ETX Signal-X

Daily Intelligence Digest

Monday, February 23, 2026

10	15
ARTICLES	TECHNIQUES

Article 1

How I Discovered a WordPress Vulnerability: Exposed Usernames & XML-RPC Exploitation

Source: securitycipher

Date: 12-Feb-2025

URL: https://medium.com/@iamshafayat/how-i-discovered-a-wordpress-vulnerability-exposed-usernames-xml-rpc-exploitation-b35b0ec63a54

T1 Username Enumeration via WordPress REST API

💉 Payload

```
https://[TARGET_WEBSITE]/wp-json/wp/v2/users
```

⚔️ Attack Chain

1. Access the REST API endpoint `/wp-json/wp/v2/users` on the target WordPress site.
2. Review the JSON response for user objects.
3. Extract the `slug` field, which reveals the actual WordPress username.

🔍 Discovery

Manual browsing to the REST API endpoint, observing that user information (including usernames) is exposed in the JSON response.

🔓 Bypass

No authentication required; endpoint is accessible to unauthenticated users unless specifically restricted.

🔗 Chain With

Enumerated usernames can be used as targets for subsequent brute force attacks, credential stuffing, or privilege escalation attempts.

T2 Brute Force Attack via XML-RPC system.multicall

💉 Payload

```
wpscan --url https://[TARGET_WEBSITE]/ -U '[EXPOSED_USERNAME]' -P 'rockyou.txt' --password-attack xmlrpc --random-user-agent
```

⚔️ Attack Chain

1. Confirm XML-RPC is enabled on the target by accessing `/xmlrpc.php` or using WPScan.
2. Use the previously enumerated username (`[EXPOSED_USERNAME]`).
3. Launch a brute force attack with WPScan, specifying the XML-RPC attack vector (`--password-attack xmlrpc`).
4. WPScan sends multiple password attempts in a single request via the `system.multicall` method.
5. If a valid password is found, gain unauthorized access to the WordPress admin panel.

🔍 Discovery

Used WPScan to detect XML-RPC accessibility; then leveraged the `system.multicall` method for efficient brute force attempts.

🔓 Bypass

Many security plugins do not monitor or block XML-RPC login attempts, making this attack stealthy and less likely to trigger lockouts or alerts.

🔗 Chain With

Combine with username enumeration (Technique 1) for targeted brute force. Successful compromise enables privilege escalation, data theft, and further lateral movement within the WordPress environment.

T3 XML-RPC Detection via WPScan

💉 Payload

```
wpscan --url https://[TARGET_WEBSITE]/ --random-user-agent
```

⚔️ Attack Chain

1. Run WPScan against the target WordPress site.
2. Analyze output for confirmation that XML-RPC is enabled (look for `/xmlrpc.php` accessibility).
3. Use this information to plan further attacks (e.g., brute force via XML-RPC).

🔍 Discovery

Automated scanning with WPScan, specifically checking for XML-RPC endpoint exposure.

🔓 Bypass

Randomized user agent (`--random-user-agent`) helps evade basic detection and rate limiting.

🔗 Chain With

Detection of XML-RPC enables selection of advanced brute force vectors and chaining with enumeration techniques for maximum impact.

Article 2

PortSwigger — LAB-5 Web shell upload via obfuscated file extension (Bug Bounty Prep)[by...

Source: securitycipher

Date: 17-Feb-2024

URL: https://dollarboysushil.medium.com/portswigger-lab-5-web-shell-upload-via-obfuscated-file-extension-bug-bounty-prep-by-5232dd3fb8fa

T1 Null Byte File Extension Obfuscation for Web Shell Upload

💉 Payload

```
shell.php%00.jpg
```

⚔️ Attack Chain

1. Prepare a PHP web shell file containing:
2. Attempt to upload the file directly as `shell.php` via the `/my-account/avatar` endpoint (POST request).
3. Observe that direct `.php` uploads are blocked by server-side extension filtering.
4. Modify the filename to `shell.php%00.jpg` in the intercepted POST request using BurpSuite Repeater.
5. Upload the file; server-side code interprets the filename up to the null byte, treating it as a `.php` file.
6. Access the uploaded file via the `/files/avatar/<filename>` endpoint (GET request), triggering PHP execution and reading `/home/carlos/secret`.

🔍 Discovery

Initial upload attempts with `.php` files were blocked. The researcher intercepted the upload request, analyzed server behavior, and tested null byte injection based on prior knowledge of legacy PHP file handling.

🔓 Bypass

By injecting `%00` (null byte) before the allowed extension, the server's sanitization logic is bypassed, causing it to treat the file as a PHP script despite the appended `.jpg` extension.

🔗 Chain With

Combine with further path traversal or chained file uploads for deeper access. Use the web shell for privilege escalation, lateral movement, or additional file reads.

Article 3

Uncovering Business Logic Vulnerabilities: A Real-World Case Study

Source: securitycipher

Date: 04-Jun-2025

URL: https://medium.com/@nishanthannisha008/uncovering-business-logic-vulnerabilities-a-real-world-case-study-18bfbdae46b3

T1 Price Manipulation via Client-Side Request Tampering

💉 Payload

```
POST /cart HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded

product_id=1234&price=0.13
```

⚔️ Attack Chain

1. Log in as a user with insufficient store credit for the target product.
2. Add the expensive product (e.g., "Lightweight 'l33t' Leather Jacket") to the cart.
3. Intercept the cart request using Burp Suite.
4. Modify the price field in the intercepted request to a much lower value (e.g., from 1337.00 to 0.13).
5. Forward the manipulated request to the server.
6. Confirm the modified price is reflected in the browser/cart UI.
7. Place the order and observe successful purchase at the altered price.

🔍 Discovery

Manual interception and manipulation of cart-related HTTP requests using Burp Suite. The researcher observed that client-side price changes were not validated server-side, enabling the exploit.

🔓 Bypass

The backend failed to validate the price against the database or enforce business rules, trusting client-supplied values. No server-side checks prevented price tampering.

🔗 Chain With

Combine with account creation automation to repeatedly purchase high-value items at minimal cost. Chain with coupon/discount logic flaws for further price reduction. Abuse for inventory depletion, denial of service, or financial fraud.

T2 Coupon Abuse via Account Creation (Business Logic Flaw)

💉 Payload

```
Coupon: SAVE50
```

⚔️ Attack Chain

1. Register multiple fake accounts using different email addresses.
2. Log in with each new account.
3. Apply the one-time-use coupon (e.g., SAVE50) intended for first-time users.
4. Complete purchases with the repeated discount.

🔍 Discovery

Analysis of business rules revealed lack of device fingerprinting, payment method history, or IP address validation. Manual testing confirmed coupon could be reused across multiple accounts.

🔓 Bypass

Server-side logic only checked account status ("new user") but did not enforce uniqueness across device, payment, or IP, allowing repeated coupon exploitation.

🔗 Chain With

Automate mass account creation for bulk coupon abuse. Combine with price manipulation for extreme purchase discounts. Use with referral/loyalty program logic flaws for additional financial gain.

Article 4

How I Discovered a Price Manipulation Bug While Buying a Simple Product

Source: securitycipher

Date: 08-Dec-2025

URL: https://xamiron.medium.com/how-i-discovered-a-price-manipulation-bug-while-buying-a-simple-product-d2584addbc74

T1 Client-Side Price Manipulation via Unvalidated Parameter

💉 Payload

```
{  "price_id": "test001",  "product_price": 50 // Manipulated from original 150}{"price_i
d": "test001", "product_price": 50
// Manipulated from original 150}
```

⚔️ Attack Chain

1. Visit the product details page and note the original price (e.g., 150 BDT).
2. Add the item to cart and proceed to checkout.
3. Intercept the checkout request using Burp Suite or similar proxy.
4. Identify the `product_price` field in the request payload.
5. Modify the `product_price` value from the original (150) to a lower value (e.g., 50).
6. Forward the modified request to the server.
7. Observe that the server accepts the manipulated price and forwards it to the payment gateway.
8. Complete the payment at the reduced price; transaction succeeds without backend validation.

🔍 Discovery

Careful observation of the checkout flow revealed the price was sourced from the client side. Inspection of network requests via Burp Suite exposed the unvalidated `product_price` parameter.

🔓 Bypass

No backend validation or integrity checks on price; server trusts client-supplied values without recalculation or signature verification.

🔗 Chain With

Combine with discount/coupon manipulation, multi-item cart tampering, or loyalty point abuse for amplified financial impact. Potential for chaining with IDOR on price_id or other business logic flaws affecting order totals.

Article 5

From Recon to Report: Exploiting SQL Injection in Hidden Parameter

Source: securitycipher

Date: 28-Aug-2025

URL: https://mugh33ra.medium.com/from-recon-to-report-exploiting-sql-injection-in-hidden-parameter-a2bce655e055

T1 Time-Based SQL Injection via XOR Payload in Hidden Parameter

💉 Payload

```
approval=true&inputCellPhone=555-666-0606&inputEmail=test@example.com&inputFirstName=John&inputLastName=Doe&inputSecurityAnswer=1&inputSecurityQuestion=1&subscription_list%5B%5D=6030'XOR(603*if(now()=sysdate(),sleep(6),0))XOR'Z
```

⚔️ Attack Chain

1. Recon large-scope targets using ASN lookups and search engines (Shodan, ZoomEye, FOFA) to collect IPs.
2. Identify origin IPs with no WAF/rate limits and outdated Apache version.
3. Locate a subscription form and intercept POST request in Burp Suite.
4. Identify hidden parameter `subscription_list[]` in the request body.
5. Manually test common SQLi payloads; escalate to scanner (Acunetix) for deeper testing.
6. Use Acunetix to detect time-based SQLi via XOR payloads.
7. Confirm vulnerability manually with crafted payload using `curl` and measure response delay.

🔍 Discovery

Manual interception and inspection of POST requests for hidden parameters during form submission. Automated scanning after manual payloads failed. Focused on origin IP due to lack of WAF/rate limits.

🔓 Bypass

Standard SQLi payloads failed; XOR-based payloads bypassed input validation and triggered time-based delay. No WAF or rate limiting allowed unrestricted testing.

🔗 Chain With

Hidden parameter exploitation can be chained with session hijacking (previous XSS on same target) for privilege escalation or lateral movement. Extraction of DB info enables further attacks (e.g., credential harvesting, pivoting).

T2 Extraction of Database and User Information via Ghauri Tool (XOR SQLi)

💉 Payload

```
ghauri -r req.txt --random-agent --banner --current-user --current-db
```

⚔️ Attack Chain

1. Prepare a request file (`req.txt`) with vulnerable POST data including XOR SQLi payload.
2. Use Ghauri tool (supports XOR-based SQLi) to automate exploitation.
3. Extract current database name, user, and banner info from the server.
4. Provide extracted information to triager for report validation.

🔍 Discovery

After confirming XOR-based SQLi, switched from sqlmap (which does not support XOR) to Ghauri for automated extraction. Used tool-specific features to retrieve DB metadata.

🔓 Bypass

sqlmap unable to exploit XOR-based SQLi; Ghauri's support for XOR payloads enabled full exploitation and data extraction.

🔗 Chain With

Extracted DB/user info can be used for privilege escalation, lateral movement, or chaining with other vulnerabilities (e.g., authentication bypass, data exfiltration).

Article 6

Lap 1: JWT authentication bypass via unverified signature

Source: securitycipher

Date: 08-Sep-2024

URL: https://abdelrahmansalaheldeen.medium.com/lap-1-jwt-authentication-bypass-via-unverified-signature-8e450a7b2f59

T1 JWT Authentication Bypass via Unverified Signature

💉 Payload

```
{
  "username": "administrator",
  ...
}
```

⚔️ Attack Chain

1. Log in with valid credentials (e.g., username: wiener, password: peter) to obtain a JWT token.
2. Intercept the JWT token and modify the payload, changing the "username" field to "administrator".
3. Send the modified token in a request to the admin panel endpoint.
4. Observe that the server grants access (HTTP 200) due to lack of signature verification.
5. Use the admin token to send a request to the user deletion endpoint (e.g., delete user "carlos").
6. Confirm successful action (HTTP 302 response).

🔍 Discovery

Manual inspection of JWT handling in HTTP history using Burp. Noticed that changing the username in the token altered access rights, indicating missing signature verification.

🔓 Bypass

The backend decodes JWT tokens without verifying their signature, allowing arbitrary manipulation of token claims (e.g., privilege escalation to administrator).

🔗 Chain With

Can be chained with any endpoint protected by JWT-based authorization. Potential for full account takeover, privilege escalation, and destructive actions (e.g., user deletion, configuration changes).

Article 7

How I Hacked Over 150k PII on a Program

Source: securitycipher

Date: 22-Oct-2024

URL: https://medium.com/@rootplinix/how-i-hacked-over-150k-pii-on-a-program-f58b8b141d4a

T1 Exploiting ZIP Backup Files on Redirecting Subdomains

💉 Payload

```
cat httpx_report.txt | grep 302 | awk '{print $1}' | anew 302_status.txt
cat 302_status.txt | nuclei -t ~/private/zip-backup.yaml -c 40 -project armin -rl 400 -v |
anew lowkey.txt
nuclei -u https://redacted.supersecret.com/ -t ~/private/zip-backup.yaml -v -debug
wget https://redacted.supersecret.com/uploads.zip
```

⚔️ Attack Chain

1. Enumerate ~1,000 subdomains of the target, noting status codes (200, 302, 403, 404).
2. Filter for subdomains returning 302 redirects (usually SSO/login pages) using:
3. Run Nuclei with a custom ZIP backup template against the list of 302 subdomains:
4. Identify accessible backup files (e.g., uploads.zip) exposed via redirecting subdomains.
5. Confirm the hit by scanning the specific endpoint:
6. Download the backup file directly to a cloud machine:
7. Extract and analyze contents (PII, Excel, CSV files, etc.).

🔍 Discovery

Focused on subdomains with 302 redirects, hypothesizing that legacy login pages may host overlooked backup files. Used Nuclei with custom ZIP backup templates, ignoring typical 200 OK pages. Leveraged historical context (target since 2016) to prioritize backup hunting.

🔓 Bypass

Targeted 302 redirect endpoints instead of standard content endpoints, bypassing rate limits and avoiding detection by focusing on less-monitored login pages.

🔗 Chain With

Combine with subdomain takeover checks (e.g., Gemfury hosting) for lateral access. Use extracted PII for further phishing, credential stuffing, or privilege escalation.

T2 Cloud-Based Download to Circumvent Local Bandwidth Limitations

💉 Payload

```
wget https://redacted.supersecret.com/uploads.zip
```

⚔️ Attack Chain

1. Identify large backup file (uploads.zip, ~1 GB) exposed on a target subdomain.
2. Instead of downloading locally (limited bandwidth), use Google Cloud Console to run `wget` and fetch the file directly to a cloud VM.
3. Unzip and analyze data remotely (Excel, CSV, PII records).

🔍 Discovery

Realized local bandwidth constraints would hinder exploitation; pivoted to cloud infrastructure for efficient data extraction.

🔓 Bypass

Circumvented local download limitations and potential ISP throttling by leveraging remote cloud resources.

🔗 Chain With

Use cloud-based analysis to automate parsing, search for additional sensitive files, or run further vulnerability scans on extracted data.

Article 8

0-Click Account Takeover Through a Simple Password Reset Parameter

Source: securitycipher

Date: 14-Aug-2025

URL: https://g0w6y.medium.com/0-click-account-takeover-through-a-simple-password-reset-parameter-482ad44019a2

T1 0-Click Account Takeover via Password Reset Parameter Manipulation

💉 Payload

```
https://redacted.com/reset-password?email=testshiozhy@gmail.com&token=sYrUIKzYt46
```

⚔️ Attack Chain

1. Request a password reset for your own account (e.g., shiohz0test@gmail.com).
2. Receive a reset link containing your email and a token: `https://redacted.com/reset-password?email=shiohz0test@gmail.com&token=O4AiLBPhZNvr`.
3. Request another reset link for your own account to get a fresh token.
4. Copy the reset link and replace the `email` parameter with the target account's email (e.g., testshiozhy@gmail.com).
5. Open the modified link in the browser.
6. The application allows you to reset the password for the target account without any prior reset request from the victim.

🔍 Discovery

Initial curiosity about user enumeration led to manual inspection of password reset links. Experimentation with parameter manipulation revealed the vulnerability.

🔓 Bypass

The application fails to validate whether the reset token is actually associated with the email provided in the URL, allowing a token generated for one account to be used to reset the password of any other account by swapping the email parameter.

🔗 Chain With

Can be combined with automated email enumeration to mass takeover accounts. Potential for privilege escalation if admin accounts are targeted. Exploit is 0-click and requires no victim interaction.

Article 9

How I Found The open redirect vulnerability?

Source: securitycipher

Date: 24-Feb-2025

URL: https://doordiefordream.medium.com/how-i-found-the-open-redirect-vulnerability-e0c3583b4e89

T1 Open Redirect via Login Return Parameter

💉 Payload

```
https://abc.example.com/login?returnTo=https://evil.com
```

⚔️ Attack Chain

1. Navigate to the login page at `https://abc.example.com/login`.
2. Append the `returnTo` parameter with a malicious external URL (e.g., `https://evil.com`).
3. Enter valid login credentials and submit the form.
4. Upon successful authentication, the application redirects the user to the URL specified in `returnTo`, enabling open redirect exploitation.

🔍 Discovery

The researcher systematically tested login-related parameters for open redirect vectors after exhausting other vulnerability types (authentication bypass, OTP bypass, SQLi). Prior experience with open redirects on login and 404 pages prompted targeted testing of the `returnTo` parameter.

🔓 Bypass

No explicit bypass logic described; the application directly honored the external URL in the `returnTo` parameter without validation.

🔗 Chain With

Open redirect can be chained with phishing campaigns, session hijacking, or OAuth token theft by redirecting users to attacker-controlled endpoints post-authentication.

Article 10

Forgot password link doesn’t expire after used.

Source: securitycipher

Date: 04-Oct-2025

URL: https://medium.com/@edahmed008/forgot-password-link-doesnt-expire-after-used-e55f0c5fe96f

T1 Replayable Password Reset Token

💉 Payload

```
https://example.com/reset-password?token=<REPLAYABLE_TOKEN>
```

⚔️ Attack Chain

1. Initiate a password reset for a known account via the "Forgot password" page.
2. Receive the password reset email and extract the reset link (token).
3. Use the link to set a new password (first use).
4. Reuse the exact same reset link in a new browser session or private tab.
5. Observe that the link still allows setting a new password, even after previous use.
6. Repeat step 4 as many times as desired; the token remains valid.

🔍 Discovery

Manual retesting of password reset flows. After successfully resetting a password, the researcher attempted to reuse the same reset link and found it was still valid.

🔓 Bypass

The application fails to invalidate password reset tokens after successful use, allowing unlimited replays. No session or token revocation logic is triggered post-reset.

🔗 Chain With

Combine with email compromise, referrer leakage, or shared browser history to escalate to full account takeover. Can be chained with session fixation or persistent authentication flaws for persistent access.