LinkAce STORED XSS in Username on Audit Page

William Hanjaya

Summary

A Stored Cross-Site Scripting (XSS) vulnerability has been identified on the **`/system/audit`** page. The application fails to properly sanitize the `username` field before it is rendered in the audit log. An authenticated attacker can set a malicious JavaScript payload as their username. When an action performed by this user is recorded (e.g., generate or revoke an API token), the payload is stored in the database. The script is then executed in the browser of any user, particularly administrators, who views the **`/system/audit`** page.

This can lead to session hijacking, unauthorized actions on behalf of the victim, or redirection to malicious websites.

Vulnerable Endpoint

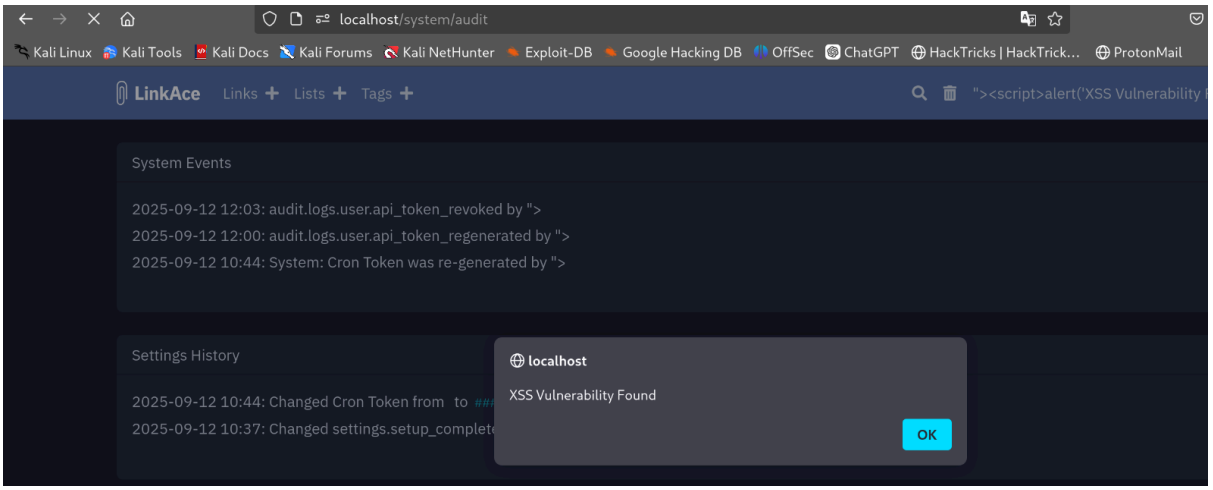
- `https://[your-domain.com]/system/audit`

Steps to Reproduce (Proof of Concept)

1. **Log in** to the application as a standard user.
2. Navigate to the user profile or account settings page where the **username can be modified**.
3. Change your username to the following JavaScript payload and save the changes:
   <script>alert('XSS Vulnerability Found')</script>

4. Perform an action that will be recorded in the system audit log. For this example, generate a new API token.
5. **Log out** of the attacker's account.

6. **Log in as an administrator** or another user who has permission to view the system audit logs.
7. Navigate to the audit log page: **`https://[your-domain.com]/system/audit`.**
8. **Observe the result**: An alert box with the message "XSS Vulnerability Found" will appear in the browser. This confirms that the stored JavaScript payload was executed.



Impact

Since this is a stored XSS vulnerability that affects a page likely viewed by privileged users, the impact is **High**. A successful exploit could lead to:

- **Session Hijacking:** The attacker can steal the session cookies or tokens of administrators, allowing them to impersonate the administrator and gain full control over their account.
- **Account Takeover:** The malicious script can be used to perform actions on behalf of the logged-in administrator, such as changing their password or email, creating new admin accounts, or deleting data.
- **Phishing:** Redirecting administrators to a fake login page to steal their credentials.
- **Data Exfiltration:** Capturing and sending sensitive information displayed on the audit page to an attacker-controlled server.
- **Website Defacement:** The script can alter the content of the `/system/audit` page.

Remediation

To fix this vulnerability, all user-supplied input must be treated as untrusted and properly encoded before being rendered on the page.

1. **Primary Fix: Output Encoding**
    - Apply context-aware output encoding to the username when it is displayed on the **`/system/audit` page**. Since the username is rendered within HTML, it should be **HTML entity encoded**.
    - For example, special characters should be converted to their HTML entity equivalents:
        - `<` becomes `<`
        - `>` becomes `>`
        - `"` becomes `&quot;`
        - `'` becomes `&#x27;`
    - Most modern web frameworks have built-in functions to handle this automatically (e.g., `htmlspecialchars()` in PHP, or default auto-escaping in frameworks like Django and Ruby on Rails). Ensure this feature is enabled and used correctly for this field.
2. **Defense-in-Depth: Content Security Policy (CSP)**
    - Implement a strict Content Security Policy (CSP) header. A well-configured CSP can act as a second layer of defense by restricting the sources from which scripts can be executed, mitigating the impact of any XSS vulnerabilities that might be missed.