MyESI Automated Compliance & Vulnerability Assessment

Project: myesi-risk-service

Generated: 2025-11-23 10:00 UTC **User:** 1

1. Executive Summary

• Compliance Score: **60.0%**

• Average Risk Score: **10.00**

• Total Vulnerabilities: **4**

Standards Compliance:

Standard	Status
ISO_27001	Partially
NIST_SP_800_53	Needs Review
OWASP	Needs Review

2. Compliance Controls Overview

critical — *(General)* — Score: **50.0%**

1 Conduct a Vulnerability Assessment Identify and assess the systems or applications that may be impacted by the unknown vulnerability.

2 Update Software and Systems Ensure that all software and systems are up to date with the latest security patches. Example command for Linux systems: bash sudo apt-get update && sudo apt-get upgrade

3 Implement a Security Policy Review Review and update security policies to ensure they address potential risks associated with unknown vulnerabilities. Include guidelines for regular vulnerability scanning and incident response procedures.

3. Vulnerability Findings

#	Component	Version	Vuln ID	Severity	Fix / Recommendation
1	**pydantic**	2.12.3	MAL-2025-4867	**Critical**	### Summary The vulnerability identified as MAL-2025-4867 in the `pydantic` library poses a critical risk that could allow an attacker to exploit validation errors, potentially leading to arbitrary code execution or data leakage. Immediate action is required to mitigate this risk. ### Remediation Steps 1. **Upgrade the `pydantic` library** Update to the latest version where the vulnerability has been patched. 2. **Example Command** ```bash pip install --upgrade pydantic ``` 3. **Configuration Guidance** Review and adjust any custom validation rules to ensure they do not rely on potentially vulnerable features of the library. Implement strict data validation practices to minimize the risk of exploitation.
2	**python-dotenv**	1.2.1	MAL-2025-48037	**Critical**	### Summary The vulnerability identified as MAL-2025-48037 in the `python-dotenv` package poses a critical risk, potentially allowing unauthorized access to sensitive environment variables. This can lead to data breaches or application misconfigurations. ### Remediation Steps 1. **Upgrade the `python-dotenv` package** to the latest version to mitigate the vulnerability. 2. Run the following command to update the package: ```bash pip install --upgrade python-dotenv ``` 3. **Review and update your environment variable management policies** to ensure that sensitive information is adequately protected and not hard-coded in your application. Consider using secure vaults or encryption for sensitive data storage.

3	**typing-extensions**	4.15.0	MAL-2025-47895	**Critical**	### Summary The vulnerability identified as MAL-2025-47895 in the `typing-extensions` package poses a critical risk, potentially allowing attackers to execute arbitrary code or gain unauthorized access to sensitive information. Immediate action is required to mitigate this risk. ### Remediation Steps 1. **Update the `typing-extensions` Package** Ensure that your project uses the latest version of `typing-extensions` that contains the necessary security patches. 2. **Example Command** Run the following command to update the package via pip: ```bash pip install --upgrade typing-extensions ``` 3. **Policy Guidance** Establish a policy for regular dependency audits and updates. Implement automated tools to scan for vulnerabilities in third-party libraries and ensure timely updates are applied.
4	**uvicorn**	0.38.0	MAL-2025-4901	**Critical**	### Summary The vulnerability identified as MAL-2025-4901 in the Uvicorn server poses a critical risk, potentially allowing unauthorized access or execution of arbitrary code. Immediate remediation is essential to protect your applications and data from exploitation. ### Remediation Steps 1. **Upgrade Uvicorn to the Latest Version** Update Uvicorn to the latest stable version that addresses the vulnerability. Ensure you are using a version that is not affected by MAL-2025-4901. 2. **Example Command** Run the following command to upgrade Uvicorn using pip: ```bash pip install --upgrade uvicorn ``` 3. **Review Configuration and Security Policies** - Ensure that Uvicorn is not exposed to the public internet without proper security measures in place (e.g., firewalls, VPNs). - Implement strict access controls and authentication mechanisms for any services running on Uvicorn. - Regularly review and update your dependency management policies to include security scanning for vulnerabilities.

4. Code Findings (Static Analysis)

1. dockerfile.security.missing-user.missing-user (Error / MEDIUM)

/app/tmp/repos/semgrep-20251122112320/Dockerfile

By not specifying a USER, a program in the container may run as 'root'. This is a security hazard. If an attacker can control a process running as root, they may have control over the container. Ensure that the last USER in a Dockerfile is a USER other than 'root'.

```
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8004", "--reload"]
```

https://owasp.org/Top10/A04_2021-Insecure_Design

Suggested Fix:

Summary The absence of a specified user in the Dockerfile can lead to security vulnerabilities, as containers may run with elevated privileges, increasing the risk of unauthorized access and potential exploitation. Remediation Steps Specify a Non-Root User: Modify the Dockerfile to include a non-root user to run the application. Example Command: Add the following lines to your Dockerfile: dockerfile RUN useradd -ms /bin/bash appuser USER appuser Policy Guidance: Ensure that all Docker images are built with a non-root user by implementing a policy that mandates user specification in Dockerfiles across your organization. Regularly review Dockerfiles for compliance.

Generated by MyESI on 2025-11-23 10:00 UTC — User: 1