به نام خدا

مروری بر کدهای مربوط به XSS

Reviewing Code for Cross-site scripting

یادآوری

حمله ی XSS زمانی اتفاق می افتد که هکر از برنامه ی تحت وب برای ارسال کدهای مخرب خود استفاده نماید. این کار عموما در شکل اسکریپت سمت مرورگر و ارسال آن به کاربر نهاییِ دیگری صورت می پذیرد. رخنه هایی که باعث موفقیت آمیز بودن این حمله می شوند بسیار شایع بوده و در هر برنامه ی وبی که ورودی کاربر را بدون اعتبارسنجی و یا کدینگ به او بازگرداند، رخ می دهد.

نمونه هایی از کدهای آسیب پذیر

در صورتی که در یک باکس متنی، ورودی ای که کاربر وارد می کند بدون کدینگ مناسب به او بازگردانده شود، مرورگر مقدار ورودی که می تواند شامل اسکریپت هم باشد را به عنوان بخشی از صفحه تفسیر کرده و کدهای آن را اجرا می کند. برای جلوگیری از این آسیب پذیری لازم است که کارهای امنیتی زیر را روی کدهای خود انجام دهیم.

۱. اعتبارسنجی داده ها

۲. کد کردن خروجی های نا امن

```
import org.apache.struts.action.*;
import org.apache.commons.beanutils.BeanUtils;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public final class InsertEmployeeAction extends Action {

public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) throws
Exception{

// Setting up objects and vairables.

Obj1 service = new Obj1();
ObjForm objForm = (ObjForm) form;
InfoADT adt = new InfoADT ();
BeanUtils.copyProperties(adt, objForm);

     String searchQuery = objForm.getqueryString();
     String payload = objForm.getPayLoad();
try {
service.doWork(adt);  / /do something with the data
ActionMessages messages = new ActionMessages();
ActionMessage message = new ActionMessage("success", adt.getName() );
messages.add( ActionMessages.GLOBAL_MESSAGE, message );
saveMessages( request, messages );
request.setAttribute("Record", adt);
return (mapping.findForward("success"));
}
catch( DatabaseException de )
{
ActionErrors errors = new ActionErrors();
ActionError error = new
```

```
ActionError("error.employee.databaseException" + "Payload:
"+payload);
errors.add( ActionErrors.GLOBAL_ERROR, error );
saveErrors( request, errors );
return (mapping.findForward("error: "+ searchQuery));
}
}
}
```

کد بالا نشان دهنده ی بعضی از مشکلات رایجی است که در توسعه چنین classهایی وجود دارد. داده های ارسالی به HttpServletRequest بدون هیچ اعتبارسنجی درون یک پارامتر قرار می گیرند.

اگر بخواهیم روی XSS تمرکز کنیم؛ مشاهده می نماییم که این کلاس پیامی را باز می گرداند. اگر تابع کار خود را با موفقیت انجام دهد پیام ActionMessage تولید می شود و اگر خطایی در برنامه و در بلوک Try/Catch رخ دهد، داده هایی که در HttpServletRequest وجود دارند، بدون هیچ اعتبارسنجی و دقیقا در همان شکل و فرمتی که کاربر وارد کرده به وی بازگردانده می شود.

```
import java.io.*;
import javax.servlet.http.*;
import javax.servlet.*;

public class HelloServlet extends HttpServlet
{
public void doGet (HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException
{

String input = req.getHeader("USERINPUT");

PrintWriter out = res.getWriter();
out.println(input);  // echo User input.
out.close();
```

```
}
}
```

در ادامه، مثالی دیگر از تابعی با آسیب پذیری XSS را آورده ایم. چاپ ورودی های کاربر به او و بدون اعتبارسنجی آن، باعث به وجود آمدن این آسیب پذیری می شود.

مثالی از .NET. در ASP.NET نسخه 1.1 و نسخه 2.0

کد سمت سرور برای یک برنامه ای که به زبان VB.NET نوشته شده است نیز ممکن است عملکردی مشابه داشته باشد:

```
' SearchResult.aspx.vb
Imports System
Imports System.Web
Imports System.Web.UI
Imports System.Web.UI.WebControls

Public Class SearchPage Inherits System.Web.UI.Page

Protected txtInput As TextBox
Protected cmdSearch As Button
Protected lblResult As Label Protected

Sub cmdSearch _Click(Source As Object, _ e As EventArgs)

// Do Search…..
     // ……

lblResult.Text="You Searched for: " & txtInput.Text

// Display Search Results…..
// ……
```

```
End Sub
End Class
```

قطعه کد بالا، که به زبان VB.NET است برای عمل جستجو نوشته شده و در آن، ورودی کاربر روی صفحه چاپ می شود. برای مقابله با این آسیب پذیری می بایست داده ها را به صورت مناسب اعتبارسنجی کرده و در حمله های XSS ذخیره شده (Stored)، داده های ورودی را کدینگ کنیم. توجه داشته باشید که در صورت اعتبارسنجی مناسبِ اعلان ها، تعریف متغیرها و ... توسط برنامه نویس، می توانستیم از این حمله جلوگیری کنیم. (یعنی ASPX regexp validator یا routine و validateRequest not set to False)

مثالی از ASP کلاسیک

ASP کلاسیک نیز مثل سایر تکنولوژی های وب دارای آسیب پذیری XSS است:

```
<%
    ...
    Response.Write "<div class='label'>Please confirm your
data</div><br />"
    Response.Write "Name: " & Request.Form("UserFullName")
    ...
 %>
```

راه های مقابله با XSS

در فریم ورک .NET. توابعی از پیش ساخته برای اعتبارسنجی داده ها و کدینگ HTML به نام های request validation و HttpUtility.HtmlEncode وجود دارد.

اگرچه که مایکروسافت این امکانات را فراهم کرده، اما بهتر است که تنها روی آن ها تکیه نکرده و خودتان نیز داده ها را اعتبارسنجی کنید. مثلا با استفاده از عبارات منظم (regular expression) در صفحات شخصی و به شکل زیر، می توان گزینه اعتبارسنجی درخواست ها را غیر فعال کرد:

```
<%@ Page validateRequest="false" %>
```

یا با تنظیم "ValidateRequest="false در المنت @Pages

یا در web.config با اضافه کردن عبارت زیر:

```
<pages> element with validateRequest="false"
```

بنابراین زمانی که کد خود را بازبینی می کنید از فعال بودن دستور validateRequest مطمئن شده و ببینید که برای اعتبارسنجی داده ها از چه روشی استفاده می شود. فایل Machine.config را برای فعال بودن اعتبارسنجی ASP.NET Request چک کنید. اعتبارسنجی درخواست ها به صورت پیش فرض توسط ASP.NET فعال است. بنابراین می بایست دستور زیر را در فایل Machine.config مشاهده نمایید:

```
<pages validateRequest="true" ... />
```

کدینگ HTML

کد کردن خروجی به سادگی و با استفاده از تابع HTMLEncode امکان پذیر است. این کار به شکل زیر صورت می پذیرید:

```
Server.HtmlEncode(string)
```

برای مثال استفاده از کدینگ HTML در یک فرم به شکل زیر است:

Text Box: <%@ Page Language="C#" ValidateRequest="false" %>

```
<script runat="server">
void searchBtn _Click(object sender, EventArgs e) {
Response.Write(HttpUtility.HtmlEncode(inputTxt.Text)); }
</script>
<html>
<body>
<form id="form1" runat="server">
<asp:TextBox ID="inputTxt" Runat="server" TextMode="MultiLine"
Width="382px" Height="152px">
</asp:TextBox>
<asp:Button ID="searchBtn" Runat="server" Text="Submit" OnClick="
searchBtn _Click" />
</form>
</body>
```

```
</html>
```

تابع کدگذاری در صفحات ASP کلاسیک بسیار شبیه ASP.NET است.

```
Response.Write Server.HtmlEncode(inputTxt.Text)
```

XSS از نوع ذخیره شده Stored

استفاده از کدینگ HTML برای کد کردن خروجی هایی که می تواند نا امن باشد

اسکریپت های مخرب ممکن است در دیتابیس ذخیره/پایدار شوند و تا زمانی که کاربر دیگری آن را فراخواند نکند، اجرا نشوند. این مورد اکثرا در سایت های خبری و یا ایمیل های تحت وب قدیمی رخ می دهد. این حمله ممکن است برای زمان زیادی منتظر بماند تا بالاخره کاربری تصمیم به اجرای صفحه ای که اسکریپت در آن تزریق شده است، بگیرد. بعد از آن اسکریپت در مرورگر قربانی اجرا خواهد شد.

مکان اصلی، برای وارد کردن داده ها که به وسیله ی آن اسکریپت خود را تزریق می کنیم، ممکن است یک برنامه ی آسیب پذیر دیگری باشد. یعنی ممکن است برنامه ی تحت وبی که خروجی را به کاربر نشان می دهد، دارای اعتبارسنجی مناسبی برای داده ها بوده و آسیب پذیر نباشد اما برنامه ای که به وسیله ی آن سایت خبری را تغذیه و مقداردهی می کنیم، آسیب پذیر باشد. در مدل ها و معامری های تجاری استفاده از یک برنامه برای تولید و محتوای یک برنامه ی دیگر، کاری مرسوم و رایج است. در این حالت تمی توانیم به صورت ۱۰۰٪ای از امن بودن داده هایی که به کاربر نمایش می دهیم، مطمئن شویم (چرا که ممکن است آسیب پذیری از طرف برنامه ی دیگری باشد.) راه مقابله ای که برای جلوگیری از این حمله وجود دارد، این است که مطمئن شویم که داده هایی که به کاربر نمایش داده می شوند، از نظر مرورگر به عنوان دستورات (mark-up) تفسیر نشده و صرفا مقدار آن ها نمایش داده می شوند.

برای مقابله با این «دشمن درونی» می بایست داده ها را کد کنیم. فایده ی این کار این است که مرورگر تمام کارکترهای خاص را به عنوان داده های خام تفسیر خواهد کرد. کدینگ در HTML معمولا به این معناست که هرجایی که با کارکترهای زیر روبرو شد آن را با کارکتر نظیرش جایگزین نماید:

From	To
<	&lt;
>	&gt;
(	&#40;
)	&#41;

# #

& &

" "

بنابراین در صورتی که هکر رشته ی "<script>" را وارد نماید و مرورگر بخواهد آن را نمایش دهد، ابتدا آن را کدینگ می کند و سپس به مرورگر ارسال می کند. یعنی:

"<script>"

تاریخ ساخت: April 10, 2014 یا ۲۱ فروردین ۱۳۹۳

تاریخ تحقیق: August 20, 2014 یا ۲۹ مرداد ۱۳۹۳

لینک مطلب: https://www.owasp.org/index.php/Reviewing_Code_for_Cross-site_scripting

/* تصحیح این مقاله، چه در ترجمه و چه در مباحث علمی ، توسط شما دوستان باعث خوشحالی خواهد بود. لطفا آن را با tamadonEH@gmail.com مطرح نمایید.*/

برای مشاهده لیست مقالات کار شده توسط گروه ما به لینک زیر مراجعه فرمایید

https://github.com/tamadonEH/list/blob/master/list.md