measure	6	**27%**	438	**41%**
Piecewise response hashing	14	**64%**	412	**39%**
A self-developed undirected graph	9	**41%**	310	**29%**
Vulnweb Forum http://testasp.vulnweb.com/				
Basis (without modification)	*15*	*100%*	*4867*	*100%*
HTML tree similarity measure	12	**80%**	1409	**29%**
Piecewise response hashing	13	**87%**	1339	**28%**
A self-developed undirected graph	13	**87%**	2404	**49%**
AltoroMutual http://demo.testfire.net/				
Basis (without modification)	*19*	*100%*	*1571*	*100%*
HTML tree similarity measure	17	**89%**	1020	**65%**
Piecewise response hashing	14	**74%**	1373	**87%**
A self-developed undirected graph	17	**89%**	847	**54%**

C Results 'a user friendly product'

Sub-question three

The next page contains a PDF of results from sub-question three; 'In which user friendly way can the most efficient and effective technology be integrated with Automated Web Application Vulnerability Scanners?' The vertical axis contains the possible solutions on how to integrate the technology. The horizontal axis contains the three scanners that the solution was tested on.

Some columns state N.A. This means that the relevant solution is not available or capable as integration in the scanner.



	Acunetix		*Burp Suite*		*NYAWC*	
	Interactions	*Seconds*	*Interactions*	*Seconds*	*Interactions*	*Seconds*
An extension for every scanner	N.A.	N.A.	2	11	0	0
Crawler that outputs URLs to a file	6	57	6	42	3	39
Available as API	6	57	2	11	0	0
A scanner extension that outputs URLs to a file	6	68	2	11	5	50

D The settings in GraphWave

- **Status**: enables or disables the extension.
- **Minimum edge threshold**: the minimum amount of similar responses required before ignoring further responses.
- **Minimum points threshold**: the minimum amount of points a response needs to be marked as similar.
- **Matching stylometry threshold**: the minimum amount of similarity between HTTP responses to be marked as similar.
- **URL path - exact match**: the amount of points a response gets when there is an exact url path match.
- **URL path - slug match**: the amount of points a response gets when there is a slug match in the url path.
- **URL path - number match**: the amount of points a response gets when there is a number match in the url path.
- **URL path - word match**: the amount of points a response gets when there is a word match in the url path.
- **URL path - slug match**: the amount of points a response gets when there is a slug match in the url path.
- **URL query - exact match**: the amount of points a response gets when there is an exact url query match.
- **URL query - slug match**: the amount of points a response gets when there is a slug match in the url query.
- **URL query - number match**: the amount of points a response gets when there is a number match in the url query.
- **URL query - word match**: the amount of points a response gets when there is a word match in the url query.
- **URL query - slug match**: the amount of points a response gets when there is a slug match in the url query.

Abstract

In a world where desktop applications are shifting more and more towards web applications, web application security is a rising concern. Manually testing these applications for security vulnerabilities is infeasible; as security specialists are scarce.

Automated Web Application Vulnerability Scanners are tools that scan web applications to look for security vulnerabilities in an automated way. The advantage of these scanners is that they can replace parts of the manual security vulnerability scanning. However, a major disadvantage is that they tend to be inefficient depending on the type of web application that is being scanned.

Using qualitative and quantitative research methodologies, this research shows why the scanners are occasionally inefficient and how that inefficiency can be resolved. The proposed solution (constructed from the research results) can be applied on a broad range of scanners. It is based on one of the key concepts of scanners: reducing the scope of a scan. Three techniques were introduced that could reduce the scope in an automated way.

The best technique, based on graph theory, was used to develop a product that helps scanners to automatically reduce the scope of a scan while maintaining effectiveness. It is integrated in a user friendly way as a scanner specific extension, but it has functionalities that make it compatible with many others.