jscrambler

Securing Content and
Intellectual Property in

OTT Media Delivery

A WHITE PAPER BY **JSCRAMBLER**

+1 650 999 0010 hello@jscrambler.com jscrambler.com

Table of Contents

Executive Summary

With users wanting to access content when and where they want it, content providers have been making their content available on as many devices as possible. This includes web browsers, with web players increasing their market share fast in comparison to traditional IPTV systems.

While IPTV has been around for many years and several of its flaws have been fixed, OTT is a more recent technology. Innovation in this space is likely to push OTT technology as the standard for media and entertainment. OTT services are expected to represent a $332.5 B industry by 2025. As we see providers developing their web applications for OTT content, it is crucial to assess the security of these platforms.

A key business threat to OTT providers is piracy, which is expected to cost $12.5 B by 2024. To reduce this risk in the current threat landscape, providers must go beyond server-side security solutions and actively employ client-side security mechanisms. This means protecting the JavaScript code that handles all the logic of the player and of the watermarking solutions while using a Webpage Monitoring solution to detect client-side code injections and prevent tampering in real-time.

Jscrambler provides cutting-edge solutions to address both these dimensions, enabling OTT content providers to reduce their exposure to piracy attacks, mitigate data exfiltration attempts, and prevent any tampering and reverse engineering of the source code.

Introduction

The increase of high bandwidth Internet access around the world has created the opportunity for over-the-top businesses to grow their user base worldwide, including new markets in developing countries. In 2017, OTT services cumulatively generated revenue of around $97.5B and some estimations estimate growth to $332.5B by 2025[1].

A key threat to revenue and business sustainability in OTT is piracy. Even considering the security solutions commonly employed in OTT services, attackers can bypass security measures (*e.g.* watermarking techniques), re-distribute content, remove ads, or generate fraudulent clicks.

The exposure of premium content means the loss of potential revenue and a breach of compliance with content rights owners. These owners trust OTT stream providers to ensure that their copyrighted content is kept secure. Exposed content might lead to legal charges against the attacked OTT provider. These and other types of attacks cost pay-TV and OTT providers $9.1B in 2019 and it is expected that this cost reaches $12.5B by 2024.

[1] https://www.alliedmarketresearch.com/over-the-top-services-market

Web-based data breaches result in fines that top $230 million.

Clients' data has also become a very valuable asset and a common target of cyberattacks. The OTT provider might unexpectedly leak user data such as credit card info, user credentials, or personally identifiable information (PII). Since 2018, a growing wave of attacks dubbed Magecart has breached major companies to steal over half a million credit cards. Magecart-like approaches can be used to silently leak user data without attackers ever touching the server and remain undetected for months. Attacks to web applications can also trick the user into — unknowingly — providing sensible data to the attacker. OTT providers are susceptible to these attacks and such data breaches have resulted in $230M fines.

The security of Web applications versions from OTT media services is imperative — not only for safeguarding the company's revenue from its copyrighted content, but also to increase compliance with data privacy regulations to protect its user's data.

In this white paper, we discuss some of the most important security weaknesses of Web OTT applications and the existing techniques that prevent or mitigate attacks.

Weaknesses of Web OTT Applications

Before the introduction of the HTML5 standard on the web in 2014, the viewing of multimedia contents and the streaming of audio and video were usually done with the help of a browser plugin. The most commonly used one was Adobe's Flash Player which was an innovation in 1996 when it was released. However, it also had several security issues, required users to keep both browser and plugin updated to have protection against the most recently found flaws, and lacked the performance required for some complex Web Applications.

For these reasons, Adobe itself supported developers moving away from creating Flash-based applications and migrating to the current web standards. As a result, Adobe will end support for the Flash Player by the end of 2020.

Using new technologies in Web OTT requires due diligence on security implications.

The HTML5 standard, new Web APIs (like EME[2]), and JavaScript have been the power tools behind modern OTT Services. Providers leveraged these technologies for a faster and more reliable way of delivering online streaming content as a replacement for Adobe's Flash. However, blindly using these tools without due diligence on new web security implications jeopardizes the content that is being transmitted to the user, as well as the user itself.

When developing a Web Application for an OTT Service, there are some key security concerns that need to be addressed early-on:

1. How is the media content being protected from theft while on transmission?
2. How is the media content being protected from theft when it reaches the client-side?
3. How to ensure that only a valid client can access the media content?
4. How to keep important logic and intellectual property secure?
5. How to minimize exposure to attacks that seek to exfiltrate user data?

[2] https://developer.mozilla.org/en-US/docs/Web/API/Encrypted_Media_Extensions_API

Security in OTT

Traditionally, when trying to increase the integrity and security of web content, one of the first approaches to consider is HTTPS. While this does confirm that the user is connecting to the expected server, as well as ensure that the content is encrypted while it is being transmitted, it isn't the right tool to protect OTT content itself.

OTT content providers rely on Content Delivery Networks (CDN) to store content closer to the end-user with the goal of reducing latency and improving the user experience. Since CDNs don't usually belong to the content providers, their own SSL/TLS certificates used for establishing HTTPS connections can not be used. As so, the content is delivered in plain HTTP to end-users or through HTTPS but using the CDN's SSL/TLS certificate. But even if the content was delivered through HTTPS, it would only be protected during the transmission, with no protection on the client-side to prevent attackers from hijacking such content.

There are different approaches to solve the problems mentioned above and they can be used to mitigate different attack vectors that a malicious entity might use to illegally obtain the content provided by an OTT service. These approaches can be based on authentication tokens (to prevent content sharing), DRM systems (to protect content access), watermarking techniques, JavaScript/HTML5 protection, and webpage monitoring (to protect and monitor tamper attempts against DRM and Watermarking logic).

Tokens

In this section, we will describe two ways of using tokens in a Web OTT environment as protection from some possible attacks to the applications themselves but not necessarily against the OTT content.

The first approach is to have the server generate a random token for the authenticated client that will then be saved as a hidden same-site cookie. Then, for every request, the client has to send the cookie alongside it. If the weblink to the content was illegitimately shared with other people or used in another player, it will not work, as it lacks the corresponding valid token.

The other approach is to attach a token ID to the URL of the content. This token is usually a hash based on the content that is being watched, the client IP, and some random value generated by the server that only the server can validate. This allows the server to quickly detect if a request is coming from a legitimate client or not.

While DRM directly protects the OTT content and watermarking allows content owners to identify the origin of leaked content (as we'll explore next), tokens tackle the dangers of key/content sharing by a valid client. It can also be used for the server to quickly identify if a user is trying to access content they do not have access to by tampering with a link ID.

Digital Rights Management (DRM)

DRM is a digital licensing system that allows content owners to define how and by whom their content can be accessed. This system has been applied to different industries — most notably video games, music, movies or e-books. It has even expanded to hardware-based products like coffee makers or tractors. Companies use different approaches to apply the system, but the goal remains the same: protecting their own Intellectual property from being distributed through unauthorized means (mostly known as piracy).

There are three major DRM systems on the market today:

- **Microsoft PlayReady** — mostly used by Microsoft products;
- **Apple FairPlay** — specially designed for Apple's HLS and Apple devices and applications;
- **Google Widevine** — bought in 2010 by Google, it is now part of the Google ecosystem, including the Chrome browser and Android platforms.

OS	Browser	DRM
Windows 8.1 or later	IE11,Edge12+	PlayReady
Windows 7 or later Mac Os 10.10 or later	Chrome 35+ Firefox 47+	Widevine Modular
Android 4.4 or Later	Chrome 57+	
Mac Os 10.10 or later	Safari 8+	FairPlay Streaming
iOS 11.2 or later	Safari 11.2+	

Table 1 — DRM System Compatibility

In the context of Web, DRM can be divided in three steps: encryption, licensing and decryption, which can be observed in Fig. 1 below.

Firstly, after the content has been properly encoded, an encryption key is requested from the DRM system and the content is encrypted with a symmetrical algorithm, traditionally AES. In some cases, the key can be generated outside and then added to the DRM system. For every key, there is an associated content ID stored in the licensing server.

Whenever the user wants to access the content, it makes a request to the server. If the user has permission to do so, it receives the encrypted content stored in the Server/CDN. With the help of the EME API, the client requests the decryption key from the licensing server in order to access the original content. The licensing server can be configured with different policies that define how the content can be used. For example, it can enable offline playback, geolocation access or how long can the client cache the decryption keys.

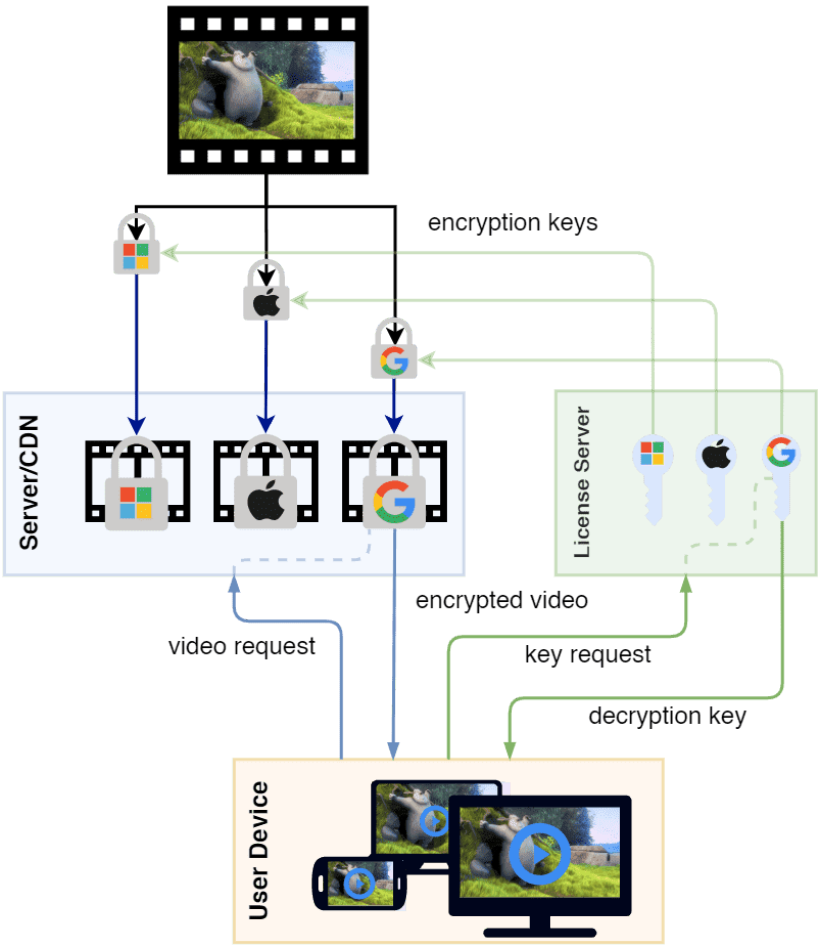


Fig. 1 — Flow example of a DRM copyright protection system[3].

There are two approaches for storing content on CDNs. One option is each DRM system having its own content encryption methodology - as such, the OTT provider might be required to store different copies of the same file at the CDNs depending on the target devices it intends to support.

The other option is to use the MPEG Common Encryption (MPEG-CENC) standard. This is supported by all the major DRM systems and only requires the encryption of

[3] Snapshot taken from the short film "Big Buck Bunny", available at http://bbb3d.renderfarming.net/

the file once. Providers are still required to have the different Licensing Servers since each one is designed in a different way and, as such, provide different features. The standards for this technology are ISO/IEC 23001-7:2016[4] and ISO/IEC 23001-9:2016[5].

Encrypted Media Extension API(EME)[6]

The media streaming market is truly massive in size and the potential business losses at stake resulting from cyberattacks pose a key threat. As so, it is imperative to ensure that the DRM process is as secure as possible from the perspective of content providers, streaming services and browser vendors. This need pushed the Encrypted Media Extension API to be designed and introduced in browsers.

This API allows the application to securely interact with the licensing server, obtain the decryption key associated with the content, and deliver it to the Content Decryption Module. This module is a closed-source part of the browser and, as such, it's executed in a sandboxed environment. It is also protected with anti-debugging and anti-tampering techniques so that the information about the key being used will hardly ever be leaked to the outside.

[4] https://www.iso.org/standard/68042.html
[5] https://www.iso.org/standard/70175.html
[6] https://developer.mozilla.org/en-US/docs/Web/API/Encrypted_Media_Extensions_API

DRM May Not Be Enough

DRM is ideal to protect content from being accessed by unauthorized personnel. However, after DRM has verified and decrypted the content, there is no further protection between the protected player and the display where content is viewed.

After someone gains access to the content — either legitimately or with stolen credentials — it is very difficult to stop this person from copying the content (e.g. by directly recording from the video/sound card) and illegally broadcast it. When that happens, watermarking techniques have to be used to detect pirated content and to find the culprit.

Watermarking

Watermarking is a common process for embedding metadata inside a target digital content. This usually includes information related to the authenticity and integrity of the digital content and might include other relevant metadata, such as copyright ownership, user identification, among others.

While DRM has protection capabilities against content hijacking, watermarking does not prevent the content from being stolen. Instead, it can track and find unauthorized access and copyright infringement (*e.g.* when DRM is breached).

The watermarking is embedded when a video instance is requested and can include one of two types of marks: Distributor Mark and Subscriber Mark. Distributor Mark is

typically applied from content owners (e.g. Hollywood) before sending it to movie theaters as a way to quickly identify possible leaks. Subscriber — or Session — Mark is applied by content providers (e.g. Netflix) and has medatada of the current instance, the timestamp and the recipient (such as user ID, device ID and IP).

From the existing types of watermarking techniques, for OTT we highlight visual watermarking and forensic watermarking. Visual watermarks are embedded at the source and are perceptible to the user, as seen in Fig. 2. This approach is commonly used to add the copyright logo of the content owner or streaming company, namely as part of a branding strategy.

Fig. 2 — Example of a video with visual watermarking represented by the logo on the top right corner.

Forensic Watermarking, as portrayed in Fig. 3, is used to identify the origin of leaked content by adding ownership and distributor/subscriber identification information in premium content without it being detected by the user. It can be divided in bitstream and A/B watermarking.

In bitstream watermarking, specific changes that are imperceptible to the human eye (forensic watermarking) — e.g. modifications to some pixels or bits of an image — can be applied and will uniquely identify the client that leaked the content. When that content is leaked and found to be used in the wild (namely, piracy websites), the analysis of the embedded marks will allow investigation teams to track down the origin of the leak and to stop that source of piracy.

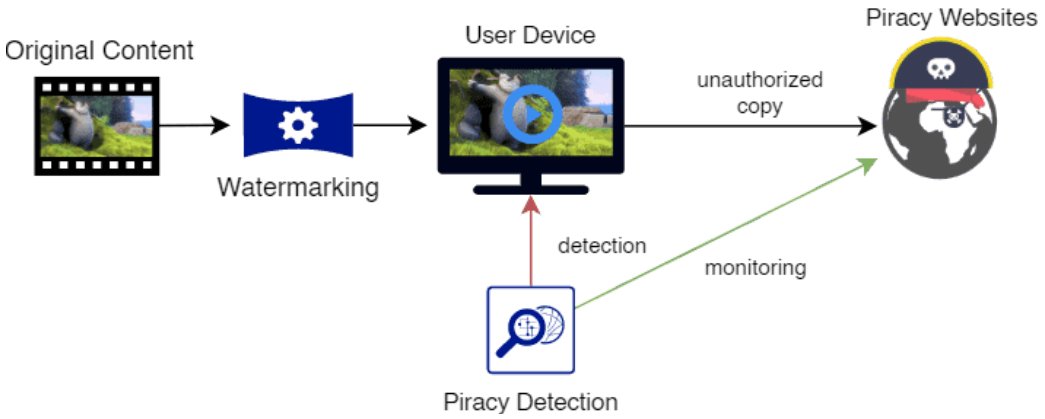


Fig. 3 — Example of a Forensic Watermarking flow for video streaming. The original video is watermarked for the given instance and, if illegally broadcasted, a forensic monitoring system will detect the culprit.

In the A/B watermarking version, the server pre-processes the content to build two watermarked versions which are combined through the use of a manifest specific to that client and session when streaming the content. There are two problems with this approach. Firstly, the CDN's would have to store the same content twice for each watermarked version. The second problem is that if attackers are able to access the same content with two different accounts, they can then mix different segments from each file and thus remove the possibility of the server tracing the client that leaked the content.

Both these forensic watermarking techniques can be embedded at the server-side, the client-side or at both ends. At the client-side, the logic is implemented at the firmware or at the SDK level and it inserts OTT client related information. The client-side approach, while not requiring integration effort at the server-side, requires integration at the client device level. This approach has to be applied on the device and security measures — including at the hardware level — have to be added. If not done thoroughly, this is likely to expose its logic to attackers.

As Watermarking solutions evolve to enable OTT providers to maximize the end-user's experience (better player performance) and reduce costs, client-side or "hybrid" approaches have grown in adoption. In the "hybrid" approach, the server will still preprocess the content to build different versions and the watermark — with the relevant metadata — is inserted at the edge servers or on the client-side.

Securing Client-Side Watermarking

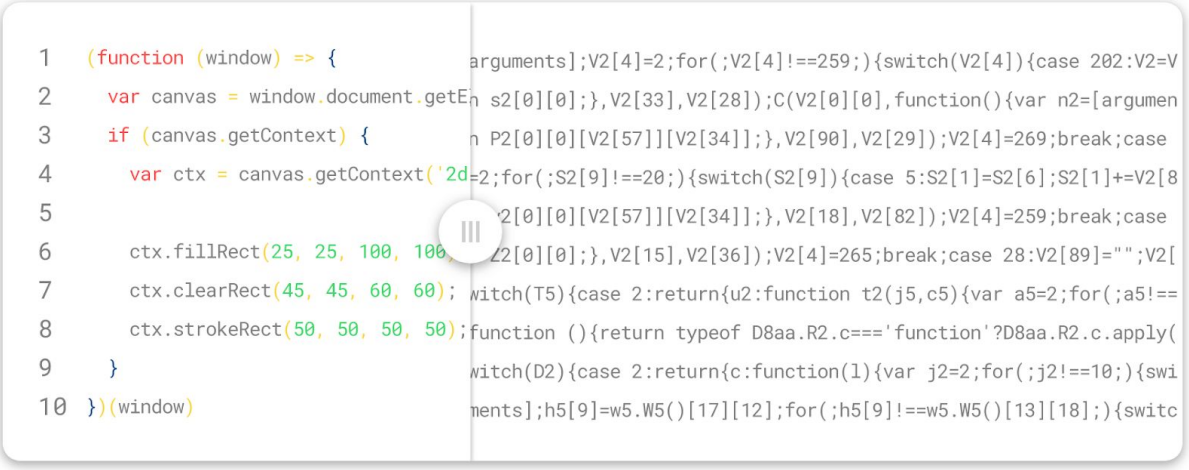
Growing demand and adoption of client-side and hybrid Watermarking solutions highlights one key security concern that Watermarking and OTT providers must address: securing the logic of the Watermarking client.

As mentioned above, placing the Watermarking client on an adversarial environment (the client-side) means that its logic is exposed. Using readily available tools such as a browser debugger, an end-user may find ways to tamper with and ultimately bypass Watermarking. This can both be achieved by reverse engineering the agent's exposed JavaScript code or by tampering with the DOM[7] — introducing, changing or removing visual elements in the application.

Client-side watermarking with JavaScript protection and webpage monitoring maximizes security, performance, and end-user experience.

[7] Document Object Model - The interface where Web players are presented to the user.

Providers can address the first problem — JavaScript reverse-engineering — by protecting the Watermarking agent's JavaScript code. While JavaScript encryption is not feasible, an industry-recommended approach is JavaScript protection. This layered security approach starts off with JavaScript obfuscation, which transforms the code into something that is extremely hard to understand and reverse-engineer, while still running on the Web browser just like the original code, as shown below.



Original **Obfuscated**

On top of obfuscation, a robust JavaScript protection approach adds anti-tampering and anti-debugging capabilities. These break the web player whenever an ill-intentioned user tries to debug or modify the Watermarking agent's logic. As a result, these capabilities prevent any type of dynamic or static code analysis.

To address the second problem — DOM tampering — providers must monitor the DOM in real-time to detect/block any attempt by an end-user to hide, remove or modify the Watermark. This includes changes to overlays, either achieved by

tampering with the DOM or by changing HTML/CSS. Such a webpage monitoring solution must detect these changes regardless of their delivery mechanism.

Major Forensic Watermarking providers employ cutting-edge JavaScript Protection and Webpage Monitoring provided by Jscrambler to ensure that their solutions can run on the client-side with minimal exposure to attacks.

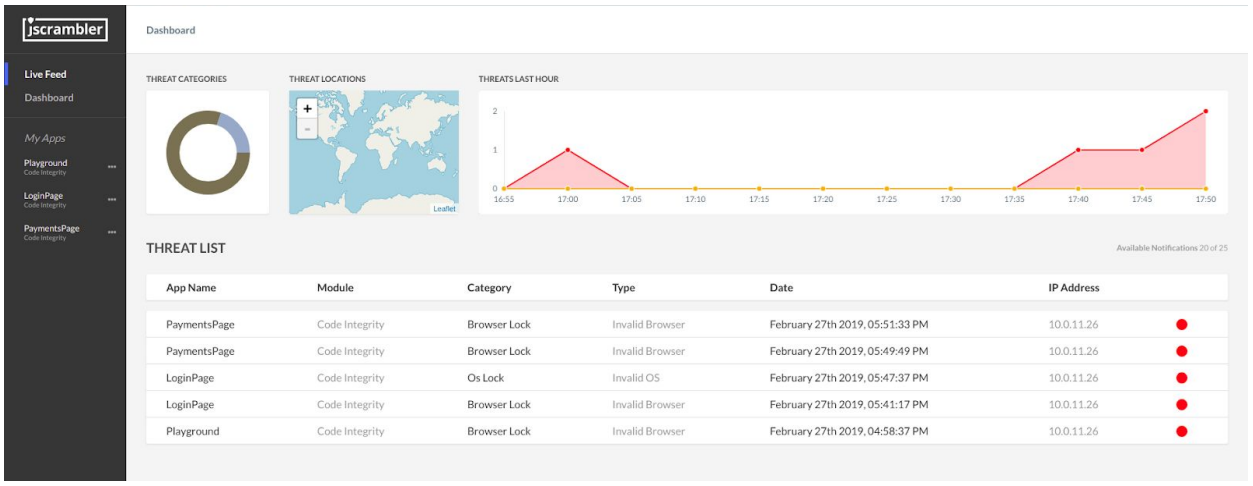
Securing the Logic of the Web Player

Competition in OTT media delivery is flourishing. The entrance of well-established players threatens subscription services, as consumers look to dial-down the number of active subscriptions. Offering exclusive content is a key driver of lower customer churn and so is the actual usability of the player. Customer behavior analyses show that a top reason to cancel a certain provider is the experience of using the player[8].

Given how Web Players become a central part of the business, providers want to ensure that their proprietary logic remains protected from the prying eyes of competitors. Some providers have disruptive solutions to handle buffering, analytics, and the user interface. Seeing how modern players rely on JavaScript/HTML5, again the answer to protect this disruptive logic is JavaScript protection. A solid JavaScript protection solution must combine polymorphic obfuscation, anti-debugging and anti-tampering capabilities. This raises the cost of reverse-engineering the web player to a point where the attack becomes absolutely uneconomical.

[8] https://www.forbes.com/sites/tonifitzgerald/2019/04/27/3-reasons-people-cancel-streaming-services-and-3-reasons-they-dont/

Unlike other solutions that protect JavaScript (or give off an illusion of protection), Jscrambler packs specific features built with OTT providers in mind. One such feature is Jscrambler Profiling, which helps ensure that the performance of the protected Web Player is identical to that of the original player. Other solutions, by lacking such a feature, often bring significant performance losses that negatively impact the end-user's experience. Another Jscrambler feature is JavaScript Threat Monitoring, which displays a live feed with all attempts to debug or tamper with the protected Player's code, as shown below. This gives security teams the upper hand in anticipating attacks and adjusting the JavaScript code protection accordingly.



The JavaScript protection solution provided by Jscrambler is employed by some of the top 5 OTT subscription companies as a major competitive advantage. By ensuring that their player's innovative features cannot be retrieved by third-parties, these companies tackle one of the main drivers of customer churn and are given solid ground upon which to battle for a share of the OTT market.

Preventing Data Leakage

Most modern OTT providers offer a one-stop shop for accessing content and managing the subscription within the same Web application. Invariably, this means that this application will be handling valuable user data, such as credentials, personally identifiable information (PII) and credit card details. As a result, such Web applications have become a prime target of attacks that seek to exfiltrate this data. A widespread threat, known as Magecart, injects a credit card skimmer on payment pages in a way that is imperceptible for the end-user and the infected company.

Magecart attacks often occur without attackers ever breaching the servers of the target company. In an approach known as a Web Supply Chain Attack, Magecart attackers compromise a third-party service that is used by the targeted OTT service provider. This is done by injecting malicious code that silently makes its way into the Web application and often remains active and undetected for months. Such Magecart attacks have resulted in well over 500,000 stolen credit cards and the attack on British Airways resulted in a GDPR fine of $230 million[9].

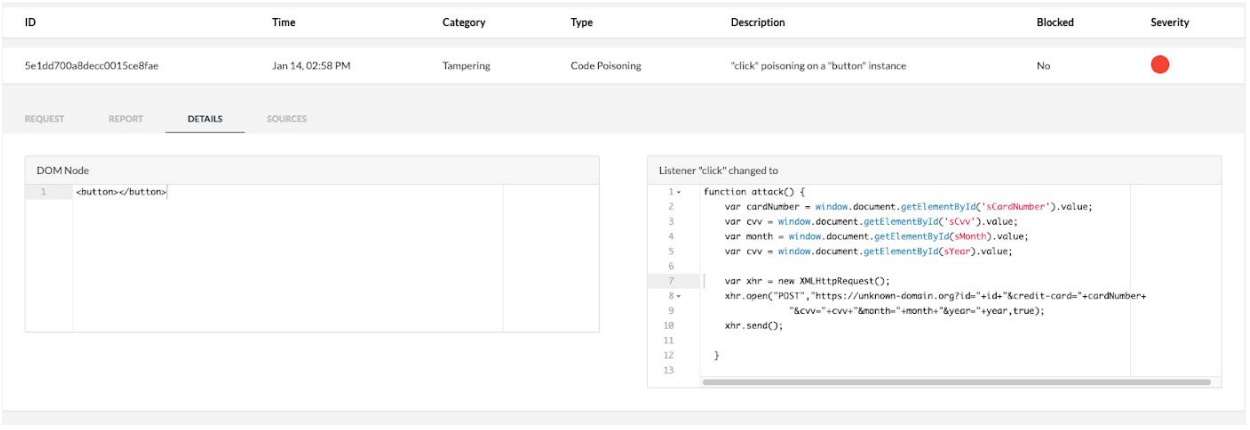
The lack of proper knowledge about Magecart attacks generates a lot of confusion on how to properly address this threat. While it is true that companies should seek to vet externally sourced code to avoid trusting an ill-secured provider, this is often unfeasible. Modern Web apps rely on thousands of pieces of third-party code and companies can't feasibly vet each one. Security systems like Web Application

[9] https://edition.cnn.com/2019/07/08/tech/british-airways-gdpr-fine/index.html

Firewalls don't catch the malicious code as it originates from a trusted source. The task ultimately falls to client-side security systems, which are often bypassed by Magecart attackers, namely with bot detection techniques.

A safer approach when dealing with Magecart-like attacks is assuming that there is no guaranteed way of preventing this injection of malicious code. Instead, companies must have systems in place that effectively **detect** Magecart attacks and can **block** the attack and attempt to exfiltrate users' data.

To this extent, [Jscrambler's Webpage Integrity module] brings a proven Magecart mitigation approach that, using behavior-based techniques, is able to detect and block Magecart-like attacks in real-time, as shown below.



Filling the Security Gap

Innovation and customer-centrism are bound to shape the OTT industry in the coming years. Technological breakthroughs have unlocked new ways of delivering value to consumers but OTT providers cannot afford to ignore the security gaps that arise from this disruption.

A sustainable approach to client-side security means going in-depth.

As the Web becomes a prime location to deliver content to end-users, attackers exploit a number of client-side security weaknesses that too often are left unaddressed by companies. A sustainable approach to client-side security means going in-depth, securing any important logic that runs on the client-side — both the Watermarking client and the logic of the Web Player — with JavaScript protection and gaining full visibility of attacks that tamper with the DOM — to bypass Watermarks — and seek to exfiltrate user data — Magecart attacks.

Recognized by the industry and trusted by several major OTT providers, Jscrambler's holistic approach to both these dimensions provides an in-depth approach to minimize companies' exposure to these attacks.

Contact Us

If you want to know more about how Jscrambler can help you
Secure your OTT platform, don't hesitate to contact us

hello@jscrambler.com

+1 **650 999 0010**

Gartner®

Jscrambler is the leader in Client-Side Application Security
Recognized in **Gartner's Market Guide for Online Fraud Detection**
and in **Gartner's Market Guide for In-App Protection**