**XDoS Attack ( XML denial-of-service attack )**

**Description 1**

An XML denial-of-service attack (XDoS attack) is a content-borne denial-of-service attack whose purpose is to shut down a web service or system running that service. A common XDoS attack occurs when an XML message is sent with a multitude of digital signatures and a naive parser would look at each signature and use all the CPU cycles, eating up all resources. These are less common than inadvertent XDoS attacks which occur when a programming error by a trusted customer causes a handshake to go into an infinite loop. [1]

**Description 2**

XML DoS attacks are type of application-level DoS attacks and are extremely asymmetric. To deliver the attack payload, an attacker needs to spend only a fraction of the processing power or bandwidth that the victim needs to spend to handle the payload. Worse still, DoS vulnerabilities in code that processes XML are also extremely widespread. Even if you’re using thoroughly tested parsers like those found in the Microsoft .NET Framework System.Xml classes, your code can still be vulnerable unless you take explicit steps to protect it.

XML bomb attack

One type of especially nasty XML DoS attack is the XML bomb—a block of XML that is both well-formed and valid according to the rules of an XML schema but which crashes or hangs a program when that program attempts to parse it. The best-known example of an XML bomb is probably the Exponential Entity Expansion attack. This attack is based on the three properties of XML which are;

Substitution entities

Nested entities

Inline DTDs

An attacker can take advantage of these three properties of XML to craft a malicious XML bomb. The attacker writes an XML document with nested entities with more than just one level deep. He nests his entities many levels deep, as shown here:

<?xml version="1.0"?>

<!DOCTYPE lolz [

<!ENTITY lol "lol">

<!ENTITY lol2 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">

<!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">

<!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">

<!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">

<!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">

<!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">

<!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">

<!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">

]>

<lolz>&lol9;</lolz>

This attack is called "Billion Laughs Attack". It should be noted that this XML is both well-formed and valid according to the rules of the DTD. When an XML parser loads this document, it sees that it includes one root element, “lolz”, that contains the text “&lol9;”. However, “&lol9;” is a defined entity that expands to a string containing ten “&lol8;” strings. Each “&lol8;” string is a defined entity that expands to ten “&lol7;” strings, and so forth. After all the entity expansions have been processed, this small (< 1 KB) block of XML will actually contain a billion “lol”s, taking up almost 3GB of memory!

Defending Against XML Bombs

The easiest way to defend against all types of XML entity attacks is to simply disable altogether the use of inline DTD schemas in your XML parsing objects. This is a straightforward application of the principle of attack surface reduction: if you’re not using a feature, turn it off so that attackers won’t be able to abuse it. [2] [3]

**Reference**

**[1]** [**https://en.wikipedia.org/wiki/XML\_denial-of-service\_attack**](https://en.wikipedia.org/wiki/XML_denial-of-service_attack)

**[2]** [**http://download.microsoft.com/download/b/1/9/b1919f9a-4cde-4b67-8b68-c143f9defcd9/msdnmagazine2009\_11en-us.pdf**](http://download.microsoft.com/download/b/1/9/b1919f9a-4cde-4b67-8b68-c143f9defcd9/msdnmagazine2009_11en-us.pdf)

**[3]** [**http://mytechnicaldocs.blogspot.com/2012/04/xml-dos-attacks.html**](http://mytechnicaldocs.blogspot.com/2012/04/xml-dos-attacks.html)