



## NETSPARKER SCAN REPORT SUMMARY

TARGET URL	<a href="http://localhost:55792/ECSP Labs/">http://localhost:55792/ECSP Labs/</a>
SCAN DATE	4/19/2017 4:51:20 PM
REPORT DATE	4/19/2017 5:08:41 PM
SCAN DURATION	00:15:28
NETSPARKER VERSION	4.8.1.14376-4.8.1-hf1-9a19bce

Total  
Requests  
10230

Average  
Speed  
11.01 req/sec.

21  
Identified

3  
Confirmed

5  
Critical

7  
Informational

## SCAN SETTINGS

**ENABLED ENGINES** SQL Injection, SQL Injection (Boolean), SQL Injection (Blind), Cross-site Scripting, Command Injection, Command Injection (Blind), Local File Inclusion, Remote File Inclusion, Code Evaluation, HTTP Header Injection, Open Redirection, Web App Fingerprint, WebDAV, Reflected File Download, Insecure Reflected Content, XML External Entity, File Upload, Windows Short Filename, Server-Side Request Forgery (Pattern Based), Cross-Origin Resource Sharing (CORS), HTTP Methods, Server-Side Request Forgery (DNS), SQL Injection (Out of Band), XML External Entity (Out of Band), Cross-site Scripting (Blind), Code Evaluation (Out of Band)

**URL REWRITE MODE** Heuristic

**DETECTED URL** None

**REWRITE RULES**

Authentication

Scheduled

## VULNERABILITIES

CRITICAL

24 %

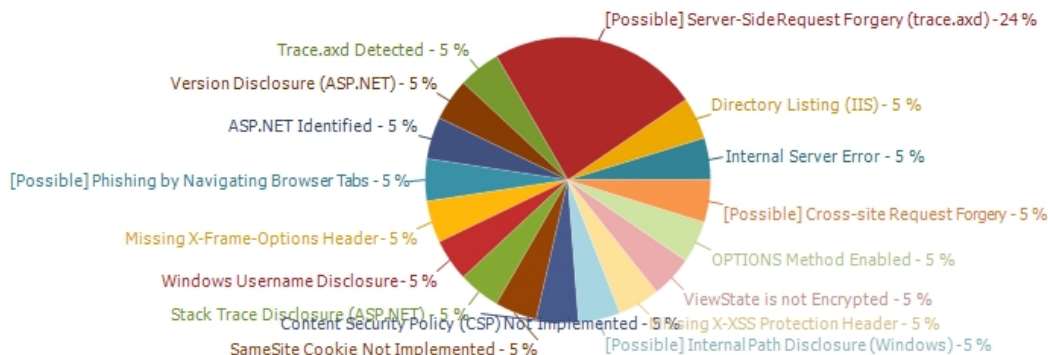
IMPORTANT  
5 %

LOW

38 %

INFORMATION

33 %



# VULNERABILITY SUMMARY

URL	Parameter	Method	Vulnerability	Confirmed
		<a href="#">[Possible] Server-Side Request Forgery (trace.axd)</a>	No	
		<a href="#">[Possible] Server-Side Request Forgery (trace.axd)</a>	No	
		<a href="#">[Possible] Server-Side Request Forgery (trace.axd)</a>	No	
		<a href="#">[Possible] Server-Side Request Forgery (trace.axd)</a>	No	
		<a href="#">[Possible] Server-Side Request Forgery (trace.axd)</a>	No	
		<a href="#">Trace.axd Detected</a>	No	
		<a href="#">Internal Server Error</a>	Yes	
		<a href="#">Version Disclosure (ASP.NET)</a>	No	
		<a href="#">Stack Trace Disclosure (ASP.NET)</a>	No	
		<a href="#">ViewState is not Encrypted</a>	No	
		<a href="#">Missing X-Frame-Options Header</a>	No	
		<a href="#">Windows Username Disclosure</a>	No	
		<a href="#">[Possible] Cross-site Request Forgery</a>	No	
		<a href="#">[Possible] Phishing by Navigating Browser Tabs</a>	No	
		<a href="#">ASP.NET Identified</a>	No	
		<a href="#">Directory Listing (IIS)</a>	No	
		<a href="#">OPTIONS Method Enabled</a>	Yes	
		<a href="#">Missing X-XSS Protection Header</a>	No	

		<a href="#">SameSite Cookie Not Implemented</a>	Yes
		<a href="#">Content Security Policy (CSP) Not Implemented</a>	No
		<a href="#">[Possible] Internal Path Disclosure (Windows)</a>	No

# 1. [Possible] Server-Side Request Forgery (trace.axd)

Netsparker detected a Server-Side Request Forgery based on pattern matching but was unable to confirm the vulnerability.

5 TOTAL

CRITICAL

## Impact

Server-Side Request Forgery allows an attacker to make local and/or remote network requests while masquerading as the target server.

Having trace.axd endpoint that is accessible through SSRF leads to attacker gaining access to request details and server variables of all sessions, among other information.

## Remedy

- Where possible, do not use users' input for URLs.
- If you definitely need dynamic URLs, use whitelisting. Make a list of valid, accepted URLs and do not accept other URLs.
- Ensure that you only accept URLs those are located on the trusted domains.

In addition to above, apply the following changes on your `web.config` file to disable ASP.NET tracing:

```
<System.Web>
  <trace enabled="false" />
</System.Web>
```

## External References

- [CWE-918: Server-Side Request Forgery \(SSRF\)](#)
- [ASP.NET Tracing](#)

## Classification

[OWASP 2013-A5](#) [PCI V3.1-6.5.6](#) [PCI V3.2-6.5.6](#) [CWE-16](#) [CAPEC-347](#) [WASC-15](#) [HIPAA-164.306\(A\), 164.308\(A\)](#)

## CVSS 3.0

CVSS Vector String: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N/E:H/RL:O/RC:C

Base: 7.5 (High)

Temporal: 7.2 (High)

Environmental: 7.2 (High)

### 1.1.

Certainty

Request

Response

### 1.2.

Certainty

Request

Response

### 1.3.

Certainty

Request

Response

### 1.4.


Certainty

Request  
Response

1.5.

---

Certainty

  
Request  
Response

## 2. Trace.axd Detected

1 TOTAL

IMPORTANT

Netsparker detected that ASP.NET tracing is enabled and `Trace.axd` is accessible remotely.

This vulnerability can be used to obtain sensitive data on current sessions.

### Impact

ASP.NET's trace feature is a powerful mechanism that helps developers debug and resolve problems in their applications, but it can also be used by attackers to gain information about requests and responses to the application. An attacker can obtain information such as:

- Session cookies
- Session state
- Query string and POST variables
- Physical path of the requested file
- Execution time

This means that the attacker can hijack almost every active user's session by using their session details.

### Remedy

Apply the following changes on your `web.config` file to disable ASP.NET tracing:

```
<System.Web>
  <trace enabled="false" />
</System.Web>
```

### Classification

[OWASP 2013-A5](#) [PCI V3.1-6.5.6](#) [PCI V3.2-6.5.6](#) [CWE-16](#) [CAPEC-347](#) [WASC-15](#) [HIPAA-164.306\(A\), 164.308\(A\)](#)

### CVSS 3.0

CVSS Vector String: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N/E:H/RL:O/RC:C

Base: 7.5 (High)

Temporal: 7.2 (High)

Environmental: 7.2 (High)

## 2.1.

### Certainty

### Request

### Response

# 3. Internal Server Error

1 TOTAL

LOW

CONFIRMED

1

Netsparker identified an internal server error.

The server responded with an HTTP status 500, indicating there is a server-side error. Reasons may vary, and the behavior should be analyzed carefully. If Netsparker is able to find a security issue in the same resource, it will report this as a separate vulnerability.

## Impact

The impact may vary depending on the condition. Generally this indicates poor coding practices, not enough error checking, sanitization and whitelisting. However, there might be a bigger issue, such as SQL injection. If that's the case, Netsparker will check for other possible issues and report them separately.

## Remedy

Analyze this issue and review the application code in order to handle unexpected errors; this should be a generic practice, which does not disclose further information upon an error. All errors should be handled server-side only.

## Classification

### 3.1. Confirmed

Request

Response

# 4. Version Disclosure (ASP.NET)

1 TOTAL

LOW

Netsparker identified a version disclosure (ASP.NET) in target web server's HTTP response.

This information can help an attacker gain a greater understanding of the systems in use and potentially develop further attacks targeted at the specific version of ASP.NET.

## Impact

An attacker might use the disclosed information to harvest specific security vulnerabilities for the version identified.

## Remedy

Apply the following changes to your `web.config` file to prevent information leakage by using custom error pages and removing `X-AspNet-Version` from HTTP responses.

```
<System.Web>
  <httpRuntime enableVersionHeader="false" />
  <customErrors mode="On" defaultRedirect="~/error/GeneralError.aspx">
    <error statusCode="403" redirect="~/error/Forbidden.aspx" />
    <error statusCode="404" redirect="~/error/PageNotFound.aspx" />
    <error statusCode="500" redirect="~/error/InternalServerError.aspx" />
  </customErrors>
</System.Web>
```

## Remedy References

- [Error Handling in ASP.NET Pages and Applications](#)
- [Remove Unwanted HTTP Response Headers](#)

## Classification

[CWE-205](#) [CAPEC-170](#) [WASC-45](#) [HIPAA-164.306\(A\)](#), [164.308\(A\)](#)

### 4.1.

Certainty



Request

Response



# 5. Stack Trace Disclosure (ASP.NET)

1 TOTAL

LOW

Netsparker identified a stack trace disclosure (ASP.NET) in the target web server's HTTP response.

## Impact

An attacker can obtain information such as:

- ASP.NET version.
- Physical file path of temporary ASP.NET files.
- Information about the generated exception and possibly source code, SQL queries, etc.

This information might help an attacker gain more information and potentially focus on the development of further attacks for the target system.

## Remedy

Apply following changes on your `web.config` file to prevent information leakage by applying custom error pages.

```
<System.Web>
  <customErrors mode="On" defaultRedirect="~/error/GeneralError.aspx">
    <error statusCode="403" redirect="~/error/Forbidden.aspx" />
    <error statusCode="404" redirect="~/error/PageNotFound.aspx" />
    <error statusCode="500" redirect="~/error/InternalError.aspx" />
  </customErrors>
</System.Web>
```

## Remedy References

- [Error Handling in ASP.NET Pages and Applications](#)

## Classification

[OWASP 2013-A5](#) [PCI V3.1-6.5.5](#) [PCI V3.2-6.5.5](#) [CWE-248](#) [CAPEC-214](#) [WASC-14](#) [HIPAA-164.306\(A\), 164.308\(A\)](#)

### 5.1.

Certainty



Request

Response

# 6. ViewState is not Encrypted

1 TOTAL

LOW

Netsparker detected that ViewState encryption is disabled.

## Impact

An attacker can study the application's state management logic for possible vulnerabilities; if your application stores application-critical information in the ViewState, it will also be revealed.

## Remedy

ASP.NET provides encryption for ViewState parameters.

For page based protection, place the following directive at the top of affected page.

```
<%@Page ViewStateEncryptionMode="Always" %>
```

You can also set this option for the whole application by using web.config files. Apply the following configuration for your application's web.config file.

```
<System.Web>
  <pages viewStateEncryptionMode="Always">
</System.Web>
```

## Remedy References

- [ASP.NET View State Security](#)
- [ViewState Security & WebFarms](#)

## Classification

[CWE-16 WASC-15 HIPAA-164.306\(A\), 164.308\(A\)](#)

### 6.1.

Certainty



Request

Response

## 7. Missing X-Frame-Options Header

1 TOTAL

LOW

Netsparker detected a missing `X-Frame-Options` header which means that this website could be at risk of a clickjacking attack.

The `X-Frame-Options` HTTP header field indicates a policy that specifies whether the browser should render the transmitted resource within a `frame` or an `iframe`. Servers can declare this policy in the header of their HTTP responses to prevent clickjacking attacks, which ensures that their content is not embedded into other pages or frames.

### Impact

Clickjacking is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to other another page, most likely owned by another application, domain, or both.

Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker.

### Remedy

- Sending the proper `X-Frame-Options` in HTTP response headers that instruct the browser to not allow framing from other domains.
- Employing defensive code in the UI to ensure that the current frame is the most top level window.

### External References

- [Clickjacking](#)

### Remedy References

- [Clickjacking Defense Cheat Sheet](#)

### Classification

[OWASP 2013-A5](#) [CWE-693](#) [CAPEC-103](#)

## 7.1.

Certainty



Request

Response

# 8. Windows Username Disclosure

1 TOTAL

LOW

Netsparker identified a Windows username disclosure in the error message.

## Impact

An attacker can perform brute-force or dictionary-based password guessing on the disclosed username. It may also help the attacker identify other vulnerabilities or further their exploitation of other identified vulnerabilities.

## Remedy

- Error messages should be disabled.
- Remove this kind of sensitive data from the output.

## External References

- [Top 10 2010-A6-Security Misconfiguration - OWASP](#)
- [OWASP - Information Leakage](#)

## Classification

[PCI V3.1-6.5.5](#) [PCI V3.2-6.5.5](#) [CWE-200](#) [CAPEC-118](#) [WASC-13](#)

### 8.1.

Certainty



Request

Response

# 9. [Possible] Cross-site Request Forgery

1 TOTAL

LOW

Netsparker identified a possible Cross-Site Request Forgery.

CSRF is a very common vulnerability. It's an attack which forces a user to execute unwanted actions on a web application in which the user is currently authenticated.

## Impact

Depending on the application, an attacker can mount any of the actions that can be done by the user such as adding a user, modifying content, deleting data. All the functionality that's available to the victim can be used by the attacker. Only exception to this rule is a page that requires extra information that only the legitimate user can know (such as user's password).

## Remedy

- Send additional information in each HTTP request that can be used to determine whether the request came from an authorized source. This "validation token" should be hard to guess for attacker who does not already have access to the user's account. If a request is missing a validation token or the token does not match the expected value, the server should reject the request.
- If you are posting form in ajax request, custom HTTP headers can be used to prevent CSRF because the browser prevents sites from sending custom HTTP headers to another site but allows sites to send custom HTTP headers to themselves using XMLHttpRequest.

- For native XMLHttpRequest (XHR) object in JavaScript;

```
xhr = new XMLHttpRequest();  
xhr.setRequestHeader('custom-header', 'value');
```

For JQuery, if you want to add a custom header (or set of headers) to

### a. individual request

```
$.ajax({  
  url: 'foo/bar',  
  headers: { 'x-my-custom-header': 'some value' }  
});
```

### b. every request

```
$.ajaxSetup({  
  headers: { 'x-my-custom-header': 'some value' }  
});  
OR  
$.ajaxSetup({  
  beforeSend: function(xhr) {  
    xhr.setRequestHeader('x-my-custom-header', 'some value');  
  }  
});
```

## External References

- [OWASP Cross-Site Request Forgery \(CSRF\)](#)

## Remedy References

- [OWASP Cross-Site Request Forgery \(CSRF\) Prevention Cheat Sheet](#)

## Classification

[OWASP 2013-A8](#) [PCI V3.1-6.5.9](#) [PCI V3.2-6.5.9](#) [CWE-352](#) [CAPEC-62](#) [WASC-9](#) [HIPAA-164.306\(A\)](#)

## 9.1.

### Certainty



### Request

### Response

## 10. [Possible] Phishing by Navigating Browser Tabs

1 TOTAL

LOW

Opened windows through normal hrefs with target="\_blank" can modify window.opener.location and replace the parent webpage with something else, even on a different origin.

While this doesn't allow script execution, it does allow phishing attacks that silently replace the parent tab.

### Impact

If the links lack of rel="noopener noreferrer" attribute, third party site can change the URL of source tab using window.opener.location.assign and trick the user as if he is still in a trusted page and lead him to enter his secret information or credentials to this malicious copy.

### Remedy

To prevent pages from abusing window.opener, use *rel=noopener*. This ensures window.opener is null in Chrome 49 and Opera 36.

For older browsers and in Firefox, you could use *rel=noreferrer* which also disables the Referer HTTP header.

```
<a href="..." target="_blank" rel="noopener noreferrer">...</a>
```

### External References

- [Target=" blank" - the most underestimated vulnerability ever](#)
- [Blankshield & reverse tabnabbing attacks](#)

### Classification

[OWASP 2013-A5](#)

#### 10.1.

Certainty

Request

Response

# 11. ASP.NET Identified

1 TOTAL  
INFORMATION

Netsparker identified that the target website is using ASP.NET as its web application framework.

This issue is reported as extra information only.

## Impact

This issue is reported as additional information only. There is no direct impact arising from this issue.

## Classification

[OWASP-PC-C7](#)

## CVSS 3.0

CVSS Vector String: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N/E:H/RL:O/RC:C

Base: 5.3 (Medium)

Temporal: 5.1 (Medium)

Environmental: 5.1 (Medium)

## 11.1.

### Certainty



### Request

### Response

# 12. Directory Listing (IIS)

1 TOTAL  
INFORMATION

Netsparker identified a directory listing (IIS).

The web server responded with a list of files located in the target directory.

## Impact

An attacker can see the files located in the directory and could potentially access files which disclose sensitive information.

## Actions to Take

- 1. Configure the web server to disallow directory listing requests.
- 2. Ensure that the latest security patches have been applied to the web server and the current stable version of the software is in use.

## External References

- [WASC - Directory Indexing](#)

## Remedy References

- [How to configure Web server permissions for Web content in IIS](#)

## Classification

[OWASP 2013-A5](#) [CWE-548](#) [CAPEC-127](#) [WASC-16](#) [OWASP-PC-C6](#)

## CVSS 3.0

CVSS Vector String: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N/E:H/RL:O/RC:C  
Base: 5.3 (Medium)  
Temporal: 5.1 (Medium)  
Environmental: 5.1 (Medium)

## 12.1.

### Certainty



### Request

### Response



# 13. OPTIONS Method Enabled

1 TOTAL

INFORMATION

CONFIRMED

1

Netsparker detected that OPTIONS method is allowed. This issue is reported as extra information.

## Impact

Information disclosed from this page can be used to gain additional information about the target system.

## Remedy

Disable OPTIONS method in all production systems.

## External References

- [Testing for HTTP Methods and XST \(OWASP-CM-008\)](#)
- [HTTP/1.1: Method Definitions](#)

## Classification

[OWASP 2013-A5](#) [CWE-16](#) [CAPEC-107](#) [WASC-14](#)

### 13.1. Confirmed

Request

Response

# 14. Missing X-XSS Protection Header

1 TOTAL  
INFORMATION

Netsparker detected a missing `X-XSS-Protection` header which means that this website could be at risk of a Cross-site Scripting (XSS) attacks.

## Impact

This issue is reported as additional information only. There is no direct impact arising from this issue.

## Remedy

Add the `X-XSS-Protection` header with a value of `"1; mode= block"`.

- `X-XSS-Protection: 1; mode=block`

## External References

- [MSDN - Internet Explorer 8 Security Features](#)
- [Internet Explorer 8 XSS Filter](#)

## Classification

[HIPAA-164.308\(A\)](#) [OWASP-PC-C9](#)

### 14.1.

Certainty

Request

Response

# 15. SameSite Cookie Not Implemented

1 TOTAL

INFORMATION

CONFIRMED

1

Cookies are typically sent to third parties in cross origin requests. This can be abused to do CSRF attacks. Recently a new cookie attribute named *SameSite* was proposed to disable third-party usage for some cookies, to prevent CSRF attacks.

Same-site cookies allow servers to mitigate the risk of CSRF and information leakage attacks by asserting that a particular cookie should only be sent with requests initiated from the same registrable domain.

## Remedy

The server can set a same-site cookie by adding the SameSite=... attribute to the Set-Cookie header:

Set-Cookie: key=value; SameSite=strict

There are two possible values for the same-site attribute:

- Lax
- Strict

In the strict mode, the cookie is not sent with any cross-site usage even if the user follows a link to another website. Lax cookies are only sent with a top-level get request.

## External References

- [Using the Same-Site Cookies Attribute to Prevent CSRF Attacks](#)
- [Same-site Cookies](#)
- [Preventing CSRF with the same-site cookie attribute](#)

## Classification

[OWASP-PC-C9](#)

### 15.1. Confirmed

Request

Response

# 16. Content Security Policy (CSP) Not Implemented

1 TOTAL  
INFORMATION

CSP is an added layer of security that helps to mitigate mainly Cross-site Scripting attacks.

CSP can be enabled instructing the browser with a Content-Security-Policy directive in a response header;

```
Content-Security-Policy: script-src 'self';
```

or in a meta tag;

```
<meta http-equiv="Content-Security-Policy" content="script-src 'self';">
```

In the above example, you can restrict script loading only to the same domain. It will also restrict inline script executions both in the element attributes and the event handlers. There are various directives which you can use by declaring CSP:

- **script-src**: Restricts the script loading resources to the ones you declared. By default, it disables inline script executions unless you permit to the evaluation functions and inline scripts by the unsafe-eval and unsafe-inline keywords.
- **base-uri**: Base element is used to resolve relative URL to absolute one. By using this CSP directive, you can define all possible URLs which could be assigned to base-href attribute of the document.
- **frame-ancestors**: It is very similar to X-Frame-Options HTTP header. It defines the URLs by which the page can be loaded in an iframe.
- **frame-src / child-src**: frame-src is the deprecated version of child-src. Both define the sources that can be loaded by iframe in the page. (Please note that frame-src was brought back in CSP 3)
- **object-src**: Defines the resources that can be loaded by embedding such as Flash files, Java Applets.
- **img-src**: As its name implies, it defines the resources where the images can be loaded from.
- **connect-src**: Defines the whitelisted targets for XMLHttpRequest and WebSocket objects.
- **default-src**: It is a fallback for the directives that mostly ends with -src suffix. When the directives below are not defined, the value set to default-src will be used instead:
  - child-src
  - connect-src
  - font-src
  - img-src
  - manifest-src
  - media-src
  - object-src
  - script-src
  - style-src

When setting the CSP directives, you can also use some CSP keywords:

- **none**: Denies loading resources from anywhere.
- **self**: Points to the document's URL (domain + port).
- **unsafe-inline**: Permits running inline scripts.
- **unsafe-eval**: Permits execution of evaluation functions such as eval().

In addition to CSP keywords, you can also use wildcard or only a scheme when defining whitelist URLs for the points. Wildcard can be used for subdomain and port portions of the URLs:

```
Content-Security-Policy: script-src https://\*.example.com;
```

```
Content-Security-Policy: script-src https://example.com*;
```

```
Content-Security-Policy: script-src https;
```

It is also possible to set a CSP in Report-Only mode instead of forcing it immediately in the migration period. Thus you can see the violations of the CSP policy in the current state of your web site while migrating to CSP:

```
Content-Security-Policy-Report-Only: script-src 'self'; report-uri: https://example.com;
```

## Impact

There is no direct impact of not implementing CSP on your website. However, if your website is vulnerable to a Cross-site Scripting attack CSP can prevent successful exploitation of that vulnerability. By not implementing CSP you'll be missing out this extra layer of security.

## Actions to Take

- Enable CSP on your website by sending the Content-Security-Policy in HTTP response headers that instruct the browser to apply the policies you specified.
- Apply the whitelist and policies as strict as possible.
- Rescan your application to see if Netsparker identifies any weaknesses in your policies.

## Remedy

Enable CSP on your website by sending the Content-Security-Policy in HTTP response headers that instruct the browser to apply the policies you specified.

## External References

- [An Introduction to Content Security Policy](#)
- [Content Security Policy \(CSP\)](#)

## Classification

[OWASP-PC-C9](#)

## 16.1.

---

Certainty



Request

Response

# 17. [Possible] Internal Path Disclosure (Windows)

1 TOTAL

INFORMATION

Netsparker identified a possible Internal Path Disclosure (Windows) in the document.

## Impact

There is no direct impact, however this information can help an attacker identify other vulnerabilities or help during the exploitation of other identified vulnerabilities.

## Remedy

Ensure this is not a false positive. Due to the nature of the issue, Netsparker could not confirm that this file path was actually the real file path of the target web server.

- Error messages should be disabled.
- Remove this kind of sensitive data from the output.

## External References

- [OWASP - Full Path Disclosure](#)

## Classification

[CWE-200](#) [CAPEC-118](#) [WASC-13](#) [HIPAA-164.306\(A\), 164.308\(A\)](#) [OWASP-PC-C7](#)

## 17.1.

### Certainty



### Request

### Response