OS command injection, simple case

**1. Summary**

**Bug Title**: OS Command injection in POST request to endpoint /product/stock in productId parameter.

**Severity:** Critical.

**Description:** Command injection is an attack in which the goal is execution of arbitrary commands on the host operating system via a vulnerable application. Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell. In this attack, the attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application. Command injection attacks are possible largely due to insufficient input validation..

**Date Discovered:** 31/7/2024.

**Status:** Solved.

**2. Bug Details**

**Vulnerability Type:**  OS Command Injection.

**Affected URL/Endpoint:**  /product/stock, productId parameter.

**Description:**  When checking the stock of any product, the application directly incorporates the productId input value into the shell without any validation. This vulnerability allows command injection, enabling attackers to execute arbitrary commands on the target system and potentially gain full control over the operating system. The vulnerable parameter is productId, and the vulnerable endpoint is /product/post, which accepts POST requests at the URL: <https://0a05006603c741ab81b19eae00ce00b5.web-security-academy.net/product?productId=2>

note: it works in any product id not just 2

**Steps to Reproduce:**

1. go to the lab link <https://portswigger.net/web-security/os-command-injection/lab-simple> and click acess the lab button

2. click on any product and scroll down to hit the check stock feature

3. click check the stock button and intercept the request with any interception proxy i used burp suite

4. in POST request to /product/stock endpoint there is a parameter called productId inject this payload to it ;whoami => productId=2;whoami

5. examine the response and notice that the command get excuted



**Impact:** If this bug is exploited, it could have severe consequences, including:

* System Compromise: Attackers could gain full control over the operating system,
* Data Breach: Sensitive data stored on the system could be accessed and exfiltrated
* Service Disruption: Attackers could disrupt services by terminating processes, altering system configurations, or consuming system resources, leading to downtime and a loss of service availability.
* Propagation of Malware: The compromised system could be used to propagate malware to other systems within the network, potentially leading to a wider compromise
* Legal and Regulatory Consequences: Unauthorized access and data breaches could result in legal action and penalties from regulatory bodies, especially if sensitive personal or financial data is compromised.

**3. Recommendations**

The most effective way to prevent OS command injection vulnerabilities is to never call out to OS commands from application-layer code. In almost all cases, there are different ways to implement the required functionality using safer platform APIs.If you have to call out to OS commands with user-supplied input, then you must perform strong input validation. Some examples of effective validation include:

* Validating against a whitelist of permitted values.
* Validating that the input is a number.
* Validating that the input contains only alphanumeric characters, no other syntax or whitespace.

Never attempt to sanitize input by escaping shell metacharacters. In practice, this is just too error-prone and vulnerable to being bypassed by a skilled attacker.

**4. Conclusion**

**Summary:** Fixing this bug is critical to prevent unauthorized system control, data breaches, service disruptions, malware propagation, financial loss, and legal consequences. Addressing it will significantly enhance the organization's security posture and safeguard its assets and reputation.

**5. Appendices**

**Tools Used:**  burp suite.

**References:**

* <https://owasp.org/www-community/attacks/Command_Injection>
* <https://portswigger.net/web-security/os-command-injection>