Unvalidated redirects and forwards can occur on websites that redirect users to destinations obtained from untrusted external inputs.

A typical example of a vulnerability is a sign-in page that redirects users to a return URL specified by a query string parameter. Without sufficient protection, the following type of attack could occur:

1. An attacker creates a forged version of the site's sign-in page.
2. The attacker sends out links to the legitimate sign-in page with the forged page in the return URL parameter.
   For example: *http://domain.com/SignIn?returnUrl=http://forgeddomain.com/SignIn*
3. A user clicks the link, signs in, and is redirected to the forged page.
4. The page informs the user that the authentication failed and requests another attempt to enter the sign-in credentials.
5. The user submits their authentication credentials on the forged page.
6. The page redirects the user back to the home page of the original site, where the user is already signed in.

An attacker could obtain the authentication credentials of users without them even noticing that an attack has occurred.

Handling of redirects in Kentico

All redirects performed by default in Kentico are secured against unvalidated redirection attacks.

Certain Kentico components do perform redirects to URLs obtained from external inputs, but the system only allows local URLs in such cases. For redirects that lead away from the website, the destination is automatically changed to the application root. If a redirect URL is set in the Kentico administration interface (a trusted source), no validation occurs and external URLs are allowed.

For example, the built-in **Logon form** web part redirects users after successful sign-in. The target of the redirect can be specified in two ways:

- The value of the **returnUrl** query string parameter (for example when a user is redirected to the sign-in page after they attempt to access a page that requires authentication).
- If there is no *returnUrl* query string parameter, the redirect target is taken from the web part's **Default target URL** property.

When using the *returnUrl* query string parameter, the redirection is protected and only allows local relative URLs (pages on the same website). If the entered return URL is absolute, the redirect automatically targets the application's root page instead.

The *Default target URL* web part property can only be configured in the administration interface and is considered a trusted source. In this case, no redirect protection is present and absolute URLs with external domain names are allowed.

Performing safe redirects in custom code

If you have any custom functionality or components that perform redirects, consider the possibility of unvalidated redirection attacks. Your code may contain security vulnerabilities if the redirect URL originates from an external input (such as a query string parameter or a posted form field).

✓ **Tip**: To find possible vulnerabilities, search your custom code for occurrences of the **URLHelper.Redirect** Kentico API method and the HttpResponse.Redirect .NET method.

The **safest approach** is to completely avoid redirects to URLs obtained from untrusted inputs.

If you cannot avoid such redirects, use the following methods from the Kentico API to perform the redirects safely:

- **URLHelper.LocalRedirect(string url)** – redirects to the specified URL if it is local, otherwise redirects to the application's root path.
- **URLHelper.IsLocalUrl(string url)** – returns a bool value indicating whether the specified URL is local. Use to create conditions before you perform redirects.

The methods consider URLs to be local if they are relative (starting with ~/ or /). All absolute URLs are evaluated as not local (invalid), even if the domain in the URL matches the current site's domain.

Unvalidated redirect protection on MVC sites

If your site has pages handled by MVC controllers and views, use the System.Web.MVC.UrlHelper.IsLocalUrl method to validate untrusted inputs before performing redirects.

Examples

```
using CMS.Helpers;

...

public void CustomRedirect()
{
        // Gets a redirection URL from the 'returnurl' query string parameter
        string returnUrl = QueryHelper.GetString("returnurl", "");

        // Performs redirection with protection against unvalidated redirect attacks
        URLHelper.LocalRedirect(returnUrl);
}
```

```
using CMS.Helpers;

...

public void CustomRedirect(string returnUrl, bool trustedUrlSource)
{
        // Verifies that the URL is local or originates from a trusted input
        if (trustedUrlSource || URLHelper.IsLocalUrl(returnUrl))
        {
                // Performs standard redirection
                URLHelper.Redirect(returnUrl);
        }
        else
        {
                // Perform a different action for untrusted non-local URLs
        }
}
```