Detection of Security Vulnerabilities in Web Application using Security Headers in Python

Monika A. Solanki[1], Dr. Ravi Sheth[2]

[1]*MTech Student, School of Information Technology & Cyber Security, Raksha Shakti University, Dahegam, Gandhinagar, Gujarat*

[2]*Guide, Assistant Professor, School of Information Technology & Cyber Security, Raksha Shakti University, Dahegam, Gandhinagar, Gujarat*

***Abstract*- A security breach is an early-phase breach that can lead to things like system damage, data loss and many more. As security breaches happened, new security patches were created and bolted on. What is vulnerable, needs to be patched. For HTTP, enter HTTP Security Headers. There's a lot of aspects to observe when looking for secure a site, and HTTP security headers are a good place to start. Keeping up with HTTP security headers best practices provide another security layer on top of your web applications. Different types of security headers are used for that. We can use these headers to outline communication and improve security of web applications. We can also write the python script to check which HTTP security headers are implemented by web application. And can fetch the value of the header, from the value of these headers we can conclude that the web application is vulnerable or not.**

***Index terms*- HTTP response header, HTTP Security Headers, Security Testing, Vulnerability Assessment**

I.INTRODUCTION

The world is becoming more and more connected every day, and online services like social media and e-commerce are contributing to giant treasure of sensitive business and personal data. These developments introduce new threats and vulnerabilities for hackers to exploit it via man-in-the-middle attacks, cross-site scripting (XSS), cross-site request forgery, click jacking and other threats.

Application developer teams should consider security from the start when designing and developing web applications. We can use HTTP security headers to increase security of web application. Once security headers are set, they can restrict modern browser from running into easily preventable vulnerabilities.

II. HTTP HEADERS

HTTP headers are the name or value pairs. And they are shown in the request and response messages of message headers for Hypertext Transfer Protocol (HTTP). The header name and value are separated by single colon. HTTP headers are an inherent part of HTTP requests and responses. In simpler terms, HTTP headers are the code that transfers data between a Web server and a client (browser). HTTP headers are fundamental for the communication between the web server and client in both directions.

HTTP headers are mainly divided into two types:

1 HTTP Request Header
2 HTTP Response Header

HTTP Request Header: When you type a URL into the address bar of web browser and try to access it, browser sends an HTTP request to the web server. It contains information in a text-record form, which includes the type, capabilities and version of the browser that generates the request, the OS used by the client, the page that was requested, the different types of outputs that are accepted by the browser, and so on.

HTTP Response Header: Upon receiving the request header, the server will send an HTTP response header back to the client's browser. An HTTP response header contains information in a text-record form that a Web server sends back to the client's browser. The response header contains the type, date and size of the file sent back by the server, and information regarding the server.

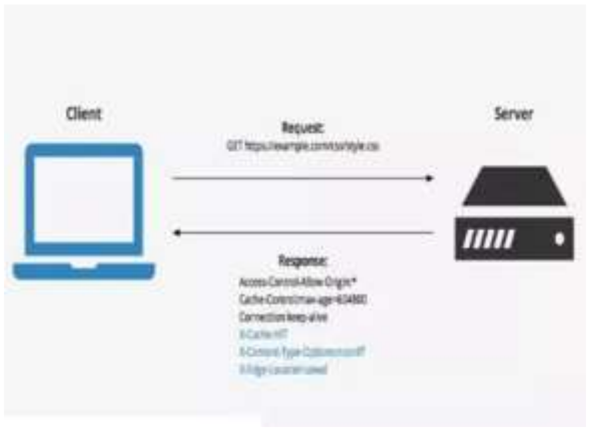
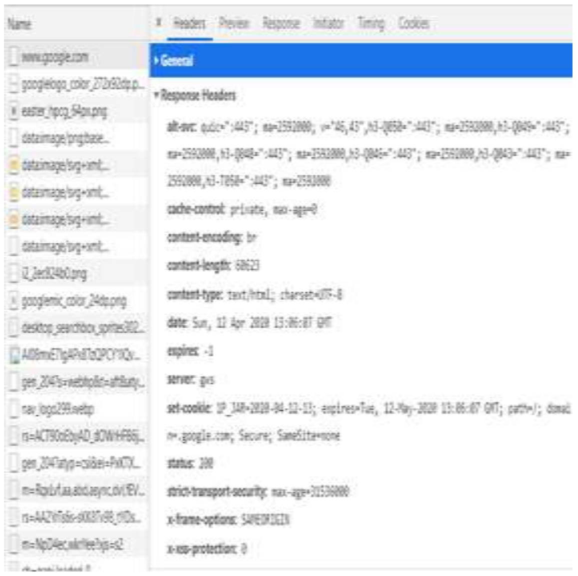

Client
Request:
GET https://example.com/css/style.css
Server
Response:
Access-Control-Allow-Origin:*
Cache-Control:max-age=604800
Connection:keep-alive
X-Cache:HIT
X-Content-Type-Options:nosniff
X-Edge-Location:used

Fig. 1 Client - Server Communication

III. HTTP SECURITY HEADERS

Whenever a browser requests a web page from a web server, the server responds with the content along with HTTP response headers. Some of these headers contain content Meta data such as the content-encoding, cache-control, content-length, status error codes, etc.



```
Name                          × Headers Preview Response Initiator Timing Cookies
www.google.com                ▸ General
googlelogo_color_272x92dp.p…  ▾ Response Headers
easter_hpcg_54px.png            alt-svc: quic=":443"; ma=2592000; v="46,43",h3-Q050=":443"; ma=2592000,h3-Q049=":443";
dataimage/png;base…             ma=2592000,h3-Q048=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=
dataimage/svg+xml…              2592000,h3-T050=":443"; ma=2592000
dataimage/svg+xml…              cache-control: private, max-age=0
dataimage/svg+xml…              content-encoding: br
dataimage/svg+xml…              content-length: 68623
i2_2ed824b0.png                 content-type: text/html; charset=UTF-8
googlemic_color_24dp.png        date: Sun, 12 Apr 2020 13:06:07 GMT
desktop_searchbox_sprites302…   expires: -1
AI0ilmxE7lgAPkBTdQPCY1Qy…       server: gws
gen_204?s=webhp&t=aft&aty…      set-cookie: 1P_JAR=2020-04-12-13; expires=Tue, 12-May-2020 13:06:07 GMT; path=/; domai
nav_logo299.webp                n=.google.com; Secure; SameSite=none
rs=ACT90oEbyAD_dOWHF86j…        status: 200
gen_204?atyp=csi&ei=PxKTX…      strict-transport-security: max-age=31536000
m=Rqdvf,aa,abd,async,dvl,fEV…   x-frame-options: SAMEORIGIN
rs=AA2YrTs6s-sX03Tv98_tIDs…     x-xss-protection: 0
m=NpD4ec,wkrYee?xjs=s2
```

Fig. 2 Response header of google.com no chrome browser

Along with these are also HTTP security headers that tell browser how to behave when handling website's content. For example, by using the strict-transport-security (HSTS) you can force the browser to communicate only over HTTPS. There are seven different HTTP security headers that we will explore here that you should be aware of.

IV. DIFFERET TYPES OF SECURITY HEADERS

X-Frame-Options

The X-Frame-Options response header protects websites against click jacking attacks. The application server sets the X-Frame-Options header policy in the client's browser to allow or deny rendering of the web content in the frames of third-party websites.

The value of an X-Frame-Options header:

X-Frame-Options: DENY | SAMEORIGIN | ALLOW-FROM URL

Strict-Transport-Security (HSTS)

HTTP Strict Transport Security is an HTTP response header that mandates that web browsers or compatible agents should only interact with applications using HTTPS connections and never plain HTTP or lower version of Transport Layer Security (TLS). The application server implements an HSTS policy by giving the header "Strict-Transport-Security" over an HTTPS connection. Setting this HSTS header can help protect applications against version rollback attacks, such as Poodle.

The value of HSTS header:

Strict-Transport-Security: max-age=31536000; include Sub Domains; preload

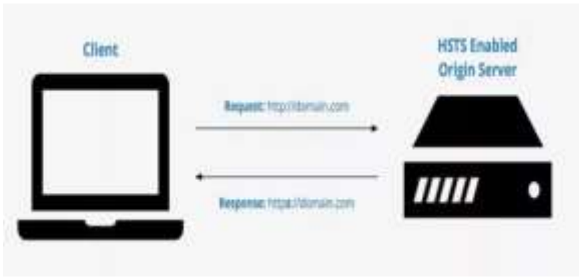

Client
HSTS Enabled Origin Server
Request: http://domain.com
Response: https://domain.com

Fig. 3 Strict-transport -security header

X-Powered-By

May be set by hosting environment or other framework. It contains information about them while not providing any usefulness to the application or its visitors. Unset this header to ignore exposing potential vulnerabilities.

X-Content-Type-Options

When the application server sets the X-Content-Type-Options header to “nosniff” in the client’s browser, the script and style sheet elements will reject responses with incorrect Multipurpose Internet Mail Extensions (MIME) types. Thus limiting exposure to download and the risk of uploaded data that could be treated as executable or dynamic HTML files. X-Content-Type-Options security header helps to prevent attacks based on MIME type confusion.

The value of an X-Content-Type-Options header:

X-Content-Type-Options: nosniff

Content-Security-Policy (CSP)
The Content-Security-Policy header provides defense against content injection attacks, such as XSS and click jacking. This header provides a standard method for websites to establish confidence and determine the origins of content the browser is allowed to load on the webpage. CSP requires careful tuning and exact definition of the policy.

The value of a CSP header:

Content-Security-Policy: frame-ancestors 'none'; // No URL can load the page within an iframe.

Content-Security-Policy: frame-ancestors 'self'; // Serves the same function as the SAMEORIGIN parameter.

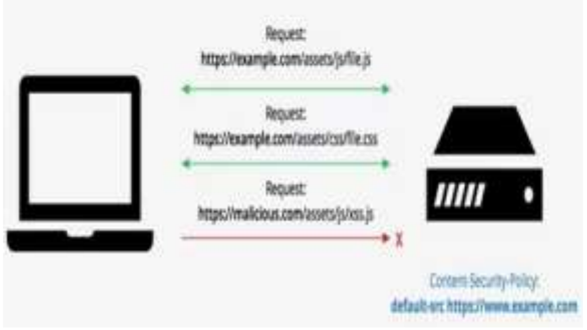

Request:
https://example.com/assets/js/file.js
Request:
https://example.com/assets/css/file.css
Request:
https://malicious.com/assets/js/xss.js
X
Content-Security-Policy:
default-src https://www.example.com

Fig. 4 Content-Security-Policy header

X-XSS-Protection
The X-XSS-Protection header is designed to defend websites against reflective XSS attacks. The application server sets X-XSS-Protection header in the client’s browser with proper values to disable the protection or to sanitize or block reflective XSS attacks.

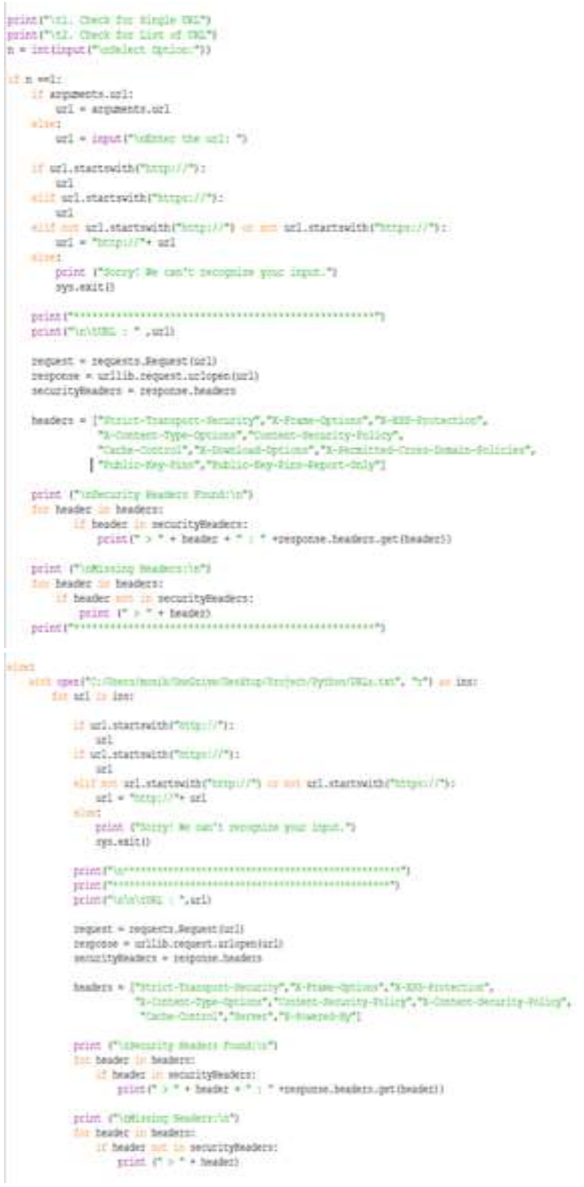
The value of an X-XSS-Protection header:

X-XSS-Protection: 1; mode=block; report=https://domain.tld/folder/file.ext

Cache-Control
The Cache-Control HTTP header holds directives (instructions) for caching in requests and responses. A given directive in a application request does not mean that the same directive should be in the response.

V. PROPOSED WORK

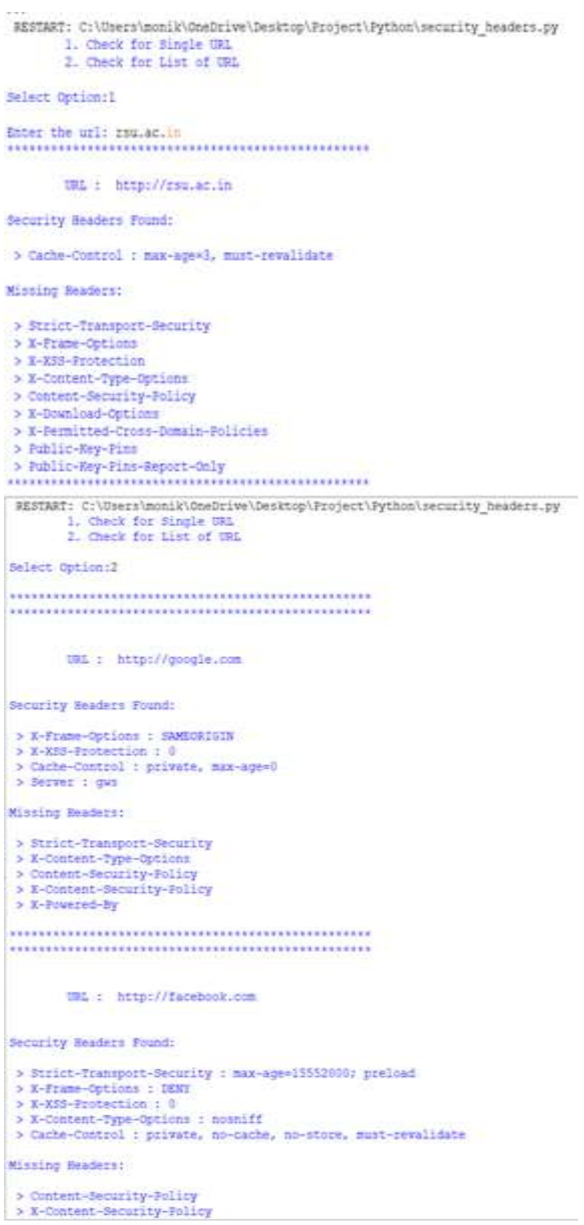
A. Python Script



```
print("\t1. Check for Single URL")
print("\t2. Check for List of URL")
n = int(input("\nSelect Option:"))

if n ==1:
    if arguments.url:
        url = arguments.url
    else:
        url = input("\nEnter the url: ")

    if url.startswith("http://"):
        url
    elif url.startswith("https://"):
        url
    elif not url.startswith("http://") or not url.startswith("https://"):
        url = "http://"+ url
    else:
        print ("Sorry! We can't recognize your input.")
        sys.exit()

    print("*************************************************")
    print("\n\tURL : " ,url)

    request = requests.Request(url)
    response = urllib.request.urlopen(url)
    securityHeaders = response.headers

    headers = ["Strict-Transport-Security","X-Frame-Options","X-XSS-Protection",
               "X-Content-Type-Options","Content-Security-Policy",
               "Cache-Control","X-Download-Options","X-Permitted-Cross-Domain-Policies",
              |"Public-Key-Pins","Public-Key-Pins-Report-Only"]

    print ("\nSecurity Headers Found:\n")
    for header in headers:
            if header in securityHeaders:
                print(" > " + header + " : " +response.headers.get(header))

    print ("\nMissing Headers:\n")
    for header in headers:
        if header not in securityHeaders:
            print (" > " + header)
    print("*************************************************")

else:
    with open("C:/Users/monik/OneDrive/Desktop/Project/Python/URLs.txt", "r") as ins:
        for url in ins:

            if url.startswith("http://"):
                url
            if url.startswith("https://"):
                url
            elif not url.startswith("http://") or not url.startswith("https://"):
                url = "http://"+ url
            else:
                print ("Sorry! We can't recognize your input.")
                sys.exit()

            print("\n*************************************************")
            print("*************************************************")
            print("\n\n\tURL : ",url)

            request = requests.Request(url)
            response = urllib.request.urlopen(url)
            securityHeaders = response.headers

            headers = ["Strict-Transport-Security","X-Frame-Options","X-XSS-Protection",
                       "X-Content-Type-Options","Content-Security-Policy","X-Content-Security-Policy",
                        "Cache-Control","Server","X-Powered-By"]

            print ("\nSecurity Headers Found:\n")
            for header in headers:
                if header in securityHeaders:
                    print(" > " + header + " : " +response.headers.get(header))

            print ("\nMissing Headers:\n")
            for header in headers:
                if header not in securityHeaders:
                    print (" > " + header)
```

Fig. 5 Python Script

B. Output



```
...
 RESTART: C:\Users\monik\OneDrive\Desktop\Project\Python\security_headers.py
        1. Check for Single URL
        2. Check for List of URL

Select Option:1

Enter the url: rsu.ac.in
**************************************************

        URL :  http://rsu.ac.in

Security Headers Found:

 > Cache-Control : max-age=3, must-revalidate

Missing Headers:

 > Strict-Transport-Security
 > X-Frame-Options
 > X-XSS-Protection
 > X-Content-Type-Options
 > Content-Security-Policy
 > X-Download-Options
 > X-Permitted-Cross-Domain-Policies
 > Public-Key-Pins
 > Public-Key-Pins-Report-Only
**************************************************
 RESTART: C:\Users\monik\OneDrive\Desktop\Project\Python\security_headers.py
        1. Check for Single URL
        2. Check for List of URL

Select Option:2

**************************************************
**************************************************


        URL :  http://google.com


Security Headers Found:

 > X-Frame-Options : SAMEORIGIN
 > X-XSS-Protection : 0
 > Cache-Control : private, max-age=0
 > Server : gws

Missing Headers:

 > Strict-Transport-Security
 > X-Content-Type-Options
 > Content-Security-Policy
 > X-Content-Security-Policy
 > X-Powered-By

**************************************************
**************************************************


        URL :  http://facebook.com


Security Headers Found:

 > Strict-Transport-Security : max-age=15552000; preload
 > X-Frame-Options : DENY
 > X-XSS-Protection : 0
 > X-Content-Type-Options : nosniff
 > Cache-Control : private, no-cache, no-store, must-revalidate

Missing Headers:

 > Content-Security-Policy
 > X-Content-Security-Policy
```

Fig. 6 Output

VI. CONCLUSION

We have to identify various issues and challenges faced by security testing of web based applications. A security tester thus should keep track of all the issues while conducting testing of web application for security purpose. HTTP security headers can help harden security of website. We can make python script to add more features and can detect more security vulnerabilities in web application. By using python we can also develop vulnerabilities scanner which scans for different kinds of vulnerabilities.

VIII. ACKNOWLEDGMENT

The author like to thanks IT department of Raksha Shakti University for giving all support and guidance which have enhanced the quality of the paper.

REFERENCES

[1] Henry KM (2012) Penetration testing: protecting networks and systems. IT Governance Publishing, UK.
[2] Geer D, Harthorne J (2002) Penetration testing: a duet In: Proceedings of the 18th Annual Computer Security Applications Conference, 185–195. IEEE.
[3] Bishop M (2007) about penetration testing. IEEE Secur Priv 5(6): 84–87.
[4] An Introduction to HTTP Response Headers for Secuirty https://securityintelligence.com/an-introduction-to-http-response-headers-for-security/
[5] Hardening your HTTP security headers by Brian Jackson https://www.keycdn.com/blog/http-security-headers
[6] https://geekflare.com/http-header-implementation /
[7] Security Response Headers Every Security Tester Should Know by Johne Jacob https://medium.com/@Johne_Jacob/7-security-response-headers-every-security-tester-should-know-77576ffdfc0f
[8] https://github.com/koenbuyens/securityheaders
[9] https://infosec.mozilla.org/guidelines/web_security