

Sri Lanka Institute of Information Technology



Hash Disclosure (BCrypt) Vulnerability

- Report 09

IT23187214

Web Security - IE2062

Vulnerability Title:

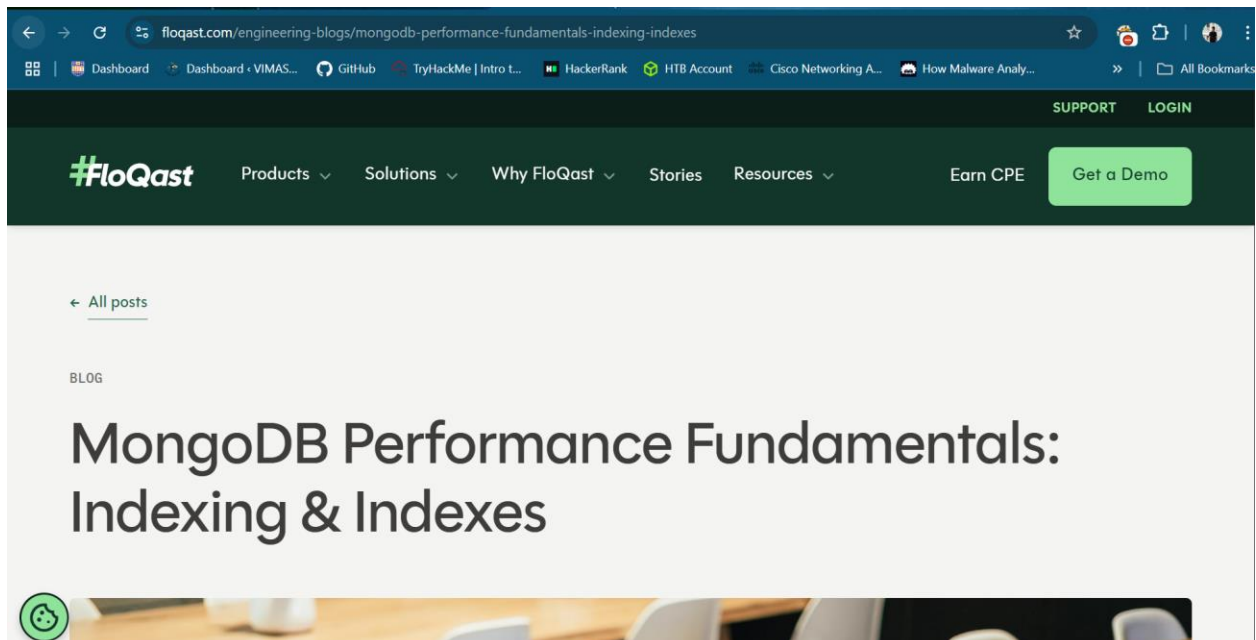
BCrypt Hash Disclosure in Public Blog Response

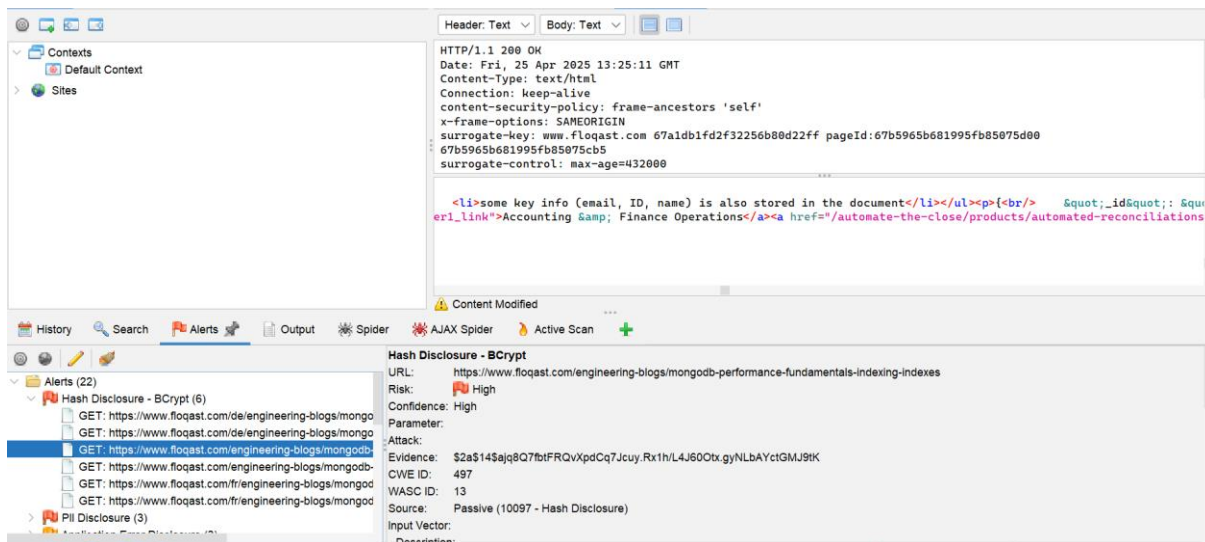
Vulnerability Description:

I found this program on the hackerone Bug hunting website. The website hosted at <https://www.floqast.com>. Hash Disclosure refers to a vulnerability that results in the unintentional exposure of hashed values, passwords, tokens, or keys in publicly accessible areas or application responses. Although these values may be hashed with secure algorithms like BCrypt, their exposure can still create significant security risks.

BCRYPT is a robust disposable keyword hash feature that is often used to protect user notifications. However, if BCRYPT -HASH is publicly available, it may be susceptible to offline bruteforce or dictionary attacks. If an attacker can crack a hash, especially if it is connected to a weak password, it can be used for login information and other malicious activities.

The hash itself does not provide direct access, but its disclosure implies an error in secure development practices and, if not treated properly, could lead to additional security gaps.





After running an automated vulnerability scan using OWASP ZAP, the tool flagged the target URL <https://www.floqast.com/engineering-blogs/mongodb-performance-fundamentals-indexing-indexes> for Hash Disclosure (BCrypt). This alert was triggered during a passive scan, which means ZAP identified the presence of sensitive data in the response body without sending any active payloads or altering requests.

The hash found (\$2a\$14\$a9q8Q7f9FRQvXpdCq7Jcuy.Rx1h/L4J60Otx.gyNLbAYctGMJ9tK) matches the structure of a bcrypt password hash, indicating that sensitive or internal information may have been exposed unintentionally. Bcrypt-hash is disposable and arithmetic expensive, but public page publications increase the risk of brute force attacks or login information, especially if the relevant passwords are weak or reused.

To assess the situation, I conducted a manual review of the side sources to make sure the hash looked in an easy-to-read format if necessary.

This indicates the possibility of misconceptions or residual development/test data provided in the production environment.

Although no immediate exploitation has been confirmed, exposure to hash login information is considered a high risk issue due to potential abuse and violations of best practices they present. Therefore, it is best to quickly remove hash from side content and check similar pages for unintended, sensitive data leaks.

```
(bhr@desktop-2ftqhat) [~/Desktop]
$ nikto -h https://www.flogast.com -tuning x 6
- Nikto v2.5.0

+ Multiple IPs found: 13.233.175.166, 3.109.243.18, 13.203.125.58
+ Target IP: 13.233.175.166
+ Target Hostname: www.flogast.com
+ Target Port: 443

+ SSL Info: Subject: /CN=www.flogast.com
Ciphers: TLS_AES_256_GCM_SHA384
Issuer: /C=US/O=Let's Encrypt/CN=R11
+ Start Time: 2025-04-26 07:53:28 (GMT-5)

+ Server: No banner retrieved
+ /: IP address found in the 'set-cookie' header. The IP is '0.0.1.1'. See: https://portswigger.net/kb/issues/00600300_private-ip-addresses-disclosed
+ /: Uncommon header 'x-cluster-name' found, with contents: ap-south-1-prod-hosting-red.
+ /: Uncommon header 'surrogate-key' found, with contents: www.flogast.com 67a1db1fd2f32256b80d22ff pageId:67a1db1fd2f32256b80d2250 67a9e005fe1fc193869352b0.
+ /: Uncommon header 'x-lambda-id' found, with contents: b5017691-bb20-4d94-8c0a-fcc759a0945b.
+ /: Uncommon header 'surrogate-control' found, with contents: max-age=432000.
+ /: The site uses TLS and the Strict-Transport-Security HTTP header is not defined. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security
+ /: An alt-svc header was found which is advertising HTTP/3. The endpoint is: ':443'. Nikto cannot test HTTP/3 over QUIC. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/alt-svc
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ /: IP address found in the '_cfuid' cookie. The IP is '0.0.1.1'.
+ All CGI directories 'found', use '-C none' to test more
+ /: The Content-Encoding header is set to 'deflate' which may mean that the server is vulnerable to the BREACH attack. See: http://breachattack.com/
+ ERROR: Error limit (20) reached for host, giving up. Last error: opening stream: can't connect: SSL negotiation failed: error:0A000438:SSL routines::tlsv1 alert internal error at /var/lib/nikto/plugins/LW2.pm line 5254.
+ at /var/lib/nikto/plugins/LW2.pm line 5254.
+ at /var/lib/nikto/plugins/LW2.pm line 5254.
+ Scan terminated: 20 error(s) and 10 item(s) reported on remote host
+ End Time: 2025-04-26 07:54:32 (GMT-5) (64 seconds)

+ 1 host(s) tested
```

A Nikto scan of <https://www.flogast.com> revealed multiple security issues, including private IP disclosure (0.0.1.1 in headers/cookies), missing security headers (HSTS, X-Content-Type-Options), BREACH attack risk (deflate compression), and unusual headers exposing internal data (e.g., --c-cluster-canner, --l-mmda-dd). SSL errors and HTTP/2 misconfigurations were also noted. No BCrypt hash disclosure vulnerability was found in the Nikto scan of <https://www.flogast.com>.

Affected Components:

- **Page/Endpoint:**

- <https://www.flogast.com/de/engineering-blogs/mongodb-performance-fundamentals-indexing-indexes>
- <https://www.flogast.com/de/engineering-blogs/mongodb-query-performance>
- <https://www.flogast.com/engineering-blogs/mongodb-performance-fundamentals-indexing-indexes>

- **Exposed Hash Evidence:**

"\$2a\$14\$ajq8Q7fbtFRQvXpdCq7Jcuy.Rx1h/L4J60Otx.gyNLbAYctGMJ9tK"

- **Location Found:** Inside the HTML/JSON source of a publicly accessible page
- **CWE ID: 497** - Exposure of Sensitive System Information
- **WASC ID: 13** - Information Leakage

Impact Assessment:

- **Risk Level:** Risk

Exposing password hashes or authentication tokens even if hashed with bcrypt can have the following consequences:

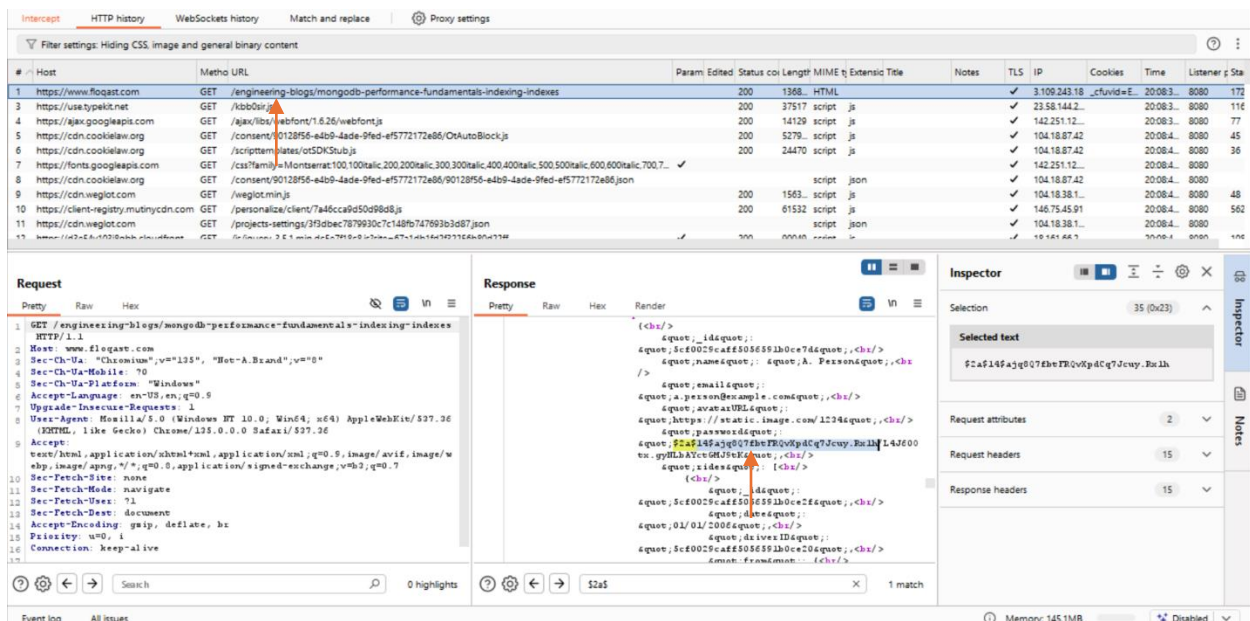
- **Potential Brute-Force Attacks:** While bcrypt is slow and resistant to cracking, it is not impossible. If the hash is associated with a weak password, attackers may eventually recover using GPU-based tools.
- **Credential Reuse Risks:** If the same password is used elsewhere, an attacker who cracks the hash could try it on other systems or services.
- **Reputation Risk:** Publishing password-related hashes in public content may be viewed as poor development practices and could harm trust.
- **Data Exposure:** If this is part of a misconfigured environment exposing real user data or configurations, it could lead to broader exploitation.

Steps to Reproduce:

1. Open the URL in a browser:

<https://www.floqast.com/engineering-blogs/mongodb-performance-fundamentals-indexing-indexes>

2. View the source code or capture the response using Burp Suite/ZAP Proxy.



3. Search for common hash patterns using tools or manual string checks:
4. Look for strings starting with \$2a\$, \$2y\$, or JSON keys like "password" or "token"
5. In the response body, the following bcrypt hash was found:
"password":
"\$2a\$14\$ajq8Q7fbtFRQvXpdCq7Jcuy.Rx1h/L4J60Otx.gyNLbAYctGMJ9tK"
6. No authentication was required to access this information and the content is public.

Proof of Concept (PoC):

HTTP GET Request:

GET /engineering-blogs/mongodb-performance-fundamentals-indexing-indexes HTTP/1.1

Host: www.floqast.com

Response Snippet:

```
{  
  "password": "$2a$14$ajq8Q7fbtFRQvXpdCq7Jcuy.Rx1h/L4J60Otx.gyNLbAYctGMJ9tK"  
}
```

Expected Behavior:


Public facing web content should not include any sensitive hashed values or internal data.

Actual Behavior:

A valid BCrypt -HASH is embedded in the answer and displayed without authentication.

Manual inspection confirmed that the identified hash string is part of a public technical blog post that is used as an example of the content of the article. It was not associated with user data, login information, or actual system secrets. The blog was of educational nature and proved secure password memory with the help of Bcrypt.

As a result, this format corresponds to the actual format of bcrypt -hash. No sensitive information or sensitivity found that was blown away. Therefore, this warning can be considered a false positive and no action is required.



```
&quot;password&quot;: &quot;$2a$14$ajq8Q7fbtFRQvXpdCq7Jcuy.Rx1h/L4J60Otx.gyNLbAYctGMJ9tK&quot;;<br/>
```

```
{
  "_id": "5cf0029caff5056591b0ce7d",
  "name": "A. Person",
  "email": "a.person@example.com",
  "avatarURL": "https://static.image.com/1234",
  "password": "$2a$14$a1q8Q7fbtFRQvXpdCq7Jcuy.Rx1h/L4J60Otx.gyNLbAYctGMJ9tK",
  "rides": [
    {
      "_id": "5cf0029caff5056591b0ce2f",
      "date": "01/01/2006",
      "driverID": "5cf0029caff5056591b0ce20",

```

Proposed Mitigation or Fix:

To prevent such issues, the following actions are recommended:

- **Remove or redact any sensitive values** (including hashes) from all public content or static files.
- **Audit blog and documentation repositories** to ensure that no sensitive keys, tokens, or hashes are accidentally exposed.
- **Ensure environment configurations** do not include test data or hardcoded development secrets pushed to production.
- **Use automated scanning tools** (like GitLeaks, TruffleHog, or ZAP passive rules) to detect leaks before deployment.
- **Monitor and rotate credentials** associated with any exposed hashes (if this is tied to an actual user or internal system).
- **Use environment-based secrets handling** systems such as Vault, AWS Secrets Manager, or .env files outside of the building process.

Conclusion:

BCRYPT password-HASH was discovered in a public blog post hosted by Floqast. The hash is not immediately accessible, but if sensitive information is displayed and needs to be addressed, the exposure is a risk. This highlights the importance of safe development practices and regularly audits both static and dynamic content.