Identifying & Escalating HTTP Host Header Injection attacks

goswamiijaya Following
Oct 30, 2020 · 6 min read

The purpose of the HTTP Host header is to help identify which back-end component the client wants to communicate with. Several misconfigurations and flawed business logic can expose websites to a variety of attacks via the *HTTP Host header*. Before diving in, let's understand some basic terminology.

What is an HTTP Header?

HTTP headers let the client and the server pass additional information with an HTTP request or response. An HTTP header consists of its ***case-insensitive*** name followed by a colon (:), then by its value.

What is a HOST Header?

The *Host* request header is the mandatory header (as per HTTP/1.1)that specifies the host and port number of the server to which the request is being sent.

If no port is included, the default port for the service requested is implied, *443 for an HTTPS URL, and 80 for an HTTP URL.*

Example: `Host: mysite.net`

What is a FORWARDED Header?

The `Forwarded` header contains information from the reverse proxy servers that is altered or lost when a proxy is involved in the path of the request.

The alternative and de-facto standard versions of this header are the `X-Forwarded-For`, `X-Forwarded-Host` and `X-Forwarded-Proto headers`.

This header is used for debugging, statistics, and generating location-dependent content and by design, it exposes privacy sensitive information, such as the IP address of the client.

Example: `X-Forwarded-For: yoursafesite.net`

What is the HOST header attack?

HTTP Host header attacks exploit vulnerable websites that handle the value of the Host header in an unsafe way. If the server implicitly trusts the Host header and fails to validate or escape it properly, an attacker may be able to use this input to inject harmful payloads that manipulate server-side behaviour.

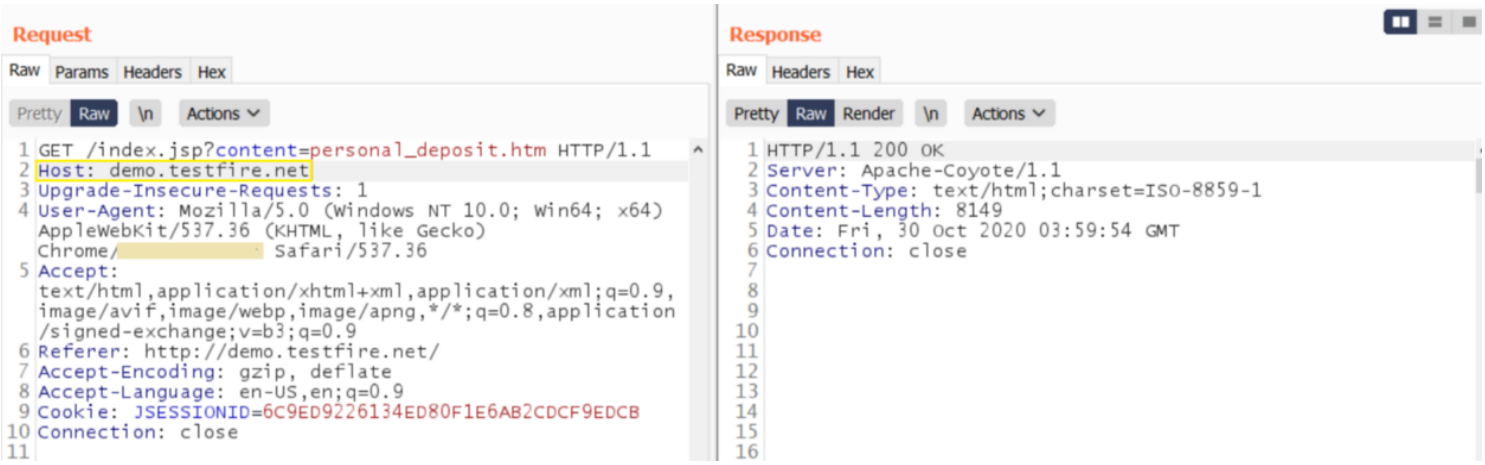
What exactly could be the flaw, where it could go wrong?

Earlier each IP address would only host content for a single domain. But today, it is common for multiple websites and applications to be accessible under the same IP address. As multiple applications are accessible via the same IP address as regards to — *Virtual hosting or Routing traffic via a proxy*. It is easy to get lost searching for the origin. Therefore the application relies on the **Host** header to specify the intended recipient.

How to test if the application could be vulnerable to Host Header injections?

Testing for Host Header injections is simple, all you need to do is to identify whether you are able to modify the ***Host*** header and still reach the target application with your request. If so, examine the application and observe what effect this has on the response.

The below image depicts the valid request-response from a web application.

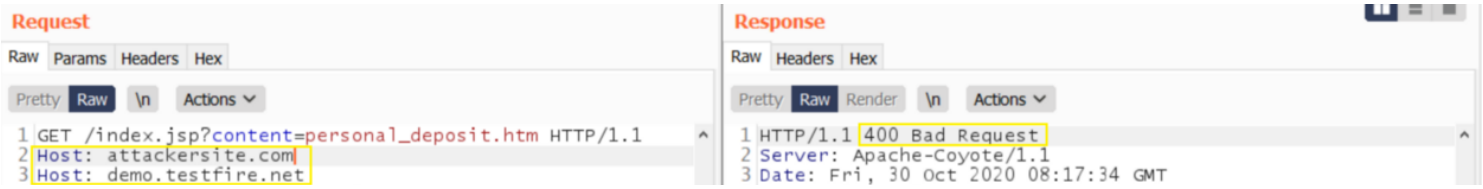


Host: demo.testfire.net

1. Supply an arbitrary Host header- try supplying a random host in the request and observe the application behavior. If a 200 OK is received, the attack could be escalated further.

Host: attackersite.net — 200 OK

2. Inject duplicate Host headers- try injecting multiple *host* headers, if a 200 OK is received, you could take it as a positive. But here, in this case, the application is not recognizing multiple host headers. So, a 400 Status code is received.



```
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/[redacted] Safari/537.36
6 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,im
  age/avif,image/webp,image/apng,*/*;q=0.8,application/sig
  ned-exchange;v=b3;q=0.9
7 Referer: http://demo.testfire.net/
8 Accept-Encoding: gzip, deflate
9 Accept-Language: en-US,en;q=0.9
10 Cookie: JSESSIONID=6C9ED9226134ED80F1E6AB2CDCF9EDCB
11 Connection: close
12
```

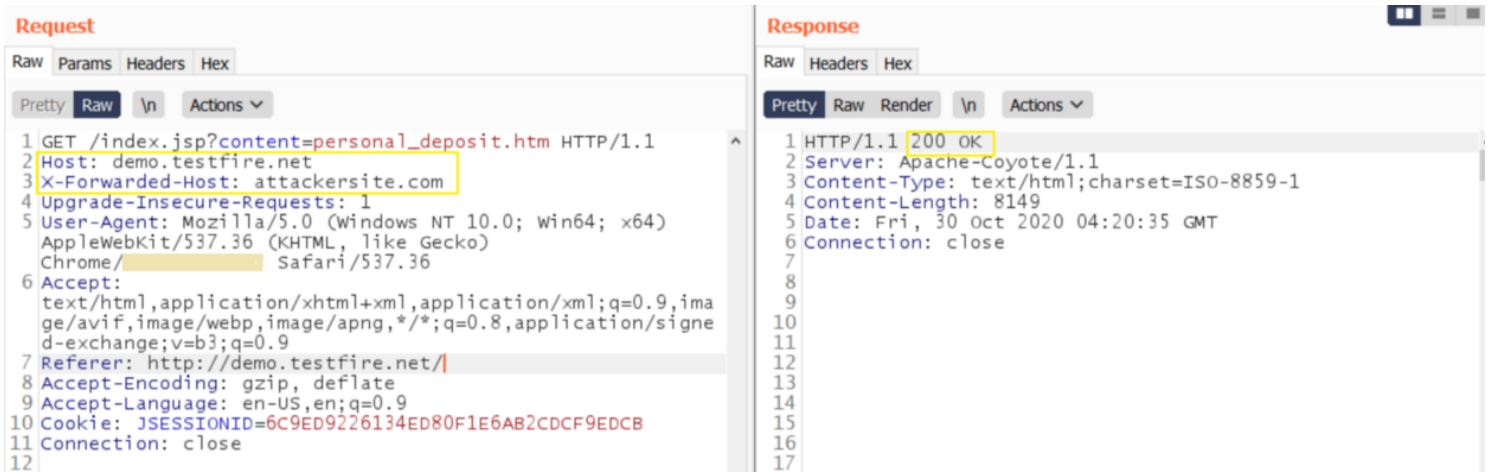
```
4 Connection: close
5 Content-Length: 0
6
7
```

Multiple Host headers

3. Add line wrapping- try adding a line wrapping with the “attackersite” in the *Host* header. Look for a successful 200 OK response.

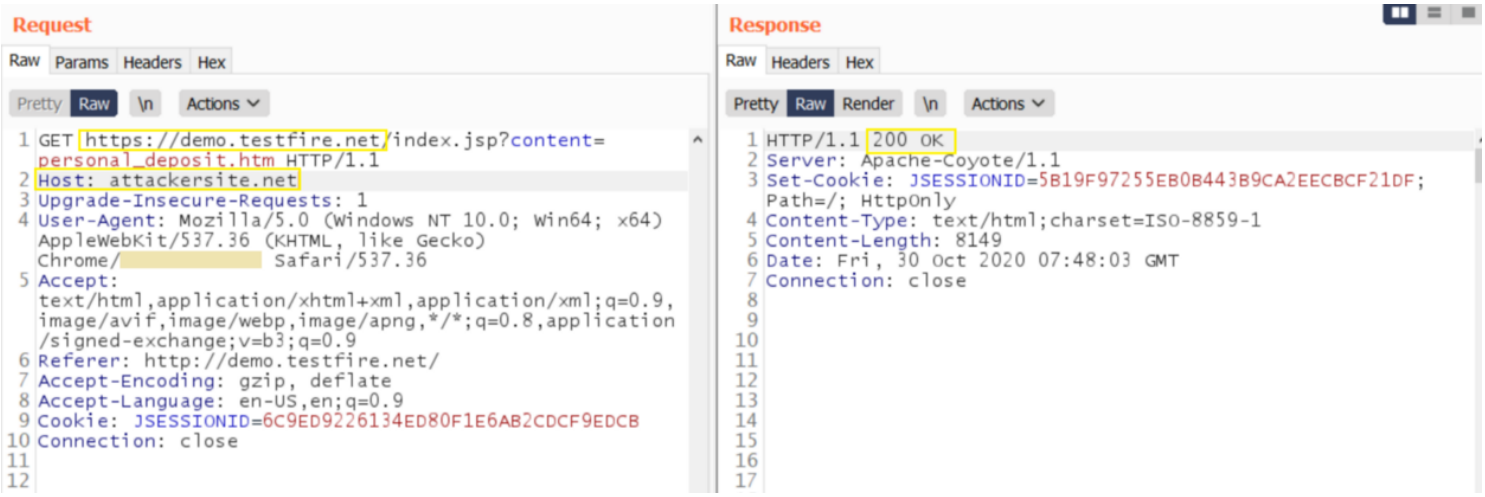
Line Wrapping — 200 OK

4. Inject host override headers- try injecting host override headers like — `X-Forwarded-Host, X-Host, X-Forwarded-Server, X-HTTP-Host-Override Forwarded` ; look if headers are supported via a successful 200 OK in response.



X-Forwarded-Host: attackersite.com — 200 OK

5. Supply an absolute URL- try supplying absolute URL in request & attackersite in host header. Keep an eye on a successful 200 OK.

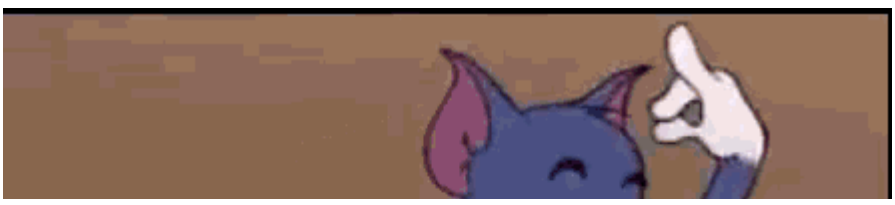


Absolute URL — 200 OK

What you as an attacker can do?

Without proper validation of the header value, the attacker can supply invalid input to cause the webserver to:

1. dispatch requests to the first virtual host on the list
2. cause a redirect to an attacker-controlled domain
3. perform web cache poisoning
4. manipulate password reset functionality
5. manipulate business logic flaws in specific functionality
6. perform routing-based SSRF
7. disclose classic server-side vulnerabilities, such as SQL injection



Escalate the attack — How you can exploit misconfigured Host Headers?

HTTP Host header vulnerabilities typically arise due to **the flawed assumption that "the header is not user-controllable".** This creates certain trust in the Host header and results in improper validation or escaping of user input.

Perform web cache poisoning

Look out if the *Host* header is reflected in the *response markup* without HTML-encoding, or even used directly in script imports.

```
GET / HTTP/1.1
Host: attackersite.com
```

The following will be served from the web cache when a victim visits the vulnerable application.

```
<link src="http://attackersite.com/link" />
```

Cause a redirect to an attacker-controlled domain

Try using the above-stated methods: to redirect the application to an attacker-controlled domain. Look if any sensitive tokens or keys are leaked or logged in the attacker's domain.

Manipulate password reset functionality

If a password reset functionality includes the *Host* header value when creating password reset links that use a *generated secret token*.

And if the application processes an *attacker-controlled domain* to create a password reset link, the victim may **click on the link in the email** and **allow the attacker to obtain the reset token**, thus resetting the victim's password.

> *… Email…*
>
> *Click on the following link to reset your password:* `http://www.attackersite.com/account.php?module=login&action=resetpassword&token=<YOUR_SECRET_TOKEN>`
>
> *… Email…*

Cause a redirect to restricted internal sites — SSRF

Vulnerable host headers can also lead to SSRFs, look out if you can access internal restricted sites, via redirection.

& Remember if:

Ciao!

References:

HTTP Host header attacks | Web Security Academy

In this section, we'll discuss how misconfigurations and flawed business logic can expose websites to a variety of...

portswigger.net

WSTG - Latest

A web server commonly hosts several web applications on the same IP address, referring to each application via the...

owasp.org

Infosec Bug Bounty Security Https Http Headers