Web Security
Guideline

THE NINJA COMBAT

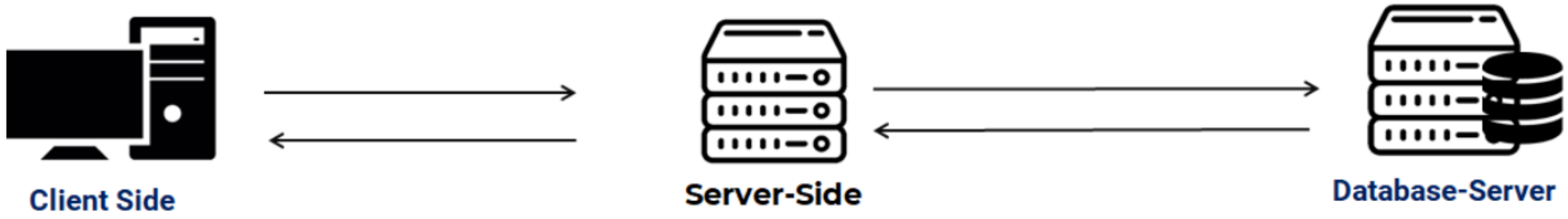
Guide line We Follow

OWASP

The Open Web Application Security Project® (OWASP) is a nonprofit foundation that works to improve the security of software.

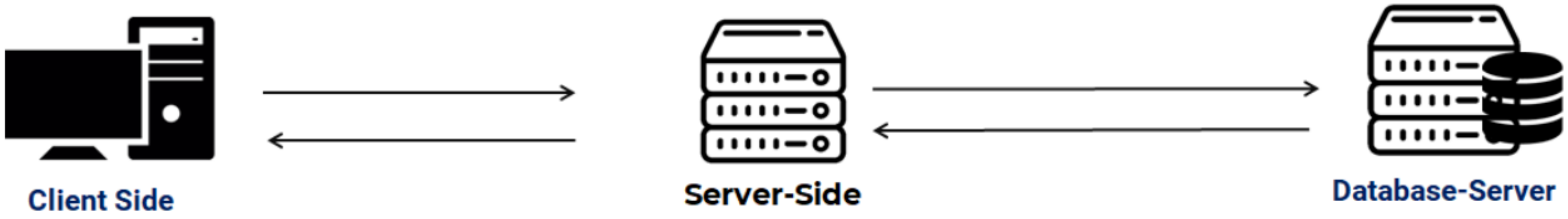
Guide line We Follow

- Secure your back-end
- Front-end security parameters will not works
- Front-end is a presentation layer, let it perform his own job



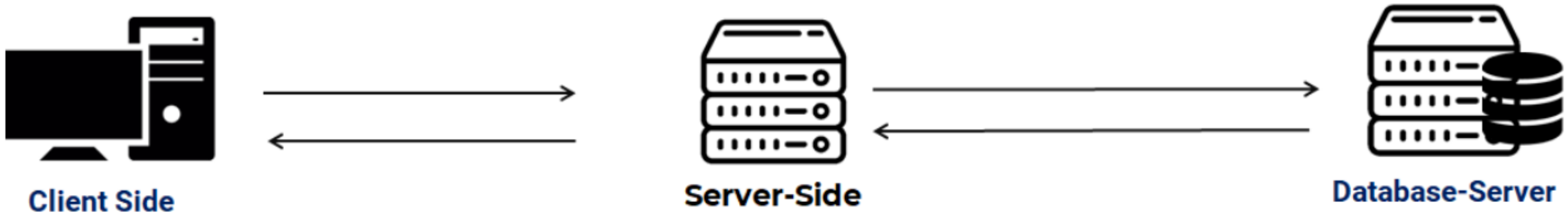
Set request size limits

- If there is no limit on the size of requests.
- Attackers can send requests with large request bodies.
- That can exhaust server memory and/or fill disk space.



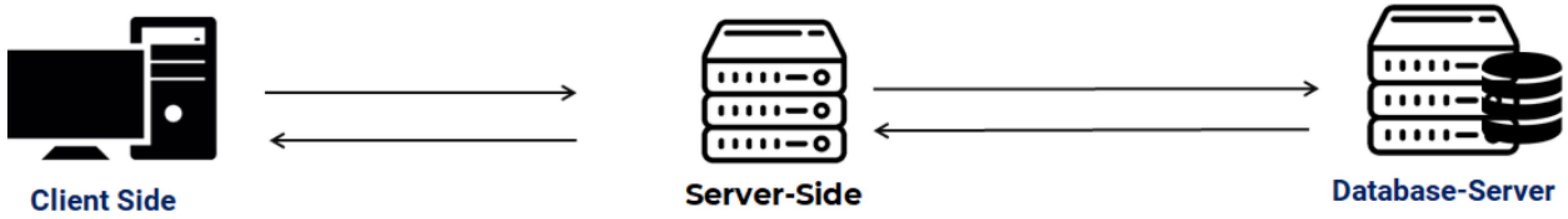
Perform input validation

- Input validation is a crucial part of application security
- Input validation failures can result in many different types of application attacks
- These include SQL Injection, Cross-Site Scripting, Command Injection
- Local/Remote File Inclusion, Denial of Service, Directory Traversal
- LDAP Injection and many other injection attacks.



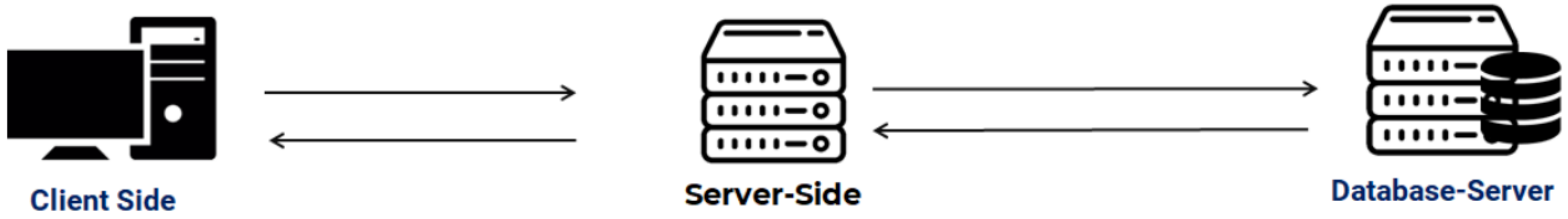
Perform output escaping

In addition to input validation, you should escape all HTML and JavaScript content shown to users via application in order to prevent cross-site scripting (XSS) attacks.



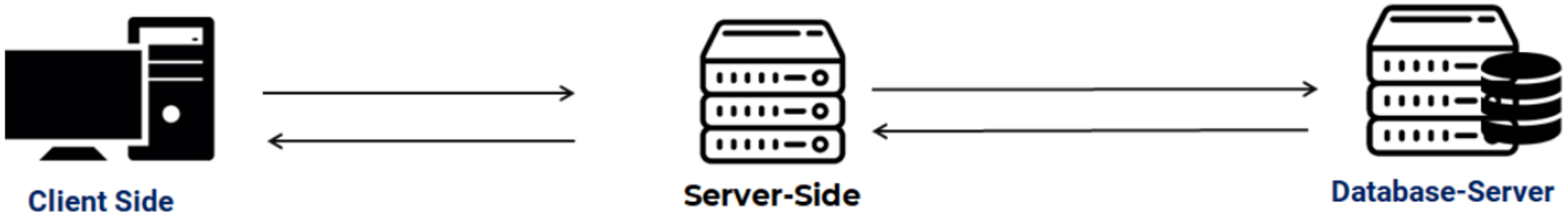
Perform application activity logging

- Logging application activity is an encouraged good practice.
- It makes it easier to debug any errors encountered during application runtime.
- It is also useful for security concerns, since it can be used during incident response.



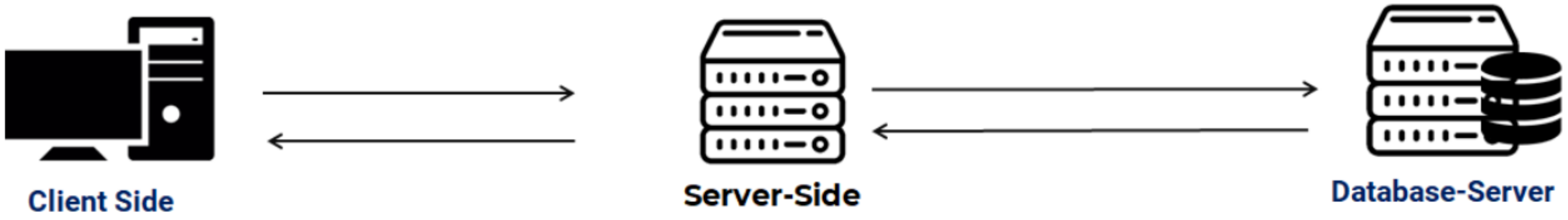
Monitor the event loop

- When your application server is under heavy network traffic.
- It may not be able to serve its users.
- This is essentially a type of Denial of Service (DoS) attack.
- Keeps track of the response time, and when it goes beyond a certain threshold.
- Stop processing incoming requests and send them 503 Server Too Busy message.



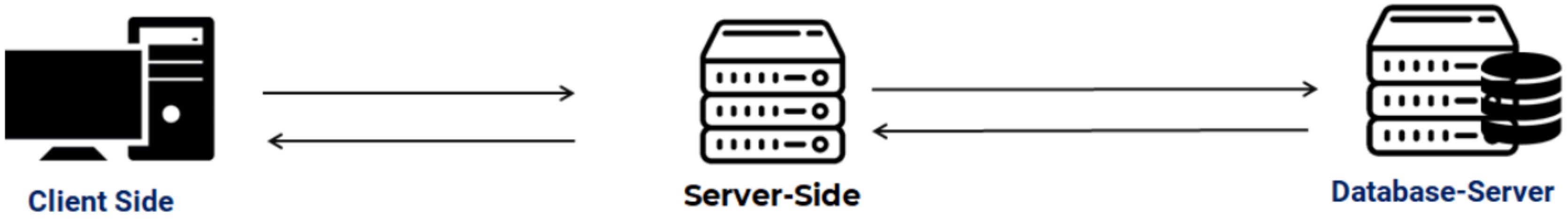
Take precautions against brute-forcing

- Brute-forcing is a common threat to all web applications.
- Attackers can use brute-forcing as a password guessing attack to obtain account passwords
- Should take precautions against brute-force attacks especially in login pages
- Enables specifying how many requests a specific IP address can make during a specified time period.
- CAPTCHA usage is also another common mechanism used against brute-forcing
- Account lockout is a recommended solution to keep attackers away from your valid users



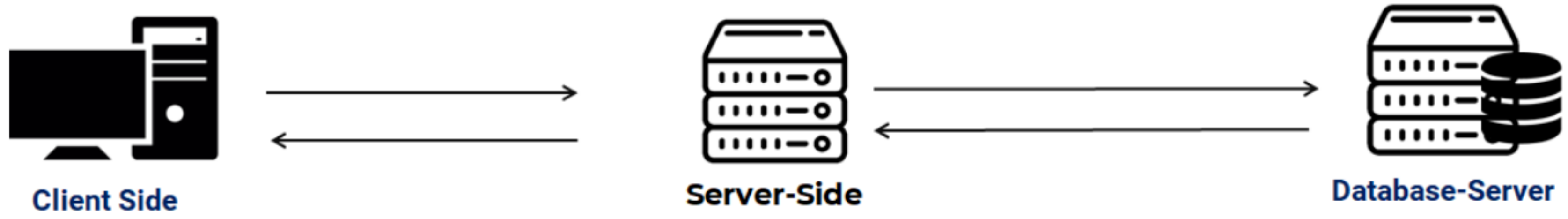
Use Anti-CSRF tokens

- Cross-Site Request Forgery (CSRF) aims to perform authorized actions on behalf of an authenticated user.
- CSRF attacks are generally performed for state-changing requests like changing a password, adding users or placing orders



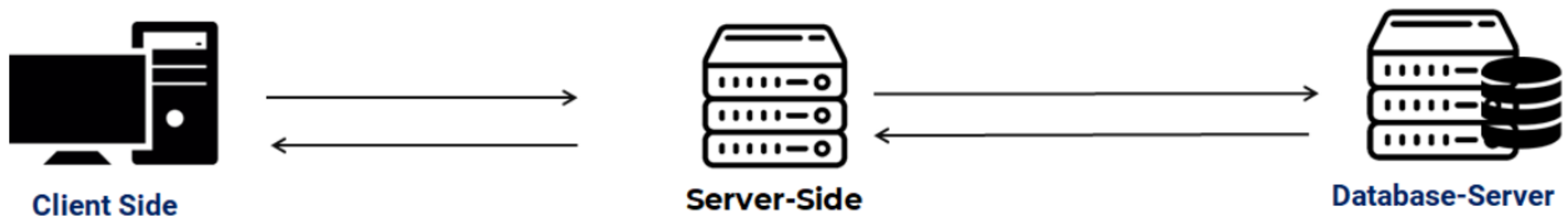
Remove unnecessary routes

- A web application should not contain any page that is not used by users
- It may increase the attack surface of the application



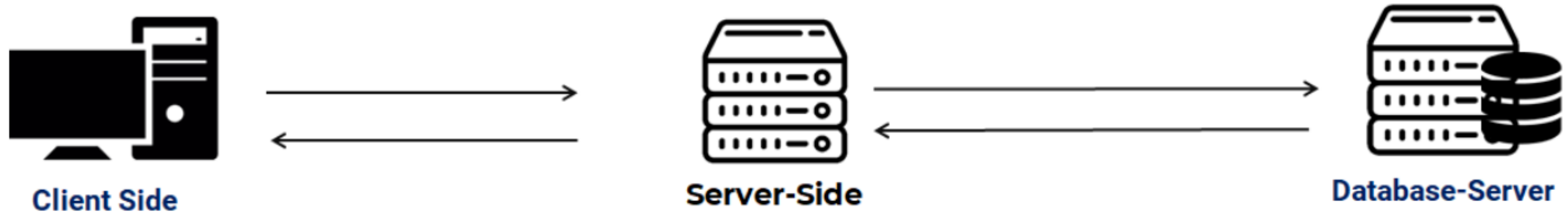
Prevent HTTP Parameter Pollution

- HTTP Parameter Pollution(HPP) is an attack in which attackers send
- Multiple HTTP parameters with the same name.
- This causes your application to interpret them in an unpredictable way



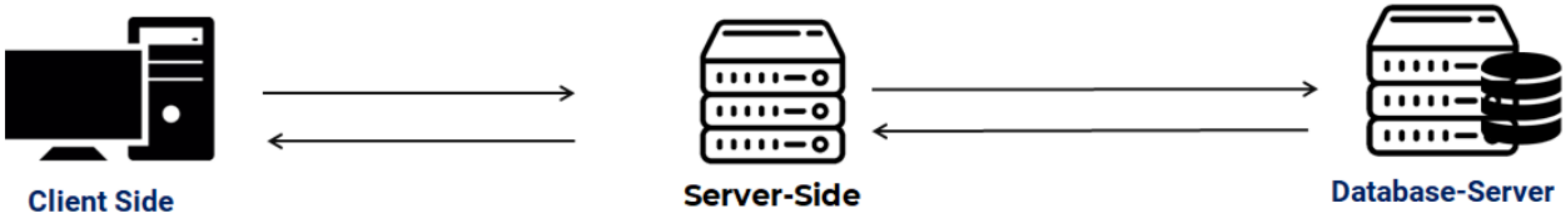
Use access control lists

- Authorization prevents users from acting outside of their intended permissions.
- Users and their roles should be determined with consideration
- Each user role should only have access to the resources they must use



Set cookie flags appropriately

- Generally, session information is sent using cookies in web applications
- Improper use of HTTP cookies can render an application to several session management vulnerabilities.
- Some flags can be set for each cookie to prevent these kinds of attacks. httpOnly, Secure and SameSite



Use appropriate security headers

Strict-Transport-Security:

Dictates browsers that the application can only be accessed via HTTPS connections.

X-Frame-Options:

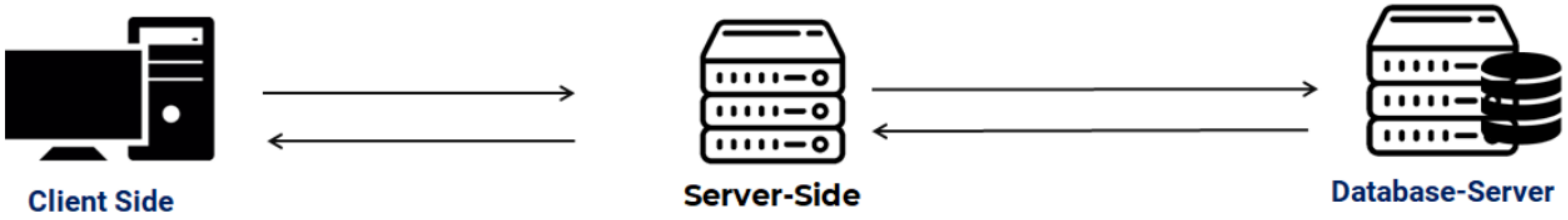
Determines if a page can be loaded via a <frame> or an <iframe> element

X-XSS-Protection::

This header should be set to 0 to disable the XSS Auditor.

X-XSS-Protection::

This header should be set to 0 to disable the XSS Auditor.



Use appropriate security headers

X-Content-Type-Options:

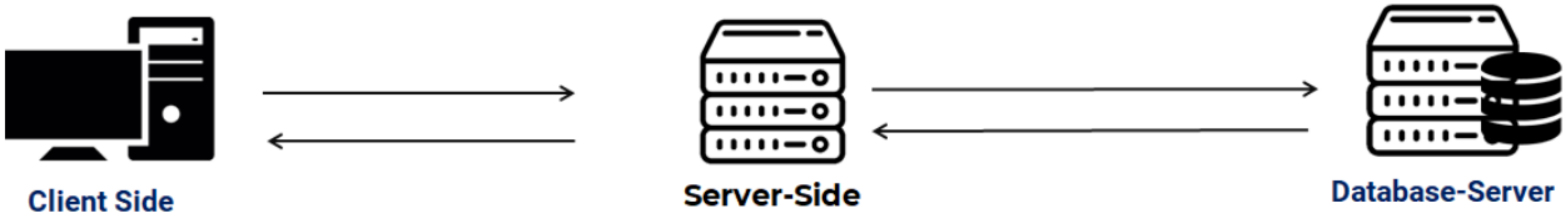
Even if the server sets a valid Content-Type header in the response, browsers may try to sniff the MIME type of the requested resource.

Content-Security-Policy:

Content Security Policy is developed to reduce the risk of attacks like Cross-Site Scripting (XSS) and Clickjacking.

Cache-Control and Pragma:

Cache-Control header can be used to prevent browsers from caching the given responses.



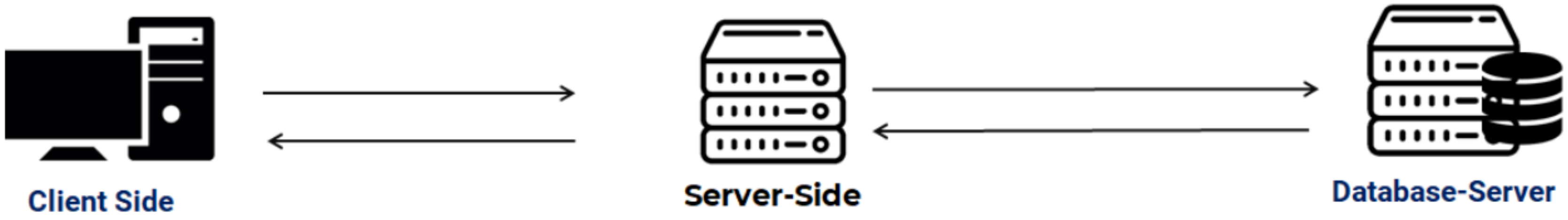
Use appropriate security headers

X-Download-Options:

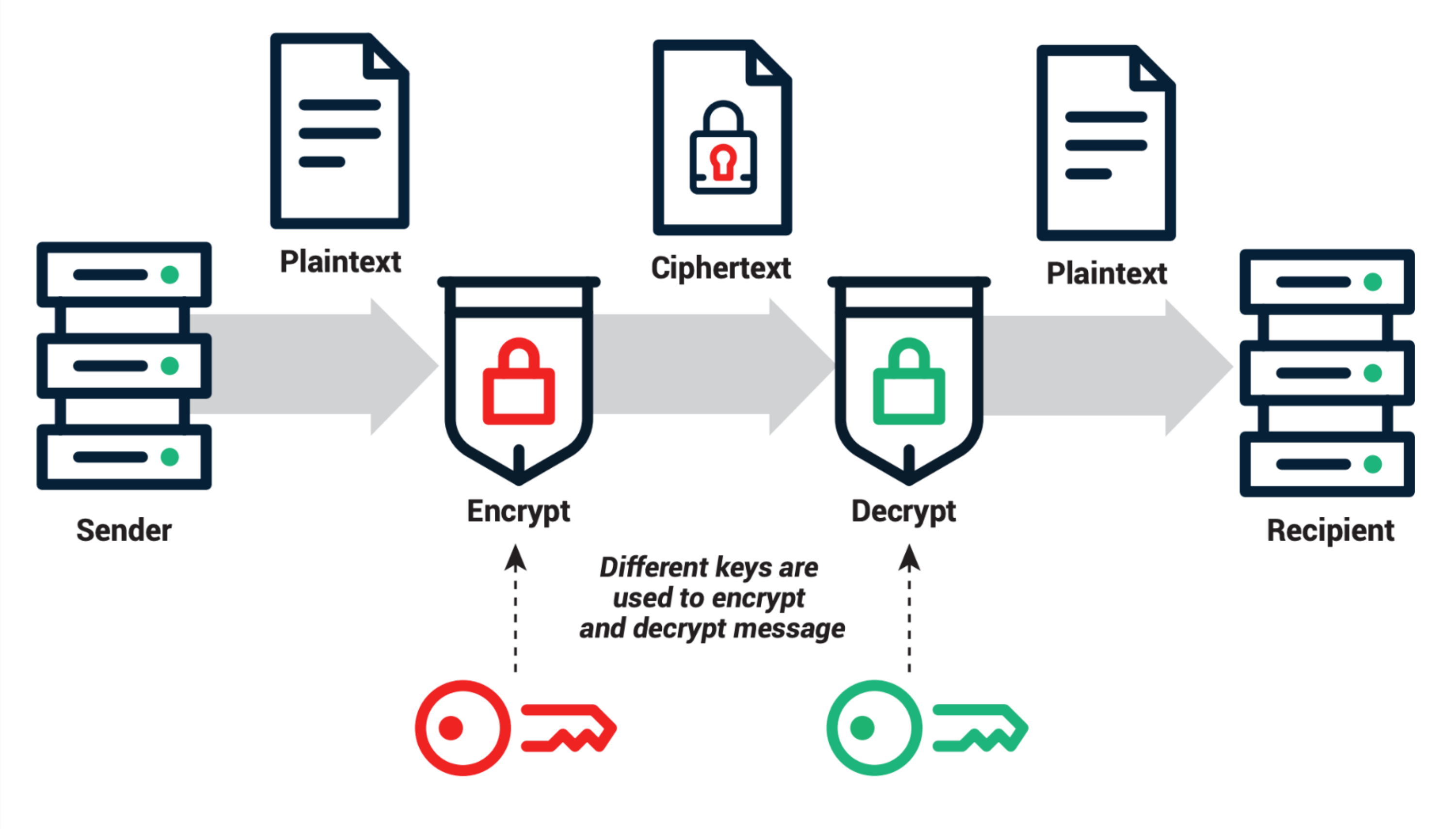
This header prevents Internet Explorer from executing downloaded files in the site's context.

X-Powered-By:

X-Powered-By header is used to inform what technology is used in the server side.



Encryption decryption public private key


Plaintext
Ciphertext
Plaintext
Sender
Encrypt
Decrypt
Recipient
Different keys are used to encrypt and decrypt message