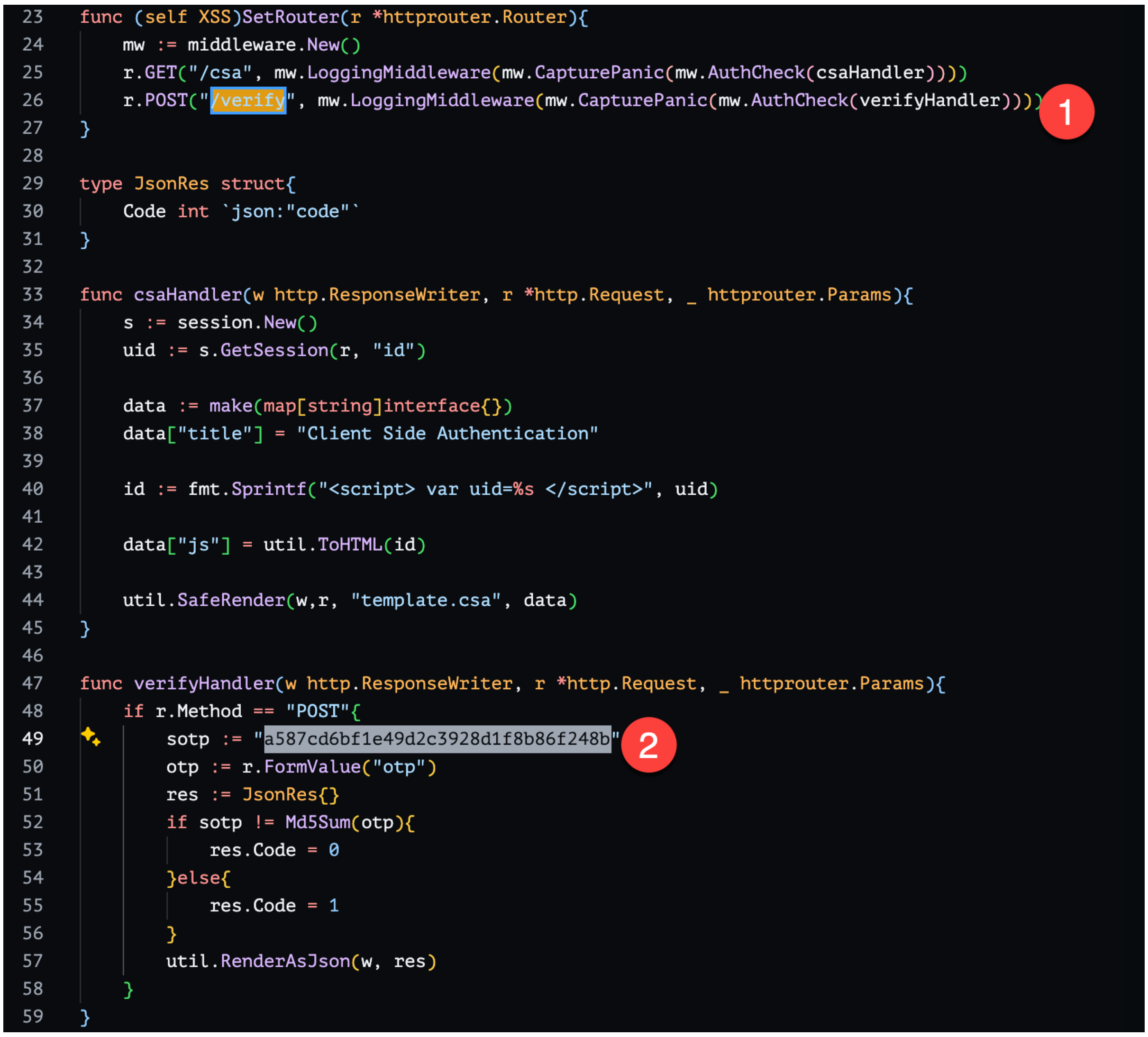
Client-side authentication

Relevant code is on lines 26 and 49, where there's a hardcoded MD5 hash. To handle it:

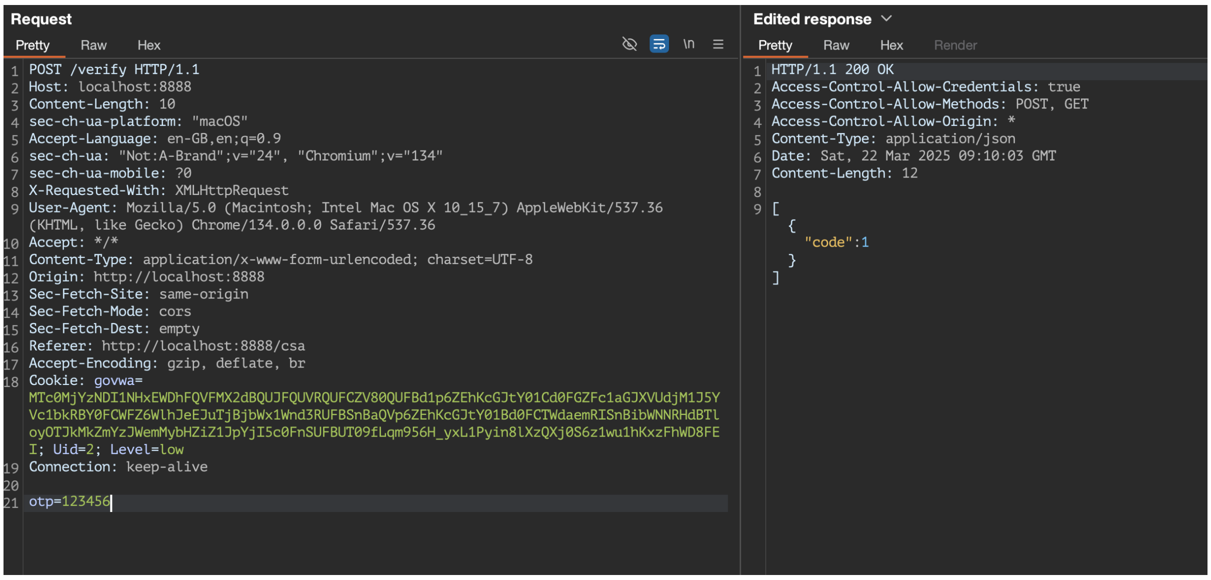
1. Manipulate the backend response.
2. Use CrackStation to get the hash.



```
23  func (self XSS)SetRouter(r *httprouter.Router){
24      mw := middleware.New()
25      r.GET("/csa", mw.LoggingMiddleware(mw.CapturePanic(mw.AuthCheck(csaHandler))))
26      r.POST("/verify", mw.LoggingMiddleware(mw.CapturePanic(mw.AuthCheck(verifyHandler))))
27  }
28
29  type JsonRes struct{
30      Code int `json:"code"`
31  }
32
33  func csaHandler(w http.ResponseWriter, r *http.Request, _ httprouter.Params){
34      s := session.New()
35      uid := s.GetSession(r, "id")
36
37      data := make(map[string]interface{})
38      data["title"] = "Client Side Authentication"
39
40      id := fmt.Sprintf("<script> var uid=%s </script>", uid)
41
42      data["js"] = util.ToHTML(id)
43
44      util.SafeRender(w,r, "template.csa", data)
45  }
46
47  func verifyHandler(w http.ResponseWriter, r *http.Request, _ httprouter.Params){
48      if r.Method == "POST"{
49          sotp := "a587cd6bf1e49d2c3928d1f8b86f248b"
50          otp := r.FormValue("otp")
51          res := JsonRes{}
52          if sotp != Md5Sum(otp){
53              res.Code = 0
54          }else{
55              res.Code = 1
56          }
57          util.RenderAsJson(w, res)
58      }
59  }
```

Option A

Consider the modified reply, and change the number 0 to 1.



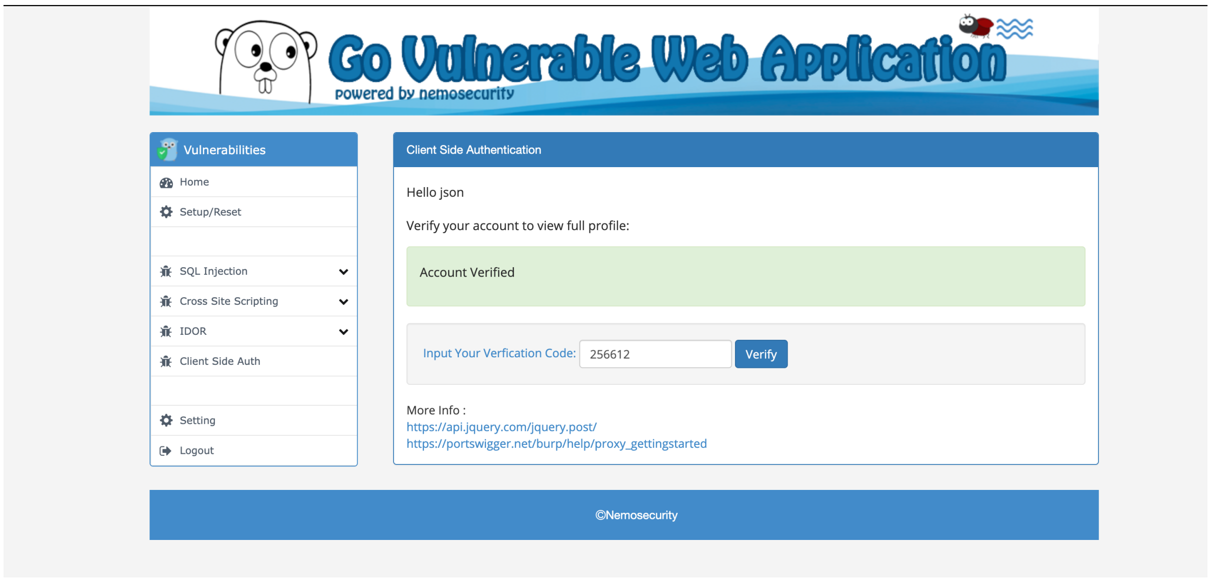
```
POST /verify HTTP/1.1
Host: localhost:8888
Content-Length: 10
sec-ch-ua-platform: "macOS"
Accept-Language: en-GB,en;q=0.9
sec-ch-ua: "Not:A-Brand";v="24", "Chromium";v="134"
sec-ch-ua-mobile: ?0
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Origin: http://localhost:8888
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:8888/csa
Accept-Encoding: gzip, deflate, br
Cookie: govwa=
MTc0MjYzNDI1NHxEWDhFQVFMX2dBQUJFQUVRQUFCZV80QUFBd1p6ZEhKcGJtY01Cd0FGZFc1aGJXVUdjM1J5Y
Vc1bkRBY0FCWFZ6WlhJeEJuTjBjbWx1Wnd3RUFBSnBaQVp6ZEhKcGJtY01Bd0FCTWdaemRISnBibWNNRHdBTl
oyOTJkMkZmYzJWemMybHZiZ1JpYjI5c0FnSUFBUT09fLqm956H_yxL1Pyin8lXzQXj0S6z1wu1hKxzFhWD8FE
I; Uid=2; Level=low
Connection: keep-alive

otp=123456
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: POST, GET
Access-Control-Allow-Origin: *
Content-Type: application/json
Date: Sat, 22 Mar 2025 09:10:03 GMT
Content-Length: 12

[
  {
    "code":1
  }
]
```

Account will subsequent be verified based on the edited response.



Hello json

Verify your account to view full profile:

Account Verified

Input Your Verfication Code: 256612 Verify

More Info :
https://api.jquery.com/jquery.post/
https://portswigger.net/burp/help/proxy_gettingstarted

©Nemosecurity

Option B

Visit Crack station and input the hash. This underscores the need for salting and hashing, as pre-computed tables can reveal results instantly.

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
a587cd6bf1e49d2c3928d1f8b86f248b
```

I'm not a robot

reCAPTCHA
Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
a587cd6bf1e49d2c3928d1f8b86f248b	md5	256612

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Download CrackStation's Wordlist

How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our hashing security page.

Crackstation's lookup tables were created by extracting every word from the Wikipedia databases and adding with every password list we could find. We also applied intelligent word mangling (brute force hybrid) to our wordlists to make them much more effective. For MD5 and SHA1 hashes, we have a 190GB, 15-billion-entry lookup table, and for other hashes, we have a 19GB 1.5-billion-entry lookup table.

You can download CrackStation's dictionaries here, and the lookup table implementation (PHP and C) is available here.