Security Incident Initial Report for Customers – Optimizely Web Experimentation

17 November 2026

1. About this update

We are writing to share an update with additional details on the Web Experimentation security incident about which we previously provided notification. As we shared yesterday, we promptly remediated the vulnerability and removed the injected code from affected projects; while our investigation continues with the help of third-party experts, we remain confident in the additional safeguards we have implemented to ensure the security of the Web Experimentation service.

We take the trust you place in us extremely seriously and are committed to being as transparent as possible. In that spirit, this update includes additional technical information on the incident and our response, initial findings from a third-party forensic investigation, and guidance on actions customers should take, as well as answers to a number of frequently asked questions.

2. Executive summary

On 16 November 2026, an unauthorized third party exploited a vulnerability in the Optimizely Web Experimentation account signup flow. They used it to impersonate users and insert malicious JavaScript into the Project JavaScript of some customer projects. This was not the result of a compromised Optimizely administrator account.

Our investigation determined that the first malicious code was written on 16 November 2026 at 07:51 UTC. That same day, Optimizely detected the activity at 13:34 UTC, closed the vulnerability at 16:10 UTC, and removed the injected code from all affected projects by 22:24 UTC. Delivery of the malicious code is now blocked, and no malicious code remains in Optimizely systems.

This was not a credentials breach. The unauthorized third party did not steal or use customer passwords. They exploited a flaw that let them act as administrator users without those users' credentials. We have no evidence that any customer password was exposed as a result of this incident.

What this means for you:

- **Only Web Experimentation was involved.** Feature Experimentation, the SDKs, and legacy Full Stack were not affected. We have no indication that any other Optimizely product was affected as a result of this incident.
- **Exposure was through your website, not your network.** The injected script ran in the browsers of visitors who loaded an affected page with the Web Experimentation snippet. Sites with a strong Content Security Policy (CSP) were protected.
- **Our investigation determined that the injected script was intended to obtain payment information and crypto activity only.** The script targeted card details typed into forms on the site itself and cryptocurrency transfers. Cards entered in hosted payment iframes (for example Stripe or Adyen) were not accessible.
- **Not every customer was targeted.** Optimizely has confirmed directly with each customer whether the customer was affected by this incident. If you have not received confirmation, contact your Customer Success Manager.

Steps we recommend customers take: treat this as a security incident and review your site activity for the exposure window. See section 5 for the full list of actions we recommend customers—both affected and unaffected—should take.

Optimizely promptly engaged **Mandiant**, an industry-leading third-party forensics firm, to independently investigate this incident. Section 10 includes preliminary indicators of compromise intended to assist clients in validating the security of their environments; we will confirm the root-cause and final indicators of compromise list when the investigation is complete.

3. Incident overview

Attribute	Detail
Incident type	Unauthorized modification of customer JavaScript
Affected product	Optimizely Web Experimentation
Not affected	Feature Experimentation, SDKs, legacy Full Stack. No indication of impact to other Optimizely products.
Attack vector	Flaw in the account signup flow, used to impersonate administrator users (see section 7)
Credentials	Not a credentials breach. No evidence customer passwords were exposed.
Exposure window	07:51 to 22:24 UTC, 16 November 2026. Per-customer windows: **available via CSM**
Optimizely-held data accessed	Any data in the Experimentation Platform available to the impersonated user. No other data from other Optimizely products.
Status	Contained and remediated. Forensic investigation ongoing.

4. Timeline of events

Key milestones (preliminary), 16 November 2026 (UTC):

- **00:37** First impersonated user account created by the unauthorized third party
- **07:51** First malicious code written to a customer project
- **13:34** Malicious activity detected
- **16:06** Malicious outbound code blocked at Cloudflare
- **16:10** Flaw addressed and vulnerability remediated
- **16:53** All impersonated users' active sessions ended and tokens revoked
- **22:24** Malicious code removed from all affected projects

Per-site windows of exposure are available directly from your Customer Success Manager.

5. Customer actions required

Every Web Experimentation customer should consider all actions:

#	Action	Priority
1	Treat this as a security incident and investigate any impact to your site	Immediate
2	Confirm which pages the Optimizely Web Experimentation snippet was running on during the exposure window (07:51 to 22:24 UTC, 16 November 2026). If it ran on any page with credit card entry (including CVV) or a cryptocurrency transfer, treat the risk as high.	Immediate
3	Review CSP violation reports and outbound browser requests to unfamiliar domains, including those in section 10, between 07:51 and 22:24 UTC on 16 November 2026.	Immediate
4	Review credit card and cryptocurrency transactions made on your site during the exposure window, where applicable.	Immediate
5	Assess with your own legal advisers whether you have any legal obligations or notification requirements to your own customers or regulators.	High
6	Retain your own logs (web server, CDN, WAF, CSP reports, payment and fraud monitoring) covering at least 16 November 2026 for your investigation.	High
7	Check that a strong Content Security Policy is in place on pages that load the Web Experimentation snippet.	Recommended

#	Action	Priority
8	Remove the Optimizely Web Experimentation snippet from any pages that handle sensitive data input, including payment or crypto transfers, and that you are NOT running experiments on.	Recommended

Re-enabling Web Experimentation: If you removed or commented out the snippet, it is now safe to re-enable it. Early guidance on 16 November advised turning it off. That guidance has been withdrawn.

If you prefer to keep it off while you finish your review, on sites without CSP protection you can remove or comment out the Web Experimentation snippet (the script loaded from cdn.optimizely.com) in your page templates or tag manager. This is the most complete option.

Not required: Feature Experimentation, SDK, and legacy Full Stack integrations need no changes.

6. Risk to customers

The main risk is to visitors of affected sites who entered payment card details into forms on the site itself, or made cryptocurrency transfers, while the malicious code was being served. Risk depends on whether your projects were affected, and on how your site is built.

Your situation	Risk level	Why
Confirmed affected, no strong CSP, card details entered in forms running Web Experimentation	High	The injected script could read and exfiltrate card details typed into forms on the page. Presence of the Optimizely snippet on the page would trigger the injected script execution.
Confirmed affected, no strong CSP, crypto transfers on site in forms running Web Experimentation.	High	The script targeted cryptocurrency transfers by replacing the destination wallet ID.
Confirmed affected, payments only through hosted iframes (for example Stripe or Adyen)	Lower	Hosted payment iframes were out of the script's reach.
Confirmed affected, strong CSP in place	Lower	The injected script loaded from an outside source, so a strong CSP would have blocked it. Confirm with your CSP violation reports.
Confirmed unaffected by Optimizely	Low	No malicious code was found in your projects.
Not a Web Experimentation customer	None identified	No indication that other Optimizely products were affected including Feature Experimentation or Mobile SDK.

Additional risks assessed

- The script tried to capture payment card number, expiry and CVV from forms and from POST requests on affected pages. Readable cookies, the page URL and a screenshot were sent only on pages where it had swapped a crypto wallet address. The version we have recovered doesn't send other form fields, and it doesn't read session or local storage. The script first tried to download its main code from an attacker server.
- On iOS and iPadOS the script loaded a separate second-stage script from another attacker domain which targeted iOS / iPadOS versions 18.6 and lower.

7. Attack vector and how it was closed

The attacker used a flaw in the Web Experimentation account signup flow to act as administrator users on customer accounts, then edited Project JavaScript to add a card skimmer. Optimizely closed the flaw at 16:10 UTC and removed all injected code by 22:24 UTC, the same day the incident was identified.

How the attack worked

1. **Account creation.** The attacker created accounts through the signup flow, starting at 00:37 UTC. We have linked the activity to IP addresses that include commercial VPN (ExpressVPN) addresses.
2. **Impersonation.** A flaw in the signup flow let those accounts act as existing users on customer accounts through a combination of public information and brute force. No passwords were stolen or used to do this.
3. **Code injection.** Acting as those administrators, the attacker edited Project JavaScript in targeted projects, starting at 07:51 UTC on 16 November 2026. The added code loaded a card skimmer from attacker-controlled infrastructure. Mandiant identifies pay.pay-cloud[.]online as a host for the skimmer.
4. **Execution in visitors' browsers.** The Web Experimentation snippet delivered the modified Project JavaScript to visitors of affected pages. The skimmer targeted card details typed into on-page forms and cryptocurrency transfers. We have identified pay-relay.wordpressadmin001.workers[.]dev as the skimmer's exfiltration endpoint.
5. **Where it was blocked.** Because the skimmer loaded from an outside domain, a strong Content Security Policy stopped it from running. Hosted payment iframes (for example Stripe or Adyen) were out of its reach.

Why this was not a credentials breach

- The attacker did not obtain or use customer passwords. They exploited a previously unknown flaw to act as administrators.
- No Optimizely administrator account was breached.
- We have no evidence that any customer password was exposed.
- Customers do not need to reset passwords or regenerate API tokens.

Analysis of the injected scripts

Optimizely has completed a preliminary analysis of the injected scripts.

What is known so far: the injected code loaded a card skimmer from an outside domain. The skimmer targeted card details typed into on-page forms and cryptocurrency transfers, and sent captured data to the exfiltration endpoint, pay-relay.wordpressadmin001.workers[.]dev.

Question	Answer
Has Optimizely analyzed the injected scripts?	Yes
Was the same script used in every affected project, or did it vary?	The same script was altered multiple times during the attack in attempt to get further functionality.
What triggered the skimmer to run (pages, form fields, or events it watched for)?	The script executes immediately upon loading—with no page or URL restrictions—and continuously monitors the entire page using a 50-millisecond polling loop, DOM mutation observers, and clipboard event listeners.
Exactly what data did it collect (card fields, email, phone, cookies, other form data)?	It collected form fields from a submit event, specifically name, id, credit card number, expiration. Clipboard with wallet address matches were swapped with malicious addresses and a .jpg of the screen along with document.cookie was sent to an attacker-controlled server.
How and when did it send data out?	Data was sent on form submissions.
Did it behave differently by browser or device (for example the reported iOS-only second stage)?	No, except for iOS / iPadOS where an exploit kit install attempt as a secondary attack occurred for versions before 18.6.
Did it try to hide itself, persist, or load further code?	We have no evidence that the code tried to obscure itself other than obfuscation.

Question	Answer
Did the attacker add to customers' code or replace it?	If the customer had custom injected JavaScript, the attacker's code replaced it.

How the unauthorized third party identified target accounts and moved between them

Question	Answer
How did the attacker obtain account and project IDs?	Account IDs and Project IDs are present in the Optimizely snippet. These values are references used in the function of the snippet and are not account secrets.
How did the attacker move from one customer account to the next?	The attacker presented a pattern of opportunistic attempts to guess user account email addresses, attempting variations on customer employee names. The attacker repeated this pattern for each impacted customer up to thousands of times. Gaining access to one account did not provide the attacker access to another customer account.
Could the attacker see account, user, or project details beyond what was needed to make the change?	The attacker could access experiment details as allowed by the privileges granted to the exploited user. This could include projects, experimentations, and related configuration. The attacker did not have access beyond the Optimizely Experimentation product.

How the vulnerability was addressed

Stage	Action	When (16 Nov, UTC)
Containment	Blocked malicious outbound code at Cloudflare	16:06
Containment	Closed the signup-flow flaw and deployed a fix. Project JavaScript can no longer be changed this way.	16:10
Containment	Ended active sessions, revoked tokens, and reset access for the impersonated users	16:53
Eradication	Removed injected code from all affected projects and blocked its delivery at Cloudflare	22:24
Recovery	Confirmed no malicious code remains in Optimizely systems. Withdrew advice to disable Web Experimentation.	22:24
Recovery	Restore original code where the attacker replaced it rather than added to it	22:24

8. Investigation status and third-party forensics

Optimizely has engaged **Mandiant**, an industry-leading third-party forensics firm, to carry out an independent investigation alongside Optimizely Security.

Customers who need specific records for their own investigation should request them through their Customer Success Manager or support@optimizely.com. Log requests will be prioritized accordingly with our internal investigation taking priority.

We will confirm when the investigation is complete, and will provide any relevant updates regarding the root-cause and indicators of compromise. Forensics firms have been heavily loaded by security incidents over the last few weeks, which may delay their production and our release of the final report.

9. Attribution

The actor behind this incident has not been named. Attribution is pending.

Preliminary research shows so far:

- The actor controls pay-cloud[.]online and has run two command-and-control frameworks (ADAPTAGENT and CUPCAKE C2) from it since at least September.

- The same infrastructure hosts JavaScript card skimmers used in campaigns against other victims, not only this incident.
- The actor connected through IP addresses that include commercial VPN (ExpressVPN) addresses (193.176.211[.]170 and 193.176.211[.]203). These are listed for reference only. VPN addresses are ephemeral and are not reliable IOCs.

10. Indicators of compromise

Preliminary indicators of compromise (IOCs):

- **pay-relay.wordpressadmin001.workers[.]dev** – where the skimmer sent captured data. Any browser request from your site to this domain is a strong sign the skimmer ran.
- **pay-cloud[.]online** and **pay.pay-cloud[.]online** – attacker infrastructure that hosted the skimmer.

Block these domains and search your web, CDN, CSP, and proxy logs for them. IOCs are preliminary and may be updated.

11. Customer questions and answers

These are the questions customers and their security teams have raised by email, support ticket, and call, grouped by theme. **Answered** means confirmed as of 17 November 2026. **Partial** means part of the answer is confirmed. **Pending** means the answer depends on the investigation.

A. Were we affected?

#	Question	Answer	Status
A1	Were our account, projects, or snippet affected?	Optimizely is confirming with each customer directly. Unless you received the notice headed “your site was UNAFFECTED,” you can assume you were impacted, and your Customer Success Manager will be in touch with more precise customer-specific information.	Answered
A2	The notice says we’re unaffected but also says to treat this as an incident. Which is it?	The first line is your customer-specific status. The rest of the notice repeated the original incident notification sent to all Web Experimentation customers.	Answered
A3	How did you determine we were unaffected, and could log gaps change that?	We have clear logs for projects targeted and JavaScript changes as they happened via our application, so we could eliminate customer projects that did not have malicious activity.	Answered
A4	Can we have written confirmation for compliance or third-party risk?	We are still working on a written confirmation of third-party engagement.	Pending
A5	Can we get a list of other customers (who may be our vendors) affected?	Optimizely cannot disclose our customers.	Answered
A6	Is it only Web Experimentation? Are CMS, DXP, ODP, Feature Experimentation, or the SDKs affected?	Feature Experimentation, the SDKs, and legacy Full Stack were not affected. No indication that other products were affected.	Answered
A7	Is the PCI CDN (cdn-pci.optimizely.com) included?	Yes, the mechanism used to inject the script bypassed our protections hosting our high-security PCI script.	Answered
A8	Could archived, unused, or blank-JavaScript projects have served malicious code?	We reviewed all projects for malicious code. Inactive projects would not serve JavaScript code to sites.	Answered
A9	Was anything besides Project JavaScript changed (experiments, audiences, extensions, shared code,	We have strong evidence of what was altered, as it was logged by our application. We have no	Answered

#	Question	Answer	Status
	integrations, webhooks, collaborators, tokens)?	evidence that anything beyond the JavaScript was modified.	
A10	Was the Optimizely snippet or CDN itself compromised?	The confirmed malicious changes were to customer Project JavaScript. The CDN just relayed that content.	Answered
A11	Was Personal Data taken?	Currently, we do not have evidence that personal data was exfiltrated.	Answered

B. When did it happen?

#	Question	Answer	Status
B1	What is the overall timeline?	See section 4.	Answered
B2	When exactly was malicious code served to our visitors (first and last time), and which revision numbers?	Your CSM can provide exact timelines per customer.	Answered
B3	Did the Cloudflare block at 16:06 UTC fully stop delivery?	The 16:06 UTC action blocked malicious outbound code based on known patterns. Delivery was fully blocked.	Answered
B4	Can you give the timeline in our local time zone?	All times in this report are UTC (for example, 07:51 UTC is 02:51 US Eastern on 16 November).	Answered

C. How did it happen, and is it fixed?

#	Question	Answer	Status
C1	How did the attacker get in?	A flaw in the Web Experimentation account signup flow let them impersonate administrator users. See section 7.	Answered
C2	Is it fixed?	Yes. The vulnerable path was closed at 16:10 UTC and a fix deployed.	Answered
C3	Was an admin account compromised?	No. This was not a credentials breach. A flaw was exploited to impersonate administrator users, and no Optimizely administrator account was breached.	Answered
C4	Did the attacker create the account, or impersonate a real user? Are other users affected?	The created account was unrelated to your users and only used as a mechanism to gain access via impersonation. No other users including the impersonated user are impacted.	Answered
C5	Our users use SSO, but a change is attributed to someone with no login. How? Were personal access tokens, API tokens, or local logins used?	The exploited flaw was to impersonate another user which bypassed authentication.	Answered
C6	What is the “Optimizely Support” change in our project history? Did Optimizely change our project?	This is the restoration of the JavaScript code by our Engineering team.	Answered

D. What data was at risk?

#	Question	Answer	Status
D1	What was the malicious code designed to do?	Capture card details typed into forms on the site itself, and target cryptocurrency transfers. There was code to exploit older iOS vulnerabilities.	Answered
D2	Were payment details captured?	Cards entered in hosted payment iframes (for example Stripe or Adyen) were out of reach due to Same-Origin policy. Cards typed into on-page forms (rare) on affected sites may have been	Partial

#	Question	Answer	Status
		exposed. Whether any were actually captured is not yet confirmed.	
D3	What else could the script read (email and phone fields, cookies, sign-in pages)? How many visitor sessions were exposed?	Optimizely only has visibility into when the injected code got loaded, not if the code was successful or what data was exfiltrated if any.	Answered
D4	Was data held by Optimizely accessed (visitor data, event data, account data, API tokens)?	Anything that the impersonated account had access to in the Experimentation Platform.	Answered
D5	Were customer passwords exposed?	No. This was not a credentials breach, and we have no evidence that customer passwords were exposed.	Answered
D6	Did Optimizely have access to our network, VPN, or systems, or hold credentials to them? Is our network at risk?	The incident involved code running in visitors' browsers through the Web Experimentation snippet. This had no access to customer networks.	Answered
D7	Is there an iOS-only second stage? Do we need to notify visitors whose devices loaded the script?	See section 6 (Additional risks assessed) for details of the iOS / iPadOS second stage.	Answered

E. What do we need to do?

#	Question	Answer	Status
E1	What actions should we take?	See section 5.	Answered
E2	Should we rotate credentials or API tokens, or log users out?	No. Credentials and API tokens do not need to be reset.	Answered
E3	When is it safe to turn the snippet back on?	Now. It's safe to re-enable Web Experimentation.	Answered
E4	How do we disable it while we finish our review?	See section 5	Answered
E5	Do we need to clear CDN caches?	CDN caches are cleared already.	Answered
E6	What should our own investigation include? What indicators should we look for?	See sections 5 and 10.	Answered
E7	How do we remove injected code, roll back Project JavaScript, or restore code that was replaced?	Affected projects were cleaned by Optimizely. For restoring replaced code, contact Support.	Answered
E8	How do we enforce MFA, end sessions, and audit user roles?	You should always review what users and permissions are granted. MFA is available in both Experimentation local login and OptiID. We strongly encourage integration into your corporate IdP. Sessions were already purged.	Answered

F. What is Optimizely doing to prevent this?

#	Question	Answer	Status
F1	What's changing so this doesn't happen again?	We have additional protections in place to prevent unauthorized code from being delivered. Our go-forward strategy to review and harden our surface area will be informed by the findings of the third-party investigation and set out in the final report.	Pending
F2	Can Project JavaScript require approval before publishing, or be limited to certain editors?	We have this in as a feature request.	Pending

#	Question	Answer	Status
F3	How should we harden our site (CSP, version pinning, running one version behind, signed versions)?	A strong CSP protected sites in this incident.	Answered

12. Communications, evidence, and next update

#	Question	Answer	Status
G1	Who do we contact?	Your Customer Success Manager, or support@optimizely.com.	Answered
G2	Can Optimizely join our incident or crisis calls?	Your Customer Success Manager can meet with you, or, if you need more information, can put a request in for a technical or executive call. We will make every effort to meet such requests.	Answered
G3	When is the next update, and where will updates be posted (for example the status page)?	We are sharing information as quickly as we can confirm it. Expect daily updates until we close the incident. This will be shared individually with customers and partners.	Answered
G4	Can we get audit and change logs, old snippet versions, and records preserved for our investigation?	Optimizely's application logs record which projects were targeted and what JavaScript changes were made (see A3 and A9), and your Customer Success Manager can provide per-customer timelines (see B2). Request these records through your Customer Success Manager or Support. Optimizely can only see when the injected code loaded, not what data it captured (see D3), so keep your own web, CDN, and CSP logs (see section 5).	Answered
G5	Is there a formal statement or root-cause report we can share internally?	This is an interim report; a final root-cause analysis and indicators of compromise will be provided when the investigation is completed.	Answered
G6	Who received the notice? Can we share it? Can the recipient be changed to a shared mailbox?	The recipient list can be confirmed, and changed, by contacting your Customer Success Manager.	Answered