

Sri Lanka Institute of Information Technology



Bug Bounty Report 05

WS Assignment
IE2062 – Web Security

IT23159730
W.H.M.S.R.Bandara

Vulnerability Title

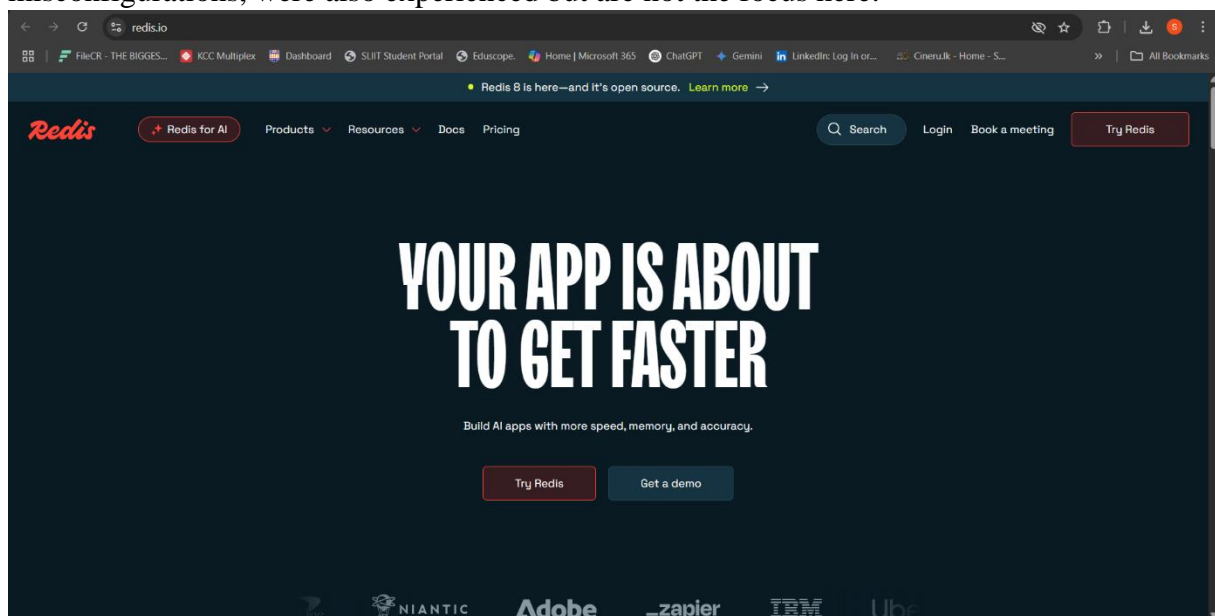
Hash Disclosure - BCrypt on <https://redis.io/>

Vulnerability Description

When a cryptographic hash like a BCrypt hash is exposed in a web application response, it is referred to as hash disclosure. BCrypt hash is often used to store passwords in a secure way, but if peered, they can be attacked by brute-force or dictionary attacks to get the original password back, mainly when the hash is leverage with a weak work factor or the password is weak.

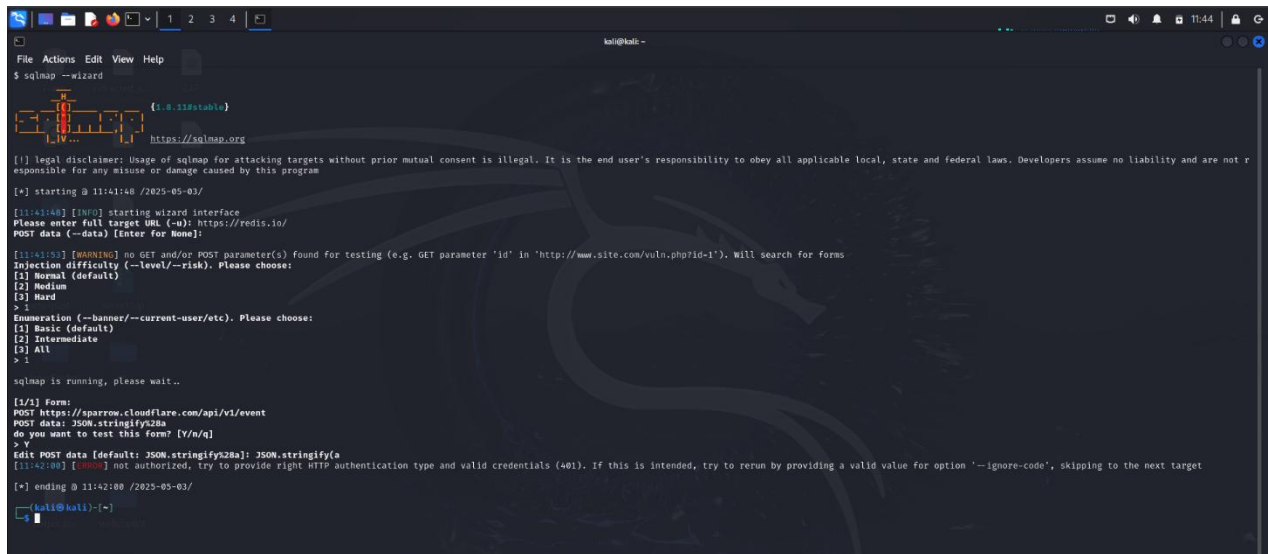
Scan of <https://redis.io/> overview using OWASP ZAP (2.16.1) io discovered hash disclosure vulnerability (CWE-497) on https://redis.io/learn/develop/java/redis-and-spring-course/lesson_4/ The Response

\$2a\$10\$spMjQ2FAGIUBT9chsXguGrBbjZmMiRvg5xaVwaQOdYLaJ2a This P1 high-severity vulnerability, found with high confidence, could allow attackers to attempt to crack the hash, leading to unauthorized access in the event the hash is similar to a user's password. This is an issue for a platform like Redis, which stores important database services and documentation. Other issues, such as missing anti-CSRF tokens and CSP misconfigurations, were also experienced but are not the focus here.



Detect and exploit SQL injection flaws

Detect and exploit SQL injection flaws



```
File Actions Edit View Help
$ sqlmap --wizard
[+] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 11:41:48 /2025-05-03/
[11:41:48] [Info] starting wizard interface
Please enter full target URL (-u): https://redis.io/
POST data (--data) [Enter for None]:
[11:41:51] [WARNING] no GET and/or POST parameter(s) found for testing (e.g. GET parameter 'id' in 'http://www.site.com/vuln.php?id=1'). Will search for forms
Injection difficulty (--level/--risk). Please choose:
[1] Normal (default)
[2] Medium
[3] Hard
> 1
Enumeration (--banner/--current-user/etc). Please choose:
[1] Basic (default)
[2] Intermediate
[3] All
> 1
sqlmap is running, please wait..
[11:41:51] Form:
POST https://sparrow.cloudflare.com/api/v1/event
POST data: 350W.stringify28a
do you want to test this form? [Y/n/q]
> Y
Edit POST data [default: 350W.stringify28a]: 350W.stringify(a
[11:42:00] [Error] not authorized, try to provide right HTTP authentication type and valid credentials (401). If this is intended, try to rerun by providing a valid value for option '--ignore-code', skipping to the next target
[*] ending @ 11:42:08 /2025-05-03/
kali@kali:~$
```

Affected Components

- **Domain:** redis.io
- **Service:** Web Application
- **Component:** HTTP Response Body
- **Affected URL:** https://redis.io/learn/develop/java/redis-and-spring-course/lesson_4/
- **Environment:** Production web server

Impact Assessment

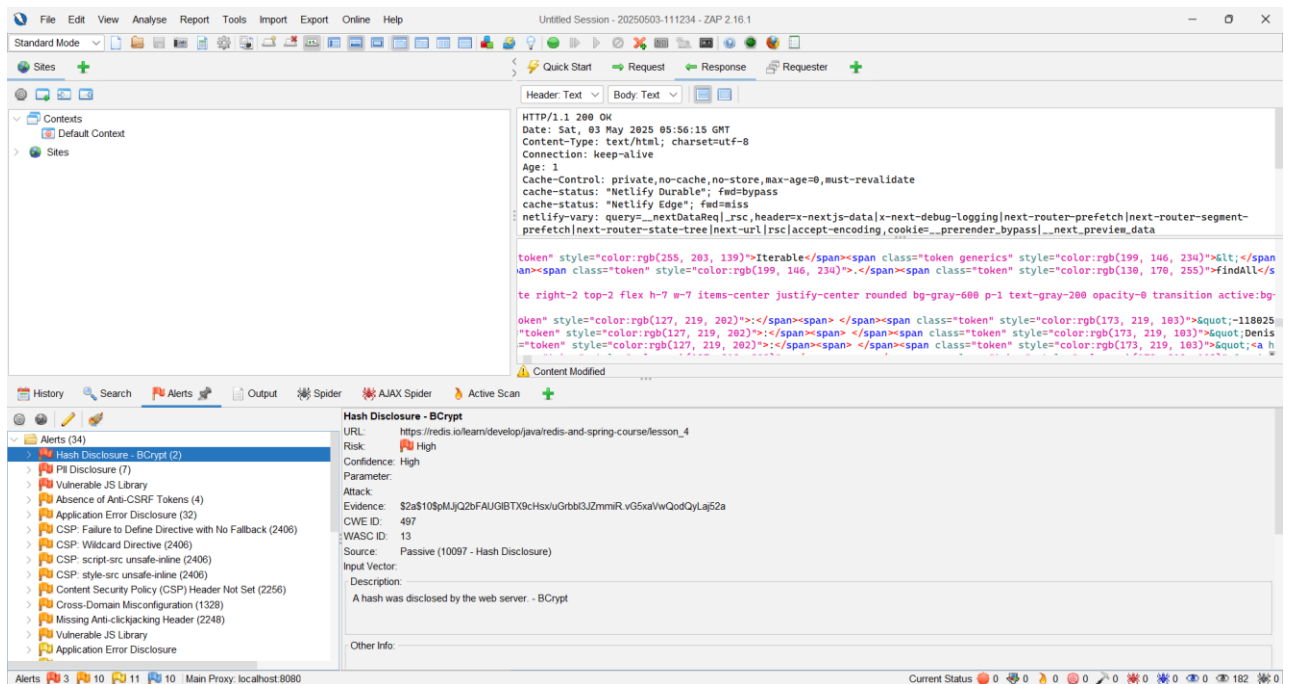
The hash disclosure poses these risks:

- **Password Cracking:** Attackers can brute-force the BCrypt hash to recover the original password, especially if the work factor is low or the password is weak.
- **Unauthorized Access:** Because the hash is for a user account, cracking it could lead to access to locked-down systems or data.
- **Business Impact:** A breach can lead to loss of confidence in Redis's security measures, damaging its reputation.
- **Reputation Risk:** Disclosure of this weakness could damage Redis's reputation in the community of developers.
- **Amplified Risk:** The other risks (e.g., lack of anti-CSRF tokens) widen the attack surface, making the severity more likely.

Steps to Reproduce

1. Configure OWASP ZAP
 - Launch OWASP ZAP and set the target to <https://redis.io/>.
 - Add the site to the context and ensure it's in scope.
2. Perform an Automated Scan
 - Spider the site with the AJAX Spider and detect dynamic content.
 - Start an automated scan by clicking the "Attack" button in the Automated Scan window.
 - Wait until the scan finishes.
3. Analyze Alerts
 - Alert: "Hash Disclosure - BCrypt" (CWE-497).
 - URL: https://redis.io/learn/develop/java/redis-and-spring-course/lesson_4.
 - Risk: High (P1), Confidence: High.
 - Source: Passive scan (10097 - Hash Disclosure).

Proof of Concept Screenshot



Proposed Mitigation or Fix

- **Remove Sensitive Data:** Remove the BCrypt hash from the response body and ensure no sensitive data is exposed in public-facing pages.
- **Use Strong Work Factors:** If BCrypt is used, ensure a high work factor (like 12 or higher) to make cracking harder.
- **Secure Storage:** Store hashes securely in a backend database, not in public responses.

Conclusion.

The OWASP ZAP scan discovered a high-severity (P1) hash disclosure weakness on <http://redis.io/>, exposing a BCrypt hash that can be broken to gain access to the system without legitimate permission. This weakness, along with weaknesses like lacking anti-CSRF tokens, increases the risk for this database service platform. Removing the disclosed hash and locking storage will decrease the risk. Periodic auditing is required to maintain security.