Broken Authentication and Session Management. What is it?

- This vulnerability category is currently ranked #7 on the OWASP 2021 Top 10 list
- Vulnerability that allows an attacker to capture credentials or bypass authentication methods used to protect against unauthorised access.
- The most common authentication scheme is the use of a username and password.
- Approximately 23 % of all Web Apps are vulnerable[1].

[1] IBM X-Force Threat Intelligence 2Q Quarterly

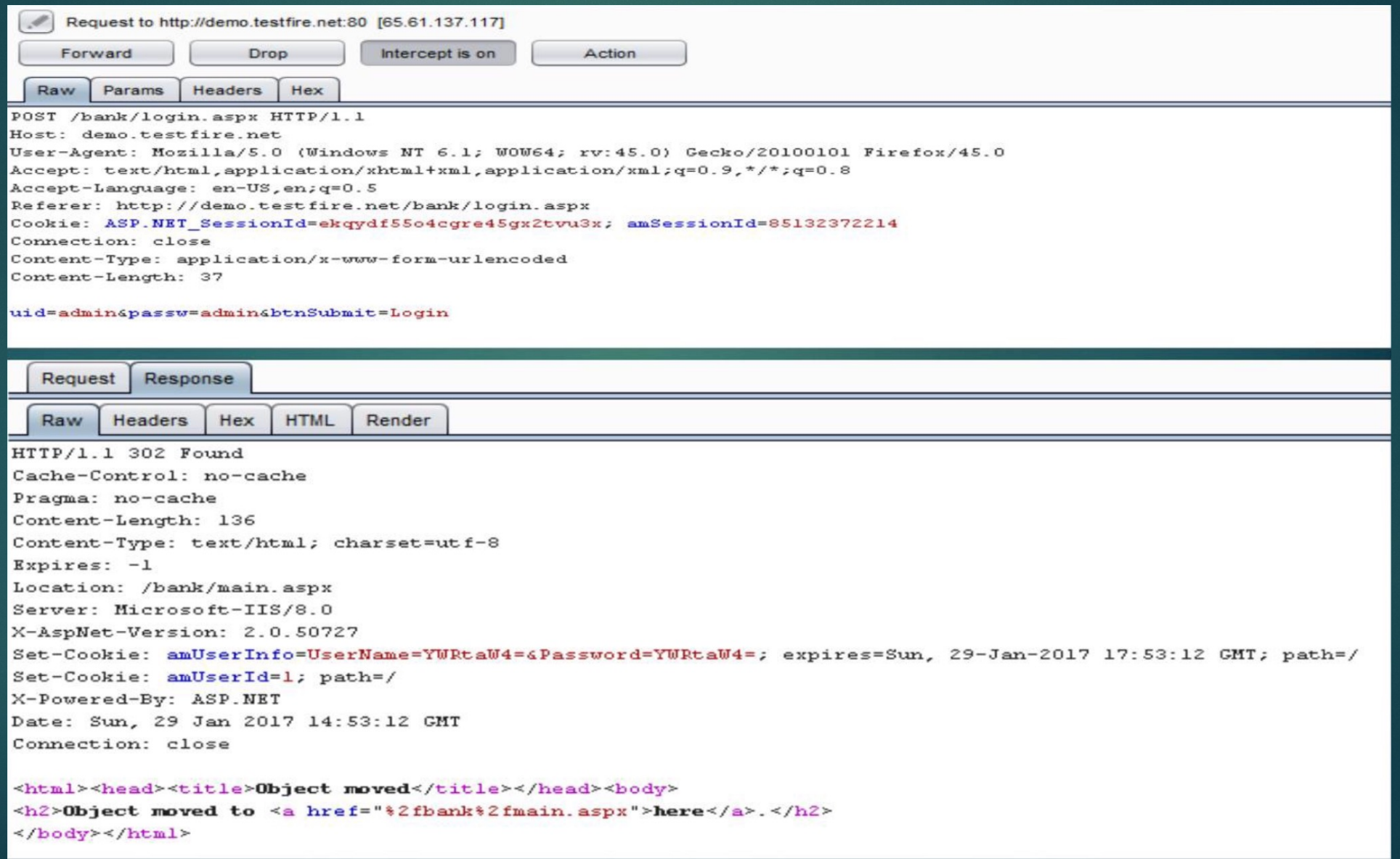
How does the application fail to protect username, password and session ID?

- Unencrypted connections to application.
- Predictable login credentials.
- Session values are not timed out or do not get invalidated after logout.
- Authentication details such as username and password are not protected when stored.
- Session IDs are used in URLs

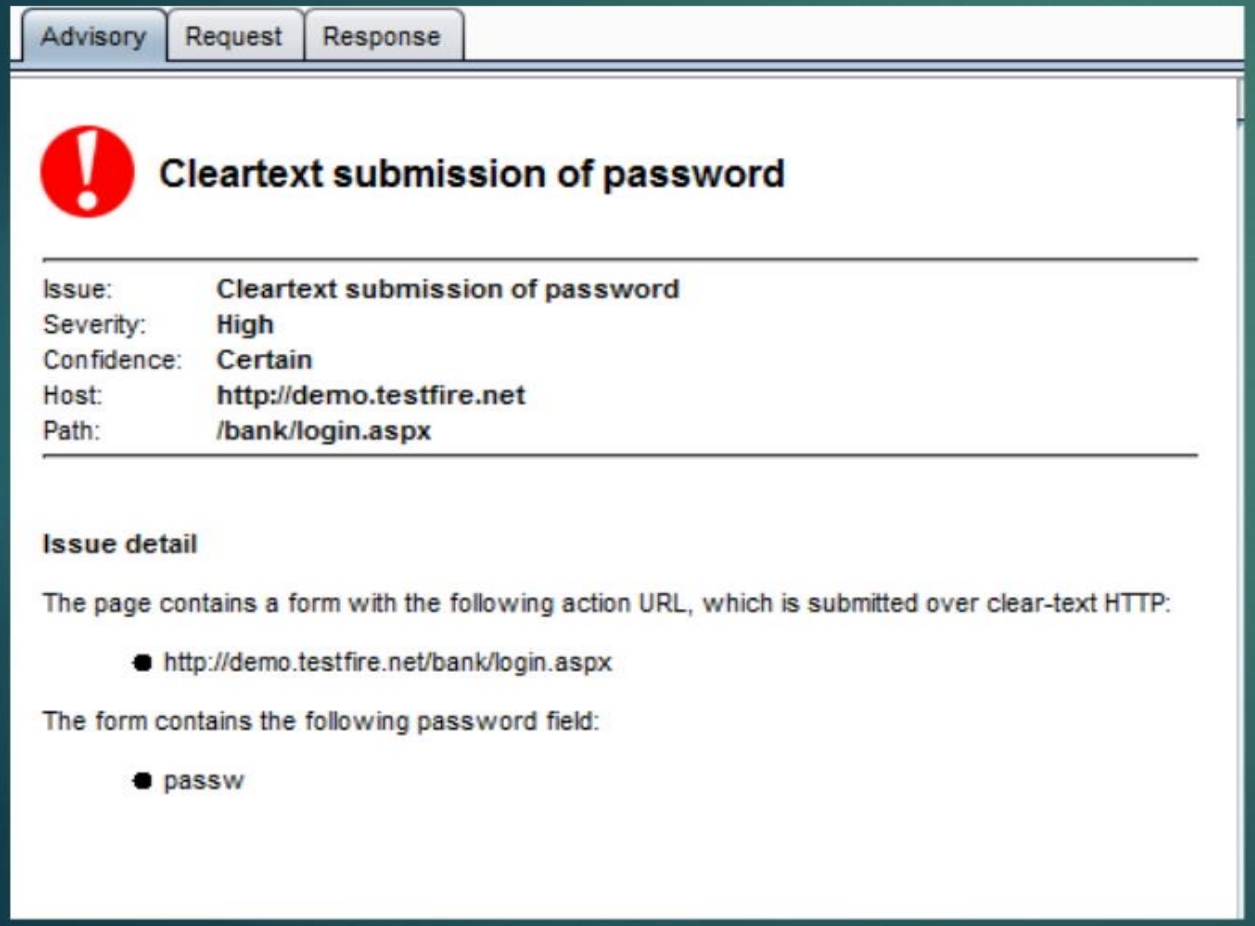
Unencrypted Connections

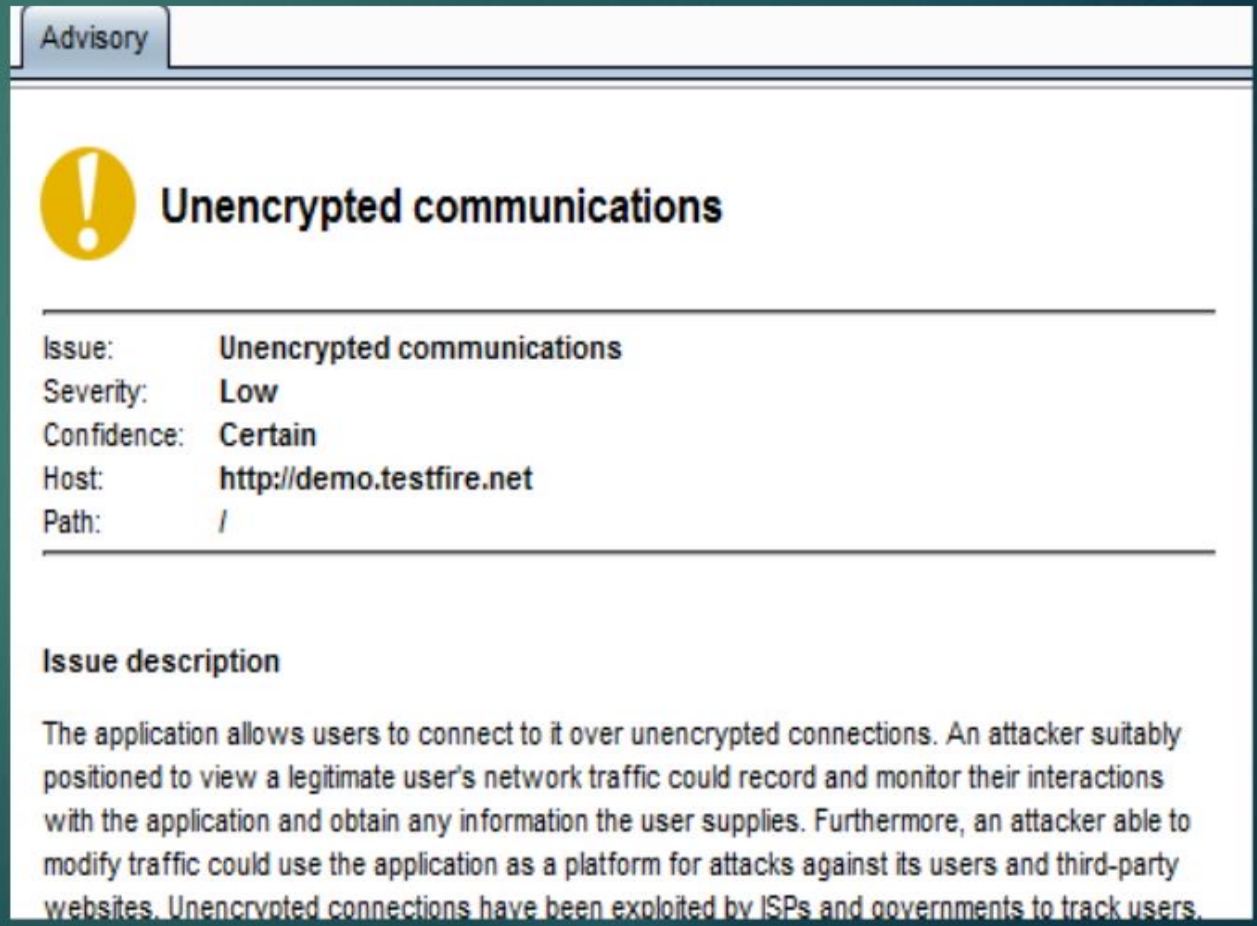
- Application allows users to connect to it over unencrypted connections (HTTP)
- MITM can intercept all info that you are sending/receiving between yourself and application without your knowledge.
- Fails to protect username password and session ID values.
- Enable transport-level encryption (SSL/TLS)/HTTPS on server.
  - This will encrypt communications between client and server.
- Set secure flags on cookies to prevent from being observed by unauthorized parties due to the transmission of the cookie in clear text.

Example Request/Response



Example Issues





Predictable Login Credentials

- This vulnerability fails to protect credentials
- Username and Passwords that are easily guessable. Default passwords on admin panels.
- Allow an attacker to obtain unauthorized access to an application.
- Users must be forced to use strong password policy.
- Users must be forced to changed default passwords on systems and applications.

Predictable Passwords[2]

Table 1: Top 20 Passwords in HIBP_v5

Password	% of Total Accounts
123456	0.596%
123456789	0.197%
qwerty	0.099%
password	0.094%
111111	0.079%
12345678	0.074%
abc123	0.072%
1234567	0.064%
password1	0.061%
12345	0.060%
1234567890	0.057%
123123	0.056%
000000	0.050%
iloveyou	0.041%
1234	0.033%
1q2w3e4r5t	0.030%
qwertyuiop	0.028%
123	0.026%
monkey	0.025%
dragon	0.025%

[2] A. Kanta, S. Coray, I. Coisel and M, Scanlon, *How Viable is Password Cracking in Digital Forensic Investigation? Analyzing the Guessability of over 3.9 Billion Real-World Accounts*, Forensic Science International: Digital Investigation, ISSN 2666-2825, July 2021. https://doi.org/10.1016/j.fsidi.2021.301186

Predictable Passwords[2]

Table 1: Top 20 Passwords in HIBP_v5

Password	% of Total Accounts
123456	0.596%
123456789	0.197%
qwerty	0.099%
password	0.094%
111111	0.079%
12345678	0.074%
abc123	0.072%
1234567	0.064%
password1	0.061%
12345	0.060%
1234567890	0.057%
123123	0.056%
000000	0.050%
iloveyou	0.041%
1234	0.033%
1q2w3e4r5t	0.030%
qwertyuiop	0.028%
123	0.026%
monkey	0.025%
dragon	0.025%

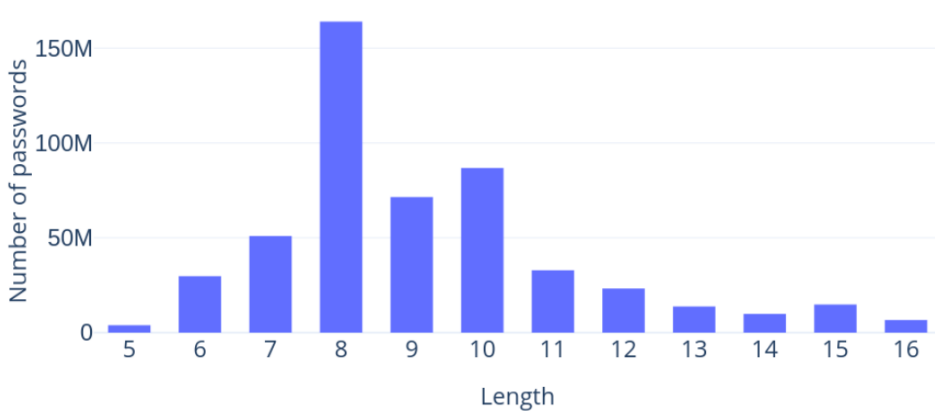


Figure 2: Most Common Password Lengths

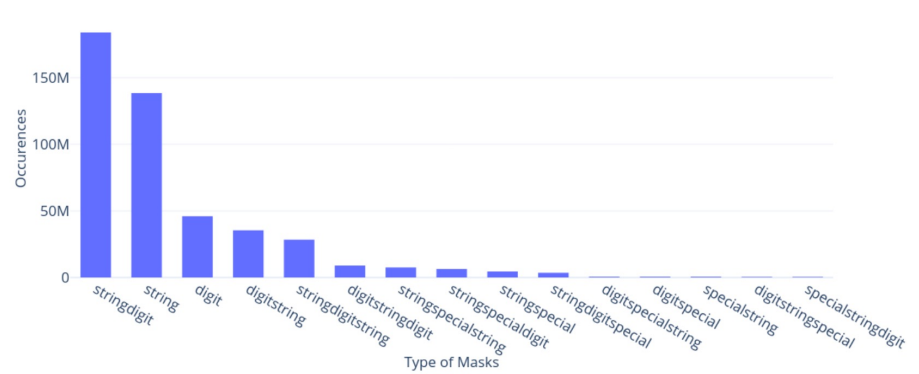


Figure 4: Simple Masks

[2] A. Kanta, S. Coray, I. Coisel and M, Scanlon, *How Viable is Password Cracking in Digital Forensic Investigation? Analyzing the Guessability of over 3.9 Billion Real-World Accounts*, Forensic Science International: Digital Investigation, ISSN 2666-2825, July 2021. https://doi.org/10.1016/j.fsidi.2021.301186

Predictable Passwords[2]

Table 4: Top 50 Letter, Number and Special Character Segments

Letter	Count	Number	Count	Special	Count
a	2.335%	1	8.240%	.	0.871%
i	1.168%	123456	5.137%	_	0.666%
qwerty	0.597%	123	2.574%	!	0.469%
password	0.510%	2	2.398%	@	0.334%
love	0.484%	123456789	2.083%	-	0.327%
my	0.356%	3	1.788%	:	0.140%
abc	0.274%	4	1.578%	#	0.105%
to	0.259%	5	1.111%	*	0.090%
an	0.259%	12	1.079%	$	0.071%
qwe	0.248%	7	1.029%		0.065%
in	0.238%	0	0.870%	&	0.045%
the	0.228%	8	0.812%	+	0.042%
qaz	0.223%	6	0.810%	?	0.037%
iloveyou	0.221%	12345	0.764%	,	0.035%
ws	0.217%	9	0.761%	/	0.031%
as	0.209%	1234	0.664%	!!	0.025%
no	0.198%	11	0.599%	.:	0.023%
ilove	0.196%	13	0.518%	&#	0.022%
by	0.191%	12345678	0.474%	=	0.021%
man	0.190%	01	0.430%	;	0.018%
baby	0.178%	10	0.425%	..	0.017%

[2] A. Kanta, S. Coray, I. Coisel and M, Scanlon, *How Viable is Password Cracking in Digital Forensic Investigation? Analyzing the Guessability of over 3.9 Billion Real-World Accounts*, Forensic Science International: Digital Investigation, ISSN 2666-2825, July 2021. https://doi.org/10.1016/j.fsidi.2021.301186

Session ID: No Timeout or Invalidation on Logout

- Fails to protect the session ID
- Issue arises when application does not discard session ID after n amount of time or does not invalidate session ID after logging out.
- This can be mitigated by setting a timeout on the session ID.
- Application may also invalidate the session ID by the following when logging out
  - `Set Cookie: sessionID= ;`

Example Request/Response

Request | Response

Raw | Params | Headers | Hex

```
GET /bank/logout.aspx HTTP/1.1
Host: demo.testfire.net
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: http://demo.testfire.net/bank/main.aspx
Cookie: ASP.NET_SessionId=ekqydf55o4cgre45gx2tvu3x; amSessionId=85132372214;
amUserInfo=UserName=YWRtaW4=&Password=YWRtaW4=; amUserId=1
Connection: close
```

Request | Response

Raw | Headers | Hex | HTML | Render

```
HTTP/1.1 302 Found
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 132
Content-Type: text/html; charset=utf-8
Expires: -1
Location: /default.aspx
Server: Microsoft-IIS/8.0
X-AspNet-Version: 2.0.50727
Set-Cookie: amUserId=; expires=Sat, 28-Jan-2017 15:18:15 GMT; path=/
Set-Cookie: amCreditOffer=; expires=Sat, 28-Jan-2017 15:18:15 GMT; path=/
X-Powered-By: ASP.NET
Date: Sun, 29 Jan 2017 15:18:15 GMT
Connection: close
```

User authentication credentials are not protected adequately when stored.

- Fails to protect user credentials
- Issue arises when little or no protection is given and values are stored using weak encryption or as plain text.
- If an attacker can gain access to stored data then there is full compromise.
- Stored user credentials should be salted and hashed in addition to being encrypted.

Example in Java

- Snippet reads a password from a properties file and uses the password to connect to a database.

Example Language: **Java**

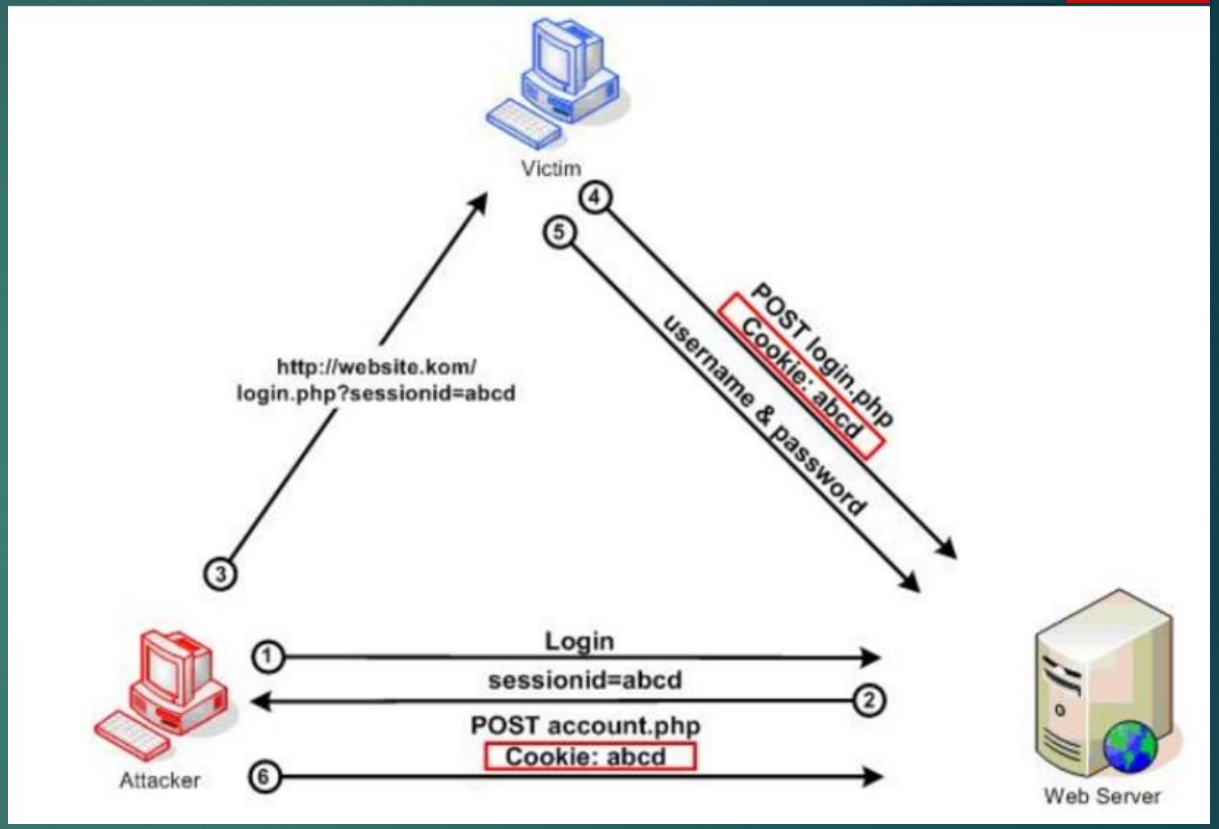
```
...
Properties prop = new Properties();
prop.load(new FileInputStream("config.properties"));
String password = Base64.decode(prop.getProperty("password"));
DriverManager.getConnection(url, usr, password);
...
```

- This code will run successfully, but anyone with access to `config.properties` can read the value of password and easily determine that the value has been base 64 encoded. If a devious employee has access to this information, they can use it to break into the system[3]

[3] https://cwe.mitre.org/data/definitions/261.html

Example Session Fixation

1. The attacker logs into web server.
2. Server issues a session ID
3. Attacker has to send a link with the established session ID to the victim, victim clicks on the link sent from the attacker
4. Web Server sees that session was already established and a new one need not to be created.
5. Victim provides credentials to the Web Server.
6. Knowing the user's account and the session ID, the attacker gains access



To defend against session fixation, ensure your Web application developers code their applications so they assign a different session cookie immediately after a user authenticates to the application, and also verify they do not include the cookie value in the URL

Session Management

- Session ID Properties
- Session management Implementation
- TLS/HTTPS
- Cookies

Session ID properties

- Session ID is a “name=value” pair
- Session ID name fingerprinting
  - Should not be too descriptive or offer unnecessary details about purpose and meaning of the ID
  - PHPSESSID (PHP), JSESSIONID (J2EE) etc.
  - These can disclose technologies and programming languages.
- Session ID length and entropy
  - Long enough to prevent brute force attacks. Must be at least 128 bits (16 bytes)
  - Must be unpredictable (random enough) to prevent guessing attacks. Use a good pseudorandom number generator
- Session ID Content
  - Must be meaningless to prevent info disclosure attacks, where an attacker is able to decode the contents and extract user details.

Session Management Implementation

- Defines the exchange mechanism to be used between the user and application.
- Deals with sharing and continuous exchange of session IDs
- Multiple mechanisms available in HTTP to maintain session state in web apps.
  - Cookies (standard http header)
  - URL parameters
  - URL arguments on GET
  - Body arguments on POST (hidden form fields)

Transport Layer Security (TLS)

- Pivotal in protecting session ID exchange securely
- Prevents active eavesdropping and passive disclosure in network traffic.
- Should be used for an entire web session, not only for authentication process. (credential exchange)
- Secure attribute will ensure session ID is only exchanged through an encrypted request.
- Protects against some session fixation attacks

Cookies

- Secure attribute
  - Instructs web browser to only send cookie via HTTPS connection
- HttpOnly attribute
  - Instructs web browser not to allow client side scripts to access the cookies via DOM
- Domain and Path attributes
  - Instructs web browser to only send the cookies to the specified domain and all subdomains
- Expire and Max-Age Attributes (Persistent)
  - Stored on disk by browser until the expiration time
  - Persistent cookies vs non-persistent (gone when session is closed)