All this text wasn’t written by me (Tiago a.k.a TigaxMT).

All of it was transcribed by me from HackerOne video lessons.

All credits and thanks go to them.

XML External Entities

XML gives you the ability to define new entities, e.g. turn &foo; into bar. Normally, these are simple text replacements that are inlined in XML declarations, but they can reference external files, optionally.

XXE attacks take advantage of the ability to read arbitrary files and URIs enable a variety of attacks scenarios.

The most obvious option is the ability to read files. If you have, say, a report system, you could read a file and embed it into data that you can see on a web interface.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <!DOCTYPE foo [
3      <!ELEMENT foo ANY >
4      <!ENTITY xxe SYSTEM "file:///etc/passwd" >
5  ]>
6  <foo>&xxe;</foo>
```

One of the more advanced ways in which this can be used is to perform network requests behind the firewall. The web server will make a connection to the URI you give, which could allow you to violate constraints in interesting way, e.g. requesting backend data directly, in a “trusted” zone.

Often, you will not see the data that you upload; it will simply go to something on the backend. In such a case, XXE can often be used as an oracle to determine if a file exists or a URI can be resolved.

Mitigating XXE attacks is generally straightforward and simply requires passing additional flags to your XML parser. Generally speaking, there will be flags for disallowing inline (Document Type Definition - DTDs) and removing entities entirely. The use of either will mitigate the vulnerability in whole.

REAL WORLD IMPACT

This may seem like a fairly uninteresting bug on the whole, but an XXE bug was discovered on a Twitter service, allowing arbitrary file reads. This was reported by Josh Brodie via HackerOne anda bounty of $10,800 was paid.
https://hackerone.com/reports/248668