**Incident Postmortem: Spring4Shell Payload Bypass on Internal Web Server**

**Summary**

**At 2022-03-20 03:16:34 UTC,** the security team detected a series of malicious POST requests sent to an internal web server endpoint (/tomcatwar.jsp). These requests were attempting to exploit the **Spring4Shell vulnerability** via custom-crafted HTTP headers. The incident was identified during routine testing using the test\_requests.py script.  
Severity: **Medium**  
Teams Involved: **Cybersecurity Team**, **Network Operations Team**, **Development Team**

**Impact**:

* Remote code execution (RCE) could have allowed full system compromise.
* Attackers could have uploaded web shells and gained persistent access.
* Potential data leakage and reputational damage.

**Detection**

The malicious payload was detected during a red-team simulation using an internal test script. The payload used suspicious headers such as:

* c1=Runtime
* c2=<%
* suffix=%>// These are commonly used to exploit Java-based servers like Apache Tomcat via Spring4Shell RCE.

**Root Cause**

The HTTP server lacked any form of **request validation** or **header filtering**, allowing malicious payloads to be processed without restriction. Additionally, no logging or alerting mechanisms were present prior to the incident.

**Resolution**

A custom **Python-based firewall rule** was implemented in firewall\_server.py, which:

* Blocked all requests containing known malicious headers.
* Returned a 403 Forbidden response for flagged requests.
* Logged details of blocked requests for auditing.

The solution was verified using the test script, which confirmed that all malicious requests were effectively blocked.

**Action Items**

| **Action** | **Owner** | **Status** |
| --- | --- | --- |
| Implement firewall rules to block malicious headers | DevSecOps | ✅ Completed |
| Add request logging to monitor future attacks | Security Team | ✅ Completed |
| Create alert system for blocked/flagged requests | Network Team | ⏳ In Progress |
| Conduct review of all HTTP endpoints for vulnerabilities | Dev Team | ⏳ In Progress |
| Include header sanitization and WAF protection in production | Security Team | ⏳ Planned |
| Train engineering staff on Spring4Shell and similar exploits | HR/Training | ⏳ Scheduled |