pentest
INFORMATION SECURITY ASSURANCE
A Shearwater Group plc Company

Report URI

Penetration Testing Report

2710

Report URI & API

29/11/2022

Author: Paul Ritchie

26a The Downs, Altrincham, Cheshire, WA14 2PU

Tel: +44 (0)161 233 0100

Web: www.pentest.co.uk

Table of Contents

1 Document Revision History

Name	Date	Version	Comment
Paul Ritchie	28/11/2022	0.1	Initial Document
Kyle Fleming	29/11/2022	0.2	QA by senior consultant.
Paul Ritchie	29/11/2022	1.0	Final Draft

2 Introduction

Report URI engaged Pentest Limited to undertake this project. This was to gain independent assurance that security controls are in-line with industry best practices.

Report URI was created to take the pain out of monitoring security policies like CSP and other modern security features. When you can easily monitor what's happening on your site in real time you react faster and more efficiently, allowing you to rectify issues without your users ever having to tell you. The Report URI platform is constantly evolving to help better protect your users.

Report URI are the best real-time monitoring platform for cutting edge web standards. Their experience, focus and exposure allow them to take the hassle out of collecting, processing, and understanding reports, giving you just the information you need.

Report URI have indicated the need for a security test, of their 'Report URI' application to identify vulnerabilities to attacks that could be launched across a computer network and to provide security assurances regarding their systems. Such a test will allow Report URI to undertake remediation efforts and increase their overall security posture.

2.1 Scope & Duration

This assessment included the following phases of work:

- Phase 1 – Web application and API assessment of the Report URI application

The duration included 5 days effort (including reporting). Work commenced on 21/11/2022 and concluded on 25/11/2022.

2.2 Scenarios Included

- **Black-box assessment** – simulating the threat of an Internet based attacker with no knowledge of the platform. This would locate unauthenticated access to services and find issues within the supporting infrastructure.
- **Rogue user** – simulating the threat of an authenticated user. This included access to paid and free tier accounts.
- **White-box assessment** – source code, code repositories, and access to Report URI staff was provided. This information is not available to Internet based attacker and helps to find defence-in-depth recommendations to further enhance a security posture.

2.3 Target(s)

The target included “*.report-uri.com” because users can generate their own subdomains. While some domains are reserved the following services are generally part of the target platform for all users:

- https://report-uri.com
- https://cdn.report-uri.com

3 Executive Summary

The targets were secure against many common vulnerabilities and presented a mature defence-in-depth security posture. In part this is because this is the third iteration of penetration testing which has helped to improve that posture. However, this is also because Report URI have demonstrated a willingness and culture to develop and deploy their solutions with a focus on security.

Report URI remained in contact throughout the engagement and provided all additional information that was requested. This collaboration enabled the consultant to deliver the highest quality assessment possible within the available time.

The following findings from previous tests were still applicable:

- Account Enumeration (Timing Difference)
- Excessive Session Timeout (24-hour validity)

Their risks have been accepted previously and there was no value in adding them again.

This report details three findings as summarised below:

- **Vulnerabilities in Outdated Software Detected** (Low Risk) – one outdated JavaScript library was located. This contained a known Cross-Site Scripting (XSS) vulnerability. Report URI did not use the vulnerable functionality so was not vulnerable to that XSS. A GitHub workflow already existed to detect outdated JavaScript libraries. This is the recommended approach to prevent outdated JavaScript libraries long term.
- **Account Enumeration** (Low Risk) – previous reports raised Account Enumeration in two other features of the site. A new instance was reported in the registration process. If a user was not previously registered, then they are automatically authenticated and redirected to "/account". Exploitation of this was already protected from automation by a CAPTCHA challenge and rate-limiting.

  An attacker with sufficient resources could pay humans to solve the CAPTCHAs and bypass the rate limiting. Doing so would allow the attacker to gather a list of valid usernames (email addresses) to permit Phishing and password guessing attacks.

  Due to the cost of exploiting it and the low value of that data the consultant doing so was considered extremely unlikely.

- **Server Side Request Forgery (SSRF)** (Informational) – Report URI offers several free tools which allow an attacker to control a URL that is requested by the backend server. These are available without registration, some do not have CSRF tokens, and none implement CAPTCHA technology. All of these are protected by rate-limiting technologies which would discourage heavily automated use.

  No access to internal resources was found on this occasion. There remains a potential that Report URI's service could be abused to send DoS traffic to third party hosts. This may impact Report URI's reputation if done.

No significant impacts were detected during this iteration of penetration testing.

3.1 Next Steps

A complete writeup of every issue is available in the body of this report. It includes required steps to confirm and replicate each issue, along with recommended remedial actions. Pentest recommend taking time to review the findings before arranging a triage meeting to determine the order of priority for remedial work. As a rule of thumb:

- **Critical Risk Items** – Address these immediately.
- **High Risk Items** – Address these as soon as possible after any Critical Risks.
- **Medium Risk Items** – Plan to address these within 3 months of discovery.
- **Low and Info Risk Items** – Track these within a risk register and discuss remediation versus acceptance.

If recommendations within this report are followed Pentest believe that the target's security posture will improve.

3.2 Caveats

Pentest provides no warranty that the target(s) are now free from other defects. Security is an ever-evolving field and consultancy is based on the opinions of the consultant, their understanding of the goals of Report URI as well as their individual experience.

The findings of this project are based on a time-limited assessment and by necessity can only focus on approved targets which are in scope. An attacker would not be constrained by either time or scope limits and could circumvent controls which are impractical to assess via structured penetration testing.

To appropriately secure assets Pentest encourage a cyclical approach to assessment. Each cycle should include:

- **Comprehensive Assessment** – where a full list of findings is produced with the widest scope possible.
- **Focused Verification Testing** – where solutions to the initial assessment's findings are verified.

Depending on how important the target is to the concerns of Report URI, Pentest recommend repeating the cycle every 6-months or 12-months at least.

3.3 Risk Categories & Rationales

Pentest use a simple risk categorisation of each vulnerability to focus the triage process at the risks which truly matter. The Common Vulnerability Scoring System (CVSS) is an industry standard formula. It generates a risk score between 0.0 and 10.0.

The table below explains the risk categories and demonstrates rule-of-thumb equivalency with CVSS scores:

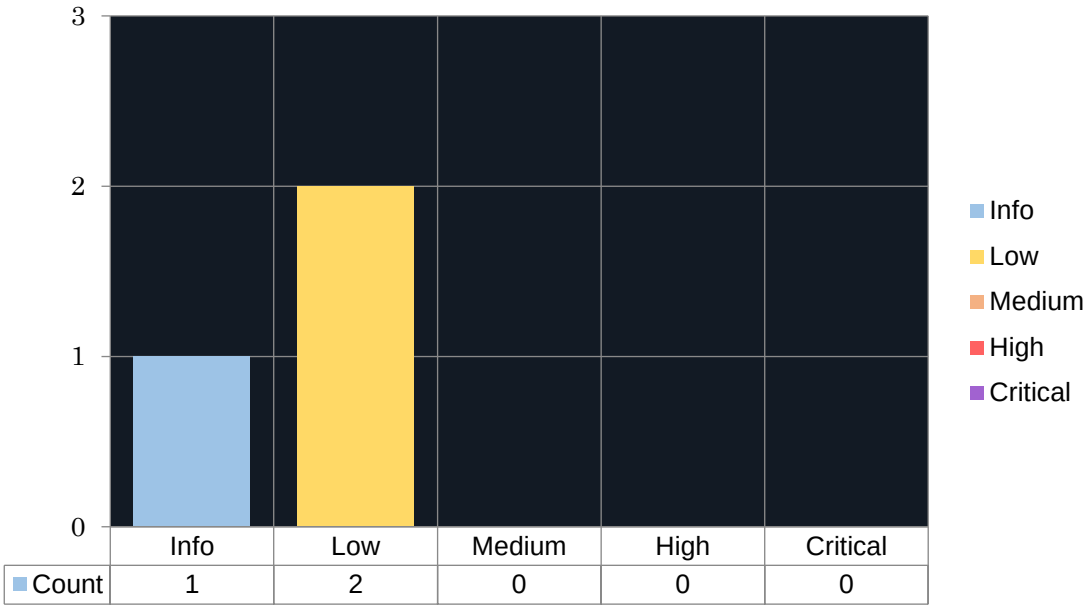
Risk Category	CVSS Score	Rationales
Critical	8.1 – 10.0	Poses a severe risk which is easy to exploit. Begin the process of remediating immediately after the issue has been presented.
High	6.1 – 8.0	Poses a significant risk and can be exploited. Address these as soon as possible after any critical risks have been remediated.
Medium	4.1 – 6.0	Poses an important risk but may be difficult to exploit. Pentest recommends remedial work within 3 months of discovery.
Low	2.1 – 4.0	Poses a minor risk or may be exceedingly difficult to exploit. Address these over the long-term during testing cycles
Info	0.0 – 2.0	Loss of sensitive information, or a discussion point. These are not directly exploitable but may aid an attacker. Remediate these to create a true defence-in-depth security posture,

CVSS is not applicable to all risks. For example, it is incapable of capturing the risk of a “flat network design”. Experience has told us that this is a “high” risk in most cases.

For this reason, the reader may find vulnerabilities which have no CVSS rating in our reports.

We endeavour to provide the reason for omitting the risk score when that is the case, and to provide CVSS by default in all applicable cases.

3.4 Visual Summary



4 Recommended Actions

ID	Vuln Title	Recommended Action	Pentest Risk Category	CVSS
1.	**Vulnerabilities in Outdated Software Detected**	Investigate why the current process did not trigger an alert in this case	Low	**6.1/Medium**
2.	**Account Enumeration**	Consider altering the registration flow to remove manual account enumeration opportunities.	Low	**3.7/Low**
3.	**Server Side Request Forgery (SSRF)**	Consider adding a CAPTCHA challenge to the tools available when unauthenticated.	Info	**0.0/None**

5 Technical Findings

The following findings from previous tests that were still applicable:

- Account Enumeration (Timing Difference)
  - Though a new instance was reported based simply on response difference.
- Excessive Session Timeout (24-hour validity)

Their risk had been accepted previously and there was no value in adding them again.

5.1 Vulnerabilities in Outdated Software Detected

5.1.1 Background

Software vendors release security updates to provide fixes to vulnerabilities in their software and missing any of these patches could result in services becoming outdated. Outdated services can expose a wide range of vulnerabilities, from client-side attacks such as Cross-Site Scripting to server-side attacks such as Remote Code Execution.

This becomes especially important for software that has become unsupported or obsolete. Unsupported software will not receive any new security patches when issues are identified. As such, any affected services utilising obsolete software will remain susceptible to any existing vulnerabilities and any new exploits that may be discovered in the future.

An abundance of outdated software on a network can also indicate a failure in policy to help ensure that software present on a network remains up to date and secure.

5.1.2 Details

The Report URI website used an outdated version of “jquery-ui.min.js”. This was confirmed by using the HTTP request and response below:

Request:

```
GET /products/ocsp_expect_staple HTTP/2
Host: report-uri.com
```

Response:

```
<script src="https://cdn.report-uri.com/libs/jqueryui/1.13.0/jquery-ui.min.js"
nonce="P1Ojek7XVIV82SxiH91HBJG5" integrity="sha256-
WpasVnoho7I5kgTE6i2dy4Ub9B0NuEZz2mWRNZ4nqJE= sha384-
SAXtuPnq/ihP6oe5Dw3x8iGQMVPpLeRT+fbdcAZh3GQdaPXKsfrN/kiuESXeKhR2 sha512-
ENRy1V8cqnDlXErBKUhUvSMMs125WTiikx/2QhNCQ9COACpP/8RsjJmkYfpJqNJgxbeSTLfbZDg+oYY6vIAuAA=="
crossorigin="anonymous">
```

The yellow highlight shows the URL which included the version number 1.13.0.

This version contained one publicly known vulnerability captured by CVE-2022-31160 (see reference [3]). The impact of this was Cross-Site Scripting (XSS) if an attacker could control parts of an HTML page on which the “checkboxradio” function was triggered. A simple proof-of-concept can be observed at reference [4].

With access to the source code the consultant used “grep” to find files which mentioned “checkboxradio”:

```
# grep -r -l "checkboxradio" report-uri
report-uri/public/cdn/libs/jqueryui/1.13.0/jquery-ui.min.js
```

Only the JavaScript file itself contained references to the vulnerable function. This was a good indicator that the application was not vulnerable to attack. However, future development of the site may introduce the vulnerability if the dependency is not updated.

5.1.3 Risk Analysis

Pentest Risk Category	Low
CVSS	6.1/Medium AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N
Explanation	The CVSS rating was set using the NIST entry for CVE-2022-31160. As this did not appear to be exploitable the consultant lowered the risk category to “low”. Future development of the site may accidentally allow exploitation.

5.1.4 Recommendation

Pentest recommend applying the latest security updates to the affected software as made available from the vendor's site.

The long-term recommendation would generally be to integrate checks that prevent future deployment of vulnerable dependencies. With access to the project’s Git repository, it was clear that this was already done by the “js_dep_check.yml” workflow. As this was the case the number of outdated dependencies was small.

Investigate how an outdated dependency had not been detected prior to this penetration test. There may be a gap in processes which could allow higher risk deviations in future.

5.1.5 References

[1]	OWASP: A9 2017 - Using Components with Known Vulnerabilities
[2]	CWE-1104: Use of Unmaintained Third Party Components
[3]	CVE-2022-31160 - Cross-Site Scripting
[4]	GitHub Security Advisories with PoC

5.1.6 Affected Item(s)

- https://report-uri.com/ - which loaded the script from the URL below:
  - https://cdn.report-uri.com/libs/jqueryui/1.13.0/jquery-ui.min.js

5.2 Account Enumeration

5.2.1 Background

Account Enumeration (also known as User Enumeration) is an issue that allows an unauthenticated/authenticated user to determine a user's account details (such as username or email address) due to information returned by an application.

Oftentimes, it is the discrepancy between responses from applications such as on Forgotten Password pages that allow attackers to determine the validity of user details. Examples of the types of account enumeration methods are as follows:

- Response discrepancy
- URL redirection
- Forced browsing

Whatever the method used the impact of this is that a list of valid usernames and/or email addresses can be created.

5.2.2 Details

On registering a new account there is an obvious difference in responses:

Condition	Response

Email address already registered	`HTTP/2 302 Found`
	`Date: Fri, 25 Nov 2022 10:22:32 GMT`
	`Content-Type: text/html; charset=utf-8`
	`Location: /login/`

Email address not registered	`HTTP/2 302 Found`
	`Date: Fri, 25 Nov 2022 10:18:42 GMT`
	`Content-Type: text/html; charset=utf-8`
	`Location: /account/`

This is because a newly registered account is automatically authenticated before the email verification process is completed.

The risk of this being exploited automatically was extremely low because:

- **CAPTCHA Technology** - the form was protected by a reCAPTCHA challenge.
- **Rate Limiting** – this defence exists to attackers coming from the Internet but for the duration of the test was disabled for the consultant.

These are best practice defences which meant that an attacker would manually be enumerating user accounts. The risk of this must simply be accepted because CAPTCHA challenges can always be defeated by paying humans to defeat them. However, Report URI is unlikely to be targeted by threat actors willing to do this.

Note: two other instances of user enumeration (in login and change email functions) were previously reported and the risks of those were already accepted.

5.2.3 Risk Analysis

Pentest Risk Category	Low
CVSS	3.7/Low AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N
Explanation	The risk posed by this was low in the consultant's opinion. Due to there being a minor loss in confidentiality this resulted in a "low" risk CVSS score even with the attack complexity set to high in this instance.

5.2.4 Recommendation

While exploitation of this was already challenging there remained an underlying risk on the registration form. It is possible to remove this difference by:

1) Altering the registration flow to force users to verify their email address before authentication.
2) In situations where the account does not exist:
    a. Email the user an activation URL.
    b. Ideally this process should not introduce a detectable delay. This can be achieved by sending the email in another thread and then redirecting as per 4) immediately instead of waiting for the email to be sent before returning.
3) In situations where the account does exist:
    a. Proceed straight to 4)
4) Redirecting the user to "/login" with the consistent message "Check your email and follow the activation process before you can authenticate".

5.2.5 References

[1]	OWASP: Authentication Cheat Sheet
[2]	Prevent account enumeration on login, reset password and registration pages
[3]	How serious is Username enumeration
[4]	CWE-200: Information Exposure
[5]	CWE-203: Information Exposure Through Discrepancy

5.2.6 Affected Item(s)

- POST /register/

5.3 Server Side Request Forgery (SSRF)

5.3.1 Background

Server-Side Request Forgery (SSRF) is a vulnerability that describes the behaviour of a server making a request that is under the attacker's control. When using a SSRF attack, an attacker can induce the server to perform actions on their behalf.

Typically, SSRF attacks are a result of the target application having the functionality for importing data from a URL or publishing data to a URL which can be tampered.

Using SSRF, an attacker an attacker may be able to connect to internal services which are not meant to be exposed to external users. This can be used for port scanning, or data exfiltration depending on the presentation.

5.3.2 Details

Report URI offer multiple features that can be used to trigger DNS, HTTP, and SMTP interactions with arbitrarily specified hosts. This is the fingerprint of an SSRF vulnerability.

However, all instances were found to be legitimate features that customers require. No data exfiltration or access to Internal resources was detected. There remained an obvious risk that an attacker could abuse this in some way. For example, they could send traffic to arbitrary hosts which may impact them by triggering a denial of service.

Even with that scenario in mind there were mitigations in place such as rate-limiting and Cloudflare protection to discourage abuse.

The following shows a discussion of one SSRF instance in the “/home/generate_hash” endpoint. The analysis provides context around what was possible through that feature.

5.3.2.1 HTTP Interaction via /home/generate_hash/

This feature was accessible while unauthenticated. It was designed to issue an HTTP request to any Internet facing URL provided. The following shows the baseline request which would illicit the HTTP request:

```
POST /home/generate_hash/ HTTP/2
Host: report-uri.com
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Content-Length: 34

url=https://pr.x-pt.net/goingforit
```

The consultant started an HTTPS service using Python which received the request as shown:

```
# python3 server.py

----- Request Start ----->

Host: pr.x-pt.net
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/40.0.2214.93 Safari/537.36
Accept: */*
Referer: https://report-uri.com
Origin: https://report-uri.com
```

```
<----- Request End -----

159.65.109.176 - - [23/Nov/2022 16:42:54] "GET /goingforit HTTP/1.1" 302 -
```

The HTTP request was issued from a Digital Ocean source IP address “159.65.109.176”. Repeating the request would find different source IP addresses such as “167.99.96.229” and “159.65.109.215”. All IPs were owned by Digital Ocean and the user agent was consistently the same indicating this was triggered by the same code path.

The table below lists probes which elicited different errors and attempts to explain the meaning:

Probe	Message	Meaning
`ftp://ftp.pentest.co.uk`	Please enter a valid address!	Protocol not allowed
`http://www.pentest.co.uk`	The server does not have CORS enabled. This resource can't be integrity checked.	Protocol allowed. DNS resolution successful. Port was open and accessible. Response did not include CORS headers.

`https://pr.x-pt.net/`
Note: listener was ncat with an invalid certificate, and it would not respond to HTTP:
`ncat -ssl -lvp 443`
	The operation timed out. Make sure the site is up and running and try again later.	Protocol allowed.
		Port was open and accessible.
		Response was not fast enough so connection was closed by client.

`https://pr.x-pt.net/`
Note: listener was ncat with an invalid certificate, but which responded with a valid 200 OK was used:
`printf 'HTTP/1.1 200 OK\r\nContent-Length: 2\r\nOK\r\n\r\n' | ncat --ssl -lvp 443`
	Something went wrong there! Maybe check the URL and try again?	Port was open and accessible.
		Possibly failed due to the certificate being insecure.

`https://pr.x-pt.net/` Note: listener was a python HTTPS service using a valid lets encrypt certificate.	The server does not have CORS enabled. This resource can't be integrity checked.	Server did not support CORS and so generate hash failed.
`https://pr.x-pt.net/`	This triggered a 500 Internal Server Error which was	The suspected cause was that the HTTPS service did not set the

Probe	Message	Meaning
Note: valid python HTTPS service with CORS header set to: `Access-Control-Allow-Origin: *`	caught by CloudFlare preventing the details being disclosed.	correct “Content-Length”. Without further access to the logs it was impossible to confirm this.
`https://pr.x-pt.net/` Note: valid python HTTPS service with CORS header. Which also set the “Content-Length” header correctly.	The same 500 Internal Server error as the previous attempt.	This check seemed to disprove the previous reasoning. It was assumed that the “Content-Type” may be necessary which had not been set in this test.

`https://pr.x-pt.net/`
Note: same as the previous attempt but with the additional header:
`Content-Type: text/javascript`
	This test resulted in the “generate_hash” function working as intended.	This was the required configuration for an HTTP server to work with this tool.

`https://pr.x-pt.net:9090/`	We only support default ports! Please remove the port number and try again.	Protocol allowed. Port 9090 was denied. Using Burp Suite’s Intruder ports 1-1024 were attempted. Same error shown for all ports apart from 443 which said “Something went wrong there! Maybe check the URL and try again?”. At the time no HTTPS listener was enabled. To access port 80 the protocol had to be switched to “http://”.
`https://192.168.0.1`	Oops, something went wrong! Check the URL and try again	Local IP addresses were banned. This was a previous finding by pentest.
`https://127.0.0.1`	Oops, something went wrong! Check the URL and try again	Access to localhost over HTTPS was not possible.

Probe	Message	Meaning
http://127.0.0.1	Oops, something went wrong! Check the URL and try again	Access to localhost over HTTP was not possible.

No access to Internal resources was detected.

Additionally, the consultant modified their HTTPS service to attempt 302 redirection such as this:

```
HTTP/1.0 302 Found
Server: BaseHTTP/0.6 Python/3.8.10
Date: Wed, 23 Nov 2022 23:19:52 GMT
Location: ftp://6dkty1vq0lkgwyaat4sbb3jnkeq5e12q.oastify.com/
```

No DNS request was issued for that domain so forcing a redirect did not allow the protocol to be altered from HTTP/HTTPS to FTP.

It was possible to downgrade from HTTPS to HTTP via a redirect which may be logically undesirable but did not seem exploitable.

It was not possible to bypass the port number restrictions using a redirect. This was an excellent configuration as sometimes SSRF requests validate the first request but not the onward redirection.

The consultant had access to a list of internal IP addresses from a previous iteration of the project. To confirm these could not be accessed Burp Suite's Intruder was configured with two payload sets:

- Position 1 – The two strings "http" and "https"
- Position 2 – A list of IPv4 and IPv6 addresses

With the injection points shown below:

```
POST /home/generate_hash/ HTTP/2
Host: report-uri.com
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Content-Length: 23

url=<POSITION_1>://<POSITION_2>
```

No access to internal resources was proved.

5.3.3 Risk Analysis

Pentest Risk Category	Info
CVSS	0.0/None AV:N/AC:L/PR:L/UI:N/S:C/C:N/I:N/A:N
Explanation	No access to internal resources was found on this occasion. There remains a potential that Report URI's service could be abused to send DoS traffic to third party hosts. This may impact Report URI's reputation if done. It is also noteworthy that there is rate limiting and other measures in place which would likely reduce the usefulness of the service for that purpose.

5.3.4 Recommendation

No impact to Report URI itself was detected on this occasion. The consultant has chosen to track this risk to continue generating a discussion over it. This is because there is always an element of risk involved in allowing external interaction.

The features on the "tools" menu could be protected further by adding CAPTCHA challenges to them. There is relatively little impact to usability if a user solves a single CAPTCHA before accessing these free tools legitimately. How often would a user legitimately need to access them? The answer was presumed to be only occasionally. Whereas threat actors would seek to automate and abuse the service and it would not be worth doing so if CAPTCHAs were enabled.

5.3.5 References

[1]	OWASP - Server Side Request Forgery
[2]	CWE-918 - SSRF
[3]	OWASP - Cheat Sheet SSRF Prevention

5.3.6 Affected Item(s)

- POST /home/generate_hash/
- POST /home/analyse_url/

6 Additional Information

6.1 WHOIS Database

The WHOIS database stores information about the individual or organisation who owns and manages a domain or IP address range. Attackers will review WHOIS entries trying to find useful information such as names and contact details for employees.

Best practices state that generic contact details should be used such as “whois@domain.com” rather than providing the name of a member of staff.

6.1.1 Entry for Domain: report-uri.com

```
whois report-uri.com
   Domain Name: REPORT-URI.COM
   Registry Domain ID: 1651365076_DOMAIN_COM-VRSN
   Registrar WHOIS Server: whois.namecheap.com
   Registrar URL: http://www.namecheap.com
   Updated Date: 2022-03-18T07:43:44Z
   Creation Date: 2011-04-17T11:55:31Z
   Registry Expiry Date: 2023-04-17T11:55:31Z
   Registrar: NameCheap, Inc.
   Registrar IANA ID: 1068
   Registrar Abuse Contact Email: abuse@namecheap.com
   Registrar Abuse Contact Phone: +1.6613102107
   Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
   Name Server: CARL.NS.CLOUDFLARE.COM
   Name Server: COCO.NS.CLOUDFLARE.COM
   DNSSEC: signedDelegation
   DNSSEC DS Data: 2371 13 2
B86DC8BE786CAFA5B1D92F52AA23CD9B62AF70DBE9D907AC61A1F9469513B5F6
   URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
```

No personal information was disclosed in this.

Additionally, the above indicated that DNSSEC was enabled. This is rarely implemented, and it was excellent to see that it had been enabled.

6.1.2 Entry for IP Address Range: 104.16.0.0 - 104.31.255.255

The target host “report-uri.com” was load balanced across five IP addresses. This was determined by using “nslookup” in a bash for loop:

```
# for i in `seq 1 100`; do dig +short report-uri.com; done | sort -u
104.17.182.88
104.17.183.88
104.17.184.88
104.17.185.88
104.17.186.88
```

The target used Cloudflare as shown:

```
NetRange:       104.16.0.0 - 104.31.255.255
CIDR:           104.16.0.0/12
NetName:        CLOUDFLARENET
NetHandle:      NET-104-16-0-0-1
Parent:         NET104 (NET-104-0-0-0-0)
NetType:        Direct Allocation
OriginAS:       AS13335
Organization:   Cloudflare, Inc. (CLOUD14)
RegDate:        2014-03-28
Updated:        2021-05-26
Comment:        All Cloudflare abuse reporting can be done via
https://www.cloudflare.com/abuse
Ref:            https://rdap.arin.net/registry/ip/104.16.0.0
[...] Snip [...]
```

No personal information was disclosed in this configuration.

6.2 DNS Reconnaissance

Domain Name Service (DNS) is used to translate human readable hostnames such as “www.pentest.co.uk” to the IP address which is hard for humans to recall. Threat actors use DNS reconnaissance to identify hosts which they can subsequently target.

6.2.1 Identifying DNS Servers for Domain: report-uri.com

The following shows the “dig” command being used to identify the name servers responsible for the target domain:

```
# dig ns report-uri.com

; <<>> DiG 9.16.1-Ubuntu <<>> ns report-uri.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29999
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;report-uri.com.                        IN      NS

;; ANSWER SECTION:
report-uri.com.         120     IN      NS      carl.ns.cloudflare.com.
report-uri.com.         120     IN      NS      coco.ns.cloudflare.com.
```

The target used Cloudflare’s DNS service which is designed to be “always available” and has integrated support for DDoS and DNSSEC.

This was an excellent configuration and would likely ensure the availability of DNS.

6.2.2 DNS Server Configurations

The following table summarises common insecure configurations. The data was gathered by assessing each of the NS servers listed above:

Check	Outcome
Zone Transfers Disabled	Transfer failed
DNSSEC Enabled	DNSSEC was enabled.
Recursive Queries Disabled	Cloudflare's DNS servers allowed recursive queries. This may enable DNS amplification attacks. Based on the information at this URL they prevent incoming DDoS via DNS amplification: https://www.cloudflare.com/en-gb/learning/ddos/dns-amplification-ddos-attack/ No information was found about how they prevent their name servers being used in an attack against another infrastructure.

Table 1 - DNS Server Configuration Analysis

6.2.3 List of known hostnames at Domain: report-uri.com

DNS wildcards were used such that all subdomains will point to one of the five IP addresses below:

```
104.17.184.88
104.17.185.88
104.17.186.88
104.17.183.88
104.17.182.88
```

A brute force did not find any additional hostnames for any other IP address.

6.3 Port Scan Results

To offer a service to other computers, a “port” is made available. Each open port creates a communication channel which can pose a security risk that an attacker can enumerate information from, or at worst exploit to compromise the target.

Best practices state that only the minimum number of open ports should be enabled to reduce the attack surface.

6.3.1 Target: 104.17.183.88 - report-uri.com

Port	State	Service	Product	Version	Extra
80/tcp	open	http	cloudflare	Unknown	Unknown
443/tcp	open	https	cloudflare	Unknown	Unknown
2052/tcp	open	clearvisn	Unknown	Unknown	Unknown
2053/tcp	open	http	nginx	Unknown	Unknown
2082/tcp	open	infowave	Unknown	Unknown	Unknown
2083/tcp	open	http	nginx	Unknown	Unknown
2086/tcp	open	gnunet	Unknown	Unknown	Unknown
2087/tcp	open	http	nginx	Unknown	Unknown
2095/tcp	open	nbx-ser	Unknown	Unknown	Unknown
2096/tcp	open	http	nginx	Unknown	Unknown
8080/tcp	open	http-proxy	cloudflare	Unknown	Unknown
8443/tcp	open	https-alt	cloudflare	Unknown	Unknown
8880/tcp	open	cddbp-alt	Unknown	Unknown	Unknown

The above is a typical footprint for a server fronted by CloudFlare.

6.4 SSL/TLS Assessment

Transport Layer Security (TLS) is used to ensure the confidentiality and integrity of traffic as it transits a network. It is also used to give certainty of the identity of the client, server, or both. Insecure configurations are common. The following sub-sections show information gathered using SSLScan.

6.4.1 SSLScan Results for: report-uri.com – TCP/443

```
Testing SSL server report-uri.com on port 443 using SNI name report-uri.com

  SSL/TLS Protocols:
SSLv2     disabled
SSLv3     disabled
TLSv1.0   disabled
TLSv1.1   disabled
TLSv1.2   enabled
TLSv1.3   enabled

  TLS Fallback SCSV:
Server supports TLS Fallback SCSV

  TLS renegotiation:
Session renegotiation not supported

  TLS Compression:
Compression disabled

  Heartbleed:
TLSv1.3 not vulnerable to heartbleed
TLSv1.2 not vulnerable to heartbleed

  Supported Server Cipher(s):
Preferred TLSv1.3  128 bits  TLS_AES_128_GCM_SHA256        Curve 25519 DHE 253
Accepted  TLSv1.3  256 bits  TLS_AES_256_GCM_SHA384        Curve 25519 DHE 253
Accepted  TLSv1.3  256 bits  TLS_CHACHA20_POLY1305_SHA256  Curve 25519 DHE 253
Preferred TLSv1.2  256 bits  ECDHE-ECDSA-CHACHA20-POLY1305 Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-ECDSA-AES128-GCM-SHA256 Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-ECDSA-AES128-SHA        Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-ECDSA-AES256-GCM-SHA384 Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-ECDSA-AES256-SHA        Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-ECDSA-AES128-SHA256     Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-ECDSA-AES256-SHA384     Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-CHACHA20-POLY1305   Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-GCM-SHA256   Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  AES128-GCM-SHA256
Accepted  TLSv1.2  128 bits  AES128-SHA
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-GCM-SHA384   Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA          Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  AES256-GCM-SHA384
Accepted  TLSv1.2  256 bits  AES256-SHA
Accepted  TLSv1.2  128 bits  ECDHE-RSA-AES128-SHA256       Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  AES128-SHA256
Accepted  TLSv1.2  256 bits  ECDHE-RSA-AES256-SHA384       Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  AES256-SHA256

  Server Key Exchange Group(s):
TLSv1.3  128 bits  secp256r1 (NIST P-256)
TLSv1.3  192 bits  secp384r1 (NIST P-384)
TLSv1.3  260 bits  secp521r1 (NIST P-521)
TLSv1.3  128 bits  x25519
TLSv1.2  128 bits  secp256r1 (NIST P-256)
TLSv1.2  192 bits  secp384r1 (NIST P-384)
TLSv1.2  260 bits  secp521r1 (NIST P-521)
TLSv1.2  128 bits  x25519

  SSL Certificate:
Signature Algorithm: ecdsa-with-SHA384
ECC Curve Name:      prime256v1
ECC Key Strength:    128

Subject:  *.report-uri.com
Altnames: DNS:*.report-uri.com, DNS:report-uri.com
Issuer:   E1

Not valid before: Nov 24 09:12:02 2022 GMT
Not valid after:  Feb 22 09:12:01 2023 GMT
```

No vulnerabilities were detected in the TLS configuration.

Appendix A. Testing Notes

A1. Source Code Analysis

Access to a private GitHub repository was provided. The code from the “master” branch was said to be what was deployed currently so the zip file was obtained and used.

This project was not a complete secure code review in the sense that the entire code base was to be read and validated. Instead, the code was generally used to aid otherwise black-box testing.

However, the following steps were followed:

- Static code analysis tools (VisualCodeGrepper, and semgrep) were used to point towards potential flaws.
- Manual analysis of a few “hits” for each category of vulnerability was conducted.

Any vulnerabilities were raised in the body of this report.

A2. Dependency Checking

With access to the source code the consultant used OWASP’s dependency-check to scan for known vulnerabilities in supporting libraries:

```
# dependency-check.bat --project report-uri -s
D:\work\Projects\2710_ReportUri\source\report-uri -f ALL
```

This detected one outdated dependency:

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count
jquery-ui.min.js		pkg:javascript/jquery-ui@1.13.0	MEDIUM	1

Figure 1 - Outdated version of jquery-ui.min.js

The same outdated dependency was also detectable using black-box techniques.

This was evidence that supporting libraries were generally updated and did not indicate a pattern of insecure practice.

A3. GitHub Repository Secrets Hunting

With access to the git repository analysis the consultant used “gitleaks” to check for potential secrets within the git repository. This was done using

```
# docker run -v ./source/report-uri:/path zricethezav/gitleaks:latest detect --
source="/path" -v -r /path/gitleaksecrets.json -f json --no-git
[...]
3:04PM INF scan completed in 12.7s
3:04PM WRN leaks found: 8
```

Note: “—no-git” meant no checks focused on the files as they were currently committed and did not check previous versions of files.

Several of these gits were due to the application processing public and private key files. The string used to match a secret was a false positive in these cases since they were not hard coded certificates which would pose a risk if leaked.

A few more were API keys which existed inside test classes. These also did not pose a risk as they were test API keys.

No exposure was detected as a result of this.

A4. Bug Hunting in Pem Decoder [Not Vulnerable]

URL: https://report-uri.com/home/pem_decoder/

This feature offered to decode pem files and display them on screen within the target domain. It was also not protected by a CSRF token making it possible to exploit XSS by CSRF if such a vulnerability existed.

To test this function a pem file must be generated. Initially this was done by using “openssl”:

```
# openssl req -newkey rsa:2048 -new -nodes -x509 -days 3650 -keyout key.pem -out cert.pem
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:aa
State or Province Name (full name) [Some-State]:bb
Locality Name (eg, city) []:cc
Organization Name (eg, company) [Internet Widgits Pty Ltd]:dd
Organizational Unit Name (eg, section) []:ee
Common Name (e.g. server FQDN or YOUR name) []:<xss>
Email Address []:ff
```

A simple XSS probe was provided in the “Common Name” field:

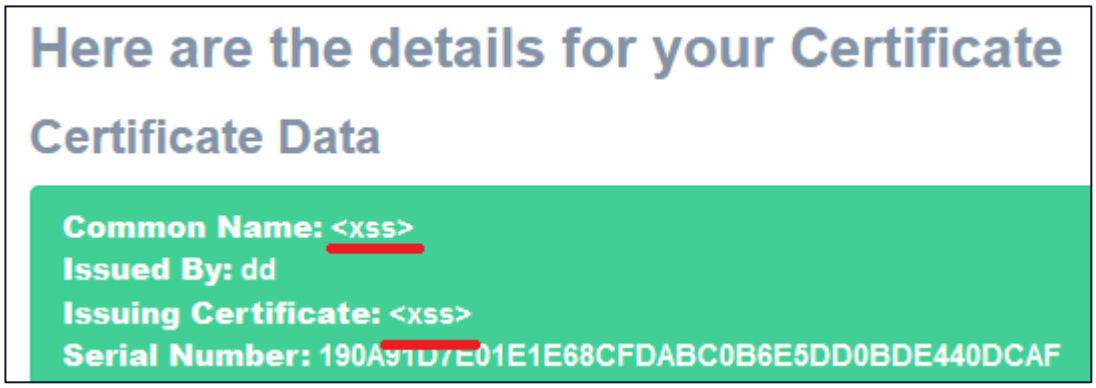


Figure 2 - Probe as Displayed in Browser

The probe was inert as displayed. This was because the server’s response had HTML entity encoded:

```
<b>Common Name:</b> &lt;xss&gt;<br><b>
```

While using output encoding like this is secure. The most secure approach is to reject requests containing inappropriate input. Attempts to sanitise input can often be bypassed (though this was unlikely in this context).

To assess this further URL encoding was used for the common name to produce the probe as:

```
Unencoded:   <xss>
URL Encoded: %3c%78%73%73%3e
```

The application handled this Pem file securely and no vulnerability was detected.

While nothing exploitable was detected the endpoint could improve its security by:

- Requiring a CSRF token.
- Not rendering certificates where the data fields do not match expected input patterns.

A5. Automated Application Scanning Challenges

The site used a CSRF token which was submitted as both a session cookie and post body parameter. This token changed with every request and had a built-in timeout feature as well making them automatically invalid.

Without configuring Burp suite appropriately this would result in inaccurate results as scans would present requests determined as invalid which may be protecting the underlying function.

Note: Attackers from the Internet would have a tougher time as they would also contend with Cloudflare and rate-limiting meaning that this section is not a recipe for how to automate an effective attack against Report URI.

The consultant used the “Stepper” (an extender for Burp Suite) to allow automated scanning. This was configured to gather a new CSRF token and derive a post response variable called “csrf”.

This was then added to requests in repeater. For example, the following shows a configuration used for the “/team/create_team” endpoint:

```
POST /team/create_team/ HTTP/2
Host: report-uri.com
Cookie: _nss=1; __Host-report_uri_sess=e7f6tfj83t0c9pu72qubbf2nd0; __Host-
report_uri_csrf=$VAR:2:csrf$; __cf_bm=bB4RNxpq9wSx1p.k1jNw8tFxXeWAyAhmTVQQ2AvfeBA-
1669115982-0-
AaPjHWFB2iNfdhVyOBVILKtDPL1V71xVwpRWGfcbi/CyWYJSELGIBL4a8UJKCXZOuhSNK3DWGFXkMe4J4UzEvD4=
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101
FirefoxXSS/98.0 [PenTestA]
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://report-uri.com/account/teams/
Content-Type: application/x-www-form-urlencoded
Content-Length: 33
Origin: https://report-uri.com
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
X-Stepper-Execute-Before: 2

csrf_token=$VAR:2:csrf$&name=test
```

Note the “X-Stepper-Execute-Before” header was added to trigger the sequence called “2”. The highlighted variables show where the “csrf” variable was injected.

With this configured it was possible to use Burp Suite’s repeater to manually alter the “name” parameter and to confirm that new teams had been created.

The consultant used Burp Suite's Intruder feature to probe for SQL Injection. By using intruder, it was easy to see the HTTP responses for each probe to monitor that the process was working as intended. During this process various responses were detected:

Code	Summary
302 to "/account/teams"	The request was valid and handled by the server.
403 with the message "Sorry, you have been blocked"	Cloudflare had blocked the request.

The above indicated that Cloudflare was likely rate limiting the number of automated requests making it harder to find underlying weaknesses.

During this automated check if the consultant browsed to the "/account/teams" URL a different response was observed:

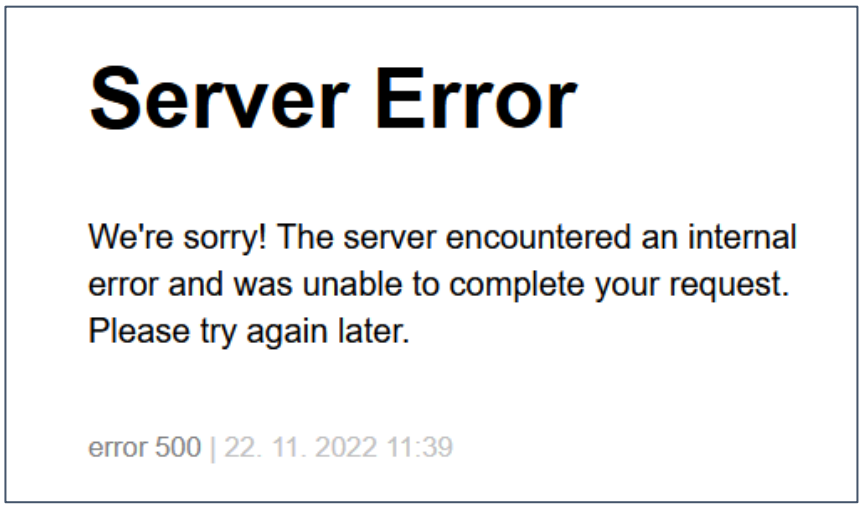


Figure 3 - Server Error

This error persisted long after the Intruder scan had been completed. The following shows part of the HTML source:

```
<p><small>error 500 <span> | 22. 11. 2022 12:05</span></small></p>
</div>
</div>
<script>
    document.body.insertBefore(document.getElementById('tracy-error'), d
</script>
<script defer src="https://static.cloudflareinsights.com/beacon.min.js/v
```

Figure 4 - Cloudflare

Scott Helme confirmed that the "tracy-error" was likely coming from the target server and not from Cloudflare. This error denied access to the feature for the duration of the test within the privileges of that user account. It had no impact on other users. This error may have obscured a vulnerability, but it was unknown what that would be.

That user account was then unable to authenticate to the application without causing an error and all HTTP post requests after login resulted in the same 500 error page. This was likely due to an error where the user's teams were retrieved. Any part of the site that needed to do this was inaccessible. It was not possible to proceed with testing using this account.

The consultant registered a new account and then used the same Stepper configuration to scan the "/account/ct_filters" endpoint. The baseline request for Burp's Intruder is shown below:

```
POST /account/ct_filters/ HTTP/2
Host: report-uri.com
Cookie: _nss=1; __Host-report_uri_csrf=$VAR:3:csrf$; __Host-
report_uri_sess=2nr0a4kur04iebthctkcar6857;
__cf_bm=YtURNrK6jxAIqqw7UHpJdUbRR8jHPjQFaOaItsoP9bA-1669133712-0-
AZ3n/KaM+Hwk63eh4pvcZUuvRUJ00hSZryfkvsqNjvawSil+tBzibHBnyamMSj58U3foIDu3IK6iMV7BeGUZcqM=
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101
FirefoxXSS/98.0 [paulr2]
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://report-uri.com/account/filters/
Content-Type: application/x-www-form-urlencoded
Content-Length: 35
Origin: https://report-uri.com
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
X-Stepper-Execute-Before: 3
Connection: close

csrf_token=$VAR:3:csrf$&ctHosts=<INJECT_HERE>
```

Using the default "fuzzing full" word list worked without causing the same error as discussed previously. The process found that input validation of the "ctHosts" parameter was robust, and all probes triggered an error about the format:

Payload	Status	Error	Redirect...	Timeout	Length	-danger"><p>
http://{domain}	200	☐	1	☐	34160	ERROR: The collect hosts string contained invalid characters.
https://{domain}	200	☐	1	☐	34160	ERROR: The collect hosts string contained invalid characters.
%20{!xmlparser v='<!DOCTYPE ...	200	☐	1	☐	34160	ERROR: The collect hosts string contained invalid characters.
" {!xmlparser v='<!DOCTYPE a ...	200	☐	1	☐	34161	ERROR: The collect hosts string contained invalid characters.

Figure 5 - Intruder Showing Some Responses with the input validation error

The following shows how the error appeared rendered in a browser:

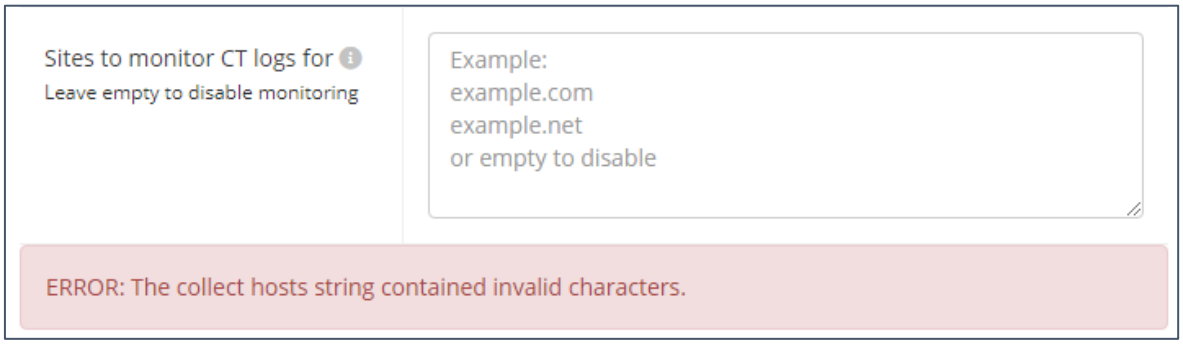


Figure 6 - "The collect hosts string contained invalid characters"

This filter was not bypassed, and no risk was determined.

This section has discussed how automation was configured to enable it to find results. After the first endpoint (“/teams/create_team”) presented with problems, the consultant documented similar work on “/account/ct_filters” to confirm that the process was generally working for scanning.

No additional effort to document how things would be setup was made and instead any vulnerabilities would be documented.

pentest

INFORMATION SECURITY ASSURANCE

A Shearwater Group plc Company

26a, The Downs
Altrincham
Cheshire
WA14 2PU

+44 (0)161 233 0100