XXE Prevention

We have seen that XXE vulnerabilities mainly occur when an unsafe XML input references an external entity, which is eventually exploited to read sensitive files and perform other actions. Preventing XXE vulnerabilities is relatively easier than preventing other web vulnerabilities, as they are caused mainly by outdated XML libraries.

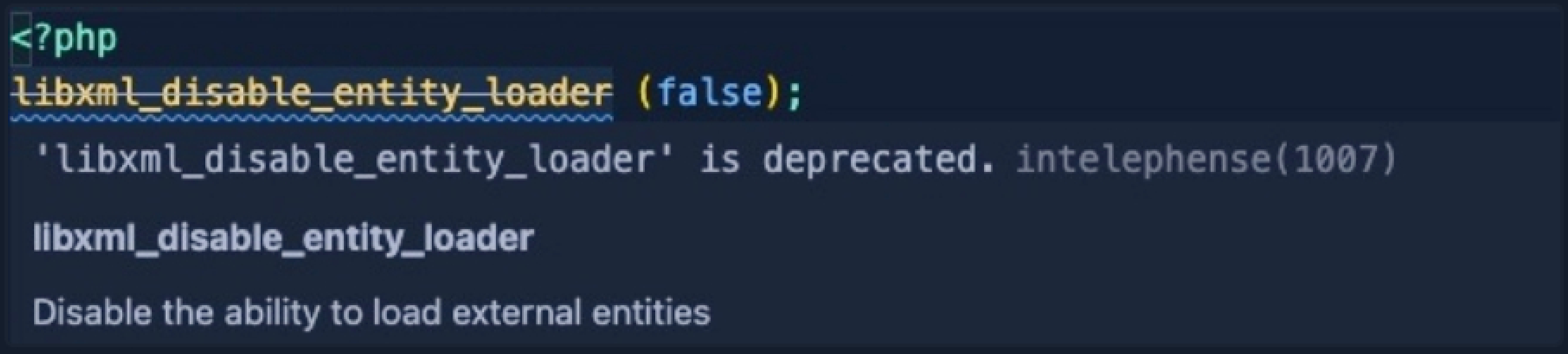
Avoiding Outdated Components

While other input validation web vulnerabilities are usually prevented through secure coding practices (e.g., XSS, IDOR, SQLi, OS Injection), this is not entirely necessary to prevent XXE vulnerabilities. This is because XML input is usually not handled manually by the web developers but by the built-in XML libraries instead. So, if a web application is vulnerable to XXE, this is very likely due to an outdated XML library that parses the XML data.

For example, PHP's libxml_disable_entity_loader function is deprecated since it allows a developer to enable external entities in an unsafe manner, which leads to XXE vulnerabilities. If we visit PHP's documentation for this function, we see the following warning:

Warning This function has been *DEPRECATED* as of PHP 8.0.0. Relying on this function is highly discouraged.

Furthermore, even common code editors (e.g., VSCode) will highlight that this specific function is deprecated and will warn us against using it:



```
<?php
libxml_disable_entity_loader (false);
 'libxml_disable_entity_loader' is deprecated. intelephense(1007)
 libxml_disable_entity_loader
 Disable the ability to load external entities
```

Note: You can find a detailed report of all vulnerable XML libraries, with recommendations on updating them and using safe functions, in OWASP's XXE Prevention Cheat Sheet.

In addition to updating the XML libraries, we should also update any components that parse XML input, such as API libraries like SOAP. Furthermore, any document or file processors that may perform XML parsing, like SVG image processors or PDF document processors, may also be vulnerable to XXE vulnerabilities, and we should update them as well.

These issues are not exclusive to XML libraries only, as the same applies to all other web components (e.g., outdated `Node Modules`). In addition to common package managers (e.g. `npm`), common code editors will notify web developers of the use of outdated components and suggest other alternatives. In the end, `using the latest XML libraries and web development components can greatly help reduce various web vulnerabilities`, including XXE.

Using Safe XML Configurations

Other than using the latest XML libraries, certain XML configurations for web applications can help reduce the possibility of XXE exploitation. These include:

- Disable referencing custom `Document Type Definitions (DTDs)`
- Disable referencing `External XML Entities`
- Disable `Parameter Entity` processing
- Disable support for `XInclude`
- Prevent `Entity Reference Loops`

Another thing we saw was Error-based XXE exploitation. So, we should always have proper exception handling in our web applications and `should always disable displaying runtime errors in web servers`.

Such configurations should be another layer of protection if we miss updating some XML libraries and should also prevent XXE exploitation. However, we may still be using vulnerable libraries in such cases and only applying workarounds against exploitation, which is not ideal.

With the various issues and vulnerabilities introduced by XML data, many also recommend `using other formats, such as JSON or YAML`. This also includes avoiding API standards that rely on XML (e.g., SOAP) and using JSON-based APIs instead (e.g., REST).

Finally, using Web Application Firewalls (WAFs) is another layer of protection against XXE exploitation. However, we should never entirely rely on WAFs and leave the back-end vulnerable, as WAFs can always be bypassed.

Previous Next Mark Complete & Next

Cheat Sheet

Table of Contents

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left