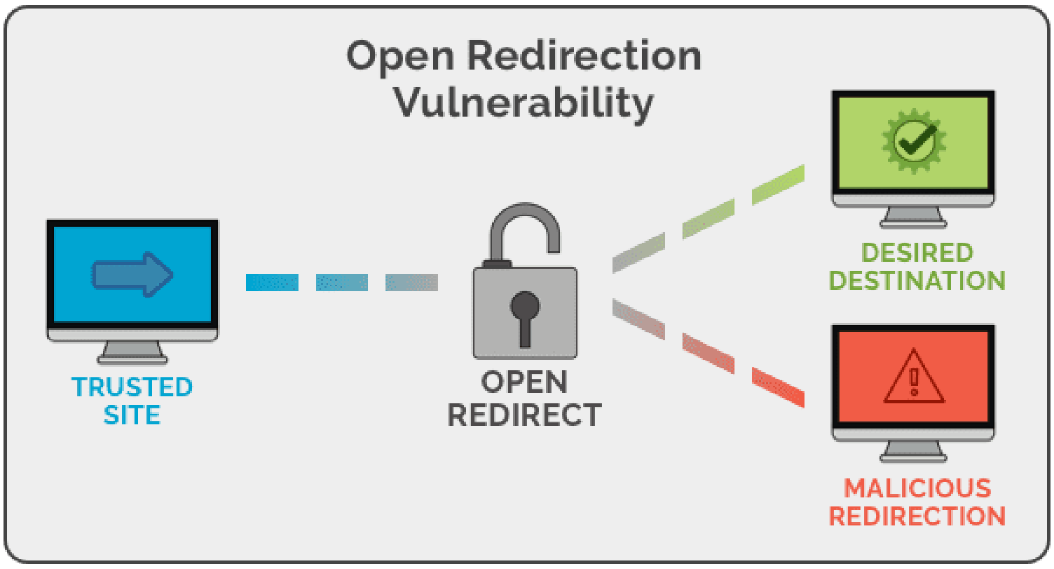
Open Redirection

#Session_Task_12 by Sukhveer singh

1. Open redirection –



Open redirection is a security vulnerability that occurs when a web application takes a user-supplied input and uses it to redirect the user to another page without proper validation. This vulnerability can be exploited by attackers to redirect users to malicious websites, phishing pages, or other harmful destinations.

For example, suppose a website has a login page with a parameter called "redirect" in the URL that determines where the user should be redirected after successful login. If the application does not properly validate this parameter, an attacker could craft a URL with a malicious redirect to a phishing site. When unsuspecting users click on this link and log in, they would be redirected to the attacker's phishing page, potentially leading to their credentials being stolen.

2. Parameters to look for –

```
/url?q=
/url?sa=t&url=
/external-link.jspa?url=
/proxy.php?link=
/redir.asp?url=
/page/pmv?url=
/exit.aspx?url=
/bitrix/redirect.php?event1=&event2=&event3=&goto=
/register/crowdsupport?gotoURL=
/dk?st.cmd=outLinkWarning&st.rfn=
```

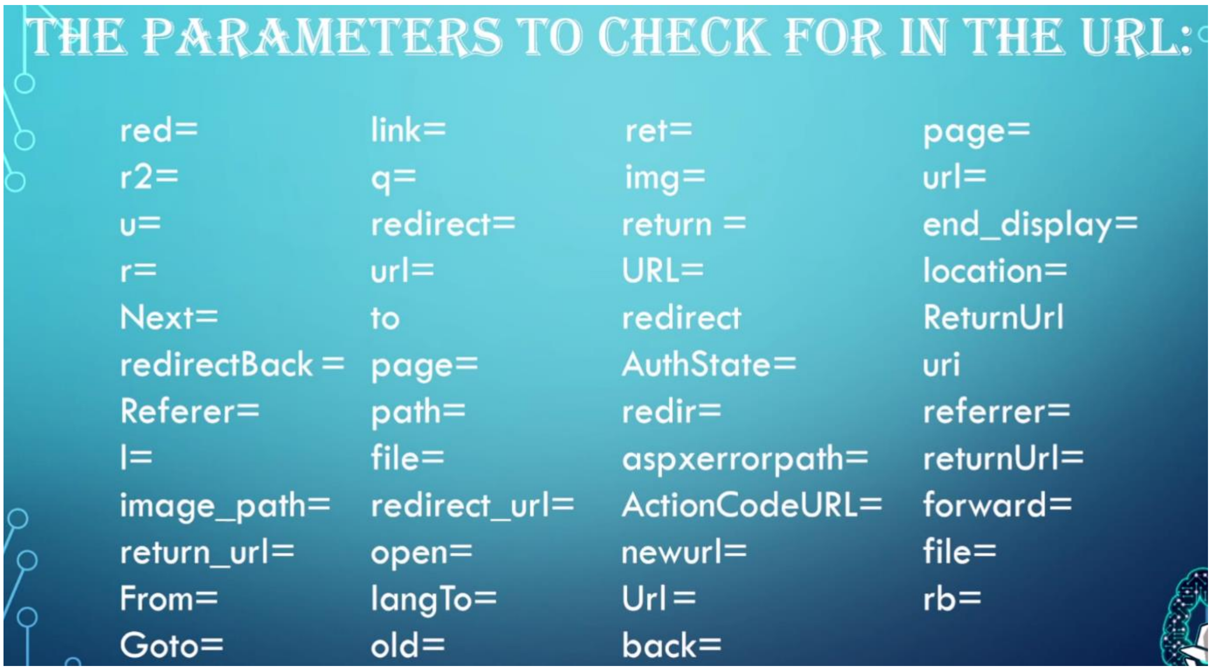
- **Redirect URLs:** Identify any parameters or inputs that determine where the user will be redirected. These could be query parameters in the URL, form fields, or cookies.

- **User-controlled Input**: Determine if the redirection destination is influenced by user-supplied data. This could include parameters like "redirect," "url," or any other input fields where users can specify a URL.

- **Validation:** Check if the application properly validates and sanitizes the input used for redirection. Look for server-side validation logic that ensures the redirect URL is safe and belongs to the same domain or a whitelist of trusted domains.

- **Redirection Mechanisms:** Examine how the application handles redirections. Look for any code or configurations that perform redirects, such as HTTP 3xx status codes or JavaScript redirect functions.

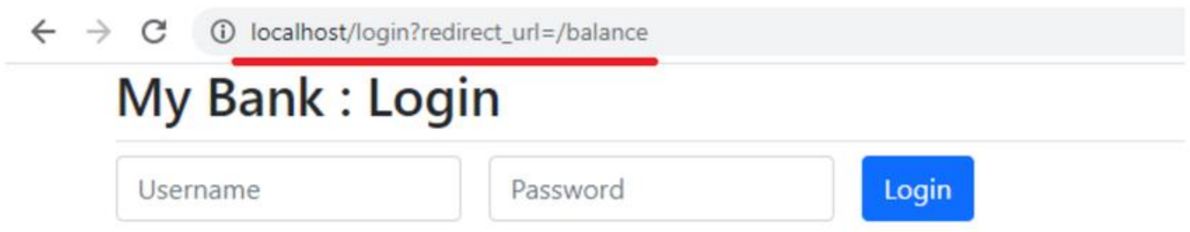
- **Access Controls:** Assess whether the application enforces proper access controls to prevent unauthorized redirections. Ensure that users are only allowed to redirect to legitimate and authorized destinations.



THE PARAMETERS TO CHECK FOR IN THE URL:

red=	link=	ret=	page=
r2=	q=	img=	url=
u=	redirect=	return =	end_display=
r=	url=	URL=	location=
Next=	to	redirect	ReturnUrl
redirectBack =	page=	AuthState=	uri
Referer=	path=	redir=	referrer=
l=	file=	aspxerrorpath=	returnUrl=
image_path=	redirect_url=	ActionCodeURL=	forward=
return_url=	open=	newurl=	file=
From=	langTo=	Url =	rb=
Goto=	old=	back=	

3. Impact of open redirection vulnerability –


localhost/login?redirect_url=/balance

My Bank : Login

Phishing Attacks: Attackers can exploit open redirection vulnerabilities to redirect users to malicious websites designed to steal sensitive information such as login credentials, financial data, or personal information through phishing attacks.

Malware Distribution: Malicious actors may use open redirection vulnerabilities to redirect users to websites hosting malware. This can lead to the unintended installation of malware on users' devices, compromising their security and privacy.

Identity Theft: By tricking users into visiting phishing pages through open redirections, attackers can potentially steal their identities, leading to various forms of identity theft, including financial fraud and account takeovers.

Brand Damage: If a website is compromised due to an open redirection vulnerability, it can damage the reputation and trustworthiness of the organization or brand associated with the website. Users may lose confidence in the security of the platform, leading to a loss of customers or clients.

SEO Manipulation: Attackers can abuse open redirection vulnerabilities to manipulate search engine optimization (SEO) by redirecting users to websites that promote spam, illicit products, or illegal activities. This can negatively impact the search engine ranking of legitimate websites.

Data Breaches: Open redirection vulnerabilities can sometimes be part of a larger security flaw that could lead to data breaches. For example, if an attacker gains unauthorized access to sensitive information through a phishing page, it could result in a data breach with severe consequences for individuals and organizations.

4. Mitigations –

Input Validation: Ensure that all user-supplied input used for redirection is properly validated and sanitized. Validate the input to ensure it conforms to expected formats and does not contain malicious or unexpected characters.

Whitelisting: Maintain a whitelist of trusted redirection destinations or domains. Only allow redirects to URLs that are explicitly whitelisted, reducing the risk of users being redirected to malicious websites.

Encode Redirect URLs: Encode the redirect URLs to prevent attackers from injecting malicious characters or payloads. Encoding helps ensure that the URLs are interpreted correctly by the application and reduces the likelihood of injection attacks.

Use Safe Redirection Methods: Instead of relying solely on user-supplied input for redirection, use safe redirection methods provided by the web application framework or libraries. Avoid using user-controlled input to construct redirection URLs whenever possible.

Implement Access Controls: Enforce proper access controls to restrict access to sensitive functionality and resources. Only allow authorized users to perform redirections, and ensure that users can only redirect to legitimate and authorized destinations.

Security Headers: Utilize security headers such as Content Security Policy (CSP) and X-Frame-Options to mitigate the risk of malicious redirections and frame-based attacks. These headers can help prevent unauthorized redirections and protect against clickjacking vulnerabilities.

Regular Security Audits: Conduct regular security audits and penetration testing to identify and remediate open redirection vulnerabilities. Test the application thoroughly for potential security flaws and ensure that proper security measures are in place to protect against redirection attacks.

5. References –

https://brightsec.com/blog/open-redirect-vulnerabilities/
https://portswigger.net/kb/issues/00500100_open-redirection-reflected
https://www.acunetix.com/blog/web-security-zone/what-are-open-redirects/
https://www.wallarm.com/what/open-redirect-vulnerability