Introduction to HTTP Headers

Cybersecurity Internship – Phase 1, Task 3

Name: Parmar Rakesh
Date:15,AUG,2025
Internship Program: Cybersecurity Internship – UJAR TECH

What is HTTP Headers?

An **HTTP Header** works like a **rule board** for the browser — it tells the browser how to handle the response it has received from the server: what to allow, what to block, and how to display the content.To understand this, let's take a simple example,Imagine you travel from one country to another. In the new country, you might not have the same freedoms you had in your own country — or the opposite could happen, and you might have more freedom there.HTTP Headers work in a similar way. When the server sends a response, it also sends some extra hidden instructions telling the browser:

- How the page should load
- How it should behave
- Which things are allowed
- Which things should be denied

Why are HTTP Headers Important?

In the past, a single IP address hosted only one website, so the server instantly knew what you wanted to see. Today, however, a single IP address can host multiple websites (due to cloud hosting, shared hosting, and the shortage of IPv4 addresses)[9], which can confuse the server about which site you're requesting. The **Host header** solves this by giving a clear instruction — for example: *"I want the website example.com on this IP."* It's like a building **(IP address)** with 10 different flats **(websites)**: when you tell the guard, *"I need to go to flat number 5,"* that flat number is like the **HTTP Host header**, letting the guard know exactly where to send you.

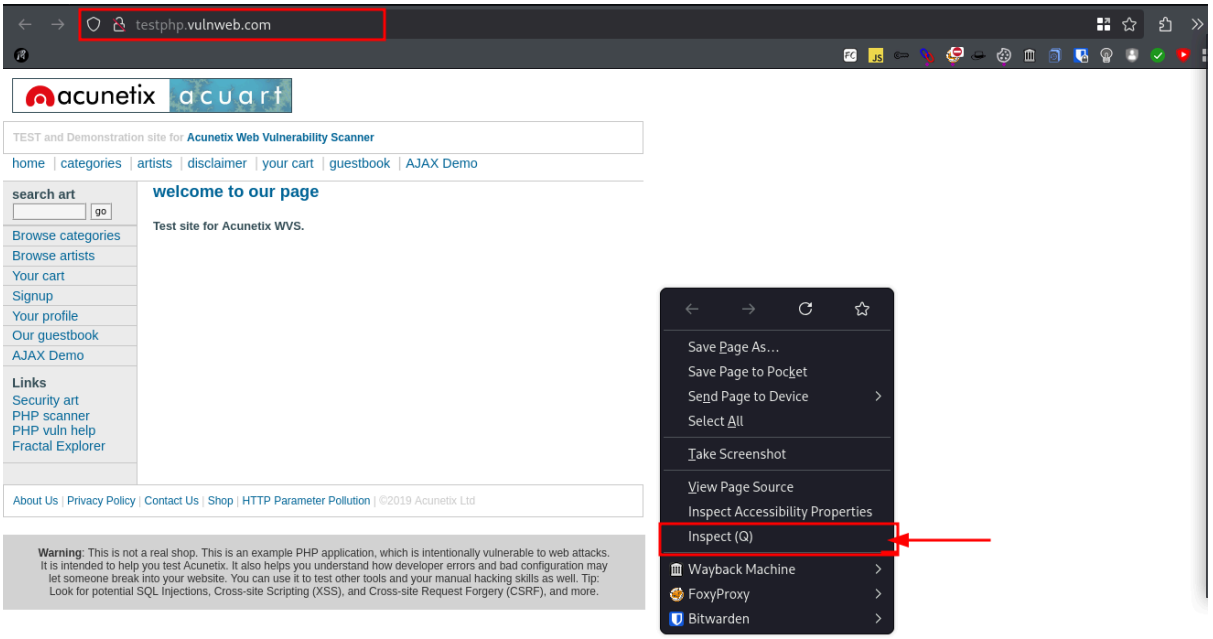
How To Check HTTP Headers?

First, we will understand how to read the headers of any website and what information we should look for — and what we should ignore. Website HTTP headers can be viewed in many ways: for example, directly from the browser, using online tools like securityheaders.com, Mozilla Observatory, and more. There are also command-line tools such as **nmap**, **curl**, and **wget** that can be used. Let's first look at how to check headers using online tools.

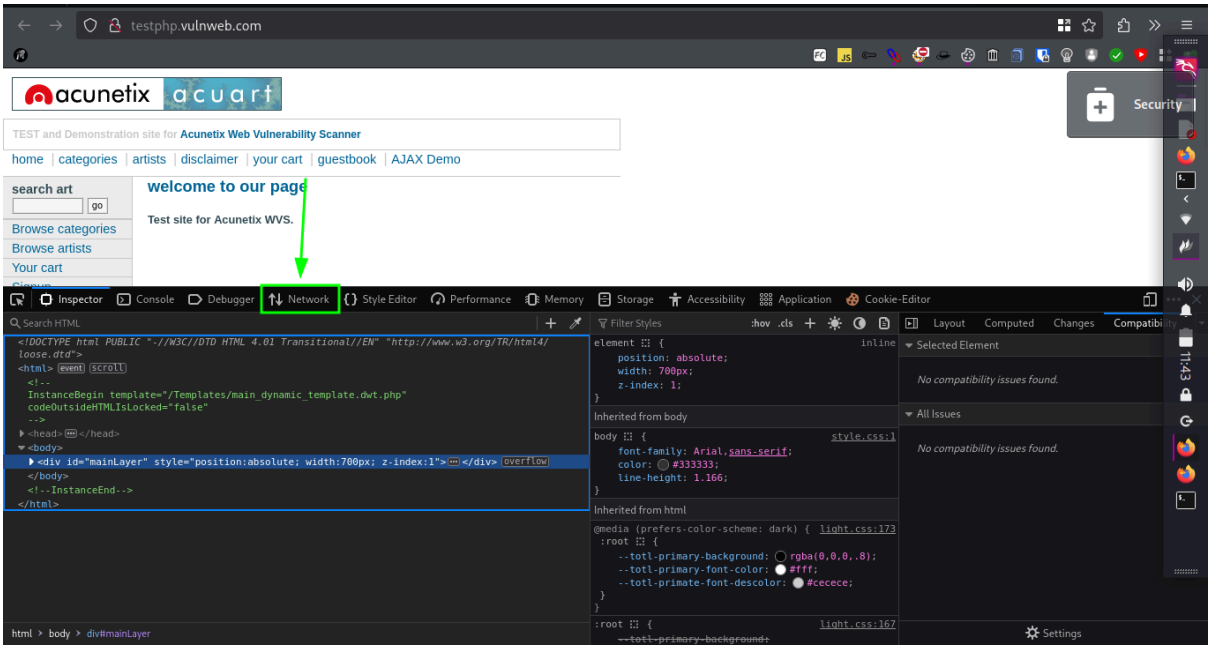
1.HTTP Headers Using Browser:

Open any browser and visit you targeted website.**Right click** or press **CTRL+Shift+I** anywhere and go to inspect and check network tab and reload the site after reloading site you need to click on web request, Yep we got it right side we can see HTTP Headers.

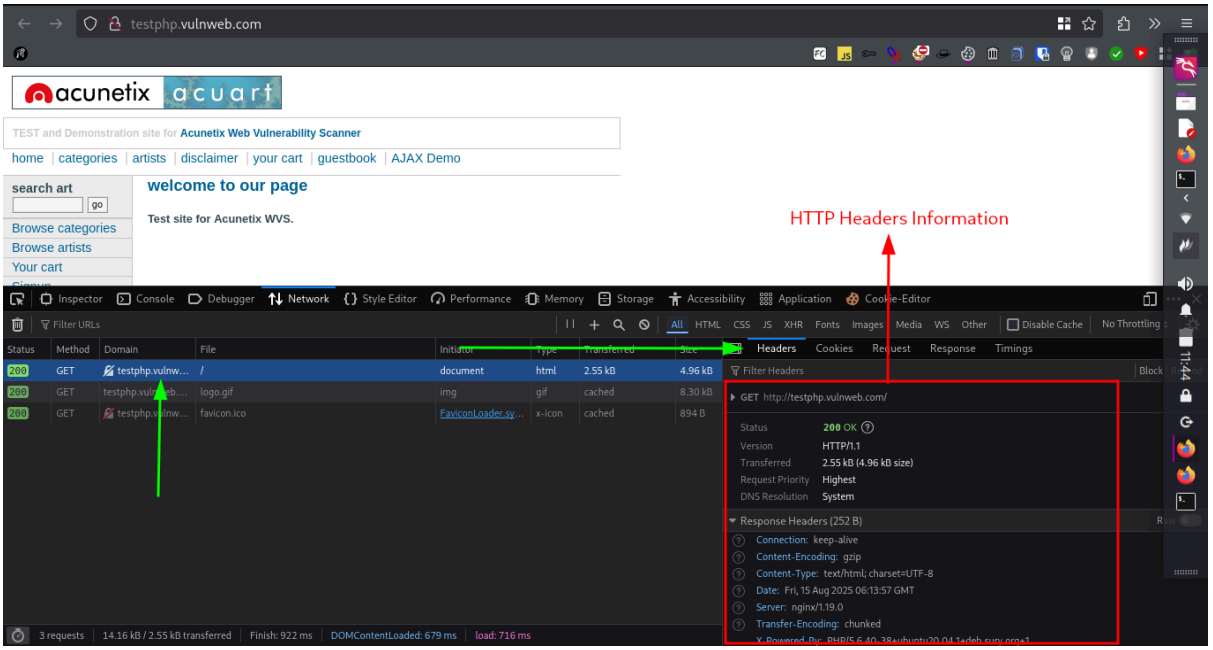
Step 1


testphp.vulnweb.com
acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer
welcome to our page
Test site for Acunetix WVS.
About Us | Privacy Policy | Contact Us | Shop | HTTP Parameter Pollution | ©2019 Acunetix Ltd
Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.
Save Page As…
Save Page to Pocket
Send Page to Device
Select All
Take Screenshot
View Page Source
Inspect Accessibility Properties
Inspect (Q)
Wayback Machine
FoxyProxy
Bitwarden

Step 2


testphp.vulnweb.com
acunetix acuart
Security
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
search art
go
Browse categories
Browse artists
Your cart
welcome to our page
Test site for Acunetix WVS.
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Cookie-Editor
Search HTML
Filter Styles
Layout
Computed
Changes
Selected Element
No compatibility issues found.
All Issues
No compatibility issues found.
Settings
html > body > div#mainLayer

Step 3

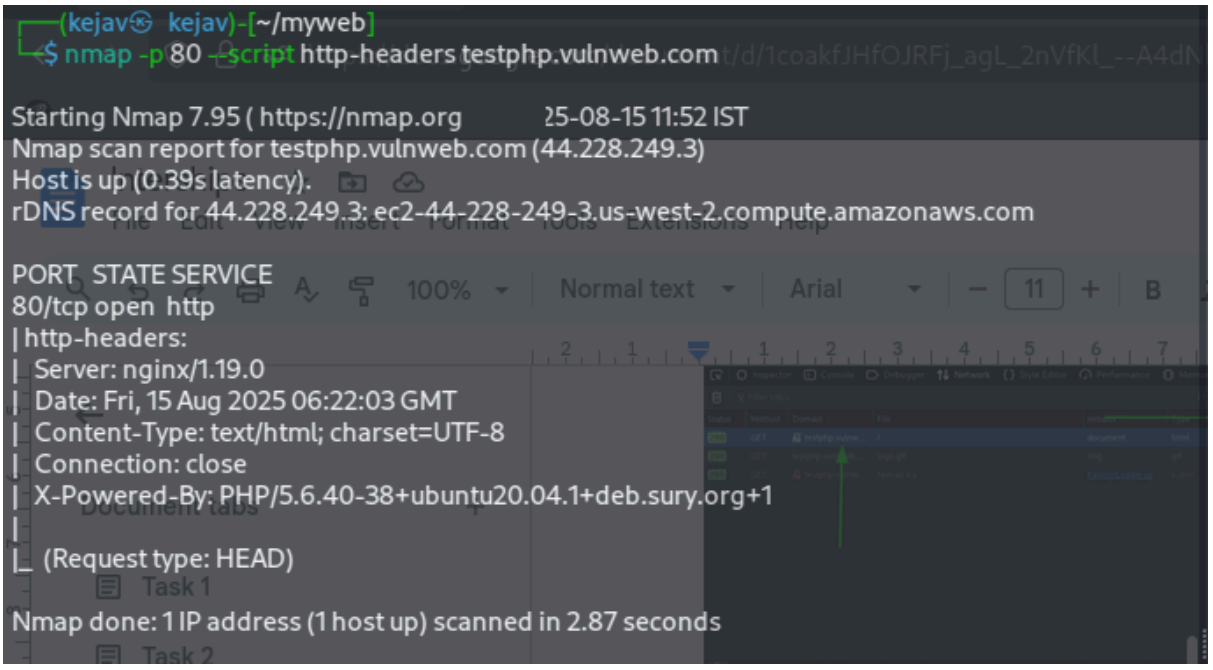


2.HTTP Headers Using tools

Now, let's move to the **tools** section, where we will see how to use **nmap** to view HTTP headers. For this, we first open our terminal and run the following command:

- **nmap -p 80 --script http-headers testphp.vulnweb.com**

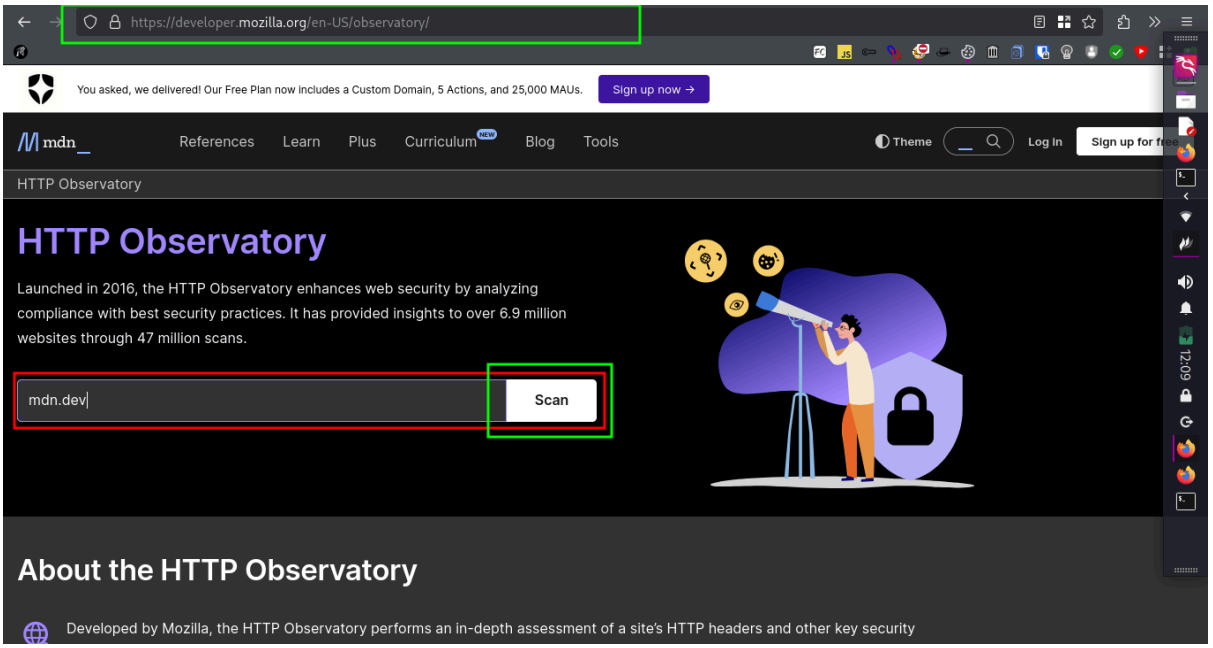
Here, we are scanning the default HTTP port (80) of the target using nmap, and by adding the **--script http-headers** parameter, we are retrieving the website's HTTP headers — as shown in the image.



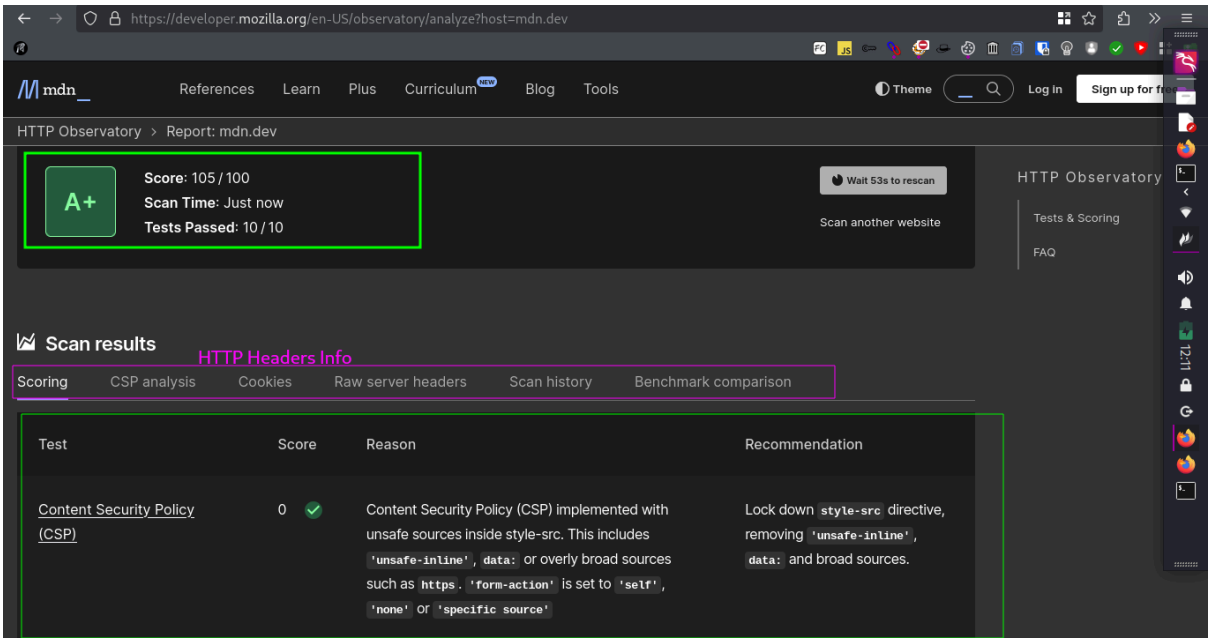
3.Online Tools:

Now, let's look at how we can view HTTP headers using **online tools**. There are many such tools available, but here I'll use Mozilla's https://observatory.mozilla.org/. We simply visit the site, enter the URL of our target website, and click on **Scan**. The online tool will then display the missing headers and the overall security level of that webpage based on its HTTP headers. For example, in the image shown, I scanned my own URL and received an **F** security rating — meaning the website's security is weak according to its HTTP headers. This rating scale goes from **A+** (best) down to **F** (worst).

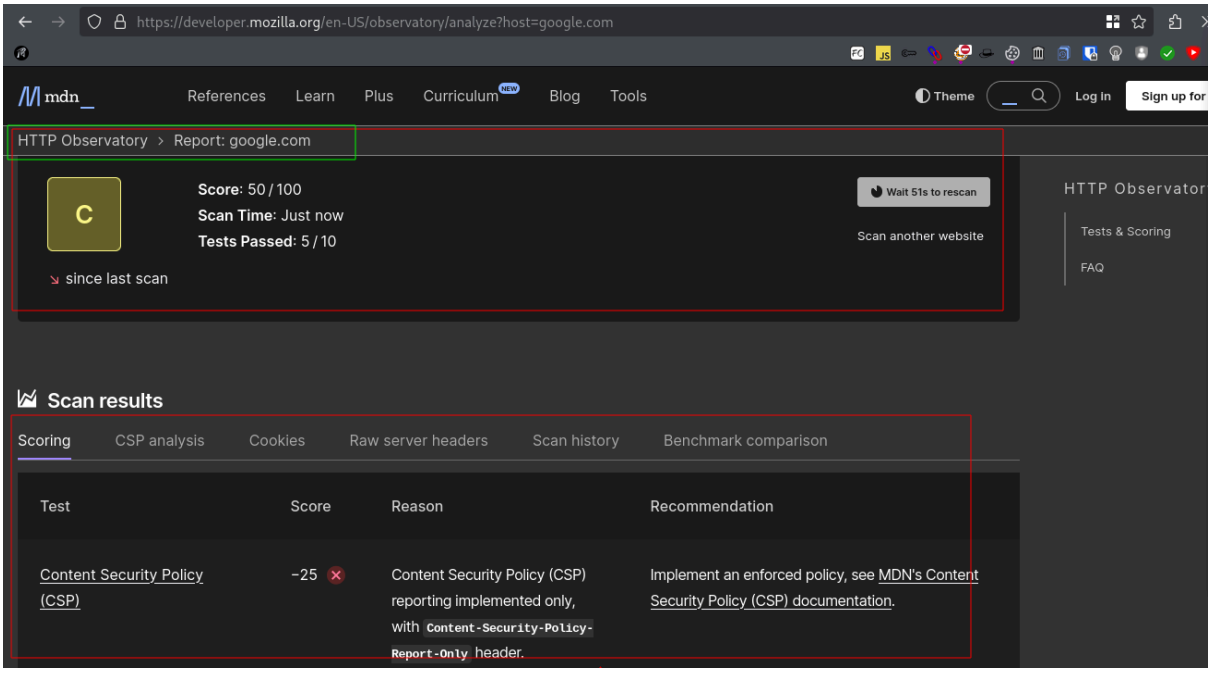
Step 1:



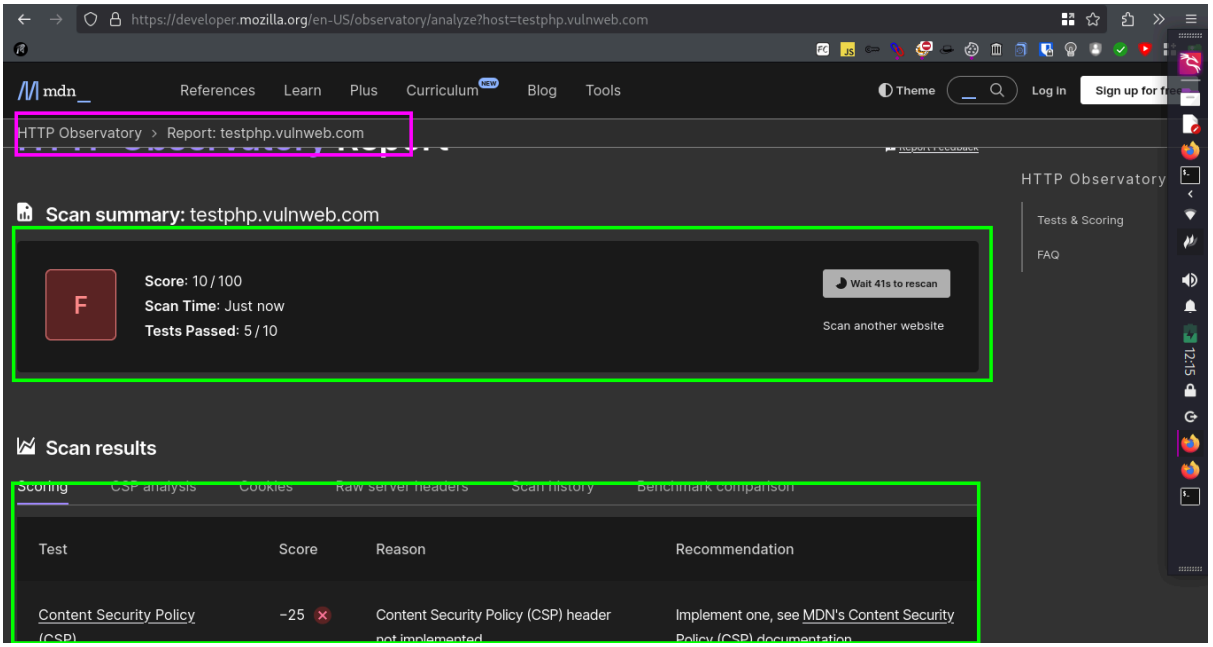
mdn.dev



google.com

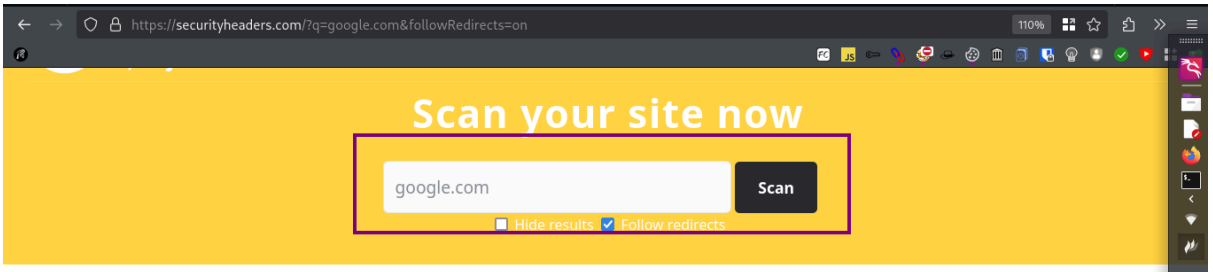


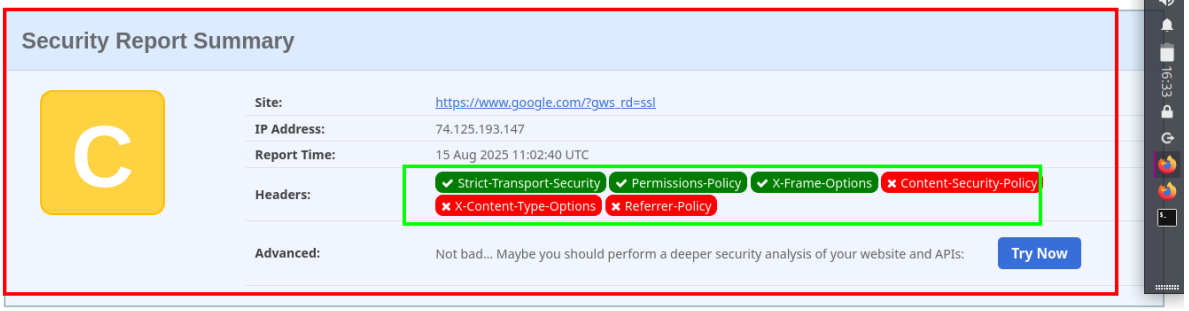
testphp.vulnweb.com

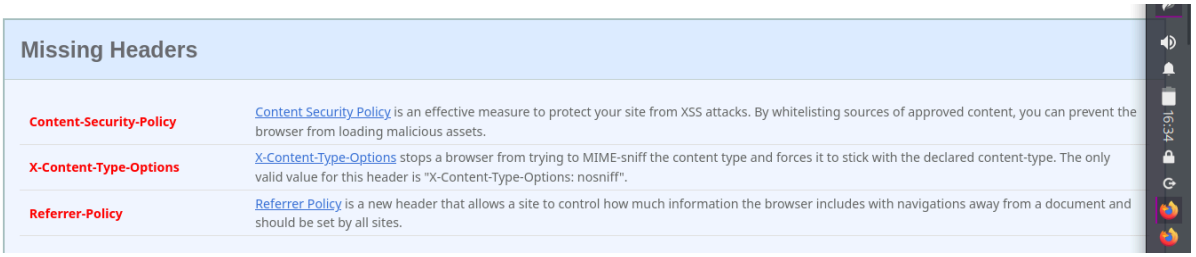


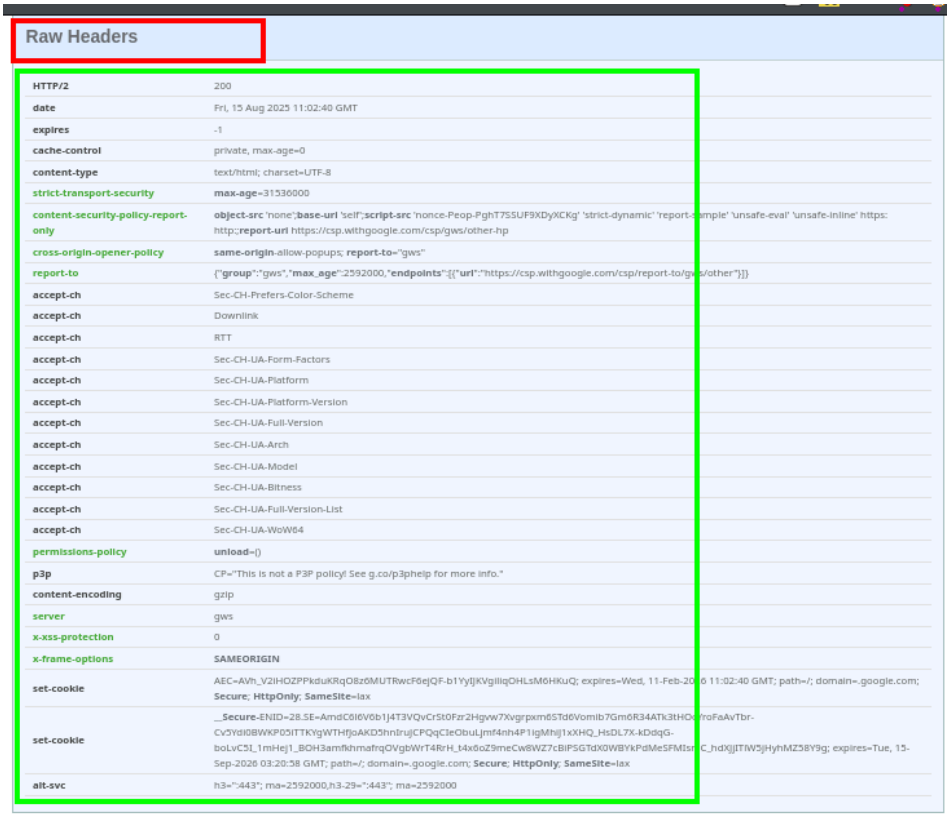
One more Onliner tool that I'm gonna show you that is https://securityheaders.com/

Scanning Google.com









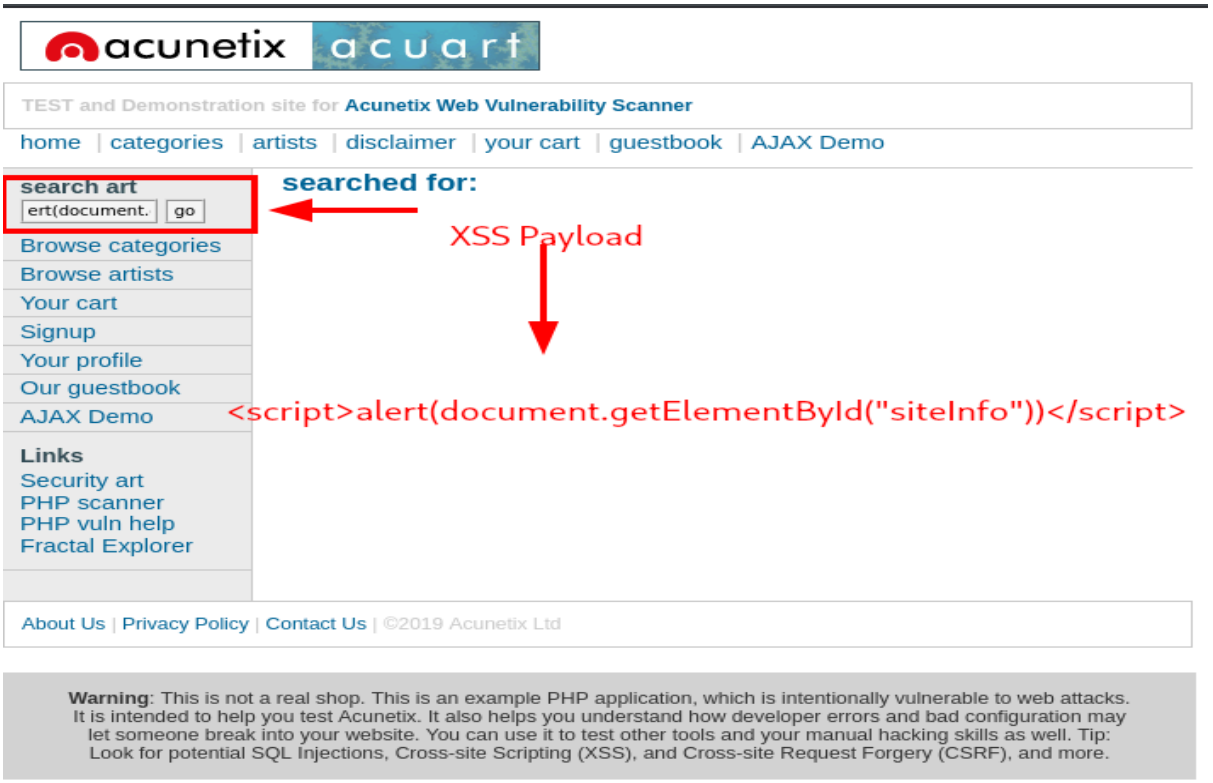
Here we can see that security Headers can give you more details about your target like which HTTP Headers are missing and raw info of the HTTP Headers

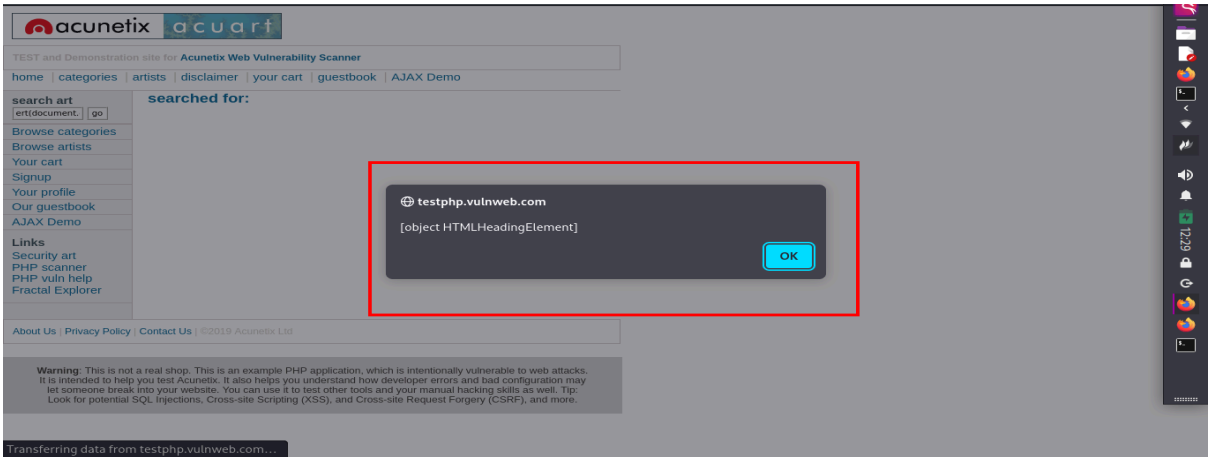
What is the risk of a missing HTTP security header?

When an HTTP security header is missing, an application may be more vulnerable to specific attack vectors. Here are some common risks associated with missing (or misconfigured) security headers:

1.**Missing Content Security Policy (CSP) header**

Without CSP to block unexpected content sources, an application may allow attackers to inject and execute malicious scripts in users' browsers to perform cross-site scripting (XSS). Let's take an example: For this example I am using testphp.vulnweb.com for the testing. Here we can see I added XSS payload on **testphp.vulnweb.com**





As shown in the image, I inserted an XSS payload, and it was reflected on the page. If the website owner had properly implemented a **CSP (Content Security Policy) header**, this XSS attack would not have been reflected. This is how CSP provides protection against such attacks.

2.**Missing CSP or X-Frame-Options header**

Clickjacking is basically *tricking you into clicking something different from what you think you're clicking*.Imagine you're pressing a game's **"Play"** button, but an attacker has secretly placed something else on top of it (invisible to you). You think you're starting the game, but in reality, you've just pressed your bank's **"Send Money"** button.A common trick is the **transparent overlay** method. Here, the attacker takes a trusted page (like a login or payment confirmation page), makes it transparent, and places it over something enticing — like a game or a special offer. When you click, you believe you're interacting with the visible content, but the click is actually going to the hidden, trusted page underneath.

Example:

- Visible (below): "Play Game" button.
- Hidden (above): Transparent "Confirm Purchase" button from a legitimate site.
  You think you're starting a game, but you just confirmed a payment.

Why it's dangerous: You believe you're performing a safe action, but in reality, the attacker is making you perform actions for them without your knowledge.Use the **X-Frame-Options: DENY** header so your site can't be loaded inside an iframe at all.

3.**Missing X-Content-Type-Options header**:

The **X-Content-Type-Options** response HTTP header is used by the server to indicate to the browsers that the MIME types advertised in the Content-Type headers should be followed and not guessed.This header is used to block browsers' MIME type sniffing[10], which can transform non-executable MIME types into executable MIME types (MIME Confusion Attack[11]).

Set the Content-Type header correctly throughout the site.

X-Content-Type-Options: nosniff

4.**Missing Strict-Transport-Security (HSTS) header**:

This header tells the browser to **always open the website over HTTPS**.If a user accidentally types **http://**, the browser will automatically switch to **https://** – without even sending an insecure HTTP request to the server.

- **Strict-Transport-Security: max-age=31536000; includeSubDomains**

1. Downgrade Attacks (SSL Stripping)
Without HSTS, an attacker can force your connection to use HTTP instead of HTTPS, allowing them to capture unencrypted data. Imagine you're on public Wi-Fi at a coffee shop. An attacker downgrades your session to HTTP and steals your credentials.

2. First Request Over HTTP
If you type **http://** manually or click an unsafe link, the browser will first send an HTTP request before redirecting to HTTPS – giving attackers a small window to intercept the connection.

3. Man-in-the-Middle (MITM) Risk
Without HSTS, attackers can intercept HTTP traffic and show you a fake version of the site.

4. Subdomain Vulnerability
If **includeSubDomains** is missing, only the main domain is forced to HTTPS.
Example:

- **secure.example.com** is secure (HTTPS)
- **login.example.com** might still load over HTTP, leaving it exposed

5.Missing Referrer-Policy header:

When you move from one page to another – for example, by clicking a link, loading an image, or making an API call — the browser sends a **Referrer** header that tells the destination page where you came from.The **Referrer-Policy** decides **how much** of this information is sent.Imagine You're on: **https://bank.com/account?balance=5000**and you click a link to:**https://other-site.com**If there's no Referrer-Policy, the other site will see your full URL, including sensitive details like **balance=5000.**

Preventing HTTP Header Attacks

1. Clickjacking Prevention
Problem: Attackers load your site inside an iframe to trick users into clicking hidden elements.
Fix:

- Set **X-Frame-Options: DENY or SAMEORIGIN**.
- Use **frame-ancestors 'none'** in CSP.

2. Insecure Connection Prevention (SSL Stripping)
Problem: If a user opens the HTTP version, attackers can intercept the data.
Fix:

- Set **Strict-Transport-Security: max-age=31536000; includeSubDomains; preload**.
- Always use a valid SSL certificate.

3. MIME Type Spoofing Prevention
Problem: The browser guesses the content type and may execute harmful files.
Fix:

- Set **X-Content-Type-Options: nosniff**.
- Configure the correct **Content-Type** header (e.g., **text/html; charset=UTF-8**).

4. Sensitive Data Leak Prevention
Problem: The Referrer header may leak sensitive query parameters to other sites.
Fix:

- Use **Referrer-Policy: strict-origin-when-cross-origin or no-referrer**.

5. XSS & Content Injection Prevention
Problem: Malicious scripts can steal data or hijack user sessions.
Fix:

- Apply a strong Content Security Policy (e.g., `default-src 'self'`).
- Avoid inline JavaScript.

6. Information Disclosure Prevention
Problem: Server version leaks in headers (e.g., **Server: Apache/2.4.41**).

Fix:

- Remove or replace **Server** and **X-Powered-By** headers with generic values.

Conclusion

HTTP Security Headers are an essential part of web security. They give browsers specific instructions on how to handle a website's content and what actions to allow or block. These headers help protect against common threats like **XSS (Cross-Site Scripting)**, **Clickjacking**, **data leakage**, **MIME sniffing**, and **insecure connections**.

Each header serves a unique purpose:

- **X-Frame-Options** and **CSP frame-ancestors** prevent clickjacking.
- **Strict-Transport-Security (HSTS)** enforces HTTPS, protecting against SSL stripping attacks.
- **X-Content-Type-Options** prevents MIME type spoofing.
- **Referrer-Policy** stops sensitive URL data from leaking.
- **Content-Security-Policy (CSP)** controls script and resource loading to minimize XSS and code injection risks.

When configured properly, these headers significantly improve a website's security posture. However, each must be planned carefully to avoid breaking essential site functionality with overly strict rules.In short, HTTP Security Headers act as a **first line of defense**—forcing browsers to follow safe behavior, reducing the attack surface for hackers, and ensuring a secure browsing experience for users.