# Host Header Injection

**Host Header Injection** is a web vulnerability where an attacker **modifies the Host header** in an HTTP request to manipulate how the server behaves. If the server **trusts the Host header without validation**, it may result in:

* **Web cache poisoning**
* **Password reset poisoning**
* **URL redirection**
* **Bypass of access controls**
* **Information disclosure**

# What is the Host Header?

In HTTP/1.1, the Host header specifies the **domain name** of the target server:

GET / HTTP/1.1   
Host: vulnerable.com  
  
If the server uses this header to construct links, validate requests, or generate content — **without verifying its legitimacy** — it becomes vulnerable.

# Impact

| Impact Type | Description |
| --- | --- |
| Password Reset Poisoning | Malicious reset links with attacker’s domain |
| Web Cache Poisoning | Caches malicious Host-based responses |
| Open Redirect | Redirect user to attacker-controlled site |
| Bypass Access Controls | Confuse proxy/CDN by spoofing internal hostnames |
| Sensitive Data Exfiltration | Form submissions, metadata leaks |
| Virtual Host Routing Manipulation | In environments using virtual hosts |

# Attack Scenarios (With Examples)

## 1. Password Reset Poisoning

If an app sends emails like: https://vulnerable.com/reset?token=abc

But generates the URL like: $reset\_url = "https://" . $\_SERVER['HTTP\_HOST'] . "/reset?token=$token";

Then attacker sends: Host: attacker.com

Victim receives: <https://attacker.com/reset?token=abc>

Token is leaked to attacker when clicked.

## 2. Web Cache Poisoning

Attacker sends: Host: evil.com

If caching is based on full host, now **malicious content** is cached and served to others visiting the real site.

## 3. Open Redirect via Host

Login redirect: /login?next=https://<host>/dashboard  
If host is trusted blindly: Host: evil.com  
Redirects to: https://evil.com/dashboard

# Common Signs of Vulnerability

| Sign | Risk |
| --- | --- |
| Host header reflected in response body | May lead to HTML injection or phishing |
| Host used in password reset links | Reset poisoning risk |
| Application uses $SERVER['HTTP\_HOST'] or similar | Possible injection |
| Different behavior per Host | May exploit vhost routing or bypass logic |

# Testing Techniques

| Method | Tool | Payload |
| --- | --- | --- |
| Manual Tampering | Burp Suite / curl | Add Host: evil.com |
| Password Reset | Check email link generation |  |
| Cache Poisoning | Try injecting Host with X-Forwarded-Host |  |
| Redirect Abuse | Tamper Host with redirect URLs |  |
| Header Injection | Use newline characters for multi-header injection (advanced) |  |

# Tools

| Tool | Use |
| --- | --- |
| Burp Suite | Manual header manipulation, Repeater |
| curl | Command-line header testing |

curl -H "Host: evil.com" https://target.com

| **Param Miner (Burp Extension)** | Detect header injection and cache poisoning |  
| H**TTP Request Smuggler (Burp)** | Test internal proxy header trust issues |  
| **Nuclei Templates** | Detect known header injection patterns

# Mitigation

| Area | Fix |
| --- | --- |
| Whitelist Valid Hostnames | Allow only trusted domains in Host and X-Forwarded-Host |
| Don’t Trust Host Header for Logic | Use hardcoded domain names for email links, redirects |
| Secure Reverse Proxy Configuration | Ensure proper header sanitization by NGINX/Apache |
| Sanitize and Encode User-Supplied Data | Especially in headers, emails, and redirect URLs |
| Log and Monitor Anomalies | Flag suspicious host headers in logs and SIEM |

# Real-World Exploits

| Platform | Impact |
| --- | --- |
| GitHub (2014) | Password reset poisoning via Host |
| Netflix (2015) | Open redirect via unvalidated Host |
| HackerOne / Bug Bounty | Frequent reports for cache poisoning via Host injection |

# Points

“Host Header Injection is often overlooked because it exploits **how web servers trust headers by default**.”

“Test for this during VAPT by sending malformed Host, X-Forwarded-Host, and checking their reflection in responses, redirects, or emails.”

“In secure deployments, use **proxy configs and frameworks that strictly validate the Host header**, and always **hardcode sensitive links**.”