Incident Postmortem: Spring4shell

## **Summary**

Incident Start Time: 2022-03-20T03:16:34Z

Incident End Time: 2022-03-20T05:16:34Z

Participants: Telstra Security Operations, nbn team, Networks team

Status: Resolved

Impact: Severity 1 - Critical

Detection Time: 2022-03-20T03:16:34Z

Root Cause Fixed Time: 2022-03-20T02:16:34Z

**Impact**

* **Unauthorized access attempts** were made to the **nbn.external.network**, exposing critical systems to possible malicious payload execution.
* The vulnerability allowed attackers to execute arbitrary code remotely by leveraging a weakness in **Spring MVC running on Apache Tomcat**.
* **Potential risks:**
  + Data exfiltration
  + System compromise
  + Further lateral movement within the network

**Detection**

* The **Security Information and Event Management (SIEM) system** triggered alerts for **multiple unauthorized HTTP requests** to the application running on Tomcat.
* Web Application Firewall (WAF) logs indicated repeated requests containing **class.module.classLoader** references, a known Spring4Shell exploit signature.
* Incident Response (IR) team manually verified traffic patterns using **Wireshark and network monitoring tools**.

**Root Cause**

* The network was running a **Spring MVC** application packaged as a **WAR deployment** on **Apache Tomcat**.
* The system was using an outdated **Spring Framework version (5.3.17)**, which is vulnerable to **CVE-2022-22965**.
* The exploit leveraged Java **JDK 9+**, which allowed an attacker to **manipulate class loaders and execute arbitrary code remotely**.
* Insufficient **firewall rules** and **lack of specific security patches** contributed to the incident.

## **Resolution**

* **Immediate mitigation:**
  + Blocked incoming HTTP requests targeting tomcatwar.jsp using **firewall rules**.
  + Deployed emergency **Web Application Firewall (WAF) rules** to detect and block payloads containing:
    - class.module.classLoader.resources.context.parent.pipeline.first
    - Suspicious HTTP headers with content including %>//, <%, Runtime, and application/x-www-form-urlencoded.
  + Isolated and patched affected servers.
* **Permanent Fix:**
  + **Upgraded Spring Framework** to version **5.3.18** to mitigate CVE-2022-22965.
  + Applied **strict firewall filtering rules** to detect and block similar attacks.
  + Conducted a **full security audit** to ensure no backdoors were planted.

**Action items**

**1. Patch Management & System Updates**

* Regularly update Spring Framework and dependencies to the latest secure versions.
* Apply security patches as soon as they are released.

**2. Web Application Firewall (WAF) Rules**

* Configure a WAF to detect and block malicious requests containing exploit signatures.
* Implement custom firewall rules to block payloads exploiting vulnerabilities like Spring4Shell.

**3. Secure Application Deployment**

* Avoid deploying Spring applications as **WAR** files on **Tomcat** if possible. Instead, use **Spring Boot executable JARs**, which mitigate the vulnerability.
* Implement strict access control for servers running Tomcat.

**4. Input Validation & Security Hardening**

* Enforce strict input validation to block malicious payloads in HTTP requests.
* Disable unnecessary features in Tomcat that could allow remote code execution.