



INCIDENT RESPONSE & COMPROMISE ASSESSMENT REPORT

Client

- Affected Shopify Application

Web File Manager Compromise → DB Credential Theft → Shopify Token Exposure

FINAL STATUS: PATCHED / REMEDIATED

Affected application	Affected Shopify Application	
Primary environment	[REDACTED ENVIRONMENT] backend	/ Client
Original severity	CRITICAL	
External detection start	16 Aug 2026, 10:59:27 UTC	
Prepared by	Farchase Solutions Pvt. Ltd.	

Real Bugs. Real Impact. Real Fixes.



01 Executive Summary

Farchase Solutions reviewed a critical security incident affecting the Client Shopify application environment. The incident chain involved abuse of executable PHP upload functionality to place a web-based file manager on the application server. The attacker then used that server-side file manager to browse application files and read config.php, exposing backend database credentials. A second PHP artifact, exp.php, was then used with those credentials to connect to the application database, enumerate tables and export readable table contents. Where Shopify access tokens were stored in readable database fields, those tokens were included within the accessible data set.

Key Incident Finding

The attack was not limited to a vulnerable upload endpoint. The uploaded web file manager created a pivot into the server filesystem, the configuration file exposed database secrets, and the database export script converted those secrets into broad data-access capability. This chained compromise could expose data belonging to multiple Shopify merchants, including access tokens.

Independent Shopify Detection

Shopify separately reported approximately 906 potentially anomalous script-tag modification requests beginning at 2026-08-16T10:59:27.044+00:00 against Affected Shopify Application originating from numerous external IP addresses and multiple autonomous systems. This alert provides an external incident signal that should be correlated with web-server, database and Shopify API logs.

Current Status

PATCHED / REMEDIATED

The vulnerable application paths are reported as patched. Incident closure should additionally confirm credential rotation, Shopify-token review/revocation where applicable, removal of uploaded attacker artifacts and completion of historical log analysis.



02 Incident Scope & Evidence

Item	Evidence / Position
Initial access	Executable PHP upload weakness in Client Announcement / Web Push functionality.
Attacker artifact #1	webadmin.php - browser-accessible web file manager uploaded to the server.
Credential source	config.php - application configuration read through the compromised server context.
Attacker artifact #2	exp.php - database export script configured with database connection details.
Data target	Backend application database, including merchant and token-related records where readable.
External signal	Shopify alert reporting ~906 anomalous script-tag modification requests from 16 Aug 2026.

Evidence Classification

The uploaded webadmin.php source confirms file-manager capabilities including directory navigation, file viewing/downloading, upload, creation, editing, moving/copying and, under certain conditions, script execution. The exp.php artifact confirms database connection, SHOW TABLES enumeration, full-table SELECT retrieval and SQL-dump output. The precise historical sequence should be corroborated with timestamps and access logs where available.

Affected Security Boundaries

- Web application upload boundary
- Application server filesystem
- Application configuration / secrets
- Backend database authentication
- Multi-tenant merchant data
- Shopify access-token confidentiality
- Potential Shopify API integrity if exposed tokens were subsequently used



03 Attacker Execution Walkthrough

1. Exploit PHP Upload

The attacker identified an upload path that accepted executable PHP content. The attacker uploaded webadmin.php to a web-accessible location where the PHP runtime processed the file.

2. Execute webadmin.php

The attacker requested the uploaded file through a browser. Once executed, webadmin.php presented a web-based file-management interface running with the permissions of the application/web-server process.

3. Browse Server Files

Using the file manager, the attacker navigated directories accessible to the web-server account and enumerated application files. The file manager supported viewing and downloading readable files.

4. Read config.php

The attacker located config.php and read its contents. This exposed the database host, database username, database password and database name used by the application.

5. Prepare exp.php

The attacker placed the recovered database connection information into exp.php, a PHP database-export script.

6. Upload / Execute exp.php

The attacker uploaded or otherwise placed exp.php into the compromised server environment and requested it through the browser.

7. Connect to Database

exp.php connected directly from the application server to the MySQL backend using the compromised credentials. This leveraged the server's trusted network position rather than requiring external database exposure.

8. Enumerate Tables

The script executed SHOW TABLES to discover every database table visible to the compromised database account.

9. Export Data

For each table, the script retrieved schema information and executed SELECT * against the table, returning every readable row and column.

10. Obtain Shopify Tokens

If Shopify access tokens were stored in readable columns within those accessible tables, the tokens were included in the generated database dump and became available to the attacker.



04 Confirmed Compromise Chain

Stage	Attacker Action	Security Impact
Initial Access	Upload webadmin.php through vulnerable PHP upload	Server-side executable code introduced
Server Pivot	Open webadmin.php and browse server files	Filesystem access within web-server permissions
Secret Discovery	Locate and read config.php	Database credentials compromised
DB Tooling	Configure and execute exp.php	Trusted server makes DB connection
Enumeration	SHOW TABLES / inspect schema	Database structure exposed
Extraction	SELECT * from accessible tables	Bulk database contents disclosed
Token Exposure	Identify Shopify access-token records	Cross-tenant authentication material exposed
Potential Follow-on	Use valid tokens within granted scopes	Possible unauthorized Shopify API actions - requires log confirmation

Attack Path

Vulnerable PHP Upload → webadmin.php → Server Filesystem → config.php → DB Credentials → exp.php → Backend Database → Table Dump → Shopify Token Exposure

Why the Chain Was Critical

Each weakness amplified the next. File upload became interactive server access; server access exposed secrets; exposed secrets enabled database authentication; database access crossed merchant boundaries; and readable Shopify tokens could extend the compromise into connected Shopify stores according to the tokens' granted scopes.



05 Malicious / Attacker-Used Artifacts

Artifact A - webadmin.php

The uploaded PHP file is a web-based file manager. Its source provides functionality to browse directories, view readable files, download files, upload files, create and edit files, copy/move/delete files and inspect permissions. On compatible hosts it also exposes execution functionality for executable scripts.

Capability	Incident Relevance
Directory listing	Enabled discovery of application files and configuration locations.
View / download	Enabled reading of config.php and other sensitive files readable by the web-server account.
Upload / create / edit	Enabled placement or modification of additional PHP tooling such as exp.php.
Copy / move / delete	Enabled manipulation of server files and potential evidence destruction/persistence.
Execute where available	Could further extend server-side execution capabilities.

Artifact B - exp.php

The second PHP artifact is a database export script. It uses hard-coded database connection values, establishes a MySQL connection, enumerates tables, retrieves table structure and executes SELECT * for each table. The response is formatted as a downloadable SQL dump.

Observed Function	Security Consequence
Database credentials embedded	Uses secrets recovered from application configuration.
MySQL connection	Authenticates directly to production backend database.
SHOW TABLES	Enumerates all tables visible to the DB account.
SHOW CREATE TABLE	Retrieves database schema details.
SELECT *	Reads all rows/columns accessible to the DB account.
SQL dump output	Returns extracted data through the browser/HTTP response.

Credential Handling

The actual database password must remain redacted from external IR documents. The credential should be treated as compromised and rotated, even though the application vulnerability is patched.



06 Shopify Token Exposure Assessment

The database export behavior is broad rather than selective. Because exp.php retrieves every readable column from every accessible table, any Shopify access tokens stored in plaintext or otherwise readable form within those tables would be retrieved as part of the export.

Conditions Required for Token Exposure

- The token must be stored in a database table accessible to the compromised DB user.
- The token field must be readable by that database account.
- The value must be stored in plaintext or a recoverable/decryptable form from the compromised server context.
- The token must not have already been revoked or expired for subsequent API use.

Potential Follow-On Impact

- Unauthorized Shopify GraphQL or REST requests using valid tokens
- Access to merchant data permitted by granted app scopes
- Modification of application-managed merchant state
- Potential script-tag or storefront-related changes where the token scopes/API supported them
- Cross-tenant impact where multiple merchants' tokens were stored in the same backend data set

Evidence Boundary

The artifacts establish the capability to obtain readable Shopify tokens from the database. They do not, by themselves, prove which specific token was later used, which merchant APIs were accessed, or whether PII was exfiltrated. Those conclusions require Shopify/API and application log correlation.



07 Shopify Detection & Indicators of Compromise

Shopify reported unusual script-tag modification activity affecting Affected Shopify Application identified approximately 906 requests beginning on 16 August 2026 at 10:59:27 UTC.

The alert

Shopify Observation	Reported Detail
Detection start	2026-08-16T10:59:27.044+00:00
Approximate volume	906 requests
User agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:152.0) Gecko/20100101 Firefox/152.0
Affected app	Affected Shopify Application
Activity type	Script-tag modifications from potentially anomalous sources

Reported Source IP Addresses

185.220.100.247	45.84.107.128	185.220.100.243
185.246.190.136	193.189.100.203	194.32.107.14
185.181.60.204	185.220.101.18	185.220.101.29
193.189.100.194	31.25.11.143	185.220.101.104
192.42.116.14	5.255.98.151	159.195.137.114
192.42.116.57	171.25.193.234	45.66.35.29
185.220.101.184	204.8.96.165	204.8.96.101
171.25.193.25	192.42.116.118	82.118.248.205
171.25.193.82	107.189.8.226	

Additional IOCs to Search

- Requests to webadmin.php, exp.php, config.php or similarly named PHP utilities
- New or modified PHP files in upload directories
- Database connections originating from web/application servers at anomalous times
- Bulk SHOW TABLES / SHOW CREATE TABLE / SELECT activity
- Large SQL dump responses or unusual response sizes
- Access to token-related database tables
- Shopify API requests following suspected token-access timestamps
- File deletion or modification shortly after exfiltration activity



08 Incident Timeline

Date / Sequence	Event	Assessment
16 Aug 2026 10:59:27 UTC	Shopify begins observing anomalous script-tag modification requests; ~906 requests reported.	External detection
11 Aug 2026	Attacker exploits the executable PHP upload weakness and uploads webadmin.php to the server.	Reported file-upload date
11 Aug 2026 - after upload	Attacker accesses webadmin.php and begins browsing server-side application files.	Supported by artifact capability
Credential theft	Attacker locates and reads config.php, obtaining DB connection secrets.	Reported incident sequence
DB exfiltration	Attacker configures/executes exp.php using recovered credentials.	Supported by exp.php artifact
Database access	exp.php enumerates tables and exports readable records.	Confirmed script behavior
Token exposure	Readable Shopify token records become available in database dump.	Confirmed capability if tokens stored in accessible tables
Remediation	Vulnerable upload/configuration paths patched; malicious artifacts removed/should be removed.	Patched / remediated
Post-remediation	Credential rotation, token review and historical log correlation required for complete IR closure.	Residual assurance

Attribution Note

The Shopify-reported IPs and script-tag activity should be correlated with webadmin.php / exp.php timestamps and database/API logs before asserting that every reported Shopify request was generated through this exact compromise path.



09 Containment, Eradication & Recovery

Containment

- Disable executable PHP uploads across Announcement and Web Push components.
- Block PHP execution from all user-controlled upload directories.
- Remove webadmin.php, exp.php and any additional unauthorized server-side files.
- Block public access to config.php, .env, backups and secret-bearing files.
- Rotate compromised database credentials and invalidate old secrets.
- Review, revoke and rotate Shopify access tokens exposed in readable database records where applicable.

Eradication

- Search the server for unknown PHP, PHTML, PHAR and recently modified executable files.
- Review cron jobs, web-server configuration, .htaccess changes and persistence mechanisms.
- Review database users, grants, schema modifications and suspicious query history.
- Review Shopify GraphQL/REST activity and script-tag changes during the incident window.
- Preserve relevant logs and forensic copies before deleting attacker artifacts where possible.

Recovery / Hardening

- Store uploaded content outside the web root or in non-executable object storage.
- Use extension allowlists, MIME checks, magic-byte checks and generated filenames.
- Move application secrets to environment-based or dedicated secret management.
- Use least-privilege DB credentials and network isolation.
- Encrypt sensitive Shopify tokens at rest and minimize token scopes.
- Alert on token-table reads, executable uploads, unexpected PHP files and bulk database exports.



10 Root Cause Analysis & Remediation Validation

Root Cause	Security Failure	Remediation Position
Unrestricted executable upload	PHP content accepted and executed server-side	Patched
Executable upload location	User-controlled files could run as PHP	Patched / should remain disabled
Sensitive config access	Web-server process could read config.php containing DB secrets	Hardened / secrets should be isolated
Reusable DB credential	Recovered secret allowed backend authentication	Rotate and invalidate old credential
Broad DB read permissions	DB account could retrieve sensitive tables	Apply least privilege
Readable token storage	Merchant tokens could be returned in full-table export	Encrypt / restrict / rotate exposed tokens
Insufficient detection	Attacker tooling could operate until anomaly was detected	Improve file integrity, upload and DB monitoring

Closure Checklist

- ☐ PHP upload vulnerability retested and blocked
- ☐ PHP execution disabled in upload paths
- ☐ webadmin.php removed
- ☐ exp.php removed
- ☐ config.php no longer publicly exposed
- ☐ Database password rotated and old password invalid
- ☐ Shopify tokens reviewed/revoked/rotated as necessary
- ☐ Database and web logs reviewed
- ☐ Shopify anomaly window correlated
- ☐ No additional persistence found

Current Technical Status

The application vulnerabilities are reported as PATCHED / REMEDIATED. Full incident closure should only be declared after credential/token hygiene and historical compromise analysis are completed.



11 Final Assessment

This incident demonstrates a complete attacker progression from an application-layer upload weakness to server-side file access, secret theft, backend database extraction and potential cross-tenant Shopify token exposure. The key attacker pivot was `webadmin.php`: once executable on the server, it provided the capability to inspect application files and obtain `config.php`. The recovered database credentials were then operationalized through `exp.php`, which connected to the backend database and exported all readable table contents.

Where Shopify access tokens were stored in readable fields, the export mechanism would retrieve those values together with the rest of the table data. Any such tokens should therefore be treated as potentially compromised until revocation/rotation and Shopify API log review demonstrate otherwise.

Final Risk Statement

Original incident severity: CRITICAL. Current application vulnerability status: PATCHED / REMEDIATED. Residual incident-response work: verify database credential invalidation, review/rotate potentially exposed Shopify tokens, correlate Shopify/API logs with the anomalous activity window, and confirm that no persistence or additional unauthorized files remain.

Prepared By

Farchase Solutions Pvt. Ltd.

Incident Response & Application Security
www.farchase.com

Document Classification

CONFIDENTIAL - Security Incident Response / Compromise Assessment