This type of attack is also known as path traversal. The main goal is to show content of a file or directory via an application. Applications read data from the file system in many cases. Paths to these files or directories are often taken from input. If a user's input is not handled carefully, users can read data from the root directory of the server's file system.

Example of directory traversal

Let's inspect the following code:

```
Response.Write("<strong>Absolute path e:\\: </strong><br />");
string[] dirs = Directory.GetDirectories("e:\\");
foreach (string dir in dirs)
        {
            Response.Write(dir + "<br />");
        }

Response.Write("<br /><strong>Read e:\\StorageHelper.cs with absolute path: <
/strong><br />");

Response.Write(File.ReadAllText("e:\\StorageHelper.cs") + "<br />");

Response.Write("<br /><strong>Root path of C: taken by ./../../../: </strong><br />");
string[] dirs2 = Directory.GetDirectories("./../../../");
foreach (string dir in dirs2)
        {
            Response.Write(dir + "<br />");
        }

Response.Write("<br/><strong>Read c:\\StorageHelper.cs with ./../../../: </strong><br
/>");

Response.Write(File.ReadAllText("./../../../StorageHelper.cs") + "<br />");

Response.Write("<br /><strong>Directory with application: taken by ServerMapPath(\"~/..
/\"): </strong><br />");
string[] dirs3 = Directory.GetDirectories(Server.MapPath("~/../"));
foreach (string dir in dirs3)
        {
            Response.Write(dir + "<br />");
        }

//This code throws an exception - but the security is poor here
Response.Write("Directory with application: taken by /../: <br/>");
string[] dirs4 = Directory.GetDirectories(Server.MapPath("/../."));
foreach (string dir in dirs4)
      {
                           Response.Write(dir + "<br />");
      }

Response.Write("<br /><strong>Directory with application: taken by Server.MapPath(\"
/\") + \"../: </strong><br />");
string[] dirs5 = Directory.GetDirectories(Server.MapPath("/") + "../");
foreach (string dir in dirs5)
        {
            Response.Write(dir + "<br />");
        }
```

The result of this code will look like this:

Absolute path e:\:
e:\$RECYCLE.BIN
e:\ASP.NET_TEMP
e:\AzureSLN

Read e:\StorageHelper.cs with absolute path:
************ Some code 2 ************

Root path of C: taken by ./../../../:
./../../../$Recycle.Bin
./../../../Backup
./../../../Boot

Read c:\StorageHelper.cs with ./../../../:
*************************** Some code StorageHelper.cs
****

Directory with application: taken by ServerMapPath("~/../"):
C:\inetpub\wwwroot\3855_10914
C:\inetpub\wwwroot\3889_32518
C:\inetpub\wwwroot\3968_07397

Directory with application: taken by Server.MapPath("/") + "../:
C:\inetpub\wwwroot\../AdminScripts
C:\inetpub\wwwroot\../custerr
C:\inetpub\wwwroot\../history

These are listings from one particular hard disk. The exact paths or listings aren't important, the important thing is that you can simply read any file or list any directory. And you can either use a relative or an absolute path. In this case, it is even a more severe vulnerability called full path disclosure. This bug enables the attacker to see a full path in an error message. The attacker finds out your directory structure and can effectively exploit a directory traversal vulnerability.

What can directory traversal attacks do

The first dangerous thing is that the attacker knows your directory structure. And if there are any sensitive information in files in your application directory, the attacker can simply download these files. A worse case is when your application enables the attacker to read files. The attacker can then read your configuration files (e.g., the connection string in web.config file) or the configuration files of the whole system (if the application is used by a highly privileged user).

Finding directory traversal vulnerabilities

Search for parts of your application where the application reads files and directories. Then try to change the input parameters to get content from a place outside of the application directory. In code, you can search for these strings:

- ServerMapPath
- FileStream
- StreamReader

Avoiding directory traversal

Validate user inputs, check for absolute paths for sequences of "../", and so on. Also, if you read a file, check if the file is within the application path. The application should never read data from a random path on a hard disk.