APPLICATION PENETRATION TESTING REPORT

CONFIDENTIAL

DOCUMENT CONTROL

CLASSIFICATION	CONFIDENTIAL	DATE	14/11/2019
VERSION NO.	2.2	DOCUMENT REFERENCE	**BPEC210619-16**
DOCUMENT TITLE	Bulletproof – SocialSignin Limited (T/As Orlo) - Penetration Test Report		
APPROVED BY	Chay Donohoe		

VERSION HISTORY

VERSION NO	DATE	AUTHOR	COMMENTS
0.1	05/08/2019	Chay Donohoe	Document Creation
0.2	09/08/2019	Chay Donohoe	Document Finalised
0.3	15/08/2019	Victoria Bucknell	QA
1.0	20/08/2019	Jason Charalambous	Document Approved
1.1	14/11/2019	Victoria Bucknell	Report Split
2.0	18/11/2019	Kieran Roberts	Document Approval
2.1	13/12/2019	Victoria Bucknell	Final QA
2.2	20/12/2019	Chay Donohoe	Document Approval

CONTENTS

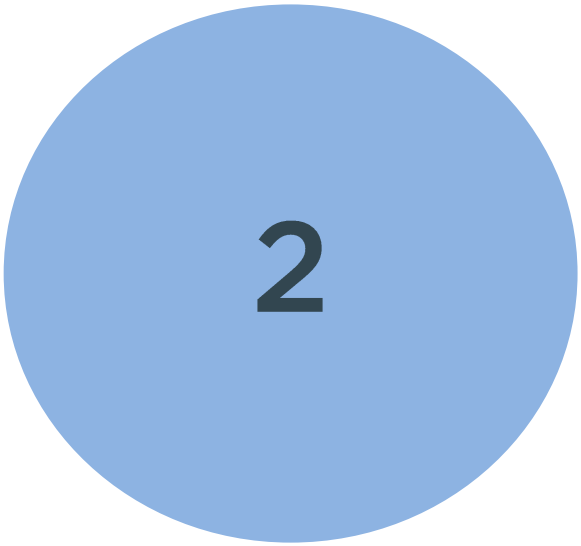
1. EXECUTIVE SUMMARY

1.1 TEST PARAMETERS

TEST REFERENCE	**BPEC210619-16**
TEST DATE	Original Test: 5th – 9th August 2019 Retest: 11th December 2019
TEST TIME	09:00 – 17:30
TEST TYPE	External Grey box
LIMITATIONS	No Denial of Service No Social Engineering
PERSONNEL	Chay Donohoe (QSTM) Kieran Roberts (re-test)

1.2 RESULTS OVERVIEW

RISK LEVEL	RISKS FOUND
Critical	0
High	0
Medium	0
Low	0
Recommendations	2
TOTAL	2



AREA	DESCRIPTION	RATING
Configuration	Overall security level of application and system configuration	GOOD
Patching	Overall patching level for framework	GOOD
Segmentation	Application user/content segmentation and segregation	GOOD

1.3 BUSINESS RISK SUMMARY

Bulletproof conducted an external penetration test of the SocialSignin application. This was carried out from both an unauthenticated and authenticated perspective and was conducted in-line with security best practices.

RETEST NOTES

During the retest all issues were found to be closed to an acceptable standard. The report has been updated to show the changes made within the environment between the original test and the retest.

For all identified vulnerabilities and recommendations for improvement, we have provided remediation advice along with the breakdown of each.

1.4 TESTING METHODOLOGY

This Bulletproof penetration test used the CREST framework as an overarching methodology, into which we embedded the PTES and OWASP practices. These are specifically tailored for infrastructure and application penetration tests. Any automated tools that were used during the assessment had all plugins and signatures up to date.

1.5 SCOPE

URL OR DESCRIPTION	IP ADDRESS
https://app.socialsignin.net	13.224.241.6
	13.224.241.105
	13.224.241.80
	13.224.241.23

Bulletproof tested the infrastructure against the following:

- Injections and input validation
- Authentication and session management
- Cross-site scripting (XSS)
- Insecure direct object references
- Security misconfiguration
- Sensitive data exposure
- Missing function-level access control
- Cross-site request forgeries
- Components with known vulnerabilities
- Un-validated redirects and forwards
- UI redress
- Cache control

2. PENETRATION TEST RESULTS

2.1 ENVIRONMENT OVERVIEW

The security assessment started by gathering all the needed information about the target to discover its running services, current patch levels, improper configurations and security controls, in order to identify associated vulnerabilities and attack vectors.

2.2 CRITICALITY INDEX

RISK LEVEL	DESCRIPTION	RECOMMENDATION
CRITICAL SCORE: 9-10	A critical-risk indicates serious and immediate risk to systems and data being compromised.	Critical-rated issues need to be addressed and resolved immediately.
HIGH SCORE: 7-9	High-risk indicates that a serious weakness or exposure exists.	High-rated issues need to be addressed and resolved immediately.
MEDIUM SCORE: 4-7	Medium-risk indicates that a significant issue needs to be addressed.	Actions need to be taken once high-risks have been addressed.
LOW SCORE: 1-4	Low-risk indicates minor issues that generally are harmless and can be used when profiling an organisation.	No immediate action is required but should be addressed through the remediation phase.
RECOMMENDATION SCORE: N/A	Recommendations are included for improvement purposes only as they pose an indirect risk to the current environment.	N/A

2.3 RISK RESULTS

DESCRIPTION	RISK RATING	AFFECTED HOSTS	REFERENCE
Host Poisoning	RECOMMENDATION	app.socialsignin.co.uk	**R-001**
Session Management	RECOMMENDATION	app.socialsignin.co.uk	**R-002**

2.4 WEB APPLICATIONS

Web application components were enumerated and checked for possible vulnerabilities. The following components were found to be in use:

app.socialsignin.net

Analytics

- Google Analytics

JavaScript Framework

- React v16.5.2
- Zone.js
- AngularJS

Font Script

- Google Font API

Miscellaneous

- webpack

Rich Text Editor

- TinyMCE v4

CDN

- Amazon Cloudfront

JavaScript Libraries

- Lodash v4.17.15

PaaS

- Amazon Web Services

2.4.1 INJECTION

2.4.1.1 CROSS-ORIGIN RESOURCE SHARING

REFERENCE	M-001
METHOD	Attempt to reflect request 'Origin' header within 'Access-Control-Allow-*' headers in the response.
AFFECTED HOSTS	app.socialsignin.co.uk
RESULT	It was possible to have the 'Origin' header value in the request reflected in the 'Access-Control-Allow-Origin' header in the response.
PASS/FAIL	(retest) PASS

EVIDENCE

The objective of the 'Access-Control-Allow-*' headers in the server response is to ensure that the client's browser will only request resources from the hosts listed in this header, and to exclude resources from anything else.

In the original test it was possible to change the 'Origin' header in the request, and have the server reflect this value in the response.

Remedial work was undertaken between the original test and the retest, and it was determined that the origin header was no longer reflected when edited during the retest. The issue has now been closed.

2.4.1.2 HOST POISONING

REFERENCE	R-001
METHOD	Verify that the web server correctly handles arbitrary host headers.
AFFECTED HOSTS	app.socialsignin.co.uk
RESULT	Server did not generate any errors in relation to the invalid host header.
PASS/FAIL	RECOMMENDATION

EVIDENCE

In many cases, developers are trusting the HTTP host header value and using it to generate links, import scripts, and even generate password reset links with its value. This is not advisable because the HTTP host header can be controlled by an attacker. This can be exploited using web-cache poisoning and by abusing alternative channels like password reset emails. Whilst the server did not return an error in response to the altered host header, this does not necessarily imply that the server isn't silently acting on this information.

```
GET /server/index HTTP/1.1
Host: app.evildoers.co.uk
Connection: close
Accept: application/json, text/plain, */*
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3848.0
Safari/537.36
Sec-Fetch-Mode: cors
Origin: https://app.socialsignin.net
Sec-Fetch-Site: cross-site
Referer: https://app.socialsignin.net/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
```

Request sent to server with host header set to 'app.evildoers.co.uk' (highlighted)

```
HTTP/1.1 200 OK
Server: Apache
Vary: Accept-Encoding
Cache-Control: no-cache
Content-Type: application/json
Access-Control-Expose-Headers: SSI-BackendServer, SSI-TraceId
Date: Tue, 06 Aug 2019 12:28:37 GMT
Keep-Alive: timeout=5, max=88
SSI-BackendServer: api-aseee0760c
Access-Control-Allow-Origin: https://app.socialsignin.net
SSI-TraceId: XUlydX8AAAEAADPFIhsAAAAZ
Connection: close
Content-Length: 21962

{"time":"2019-08-06T12:28:37+00:00","server_time_zone":"UTC","version":
```

Server response did not reflect altered host header in response. However, an expected error was not generated

SEVERITY	0.0 - RECOMMENDATION
LIKELIHOOD	N/A
EFFORT TO FIX	N/A
CVSS VECTOR	N/A
REMEDIATION	Ensure that the web server is configured to verify the host header, and to reject any requests that do not match with the configured value.

2.4.1.3 REFERRER POISONING

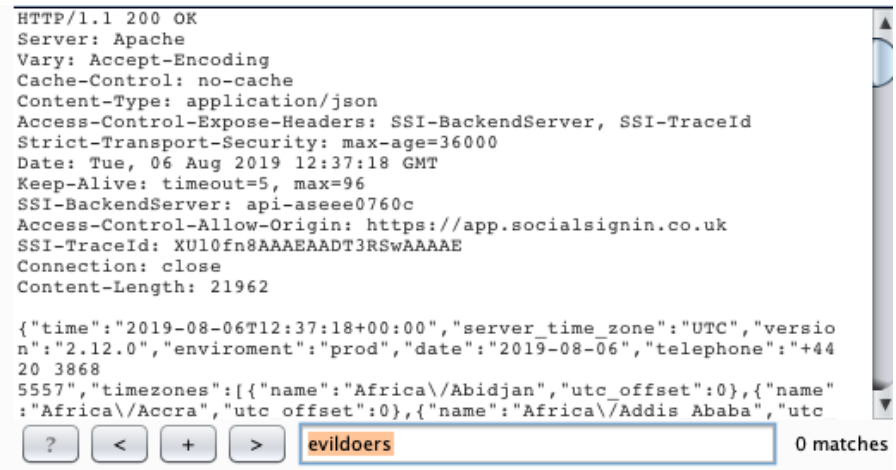
REFERENCE	P-001
METHOD	Verify that web servers correctly handle arbitrary referer headers.
TESTED HOST	• app.socialsignin.co.uk • app.socialsignin.net • orlo.tech
RESULT	Servers did not reflect any input from the altered referrer header.
PASS/FAIL	PASS

EVIDENCE

In a similar manner to which the host header can be used to pass values into the application, the same applies to the referer header. In this test, we substituted 'evildoers' into this header, then checked the response from the application to see whether it had used this as a variable when generating the resulting page:

```
GET /server/index HTTP/1.1
Host: app.socialsignin.co.uk
Connection: close
Accept: application/json, text/plain, */*
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3848.0
Safari/537.36
Sec-Fetch-Mode: cors
Origin: https://app.socialsignin.co.uk
Sec-Fetch-Site: cross-site
Referer: https://app.evildoers.net/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
```

Request sent to server with referer set to 'https://app.evildoers.net/' (highlighted)



Server response did not contain any evidence of having used the referrer string in the response (Note '0 matches' in the lower right)

2.4.1.4 SQL INJECTION

REFERENCE	M-002
METHOD	Attempt to exploit any possible SQL back-end servers by using SQL injection methods.
AFFECTED URIs	app.socialsignin.co.uk
RESULT	Whilst we were unable to obtain any information from the SQL back-end server(s), it did appear to be possible to execute SQL queries although time constraints prevented us from exploiting this further.
PASS/FAIL	(retest) PASS
EVIDENCE	
We noted an instance where SQL (specifically MySQL) appeared to be in use. In the case of the /outbox/search endpoint, SQL errors were originally returned in the response when we tried to inject invalid values into the JSON request, however remedial action was undertaken and no SQL errors were observed during the retest, and for this reason the issue has been closed.	

2.4.1.5 FILE UPLOAD HANDLING

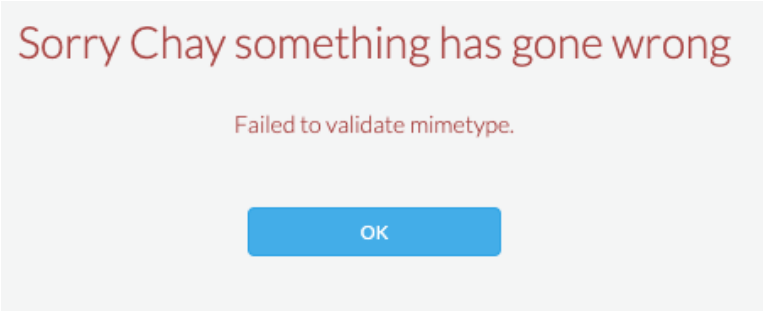
REFERENCE	P-002
METHOD	Verify that any file upload elements of the application cannot be exploited.
TESTED HOSTS	app.socialsignin.co.uk
RESULT	The application was found to verify file type and reject any that were found to not be parsable.
PASS/FAIL	PASS

EVIDENCE

The application has many instances where CSV files can be imported. These were tested to determine how the application handled parsing of imported data (i.e. would the application follow embedded links or attempt to execute embedded code etc).

Text (Do not delete this column)	Image URL (Do not delete this column)	Date (Do not
Example post 1	=cmd|'/C ping -t bp-dns.com -l 25152'!'A1'	2018-05-31 08
Example post 2	=hyperlink('https://www.bulletproof.co.uk'	'Bulletproof')

The application did not appear to execute anything during the parsing process and also returned an error when we attempted to import the above example:

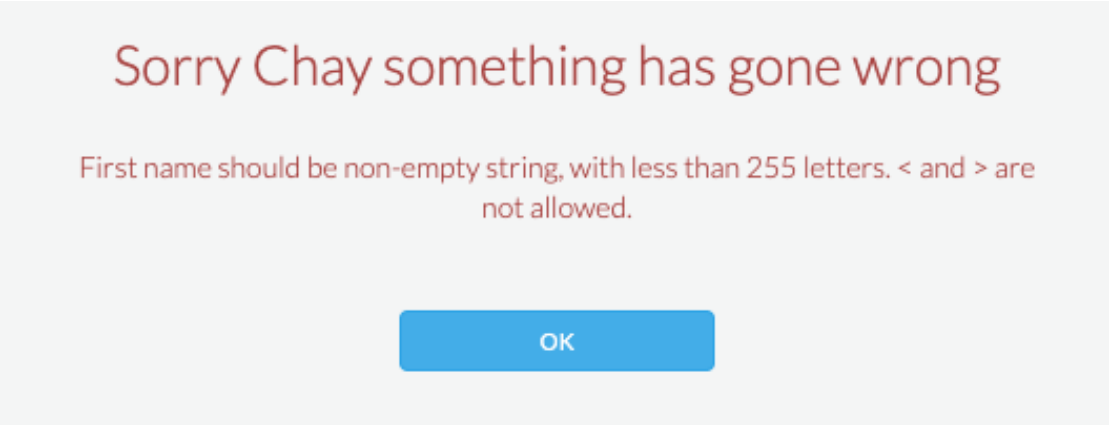


2.4.1.6 INPUT VALIDATION & CROSS-SITE SCRIPTING

REFERENCE	P-003
METHOD	Determine the efficacy of client and server-side input validation.
TESTED HOST	• app.socialsignin.co.uk • app.socialsignin.net
RESULT	The server aspect of the application generated errors when it detected any invalid user input.
PASS/FAIL	PASS

EVIDENCE

Injecting certain characters into user input fields, such as those found in the "probe fish" (‘"/><(((‘>), can be used to exploit XSS vulnerabilities. Accepting parameter and value pairs in input fields (e.g. `&param=value`) could permit additional parameters to be added to a request. We tested input validation using a variety of methods, such as entering test strings directly into user-input fields, and by modifying GET and POST requests. When sent illegal characters, we received this error:



Form fields with illegal characters and XSS elements

To confirm that this was generated by the server side, we directly sent a request through a transparent proxy and monitored the response:

```
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
{"first_name":"<img src="" onload=alert(1)>","last_name":"Donohoe
```

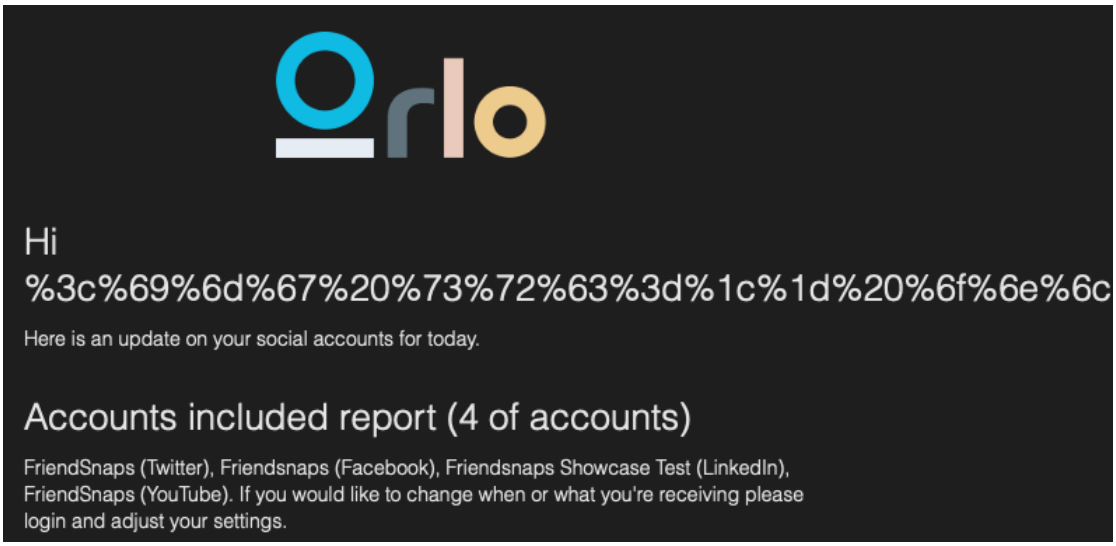
Request sent directly to server

```
{"error":1,"message":"First name should be non-empty string, with less than 255 letters. < and > are not allowed.","error_code":400}
```

Resulting server response contained expected error

We then attempted to bypass input validation using alternate methods such as URL encoding to hide the '<' and '>' characters from the filter:

The application appeared to accept the encoded input, but it did not decode internally back to the original characters, as seen in the daily summary email extract seen below:



We also tested that field contents were validated against the stored data types (e.g. numbers were checked before being processed as fixed bit-length types such as int32). The application did not appear to validate this data before sending it to the MySQL server as part of a query. This is detailed in section 2.5.1.4 above.

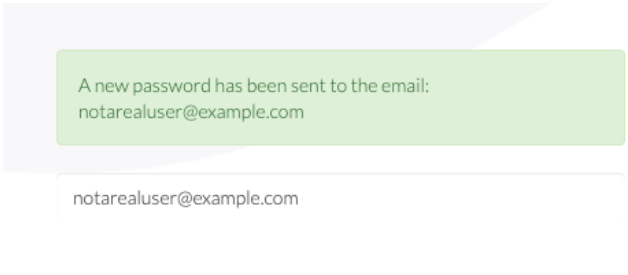
2.4.2 BROKEN AUTHENTICATION

2.4.2.1 USER ENUMERATION

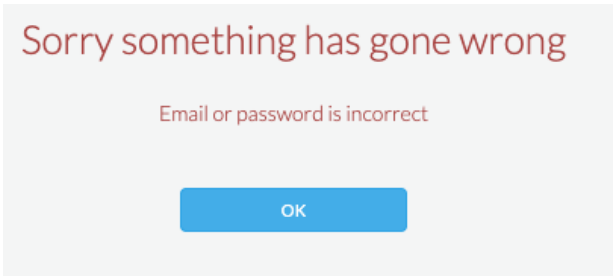
REFERENCE	P-004
METHOD	Multiple login attempts were made using valid versus invalid usernames and comparing responses between them to determine whether usernames could be enumerated. Password reset function was also tested for enumeration vulnerabilities.
TESTED HOSTS	app.socialsignin.net
RESULT	The server response was identical, regardless of username validity.
PASS/FAIL	PASS

EVIDENCE

As seen in the evidence screenshots below, the same response was given regardless of whether the supplied username was valid or invalid in both the password reset function (left) and the login (right). Due to this, an attacker would not be able to enumerate possible usernames used within the application:



Attempted user enumeration through password reset.



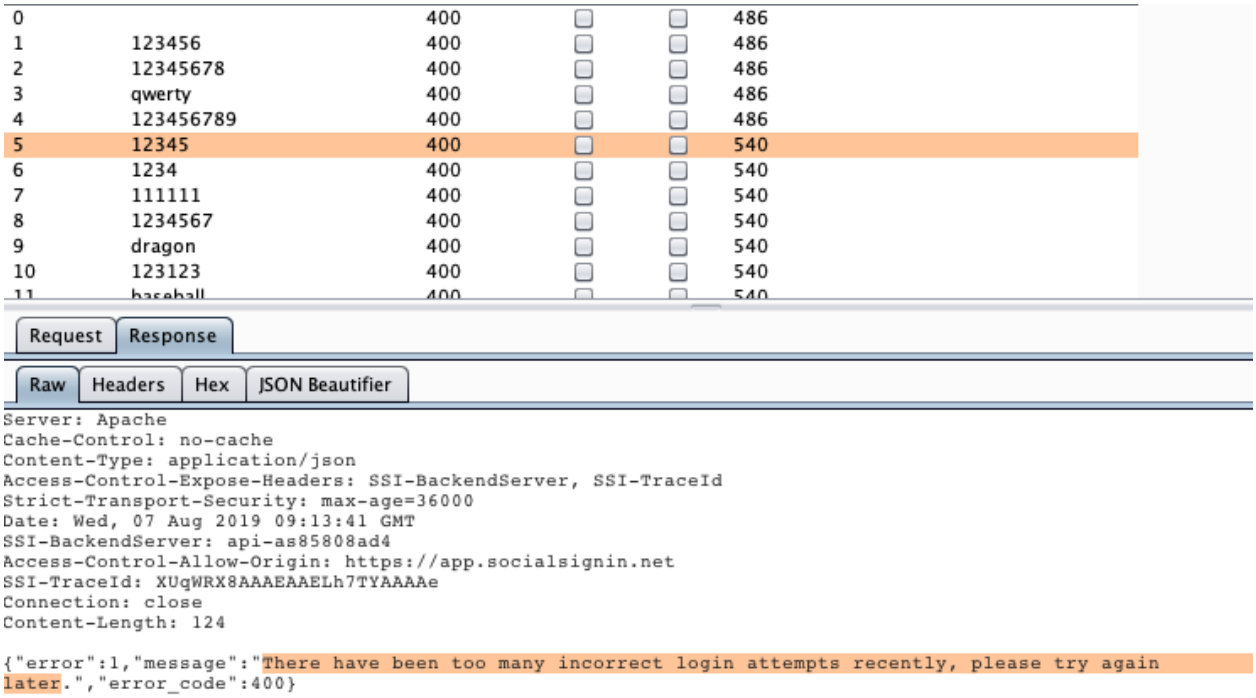
Ambiguous response to invalid login.

2.4.2.2 BRUTE-FORCE PROTECTION

REFERENCE	P-005
METHOD	Attempt to gain access to the application using brute-force methods via repeatedly submitting different passwords for a known username.
TESTED HOST	• app.socialsignin.net • app.socialsignin.co.uk
RESULT	The implemented lock-out mechanism blocked logins to the test account for between five and ten minutes.
PASS/FAIL	PASS

EVIDENCE

Initially, we captured an authentication request to the site using a proxy. This was then sent to an automated brute-forcing tool which used approximately 1,000 commonly-used passwords.
After six failed attempts, the application blocked further login attempts until we re-tried around ten minutes later:



Response to login attempt after lockout.

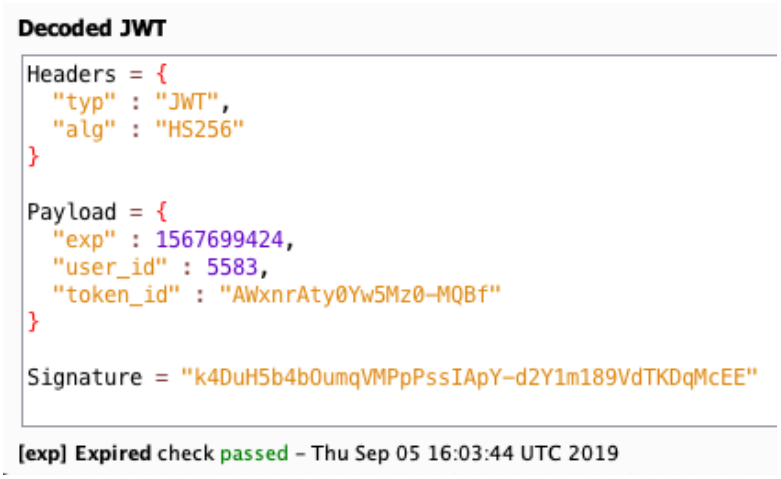
This action would limit the effectiveness of a brute-force attack, as only so many passwords could be attempted within a certain time. This is likely to discourage an attacker from continuing any further.

2.4.2.3 SESSION MANAGEMENT

REFERENCE	R-002
METHOD	Determine whether session tokens are handled correctly (e.g. logged-out tokens are revoked).
AFFECTED HOSTS	• app.socialsignin.net • app.socialsignin.co.uk
RESULT	Logged-out tokens could not be re-used, and changing the JWT signature type to 'None' caused an invalid token error. However, tokens had a validity period of one month.
PASS/FAIL	RECOMMENDATION

EVIDENCE

Initially we captured and examined session tokens to determine their type. These were JSON Web Tokens and had a validity period of one month:



JWT with highlighted Expiry ('exp') timestamp (decoded value at bottom)

In order to test session token handling, we started a new session by logging in, and capturing a request made to the API before logging off. We waited about 30 seconds to allow the logout request (if any) to propagate through any back-end systems before replaying the request to determine whether the API had revoked the token:

```
GET /monitoring/index HTTP/1.1
Host: app.socialsignin.co.uk
Connection: close
Accept: application/json, text/plain, */*
X-Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE1Njc3NjE2NjQsInVzZXJfaWQiOjU1ODMsInRva2VuX2lkIjoiQVd4cl1jRko4Z0o3NWQwNS13NEMifQ.2K
x-OpTbFzQ_lJUfnj0MbRA3JkoTQ6QVmyrNSgx3wk4
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3848.0
Safari/537.36
Sec-Fetch-Mode: cors
Origin: https://app.socialsignin.net
Sec-Fetch-Site: cross-site
Referer: https://app.socialsignin.net/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
```

Initial captured request sent to API

```
HTTP/1.1 200 OK
Server: Apache
Vary: Accept-Encoding
Cache-Control: no-cache
Content-Type: application/json
Access-Control-Expose-Headers: SSI-BackendServer, SSI-TraceId
Strict-Transport-Security: max-age=36000
Date: Wed, 07 Aug 2019 10:05:47 GMT
Keep-Alive: timeout=5, max=99
SSI-BackendServer: api-aseee0760c
Access-Control-Allow-Origin: https://app.socialsignin.net
SSI-TraceId: XUqie38AAAEAAE0XFMwAAAAW
Connection: close
Content-Length: 13133

[{"geo":{"enabled":false,"radius":25},"query":{"$or":[{"type":"text","comparison":"has","
"twitter"],"include-retweets":true,"name":"#OrloMasterclass","id":"4d35cbef31e23a4bd8a5b5
13:46:50+00:00"},{"geo":{"enabled":false,"radius":25},"query":{"$and":[{"type":"text","co
in"},{"$or":[{"type":"text","comparison":"has","value":"#socialsigninapp"},{"type":"text"
n"},{"type":"text","comparison":"has","value":"SocialSignIn"},{"type":"text","comparison"
In"},{"type":"text","comparison":"has","value":"orlo"},{"type":"text","comparison":"has",
app"},{"type":"text","comparison":"has","value":"orlo cx"},{"type":"text","comparison":"h
```

API response to above request (logged-in session)

```
HTTP/1.1 401 Unauthorized
Server: Apache
Cache-Control: no-cache
Content-Type: application/json
Access-Control-Expose-Headers: SSI-BackendServer, SSI-TraceId
Strict-Transport-Security: max-age=36000
Date: Wed, 07 Aug 2019 10:06:25 GMT
Keep-Alive: timeout=5, max=96
SSI-BackendServer: api-as36117e14
Access-Control-Allow-Origin: https://app.socialsignin.net
SSI-TraceId: XUqioX8AAAEAACE8S5IAAAAN
Connection: close
Content-Length: 84

{"error":1,"message":"Request failed authentication (Invalid JWT)","error_code":401}
```

API response to second replayed request (logged-out session)

As can be seen above, the API responded that the session token had been revoked. It would therefore not be possible for an attacker to use logged-out tokens.

We did note, however, that the token expiry time of one month was well in excess of what it should be. We would recommend a value somewhere between ten minutes and twelve hours, depending on the application.

SEVERITY	4.2 - MEDIUM
LIKELIHOOD	LOW
EFFORT TO FIX	MEDIUM
CVSS VECTOR	CVSS:3.0/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:N
REMEDIATION	Ensure session tokens are not valid for more than twelve hours, in order to reduce the chance of live session cookie capture and reuse, should an attacker gain control of a victim's machine whilst logged into the application.

2.4.3 BROKEN ACCESS CONTROL

REFERENCE	P-006
METHOD	Assess whether it is possible for one user to view another user's data and/or whether the user is able to perform privileged actions.
TESTED HOSTS	• app.socialsignin.co.uk • app.socialsignin.net
RESULT	It was not possible for one user to read or write data belonging to another user.
PASS/FAIL	PASS

EVIDENCE

User segmentation checks were made in order to ensure that one logged-in user was not able to access data belonging to a different user. To test this, two separate sessions were logged in, one for an admin-level user and one for a low-privilege user. Cookies for the low-privilege user were sent to an automated tool which replays any requests we make, replacing the cookies from the admin user. The testing software also attempts the same operations with no authentication tokens in order to determine whether an unauthenticated user can also access this data. The application performed correctly and presented an error in the case of using a UID that didn't match the data that was being retrieved (in this example we're trying to view authentication providers):

```
GET /account/authProviders HTTP/1.1
Host: app.socialsignin.co.uk
Connection: close
Accept: application/json, text/plain, */*
X-Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE1Njc4NDg3MDQsInVzZXJfaWQiOjU1NTcsInRva2VuX2lkIjoiQVd4d2tlRGxmVmVwYzVQaUlpVmIifQ
.0aikeCygRcYjKGogCevwDsx3zYIO60KInSlvY3QK32g
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3848.0
Safari/537.36
Sec-Fetch-Mode: cors
Origin: https://app.socialsignin.net
Sec-Fetch-Site: cross-site
Referer: https://app.socialsignin.net/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
```

Original Request (as admin user – JWT highlighted)

```
HTTP/1.1 200 OK
Server: Apache
Vary: Accept-Encoding
Cache-Control: no-cache
Content-Type: application/json
Access-Control-Expose-Headers: SSI-BackendServer, SSI-TraceId
Strict-Transport-Security: max-age=36000
Date: Thu, 08 Aug 2019 09:34:14 GMT
Keep-Alive: timeout=5, max=100
SSI-BackendServer: api-as36117e14
Access-Control-Allow-Origin: https://app.socialsignin.net
SSI-TraceId: XUvsln8AAAEAAEZGc9QAAAAa
Connection: close
Content-Length: 235

[{"id":3,"name":"Facebook"},{"id":9,"name":"Facebook Ads"},{"id":7,"name":"Instagram"},{"id":12,"name":"Instagram
Business"},{"id":4,"name":"LinkedIn"},{"id":8,"name":"LinkedIn Ads"},{"id":2,"name":"Twitter"},{"id":6,"name":"YouTube"}]
```

Response to original request. (200 OK status, and JSON result highlighted

```
HTTP/1.1 400 Bad Request
Server: Apache
Cache-Control: no-cache
Content-Type: application/json
Access-Control-Expose-Headers: SSI-BackendServer, SSI-TraceId
Strict-Transport-Security: max-age=36000
Date: Thu, 08 Aug 2019 09:34:14 GMT
SSI-BackendServer: api-as36117e14
Access-Control-Allow-Origin: https://app.socialsignin.net
SSI-TraceId: XUvsln8AAAEAAEWU8e0AAAAK
Connection: close
Content-Length: 91

{"error":1,"message":"You do not have permission to administer accounts.","error_code":400}
```

Response to request made by low privilege user (400 Bad Request status and JSON error highlighted)

```
HTTP/1.1 401 Unauthorized
Server: Apache
Cache-Control: no-cache
Content-Type: application/json
Access-Control-Expose-Headers: SSI-BackendServer, SSI-TraceId
Strict-Transport-Security: max-age=36000
Date: Thu, 08 Aug 2019 09:34:14 GMT
Keep-Alive: timeout=5, max=99
SSI-BackendServer: api-as36117e14
Access-Control-Allow-Origin: https://app.socialsignin.net
SSI-TraceId: XUvsln8AAAEAAEdGlSMAAAAm
Connection: close
Content-Length: 88

{"error":1,"message":"Request failed authentication (No JWT provided)","error_code":401}
```

Response from API when no JWT tokens are sent in request

2.4.4 SECURITY MISCONFIGURATION

2.4.4.1 AUTHORISATION METHOD OVERRIDE

REFERENCE	P-007
METHOD	Verify that the application does not accept HTTP basic authorisation as a possible means to bypass the extant mechanism.
TESTED HOST	• orlo.tech • app.socialsignin.net
RESULT	We were unable to use this authentication mechanism to bypass session tokens.
PASS/FAIL	PASS

EVIDENCE

One of the test sets of credentials was base64 encoded in the format (username:password). All cookies used by the application were replaced with Authorization: Basic *<base64 string>*:

```
POST /outbox/search HTTP/1.1
Host: app.socialsignin.co.uk
Connection: close
Content-Length: 279
Accept: application/json, text/plain, */*
Authorization: basic Y2hheS5kb25vaG9lQGJlbGxldHByb29mLmNvLnVrOlBhc3N3b3JkMQo=
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3848.0
Safari/537.36
Sec-Fetch-Mode: cors
Content-Type: application/json;charset=UTF-8
Origin: https://app.socialsignin.net
Sec-Fetch-Site: cross-site
Referer: https://app.socialsignin.net/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8

{"accounts":["18"],"start_date":"2019-08-08T23:00:00.000Z","types":["status_update","picture","album","video","multi_image"],"dele
ted":"exclude","scheduled":"include","sort_dir":"desc","offset":0,"limit":50,"with":["boosted_post_counts","note_counts"],"v":2,"t
z_offset":-3600000}
```

Modified request with replaced header sent to API (highlighted)

```
HTTP/1.1 401 Unauthorized
Server: Apache
Cache-Control: no-cache
Content-Type: application/json
Access-Control-Expose-Headers: SSI-BackendServer, SSI-TraceId
Strict-Transport-Security: max-age=36000
Date: Fri, 09 Aug 2019 15:01:49 GMT
Keep-Alive: timeout=5, max=100
SSI-BackendServer: api-as2bad23ec
Access-Control-Allow-Origin: https://app.socialsignin.net
SSI-TraceId: XU2K3X8AAAEAAEhB5dcAAAAC
Connection: close
Content-Length: 88

{"error":1,"message":"Request failed authentication (No JWT provided)","error_code":401}
```

Resulting '401 Unauthorized' response from server

Having received a '401 Unauthorized' message, we determined that we would not be able to use HTTP basic authorisation as an alternate means to brute-force login attempts.

2.4.4.2 HTTP STRICT-TRANSPORT-SECURITY HEADER

REFERENCE	P-008
METHOD	Verify whether the server response contains a Strict-Transport-Security (HSTS) header.
TESTED HOST	• orlo.tech • app.socialsignin.net • app.socialsignin.co.uk
RESULT	Tested hosts returned a correctly formatted HSTS header.
PASS/FAIL	PASS

EVIDENCE

The Strict-Transport-Security header ensures that once a supported browser receives this header, it will prevent any communications from being sent over HTTP to the specified domain. Instead it will send all communications over HTTPS. It also prevents HTTPS click-through prompts on browsers. In this case, the tested host returned a valid header as highlighted below:

```
HTTP/1.1 304 Not Modified
Connection: close
Date: Fri, 09 Aug 2019 15:09:44 GMT
Cache-Control: public, must-revalidate, proxy-revalidate, max-age=0
Last-Modified: Fri, 09 Aug 2019 10:52:29 GMT
ETag: "e889ce3269bc18afa33e422d020f0a01"
Server: OrloCDN
Strict-Transport-Security: max-age=604800; preload
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Content-Security-Policy-Report-Only: default-src 'unsafe-eval' 'unsafe-inline'
Content-Security-Policy: upgrade-insecure-requests
X-Cache: RefreshHit from cloudfront
Via: 1.1 8d36edc1ce736c158ddedbd7365e2a8e.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: LHR3-C1
X-Amz-Cf-Id: KvEnNGFsVOc8tHnAKqIO5KhysY5hZMbJYzFtc5awtE5XDV_tsqMFNQ==
```

2.4.4.3 SERVER SIGNATURE

REFERENCE	P-009
METHOD	Examine HTTP and HTTPS response headers to determine whether they contain a server signature that would disclose version information.
TESTED HOSTS	• orlo.tech • app.socialsignin.net • app.socialsignin.co.uk
RESULT	The tested hosts returned a response header indicating only HTTP server type, but no version information.
PASS/FAIL	PASS

EVIDENCE

Whilst not strictly a security issue in itself, disclosing component version information could give potential attackers information about what's running the application. Should any vulnerabilities that affect that version come to light in the future, the application could immediately become a target.

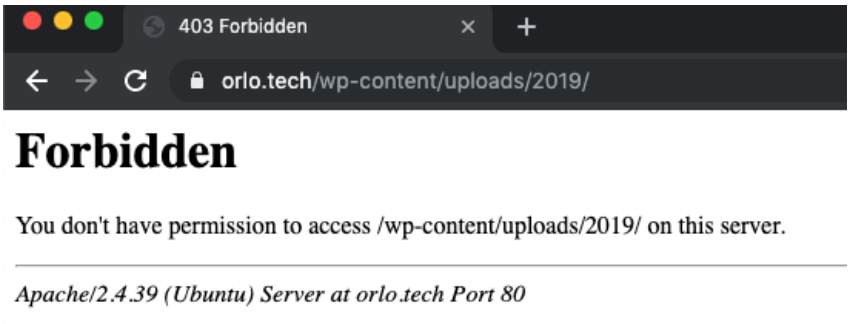
In the tested cases, the server signatures were either 'cloudflare', 'Apache' or 'OrloCDN', but no version information was disclosed. It should be noted, however, that Apache version information for orlo.tech was leaked in the '403 Forbidden' page as seen in section 2.5.3.

2.4.4.4 DIRECTORY BROWSING

REFERENCE	P-010
METHOD	Verify whether web servers allow directory browsing.
TESTED HOSTS	• app.socialsignin.co.uk • app.socialsignin.net • orlo.tech
RESULT	Hosts were found to prevent arbitrary directory browsing.
PASS/FAIL	PASS

EVIDENCE

Using a sample of directories collected as part of a dictionary directory scan (see section 2.4.1), we attempted to see if we could browse arbitrary files. Upon accessing these, the orlo.tech server responded with a 'Forbidden' error as can be seen below:



In the case of app.socialsignin.net, this redirected any directory browse attempts back to the login or dashboard page. app.socialsignin.co.uk API interface had no apparent resources, and just returned an API error.

2.4.4.5 X-FRAME-OPTIONS & CLICKJACKING

REFERENCE	P-011
METHOD	Verify that the X-Frame-Options header exists and is set to correctly prevent Clickjacking (UI Redress).
TESTED HOSTS	• app.socialsignin.net • orlo.tech
RESULT	Server response included valid X-Frame-Options header, and the webpage would not render within an iFrame element.
PASS/FAIL	PASS

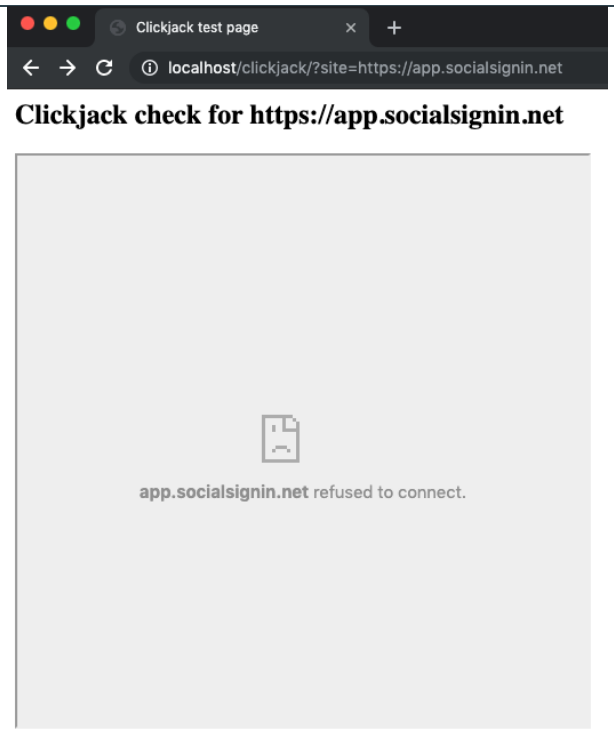
EVIDENCE

The response headers from a page GET were checked for the X-Frame-Options header, which instructs the client browser to only obtain elements from a particular source, preventing malicious parties from opening the page within an iFrame element and intercepting events:

```
HTTP/1.1 304 Not Modified
Connection: close
Date: Mon, 12 Aug 2019 16:47:54 GMT
Cache-Control: public, must-revalidate, proxy-revalidate, max-age=0
Last-Modified: Fri, 09 Aug 2019 10:52:29 GMT
ETag: "e889ce3269bc18afa33e422d020f0a01"
Server: OrloCDN
Strict-Transport-Security: max-age=604800; preload
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Content-Security-Policy-Report-Only: default-src 'unsafe-eval' 'unsafe-in
Content-Security-Policy: upgrade-insecure-requests
X-Cache: RefreshHit from cloudfront
Via: 1.1 5d21561f8325da91dd79188f8c919b09.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: LHR3-C2
X-Amz-Cf-Id: UELvshr1Wwepe9ggXVPupPiTNSCI2dCvK6qtXfmwloHsOPm3Fxh4kw==
```

Example X-Frame-Options header for app.socialsignin.net (highlighted)

Next, we ran a script from our machine which attempts to render the target site within an iFrame. As can be seen in the sample below, it was not possible to render either target site. (This can also be reproduced with the HTML code in section 4.1):



Successful clickjack check for https://app.socialsignin.net

2.4.4.6 CONTENT-SECURITY-POLICY HEADER

REFERENCE	P-012
METHOD	Check for existence and correct configuration of Content-Security-Policy header.
TESTED HOSTS	• app.socialsignin.co.uk • app.socialsignin.net • orlo.tech
RESULT	Host responses included valid CSP headers.
PASS/FAIL	PASS

EVIDENCE

A correctly configured CSP header will ensure a browser will only retrieve dynamic resources from sources approved by the application authors, therefore minimising cross-site scripting risks. A lack of this may leave the server open to XSS exploitation should any unvalidated or un-sanitised user input-fields exist within the application.

In tested instances, all hosts returned valid CSP headers:

```
HTTP/1.1 304 Not Modified
Connection: close
Date: Fri, 09 Aug 2019 15:09:44 GMT
Cache-Control: public, must-revalidate, proxy-revalidate, max-age=0
Last-Modified: Fri, 09 Aug 2019 10:52:29 GMT
ETag: "e889ce3269bc18afa33e422d020f0a01"
Server: OrloCDN
Strict-Transport-Security: max-age=604800; preload
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Content-Security-Policy-Report-Only: default-src 'unsafe-eval' 'unsafe-inline' blob: data: https: ws: wss:; img-src 'unsafe-inline' blob: data: *; report-uri https://csp.test.orlo.tech/report;
Content-Security-Policy: upgrade-insecure-requests
X-Cache: RefreshHit from cloudfront
Via: 1.1 8d36edc1ce736c158ddedbd7365e2a8e.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: LHR3-C1
X-Amz-Cf-Id: KvEnNGFsVOc8tHnAKqIO5KhysY5hZMbJYzFtc5awtE5XDV_tsqMFNQ==
```

2.4.4.7 CACHE CONTROL

REFERENCE	P-013
METHOD	Verify that the cache-control header prevents public storage of sensitive data and that upstream caches validate cached copies with the server before serving a cached copy.
TESTED HOSTS	• app.socialsignin.co.uk • app.socialsignin.net • orlo.tech
RESULT	The server correctly set a cache-control header to recommended values.
PASS/FAIL	PASS

EVIDENCE

For optimum security, having the Cache-Control header set to '`Cache-Control: no-cache, no-store`' will prevent caching by downstream devices and the client itself. Unfortunately, this isn't ideal where performance is concerned on a busy platform, so a trade-off of security versus speed may be required. For legacy browsers that do not support the Cache-Control header, it is suggested to include a '`Pragma: no-cache`' header.

In this case, the server correctly set the cache-control headers as can be seen below:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 38
Connection: close
Last-Modified: Thu, 01 Aug 2019 12:56:09 GMT
Server: OrloCDN
Strict-Transport-Security: max-age=604800; preload
Date: Mon, 05 Aug 2019 23:21:06 GMT
Cache-Control: public, must-revalidate, proxy-revalidate, max-age=0
ETag: "e269b377a4f2e69abd3861a9512bc376"
X-Cache: RefreshHit from cloudfront
Via: 1.1 9a5c4712d591c80fa6eb5cd925d9b817.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: LHR3-C1
X-Amz-Cf-Id: Ux7WyED3CQeAixxR8VHpLMTyDc47w4-kLBK7HTfJjR6Nkl2HU-uSYg==

{"latestVersion":"2019-08-01-7150bff"}
```

Cache related header in server response

3. NEXT STEPS

We strongly recommend investigating and remediating all issues listed below, which are listed in order of priority with a justification for each vulnerability. We recommend arranging a debrief phone call to discuss the points raised in this report, including any questions you may have.

PRIORITY	REFERENCE	DESCRIPTION	PRIORITY JUSTIFICATION
1	R-002	Session Management	Where possible, shorten validity period of session tokens.
2	R-001	Host Poisoning	These are relatively simple server configuration changes.

4. APPENDIX

4.1 HTML CODE FOR CLICKJACK TESTING

```
<html>
    <head>
        <title>Clickjack test page</title>
    </head>
    <body>
        <p>Website is vulnerable to clickjacking!</p>
        <iframe src="http://www.target.site" width="500" height="500"></iframe>
    </body>
</html>
```

T: 01438 532 900 E: contact@bulletproof.co.uk W: www.bulletproof.co.uk