

1) Get all of the items listed at once via SQL injection:

```
$v_cat = $_GET['category'];  
$sqlstr = "SELECT * FROM product WHERE Category='" . $v_cat . "'";  
$rs = pg_query($sqlstr);  
  
category = food' or 1=1--
```

2) Find all inputs to drop users from the database (yes, there are many other things wrong with this):

```
$user = $_GET['user'];  
$password = $_GET['password'];  
$prep = 'SELECT secretinfo FROM users WHERE ' .  
        'uid = ' . $user . ' AND password = $1';  
$rs = pg_query_params($prep, $password);  
  
user = 1; DROP users; --
```

3) True or False and why, there is an XSS here:

```
$name = $_GET['name'];  
if (preg_match("<script>", $name)) {  
    die;  
}  
  
echo("<p>Hello, " . $name . "</p>");
```

True. HTML is much much too complex to do a simple regex to properly sanitize it.

4) Assume that an attacker from `evil.com` can control the JavaScript variable `prevSite` when a victim visits this code at `profiles.com`. Find an attack and write down a payload.

```
<html> <p>  
<a id="myanchor">Return to Previous Website</a>  
<script>  
var prevSite = valueFromURL();  
var elt = document.getElementById('myanchor');  
elt.setAttribute('href', prevSite);  
</script>  
</p> </html>  
  
javascript:alert('evil!');
```