Advanced Skill Certificate in DevOps Security Patching

Secure Software Development Practices

Secure Software Development Practices are crucial in DevOps environments to ensure that applications are built with security in mind from the ground up. In this course, we will explore key terms and vocabulary related to Secure Software Development Practices to help you understand the importance of security in the software development lifecycle.

1. **Security Patching**:
Security patching is the process of applying updates to software systems to address vulnerabilities or security flaws. These patches are released by software vendors or developers to protect systems from potential cyber threats. It is essential to regularly update software with the latest security patches to mitigate the risk of cyber attacks.

2. **Vulnerability**:
A vulnerability is a weakness in a system that can be exploited by attackers to compromise the security of the system. Vulnerabilities can exist in software, hardware, or network configurations, and they can range from simple coding errors to complex design flaws. It is crucial to identify and remediate vulnerabilities to protect systems from potential security breaches.

3. **Threat Modeling**:
Threat modeling is a systematic approach to identifying and mitigating potential security threats in software systems. It involves analyzing the system architecture, identifying potential threats, and implementing security controls to address these threats. Threat modeling helps organizations proactively assess security risks and prioritize security measures to protect their systems.

4. **Secure Coding Practices**:
Secure coding practices refer to coding techniques and standards that help developers write secure and resilient code. This includes following secure coding guidelines, avoiding common coding mistakes, and implementing security best practices in the development process. Secure coding practices help reduce the risk of vulnerabilities in software applications.

5. **OWASP Top 10**:
The OWASP Top 10 is a list of the top ten most critical security risks facing web applications. It is published by the Open Web Application Security Project (OWASP) to raise awareness about common security vulnerabilities and provide guidance on how to address them. The OWASP Top 10 includes risks such as injection attacks, broken authentication, and sensitive data exposure.

6. **Static Application Security Testing (SAST)**:
Static Application Security Testing (SAST) is a method of analyzing application source code for security vulnerabilities. SAST tools scan code for potential vulnerabilities, such as buffer overflows, injection flaws, and insecure configurations. By identifying security issues early in the development process, SAST helps

developers address vulnerabilities before they are deployed in production.

7. **Dynamic Application Security Testing (DAST)**:
Dynamic Application Security Testing (DAST) is a method of testing web applications for security vulnerabilities while they are running. DAST tools simulate attacks on live applications to identify vulnerabilities such as input validation errors, session management issues, and insecure configurations. DAST helps organizations identify and remediate security risks in their web applications.

8. **Security Development Lifecycle (SDL)**:
The Security Development Lifecycle (SDL) is a security-focused software development process that helps organizations build secure applications. The SDL involves integrating security activities throughout the software development lifecycle, including requirements analysis, design, coding, testing, and deployment. By following the SDL, organizations can reduce the risk of security vulnerabilities in their software products.

9. **Penetration Testing**:
Penetration testing, also known as pen testing, is a security assessment technique that simulates real-world cyber attacks on systems to identify vulnerabilities and weaknesses. Penetration testers use a variety of tools and techniques to exploit security flaws in systems and provide recommendations for improving security. Penetration testing helps organizations assess their security posture and enhance their defenses against cyber threats.

10. **Cryptographic Controls**:
Cryptographic controls are security mechanisms that use cryptographic algorithms to protect sensitive data in software systems. This includes encryption to secure data at rest and in transit, digital signatures to verify the authenticity of data, and hashing to ensure data integrity. Cryptographic controls play a critical role in securing applications and protecting data from unauthorized access.

11. **Secure Configuration Management**:
Secure configuration management involves managing and maintaining secure configurations for software systems and infrastructure components. This includes applying security baselines, hardening system configurations, and restricting access to sensitive resources. Secure configuration management helps organizations reduce the attack surface of their systems and improve overall security posture.

12. **Security Incident Response**:
Security incident response is the process of reacting to and managing security incidents in an organization. This includes detecting and analyzing security breaches, containing and mitigating the impact of incidents, and restoring normal operations. A well-defined security incident response plan helps organizations respond effectively to security incidents and minimize the impact on their systems and data.

13. **DevSecOps**:
DevSecOps is a practice that integrates security into the DevOps process, emphasizing security throughout the software development lifecycle. DevSecOps aims to shift security left, making security considerations a fundamental part of the development process. By incorporating security into DevOps practices, organizations can build more secure and resilient software applications.

14. **Container Security**:
Container security refers to securing containerized applications and the underlying infrastructure to protect against cyber threats. This includes ensuring the integrity of container images, implementing access controls, and monitoring containerized environments for security vulnerabilities. Container security is essential for organizations deploying applications in containerized environments like Docker and Kubernetes.

15. **Secure Code Review**:
Secure code review is a process of reviewing and analyzing source code to identify security vulnerabilities and coding errors. Secure code reviews involve manual and automated code analysis to detect issues such as injection flaws, authentication bypasses, and insecure configurations. By conducting regular code reviews, organizations can identify and remediate security issues in their software applications.

16. **Least Privilege Principle**:
The least privilege principle is a security best practice that restricts users' access rights to the minimum level necessary to perform their tasks. By limiting privileges to only what is required, organizations can reduce the risk of unauthorized access and privilege escalation. Adhering to the least privilege principle helps organizations improve security and protect sensitive data from unauthorized access.

17. **Secure Software Development Frameworks**:
Secure software development frameworks provide guidelines, best practices, and tools for building secure software applications. These frameworks help developers incorporate security into the development process by addressing common security vulnerabilities and implementing security controls. Examples of secure software development frameworks include Microsoft's Security Development Lifecycle (SDL) and OWASP's Application Security Verification Standard (ASVS).

18. **Authentication and Authorization**:
Authentication is the process of verifying the identity of users or systems accessing a software application, while authorization determines what actions users are allowed to perform once authenticated. Implementing strong authentication mechanisms, such as multi-factor authentication, and fine-grained authorization controls helps organizations secure their applications and protect against unauthorized access.

19. **Secure API Development**:
Secure API development involves designing, implementing, and securing application programming interfaces (APIs) to protect against security threats. This includes implementing authentication and authorization mechanisms, encrypting data in transit, and validating input to prevent injection attacks. Secure API development is essential for ensuring the security of API-driven applications and protecting sensitive data.

20. **Secure Deployment Practices**:
Secure deployment practices involve securely deploying software applications to production environments while minimizing security risks. This includes using secure deployment pipelines, managing secrets and credentials securely, and monitoring applications for security vulnerabilities. By following secure

deployment practices, organizations can ensure that their applications are deployed securely and remain protected from cyber threats.

In conclusion, understanding key terms and vocabulary related to Secure Software Development Practices is essential for DevOps professionals to build secure and resilient software applications. By integrating security into the software development lifecycle and following best practices for secure coding, testing, and deployment, organizations can enhance their security posture and protect their systems from potential cyber threats. It is important to stay informed about the latest security trends and technologies to adapt security practices to evolving threats and ensure the security of software applications.