IASS Penetration Testing Report

Prepared by: Mohamed Elgendy

Date: June 13, 2025

Executive Summary

Multiple critical security vulnerabilities were identified in the IASS project, including Denial of Service (DoS) attacks, unrestricted file access, and missing security headers. These issues pose significant risks, such as data breaches, system downtime, or unauthorized access. Immediate action is recommended to address these vulnerabilities and implement the proposed security measures.

Introduction

The purpose of this penetration test was to identify security vulnerabilities in the IASS web application and its infrastructure, providing actionable recommendations for remediation. The assessment focused on weaknesses that could compromise the system's integrity or data confidentiality.

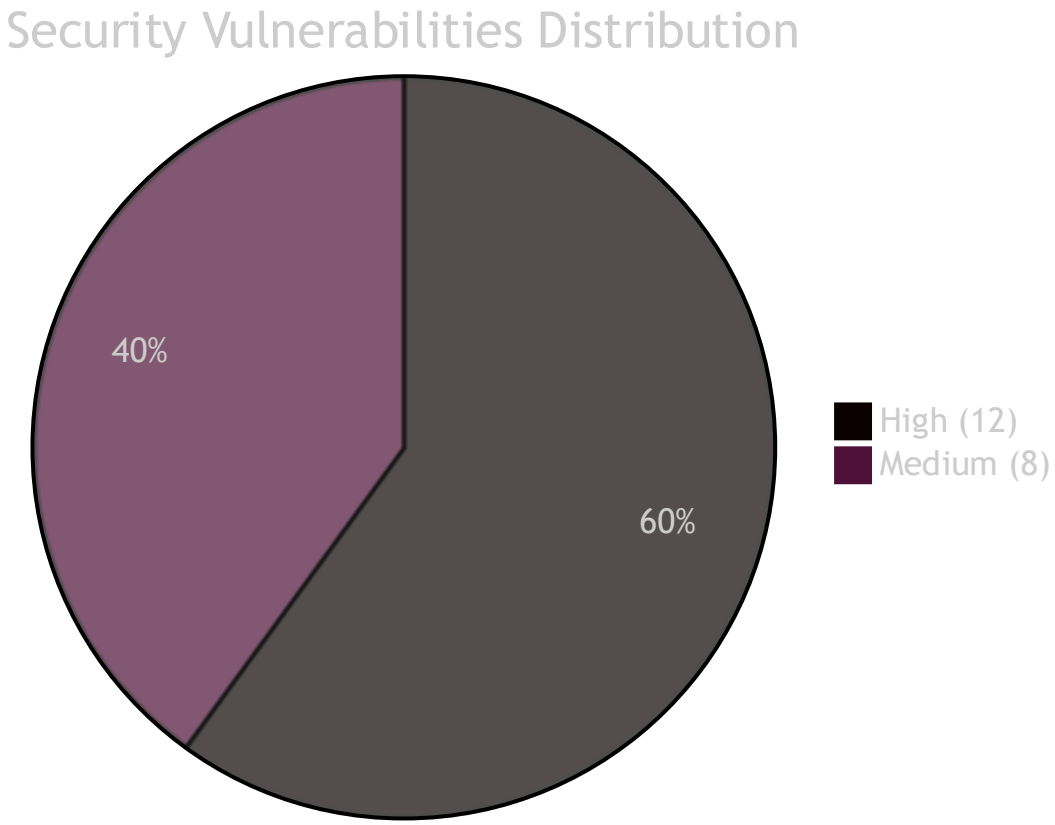
Methodology

The penetration test was conducted using a combination of automated tools and manual techniques:

- **Programs**: Burp Suite, OWASP ZAP, Composer Audit, Snyk
- **Tools**: Nmap, SQLi, XXSER, cURL, Nuclei, Dirb, Nikto, ffuf, dirsearch, mrLegacy, sunzy, sublister, w00fwaf, ...etc.
- **Techniques**: Manual hunting, Vulnerability scanning, manual code review, and exploitation testing
- **Framework**: OWASP Testing Guide, GitHub

Findings Summary

Severity	Number	Vulnerability
High	12	XXE, Env Manipulation, Code Injection
Medium	8	XSS, Open Redirect, File Inclusion



#Num	Vulnerability	Severity	Priority
1	Denial of Service (DoS) - DDoS Attack	High	1
2	Unrestricted File Access & Authentication Issues	High	2
3	Missing Security Headers - Information Disclosure	High	3
4	CVE-2024-40075: Remote Code Execution via Deserialization	High	4
5	CVE-2023-3824 - PHP Phar Stack Buffer Overflow	High	5
6	Missing Rate Limiting on Password Reset Endpoint	High	6
7	Weak Password Policy	High	7
8	Broken Session Invalidation after Password Change	High	8
9	Username Enumeration	High	9
10	Sensitive File Exposure	Medium	10

Detailed Findings

Misconfiguration Vulnerabilities

Denial of Service (DoS) – DDoS Attack

Severity:

High

Description:

I was able to send repeated or large requests to the site, **causing the site to:**

- Start not responding
- Crash

This means:

- No Rate Limiting
- No Web Application Firewall (WAF)
- Likely very weak server settings like Apache/PHP

Tool & Command:

“hping3 -S --flood -V https://dtu.life”

🛠️ **Recommended Solution:**

1. Rate Limiting (Laravel Middleware)

```
Route::middleware('throttle:60,1')-
>group(function () {
    Route::post('/login', 'AuthController@login');
});
```

⟶ 60 requests per minute per IP

2. Enable ModSecurity or WAF on Apache

- Monitor unusual request patterns and enable OWASP CRS rules

3. Use Cloudflare or CDN with Rate Limiting + DDoS Protection

4. Use **Captcha** Or **reCAPTCHA** (powerful method to stop **DoS** & **DDoS attack**)

5. Set up Log Monitoring

- Monitor /var/log/apache2/access.log or /storage/log/laravel.log for request counts

6. Limit Request Sizes

- For example, in php.ini

```
post_max_size = 2M

upload_max_filesize = 2M

max_input_time = 30
```

7. Use **cache header**

Unnecessary Service Exposure – FTP Port Open (OWASP A05:2021)

Severity:

Medium

Description:

FTP port open without use of it

🛠️ **Recommended Solution**:

Close port 21

Tools & Command:

“nmap -p- -sV dtu.life”

Insecure Data Storage

Severity:

High

Description:

It was discovered that PII data (such as names, email addresses, ID numbers, or financial data) is stored in a MySQL database as plain text without any encryption. This means that anyone who gains unauthorized access to the database (whether through hacking or internal error) can read this data directly.

Recommendation:

Implement encryption at the table level:
Use MySQL's encryption feature with AES_ENCRYPT to store sensitive data.

Information Disclosure

Username Enumeration

Severity:

Medium

Description:

Entering a correct email + wrong password → No feedback

Entering a wrong email → Feedback appears (e.g.: "Email not found")

🛠 **Recommended Solution:**

Unify login messages.

Unrestricted File Access & Authentication Issues

Severity:

High

Description:

Without any user identity verification, anyone is allowed to download files from this endpoint:

- ⇨ https://dtu.life/uploads/leave/background-border1664887697.png
- ⇨ https://dtu.life/dashboard/sample/students.xlsx
- ⇨ https://dtu.life/uploads/user/WhatsApp_Image_2025_05_30_at_02.19.52_8f0051f9_1749747713.png

🛠️ Recommended Solution:

1. Protect routes behind authentication:
   Use Middleware in Laravel like:

```
Route::middleware('auth')->group(function () {
    Route::get('/dashboard/sample/{file}', 'DownloadController@secureFile');
});
```

2. Move files to storage/app/private instead of public/:
In Laravel, use:

```
Storage::disk('local')
-->download('private/students.xlsx');
```

→ Accessible only via a Route that checks the session.

2. Verify file ownership:

Before showing any file:

```
if ($user->id !== $file->owner_id) {
    abort(403);
}
```

3. Log all file access attempts:
   To detect whether files are being used for malicious purposes.

Sensitive File Exposure

Severity:

Medium

Description:

1. Identifying Frameworks and Versions:
   Laravel Mix ^4.0.7
   Vue.js ^2.5.17
   Bootstrap ^4.1.0
   jQuery ^3.2
   → This means you are using outdated technologies with known vulnerabilities (XSS, Prototype Pollution, etc.).
2. Identifying Development Tools:
   webpack, sass-loader, cross-env
   → These can help an attacker craft payload that bypass bundler or sanitizer protections.
3. No dependencies found in the production section
   → This suggests that the main work is done on the backend or that the project is weakly prepared.

🛠️ **Recommended Solution:**

Prevent access to backend files directly via HTTP:

Add a rule in .htaccess or Nginx configuration:

```
<FilesMatch "(package\.json|composer\.lock|\.env)">
    Deny from all
</FilesMatch>
```

Move sensitive files outside the public/ folder:

Keep package.json, .env, etc., in the project root, not in public_html or htdocs.

Missing Security Headers – Information Disclosure via headers

Severity:

High

Tools & Command:

"curl -I https://dtu.life"

Description:

Header	Note	Possible Impact
x-powered-by: PHP/8.2.28	❌ Reveals PHP version	Fingerprinting → RCE exploitation
server: LiteSpeed	❌ Reveals server type	Fingerprinting / Exploit mapping
platform: hostinger	❌ Reveals hosting provider	Targeting host-specific exploits
panel: hpanel	❌ Reveals control panel type	Panel attack surface
content-security-policy: upgrade-insecure-requests	Good, but very simple and weak	Does not effectively protect against XSS
set-cookie: XSRF-TOKEN	Indicates CSRF protection via Laravel	Good, but lacks SameSite policy
set-cookie: university_management_system_session	Missing SameSite=Strict, which is a vulnerability	CSRF risk / Cross-site leakage
x-content-type-options	❌ Missing	MIME sniffing → JS injection
x-frame-options	❌ Missing	Possible clickjacking
referrer-policy	❌ Missing	Data leakage via URLs
permissions-policy	❌ Missing	No browser feature restrictions
strict-transport-security	❌ Missing	Site does not enforce HTTPS always

Tools:

Burpsuite

Current Vulnerabilities and Risks:

1. Information Disclosure via headers
   (x-powered-by, server, platform, panel)
   ⟶ Facilitates fingerprinting and selecting suitable exploits.
2. Weak CSP
   Only upgrades HTTP to HTTPS, but does not prevent XSS or JS Injection.
3. Session Cookies without SameSite
   ⟶ Exposed to CSRF attacks, even if XSRF token is present.
4. Missing security headers like X-Frame-Options, X-Content-Type-Options, etc.
   ⟶ Exposes you to attacks such as:
   - Clickjacking
   - MIME-sniffing
   - Referrer leakage
   - Browser feature abuse

Recommended Headers (Ready to Copy/Paste):

```
# In .htaccess or Apache server configuration:

Header always set X-Frame-Options "DENY"

Header always set X-Content-Type-Options "nosniff"

Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"

Header always set Referrer-Policy "no-referrer"

Header always set Permissions-Policy "geolocation=(), microphone=(), camera=()"

Header always set X-Permitted-Cross-Domain-Policies "none"

Header always set Content-Security-Policy script-src 'self' https://trusted.cdn.com;
```

Laravel Middleware (برمجيًا):

```
public function handle($request, Closure $next)
{
    $response = $next($request);


    return $response
        ->header('X-Frame-Options', 'DENY')
        ->header('X-Content-Type-Options', 'nosniff')
        ->header('Strict-Transport-Security', 'max-age=31536000')
        ->header('Content-Security-Policy', "default-src 'self'")
        ->header('Referrer-Policy', 'no-referrer');
}
```

Vulnerable or Outdated Components

CVE-2024-52301: Environment Manipulation via Query String

Severity:

High

Impact:

Allows attackers to override environment variables via query strings.

Recommendation:

Upgrade to Laravel 9.52.17+ and disable register_argc_argv in php.ini.

Source:

ZAProxy + Snyk

CVE-2024-40075: Remote Code Execution via Deserialization

Severity:

High

Impact:

Allows remote code execution through Laravel's encryption component.

Recommendation:

Upgrade Laravel to 9.52.16 or later.

Source:

ZAProxy + Snyk

CVE-2024-51736: Symfony Process Hijack (Windows Only)

Severity:

Medium

Impact:

Malicious cmd.exe in working directory may be executed.

Recommendation:

Upgrade symfony/process to 6.4.14 or later.

References:

- GitHub Commit
    - Symfony Advisory
- Vulnerability Advisory
- More about this vulnerability

CVE-2023-3824 – PHP Phar Stack Buffer Overflow

Severity:

High

Description:

When loading a PHAR archive containing large entries for the directory option, a stack buffer overflow occurs due to insufficient length checking, which may allow memory corruption or even Remote Code Execution (RCE).

Source:

ZAProxy + Snyk

Impact:

If libraries that handle PHAR archives are called, the vulnerability can be exploited to execute malicious code on the server. This is a highly critical issue, potentially causing RCE and service crashes.

Solution:

Upgrade to at least the following PHP versions:

- PHP 8.0.30 : PHP 8.2.8

Multiple CVEs in PhpSpreadsheet (via maatwebsite/excel)

Severity:

High

Impact:

XXE and XSS via user-uploaded Excel files.

Recommendation:

Upgrade maatwebsite/excel to ^3.2 and PhpSpreadsheet to ^3.7.

References:

- GitHub Commit
- More about this vulnerability

Arbitrary Argument Injection in Laravel

Severity:

High

High Impact:

Injection of arguments via query string to manipulate environment.

Recommendation:

Upgrade Laravel to 11.31.0.

References

- GitHub Commit
- More about this vulnerability

Remote File Inclusion in nesbot/carbon

Severity:

Medium

Impact:

Includes external files via setLocale method.

Recommendation:

Upgrade to Carbon 2.72.6 or later.

References:

- GitHub Commit
- More about this vulnerability

File Validation Bypass (Improper Neutralization)

Severity:

Medium

Package:

Laravel/Framework (<11.44.1)

Description:

Wildcard validation flaws in Validator.php allow bypassing file type checks, enabling malicious uploads.

Impact: Code execution via uploaded files.

Solution: Upgrade to Laravel 11.44.1.

Use MIME type checks:

```
$request->validate(['file' => 'mimes:jpeg,png']);.
```

References:

- GitHub Commit
- GitHub Commit
- GitHub Commit
- More about this vulnerability

Rate Limiting

Missing Rate Limiting on Password Reset Endpoint

Severity:

High

Tools:

Burpsuite

Description:

The /admin/password/reset => endpoint lacks any protection against the number of attempts, allowing an attacker to:

- Rapidly try multiple email addresses (Email Enumeration)
- Launch a brute force attack on the verification code or reset token (if a numeric code or link is used)

🛠️ **Proposed Security Solutions:**

- Enable Rate Limiting:
  In Laravel, use:

```
Route::post('/password/email')->middleware('throttle:5,1');
```

→ Limits to 5 attempts per minute.

- Add CAPTCHA / reCAPTCHA v3:
  Especially on the email input page and the reset page.
- Do Not Reveal Email Validity:
  Always respond with:
  "If this email exists, a reset link has been sent."
- Expire Tokens Aggressively:
  Tokens should expire within 10 minutes and be deleted immediately after use.
- Log All Attempts:
  To detect abuse or bot activity.

Weak Password Policy

(OWASP A07:2021 – Identification and Authentication Failures)

Severity:

High

Tools:

Burpsuite

Description:

The system allows creating or changing a password using only one condition (8 characters), without requiring:

- Uppercase letters (A–Z)
- Lowercase letters (a–z)
- Numbers (0–9)
- Special characters (@#%!&*...)

Nor does it prevent common passwords (12345678, password123, etc.)

🛠️ Recommended Solution:

1. Implement a password complexity policy

Example in Laravel:

```
$request->validate([
  'password' => [
    'required',
    'string',
    'min:12',
    'regex:/[a-z]/',      // lowercase letter
    'regex:/[A-Z]/',      // uppercase letter
    'regex:/[0-9]/',      // number
    'regex:/[@$!%*#?&]/', // special character
  ],
]);
```

Broken Session Invalidation after Password Change

(OWASP A07:2021 – Identification and Authentication Failures)

Severity:

High

Description:

When a user changes their password from one window or device, the sessions that were open on other windows or devices remain active as if the change never happened.

🛠️ Recommended Solution:

1. Invalidate all sessions after password change
   When updating the password:

   - PHP:

     ```
     auth()->logoutOtherDevices($request->password);
     ```

   - PHP:

     ```
     Auth::logoutOtherDevices($request->password);

     Session::flush(); // optional for clearing local session
     ```

Open Redirect in symfony/http-foundation

Severity:

Medium

Impact:

Redirect users to arbitrary external sites.

References:

- GitHub Commit
- Symfony Advisory
- Vulnerability Advisory
- More about this vulnerability

Recommendation:

Sanitize redirect URLs and upgrade symfony/http-foundation.

1. Disable Dangerous PHP Settings:

   register_argc_argv = Off

   allow_url_include = Off

2. File Upload Protections:

```
$request->validate([
  'file' => 'required|mimes:pdf,xlsx|max:5120',
]);
```

3. HTTP Security Headers:

```
Header set X-Frame-Options "DENY"
Header set Content-Security-Policy "default-src 'self'"
Header set Strict-Transport-Security "max-age=31536000"
```

4. Process Isolation:

```
use Symfony\Component\Process\Process;
$process = new Process(['command']);
$process->setWorkingDirectory('/secure/path'); // Windows only
```

Conclusion

The IASS penetration test revealed 20 significant security vulnerabilities, comprising 12high-severity and 8 medium-severity issues. Critical findings include Denial of Service (DoS) attacks, unrestricted file access, remote code execution (RCE) via outdated components, and missing security headers, which collectively expose the system to risks such as data breaches, server downtime, and unauthorized access. The absence of rate limiting, weak password policies, and misconfigured security headers further exacerbate the attack surface.

Immediate remediation is critical to safeguard the application and its users. Prioritized actions include:

- Implementing rate limiting and DDoS protection to mitigate DoS risks.
- Securing file access with authentication and authorization checks.
- Updating vulnerable dependencies (e.g., Laravel, PhpSpreadsheet, nesbot/carbon) to patched versions.
- Enforcing robust security headers (e.g., CSP, HSTS) to prevent client-side attacks.

By addressing these vulnerabilities within a 30-day timeframe, the IASS project can significantly enhance its

security posture, ensuring resilience against potential attacks and maintaining user trust. Regular security assessments and dependency monitoring are recommended to prevent future vulnerabilities.

References

The following resources were consulted during the penetration test and provide additional context for the identified vulnerabilities and recommended solutions:

- OWASP Top 10:2021. *Web Application Security Risks*. Available at: https://owasp.org/www-project-top-ten/
- Laravel Documentation. *Rate Limiting*. Available at: https://laravel.com/docs/11.x/rate-limiting
- OWASP Secure Headers Project. *Best Practices for HTTP Security Headers*. Available at: https://owasp.org/www-project-secure-headers/
- National Vulnerability Database (NVD). *CVE-2024-52301, CVE-2024-40075, CVE-2024-51736, CVE-2023-3824*. Available at: https://nvd.nist.gov/
- PHP Documentation. *Security Best Practices*. Available at: https://www.php.net/manual/en/security.php

- Cloudflare Documentation. *DDoS Protection and Rate Limiting*. Available at: https://developers.cloudflare.com/ddos-protection/
- OWASP Testing Guide. *Penetration Testing Methodology*. Available at: https://owasp.org/www-project-web-security-testing-guide/

These references provide authoritative guidance for implementing the recommended remediations and maintaining a secure application environment.

Table of Contents