

Broken Access Control



Tackling OWASP's #1 Vulnerability



#whoami

Spas Genov

Senior QA Automation @NewsUK

Security enthusiast

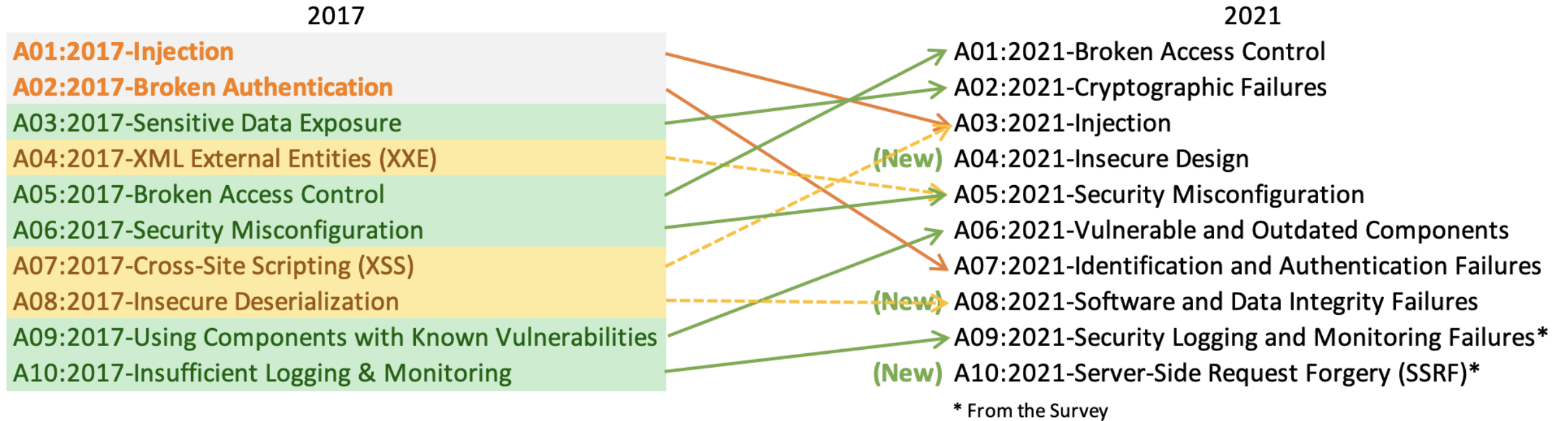


#whoami

/Spas-Genov (LinkedIn)
Senior QA Automation @NewsUK
W4nn4b3 h4x0r (eWPT, eJPT)



OWASP Top 10:2021



Terminology Confusion

Broken Access
Control (BAC)

Missing Function
Level Access
Control (MFLAC)

Broken Object
Level Access (BOLA)
(aka)

Forced
browsing



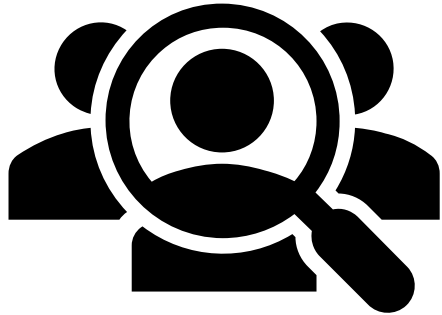
Terminology Confusion

- Broken Access Control (BAC)
- Missing Function Level Access Control (MFLAX)
- Broken Object Level Access (BOLA)
- Insecure Direct Object Reference (IDOR)
- Forceful browsing
- Authorization/Authentication issues

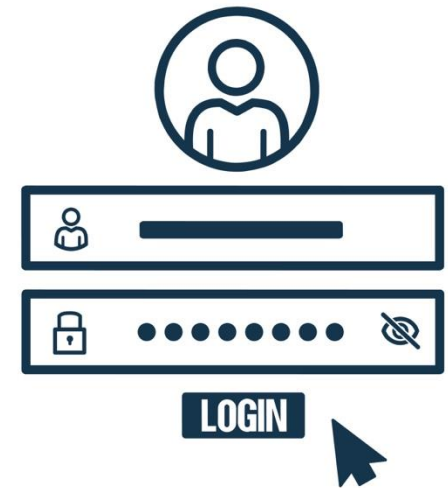


Core Concepts

- **Authentication** - Verifies a user's identity (through username/password)

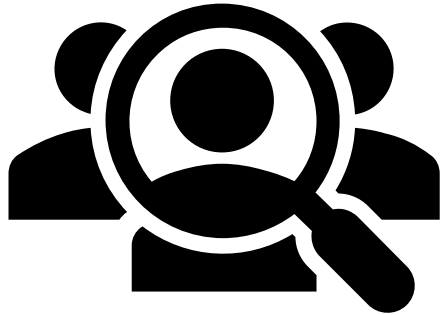


Username: admin
Password: sexylubo99



Core Concepts

- **Session Management** - Maintains the user's authenticated state across requests (using session tokens)



Username: admin
Password: sexylubo99



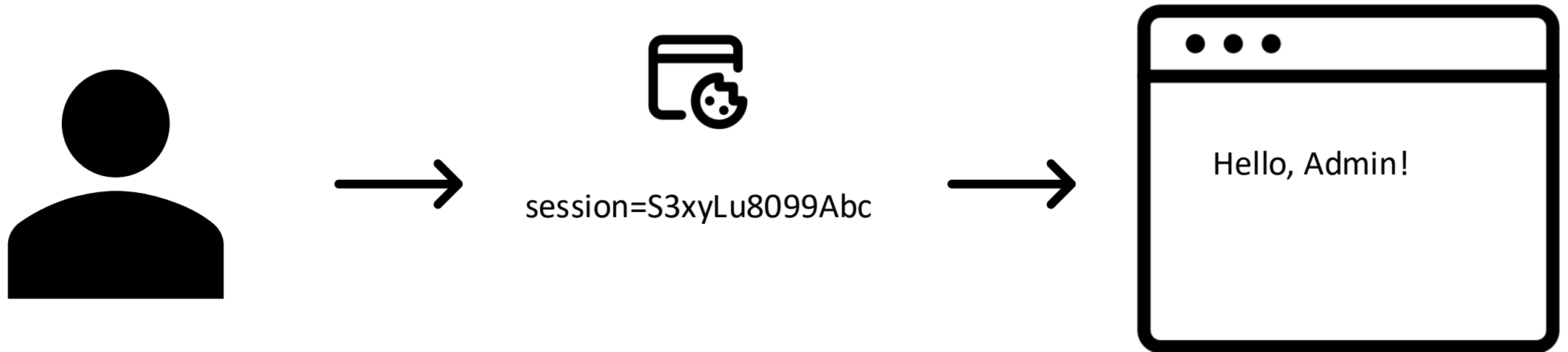
Core Concepts

- **Session Management** - Maintains the user's authenticated state across requests (using session tokens)



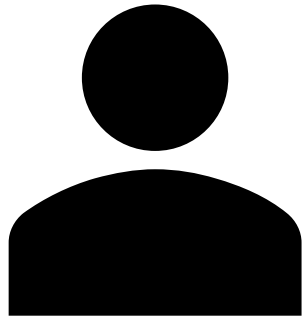
Core Concepts

- **Session Management** - Maintains the user's authenticated state across requests (using session tokens)



Core Concepts

- **Access Control** - Determines if an authenticated user has permission to perform specific actions

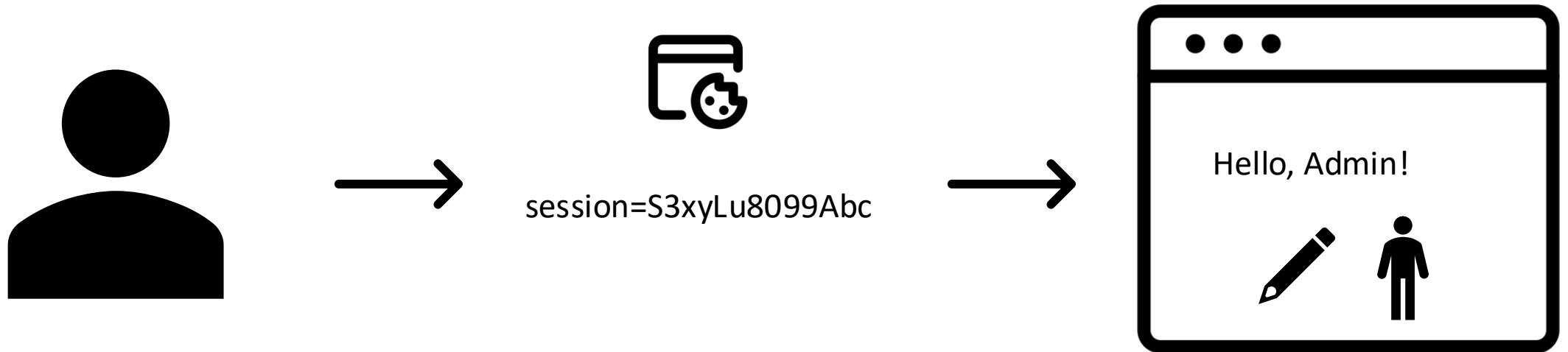


session=S3xyLu8099Abc



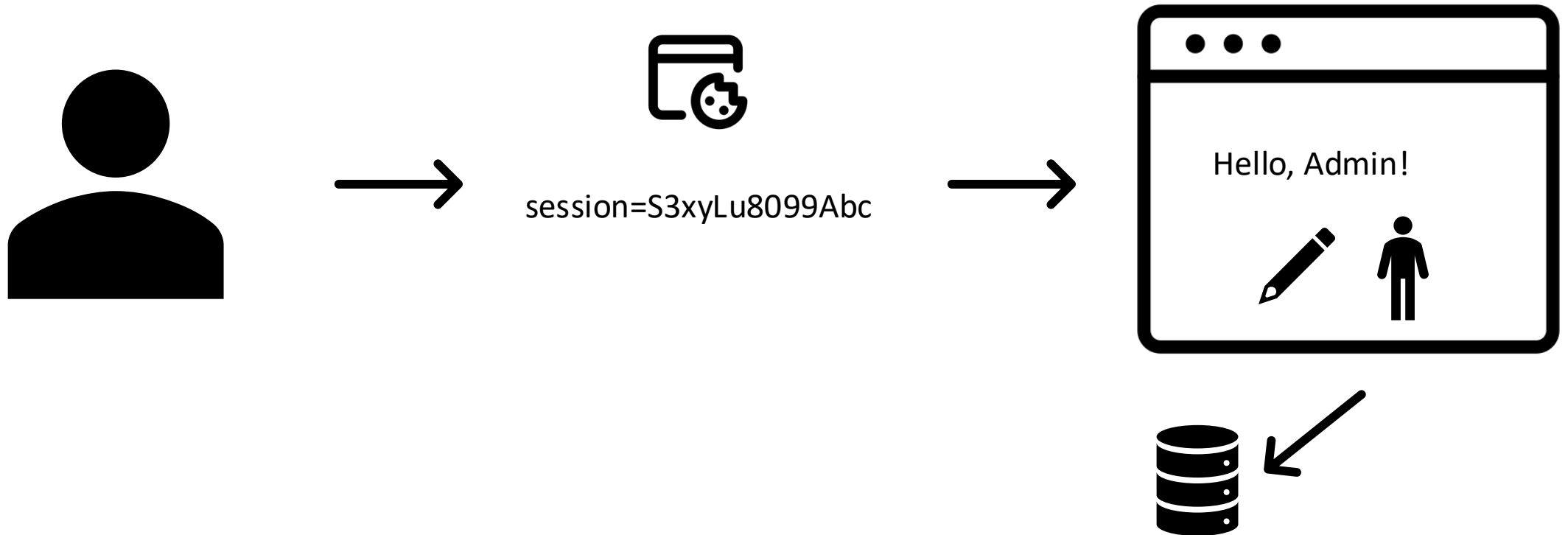
Core Concepts

- **Access Control** - Determines if an authenticated user has permission to perform specific actions



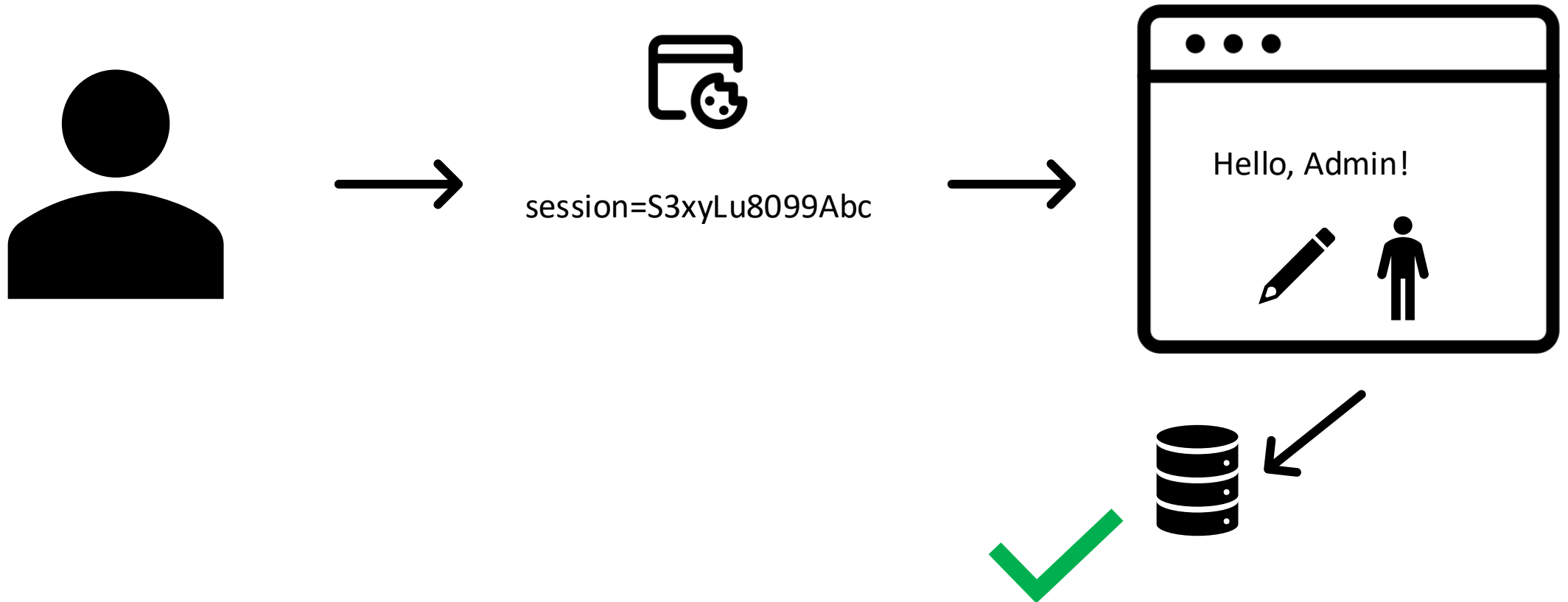
Core Concepts

- **Access Control** - Determines if an authenticated user has permission to perform specific actions



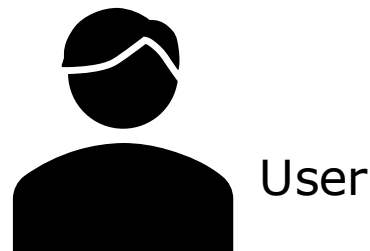
Core Concepts

- **Access Control** - Determines if an authenticated user has permission to perform specific actions



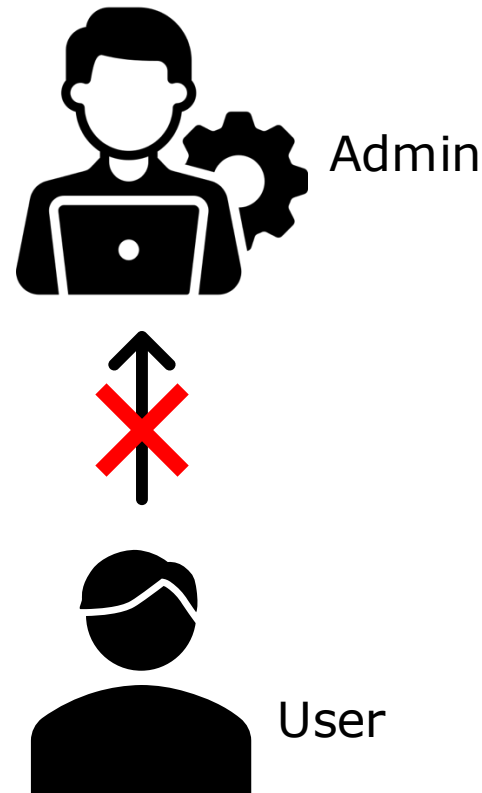
Types of Access Control

- **Vertical Access Control:** Restricts access based on privilege levels (e.g., admin vs. regular user)



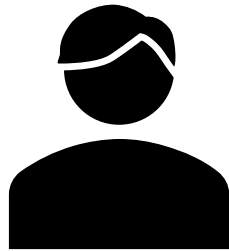
Types of Access Control

- **Vertical Access Control:** Restricts access based on privilege levels (e.g., admin vs. regular user)

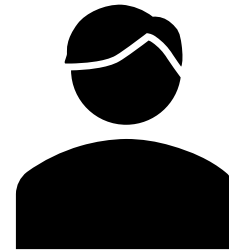


Types of Access Control

- **Horizontal Access Control:** Prevents users of the same privilege level from accessing each other's resources



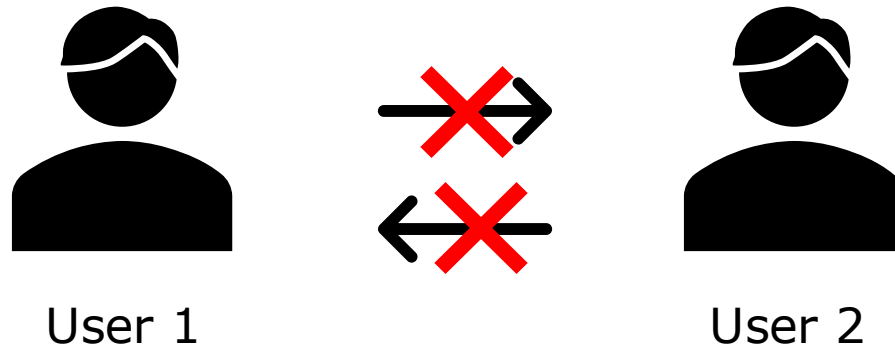
User 1



User 2

Types of Access Control

- **Horizontal Access Control:** Prevents users of the same privilege level from accessing each other's resources



Types of Access Control

- **Context-Dependent Access Control:** Restricts access to actions or resources based on application state or user interaction sequence



/cart



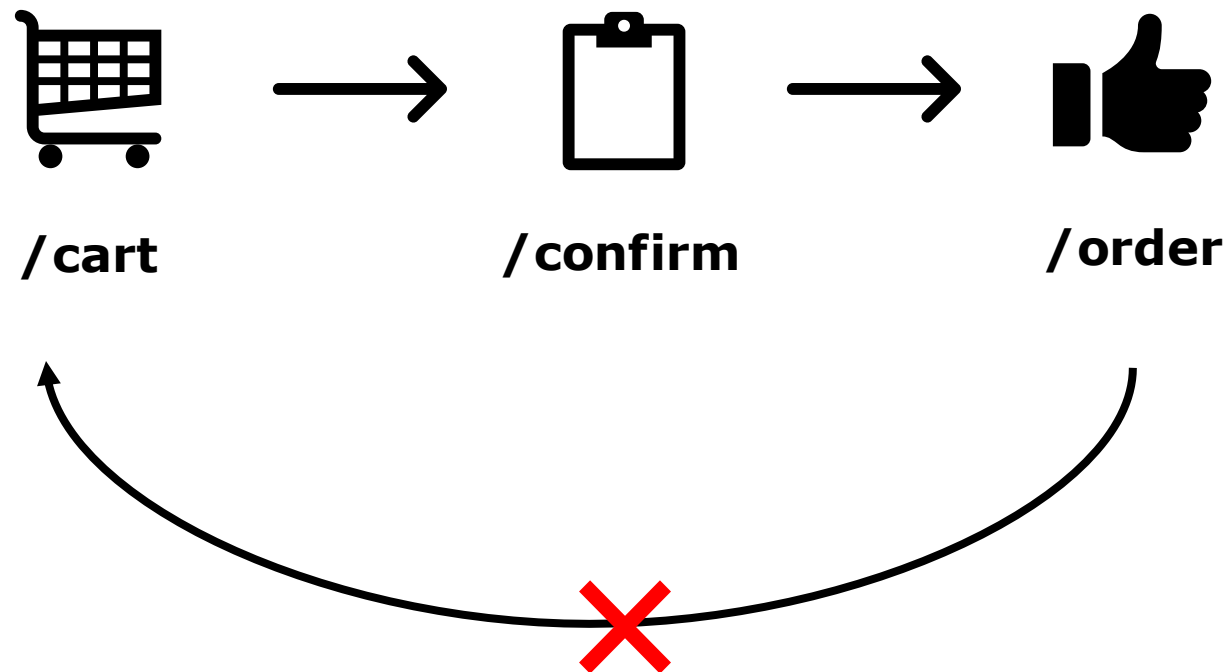
/confirm



/order

Types of Access Control

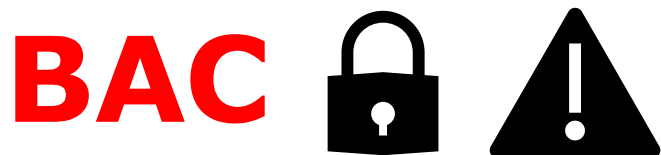
- **Context-Dependent Access Control:** Restricts access to actions or resources based on application state or user interaction sequence





Broken Access Control vulnerabilities

occur when applications fail to properly restrict what users can do, allowing them to act outside their intended permissions.



Broken Access Control vulnerabilities occur when applications fail to properly restrict what users can do, allowing them to act outside their intended permissions.

These security failures typically result in three major consequences:

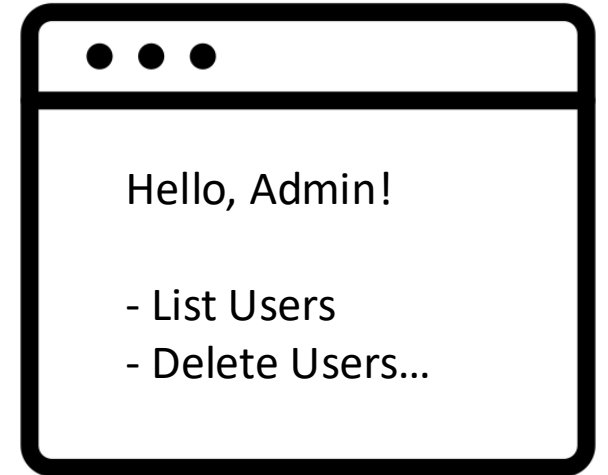
1. Improper access to restricted functionality
2. Unauthorized disclosure of sensitive information
3. Unauthorized modification or destruction of data

Vertical Privilege Escalation

- Users gain access to higher privilege functionality, for example, a regular user accessing the admin panel. Often exploited by manipulating parameters that control access



<https://site.com/login/home.php?admin=true>
<https://site.com/login/home.php?role=1>



Vertical Privilege Escalation

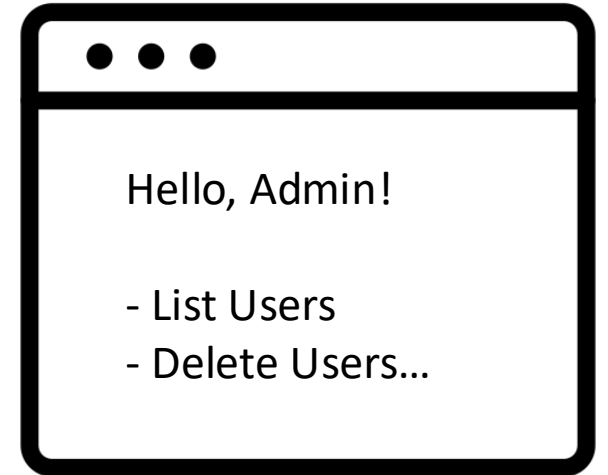
- Users gain access to higher privilege functionality, for example, a regular user accessing the admin panel. Often exploited by manipulating parameters that control access



<https://site.com/login/home.php?admin=true>
<https://site.com/login/home.php?role=1>



<https://site.com/login/home.php>
<https://site.com/login/home.php>



Vertical Privilege Escalation

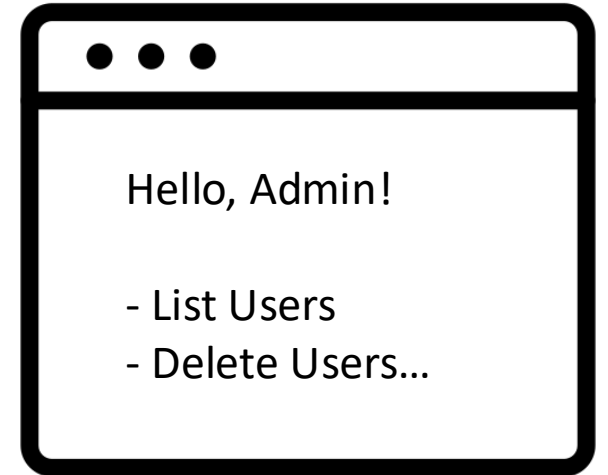
- Users gain access to higher privilege functionality, for example, a regular user accessing the admin panel. Often exploited by manipulating parameters that control access



`https://site.com/login/home.php?admin=true`
`https://site.com/login/home.php?role=1`

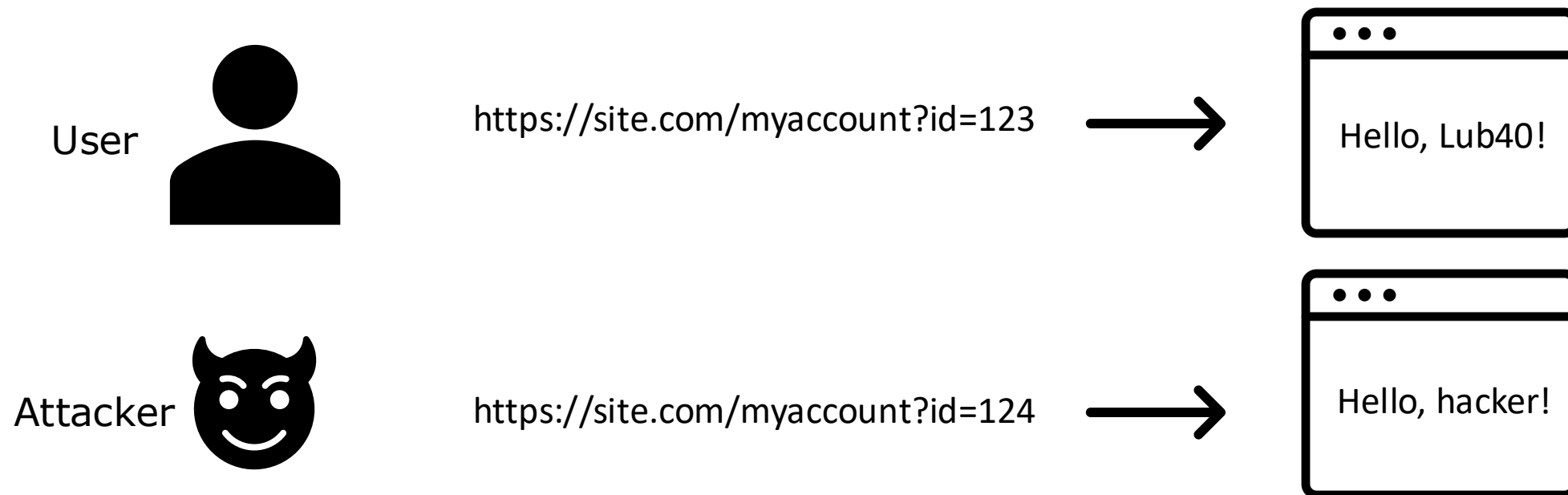


`https://site.com/login/home.php?admin=true`
`https://site.com/login/home.php?role=1`



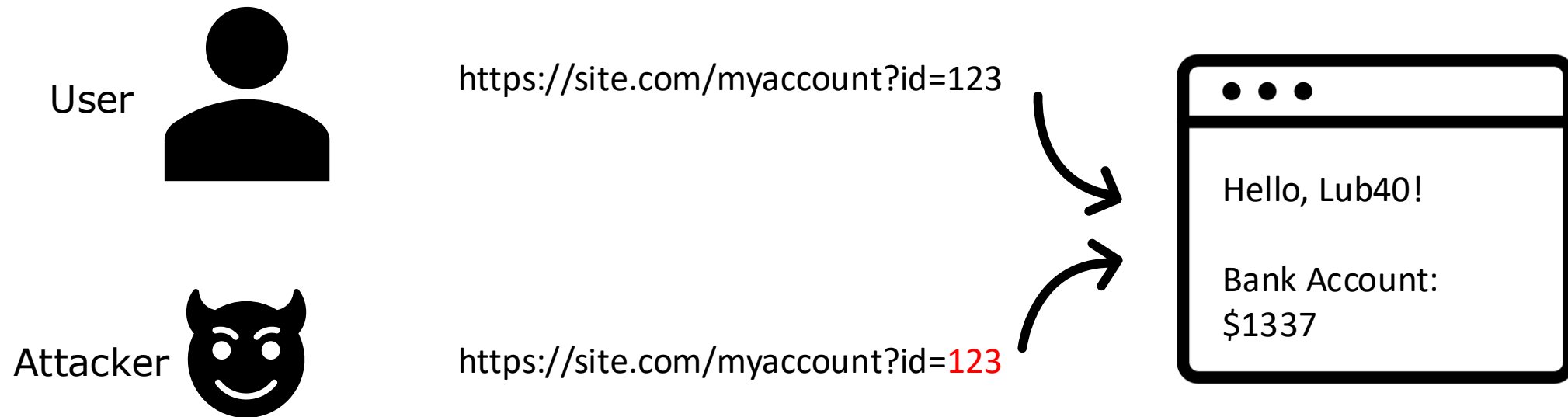
Horizontal Privilege Escalation

- Users access resources belonging to other users of the same privilege



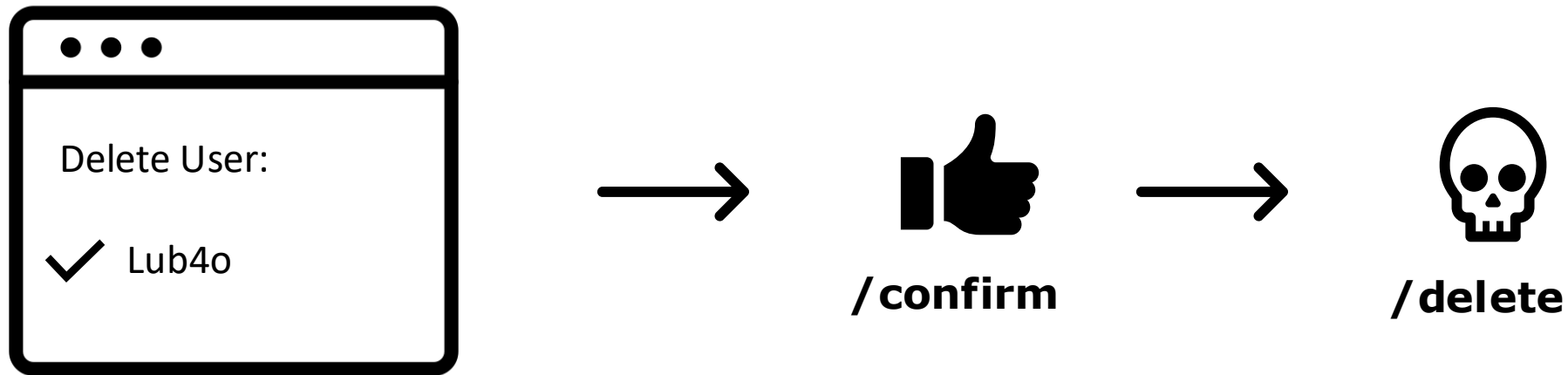
Horizontal Privilege Escalation

- Users access resources belonging to other users of the same privilege



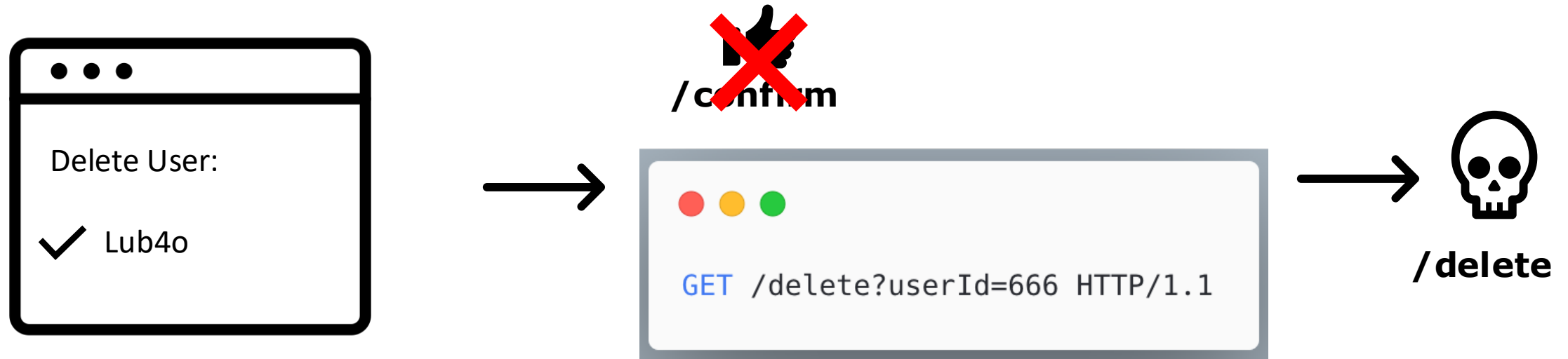
Multi-Step Process Vulnerabilities

- Access controls implemented on some steps but missing on others
- Attackers bypass the intended sequence
- Direct request to endpoint with missing controls



Multi-Step Process Vulnerabilities

- Access controls implemented on some steps but missing on others
- Attackers bypass the intended sequence
- Direct request to endpoint with missing controls



BAC Terminology

- **IDOR (Insecure Direct Object Reference):** Application exposes object IDs that can be manipulated to access unauthorized resources
- **BOLA (Broken Object Level Authorization):** Accessing objects belonging to other users by modifying identifiers
- **MFLAX (Missing Function Level Access Control):** Regular users accessing admin-only functionalit
- **Authentication Bypass:** Accessing protected resources while unauthenticated
- **Cross-Tenant Access:** Accessing another tenant's data in multi-tenant applications
- **Forceful browsing** - Directly accessing restricted URLs, bypassing intended navigation paths

Access Control (Testing Cheat Sheet)

1. Parameter Manipulation:

- Numeric ID substitution (**userId=1** → **userId=2**)
- Email address substitution in endpoints
- GUID/UUID substitution for non-sequential IDs
- Hash value substitution

2. Request Modification:

- Change HTTP methods (GET to POST, POST to /PUT/PATCH/DELETE...)
- Modify cookies or session tokens
- Alter hidden form fields

Access Control (Testing Cheat Sheet)

1. Path & Endpoint Techniques: 🔍

- Direct access to privileged endpoints
- Path traversal attacks
- Forced browsing via case changes (/admin/ vs /ADMIN/)
- Static file enumeration
- Direct function calls with parameters

2. Process Manipulation : ⚙️

- Bypass intended sequence in multi-step processes
- Skip authorization steps

IDOR PARAMS

Common parameters:

id	doc
user	key
account	email
number	group
order	profile
no	edit

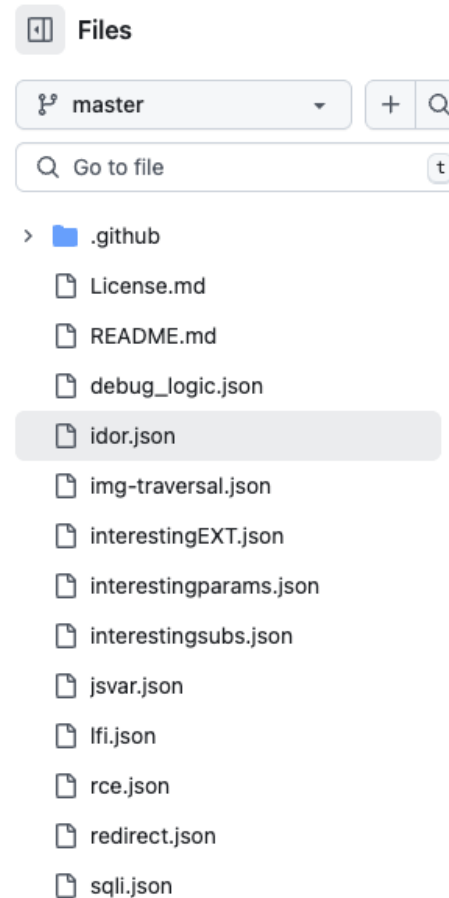
Functions:

Change mail/password

Upgrade/downgrade user role

Create/remove/update/delete context specific app data

Shipping, invoices and document viewing



Gf-Patterns / idor.json

1Indianl33t fixed idor pattern

Code Blame 26 lines (18 loc) · 191 Bytes

```
1  {
2      "flags": "-iE",
3      "patterns": [
4
5          "id=",
6          "user=",
7          "account=",
8          "number=",
9          "order=",
10         "no=",
11         "doc=",
12         "key=",
13         "email=",
14         "group=",
15         "profile=",
16         "edit=",
17         "report="
18     ]
19 }
20 }
```

<https://github.com/1Indianl33t/Gf-Patterns/blob/master/idor.json>

IDOR



```
POST /account/updatepasswd HTTP/1.1
Host: site.com
Connection: Close
Content-Length: 22
Cache-Control: max-age=0
Origin: https://site.com
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9, image/webp, image/apng
*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=3214536754363414df3142gf2341

userid=1337&action=updatepasswd
```

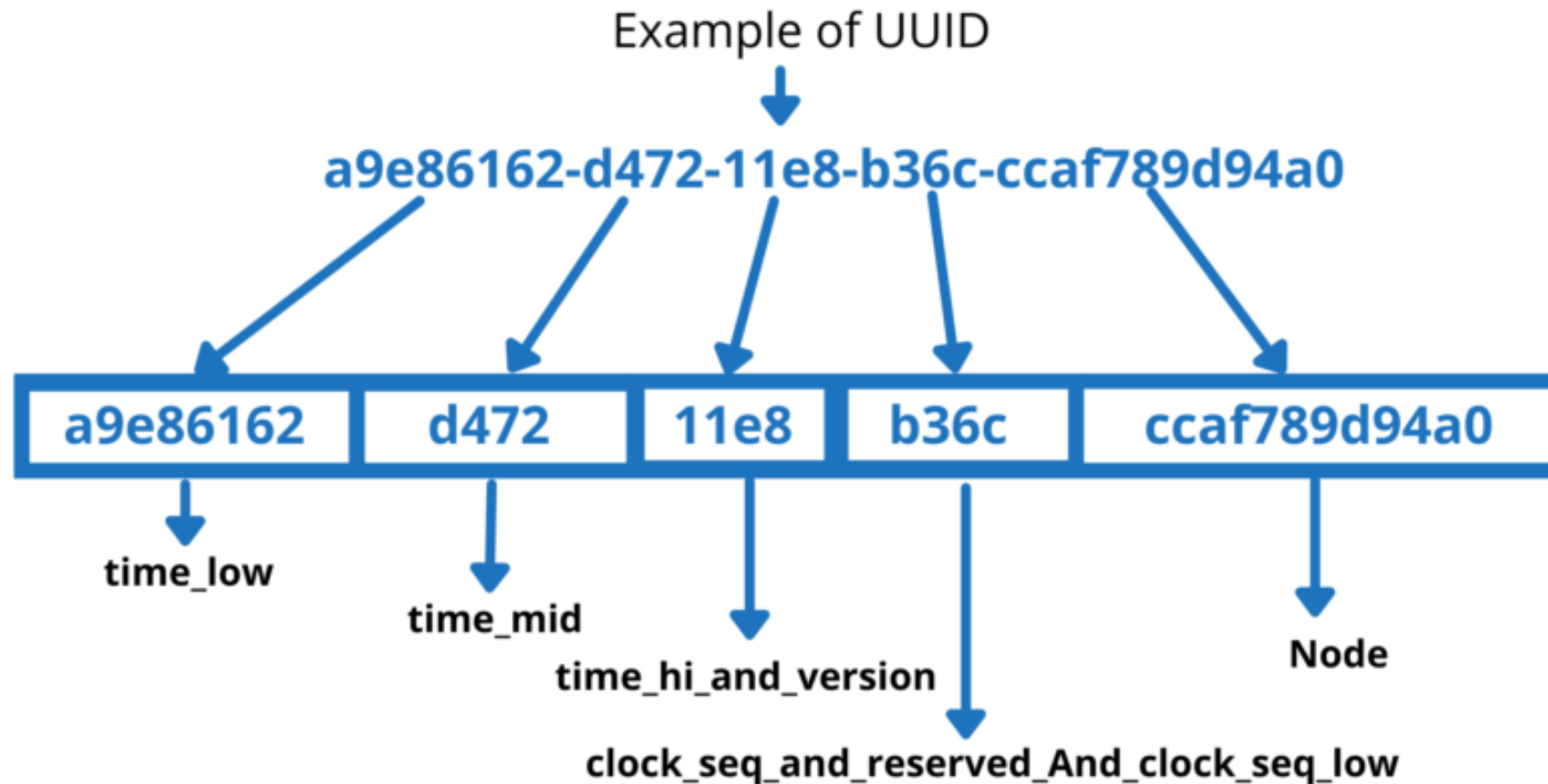
Hash based IDOR



```
POST /account/updatepasswd HTTP/1.1
Host: site.com
Connection: Close
Content-Length: 22
Cache-Control: max-age=0
Origin: https://site.com
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9, image/webp, image/apng
*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=3214536754363414df3142gf2341

userid=912134131a7611f2dfce0b92bf6b0eed&action=updatepasswd
```

UUID = No bug?



UUID = No bug?

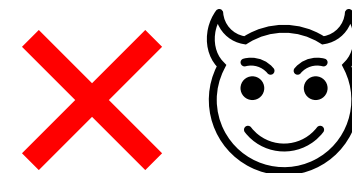
User accounts identified by UUIDs like:
8f1a2b3c-4d5e-6f7g-8h9i-0j1k2l3m4n5o

The developers believed this approach was secure because:

- UUIDs are not sequential or predictable
- The extremely large keyspace (2^{128} possibilities) makes brute-forcing impossible
- UUIDs don't reveal information about the user like sequential IDs might

- **Account data access:**
/account-details?uuid=8f1a2b3c-4d5e-6f7g-8h9i-0j1k2l3m4n5o

- **Password reset functions:**
/reset-password?user=8f1a2b3c-4d5e-6f7g-8h9i-0j1k2l3m4n5o



The Hidden Vulnerability

The application had an **"account linking"** feature that allowed users to link their accounts with family members. When viewing linked accounts, the application returned a JSON response (**UUID Leakage**)

```
{
  "primary_account": {
    "name": "Lub4o",
    "uuid": "8f1a2b3c-4d5e-6f7g-8h9i-0j1k2l3m4n5o",
    "linked_accounts": [
      {
        "name": "Mary Jane",
        "relationship": "spouse",
        "uuid": "9a8b7c6d-5e4f-3g2h-1i0j-9k8l7m6n5o4p"
      }
    ]
  }
}
```

The Hidden Vulnerability

The application had an "**account linking**" feature that allowed users to link their accounts with family members. When viewing linked accounts, the application returned a JSON response



```
POST /change-contact-info?uuid=9a8b7c6d-5e4f-3g2h-1i@j-9k8l7m6n5o4p HTTP/1.1
```

```
Host: finance.site.com
```

```
Cookie: session=spz_session_cookie
```

```
{"email": "malicious@attacker.com"}
```

Testing Approaches – Black Box

1. Map the application thoroughly
2. Identify potential parameters used for access control
3. Test with multiple accounts of different privilege levels
4. Manipulate parameters systematically
5. Use automation tools like Burp Authorize extension

Testing Approaches – White Box Testing

1. Review code for access control implementation
2. Look for these red flags:
 - Violations of principle of least privilege
 - Missing/weak access control checks
 - Missing controls on certain API methods
 - Trust on client-side input for access decisions
3. Validate findings on running application

Testing Matrix

	Update Password	Update Email	Change Account Data	Upgrade Account to Admin	View Logs
Admin	yes	yes	yes	yes	yes
User	yes	yes	no	no	no
Unauthenticated	no	no	no	no	no

Text = Should they be able to do it?

Color = could they do it? (red is bad)

Burp Suite Add-ons: Autorize, AutoRepeater, AuthMatrix, Authz

ID	Met...	URL	Orig. L...	Modif...	Unaut...	Authz. ...	Unaut...
24	OPTI...	https://signaler-pa.clients6.google.com:443/punctua...	0	0	0	Bypass...	Bypass...
25	GET	https://example.com:443/	1256	1256	1256	Bypass...	Bypass...
26	GET	https://example.com:443/	1256	1256	1256	Bypass...	Bypass...
27	GET	https://example.com:443/	1256	1256	1256	Enforc...	Bypass...
28	GET	https://content-autofill.googleapis.com:443/v1/page...	32	41	32	Enforc...	Bypass...
29	GET	https://github.com:443/Quitten/Autorize/security/ov...	0	0	0	Bypass...	Bypass...
30	GET	https://github.com:443/Quitten/Autorize/tree/master	378763	380613	385378	Enforc...	Is info...
31	POST	https://waa-pa.clients6.google.com:443/\$rpc/googl...	2	43	2	Enforc...	Bypass...
32	GET	https://github.com:443/Quitten/Autorize/tree/master	385471	380801	385234	Enforc...	Is info...
33	GET	https://avatars.githubusercontent.com:443/u/269682...	1563	1563	1563	Bypass...	Bypass...
34	GET	https://avatars.githubusercontent.com:443/u/165176...	1807	1807	1807	Bypass...	Bypass...
35	GET	https://avatars.githubusercontent.com:443/u/138052...	1251	1251	1251	Bypass...	Bypass...
36	GET	https://avatars.githubusercontent.com:443/u/213717...	2165	2165	2165	Bypass...	Bypass...
37	GET	https://avatars.githubusercontent.com:443/u/585628...	1558	1558	1558	Bypass...	Bypass...
38	GET	https://avatars.githubusercontent.com:443/u/527377...	1556	1556	1556	Bypass...	Bypass...
39	GET	https://avatars.githubusercontent.com:443/u/269640...	1868	1868	1868	Bypass...	Bypass...
40	GET	https://avatars.githubusercontent.com:443/u/828821...	1571	1571	1571	Bypass...	Bypass...
41	GET	https://avatars.githubusercontent.com:443/u/699004...	1532	1532	1532	Bypass...	Bypass...
42	GET	https://avatars.githubusercontent.com:443/u/828821...	1571	1571	1571	Bypass...	Bypass...
43	GET	https://avatars.githubusercontent.com:443/u/571037...	1534	1534	1534	Bypass...	Bypass...
44	GET	https://avatars.githubusercontent.com:443/u/828821...	1571	1571	1571	Bypass...	Bypass...
45	GET	https://avatars.githubusercontent.com:443/u/292788...	6261	6261	6261	Bypass...	Bypass...
46	GET	https://avatars.githubusercontent.com:443/u/495600...	6912	6912	6912	Bypass...	Bypass...
47	GET	https://avatars.githubusercontent.com:443/u/931075...	1526	1526	1526	Bypass...	Bypass...
48	GET	https://avatars.githubusercontent.com:443/u/435262...	20745	20745	20745	Bypass...	Bypass...
49	GET	https://content-autofill.googleapis.com:443/v1/page...	56	41	56	Enforc...	Bypass...
50	GET	https://content-autofill.googleapis.com:443/v1/page...	16	41	16	Enforc...	Bypass...
51	GET	https://github.com:443/Quitten/Autorize/branch-count	89	89	89	Bypass...	Bypass...
52	GET	https://github.com:443/notifications/indicator	15	15	15	Bypass...	Bypass...
53	GET	https://github.com:443/Quitten/Autorize/branch-count	89	89	89	Bypass...	Bypass...
54	GET	https://github.com:443/Quitten/Autorize/latest-com...	839	839	839	Bypass...	Bypass...
55	GET	https://github.com:443/Quitten/Autorize/tag-count	86	86	86	Bypass...	Bypass...
56	GET	https://github.com:443/notifications/30739183/watc...	3741	3741	3741	Bypass...	Bypass...
57	GET	https://github.com:443/Quitten/Autorize/security/ov...	0	0	0	Bypass...	Bypass...
58	GET	https://github.com:443/Quitten/Autorize/tree-commi...	4331	4331	4331	Bypass...	Bypass...
59	GET	https://github.com:443/Quitten/Autorize/tag-count	86	86	86	Bypass...	Bypass...
60	GET	https://github.com:443/Quitten/Autorize/used_by_list	0	0	0	Bypass...	Bypass...
61	GET	https://github.com:443/Quitten/Autorize/spoofed_co...	2	2	2	Bypass...	Bypass...
62	GET	https://github.com:443/Quitten/Autorize/recently-to...	417	417	417	Is info...	Is info...

Request/Response Viewers Configuration

Autorize is on

☒ Ignore 304/204 status code responses
☐ Prevent 304 Not Modified status code
☐ Intercept requests from Repeater
☒ Check unauthenticated
☐ Replace query params
☐ Auto Scroll

Clear List

Temporary headers

Authorization: SAPISIDHASH
1721337247bc3cac094f8878d9b6b5

From last request:
Fetch Cookies header Fetch Authorization header

Enforcement Detector Detector Unauthenticated Interception Filters Match/Replace Table Filter Save/Restore

Type: Scope items only: (Content is not required)

Content:

Filter List: URL Not Contains (regex): (\\js\\,css\\,png\\,jpg\\,svg\\,jpeg\\,gif\\,woff\\,map\\,bmp\\,ico)(?![a-z+)]*[\\S])\$
Ignore spider requests:

Add filter Remove filter Modify filter

“Easy IDOR hunting with Autorize?” by InsiderPHD <https://www.youtube.com/watch?v=2WzqH6N-Gbc>

Burp Suite Intruder – Sniper Attack (Numbers)

DashboardTargetProxyIntruderRepeaterCollaboratorSequencerDecoderComparerLoggerOrganizerSettings

ExtensionsLearnAuthorizeInQL

1 x2 x+

1

Sniper attack

5

Start attack

Target

http://testphp.vulnweb.com

☒ Update Host header to match target

Positions

Add \$

Clear \$

Auto \$

1

GET /listproducts.php?cat=\$1\$ HTTP/1.1

2

Host: testphp.vulnweb.com

3

Accept-Language: en-GB,en;q=0.9

4

Upgrade-Insecure-Requests: 1

5

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)

6

Accept:

7

Referer: http://testphp.vulnweb.com/categories.php

8

Accept-Encoding: gzip, deflate, br

9

Connection: keep-alive

10

11

Payloads

3

3

Payload position:

All payload positions

3

3

Payload type:

Numbers

Payload count:

11

Request count:

11

Payload configuration

^

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type:

4

☒ Sequential ☐ Random

From:

0

To:

10

Step:

1

How many:

Number format

Burp Suite Intruder – Sniper Attack (Brute Force)

Burp Suite Community Edition v2025.1.5 - Temporary Project

Dashboard Target Proxy **Intruder** Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Settings

Extensions Learn Authorize InQL

1 x 2 x +

2 ? Sniper attack 7 Start attack

Target http://testphp.vulnweb.com ☒ Update Host header to match target

Positions Add \$ Clear \$ Auto \$

```
1 GET /listproducts.php?cat=$123abc$ HTTP/1.1
2 Host: testphp.vulnweb.com
3 Accept-Language: en-GB,en;q=0.9
4 Upgrade-Insecure-Requests: 1
5 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
6 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
7 Referer: http://testphp.vulnweb.com/categories.php
8 Accept-Encoding: gzip, deflate, br
9 Connection: keep-alive
10
11
```

Payloads

Payload position: All payload positions

Payload type: 4 Brute forcer

Payload count: unknown

Request count: unknown

Payload configuration

This payload type generates payloads of specified lengths that contain all permutations of a specified character set.

5 Character set: abcdefghijklmnopqrstuvwxyz0123456789

Min length: 6

Max length: 6

6

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Resource pool

Settings

Preventing Broken Access Control

1. Design Defensively

- Implement centralized access control 
- Deny access by default 
- Principle of least privilege 

2. Validate Server-Side

- Never trust client inputs 
- Verify permissions for every request 
- Validate resource ownership 

3. Build Robust Controls

- Use mature authorization frameworks 
- Implement role/attribute-based access control 
- Log and monitor access attempts 

Thank You!

Questions? I'm here to help! 🙋🙋

Resources:

Web Security Academy - Access Control

- <https://portswigger.net/web-security/access-control>

Web Application Hacker's Handbook

- Chapter 8 - Attacking Access Controls

OWASP Top 10 - A01 Broken Access Control

- https://owasp.org/Top10/A01_2021-Broken_Access_Control/

OWASP Web Security Testing Guide - Authorization Testing

- https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/05-Authorization_Testing/README

OWASP Application Security Verification Standard - V4 Access Control

- https://owasp.org/www-pdf-archive/OWASP_Application_Security_Verification_Standard_4.0-en.pdf