**XXE**

Définition

XML External Entities attacks benefit from an XML feature to build documents dynamically at the time of processing. An XML entity allows inclusion of data dynamically from a given resource. External entities allow an XML document to include data from an external URI. Unless configured to do otherwise, external entities force the XML parser to access the resource specified by the URI, e.g., a file on the local machine or on a remote system. This behavior exposes the application to XML External Entity (XXE) attacks, which can be used to perform denial of service of the local system, gain unauthorized access to files on the local machine, scan remote machines, and perform denial of service of remote systems.

Les attaques XML External Entities bénéficient d'une fonctionnalité XML pour créer des documents dynamiquement au moment du traitement. Une entité XML permet l'inclusion dynamique de données à partir d'une ressource donnée. Les entités externes permettent à un document XML d'inclure des données provenant d'un URI externe. Sauf configuration contraire, les entités externes forcent l'analyseur XML à accéder à la ressource spécifiée par l'URI, par exemple un fichier sur la machine locale ou sur un système distant. Ce comportement expose l'application aux attaques XML Entity External (XXE), qui peuvent être utilisées pour déni de service du système local, obtenir un accès non autorisé aux fichiers sur la machine locale, analyser des machines distantes et effectuer un déni de service des systèmes distants .

An XML External Entity attack is a type of attack against an application that parses XML input. This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser. This attack may lead to the disclosure of confidential data, denial of service, server side request forgery, port scanning from the perspective of the machine where the parser is located, and other system impacts.

XML Entities are like variables: they will expand to a defined value once they are processed by the XML parser. Even many who are not familiar with XML might have seen such an entity in a different place, such as HTML code. For example Instead of copy and pasting characters like the copyright symbol (©) you can conveniently write &copy; and the browser will display it correctly. These are predefined and can't be changed using HTML code.

However, XML gives you a way to define your own entities in order to make coding and configuration easier. An example (in an XML file) where this is necessary is in a customizable configuration file in a web application. For example you need to use a website’s name in multiple locations, such as the page title or footer, or for email templates. Instead of setting the name in each and every single XML tag, it's easier to define the XML Entity called &sitename; in which to hold this information and use it where necessary. So whenever you want to change the website name, you need only change it in the entity definition once, not in every single XML tag.

XML has other advantages too. For instance, XML allows you to define external entities. These are entities that may contain the content from a remote website or API endpoint. When called the XML parser will automatically fetch the external entity and include it in the XML file. That way, you can easily change the meaning of multiple documents without having to edit them manually.

However, there is one problem. When it is possible to pass XML documents to a parser that supports external entities, attackers can retrieve content from a website behind a firewall, issue requests to certain services, and even disclose the content of files stored on the server. Because entities can be referenced within entity definitions, attackers can craft an XML document that contains only 10 entities, but that will eventually expand to a billion entities once it is parsed. This is also known as the 'Billion Laughs Attack'.

Attackers can easily exploit vulnerabilities in XML processors, by uploading malicious XML files that may contain unwanted codes and thus exploiting the vulnerable code, dependencies and integration processes.

These flaws results in data extraction, remote code execution, DOS attacks and sometimes complete system compromise.

Causes

With XML entities, the ‘SYSTEM’ keyword causes an XML parser to read data from a URI and permits it to be substituted in the document. Thus, an attacker can send his own values through the entity and make the application display it. In simple words, an attacker forces the XML parser to access the resource specified by him which could be a file on the system or on any remote system. For example, the below code would fetch the folder/file present on that system and display it to the user.

Vecteurs

* Parseur XML

Préventions

* Désactiver le parsing des entités XML externes et des DTD
* Si SOAP est utilisé, utiliser une version postérieure à la version 1.2
* Ne pas accepter les requêtes dont le Content-Type est text/xml ou application/xml lorsque des donées en format XML ne sont pas attendues
* Valider les inputs XML

Supplémentaires

* Utiliser un format autre que XML (Json par exemple)