**Incident Postmortem: Spring4Shell Exploit (CVE-2022-22965)**

**Summary**

This incident involved a series of attempted remote code execution (RCE) attacks targeting vulnerabilities (CVE-2022-22965) within the Spring Framework. The attacks were detected on  **2022-03-20 03:16:34 UTC,**and leveraged a flaw that allowed unauthenticated attackers to inject malicious code through crafted HTTP requests.

**Stakeholders:**

* Security Operations Team
* Network Security Team
* Development Teams

**Severity: Critical**

**Impact**

The potential impact of a successful Spring4Shell exploit could have been severe. Attackers could have gained unauthorized access to sensitive data, deployed malware, or disrupted critical systems.

**Detection**

The Security Operations Team identified the attack attempts by analyzing firewall logs for suspicious traffic patterns.Specifically, logs indicated a spike in malicious POST requests targeting the /tomcatwar.jsp endpoint and containing parameters associated with the Spring4Shell exploit (e.g.,”class.module.classLoader.resources.context.parent.pipeline.first.pattern").

**Root Cause**

The root cause of this incident was the presence of the Spring4Shell vulnerability (CVE-2022-22965) within potentially exploited applications. This vulnerability stemmed from the ability for attackers to manipulate specific Spring Framework properties, ultimately enabling RCE.

**Resolution**

* **Network Mitigation**: The Network Security Team implemented a firewall rule to block incoming HTTP POST requests containing the identified malicious parameters (class.module.classLoader.resources.context.parent.pipeline.first.pattern and Runtime.getRuntime().exec) targeting the /tomcatwar.jsp endpoint.
* **Application Patching**: The development teams were notified and coordinated patching efforts to update any vulnerable Spring Framework libraries within their applications.

**Action Items**

* **Security Awareness Training**: Conduct Security Awareness training to educate personnel about the Spring4Shell vulnerability and best practices for secure coding practices.
* **Vulnerability Scanning**: Implement automated vulnerability scanning for Spring Framework dependencies to proactively identify and address future vulnerabilities.
* **Penetration Testing**: Conduct regular penetration testing to identify and remediate potential security weaknesses within applications.
* **Review Firewall Rules**: Regularly review and update firewall rules to address evolving security threats and vulnerabilities.

**Lessons Learned**

* The importance of staying informed about emerging security threats and applying security patches promptly.
* The effectiveness of proactive security measures such as network monitoring and vulnerability scanning.
* The value of collaboration between Security Operations, Network Security, and Development Teams for efficient incident response.