What 1xx HTTP Status Codes Are

- **1xx codes** are **informational**. They are not final responses.
- They tell the client (browser, curl, or scanning tool) that the **request is being received and processed**.

Common codes:

Code	Meaning	Description
100	Continue	Server received initial request headers and is waiting for the rest (usually body of POST/PUT).
101	Switching Protocols	Server agrees to switch protocols (e.g., HTTP → WebSocket).
102	Processing	Server has received the request but needs more time to process.
103	Early Hints	Server sends hints (like preload links) before final response.

Why They Matter in Bug Bounty / Pentesting

- **Rarely indicate security issues directly**, but they can give **hints about server behavior**.

Examples:

1. **100 Continue**
    - Shows the server **honors the Expect: 100-continue header**.
    - Can be relevant when testing request smuggling or timing attacks.
2. **101 Switching Protocols**
    - Relevant when testing WebSocket or HTTP/2 implementations.
    - Misconfigured protocol switching can sometimes be exploited.
3. **102 Processing / 103 Early Hints**
    - Gives clues about server **workflow and headers**.
    - Can reveal caching, redirects, or preloading behavior that might leak sensitive info.

Practical Pentesting Use

- **Observation, not exploitation**: 1xx codes mainly help **understand server behavior**.
- When fuzzing or sending unusual requests (POST, PUT, HTTP smuggling), seeing 1xx responses can indicate **how the server processes large or chained requests**.

What 2xx Codes Are

- **2xx codes mean success** — the server **received, understood, and processed the request correctly**.
- They generally indicate **valid resources or successful operations**.

Common 2xx codes:

Code	Meaning	Description
200	OK	Standard response; resource exists and is returned.
201	Created	Resource successfully created (common for POST requests).
202	Accepted	Request accepted but not yet processed.
203	Non-Authoritative Information	Response from a proxy or third-party; may differ from origin.
204	No Content	Request succeeded but no content is returned.
205	Reset Content	Client should reset the document view.
206	Partial Content	Only part of the resource is returned (used with Range headers).
207	Multi-Status	Multiple independent operations (WebDAV).
208	Already Reported	WebDAV: resource already reported in previous response.
226	IM Used	Response to a GET request for a resource that has been delta-encoded.

2️⃣ **Pentesting / Bug Bounty Use**

- **200 OK**
    - Indicates the resource exists → valid path for further testing.
    - Example: /admin returns 200 → accessible page.
- **201 Created**
    - Can indicate endpoints where you can **create resources**.
    - Relevant for testing **injection, file upload, or CSRF** vulnerabilities.
- **202 Accepted**
    - Useful for async processing endpoints. Because as the request is delayed for later to process so in this time we can send payloads and all other stuff to **test for race conditions, duplicate processing, or improper input handling**..
    - **What “async processing endpoints” means**

- **Async (asynchronous) processing** = the server **accepts a request but doesn't complete it immediately**.
- Instead, it queues or schedules the task to be done later.
- Example:
- You send a **POST request** to /submit-order.
- Server responds **202 Accepted** immediately.
- The actual processing (creating the order) happens **in the background**.
- ---
- **Why this matters for pentesting**
- **Timing & race conditions**
- Because the task isn't done immediately, there's a window where **state is in transition**.
- Attackers can try sending **multiple requests quickly** to see if the server mishandles concurrent processing.
- Could lead to **duplicate orders, data leaks, or inconsistent state**.
- **Testing input handling before full processing**
- Even though the server returns 202, your input is **accepted**.
- You can try **malicious payloads** to see if background processing sanitizes them properly.
- **Monitoring side effects**
- Since the request is queued, you might check **where the result goes** (database, logs, email).
- Useful to detect **unauthorized data exposure**.
-
- Check how server handles delayed tasks → potential for **race conditions** or **data leaks**.

- **204-> This is key for discovering hidden actions, silent functionality, or flaws. / 205 No Content**
    - Useful to see **endpoint behavior**.
    - Can test **side effects**, like POST request causing a silent change.
    - When we say **"useful to see endpoint behavior"** in pentesting, it means:
    - Some HTTP responses, like **204 No Content** or **205 Reset Content**, **don't return a page or data**, but the server **still performed an action**.

- For pentesters, this is interesting because even though nothing is visible in the response, the **server might have changed something internally**.
- ---
- **Examples**
- **POST request to update a profile**
- POST /update-email
- Response: 204 No Content
- Nothing shows in the browser.
- But if you check the database or try to view your profile, the **email was actually updated**.
- Pentester can notice: "This endpoint performs an action silently" → good for **testing CSRF, access control, or data manipulation**.
- **DELETE request**
- DELETE /user/123
- Response: 204 No Content
- Server confirms the deletion **without sending a page**.
- Pentester can try to see if **unauthorized deletion is possible** → security flaw.
- **Reset / side effects**
- 205 Reset Content tells the **client to reset the form/view**.
- Pentesters can check **how the server handles these "silent" instructions** → sometimes it exposes logic flaws.
- ---
- ✅ **Key Idea**
- Even if a response **doesn't show content**, it **reveals that the server processed your request**.
- This helps pentesters **understand endpoints** that are performing hidden actions.
- These endpoints can then be tested for **vulnerabilities** (like CSRF, improper permissions, or unintended data changes).

Why this is useful for pentesting / bug bounty

- **Testing for Side Effects**
  - Even if the server returns **204 No Content**, the request **may have changed something** on the server.
  - Example:
    - Sending `POST /update-profile` returns 204.

        - You can check if the profile actually changed → could reveal **authorization flaws**.
- **Finding Hidden Functionality**
    - Some endpoints silently process data.
    - Example: `/delete-account` might return 204 → you can check if the action was performed without an explicit confirmation.
    - This helps **discover hidden features** you can test for misuse.
- **Race Conditions / Timing Attacks**
    - Endpoints that return 202 Accepted or 204 may **process requests asynchronously**.
    - Pentesters can test for **double submissions**, **race conditions**, or **data leakage**.
- **Silent Vulnerability Confirmation**
    - Some attacks don't produce output.
    - Example: Testing **CSRF or SSRF** might not return a body, but the server's response code tells you it processed the request.
    - Observing 2xx codes confirms **your payload reached the server**.

- **206 Partial Content**
    - Indicates **support for Range requests** → potential vector for **HTTP request smuggling** or **partial download attacks**.
    - **What "Range Requests" Are**
    - Normally, when you request a file via HTTP (like GET /file.zip), the server sends the **entire file**.
    - A **Range Request** allows you to ask for **only a part of the file**.
    - Example HTTP request:
    - GET /file.zip HTTP/1.1
    - Host: example.com
    - Range: bytes=0-499
    - The server responds with the first 500 bytes instead of the whole file.
    - Status code **206 Partial Content** indicates the server supports this feature.
    - ---
    - 2️⃣ **Why This Matters for Pentesting / Bug Bounty**
    - **a. Check for Misconfigured Servers**
    - Some servers **don't handle partial requests properly**.
    - Improper handling can allow attackers to:
    - Read **sensitive parts of files** without full authorization.
    - Bypass security checks for downloads or resources.

- **b. HTTP Request Smuggling**
- If a server supports **partial or chunked requests**, it might process requests differently than a **front-end proxy or load balancer**.
- Attackers can send **carefully crafted Range headers** to trick the server into mixing requests or bypass security → called **request smuggling**.
- Example scenario:
- Proxy sees one request.
- Server interprets two requests because of Range or Content-Length mismatch → can bypass authentication.
- **c. Partial File Leaks**
- Some web apps store sensitive files (like configs, backups).
- Using **range requests**, you can test if **partial content can be accessed** even without full permissions.
- ---
- 3️⃣ **Practical Pentesting Example**
- **Identify a file that returns 206 Partial Content**
- curl -I http://example.com/secret.zip
- # Response: 206 Partial Content
- **Request only part of the file**
- curl -r 0-499 http://example.com/secret.zip -o part1.zip
- curl -r 500-999 http://example.com/secret.zip -o part2.zip
- **Check if you can piece together the file**
- If the server allows **unauthorized partial downloads**, this is a **data exposure vulnerability**.
-

- **207 / 208 / 226**
  - Mostly WebDAV or advanced HTTP features → can be checked for misconfigurations or **file exposure**.
  - Example: an exposed WebDAV endpoint might allow attackers to **list, download, or upload files**.
  - These codes mostly appear in **advanced HTTP features like WebDAV or caching**. For pentesting, they **hint at endpoints that could expose files, directories, or partial content**, making them worth investigating.

3️⃣ Key Pentesting Takeaways

- **2xx codes usually indicate a valid resource or action succeeded.**
- **Scan results**: any 2xx response is worth looking at closely — it's a potential **entry point**.

What 3xx Codes Are

- **3xx = Redirection**
- The server is telling the client:

"The resource you requested exists somewhere else, follow this new location to get it."

- Often used to **move, mask, or hide content**.

Common 3xx Codes and Meaning

Code	Meaning	Description / Pentesting Use
300	Multiple Choices	Several options for the resource; could hint at **alternative paths or endpoints**.
301	Moved Permanently	Resource has a permanent new URL. Useful for **tracking old endpoints or hidden pages**.
302	Found (Temporary)	Resource temporarily moved. Common for login redirects (/admin → /login).
303	See Other	Redirects after POST requests; useful to see **where forms lead**.
304	Not Modified	Resource hasn't changed; mostly caching info. Less useful directly for pentesting.
305	Use Proxy	Deprecated; tells client to use a proxy. Could hint at **internal network usage** if ever encountered.
306	Switch Proxy	No longer used; usually safe to ignore.
307	Temporary Redirect	Like 302 but preserves request method (POST stays POST). Check for **login/form redirects**.

Code	Meaning	Description / Pentesting Use
308	Permanent Redirect	Like 301 but preserves method. Can indicate **hidden or permanent moved resources**.

2️⃣ Why 3xx Codes Are Useful in Pentesting

1. **Hidden or masked content**
   - Redirects often **point to admin panels, old backups, or restricted directories**.
   - Example: /admin → 302 → /login → confirms /admin exists.
2. **Access control hints**
   - If you get redirected to a login page or 403 page, it **shows protections are in place**.
3. **Workflow analysis**
   - Observing redirects helps you **understand how forms, POST requests, or session handling works**.
4. **Follow-up attacks**
   - 3xx redirects can reveal **forgotten endpoints** (e.g., /old-site/backup/) or **default admin pages**.

3️⃣ How Pentesters Use 3xx in Scans

- With **dirb or gobuster**, note paths that return 301/302/307/308.
- **Check the Location header** to see the actual target URL.
- Combine with **authentication testing**: if redirect goes to login, analyze login behavior.
- Example:

```
dirb http://example.com/

# discovers /admin → 302 → /login
```

✅ In short:

3xx = Redirection. In pentesting, redirects **confirm resource existence, show hidden paths, and indicate access controls**. Always **follow the Location header** to see where it leads.

If you want, I can continue with **4xx (Client Errors) and 5xx (Server Errors) codes with pentesting tips**, so you get a full HTTP status code cheat sheet.

Do you want me to do that?

What 4xx Codes Are

- **4xx = Client errors** → the server received the request but **refuses or cannot process it** due to client-side issues.
- These are **extremely valuable in pentesting**, because they reveal **how the server validates input, controls access, and enforces restrictions**.

2️⃣ Common 4xx Codes and Pentesting Use

Code	Meaning	Pentesting Relevance
400	Bad Request	Often triggered by malformed input → can hint at **input validation flaws**, injection points, or fuzzing opportunities.
401	Unauthorized	Requires authentication → useful for **testing login bypass, token flaws, or misconfigurations**.
403	Forbidden	Resource exists but **access is denied** → check for **privilege escalation, hidden endpoints, or bypass techniques**.
404	Not Found	Resource doesn't exist → helps **map the directory structure** during brute-forcing or scanning.
405	Method Not Allowed	Shows which HTTP methods are blocked → can try **PUT, DELETE, PATCH** to expand attack surface.
409	Conflict	Usually related to **state or resource conflicts** → could reveal **logic flaws or concurrency issues**.
410	Gone	Resource intentionally removed → sometimes old or cached endpoints are still reachable.
412	Precondition Failed	Server rejected request due to conditions → can expose **logic or validation flaws**.
413	Payload Too Large	Shows server limits → useful to test **buffer overflow, large payload handling, or DoS**.
422	Unprocessable Entity	Input failed validation → good for **injection testing, XSS, or malformed input bugs**.
429	Too Many Requests	Indicates **rate limiting** → check **bypass methods or DoS protections**.
451	Unavailable for Legal Reasons	Geo-blocked content → useful in **testing access controls and regional restrictions**.

3️⃣ Key Pentesting Takeaways

1. **404 is your friend** for directory/file brute-forcing (like with dirb/gobuster).
2. **401/403** → critical for **testing access control flaws**.
3. **400/422/413** → show **input validation weaknesses**, prime candidates for injection/XSS testing.
4. **405** → helps identify **alternative attack vectors** using different HTTP methods.
5. **429/451** → hints about **rate-limiting and geo-restriction mechanisms**.

✅ In short:

4xx responses tell you **what the server rejects and why**, which is extremely valuable for **finding vulnerabilities and hidden paths**. Always note **response codes and headers** when scanning.

What 5xx Codes Are

- **5xx = Server errors** → the server **failed to process a valid request** due to an internal problem.
- These are valuable because they often **reveal server misconfigurations, bugs, or exploitable points**.

2️⃣ Common 5xx Codes and Pentesting Use

Code	Meaning	Pentesting Relevance
500	Internal Server Error	**High-value code** → often triggered by malformed input or injection attempts. Can reveal **SQL, command injection, or crash points**.
502	Bad Gateway	Shows upstream server or proxy issues → may hint at **reverse proxy misconfigurations**.
503	Service Unavailable	Server overloaded or under maintenance → can be tested for **DoS potential or rate-limiting bypass**.
504	Gateway Timeout	Indicates slow upstream responses → may expose **timing attack vectors**.
507	Insufficient Storage	Shows **resource exhaustion** → can be exploited in **DoS attacks or file upload testing**.

Code	Meaning	Pentesting Relevance
508	Loop Detected	Server caught in **infinite loop** → can indicate **logic flaws**.

3️⃣ Key Pentesting Takeaways

1. **500 Internal Server Error** → usually the **highest yield** for finding bugs.
    - Example: sending unexpected input might trigger an error revealing **stack traces or database info**.
2. **502 / 503 / 504** → test **DoS, timing, or proxy-related attacks**.
3. **507 / 508** → observe how the server handles **resource limits** → potential for **exhaustion attacks or logic flaws**.

✅ In short:

5xx responses are gold for bug bounty. They **show internal problems** in the server or application, which can be leveraged to discover **vulnerabilities, misconfigurations, or crash points**.

504 Gateway Timeout

What it is:

- Occurs when a server (usually a reverse proxy, load balancer, or gateway) **cannot get a timely response from an upstream server**.
- Example: Client → Proxy → Backend Server → Timeout → 504

Pentesting relevance:

1. **Timing attacks**
    - Slow responses may indicate that the server is processing input differently depending on conditions.
    - Can be exploited to detect **information leakage**, like password length, SQL injection timing, or brute-force detection.
2. **Server behavior under stress**
    - If certain inputs cause slow processing → might reveal **resource-heavy endpoints**.
    - Useful for **DoS testing**, but only on authorized targets.

2️⃣ 507 Insufficient Storage

What it is:

- Server cannot store the resource needed to complete the request.
- Often happens with **file upload endpoints, database writes, or disk space issues**.

Pentesting relevance:

1. **Resource exhaustion attacks**
    - Repeated large uploads or requests could fill server storage → potential **Denial of Service (DoS)**.
2. **File upload testing**
    - Shows endpoints that accept files or large data → check if there's **improper validation**, **path traversal**, or **arbitrary file upload** vulnerability.

💡 **Why Bug Bounty Hunters Care**

- **504** → timing, information leakage, or server logic flaws.
- **507** → endpoints that accept user data and may be misconfigured → chance to **exploit uploads or storage handling**.

✅ **In short:**

504 = shows **slow backend behavior**, useful for timing or logic flaws.
507 = shows **resource/storage handling**, useful for DoS or file upload vulnerability testing.