System Information Leak: Incomplete Servlet Error Handling Mitigation SOP

# Development Mitigation SOP

If a Servlet fails to catch all exception, it may reveal debugging information that will help an adversary form an attack plan. When a Servlet throws an exception, the default error response the Servlet container sends back to the user typically includes debugging information. This information is of great value to an attacker. For example, a stack trace might show the attacker a malformed SQL query string, the type of database being used and the version of the application container. This information enables the attacker to target known vulnerabilities in these components.

# Defense Against [DEFECT]

A good error handling mechanism always tries to capture all exceptions and returns a generic error message that does not reveal any details about the error and the application.

# Examples

## General Example

…

Protected void doPost (HttpServletRequest req, HttpServletResponse res) throws IOException {

String ip = req.getRemoteAddr();

InetAddress addr = InetAddress.getByName(ip);

…

out.println(“hello “ + addr.getHostName());

}

…

## Explanation

A DNS lookup failure will cause the Servlet to throw an exception.

## Recommendation

Depending on the platform and the container the application is running on, there can be different options. The following are examples of how to resolve the issue.

* Set a generic custom error page for all unhandled exceptions at the container level. The generic custom error page should have a simple error message that does not reveal any details about the exception.
* Use a global error handler to capture all unhandled exceptions.
* Handle the error in the page level.

**Resources**

1. [OWASP](https://www.owasp.org/index.php/System_Information_Leak:_Missing_Catch_Block)