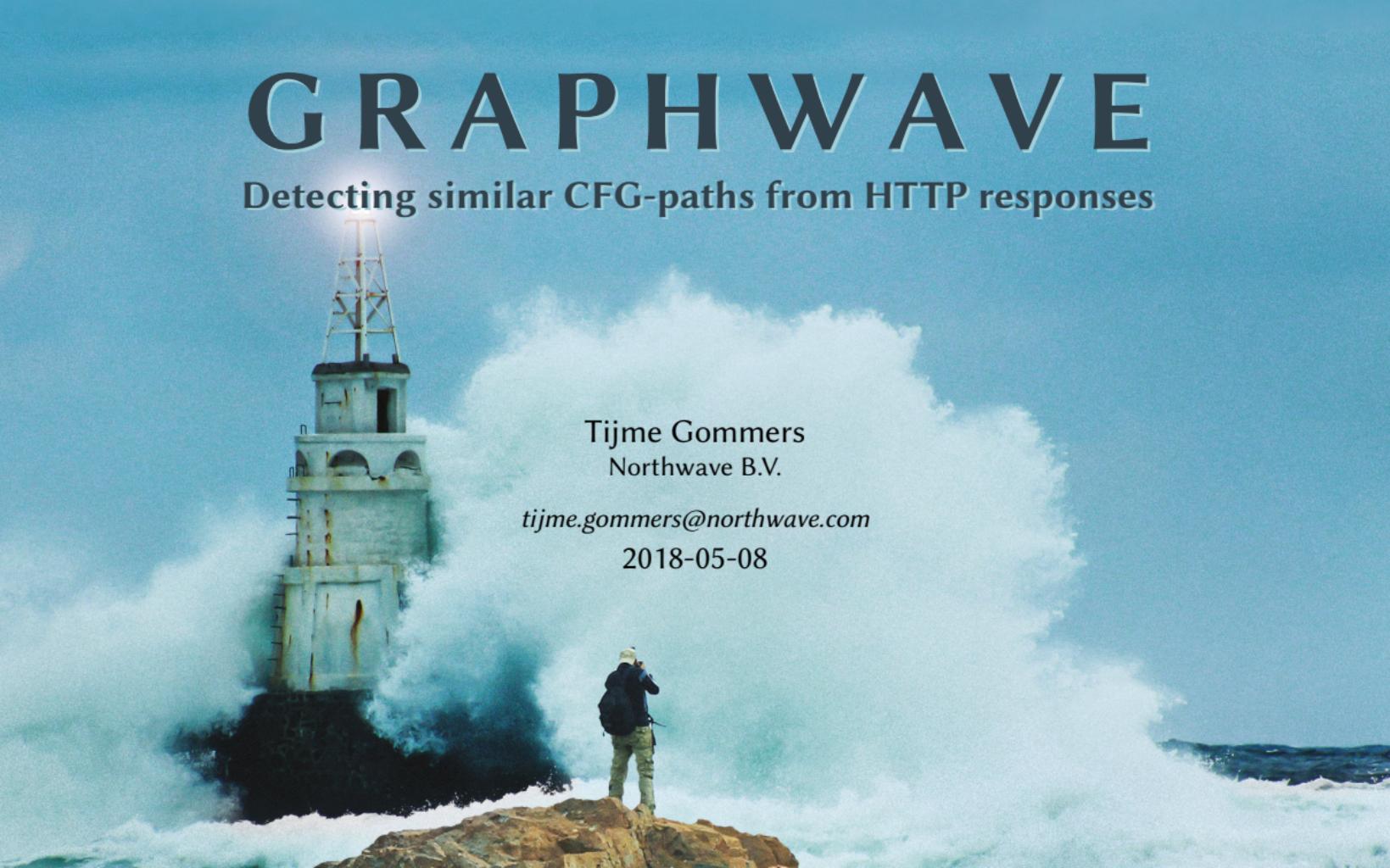


GRAPHWAVE

Detecting similar CFG-paths from HTTP responses



Tijme Gommers
Northwave B.V.

tijme.gommers@northwave.com

2018-05-08

Overview

1. Introduction

Context
Research
Questions

2. Theory

AWAVS
Key concepts
Measurements

3. Research

Methodology
Iterations

4. Results

The efficiency of scanner concepts
Technologies to improve the efficiency
A user friendly product

5. Conclusion

6. Discussion

Introduction Context

Northwave



Amsterdam University of Applied Sciences



École D'ingénieurs du Monde Numérique



Introduction Research

Automated Web Application Vulnerability Scanners

- Reconnaissance
- Crawling
- Vulnerability database
- Exploit database

Efficiency & Effectiveness

- Efficiency = Time to finish
- Effectiveness = Vulnerabilities found

Situation at Northwave

- Scans take up to **36** hours

Motivation

Goal

- Generic open-source solution
- Improve efficiency
- Maintain effectiveness

Scope

- Only generic solutions, **not** scanner specific
- Improve efficiency, **not** effectiveness
- Only web application scanners, not all scanners
- Only scanners that allow behaviour modifications

Introduction Questions

Main question

Which user friendly product can be developed to improve the efficiency of scanners while maintaining effectiveness?

Sub-questions 1

How can the efficiency and effectiveness of scanners be measured?

Which key concept of the scanners can improve efficiency the most while maintaining effectiveness?

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

In which user friendly way can the most efficient and effective technology be integrated with scanners?

Sub-questions 2

How can the efficiency and effectiveness of scanners be measured?

Which key concept of the scanners can improve efficiency the most while maintaining effectiveness?

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

In which user friendly way can the most efficient and effective technology be integrated with scanners?

Sub-questions 3

How can the efficiency and effectiveness of scanners be measured?

Which key concept of the scanners can improve efficiency the most while maintaining effectiveness?

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

In which user friendly way can the most efficient and effective technology be integrated with scanners?

Sub-questions 4

How can the efficiency and effectiveness of scanners be measured?

Which key concept of the scanners can improve efficiency the most while maintaining effectiveness?

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

In which user friendly way can the most efficient and effective technology be integrated with scanners?

Theory

Automated Web Application Vulnerability Scanners

Burp Suite

Burp Suite Professional v1.7.33 - Temporary Project - licensed to Northwave B.V. [9 user license]

Target **Proxy** **Spider** **Scanner** **Intruder** **Repeater** **Sequencer** **Decoder** **Comparer** **Extender** **Project options** **User options** **Alerts** **GraphWave**

Site map **Scope**

Logging of out-of-scope Proxy traffic is disabled **Re-enable**

Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

Contents

Host	Method	URL	Params	Status
https://finnwea.com	GET	/		200
https://finnwea.com	GET	/about/		200
https://finnwea.com	GET	/blog/a-web-applicatio...		200
https://finnwea.com	GET	/blog/adding-placehol...		200
https://finnwea.com	GET	/blog/i-decided-to-giv...		200
https://finnwea.com	GET	/blog/securing-your-ho...		200
https://finnwea.com	GET	/blog/stealing-passwor...		200
https://finnwea.com	GET	/blog/xss-on-hema-on...		200
https://finnwea.com	GET	/feeds/rss/		200
https://finnwea.com	GET	/hall-of-fame/		200
https://finnwea.com	GET	/responsible-disclosure/		200
https://finnwea.com	GET	/about		301

Issues

- Cacheable HTTPS response
- SSL certificate

Request Response

Raw Headers Hex

```
GET / HTTP/1.1
Host: finnwea.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: nl,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
If-Modified-Since: Mon, 30 Apr 2018 18:34:20 GMT
If-None-Match: "1a67-56b151aldb0c-gzip"
Cache-Control: max-age=0
```

Advisory Request Response

i Cacheable HTTPS response

Issue description

Issue: Cacheable HTTPS response
 Severity: Information
 Confidence: Certain
 Host: https://finnwea.com
 Path: /

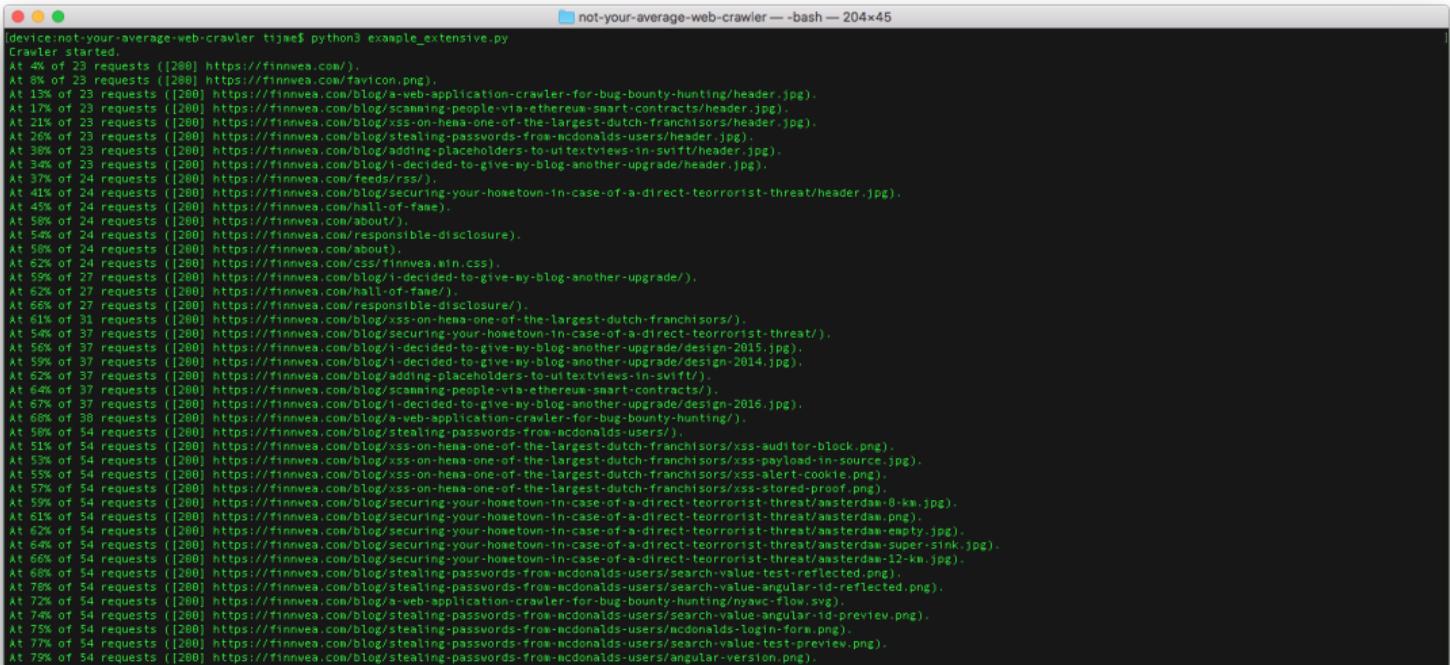
Acunetix

The screenshot shows the Acunetix Web Vulnerability Scanner interface. On the left, the 'Tools Explorer' sidebar lists various scanning tools and configuration options. The main pane displays a 'Scan Results' tree under 'Web Scanner'. A specific result for 'SQL injection (verified)' is highlighted. The details pane on the right provides a comprehensive report for this vulnerability, including:

- Vulnerability description:** This script is possibly vulnerable to SQL injection attacks.
- This is one of the most common application layer attacks currently being used on the Internet. Despite the fact that it is relatively easy to protect against, there is a large number of web applications vulnerable.**
- This vulnerability affects [http://products.php](#).**
- Discovered by Scripting (Sql_Injection script).**
- Attack details:** URL encoded GET input `product_id` was set to `(select 1 and rowid,1)>(select count(*),concat(char(52),char(57),char(117),char(58),char(52),char(117),char(78),char(77),char(72),char(75),char(55)),rowid)2)x from (select 1 union select 2)a group by x limit 1`. Injected payload found: `42d434339951`
- View HTTP headers**, **View HTML response**, **Launch exploit with HTTP Editor**, **Report alert(s)**, **Mark this alert as false positive**
- The impact of this vulnerability:** An attacker may execute arbitrary SQL statements on the vulnerable system. This may compromise the integrity of your database and/or expose sensitive information.
- Comments:** Depending on the back-end database in use, SQL injection vulnerabilities lead to varying levels of database system access for the attacker. It may be possible to not only manipulate existing queries, but to UPDATE arbitrary data, use user-defined or append additional queries. In some cases, it may be possible to read in or write out to files, or to execute shell commands on the underlying operating system.
- Critical SQL Servers such as Microsoft SQL Server contain stored and extended procedures (database server functions). If an attacker can obtain access to these procedures, it may be possible to compromise the entire machine.**
- How to fix this vulnerability:** Your script should filter malicious characters from user input. Check detailed information for more information about fixing this vulnerability.
- Detailed information:** Click here for more detailed information about this vulnerability
- Web references:**
 - [Basic SQL injection attack](#)
 - [Advanced SQL injection](#)
 - [Security Focus - Penetration Testing for Web Applications \(Part Two\)](#)
 - [More Advanced SQL Injection](#)

The bottom status bar shows activity logs and error logs, indicating the progress of the scan.

Not Your Average Web Crawler



```
[device:not-your-average-web-crawler tijme$ python3 example_extensive.py
Crawler started.
At 4% of 23 requests ([200] https://finnwea.com/).
At 8% of 23 requests ([200] https://finnwea.com/favicon.png).
At 13% of 23 requests ([200] https://finnwea.com/blog/a-web-application-crawler-for-bug-bounty-hunting/header.jpg).
At 17% of 23 requests ([200] https://finnwea.com/blog/scanning-people-via-ethereum-smart-contracts/header.jpg).
At 21% of 23 requests ([200] https://finnwea.com/blog/css-on-head-one-of-the-largest-dutch-franchisors/header.jpg).
At 26% of 23 requests ([200] https://finnwea.com/blog/stealing-passwords-from-mcdonalds-users/header.jpg).
At 38% of 23 requests ([200] https://finnwea.com/blog/adding-placeholders-to-uittextviews-in-swift/header.jpg).
At 54% of 23 requests ([200] https://finnwea.com/blog/i-decided-to-give-my-blog-another-upgrade/header.jpg).
At 37% of 24 requests ([200] https://finnwea.com/feeds/rss/).
At 41% of 24 requests ([200] https://finnwea.com/blog/securing-your-hometown-in-case-of-a-direct-terrorist-threat/header.jpg).
At 45% of 24 requests ([200] https://finnwea.com/hall-of-fame).
At 58% of 24 requests ([200] https://finnwea.com/about/).
At 54% of 24 requests ([200] https://finnwea.com/responsible-disclosure).
At 58% of 24 requests ([200] https://finnwea.com/about).
At 62% of 24 requests ([200] https://finnwea.com/css/finnwea_min.css).
At 59% of 27 requests ([200] https://finnwea.com/blog/i-decided-to-give-my-blog-another-upgrade/).
At 62% of 27 requests ([200] https://finnwea.com/hall-of-fame/).
At 66% of 27 requests ([200] https://finnwea.com/responsible-disclosure/).
At 61% of 31 requests ([200] https://finnwea.com/blog/xss-on-head-one-of-the-largest-dutch-franchisors/).
At 54% of 37 requests ([200] https://finnwea.com/blog/securing-your-hometown-in-case-of-a-direct-terrorist-threat/).
At 56% of 37 requests ([200] https://finnwea.com/blog/i-decided-to-give-my-blog-another-upgrade/design-2015.jpg).
At 59% of 37 requests ([200] https://finnwea.com/blog/i-decided-to-give-my-blog-another-upgrade/design-2014.jpg).
At 62% of 37 requests ([200] https://finnwea.com/blog/adding-placeholders-to-uittextviews-in-swift/).
At 64% of 37 requests ([200] https://finnwea.com/blog/scanning-people-via-ethereum-smart-contracts/).
At 67% of 37 requests ([200] https://finnwea.com/blog/i-decided-to-give-my-blog-another-upgrade/design-2016.jpg).
At 68% of 38 requests ([200] https://finnwea.com/blog/a-web-application-crawler-for-bug-bounty-hunting/).
At 58% of 54 requests ([200] https://finnwea.com/blog/stealing-passwords-from-mcdonalds-users/).
At 51% of 54 requests ([200] https://finnwea.com/blog/xss-on-head-one-of-the-largest-dutch-franchisors/xss-auditor-block.png).
At 53% of 54 requests ([200] https://finnwea.com/blog/xss-on-head-one-of-the-largest-dutch-franchisors/xss-payload-in-source.jpg).
At 55% of 54 requests ([200] https://finnwea.com/blog/xss-on-head-one-of-the-largest-dutch-franchisors/xss-alert-cookie.png).
At 57% of 54 requests ([200] https://finnwea.com/blog/xss-on-head-one-of-the-largest-dutch-franchisors/xss-stored-proof.png).
At 59% of 54 requests ([200] https://finnwea.com/blog/securing-your-hometown-in-case-of-a-direct-terrorist-threat/amsterdam-8-km.jpg).
At 61% of 54 requests ([200] https://finnwea.com/blog/securing-your-hometown-in-case-of-a-direct-terrorist-threat/amsterdam.png).
At 62% of 54 requests ([200] https://finnwea.com/blog/securing-your-hometown-in-case-of-a-direct-terrorist-threat/amsterdam-empty.jpg).
At 64% of 54 requests ([200] https://finnwea.com/blog/securing-your-hometown-in-case-of-a-direct-terrorist-threat/amsterdam-super-sink.jpg).
At 66% of 54 requests ([200] https://finnwea.com/blog/securing-your-hometown-in-case-of-a-direct-terrorist-threat/amsterdam-12-km.jpg).
At 68% of 54 requests ([200] https://finnwea.com/blog/stealing-passwords-from-mcdonalds-users/search-value-test-reflected.png).
At 79% of 54 requests ([200] https://finnwea.com/blog/stealing-passwords-from-mcdonalds-users/search-value-angular-id-reflected.png).
At 72% of 54 requests ([200] https://finnwea.com/blog/a-web-application-crawler-for-bug-bounty-hunting/nyanc-flow.svg).
At 74% of 54 requests ([200] https://finnwea.com/blog/stealing-passwords-from-mcdonalds-users/search-value-angular-id-preview.png).
At 75% of 54 requests ([200] https://finnwea.com/blog/stealing-passwords-from-mcdonalds-users/mcdonalds-login-form.png).
At 77% of 54 requests ([200] https://finnwea.com/blog/stealing-passwords-from-mcdonalds-users/search-value-test-preview.png).
At 79% of 54 requests ([200] https://finnwea.com/blog/stealing-passwords-from-mcdonalds-users/angular-version.png).
```

Theory

Key concepts

Target scope (reduction)

- Real-life scenario
- The subject is HTTP requests
- Moving certain requests out-of-scope

Multi-threading

- Real-life scenario
- The work is HTTP requests
- The employees are threads

Time To First Byte

- Real-life scenario
- The further the location, the longer it takes to get there
- HTTP requests need to go to a location (server)

Persistent HTTP connections

- Real-life scenario
- Call once with all information
- Instead of twice with half of the information

Theory

Measurements

Efficiency

Theorem (efficiency measurement)

The efficiency Y of a scanner is the improved runtime R_i compared to normal runtime R_n .

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

Effectiveness

Theorem (effectiveness measurement)

The effectiveness S of a scanner is the amount of vulnerabilities V an improved technology V_i finds on average compared to a normal technology V_n .

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

Research Methodology

Iterative Multimethodology

- Quantitative as well as qualitative research
- Iterate multimethodology for every sub-question

Research Iterations

Efficiency and effectiveness measurements

- Answered using theory

The efficiency of scanner concepts

- Quantitative research
- Three web applications will be tested
- Effectiveness and efficiency should be analysed
- Data should be analysed empirically

Technologies to improve the efficiency

- Qualitative and quantitative research
- Find or invent technologies to improve efficiency
- Effectiveness and compatibility should be analysed
- Data should be analysed empirically

A user friendly product

- Quantitative research
- The user-friendliness should be investigated
- Amount of user interactions should be analysed
- Data should be analysed empirically

Results

The efficiency of scanner concepts

Analysis

	Acunetix								Burp Suite							
	Effectiveness				Efficiency				Effectiveness				Efficiency			
	Vulns found (decimal & percentage)		Seconds elapsed (decimal & percentage)		Vulns found (decimal & percentage)		Seconds elapsed (decimal & percentage)		Vulns found (decimal & percentage)		Seconds elapsed (decimal & percentage)		Vulns found (decimal & percentage)		Seconds elapsed (decimal & percentage)	
	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.												
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.												
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.												

BurpSuite persistent HTTP connections

Capturing from Loopback: lo0

No.	Src port	Time	Source	Destination	Protocol	Length	Info
5	50204	0.000359	127.0.0.1	127.0.0.1	HTTP	366	GET /dvwa/vulnerabilities/ HTTP/1.1
7	80	0.043000	127.0.0.1	127.0.0.1	HTTP	4153	HTTP/1.1 200 OK (text/html)
17	50205	1.182630	127.0.0.1	127.0.0.1	HTTP	405	GET /dvwa/vulnerabilities/brute/ HTTP/1.1
59	80	1.216444	127.0.0.1	127.0.0.1	HTTP	5521	HTTP/1.1 200 OK (text/html)
73	50208	1.266287	127.0.0.1	127.0.0.1	HTTP	472	GET /dvwa/vulnerabilities/brute/?password=password&Logi
75	50209	1.266564	127.0.0.1	127.0.0.1	HTTP	405	GET /dvwa/vulnerabilities/brute/?password=password&Logi
121	80	1.289056	127.0.0.1	127.0.0.1	HTTP	5632	HTTP/1.1 200 OK (text/html)
167	80	1.309019	127.0.0.1	127.0.0.1	HTTP	5521	HTTP/1.1 200 OK (text/html)
177	50214	1.318665	127.0.0.1	127.0.0.1	HTTP	562	GET /dvwa/vulnerabilities/brute/?password=password&Logi
219	50217	1.334378	127.0.0.1	127.0.0.1	HTTP	404	GET /dvwa/vulnerabilities/exec/ HTTP/1.1
229	80	1.337771	127.0.0.1	127.0.0.1	HTTP	5632	HTTP/1.1 200 OK (text/html)
275	80	1.359320	127.0.0.1	127.0.0.1	HTTP	5365	HTTP/1.1 200 OK (text/html)
285	50220	1.378832	127.0.0.1	127.0.0.1	HTTP	652	GET /dvwa/vulnerabilities/brute/?password=password&Logi
331	80	1.407726	127.0.0.1	127.0.0.1	HTTP	5632	HTTP/1.1 200 OK (text/html)
341	50223	1.422683	127.0.0.1	127.0.0.1	HTTP	404	GET /dvwa/vulnerabilities/exec/ HTTP/1.1
347	50224	1.439540	127.0.0.1	127.0.0.1	HTTP	536	POST /dvwa/vulnerabilities/exec/ HTTP/1.1 (application.
389	80	1.456615	127.0.0.1	127.0.0.1	HTTP	5365	HTTP/1.1 200 OK (text/html)
427	50229	1.476877	127.0.0.1	127.0.0.1	HTTP	404	GET /dvwa/vulnerabilities/csrf/ HTTP/1.1
441	80	1.508409	127.0.0.1	127.0.0.1	HTTP	5376	HTTP/1.1 200 OK (text/html)
487	80	1.513268	127.0.0.1	127.0.0.1	HTTP	5467	HTTP/1.1 200 OK (text/html)
497	50232	1.521755	127.0.0.1	127.0.0.1	HTTP	742	GET /dvwa/vulnerabilities/brute/?password=password&Logi
543	80	1.539200	127.0.0.1	127.0.0.1	HTTP	5632	HTTP/1.1 200 OK (text/html)
553	50235	1.546881	127.0.0.1	127.0.0.1	HTTP	404	GET /dvwa/vulnerabilities/csrf/ HTTP/1.1
587	50238	1.564525	127.0.0.1	127.0.0.1	HTTP	484	GET /dvwa/vulnerabilities/csrf/?password_conf=password&
601	80	1.566228	127.0.0.1	127.0.0.1	HTTP	5467	HTTP/1.1 200 OK (text/html)

BurpSuite persistent HTTP connections

Wireshark - Follow HTTP Stream (tcp.stream eq 0) · wireshark_lo0_20180308112956_yceTUo

```
GET /dvwa/vulnerabilities/ HTTP/1.1
Host: localhost
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: keep-alive
Cookie: security=low; PHPSESSID=c4vcmpj8malh63vl3b144c2d50; dvwaSession=2

HTTP/1.1 200 OK
Date: Thu, 08 Mar 2018 10:30:16 GMT
Server: Apache/2.4.28 (Unix) PHP/7.1.7
Content-Length: 3879
Keep-Alive: timeout=1000, max=99999
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Packet 5. 1 client pkt, 1 server pkt, 1 turn. Click to select.

Entire conversation (4407 bytes) Show and save data as ASCII

Find: Find Next

Help Filter Out This Stream Print Save as... Back Close

Acunetix persistent HTTP connections

Loopback: lo0

The screenshot shows a NetworkMiner capture window with the expression `http && tcp.port == 80 && ip.dst_host == 192.168.22.231`. The table lists 78 captured HTTP requests. Most requests are GET requests for /dvwa/phpinfo.php, with varying parameters. Some requests result in 200 OK responses (e.g., length 63), while others result in 404 Not Found (e.g., length 505) or other status codes like 605. The protocol column shows all entries as HTTP.

No.	Src port	Time	Source	Destination	Protocol	Length	Info
10	80	-4.257489	192.168.22.231	192.168.22.231	HTTP	63	HTTP/1.1 200 OK (text/html)
12	65098	-4.124835	192.168.22.231	192.168.22.231	HTTP	644	GET /dvwa/phpinfo.php/<ScRiPt/acu
23	80	-4.121020	192.168.22.231	192.168.22.231	HTTP	63	HTTP/1.1 200 OK (text/html)
25	65098	-4.000278	192.168.22.231	192.168.22.231	HTTP	618	GET /dvwa/phpinfo.php/<%00ScRiPt%
27	80	-3.999887	192.168.22.231	192.168.22.231	HTTP	505	HTTP/1.1 404 Not Found (text/htm
29	65098	-3.938285	192.168.22.231	192.168.22.231	HTTP	628	GET /dvwa/phpinfo.php/<video><sou
40	80	-3.932120	192.168.22.231	192.168.22.231	HTTP	63	HTTP/1.1 200 OK (text/html)
42	65098	-3.890959	192.168.22.231	192.168.22.231	HTTP	623	GET /dvwa/phpinfo.php/<svg%09%0a%
44	80	-3.890685	192.168.22.231	192.168.22.231	HTTP	514	HTTP/1.1 404 Not Found (text/htm
46	65098	-3.704321	192.168.22.231	192.168.22.231	HTTP	350	GET /dvwa/ HTTP/1.1
48	80	-3.701018	192.168.22.231	192.168.22.231	HTTP	7829	HTTP/1.1 200 OK (text/html)
50	65098	-3.637471	192.168.22.231	192.168.22.231	HTTP	630	GET /dvwa/phpinfo.php/<isindex%20
61	80	-3.630809	192.168.22.231	192.168.22.231	HTTP	63	HTTP/1.1 200 OK (text/html)
63	65098	-3.516585	192.168.22.231	192.168.22.231	HTTP	697	GET /dvwa/phpinfo.php/<iframe%20s
76	80	-3.509879	192.168.22.231	192.168.22.231	HTTP	63	HTTP/1.1 200 OK (text/html)
78	65098	-3.453455	192.168.22.231	192.168.22.231	HTTP	605	GET /dvwa/phpinfo.php/<body%20onl

Results

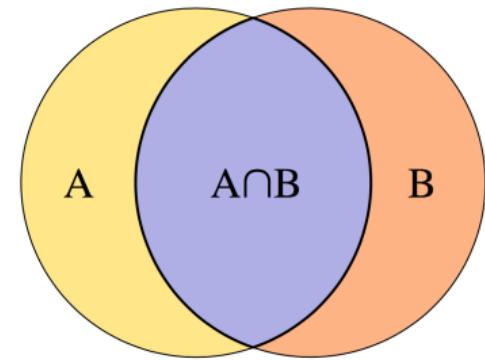
Technologies to improve the efficiency

Analysis

	Effectiveness		Efficiency	
	<i>Vulnerabilities found</i>		<i>Seconds elapsed</i>	
	<i>Decimal</i>	<i>Percentage</i>	<i>Decimal</i>	<i>Percentage</i>
Damn Vulnerable Web Application http://www.dvwa.co.uk/				
<i>Basis (without modification)</i>	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/				
<i>Basis (without modification)</i>	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/				
<i>Basis (without modification)</i>	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%

HTML tree similarity measure

- Proposed by the Northwave development team
- Jaccard similarity coefficient on HTML trees
- Can be used to measure similarity

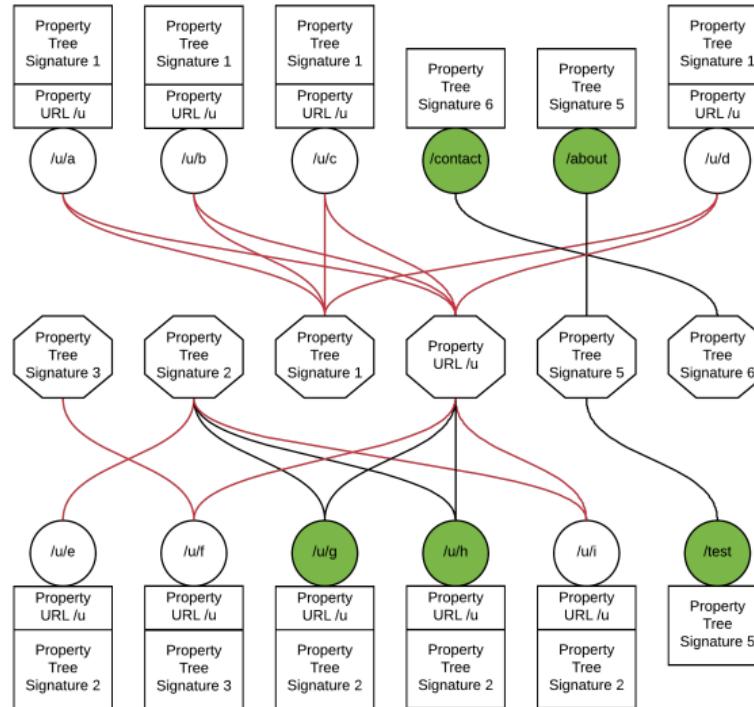


Piecewise response hashing

- Proposed by the Northwave development team
- Piecewise hashing on HTML trees
- Can be used to measure similarity

436:8dslkg48/f48fjoefjwfs:sfkjfw8w
436:8dslkg48/f48fjoefjwfs:Dfkjfw8w
Similarity: 99%

A custom undirected graph



Results

A user friendly product

Analysis

	<i>Acunetix</i>		<i>Burp Suite</i>		<i>NYAWC</i>	
	<i>Interactions</i>	<i>Seconds</i>	<i>Interactions</i>	<i>Seconds</i>	<i>Interactions</i>	<i>Seconds</i>
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

Conclusion

Goal

Get an answer to the question: Which user friendly product can be developed to improve the efficiency of scanners while maintaining effectiveness?

Efficiency and effectiveness measurements

- Important terms
- Formulas proof replicability

The efficiency of scanner concepts

- 45% efficiency improvement with scope reduction
- Persistent HTTP Connections cannot be changed
- Time To First Byte improvement is negligible

Technologies to improve the efficiency

- Three prototypes based on scope reduction
- Custom Graph Cut improves efficiency the most
 - 48% efficiency improvement
 - 12% effectiveness deterioration

A user friendly product

- An API would be the most user-friendly solution
 - Not accessible for some users
- Burp Suite extension is the second most user-friendly solution
 - Directly scan from within Burp Suite
 - Export URLs to another scanner to make it generic

Improving the efficiency of scanners

- The best solution is... GraphWave!
- A 100% effectiveness is not possible

Product Demo

GraphWave Demo Video

Discussion

The research is valid

- Tested three different scanners
- Tested three different web applications
- In case of repeat of this research;
 - The results would be the same
 - Therefore this research is valid

Unexpected results



GraphWave can improve efficiency

but



Effectiveness cannot always be maintained

Possible explanation

- Web applications require specific graph options
 - SEO sites should get more points for URL paths
 - Non SEO sites should get less points for URL paths

Advice for follow-up research

- Similar study to find out if GraphWave can be tweaked in such a way that;
 - The options are based on the web application being scanned
 - And the effectiveness can therefore be maintained at a higher level

Questions

Thank you