Report 07

URL: wiki.grab.com

Stack Trace Disclosure (Java)

MEDIUM

The Stack Trace Disclosure vulnerability refers to a security issue in Java-based web applications where sensitive information, specifically the stack trace, is exposed to potential attackers. When an application encounters an error or exception during its execution, it generates a stack trace, which contains a detailed record of the program's execution flow leading up to the error. However, in certain cases, this stack trace is inadvertently disclosed in the application's HTTP response, making it accessible to anyone who interacts with the application. This disclosure typically occurs when error messages or exception details are not properly handled or filtered in the code. The stack trace can contain critical information that can aid attackers in understanding the internal workings of the application and potentially identifying vulnerabilities

Affected URL: https://wiki.grab.com/login.action?os_destination='%22/$[].%3E&permissionViolation=true

Method	Parameter,	Parameter Type,	Value
GET	os_destination	Querystring	`'"/$[].>`
GET	permissionViolation	Querystring	true

Affected Components

The Stack Trace Disclosure (Java) vulnerability primarily affects Java-based web applications. Specifically, it relates to the handling of errors or exceptions within the application's codebase.

- ❖ Web Application Frameworks: Java web application frameworks such as Spring, JavaServer Faces (JSF), Java Servlet API, Apache Struts, Play Framework, and others can be susceptible to stack trace disclosure if error handling is not adequately implemented.

- ❖ Application Servers: Stack trace disclosure can impact the application servers that host Java-based web applications. Common application servers like Apache Tomcat, JBoss, WebLogic, WebSphere, and GlassFish can be affected if error responses are not properly controlled.

- ❖ Web Containers: The vulnerability can affect the web containers responsible for executing Java web applications. This includes servlet containers like Apache Tomcat, Jetty, and Resin, as well as Java EE containers such as WebLogic and JBoss.

- ❖ Custom Code: Any custom code within the Java application that handles errors and exceptions can potentially introduce stack trace disclosure vulnerabilities if not implemented securely.

Impact Assessment

- ➢ Information Exposure: The vulnerability exposes sensitive information, including error messages, code snippets, and execution paths, through the disclosed stack trace. Attackers can leverage this information to gain insights into the application's inner workings, potentially leading to the exposure of confidential data, proprietary algorithms, or system configurations.

- Increased Attack Surface: By obtaining details about the Tomcat version, physical file paths, and exception information, attackers can gain a deeper understanding of the target system. This knowledge expands their attack surface, allowing them to identify potential vulnerabilities, devise targeted attacks, and focus their efforts on specific areas that may be more susceptible to exploitation.

- System Compromise: Stack trace disclosure can serve as a stepping stone for further attacks. Armed with the exposed information, attackers can refine their strategies, exploit vulnerabilities, and attempt to compromise the system. This can result in unauthorized access, data breaches, disruption of services, or even complete system compromise.

- Reputational Damage: A successful exploitation of the vulnerability can lead to significant reputational damage for the organization. If customer data, sensitive information, or intellectual property is compromised, it can erode trust and confidence in the organization's ability to safeguard critical assets.

- Compliance and Legal Consequences: Depending on the industry and jurisdiction, the exposure of sensitive information may lead to legal and regulatory implications. Organizations may face penalties, legal actions, or regulatory scrutiny for failing to adequately protect sensitive data and comply with applicable privacy and security requirements.

- Operational Disruption: If an attacker successfully exploits the vulnerability, the affected system may experience operational disruptions, such as service interruptions, system crashes, or unauthorized modifications. This can result in financial losses, customer dissatisfaction, and disruption of business operations.

Steps to Reproduce

- Identify the Target: Select a Java-based web application or framework known to be susceptible to stack trace disclosure vulnerabilities. This can include popular frameworks like Spring, Apache Struts, or JavaServer Faces (JSF), or custom-developed applications.

- Generate an Error or Exception: Trigger an error or exception within the application. This can be done by intentionally providing invalid input, manipulating the application's configuration, or forcing a condition that leads to an error.

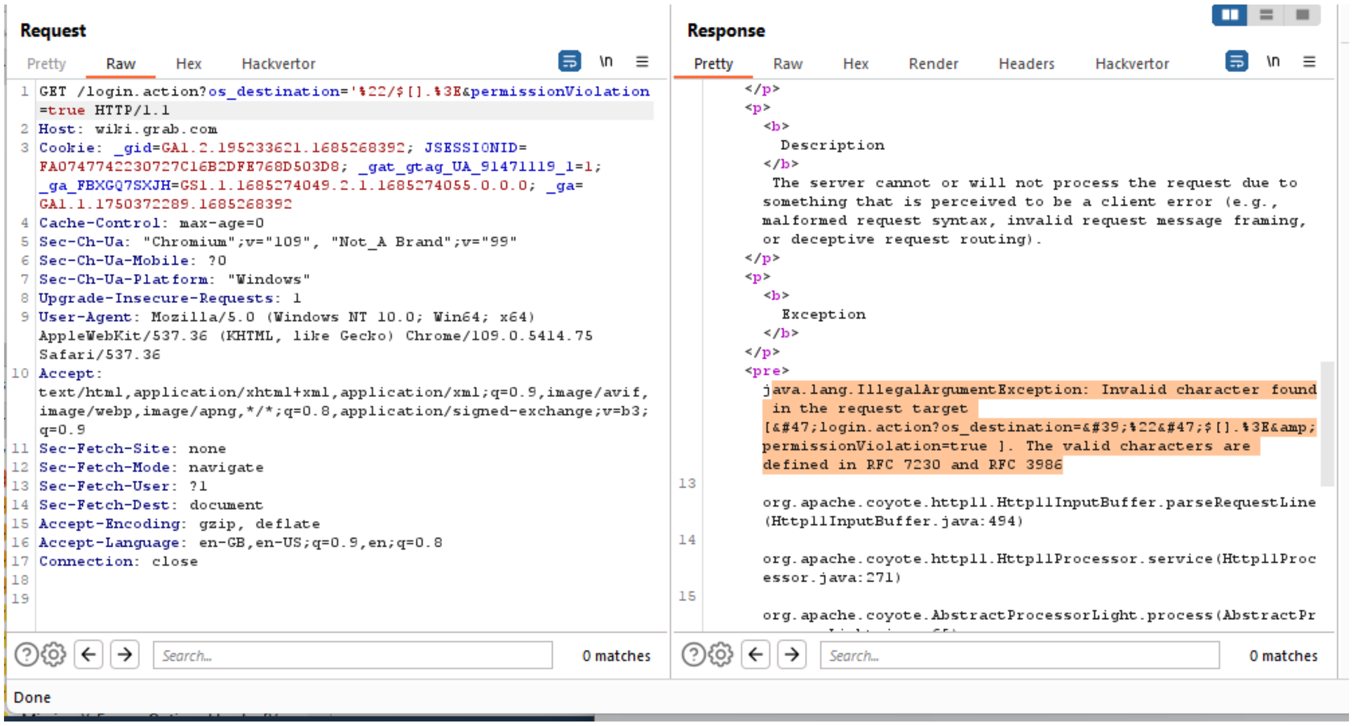
- Capture the HTTP Response: Intercept the HTTP response from the application, either by using a web debugging proxy tool like Burp Suite or by examining the server logs.

- Analyze the Response: Inspect the response for any indications of stack trace disclosure. Look for error messages, exception details, or code snippets that are included in the response. These details may provide insights into the internal workings of the application.

- Extract the Stack Trace: If the response contains a stack trace, extract and analyze it. Pay attention to any sensitive information exposed, such as system paths, database connection details, or specific error messages that reveal implementation details.

- Verify the Impact: Assess the potential impact of the disclosed stack trace information. Consider how an attacker could leverage this information to gain further insights, identify vulnerabilities, or craft targeted attacks.

Proof of Concept



Request

Pretty Raw Hex Hackvertor

```
GET /login.action?os_destination='%22/$[].%3E&permissionViolation=true HTTP/1.1
Host: wiki.grab.com
Cookie: _gid=GA1.2.195233621.1685268392; JSESSIONID=FA0747742230727C16B2DFE768D503D8; _gat_gtag_UA_91471119_1=1; _ga_FBXGQ7SXJH=GS1.1.1685274049.2.1.1685274055.0.0.0; _ga=GA1.1.1750372289.1685268392
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="109", "Not_A Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/109.0.5414.75 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close


```

Response

Pretty Raw Hex Render Headers Hackvertor

```
    </p>
    <p>
      <b>
        Description
      </b>
       The server cannot or will not process the request due to
      something that is perceived to be a client error (e.g.,
      malformed request syntax, invalid request message framing,
      or deceptive request routing).
    </p>
    <p>
      <b>
        Exception
      </b>
    </p>
    <pre>
      java.lang.IllegalArgumentException: Invalid character found
       in the request target
      [&#47;login.action?os_destination=&#39;%22&#47;$[].%3E&amp;
      permissionViolation=true ]. The valid characters are
      defined in RFC 7230 and RFC 3986

      org.apache.coyote.http11.Http11InputBuffer.parseRequestLine
      (Http11InputBuffer.java:494)

      org.apache.coyote.http11.Http11Processor.service(Http11Proc
      essor.java:271)

      org.apache.coyote.AbstractProcessorLight.process(AbstractPr
```

Proposed Mitigation or Fix

- ✓ Secure Error Handling: Implement proper error handling mechanisms throughout the application code. Ensure that error messages and stack traces are not directly exposed to end users or attackers in the production environment. Instead, consider logging error details securely or presenting user-friendly error messages without divulging sensitive information.

- ✓ Filter and Sanitize Error Responses: Apply input validation and sanitization techniques to error messages and exception details to prevent the disclosure of sensitive information. Remove any potentially sensitive data, such as file paths, system configurations, or proprietary algorithms, from error responses before they are presented to users or logged.

- ✓ Customize Error Pages: Create custom error pages that provide a generic error message to users without revealing specific technical details. These custom error pages should be designed to maintain user experience while avoiding the disclosure of sensitive information through stack traces.

- ✓ Limit Error Detail Access: Restrict access to detailed error information and stack traces. Ensure that only authorized administrators or developers have access to detailed error logs and stack trace information. Implement proper access controls and authentication mechanisms to prevent unauthorized access to error logs and sensitive error details.

- ✓ Patch and Update Dependencies: Keep the Java framework, web server, and associated dependencies up to date. Stay informed about security patches and updates provided by the framework and promptly apply them to address any known vulnerabilities.

- ✓ Web Application Firewall (WAF): Consider implementing a WAF that can help detect and block attempts to exploit stack trace disclosure vulnerabilities. WAFs can provide an additional layer of protection by filtering out potentially malicious requests and responses.

- ✓ Apply the following configuration to your web.xml file to prevent information leakage by applying custom error pages.

```
<error-page>
          <error-code>500</error-code>
          <location>/server_error.html</location>
</error-page>
```