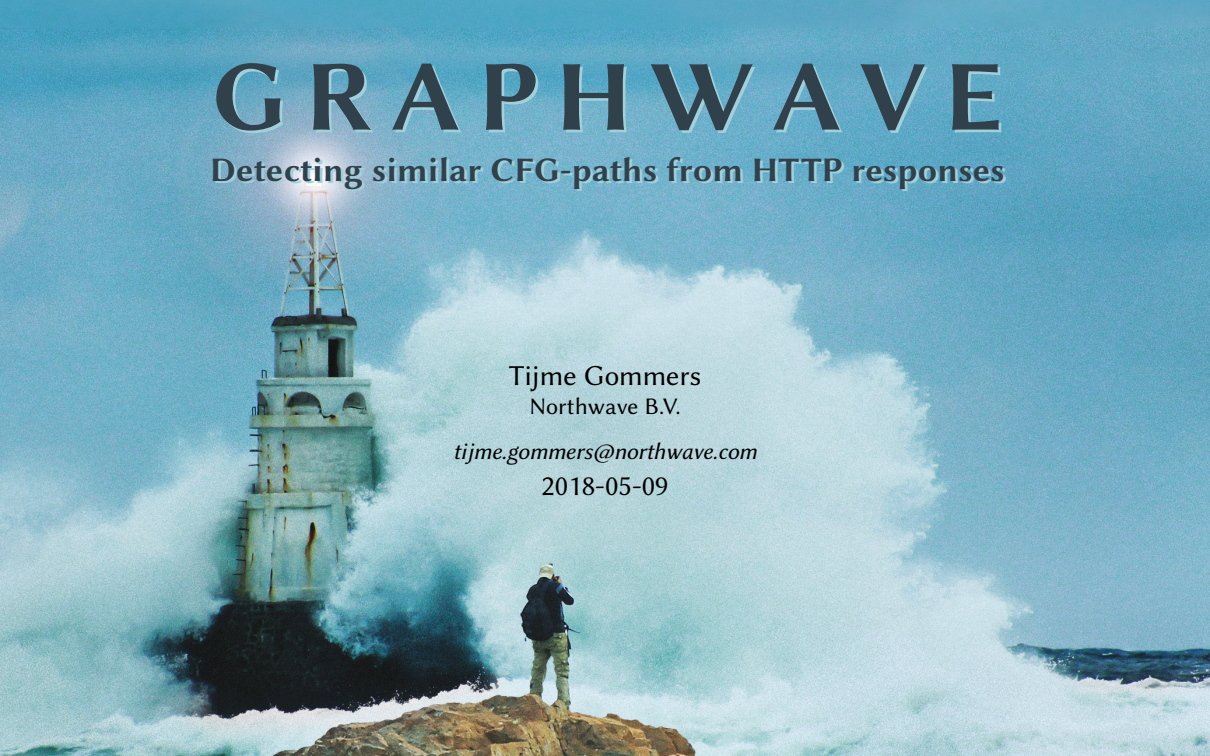
GRAPHWAVE

Detecting similar CFG-paths from HTTP responses



Tijme Gommers
Northwave B.V.

tijme.gommers@northwave.com

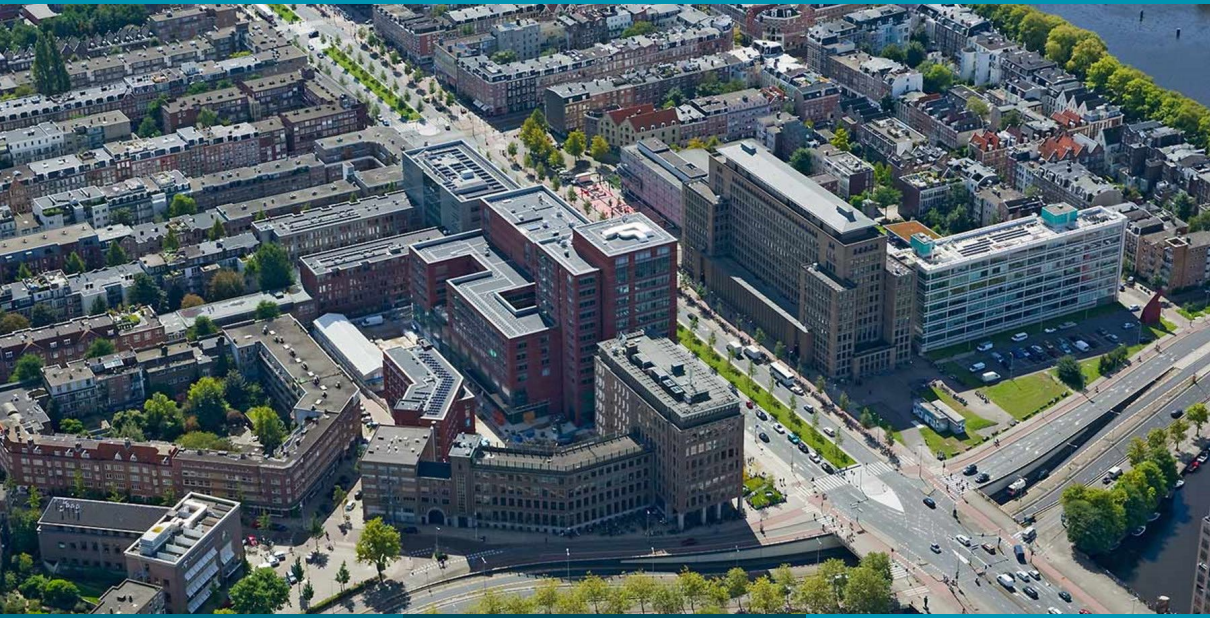
2018-05-09

Overview

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

36 hours

That's the time a scan at Northwave takes on average

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?

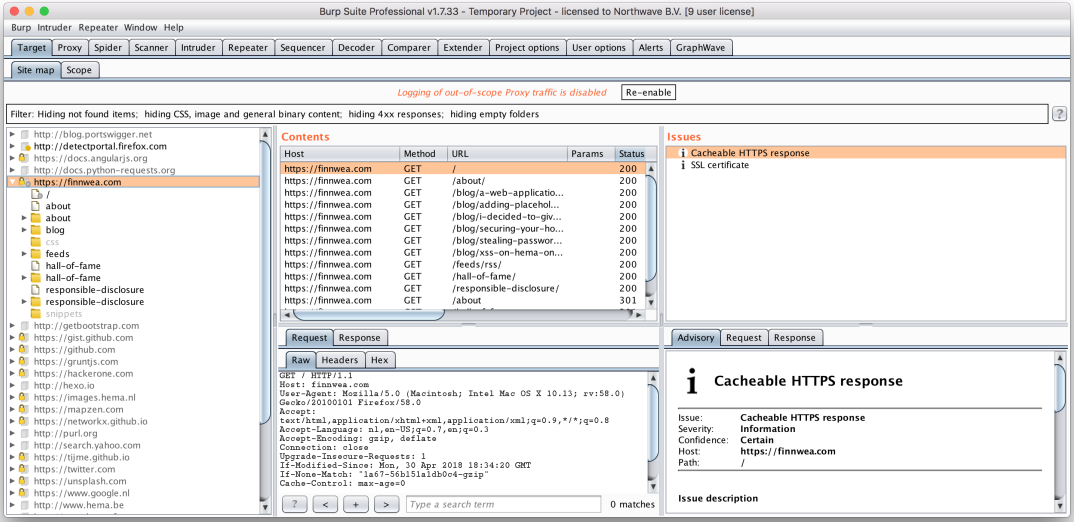
Which key concept of the scanners can improve efficiency the most while maintaining effectiveness?

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

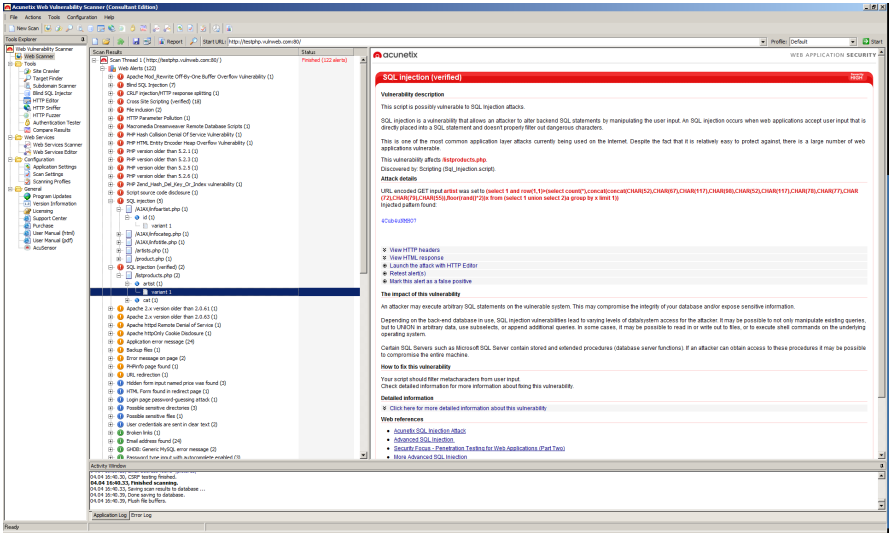
In which user friendly way can the most efficient and effective technology be integrated with scanners?

Theory
Automated Web Application Vulnerability Scanners

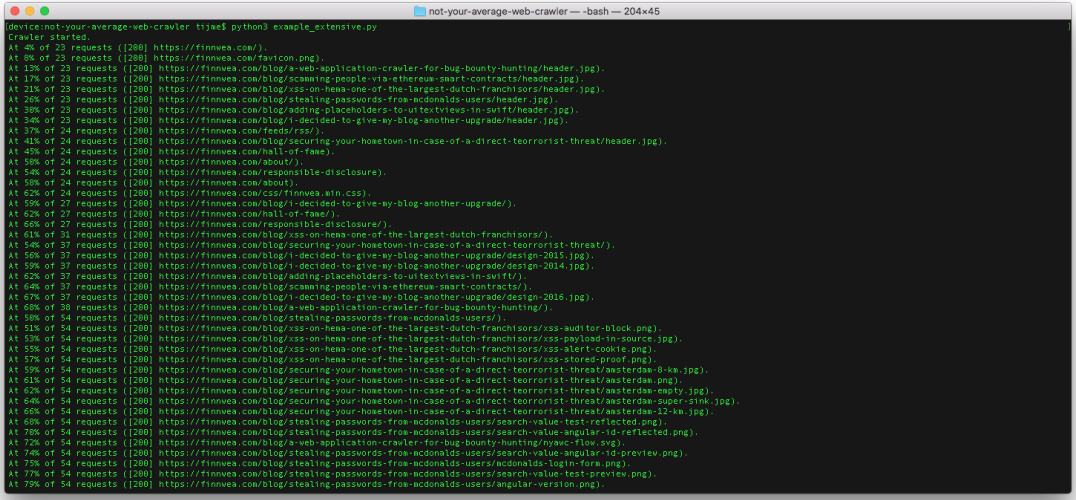
Burp Suite


Burp Suite Professional v1.7.33 - Temporary Project - licensed to Northwave B.V. [9 user license]
Burp Intruder Repeater Window Help
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
http://blog.portswigger.net
http://detectportal.firefox.com
https://docs.angularjs.org
http://docs.python-requests.org
https://finnwea.com
/
about
about
blog
css
feeds
hall-of-fame
hall-of-fame
responsible-disclosure
responsible-disclosure
snippets
http://getbootstrap.com
https://gist.github.com
https://github.com
https://gruntjs.com
https://hackerone.com
http://hexo.io
https://images.hema.nl
https://mapzen.com
https://networkx.github.io
http://purl.org
http://search.yahoo.com
https://tijme.github.io
https://twitter.com
https://unsplash.com
https://www.google.nl
http://www.hema.be
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: "1a67-56b151aldb0c4-gzip"
Cache-Control: max-age=0
? < + > Type a search term 0 matches
Advisory Request Response
Cacheable HTTPS response
Issue: Cacheable HTTPS response
Severity: Information
Confidence: Certain
Host: https://finnwea.com
Path: /
Issue description

Acunetix


Acunetix Web Vulnerability Scanner (Consultant Edition)
File Actions Tools Configuration Help
New Scan
Tools Explorer
Web Vulnerability Scanner
Web Scanner
Tools
Site Crawler
Target Finder
Subdomain Scanner
Blind SQL Injector
HTTP Editor
HTTP Sniffer
HTTP Fuzzer
Authentication Tester
Compare Results
Web Services
Web Services Scanner
Web Services Editor
Configuration
Application Settings
Scan Settings
Scanning Profiles
General
Program Updates
Version Information
Licensing
Support Center
Purchase
User Manual (html)
User Manual (pdf)
AcuSensor
Report
Start URL http://testphp.vulnweb.com:80/
Profile: Default
Start
Scan Results
Status
Finished (122 alerts)
Scan Thread 1 (http://testphp.vulnweb.com:80/)
Web Alerts (122)
Apache Mod_Rewrite Off-By-One Buffer Overflow Vulnerability (1)
Blind SQL Injection (7)
CRLF injection/HTTP response splitting (1)
Cross Site Scripting (verified) (18)
File inclusion (2)
HTTP Parameter Pollution (1)
Macromedia Dreamweaver Remote Database Scripts (1)
PHP Hash Collision Denial Of Service Vulnerability (1)
PHP HTML Entity Encoder Heap Overflow Vulnerability (1)
PHP version older than 5.2.1 (1)
PHP version older than 5.2.3 (1)
PHP version older than 5.2.8 (1)
PHP version older than 5.2.6 (1)
PHP Zend_Hash_Del_Key_Or_Index vulnerability (1)
Script source code disclosure (1)
SQL injection (5)
/AJAX/infoartist.php (1)
id (1)
variant 1
/AJAX/infocateg.php (1)
/AJAX/infotitle.php (1)
/artists.php (1)
/product.php (1)
SQL injection (verified) (2)
/listproducts.php (2)
artist (1)
variant 1
cat (1)
Apache 2.x version older than 2.0.61 (1)
Apache 2.x version older than 2.0.63 (1)
Apache httpd Remote Denial of Service (1)
Apache HttpOnly Cookie Disclosure (1)
Application error message (24)
Backup files (1)
Error message on page (2)
PHPinfo page found (1)
URL redirection (1)
Hidden form input named price was found (3)
HTML Form found in redirect page (2)
Login page password-guessing attack (1)
Possible sensitive directories (3)
Possible sensitive files (1)
User credentials are sent in clear text (2)
Broken links (1)
Email address found (24)
MySQL error message (2)
Password type input with autocomplete enabled (3)
acunetix
WEB APPLICATION SECURITY
SQL injection (verified)
Alert
Vulnerability description
This script is possibly vulnerable to SQL Injection attacks.
SQL injection is a vulnerability that allows an attacker to alter backend SQL statements by manipulating the user input. An SQL injection occurs when web applications accept user input that is directly placed into a SQL statement and doesn't properly filter out dangerous characters.
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 /listproducts.php.
Discovered by: Scripting (Sql_Injection.script).
Attack details
URL encoded GET input artist was set to (select 1 and row(1,1)=(select count(*),concat(concat(CHAR(52),CHAR(87),CHAR(117),CHAR(98),CHAR(52),CHAR(117),CHAR(78),CHAR(77),CHAR(72),CHAR(79)),CHAR(55),(floor(rand()*2)))x from (select 1 union select 2)a group by x limit 1))
Injected pattern found:
4Cub4uN9M07
View HTTP headers
View HTML response
Launch the attack with HTTP Editor
Retest alert(s)
Mark this alert as a 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.
Depending on the back-end database in use, SQL injection vulnerabilities lead to varying levels of data/system access for the attacker. It may be possible to not only manipulate existing queries, but to UNION in arbitrary data, use subselects, 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.
Certain 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 metacharacters 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
Acunetix SQL Injection Attack
Advanced SQL Injection
Security Focus - Penetration Testing for Web Applications (Part Two)
More Advanced SQL Injection
Activity Window
Application Log
Error Log
Ready

Not Your Average Web Crawler



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

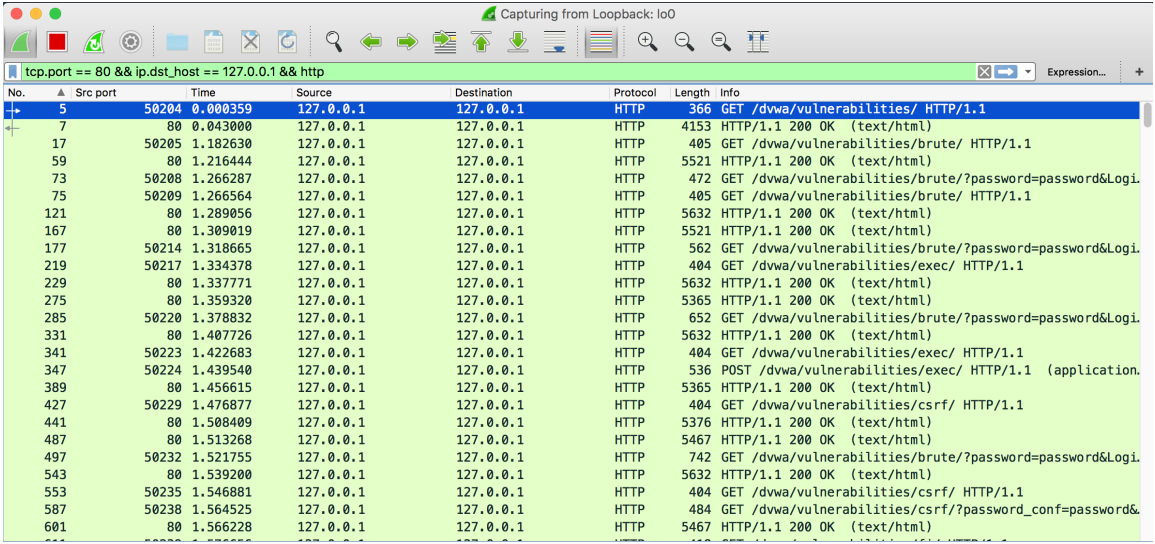
- Quantitative research
- The user-friendliness should be investigated
- Amount of user interactions should be analysed
- Data should be analysed empirically

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)*			
	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.

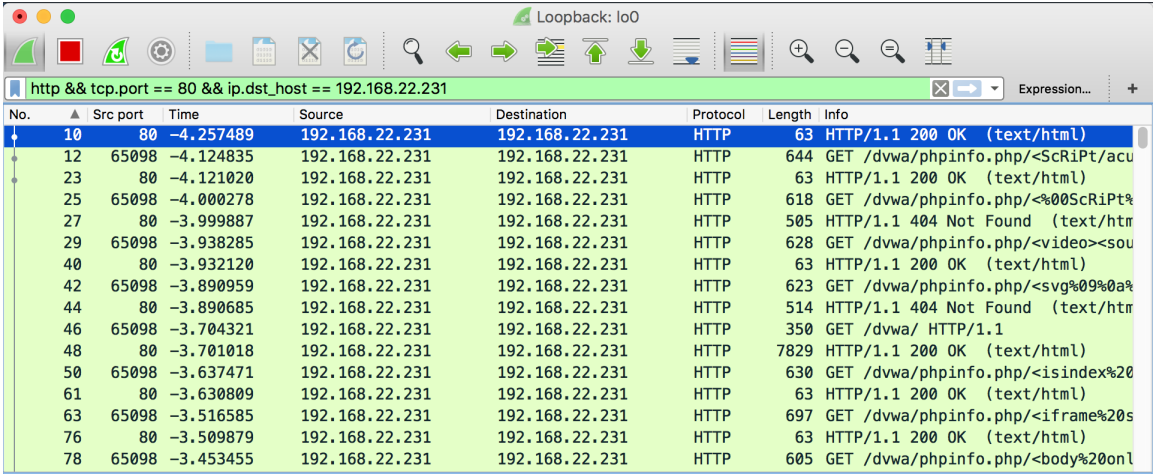
BurpSuite persistent HTTP connections



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/ HTTP/1.1
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

Acunetix persistent HTTP connections



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

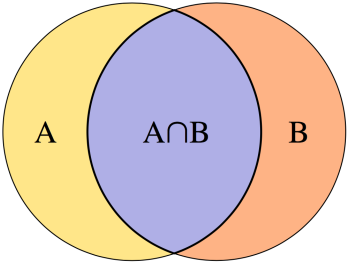
Results
Technologies to improve the efficiency

Analysis

	Effectiveness *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%**

HTML tree similarity measure

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

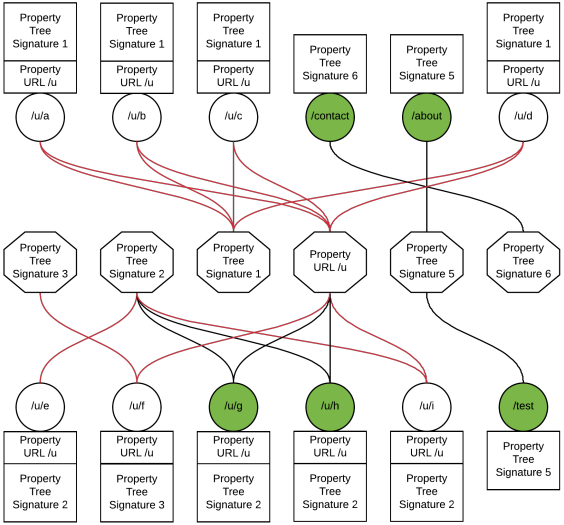


Piecewise response hashing

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

436:8dslkg48/f48fjoefjwfs:sfkjwfw8w
436:8dslkg48/f48fjoefjwfs:Dfkjwfw8w
Similarity: 99%

A custom undirected graph



Results
A user friendly product

Analysis

	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

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

- Thee 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