# PRIORITIZING VULNERABILITIES USING CVSS V3 SCORES

DION KONTOPODIAS

04/11/2023

# INTRODUCTION

**The Imperative of Vulnerability Prioritization**:
- Identifying and prioritizing vulnerabilities is key to safeguarding against escalating security threats.
- Organizations today grapple with an ever-increasing complexity and volume of security vulnerabilities.

**Introduction to CVSS v3**:
- A crucial tool for consistent and accurate assessment of vulnerability severity.
- Detailed insights into CVSS v3 to be provided in the following sections.

**Challenges in Effective Prioritization**:
- Navigating through a myriad of threats with limited resources poses significant challenges.
- Emphasizing the necessity for a structured method to manage and prioritize vulnerabilities effectively.

**Presentation Focus**:
- Proposing a methodology to leverage CVSS v3 scores for prioritizing vulnerabilities in CI/CD pipelines.
- Tailoring this strategy to fortify application security in dynamic and evolving environments.

**Relevance to Current Cybersecurity Practices**:
- This approach is in sync with current best practices and standards in cybersecurity.
- Demonstrating the real-world applicability and benefits of this method in cybersecurity management.

# UNDERSTANDING CVSS V3

**Overview of CVSS**:

• The Common Vulnerability Scoring System (CVSS) is a standardized framework for rating the severity of security vulnerabilities.

• CVSS is globally recognized and used across industries for consistent vulnerability assessment.

**Key Components of CVSS v3:**

• CVSS v3 includes **Base**, **Temporal**, and **Environmental** scores, offering a comprehensive severity assessment.

• The v3 update provides more precise scoring through detailed metric evaluation.

**Base Score Metrics:**

• **Core assessment:** The Base Score measures the intrinsic qualities of a vulnerability that are constant over time and user environments.

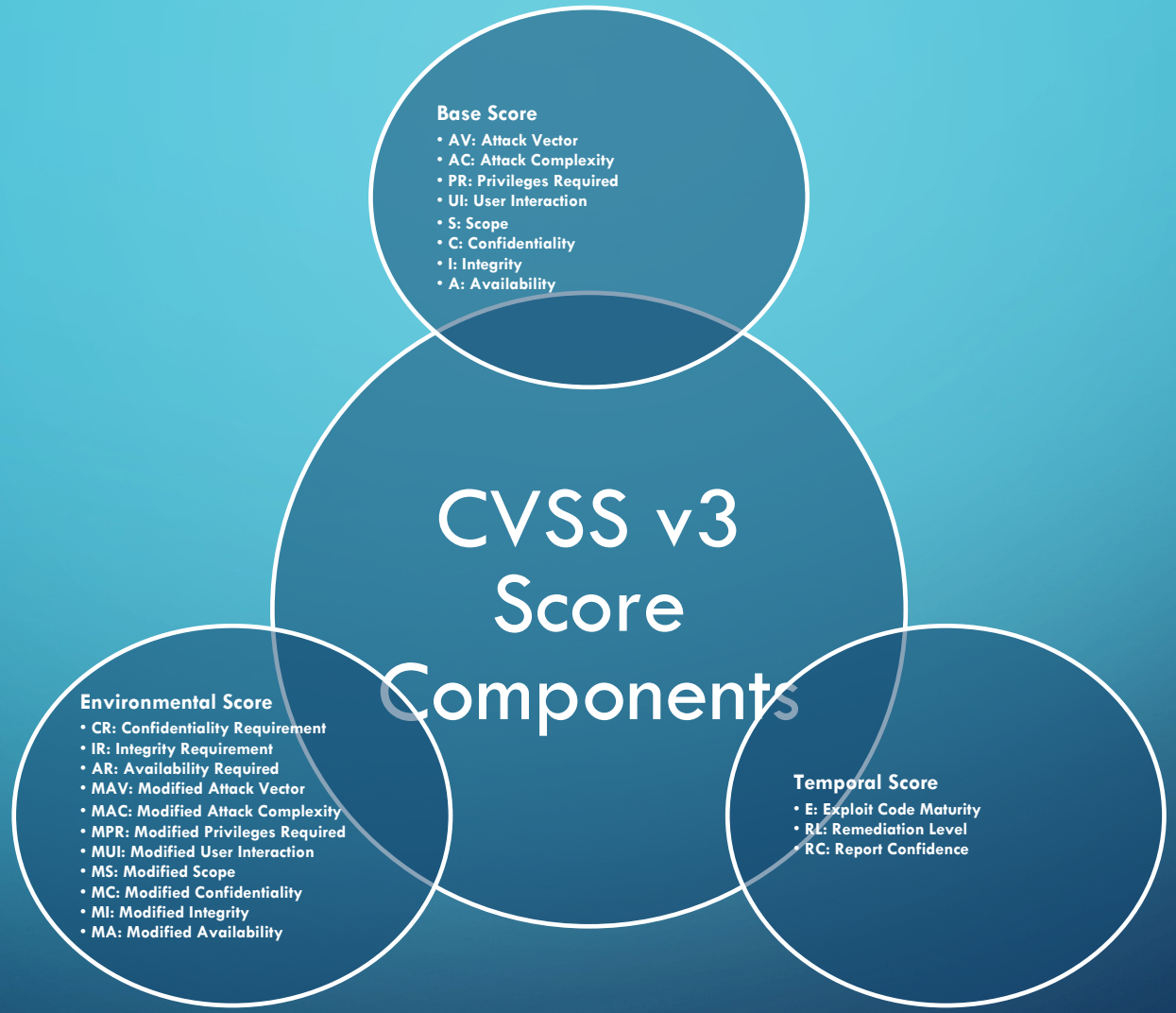
• **Metric groups:** It comprises Exploitability metrics (like attack vector, complexity) and Impact metrics (confidentiality, integrity, availability).

**Scoring System Nuances:**

• **Quantitative and qualitative:** CVSS provides both numerical scores (0-10) and qualitative severity ratings (e.g., Low, Medium, High, Critical).

• **Dynamic assessment:** Encourages re-evaluation of vulnerabilities as new information becomes available or environmental factors change.

# CVSS v3 Score Components



# VULNERABILITY IDENTIFICATION METHODS

**Software Composition Analysis (SCA)**:
- SCA tools identify vulnerabilities in open-source components and dependencies.
- SCA tools like Snyk and Veracode include CVSS v3 scores in their reports for prioritization.

**Static Application Security Testing (SAST)**:
- SAST tools analyze source code to detect security vulnerabilities.
- Tools such as Veracode and Checkmarx assign or reference CVSS v3 scores to identified vulnerabilities.

**Dynamic Application Security Testing (DAST)**:
- DAST tools test applications in their running state to find vulnerabilities.
- DAST solutions like OWASP ZAP include CVSS v3 scores in reports for real-time severity assessment.

**Infrastructure as Code (IaC) Scanners**:
- IaC scanners assess configuration and deployment scripts for security issues.
- CVSS v3 Scores: Tools like Prisma Cloud use CVSS v3 scores to highlight critical configuration vulnerabilities.

**And many more technologies utilize CVSS v3 such as:**
- Container and Orchestration Scanning Tools.
- Interactive Application Security Testing Tools (IAST).
- Threat Intelligence Platforms…

# INTEGRATING CVSS V3 IN CI/CD PIPELINES

**Introduction to CI/CD and CVSS v3 Integration**:
- CI/CD pipelines automate steps in software delivery, ensuring rapid and reliable deployment.
- Utilizing CVSS scores within CI/CD pipelines enhances vulnerability management and security decision-making.

**Incorporating SCA and SAST Tools**:
- **Early stages:** Integrate SCA and SAST tools in the development and commit phases to identify vulnerabilities.
- Use CVSS scores to prioritize fixes for detected vulnerabilities before progressing to later stages.

**DAST and IAST in Testing and Staging**:
- Incorporate DAST and IAST tools during testing and staging phases for dynamic analysis.
- Prioritize remediation of vulnerabilities based on CVSS scores, focusing on high and critical severity issues.

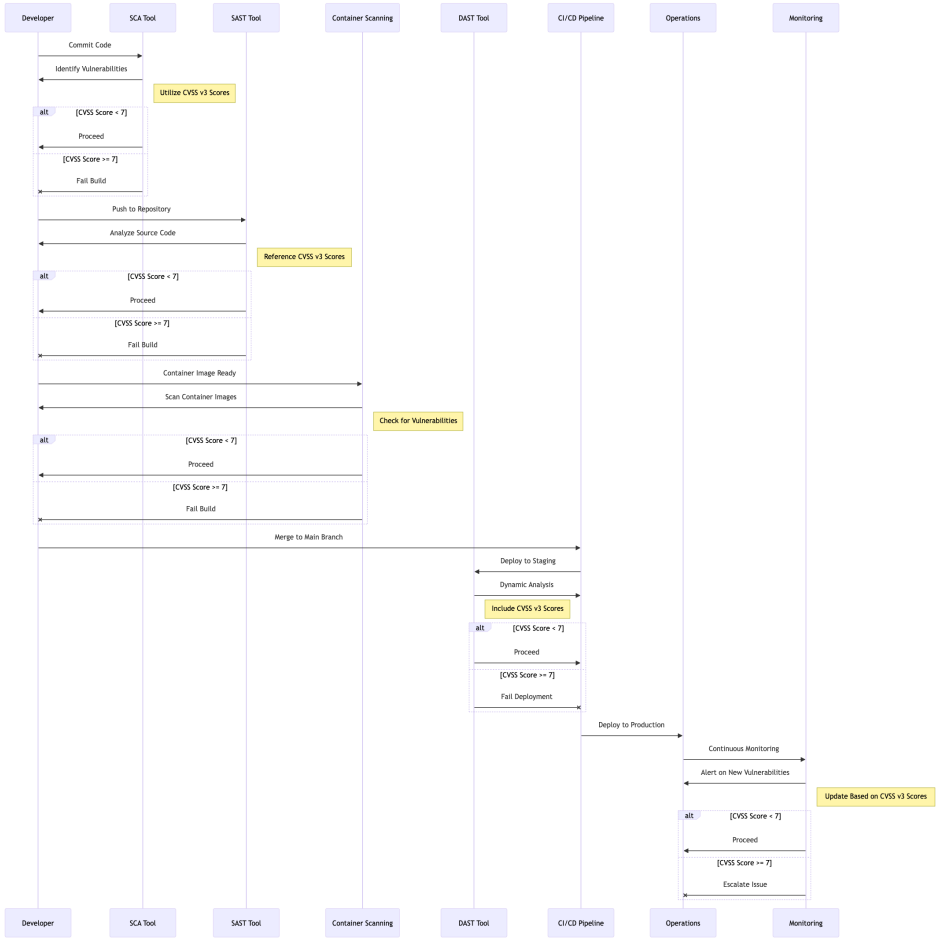
**Automated Security Gates**:
- Set up automated security gates at critical points in the pipeline, based on CVSS score thresholds.
- Configure gates to **pass**, **warn**, or **fail builds** depending on the severity of vulnerabilities identified.

**Feedback Loop for Continuous Improvement**:
- Implement a feedback loop to developers and security teams about the nature and severity of vulnerabilities.
- Use CVSS scores to communicate the urgency and impact of vulnerabilities, guiding timely remediation.

**Monitoring and Threat Intelligence Integration**:
- Continuous monitoring in production with integration of threat intelligence for real-time alerts.
- Update and adjust security measures in the pipeline based on evolving threat landscapes and CVSS score changes.

# EXAMPLE SDLC

(PLEASE ZOOM IN)



# PRIORITIZATION STRATEGY

**CVSS Score Utilization**:
- Use CVSS scores to initially assess and rank vulnerabilities based on severity.
- Consider both the Base Score and Temporal/Environmental metrics for a holistic assessment.

**Score Thresholds for Prioritization**:
- **Defining thresholds:** Establish score thresholds (e.g., 0-3.9 as Low, 4-6.9 as Medium, 7-8.9 as High, 9-10 as Critical) for categorization.
- Prioritize remediation efforts starting with **Critical** vulnerabilities, followed by **High**, **Medium**, and **Low**.

**Contextual Factors in Prioritization**:
- Prioritize vulnerabilities higher if they affect sensitive or critical systems.
- Consider the ease of exploitation and potential impact on confidentiality, integrity, and availability.

**Temporal Aspects in Decision Making**:
- Stay updated with changes in exploit code maturity, remediation levels, and report confidence.
- Reassess and reprioritize vulnerabilities as new information or patches become available.

**Customizing Prioritization for the Organization**:
- Adjust prioritization strategy based on specific business risks and operational requirements.
- Use internal threat intelligence and historical data to refine prioritization further.

# HANDLING DIFFERENT VULNERABILITY TYPES

**Categorization of Vulnerabilities**:

- Recognize vulnerabilities originating from various sources like SCA, SAST, DAST, and container scanning.
- Classify vulnerabilities into categories such as **code flaws**, **configuration errors**, and **security misconfigurations**.

**Integrating CVSS Scores for Varied Types**:

- **Unified scoring:** Apply CVSS scoring **uniformly** across different vulnerability types for consistent assessment.
- Adapt CVSS score interpretation based on the nature and context of the vulnerability.

**Prioritization Based on Vulnerability Type**:

- Prioritize vulnerabilities that have a direct impact on **critical assets** or **sensitive data**.
- Give higher priority to vulnerabilities that are easily exploitable or have known exploits in the wild.

**Customized Response Strategies**:

- Develop specific remediation strategies for different types of vulnerabilities, considering their unique characteristics.
- Allocate resources effectively by focusing on high-impact and high-probability vulnerabilities first.

**Incorporating Environmental Factors**:

- Use the Environmental Score in CVSS to adjust the severity based on your organization’s specific context.
- Consider the criticality of the affected asset in the organization to refine prioritization.

# CASE STUDIES

# CASE STUDY A: CVE-2014-3566 (POODLE VULNERABILITY)

**Overview of CVE-2014-3566**:

• Affects SSL v3.0, enabling data decryption via a man-in-the-middle attack.

**CVSS v3 Score Analysis**:

• Base Score: 3.4 (Low Severity)
• Vector String: CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:C/C:L/I:N/A:N
• Decoding the Vector:
  - Attack Vector (AV): Network (N) - exploitable remotely.
  - Attack Complexity (AC): High (H) - specialized conditions required.
  - Privileges Required (PR): None (N) - no privileges needed.
  - User Interaction (UI): Required (R) - victim must take some action.
  - Scope (S): Changed (C) - impacts beyond the vulnerable component.
  - Impact Metrics: Confidentiality (C) Low, Integrity (I) None, Availability (A) None.

**Implications in Security**:

• Risk of decrypting secure cookies, significant in specific scenarios.
• High attack complexity and user interaction requirement reduce overall threat.

**Mitigation and Response**:

• Disabling SSL v3.0 in systems and using more secure protocols like TLS.
• Regularly updating and patching systems to prevent exploitation of such vulnerabilities.

# CASE STUDY B: CVE-2023-34362

**Overview of CVE-2023-34362**:

• A critical vulnerability affecting [MOVEit Transfer web application], potentially allowing remote code execution without user interaction.

**CVSS v3 Score Analysis**:

• Base Score: 9.8 (Critical Severity)
• Vector String: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
• Decoding the Vector:
  - Attack Vector (AV): Network (N) - exploitable remotely without physical access.
  - Attack Complexity (AC): Low (L) - attack can be performed with minimal effort.
  - Privileges Required (PR): None (N) - exploitable without any privileges.
  - User Interaction (UI): None (N) - no user interaction needed for exploitation.
  - Scope (S): Unchanged (U) - does not affect resources beyond the scope.
  - Impact Metrics: High impact on Confidentiality, Integrity, and Availability.

**Implications in Security**:

• High severity due to ease of exploitation and potential for widespread damage.
• Critical for systems where confidentiality, integrity and availability are paramount.

**Mitigation and Response**:

• Immediate patching required due to the high risk of exploitation.
• Enhanced monitoring and security measures for affected systems.

# CASE STUDY C: IMPLEMENTING CVSS V3 GUARD GATES IN GITHUB ACTIONS

**Overview of GitHub Actions for Security**:

• GitHub Actions automate workflows, enabling CI/CD directly within GitHub repositories.

• Can be used to integrate security scanning tools that utilize CVSS v3 scores.

**Setting Up a Guard Gate**:

• Implement a workflow step that uses a security scanning tool, like Snyk.

• Analyze scan results to check if any vulnerabilities exceed a predefined CVSS v3 score threshold.

**Key Points**:

• Customize the severity threshold based on CVSS scores.

• Integrate with different scanning tools as per project requirements.

• Automate response actions like failing the build or sending alerts for high-risk vulnerabilities. (Even auto-remediation , e.g. automated pull request with the latest secure version of a dependency that has a critical vulnerability)

```yaml
name: Security Scan
on: [push]
jobs:
  security_check:
    runs-on: ubuntu-latest
    steps:
    - name: Checkout code
      uses: actions/checkout@v2
    - name: Run Snyk to check for vulnerabilities
      uses: snyk/actions/scan@master
      with:
        command: test
    - name: Fail build if high severity vulnerabilities are found
      run: |
        if [ $(snyk test --severity-threshold=high | grep 'High severity vulnerability found') ]; then
          exit 1
        fi
```

*The above code snippet is a very simplistic implementation of Snyk scan within a GitHub Actions workflow

# SUMMARY & BEST PRACTICES

**Key Takeaways**:

• CVSS v3 provides a standardized framework for assessing the severity of security vulnerabilities.
• Effective prioritization of vulnerabilities is crucial, with a focus on CVSS scores, environmental context, and temporal factors.
• Integrating vulnerability scanning tools in CI/CD pipelines enhances security and automates the identification of potential risks.

**Best Practices in Vulnerability Management**:

• Regularly update and patch systems to mitigate known vulnerabilities.
• Tailor vulnerability prioritization to organizational needs, considering the specific environment and asset criticality.
• Continuously monitor for new threats and reassess existing vulnerabilities, adapting strategies as needed.

**CI/CD Pipeline Security**:

• Implement automated security gates in CI/CD pipelines based on CVSS score thresholds.
• Utilize a combination of SCA, SAST, DAST, IaC and container scanning tools for comprehensive coverage.
• Ensure continuous feedback and improvement in security practices within the development lifecycle.

**Final Thoughts**:

• CVSS v3 stands as a cornerstone in cybersecurity strategy, providing a structured framework for uniform vulnerability assessment. Its integration into CI/CD pipelines elevates our security posture, enabling proactive, consistent, and efficient management of threats throughout the software development lifecycle.

# THANK YOU