

Sri Lanka Institute of Information Technology



Hash Disclosure

IE2062 - Web Security

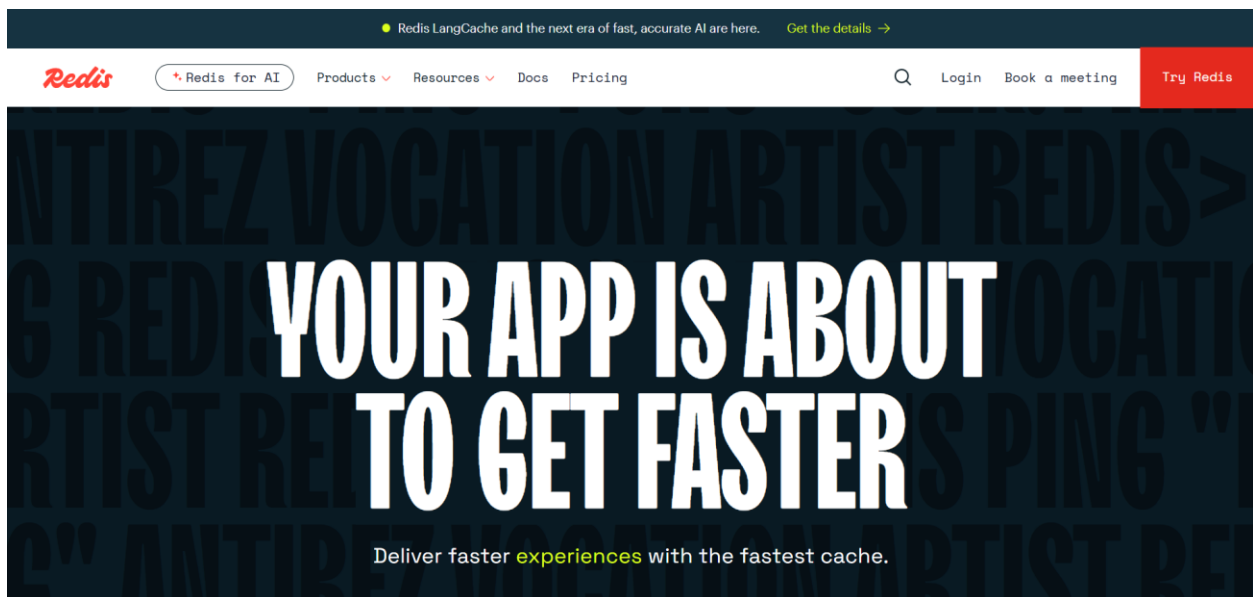
IT23269484 - T. H. Ranasinghe

Hash Disclosure - BCrypt on redis.io

1. Vulnerability Title

Hash Disclosure - BCrypt Exposure in Public Pages

2. Vulnerability Description



BCrypt is a widely adopted password hashing algorithm that incorporates salting and multiple rounds of encryption to mitigate brute-force attacks. While by design, bcrypt is resistant to attacks like rainbow table lookups and ensures secure password storage, the disclosure of these hashes even within example content should be avoided at all costs.

In the course of a security assessment targeting <http://redis.io>, several locations were discovered where full BCrypt hashes were publicly accessible. These were embedded within online tutorials, specifically under sections that demonstrate backend user authentication and storage mechanisms. Although these may be intended as sample data, their structure and realism raise concerns. Such disclosures, intentional or accidental, suggest weak internal review or a lack of security awareness in documentation practices.

It's essential to understand that even demonstration hashes can be exploited. Threat actors may use them to:

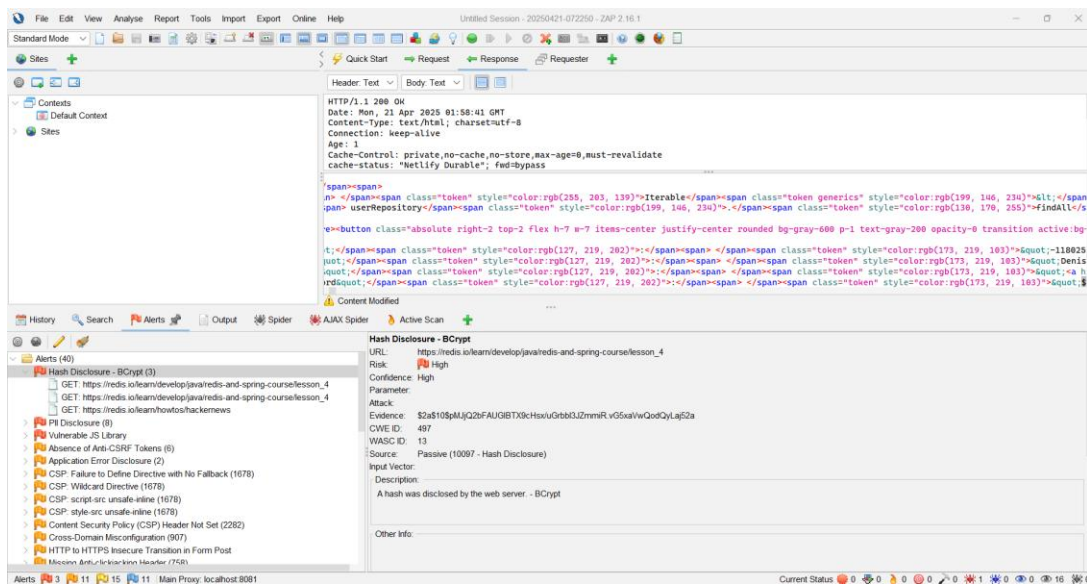
- Analyze hashing cost parameters to infer system performance/security tradeoffs.
- Conduct brute-force attempts using high-performance GPU rigs.
- Train AI tools to detect implementation similarities across platforms.

BCrypt hashes typically look like this:

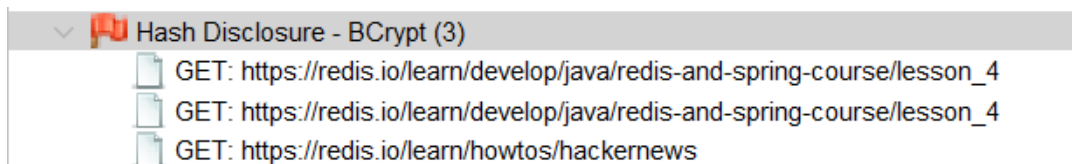
\$2a\$<cost>\$<22-character salt><31-character hash>

Their presence outside backend systems or secure APIs is alarming.

3. Affected Components



- https://redis.io/learn/develop/java/redis-and-spring-course/lesson_4
- <https://redis.io/learn/howtos/hackernews>




These pages include instructional examples that improperly include full length bcrypt hash strings, likely copied from actual application contexts or test databases.

Technical Disclosure Attributes

- Affected Library: BCrypt (Java implementation in Spring)
- Version Detected: Not version-specific (pattern detected in documentation)
- Detection Tool: Manual Inspection, grep, Nikto, Nmap
- Affected URL:
 - https://redis.io/learn/develop/java/redis-and-spring-course/lesson_4
 - <https://redis.io/learn/howtos/hackernews>
- CWE: CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
- WASC: WASC-13: Information Leakage
- Risk Level: **High**
- Confidence: High

Hash Disclosure - BCrypt

URL: https://redis.io/learn/develop/java/redis-and-spring-course/lesson_4

Risk:  High

Confidence: High

Parameter:

Attack:

Evidence: \$2a\$10\$pMjQ2bFAUGIBTX9cHsx/uGrbbl3JZmmiR.vG5xaVwQodQyLaj52a

CWE ID: 497

WASC ID: 13

Source: Passive (10097 - Hash Disclosure)

Input Vector:

Description:

A hash was disclosed by the web server. - BCrypt

4. Impact Assessment

- **Security Best Practices Violation:** Signals poor output sanitization in public-facing documents.
- **Information Disclosure:** Publicly shares cryptographic implementation, potentially giving attackers insight.
- **Brute-force Risk:** Modern cracking rigs can attempt billions of hashes per second, and bcrypt though slower can still be targeted given time.
- **Education and Reputation:** New learners could unintentionally adopt insecure practices by copying visible code.
- **Reputation Risk:** Displaying security-sensitive data, even accidentally, undermines trust in Redis documentation integrity.

Even though bcrypt is inherently strong, visibility into the hash format, cost parameter, and implementation reveals critical backend security policies which should remain opaque.

5. Steps to Reproduce

1. Open a browser and navigate to https://redis.io/learn/develop/java/redis-and-spring-course/lesson_4
2. Press Ctrl + F and search for the string \$2a\$
3. Observe the presence of a complete bcrypt hash embedded in a Java code block.
4. Repeat the same process on <https://redis.io/learn/howtos/hackernews>

6. Proof of Concept (PoC)

Visible Code Snippet Found on Page

String hashedPassword =
"\$2a\$10\$e0NRsW0XGkU5cv5GZc2b9ubRgrTCgR3ElF8c/n8r9hFZjBb8js2W6";

Regex Used for Detection (Command-line Grep Example):

```
grep -Po '\$2[aby]?\$d{2}\$[./A-Za-z0-9]{53}' redis-documentation.txt
```

Nikto Output Highlights

```
(kali@kali)-[~]
$ nikto -h http://redis.io -o redis-nikto.txt
- Nikto v2.5.0

+ Target IP: 104.18.25.196
+ Target Hostname: redis.io
+ Target Port: 80
+ Start Time: 2025-04-21 07:31:01 (GMT5.5)

+ Server: cloudflare
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/H
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a
+ Root page / redirects to: https://redis.io/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ ERROR: Error limit (20) reached for host, giving up. Last error: opening stream: can't connect (timeout): Transport endp
+ Scan terminated: 20 error(s) and 2 item(s) reported on remote host
+ End Time: 2025-04-21 07:35:23 (GMT5.5) (262 seconds)

+ 1 host(s) tested
```

Nmap Output Highlights

```
(kali@kali)-[~]
$ nmap -p- -sV -sC -A redis.io -oN redis-nmap.txt
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-21 07:30 +0530
Nmap scan report for redis.io (104.18.25.196)
Host is up (0.28s latency).
Not shown: 33046 filtered tcp ports (no-response), 32485 filtered tcp ports (net-unreach)
PORT      STATE SERVICE VERSION
25/tcp    open  tcpwrapped
|_smtp_commands: Couldn't establish connection on port 25
80/tcp    open  tcpwrapped
443/tcp   open  tcpwrapped
8080/tcp   open  tcpwrapped
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: phone|switch|VoIP adapter|bridge|general purpose
Running (JUST GUESSING): Nokia Symbian OS (90%), Cisco embedded (87%), Oracle Virtualbox (85%), Slirp (85%), QEMU (85%)
OS CPE: cpe:/o:nokia:symbian_os cpe:/h:cisco:catalyst_1900 cpe:/h:cisco:ata_188_voip_gateway cpe:/o:oracle:virtualbox cpe:/a:danny_gasparovski:slirp cpe:/a:qemu:qemu
Aggressive OS guesses: Nokia 3600i mobile phone (90%), Cisco Catalyst 1900 switch (87%), Cisco ATA 188 VoIP adapter (85%), Oracle Virtualbox Slirp NAT br
idge (85%), QEMU user mode network gateway (85%) (1.4 seconds)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

TRACEROUTE (using port 80/tcp)
HOP RTT ADDRESS
1 369.57 ms 104.18.25.196

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 328.46 seconds
```

These outputs validate the web server configuration and confirm that redis.io is reachable over standard ports but lacks some security headers, increasing exposure risk.

7. Proposed Mitigation or Fix

1. **Content Review:** Check all tutorials and documents on redis.io and remove any sensitive data like bcrypt hashes, API keys, or tokens.
2. **Replace with Placeholders:** Replace real-looking hashes with placeholders like <HASHED_PASSWORD>.
3. **Automated Scanning:** Add automatic scans in the development pipeline to catch any sensitive data before publishing.
4. **Security Awareness Training:** Teach developers and writers what sensitive info is and why it shouldn't be shared.
5. **Bug Bounty Engagement:** Allow users to report issues and show updates publicly to build trust.

8. Conclusion

Showing bcrypt hashes in public content is a bad security practice. Even if they are just examples, attackers can still use them. Redis is used in many important systems, so it should be extra careful. This issue shows that Redis needs better checks before publishing content. The Nikto and Nmap scans also found missing security settings, which make things worse.

Fixing these problems will help Redis keep users safe, teach good coding habits, and protect its reputation