Incident Postmortem: Spring4Shell Malware Attack on NBN Infrastructure

## **Summary**

## On **2022-03-20 at 03:16:34 UTC**, an alert was triggered by Telstra’s Security Operations Centre (SOC) following suspicious traffic targeting the nbn.external.network infrastructure. The incident involved a **Spring4Shell malware attack** exploiting a zero-day vulnerability in Spring Framework 5.3.0.

## The SOC team, supported by the networks team, detected the attack through analysis of the firewall logs and responded within two hours. The firewall rule deployed successfully blocked further malicious attempts, mitigating the threat.

## **Severity:** P1 – Critical **Teams Involved:**

## Telstra Security Operations

## Networks Team

## NBN Infrastructure Team

## **Impact**

## The nbn.external.network infrastructure experienced service degradation.

## Normal functionality was impaired during the first 2 hours of the attack.

## Potential unauthorized access was attempted using malicious code injection via a JSP endpoint.

## No confirmed data exfiltration, but the system’s integrity was temporarily at risk.

## **Detection**

## The attack was discovered when the SOC received an alert triggered at **03:16:34 UTC**, based on abnormal HTTP POST requests to /tomcatwar.jsp containing suspicious Java payloads and encoded class loader references.

## Firewall logs confirmed multiple attack attempts using:

## POST method

## Headers: suffix=%>//, c1=Runtime, c2=<%, DNT=1

## Payload containing class.module.classLoader.resources.context

## **Root Cause:**

## The root cause was the exploitation of an unpatched zero-day vulnerability known as **Spring4Shell**, which allowed attackers to execute remote code via crafted HTTP POST requests targeting servers running Spring Framework 5.3.0.

## The vulnerability allowed attackers to manipulate internal class loader objects and execute arbitrary system commands through .jsp endpoints.

## **Resolution**

## A custom firewall rule was scripted and deployed via Python (firewall\_server.py).

## The rule filtered out POST requests to .jsp endpoints containing:

## Malicious headers and body patterns

## The path /tomcatwar.jsp

## Once deployed, the firewall began blocking all further attempts successfully.

## The attack was considered contained within **2 hours of the initial detection**.

## **Action Items**

**Immediate Actions Taken:**

* Triaged the incident and identified affected infrastructure (nbn.external.network)
* Notified the relevant teams for containment and incident response
* Developed and implemented a firewall rule to detect/block Spring4Shell exploit attempts
* Logged the incident for internal audit and reporting

**Future Recommendations:**

* Apply vendor patch to upgrade Spring Framework beyond 5.3.0
* Implement automated patch management for public-facing services
* Train staff on identifying zero-day attack indicators
* Regularly review firewall logs and test defense rules through simulation
* Conduct a tabletop incident response drill for similar remote-code execution scenarios