jscrambler

Preventing magecart attacks

A info sheet by Jscrambler

Magecart

A major global business threat

"Magecart" refers to a collective of cybercriminal groups that inject **digital credit card skimmers** on e-commerce and payment websites. These groups have been active since 2015, but have gained momentum from 2018 onwards.

In a Magecart attack, attackers inject the skimmer (through malicious JavaScript code) into a company's payment page. This code actively listens to events that happen on the page and collects credit card details whenever a user submits them in a form (event hijacking). These details are then sent to attackercontrolled drop servers. During this whole process, neither the end-user nor the company have any awareness that the attack took place. Because of this, **many Magecart attacks remain active for months** before being detected and taken down.

There are two main attack approaches: first-party and third-party. In a first-party attack, malicious actors **gain access to the victim's website** and directly place the skimmer in the payment page. In a third-party attack, **this malicious code is injected through a third-party provider** that the victim company is using.

Third-party Magecart attacks are especially critical because they **don't require a firstparty server breach** or direct access to the company's website. Even companies with robust Web Application Firewalls and server-side security are susceptible to these attacks because they exploit **client-side security weaknesses.**

$1.3 B+
business losses from known Magecart attacks.

$230M
GDPR fine on British Airways following a 2018 Magecart attack.

104 Days
average time during which Magecart skimmers remain active before discovery.

The client-side security gap

Third-party Magecart attacks are a prime example of web supply chain attacks - a type of attack where malicious actors are able to find and exploit the weakest link of a company's web supply chain.

For example, attackers can easily find out that their target is using a specific live chat widget on their website. Then, if they manage to breach the supplier of this widget (which typically is an easier task considering that **many of these suppliers are very small and have tiny security budgets**), they can hide the skimmer inside the widget's source code and that malicious code ships down the supply chain and runs directly on the target company's website.

Most third-party code providers don't have enterprise-grade security systems.

Every piece of third-party code that a company uses in its website can become a vehicle for Magecart attacks. This not only includes **website scripts like Analytics, Ads, and widgets,** but also code **libraries used during the development process.**

Why most security approaches fail

Despite a great push to spread awareness on how to properly address Magecart, **new attacks are emerging every week and getting more sophisticated**. Companies are gradually understanding the need to think outside the firewall and looking for client-side security solutions.

Given the wide range of existing security solutions that attempt to prevent Magecart attacks, it's crucial to understand how each approach is able to (or fails to) tackle these security threats, as summarized below.

Domain sinkholing

Approach: Redirects the flow of requests to other servers, preventing the connection to attackers' drop servers.

Limitations: Signature-based. Bypassable by changing the attack injection.

Content security policy (CSP)

Approach: Restrict domains and resources based on a whitelist, preventing the connection to attackers' drop servers to send exfiltrated data.

Limitations: 94% of header-based CSPs are bypassable. Can break things.

Virtual iframes

Approach: Isolates third-party code inside individual IFrames, filtering events based on a static whitelist.

Limitations: Introduces performance drops and new race conditions that can break the app.

Subresource integrity(SRI)

Approach: Only load scripts that pass an integrity check, preventing the load of a script if its content changes.

Limitations: Locks you to specific script versions. Not all providers use SRI.

Behavior-based magecart mitigation

As shown before, most approaches fail to properly mitigate Magecart, especially when we consider the last generation of skimmers which **remain hidden by using bot detection techniques.**

Preventing Magecart attacks altogether is a near-impossible task. **There are too many ways in**, especially when we consider attacks that originate from thirdparty code. As so, conceptually, the best approach is to be able to **detect and block the malicious behavior** that Magecart attacks inflict upon a web page.

Jscrambler Webpage Integrity (WPI) does this by using **rule-based behavior control**. WPI detects several different types of malicious behavior - both in terms of resources and network events - which happen in any Magecart attack. Then, using fine-grained permission levels (based on high-level assumptions and userdefined rules), **WPI can block**, in real-time, any malicious behavior on the client-side of web applications - including Magecart attacks.

Because of this approach, Jscrambler WPI can detect/block Magecart attacks in 3 main touchpoints:

#1 New unknown script

When a new unknown script runs on the website > **Webpage Integrity can block the script**

#2 Existing script

When an existing script changes its behavior > **Webpage Integrity can block the script**

#3 Suspicious outbound event

When a suspicious outbound network event is detected > **Webpage Integrity can block the connection**

Why choose Jscrambler WPI?

Holistic solution

Unlike the security approaches presented earlier, Jscrambler WPI is holistic - it can be readily employed to detect and mitigate a wide range of client-side attacks. This holistic approach to Magecart detection and mitigation also leads to reduced false positives, easier maintenance and configuration, and no need for signatures. This is achieved by crossing multiple information sources gathered by WPI, including resources loaded to the page, DOM changes, and code poisoning.

Build reports & get insights

WPI's agent is embedded in every user session, giving full visibility of the client-side in real-time. The WPI dashboard provides multiple views, including a detection live feed, an explorer, and an inventory. These allow a comprehensive full-scope view of alerts from a macro outlook to a detailed description. The collected information can easily be sent to a SIEM to be further explored, with different filters and visualizations.

Secured with resilient code protection

Jscrambler WPI is the only solution to include extremely resilient code protection of the real-time monitoring agent. As a result, attackers cannot bypass detection. This resilient JavaScript protection can be extended to the source code of the application, to close yet another gap in client-side security: JavaScript code tampering and reverse-engineering.

Trusted by industry leaders

WPI is successfully employed by major online retailers, airlines and financial institutions to detect and stop client-side attacks like Magecart and web supply chain attacks.

If you want to know more about how Jscrambler can help you prevent client-side attacks, don't hesitate to contact us.

hello@jscrambler.com | +1 650 999 0010