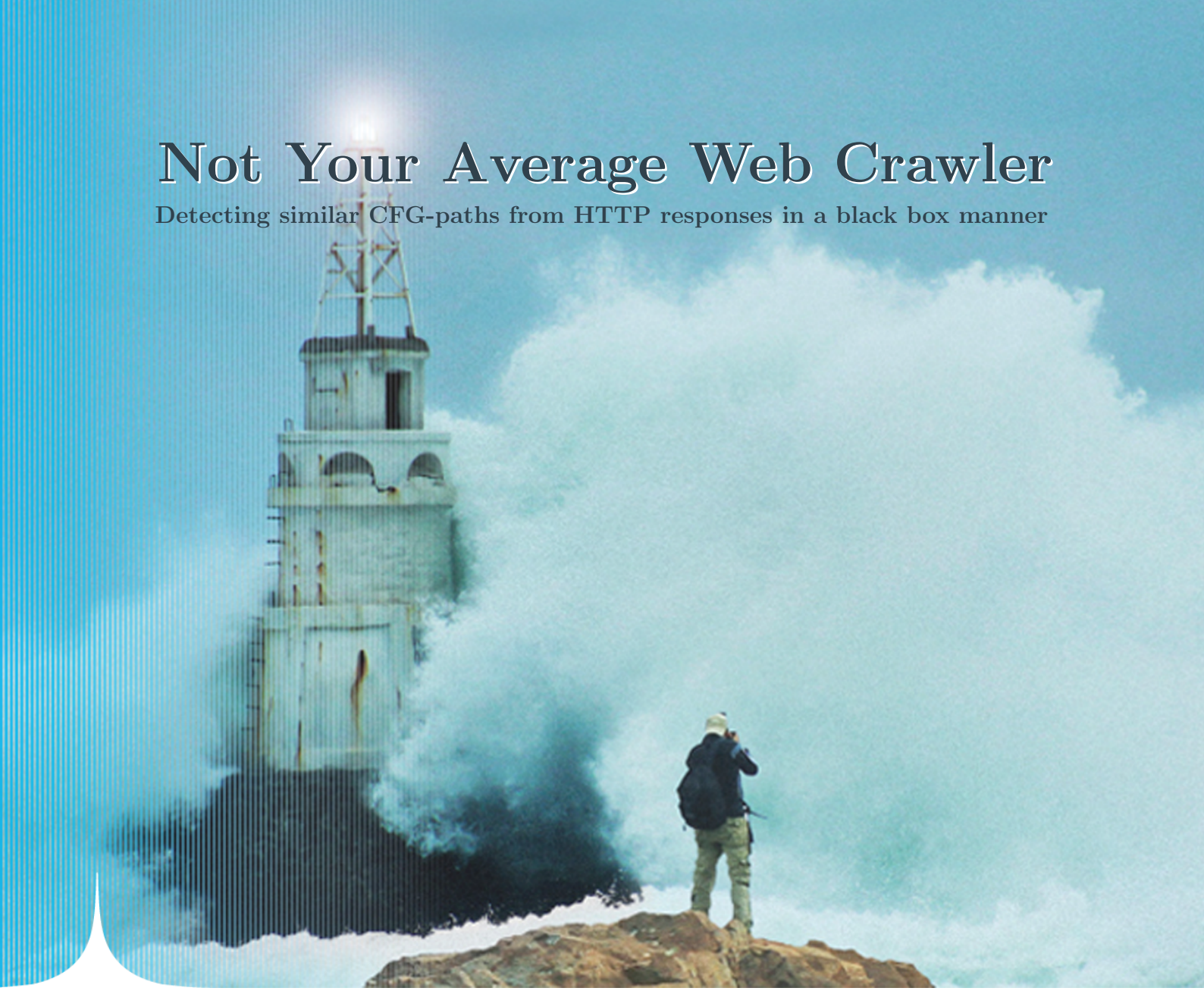
Not Your Average Web Crawler

Detecting similar CFG-paths from HTTP responses in a black box manner



Tijme Gommers
Author/Graduate

NORTHWAVE
Intelligent Security Operations

Marco Marcellis
University Supervisor

Amsterdam University of Applied Sciences

Prof. Eric Filiol
University Supervisor

esiea

Martijn Hoogesteger
Company Supervisor

NORTHWAVE
Intelligent Security Operations

Not Your Average Web Crawler

Detecting similar CFG-paths from HTTP responses in a black box manner

Amsterdam University of Applied Sciences
HBO-ICT Software Engineering

École D'ingénieurs du Monde Numérique
MINOR Information Security

Northwave BV
Red Team Department
Marconibaan 49
3439 MR Nieuwegein

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

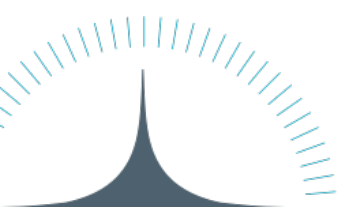
Tijme Gommers
Author/Graduate (AUAS/ESIEA)
Student^{nr} 500708891
tijme.gommers@hva.nl

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

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

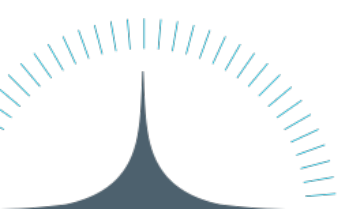
Prof. Eric Filiol
Supervisor (ESIEA)
Head of ESIEA CVO Lab
eric.filiol@esiea.fr

Nieuwegein, the Netherlands - April 13, 2018
Version 0.66.1-ba6ba2e2



Abstract

work in progress



Contents

List of Figures

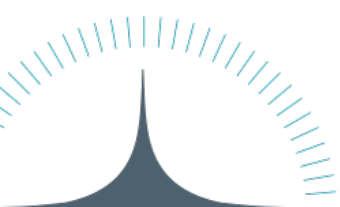
Acronyms

HTTP Hypertext Transfer Protocol. 1, 4, 12, 16, 23, 24, 27, 28, 30, 34

NYAWC Not Your Average Web Crawler. 1, 11–15, 21, 29

scanner Automated Web Application Vulnerability Scanner. 1, 2, 11–30, 34, 35, 40, 44

TTFB Time To First Byte. 1, 16



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, 4, 14, 20, 23, 24, 29

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). 11

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, 2, 4, 11, 14, 15, 23, 24, 29–31, 33, 34

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 path from the control flow graph. 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). 14, 20

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). 27

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 source code must be available (Free Software Foundation, Inc, 2018). 10–12, 14

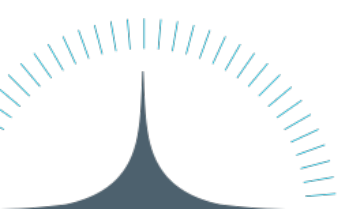
payload A payload is a text snippet that contains malware such as worms or viruses which performs a malicious action. 14, 15

red-team The Northwave red-team tests security to the full extent to see if they are able to get confidential data. They do this both digitally and through personal approaches (social engineering) (Northwave BV, 2018). 11, 13, 14, 25, 34

RESTful URL RESTful URLs or clean URLs are links to web applications that are meaningful and do not change over time (W3C, 1998). 35

run-time The time during which a computer program is executed (Blumofe et al., 1995). 11, 15, 17, 19

vulnerability A vulnerability is a hole or a weakness in the application, which can be a design flaw or an implementation bug, that allows an attacker to cause harm to the stakeholders of an application (OWASP, 2016). 11, 12, 15, 18–20, 27



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.)

Context

The author

My name is Tijme Gommers. I am a software engineer, security researcher, student, and the author of this thesis. I wrote this thesis to graduate for my study Software Engineering at the AUAS. During my study I did an international minor in Information Security at ESIEA (located in France) because I love doing security related research and programming.

Both my study at AUAS and my minor at ESIEA required me to do a graduation internship and write a thesis to finalise my studies. This is the main reason why I wrote my thesis in English instead of Dutch.

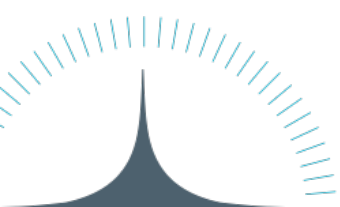
About half a year before my graduation internship I was recruited by Northwave to do my graduation internship at the Northwave Cyber Security department. I joined them for one day to see what life at Northwave is like. I enjoyed that day a lot and decided to take the offer.

Northwave

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 and intelligent 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 money and knowledge 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 units that both have their own tasks, Northwave Business Security (NBS) and Northwave Cyber Security (NCS).

Martijn Hoogesteger is the team leader of the Northwave computer emergency response team and will be my company supervisor during the internship.



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.

Northwave Cyber Security

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

AUAS

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.

ESIEA

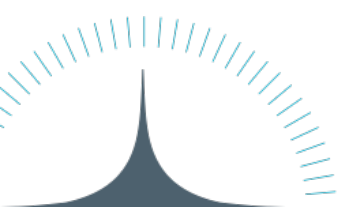
École Supérieure D'informatique, Électronique, Automatique is a French university for engineers based in France, in the city of Laval.

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

The security community

Since all the stakeholders in this thesis use open-source security techniques I have decided to contribute the results of this study back to the security community so that everyone can enjoy and use it.

This will not be discussed in depth in this thesis; however, I want to mention that this research will be open-source. Where necessary, contributing to the the security community is taken into account, but the primary goal is to get a solution for Northwave.



1. Introduction

Automated Web Application Vulnerability Scanners, hereinafter referred to as scanners, are tools that scan web applications, normally from the outside, to look for security vulnerabilities such as cross-site scripting, SQL injection, command injection, path traversal and insecure server configuration (OWASP, 2018).

Not Your Average Web Crawler (NYAWC) is such a scanner. It is an open-source variant of two scanners currently used by Northwave. Northwave is planning to use NYAWC too; however, their first priority is optimising the efficiency of the scanners (while maintaining effectiveness), since they are not satisfied with the efficiency at the moment.

It is important to know that efficiency is essentially just the speed or run-time of the scanners and effectiveness is how many vulnerabilities the scanners find. Both of these terms are covered extensively in the theoretical framework (section 2.3).

Reviews about scanners by Northwave and the online security community show that not all scanners are efficient. However, they don't state why they are efficient or inefficient, or how the efficiency can be improved.

'Some of our scanners may take up to four hours to complete', says Martijn Hoogesteger, Security Specialist at Northwave.

But this is not only a complaint from the Northwave red-team, there are enough support articles available online that show security researchers complaining about the time that it takes to successfully finish a scanner 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 cause annoyance and delays at the Northwave red-team and the security community.

1.1 Motivation

For Northwave

The red-team security audit is a small part of all the services that Northwave provides. During the audit, automated and manual security tests are per-

formed on web applications of the client. Scanners are more extensive than manual tests since they are programmed to be exhaustive.

At this moment, February 2018, Northwave does not know how to improve the efficiency of the scanners they use.

For NYAWC

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

For AUAS and ESIEA

To graduate for the Bachelor in Software Engineering from the Amsterdam University of Applied Sciences and for the Minor in Information Security from ESIEA University, thirty ECTS points are required that can be gained by successfully finishing the graduation internship (which includes writing this thesis).

1.2 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.

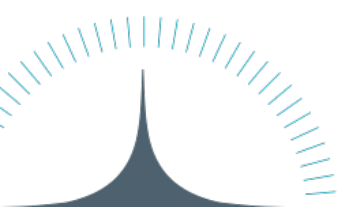
1.3 Scope

It is beyond the scope of this study to examine how to improve specific scanners that require improvements that cannot be applied broadly.

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 already high enough.

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 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. The three scanners that will be used are defined in chapter 2.1.



1.4 Relevance

The practical usefulness of a solution for Northwave is:

1. that it decreases the time they spend on monitoring scanners,
2. that the red-team can therefore spend more time on manual testing &
3. that it decreases the annoyance and delays in the red-team.

The practical usefulness of a solution for NYAWC is:

1. that it improves the efficiency of the scanner &
2. that it, therefore, increases consumer satisfaction.

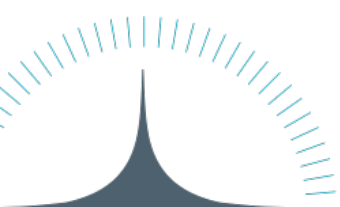
1.5 Questions

1.5.1 Main question

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

1.5.2 Sub-questions

1. How can the efficiency and effectiveness of Automated Web Application Vulnerability Scanners be measured?
2. Which key concept of the Automated Web Application Vulnerability Scanners can improve efficiency the most while maintaining effectiveness?
3. Which technologies can be used to improve the most in efficiency varying key concept in an automated way?
4. In which, automated, user friendly way can the most efficient and effective technology be integrated with Automated Web Application Vulnerability Scanners?



2. Theoretical framework

In this theoretical framework the scanners, key concepts and measurements of this study are explained by 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 or solution since they all provide functionalities to modify their own behaviour.

"Burp Suite (2.1.1) and Acunetix (2.1.2) are the two most used scanners at Northwave", says Martijn Hoogesteger, Security Specialist at Northwave.

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).

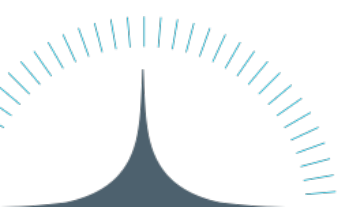
2.1.2 Acunetix

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 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 concepts that are used by, and determine the efficiency and/or run-time of the scanners.

2.2.1 Target scope (reduction)

Before starting a scanner the scanner, e.g. Burp Suite, 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.

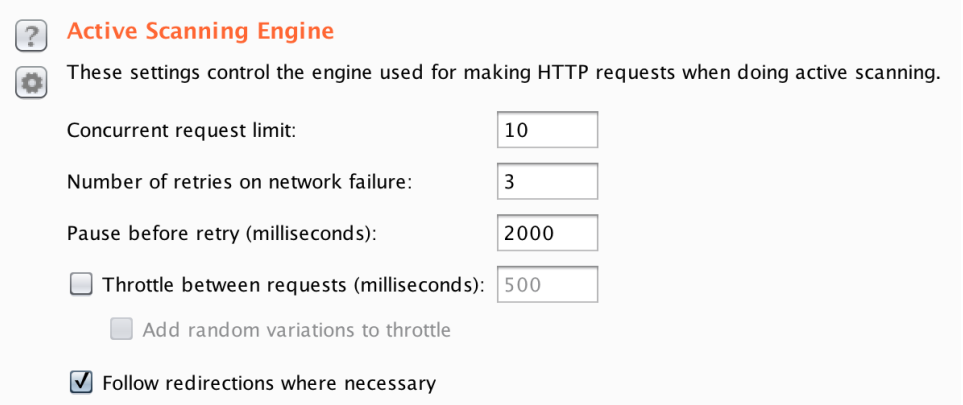
All the scanners (defined in chapter 2.1) have options to reduce the scope of the scanning run-time. Reducing the scope means less requests will be scanned (which reduces the run-time).

2.2.2 Multi-threading

The three scanners that will be tested all have the functionality to tweak the threading behaviour.

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). Figure 2.1 shows the Burp Suite options that enable multi-threading.

Figure 2.1: Burp Suite multi-threading options



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 counts for remote servers though, since locally hosted web applications do not have a round trip to a remote server.

The lead time of getting a customer's web application to work locally is ten hours on average", says Martijn Hoogesteger, Security Specialist at Northwave.

For Northwave, the amount of requests R that scanners have to scan locally before gaining a time benefit compared to scanning remotely is $R(s,t) = (\bar{s} \cdot 60 \cdot 60 \cdot 1000) \ / \ \bar{t}$, where s is the set of hours it takes Northwave to setup the customer's web applications locally and t is the set of TTFBs when scanning on remote servers.

This shows us that any performance improvement through TTFB is negligible. Using the formula it can be proven that the average amount of requests that scanners need to scan locally before gaining a time benefit compared to scanning remotely is $R(10, 200) = 180.000$, which is way more than the amount of requests web applications from Northwave customers have.

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 2.2, 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.

2.3 Measurements

The terms 'efficiency' and 'effectiveness' are important terms in the research. The goals is to improve the efficiency while maintaining the effectiveness. Therefore these terms need to be defined and measured.

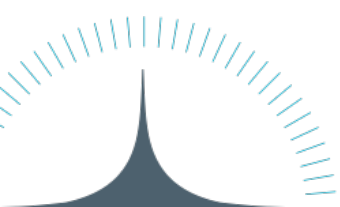
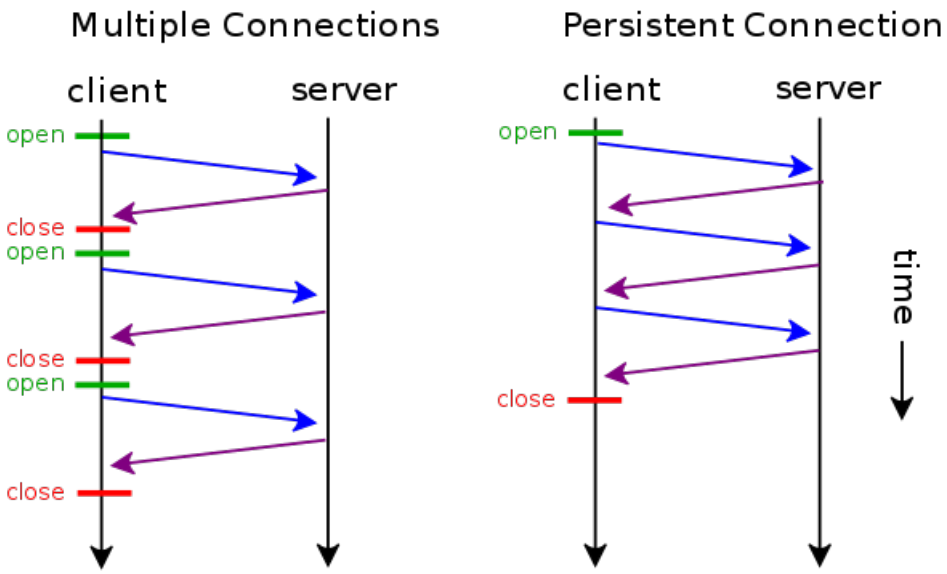


Figure 2.2: The difference between multiple connections and a persistent connection (Helix84, 2006).



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 G of a scanner is the improved runtime compared to normal runtime, measured in percentage. It can be formulated using discrete mathematics as follows (Rosen, 2012, p. 94).*

$$G(T_o, T_n) = \frac{100}{T_n} \cdot T_o$$

In this formula T is the run-time duration and the corresponding T_o is the old technology (normal) and T_n is the new technology (improved). The efficiency is 100% when the new technique has the same run-time duration as the old technique.

2.3.2 Effectiveness

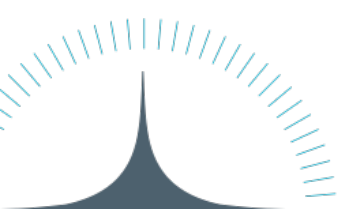
Effectiveness is being able to produce the required result (Bloomsbury, 2015, p. 107). Please note that there is a major difference between efficiency and effectiveness. One can be efficient without being effective, and vice versa.

Theorem 2.3.2.1 (Effectiveness measurement) *The effectiveness H 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).*

$$H(V_o, V_n) = \frac{100}{|V_o|} \cdot |V_n|$$

In this formula V is the set of vulnerabilities scanned and the corresponding V_o is the set of vulnerabilities from the old technology and V_n is the set of vulnerabilities from the new technology.

This means when $H(V_o, V_n) = 100$, the same amount of vulnerabilities are found. Since the tested scanners are exhaustive, H can never be higher than 100 compared to the basis.



3. Research design

This research design describes how the research can be set up, replicated and validated.

3.1 How can the efficiency and effectiveness of Automated Web Application Vulnerability Scanners be measured?

This question can be answered using the theory from chapter 2.3. Using the discrete mathematics, a percentage of improvement of the efficiency or effectiveness can be calculated based on two sets of results, e.g. before a behavioural change and after a behavioural change. Maintaining the level of effectiveness while improving the efficiency can therefore be proven.

The results that these calculations are based on can be gathered by running a scanner and measuring the run-time and amount of unique vulnerabilities found. These are the two input variables required for calculating the efficiency and effectiveness.

The other questions should be answered using qualitative and quantitative research, as defined in the following sections.

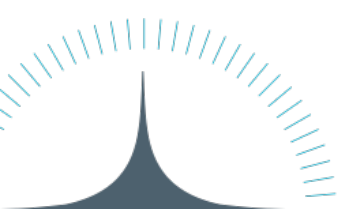
3.2 Which key concept of the Automated Web Application Vulnerability Scanners can improve efficiency the most while maintaining effectiveness?

3.2.1 Type of research

To answer this question quantitative research must be conducted.

3.2.2 Data gathering

Lab experiments can be used for data gathering, to measure which key concepts are inefficient.

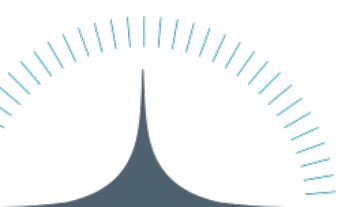


The lab experiments can be done by using the measurement formulas defined in chapter 2.3 while changing the behaviour of the key concepts (by e.g. dividing the scope in two or increasing the amount of threads to use).

In this research the three different (common) versatile types of web applications, listed below, will be tested.

1. **www.dvwa.co.uk**: a content web application.
    - A very vulnerable web application which main goals are to be an aid for security professionals to test their skills and tools in a legal environment. It was developed by Ryan Dewhurst. The web application consists of forty different kind of pages with, in total, 35 vulnerabilities. This means almost every page contains a vulnerability.
    - 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.
    - A vulnerable web application which is designed by Acunetix to test their scanner. The web application consists of seventeen different kind of pages with an unknown amount of vulnerabilities (since it's closed-source). The web application is meant to act and look as real as possible.
    - This web application contains many similar kind of pages that are all scanned by the scanners. Therefore scanners are very inefficient when scanning this web application, which is why this web application will be used for testing purposes.
3. **demo.testfire.net**: a banking web application.
    - A vulnerable web application published by Watchfire, Inc. for the sole purpose of demonstrating the effectiveness of Watchfire products. The web application consists of 51 different kind of pages with an unknown amount of vulnerabilities (since it's 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 very versatile and common type of web application on the internet and therefore it's good practise to use it for testing purposes.

The setup for testing these types of web applications is an identical virtual environment (e.g. VirtualBox). It does not matter what kind of environment (as long as it is identical) since the results are measured in percentages.



3.2.3 Data description

The data that should be analysed are the effectiveness and efficiency before and after behavioural changes.

3.2.4 Analysis method

The data gathered can be analysed empirically (by comparisons).

To see which key concept can improve the efficiency the most while maintaining effectiveness the formula $C(S,Y) = \frac{200-(S_l+S_h)}{2} + \frac{Y_l+Y_h}{2}$ can be used, where S is the effectiveness, Y is the efficiency and the corresponding $_l$ and $_h$ are the low and high values. The lowest result is the best.

This formula calculates the sum of the negative average percentage of effectiveness (negative because higher is better) and the average percentage of efficiency.

3.3 Which technologies can be used to improve the most in efficiency varying key concept in an automated way?

3.3.1 Type of research

To answer this question qualitative and quantitative research must be conducted.

3.3.2 Data gathering

Field research can be used for data gathering.

To find out which technologies can be developed and/or applied to improve the most in efficiency varying 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 may come from own experience or the experience of other security or software engineers.

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

3.3.3 Data description

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.

3.3.4 Analysis method

The data gathered 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 concerned scanner. To see which prototype is the most effective the effectiveness formula in chapter 2.3 can be used.

3.4 In which, automated, user friendly way can the most efficient and effective technology be integrated with Automated Web Application Vulnerability Scanners?

3.4.1 Type of research

To answer this question quantitative research must be conducted.

3.4.2 Data gathering

Field research can be used for data gathering.

The data that should be gathered is how many additional user interactions and seconds it takes for the user to start a scanner using the new prototypes, in contrast to without using those prototypes. User interactions may include e.g. mouse clicks or CLI commands.

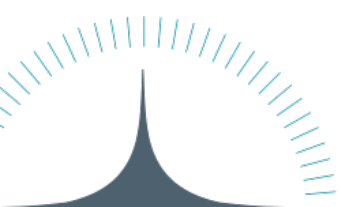
3.4.3 Data description

The data that should be analysed are the amount of additional user interactions and elapsed seconds while starting the scanner.

3.4.4 Analysis method

The data gathered can be analysed empirically (by comparisons).

To test in what user friendly way the technology from sub-question three can be used the solution with the least possible user interactions in combination with elapsed seconds should be chosen.



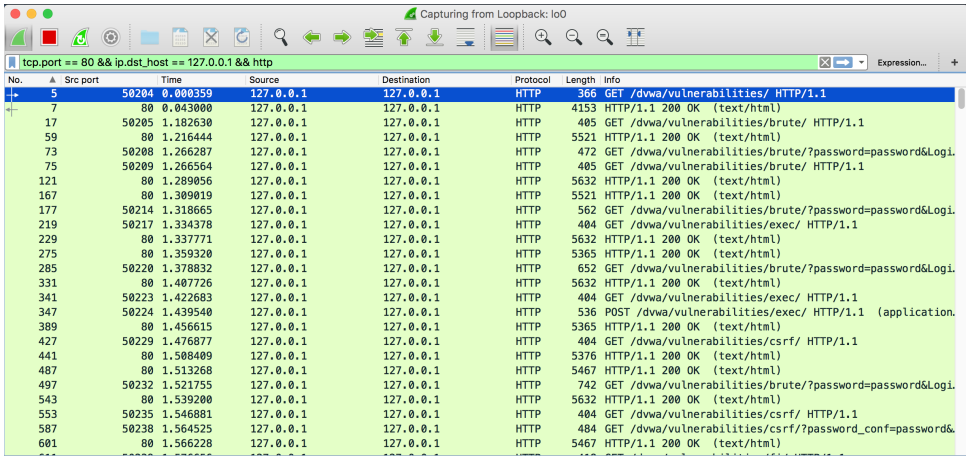
4. Research results

4.1 Which key concept of the Automated Web Application Vulnerability Scanners can improve efficiency the most while maintaining effectiveness?

The results of the measurements can be found in Appendix A. The Y axis contains the three demo web applications and the type of scans performed on these web applications. The X 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. For example, it turned out 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).

Figure 4.1: A list of requests and their corresponding responses (including the source ports which are notably different for every request).



Acunetix acts the other way around. The persistent HTTP connections cannot be disabled unless the server doesn't support it (as visible in Figure 4.3).

Differing results can be seen in the multi-threading rows. The reason for this is that a scan without modification already uses multi-threading in most

Figure 4.2: An HTTP stream showing that Burp Suite did use the correct connection header.

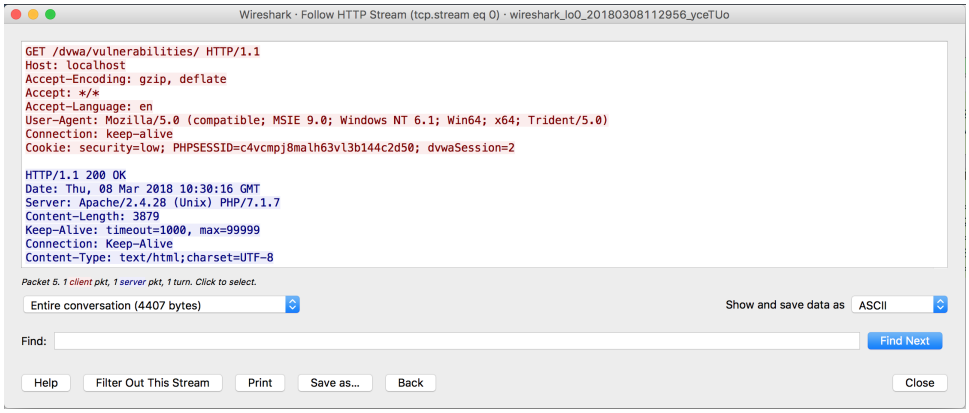
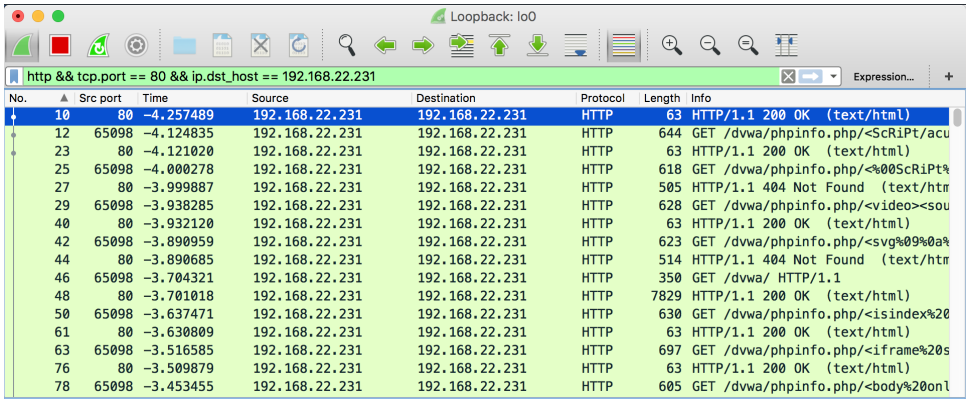


Figure 4.3: A list of requests and their corresponding responses that proof persistent HTTP connections in Acunetix.



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 maximum 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.

The results indicate that "Target scope reduction" can improve the efficiency the most while maintaining effectiveness when using the formula in chapter 3.2.4.

In addition, it can be seen that there is little to no relationship between efficiency and effectiveness. This confirms the theory from chapter 2.3.2.

4.2 Which technologies can be used to improve the most in efficiency varying key concept in an automated way?

The efficiency of scanners can be improved majorly by the use of target scope reduction. However, the results from sub-question two 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 two, one can see that the effectiveness is maintained when similar requests are ignored, but decreases when requests that are not similar are ignored. This similarity is based on the CFG-path from the requests.

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 a 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 maintain effectiveness. The technologies are also all compatible with the scanners, as defined in Chapter 3.3, (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).

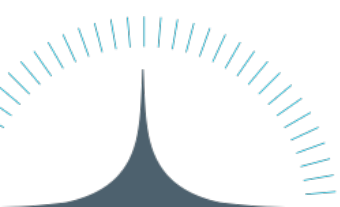
Similar CFG-paths output similar HTML, however, the content in the HTML (technically called node values) can be different (it can be different when e.g. reading a different row from a database). Therefore, to detect similar CFG-paths, one should only look at the HTML nodes and not at what text they contain (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 to generate 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 4.4: An example of similar signatures outputted by piecewise hashing.

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

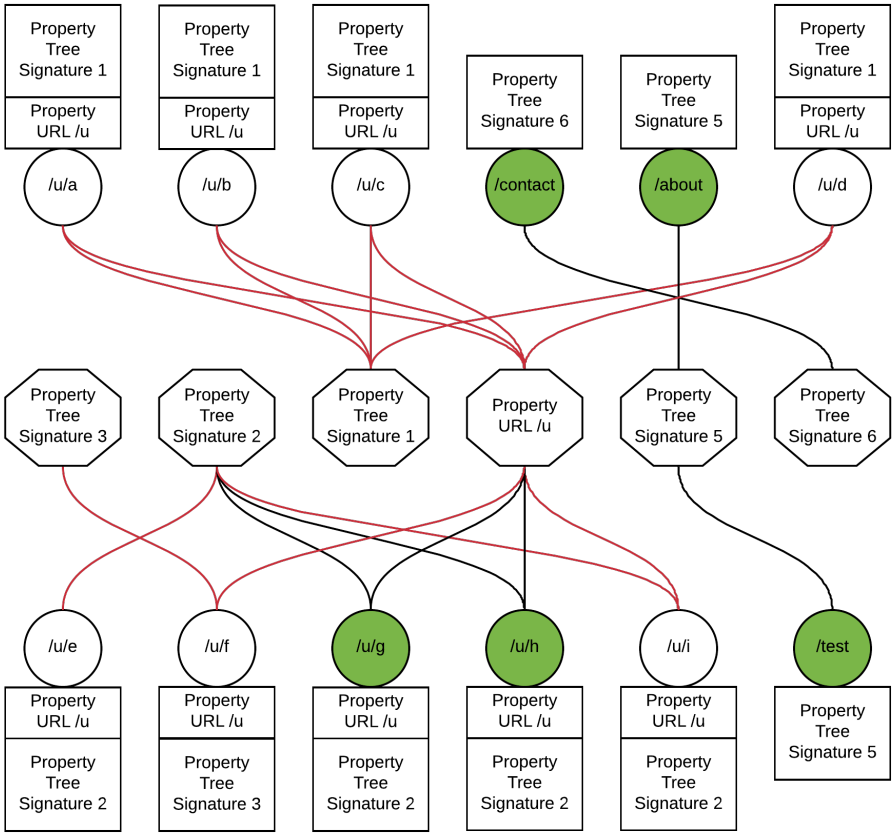
Piecewise hashing can be used to detect similarities in (HTML) trees and can therefore detect similar CFG-paths.

4.2.1.3 A custom graph cut based on response properties

The last technology is self-developed in the form of an undirected graph, as visualised in Figure 4.5.

- 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 end red lines are edges which represent which responses belong to which properties.

Figure 4.5: A custom graph cut visualization.



When a scanner crawls a web application the graph can be built in real-time 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 the weight of this property.
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 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 sum x of similar properties. 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 4.5 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 4.5, 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 every technology. The results of the effectiveness measurements of these prototypes can be found in Appendix B. The Y axis contains the three demo web applications and the technologies used to perform the scans on the web applications. The X axis contains the effectiveness and efficiency noted as percentage as well as decimal.

In contrast to the results from sub-question two, these results are not per scanner since the technology, scope reduction, is measured in percentage and this would result in identical results per scanner.

The results show divergent effectiveness percentages for the first web application (Damn Vulnerable Web Application). The reason for this is that this

web application is designed in such a way that it doesn't have a lot of similar CFG-paths.

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.

The custom 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.

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 custom 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
```

4.3 In which, automated, 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 Y axis contains the possible solutions on how to integrate the technology. The X 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 Integrated as a scanner extension

A solution is very 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. 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's 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 Separate crawler that outputs URLs to a text file

Every scanner needs a start request. This request is, 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 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.

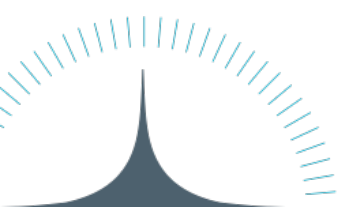
The results indicate 4,5 additional user interactions on average, which translates to 35 additional elapsed seconds.

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 consumers.

4.3.4 Burp Suite extension that outputs URLs to a text 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 test file and continue in a scanner of choice.



The vulnerability scanning process in Burp Suite is only available in de 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's compatible with all scanners, since it doesn't require any development and since it requires as less as possible 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

```
class BurpExtender(IBurpExtender, ITab):

    def registerExtenderCallbacks(self, callbacks):
        http_listener = NorthwaveHttpListener()
        http_listener._excludeFromScope = callbacks.excludeFromScope
        http_listener._saveConfigAsJson = callbacks.saveConfigAsJson
        http_listener._helpers = callbacks.getHelpers()
        http_listener._lock = Lock()

        callbacks.setExtensionName("Prototype")
        callbacks.registerHttpListener(http_listener)
```

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

```
class NorthwaveHttpListener(IHttpListener):

    def processHttpMessage(self, toolFlag, messageIsRequest, requestResponse):
        request = self._helpers.analyzeRequest(requestResponse)
        response = self._helpers.analyzeResponse(requestResponse.getResponse())
        html = self._helpers.bytesToString(requestResponse.getResponse())

        self._lock.acquire()

        added = graph.add_response(Response(request.getUrl().toString(), html))

        if added == False:
            config["target"]["scope"]["exclude"].append(request.getUrl().toString())

        self._lock.release()
```

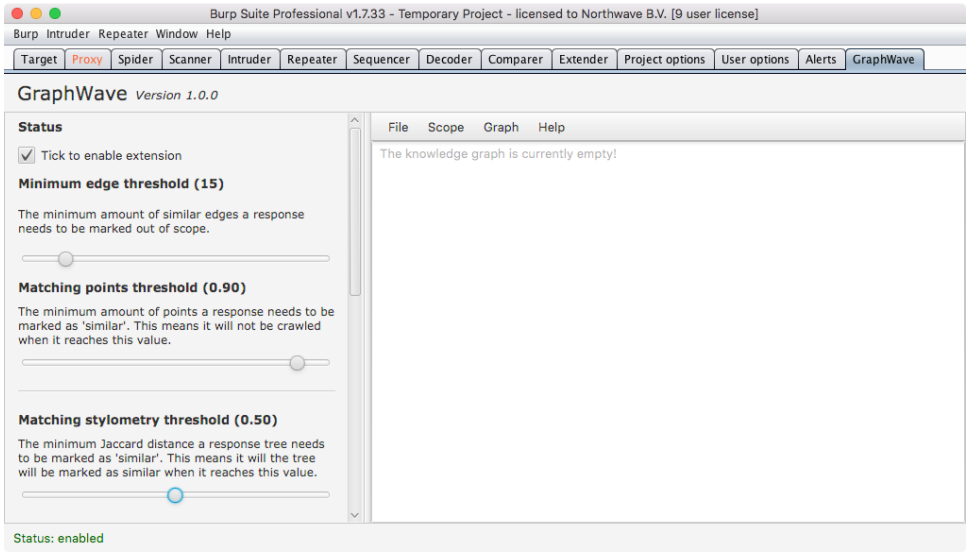
In Listing 4.3 a lock is used to prevent race conditions in the graph since the method `processHttpMessage` is asynchronous. Race conditions could cause similar request to be added to the graph while they should be ignored instead.

4.4 Main question

The main question 'Which user friendly automation can be developed to improve the efficiency of Automated Web Application Vulnerability Scanners while maintaining the effectiveness?' has been answered using the results from the sub-questions.

Figure 4.6 is a print screen of the "GraphWave" tab, which is the final product. It is a Burp Suite extension using the custom graph cut based on response properties to ignore similar URLs.

Figure 4.6: Product overview.



Burp Suite is written in Java and makes use of Jython to enable software engineers to develop extensions for Java in Python. Listing 4.4 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 4.4: Loading FXML files using JavaFX

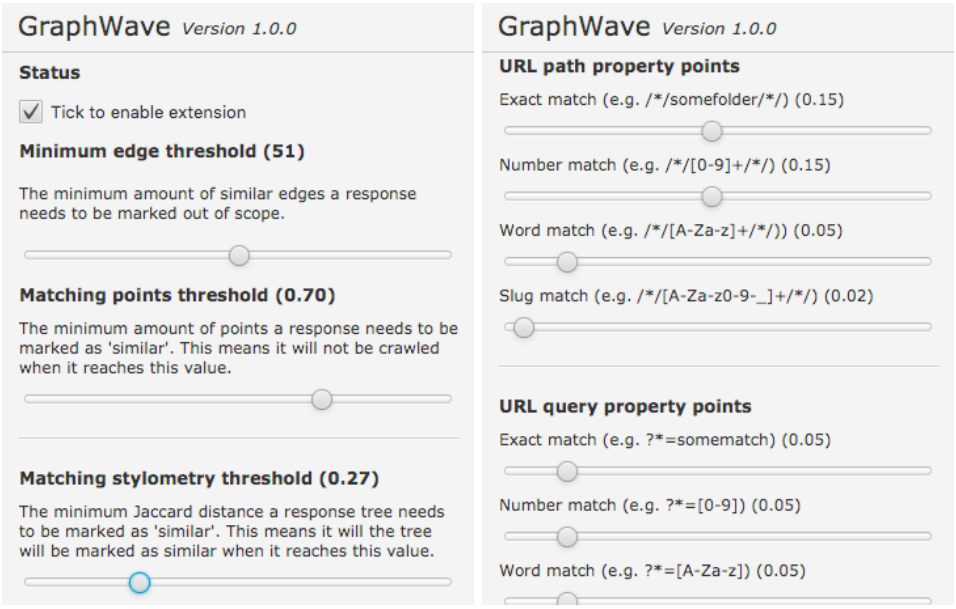
```
1  class BurpExtender(burp.ITab, burp.IBurpExtender):
2
3      def registerExtenderCallbacks(self, callbacks):
4          callbacks.setExtensionName(ExtensionDetails.TITLE)
5
6          self.layout = JFXPanel()
7
8          root = FXMLLoader.load(File(self.resourcePath + "extension_tab.fxml").toURL())
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)
```

The tab contains settings to configure the thresholds and the amount of points/weights of certain properties in the graph, as seen in Figure 4.7.

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 SEO URLs the URL path property points can be increased and the URL query properties can be

decreased.

Figure 4.7: Prodcut settings.



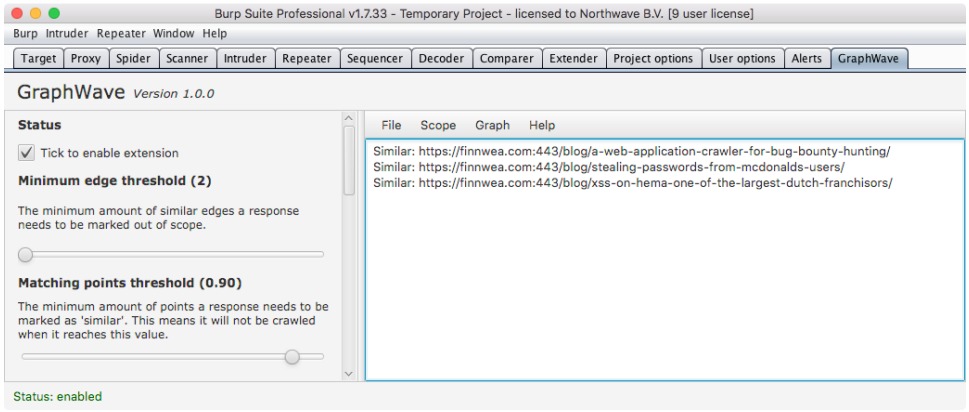
Listing 4.4 is a code snippet that demonstrates how certain properties in the graph are generated based on URLs. All properties are constructed using a weight that is retrieved from the default options or the options that the user inserted.

Code Listing 4.5: Properties being generated based on the URL path

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

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

        # Exact match (the deeper in the path, the more important it is)
        weight = ((index + 1) * options["upExactMatch"])
        weight = weight if weight < 0.3 else 0.3
        properties.append(GraphWaveProperty("url.path.exact[" + part + "][" + str(index) + "]", weight, part))

        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
```

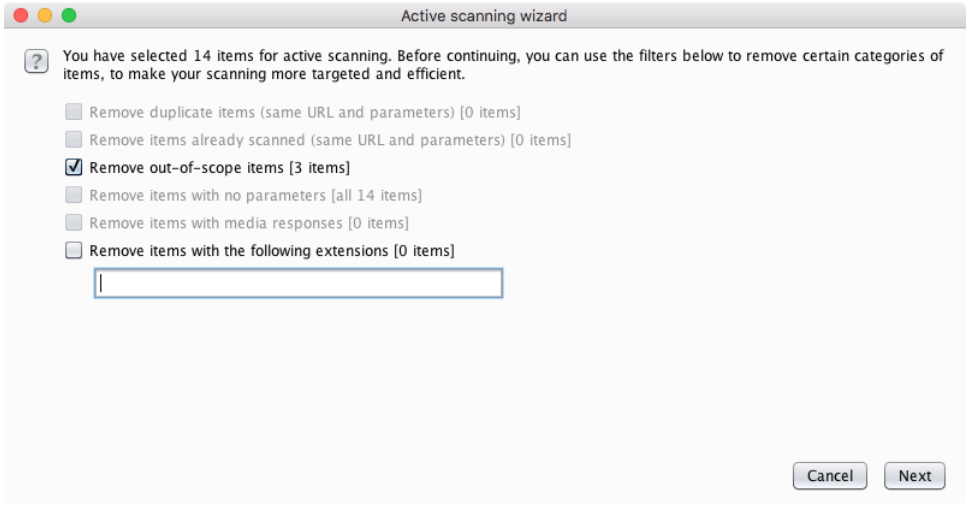
When an information gathering scan is being performed by Burp Suite the similar requests show up in the right panel of the GraphWave tab. When the scan is finished, a user can mark the similar request as "out of scope" by using the "Scope" menu.

Figure 4.8: Product overview with similar URLs.



Burp Suite has the option to ignore out of scope URLs when starting a fully automated vulnerability scan. As visible in Figure 4.9 the three similar URLs can be excluded.

Figure 4.9: Burp Suite can ignore out of scope URLs during a scan.



5. Conclusion

The goal of this research is to look for the answer to the question: 'Which user friendly automation can be developed to improve the efficiency of Automated Web Application Vulnerability Scanners while maintaining the 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.

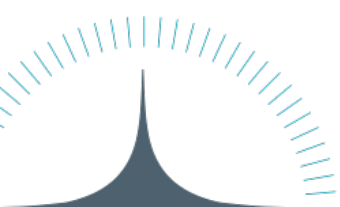
Efficiency and effectiveness are important terms since the complete research is based on them. They can be measured using mathematical formulas, which makes it easy to replicate and validate this research.

Three 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 distances between HTTP responses so that similar ones can be ignored. The 'Custom Graph Cut' prototype is the most in efficiency improving 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 we 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 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 87% 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.



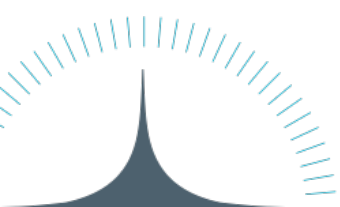
6. Discussion

For this research three different web applications have been used to test a new technology to improve efficiency of Automated Web Application Vulnerability Scanners. Therefore it can be stated that in case of a repeat of this research, the results would be the same and the results of this research would be valid.

The results of the tests showed that while some technologies improved the efficiency, the effectivity could not be fully maintained, as seen in Appendix A and B. 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 graph options. 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 B.

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.



Bibliography

Acunetix. (2017). Configuring acunetix to exclude scanning a portion of website. 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. 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. https://dl.acm.org/citation.cfm?id=209958.

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

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

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

Garbee, J. (2018). Understanding resource timing. 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. https://tijme.github.io/not-your-average-web-crawler/latest/options_callbacks.html.

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

Gommers, T. (2017c). Not-your-average-web-crawler/crawler.py. https://github.com/tijme/not-your-average-web-crawler/blob/8f3d573b/nyawc/Crawler.py#L213.

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

Helix84. (2006). Http persistent connection.svg. 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. https://dl.acm.org/citation.cfm?doid=358923.358934.

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

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

Northwave BV. (2018). Samenhang: De meerwaarde van onze services. https://www.northwave.nl/diensten/.

Ogri, E. (2017). Arachni github issue 843. https://github.com/Arachni/arachni/issues/843.

Oracle. (2015). Fxmlloader (javafx 8). https://docs.oracle.com/javase/8/javafx/api/javafx/fxml/FXMLLoader.html.

OWASP. (2016). Category:vulnerability. https://www.owasp.org/index.php/Category:Vulnerability.

OWASP. (2018). Category:vulnerability scanning tools. https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools.

PortSwigger. (2017a). Burp suite extensibility. https://portswigger.net/burp/extender.

PortSwigger. (2017b). Burp suite scanner. https://portswigger.net/burp.

PortSwigger. (2017c). Options: Http. https://portswigger.net/burp/help/options_http.

PortSwigger. (2017d). Target scope. https://portswigger.net/burp/help/target_scope.

PortSwigger. (2018). Getting started with burp suite. https://portswigger.net/burp/help/suite_gettingstarted.

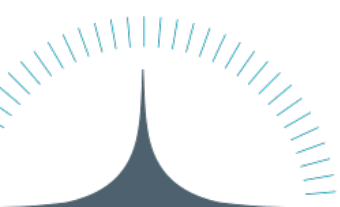
Rosen, K. H. (2012). *Discrete mathematics and its applications*. https://www.maa.org/press/maa-reviews/discrete-mathematics-and-its-applications.

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

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

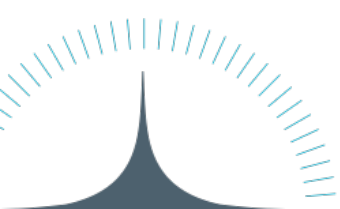
W3C. (1998). Cool uris don't change. https://www.w3.org/Provider/Style/URI.html.

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

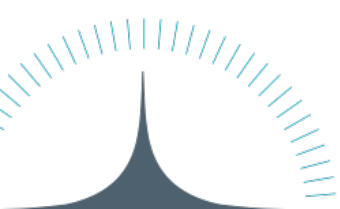


Epilogue

work in progress



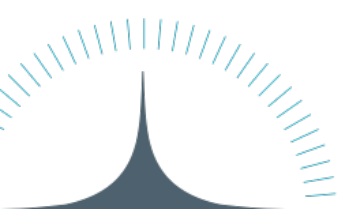
Appendices



A. Results sub-question 2

The next page contains a PDF of results from sub-question two. The Y axis contains the three demo web applications and the type of scans performed on these web applications. The X axis contains the two scanners that can be scanned and their effectiveness and efficiency noted as percentage as well as decimal.

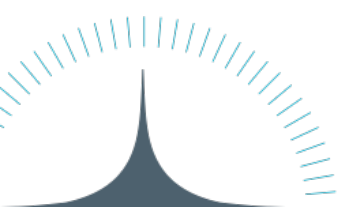
Some columns state N.A. This means that the relevant technology is not available or modifiable in the scanner.



	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 sub-question 3

The next page contains a PDF of results from sub-question four. The Y axis contains the three demo web applications and the technologies used to perform the scans on the web applications. The X axis contains the effectiveness and efficiency noted as percentage as well as decimal.

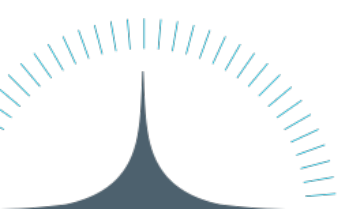


	Effectiveness *Vulnerabilities found*		*Efficiency* *Seconds elapsed*	
	Decimal	***Percentage***	***Decimal***	***Percentage***
Damn Vulnerable Web Application http://www.dvwa.co.uk/				
Basis (without modification)	*22*	*100%*	*1063*	*100%*
HTML tree similarity measure (4.2.1)	6	**27%**	438	**41%**
Piecewise response hashing (4.2.2)	14	**64%**	412	**39%**
Custom graph cut (4.2.3)	9	**41%**	310	**29%**
Vulnweb Forum http://testasp.vulnweb.com/				
Basis (without modification)	*15*	*100%*	*4867*	*100%*
HTML tree similarity measure (4.2.1)	12	**80%**	1409	**29%**
Piecewise response hashing (4.2.2)	13	**87%**	1339	**28%**
Custom graph cut (4.2.3)	13	**87%**	2404	**49%**
AltoroMutual http://demo.testfire.net/				
Basis (without modification)	*19*	*100%*	*1571*	*100%*
HTML tree similarity measure (4.2.1)	17	**89%**	1020	**65%**
Piecewise response hashing (4.2.2)	14	**74%**	1373	**87%**
Custom graph cut (4.2.3)	17	**89%**	847	**54%**

C. Results sub-question 4

The next page contains a PDF of results from sub-question five. The Y axis contains the possible solutions on how to integrate the technology. The X 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***
Solution 4.4.1	N.A.	N.A.	2	11	0	0
Solution 4.4.2	6	57	6	42	3	39
Solution 4.4.3	6	57	2	11	0	0
Solution 4.4.4	6	68	2	11	5	50