Incident Postmortem: Spring4shell,being aware

## **Summary**

We detected in your logs an suspicious log and many other key that are related to a malicious activity

## **Impact**

The attack type: remote code execution (rce) via spring4shell

Targeted server:nbn.external.network

Endpoint:/tomcatwar.jsp

## **Detection**

**Multiple suspicious POST requests** were detected targeting /tomcatwar.jsp**.**

Payloads included:

class.module.classLoader.resources.context.parent.pipeline.first.pattern — a known exploit path in Spring4Shell.

Headers like suffix=%>//, c1=Runtime, c2=<% — signatures for deploying a JSP shell.

Logs showed **attempted configuration overwrite** of Tomcat logging pipeline — a known vector for writing webshells to disk.

Alerts were triggered by the custom Python-based firewall.

**Root Cause**

**Vulnerability Exploited**: CVE-2022-22965 (Spring4Shell)

A flaw in how the Spring framework handles binding of request parameters to internal Java objects

**System Exposure**:

The /tomcatwar.jsp endpoint was publicly accessible.

Java Spring environment was not hardened against known Spring4Shell attack vectors.

No WAF or application-layer firewall in place prior to attack.

## **Resolution**

**Firewall Mitigation Implemented**:

A Python-based HTTP filter (firewall\_server.py) was developed and deployed.

The firewall inspects all POST requests to /tomcatwar.jsp:

Blocks requests with suspicious headers or payloads

Rejects attempts to override classloader or drop JSP shells

**Status**: Attack attempts are being detected and blocked in real-time.

## **Action Items**

**What was done to solve the problem :**

detected and monitor logs

Remove the hacker from the server

**For the future:**

Patch all spring framework.

Deploy Waf

Harden tomcat configurations

Conduct security audit