HTTP HOST HEADER ATTACKS

Date: 20/04/2021
Author: Sun* Cyber Security Research

Agenda

1. What is the HTTP Host header?
2. How to identify HTTP Host header vulnerabilities
3. Exploiting HTTP Host header vulnerabilities
4. Demonstrate
5. Prevent HTTP Host header attacks

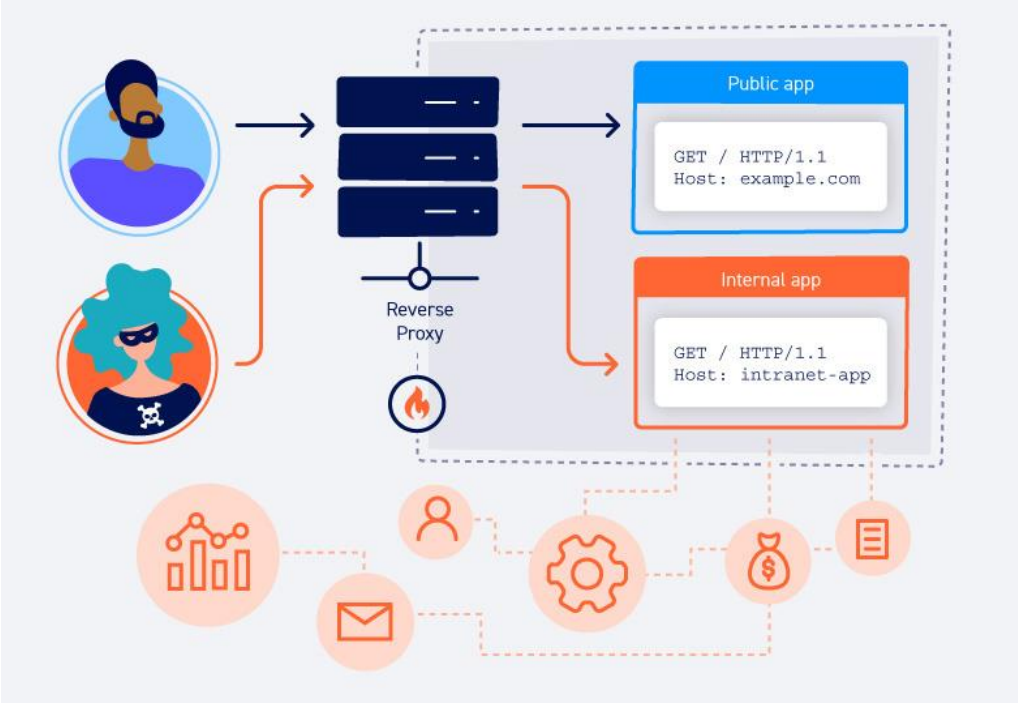
1. What is the HTTP Host header?

The HTTP Host header is a mandatory request header as of HTTP/1.1. It specifies the domain name that the client wants to access.

For example, when a user visits https://portswigger.net/web-security, their browser will compose a request containing a Host header as follows:

GET /web-security HTTP/1.1

Host: portswigger.net



Purpose of the HTTP Host header

- HTTP Host header is to help Identify which back-end component the client wants to communicate with.
- If requests didn't contain Host headers, or it was malformed in some way, this could lead to issues when routing incoming requests to the intended application.
- Cloud-based solutions and outsourcing much of the related architecture, it is common for multiple websites and applications to be accessible at the same IP address

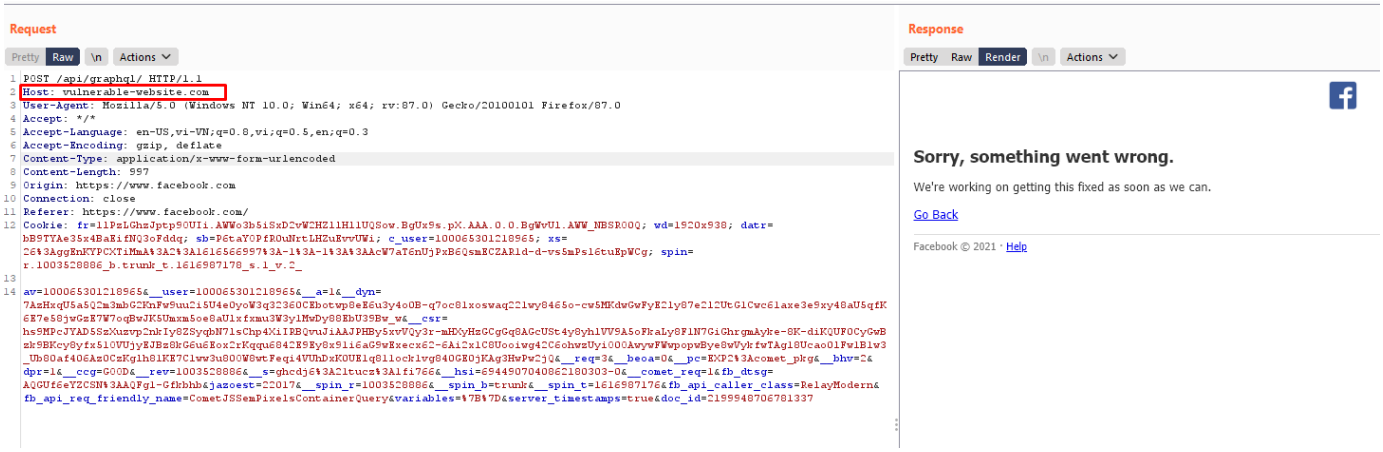
Virtual hosting: One possible scenario is when a single web server hosts multiple websites or applications. This could be multiple websites with a single owner, but it is also possible for websites with different owners to be hosted on a single, shared platform

Routing traffic via an intermediary: When websites are hosted on distinct back-end servers, but all traffic between the client and servers is routed through an intermediary system. This could be a simple load balancer or a reverse proxy server of some kind (CDN)

- When a browser sends the request, the target URL will resolve to the IP address of a particular server. When this server receives the request, it refers to the Host header to determine the intended back-end and forwards the request according

2. Identify HTTP Host header vulnerabilities

- Supply an arbitrary Host header:



- Check for flawed validation:

```
GET /example HTTP/1.1
Host: vulnerable-website.com:bad-stuff-here
```

Other sites will try to apply matching logic to allow for arbitrary subdomains. In this case, you may be able to bypass the validation entirely by registering an arbitrary domain name that ends with the same sequence of characters as a whitelisted one:

```
GET /example HTTP/1.1
Host: notvulnerable-website.com
```

Alternatively, you could take advantage of a less-secure subdomain that you have already compromised:

```
GET /example HTTP/1.1
Host: hacked-subdomain.vulnerable-website.com
```

- Check for flawed validation:

Inject duplicate Host headers	GET /example HTTP/1.1 Host: vulnerable-website.com Host: bad-stuff-here
Add line wrapping	GET /example HTTP/1.1 Host: bad-stuff-here Host: vulnerable-website.com
Supply an absolute URL	GET https://vulnerable-website.com/ HTTP/1.1 Host: bad-stuff-here

- Inject host override headers:

You can sometimes use `X-Forwarded-Host` to inject your malicious input while circumventing any validation on the Host header itself.

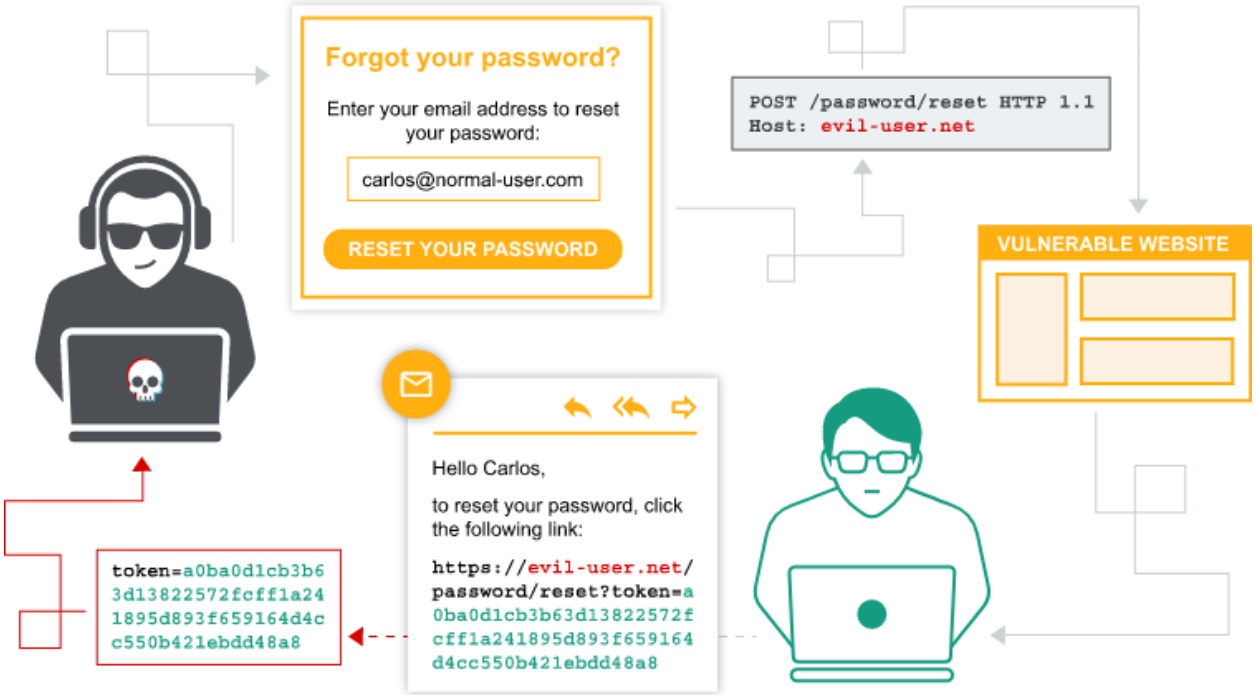
```
GET /example HTTP/1.1
Host: vulnerable-website.com
X-Forwarded-Host: bad-stuff-here
```

Although `X-Forwarded-Host` is the de facto standard for this behavior, you may come across other headers that serve a similar purpose, including:

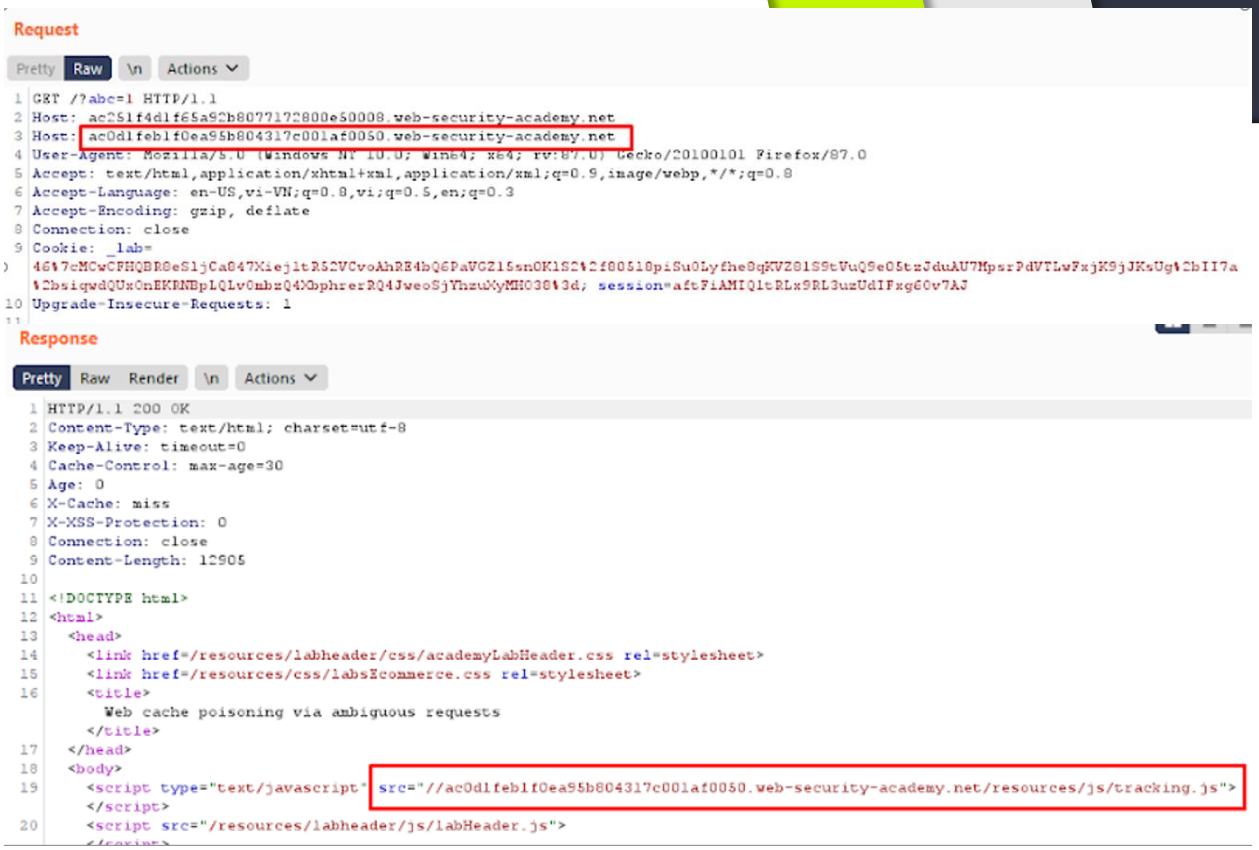
- `X-Host`
- `X-Forwarded-Server`
- `X-HTTP-Host-Override`
- `Forwarded`

3. How to exploit the HTTP Host header

- Exploiting classic server-side vulnerabilities
  - Accessing restricted functionality
  - Accessing internal websites with virtual host brute-forcing
  - Routing-based SSRF
- Exploiting classic server-side vulnerabilities
  - Accessing internal websites with virtual host brute-forcing
  - Routing-based SSRF
- **Example 1: Password reset poisoning attack to get reset password link:**



- **Example 2: Web cache poisoning via the Host header: (Duplicate Host header)**



4. Demo lab HTTP Host header on portswigger

- Password reset poisoning:
  - https://portswigger.net/web-security/host-header/exploiting/password-reset-poisoning
- Web cache poisoning:
  - https://portswigger.net/web-security/host-header/exploiting/lab-host-header-web-cache-poisoning-via-ambiguous-requests
- Host header authentication bypass:
  - https://portswigger.net/web-security/host-header/exploiting/lab-host-header-authentication-bypass
- Lab: Routing-based SSRF:
  - https://portswigger.net/web-security/host-header/exploiting/lab-host-header-routing-based-ssrf

Some report on hacker one:

- Web cache poisoning via the Host header
  - https://hackerone.com/reports/977851
  - https://hackerone.com/reports/504514

- Password reset poisoning:

    - https://hackerone.com/reports/226659
    - https://hackerone.com/reports/791293
- Inject host override headers
    - https://hackerone.com/reports/698416
    - https://hackerone.com/reports/758380

5. Prevent HTTP Host header attacks

- Protect absolute URLs
- Validate the Host header (ALLOWED_HOSTS)
- Don't support Host override headers
- Whitelist permitted domains
- Be careful with internal-only virtual hosts