TP 1

1. (File: authentication.php) Explain how looking at the http headers sent by this application after a user logs in, an attacker can find out the password used for authentication.

2. (File: formAttack.html) Explain why this web page sent by the site dansie.net is vulnerable to an attack where the client can buy a black leather purse with leather straps by less than 20 dollars. How can such an attack be mounted? Modify the page in such a way that when the user submits the form a service price.php at the server sets a cookie that lasts 7 days with the price of the article. Write a second service confirms.php that confirms the transaction of buying.
   - How an attacker (a client using the application) can now buy articles by a cheaper price than the one set by price.php?
   - If price.php adds a hash to the price in the cookie, how an attacker can buy articles by a cheaper price than the one set by price.php?

3. (File: integrator.html) Look at the code of integrator.html and write code for evilGadget.js in such a way that evilGadget.js will send the secret to an untrusted server. How do you rewrite integrator.html so the same origin policy will protect the secret?

4. (No file) Write two different services from the same server that set a cookie. On the client side include a gadget and try the following things:
   - let the gadget delete the cookie via JavaScript
   - Can the second service delete the cookie of the first? Justify why.
   - let the gadget send the cookie to another server (you can use a different port to simulate this)
   - Does the previous item work if the gadget is inside a frame?
   - and if gadget is inside a script and the cookie is initially set as httponly?
   - and if gadget is in script and the cookie is initially set as secure? Justify all your answers.

5. (File: guestbook.php, guestbookleavemessage.php. You must create file message.txt.) Use the application: to which kind of vulnerability suffers the application? Demonstrate with an attack that disables the input element of the guestbook. Mention 2 ways in which this attack could be prevented and implement them in 2 versions of guestbook: guestbookA.php and guestbookB.php.

6. (File: guestbook2.php, guestbookleavemessage2.php). This application uses a standard php function for sanitization, however the following attacker code can be executed: this is a nice message?);alert(?this is attacker code?);console.log(? Try the attack and explain why it works in spite of sanitization. How do you correct this vulnerability?

7. (File: xsrf.php, simple.php) Explain which code should attackerGadget.js have to produce a CSRF attack and answer:

   - which is the CSRF attack?
   - can the attack take place if the gadget is in an iframe? Justify your answer.
   - can the attack take place if the cookie is httponly? Justify your answer.

   Using tokens, prevent CSRF attacks in this application. Having implemented a defense against CSRF attacks, explain how attackerGadget.js could mount an XSS attack to circunvent the CSRF defense and produce an CSRF attack. After implementing the attack, explain how do you prevent this.

8. (File: api.js, mashup.html) The mashup provides an api for b.js to create new users. The code of b.js is unknown but it should create new users by using the command *new newuser("somename", "someemail")*. Assume that script b.js is correctly verified for the following: its code does not access any property of window except newuser() that is provided as an interface. Answer the following questions and justify:

   a) can b.com get the value of variable secret?

   b) can b.com access the window object?

   Now assume that b.js is in an iframe, and answer the same questions.

9. (File: attacker.js, mashup1.html, code of trusted.js or boot.js is not given) what is the vulnerability to which trusted.js is exposed? Write code for boot.js to prevent the attack.

10. (File: no file) Let adapi.js be the code for some external gadget. Assume that code for adapi.js has been verified and cannot access window directly, however it has access to function integrator whenever it is available. For each of the following versions of the mashup, can the external gadget adapi.js

    - read the value of secret?
    - obtain a pointer to window?

Justify and discuss your answer for each of the proposals. If we remove the assumption that code for adapi.js cannot access window directly, does any of your previous answers change?

Version 1

```
<script>
function integrator(){
  var secret = 42;
  return this;
}
</script>
<script src=http://adserver.com/adapi.js></script>
```

Version 2

```
<script>
function integrator(){
  secret = 42;
  return this;
}
</script>
<script src=http://adserver.com/adapi.js></script>
```

Version 3

```
<script>
(function(){
   secret = 42;
  return this;
})(undefined)
</script>
<script src=http://adserver.com/adapi.js></script>
```

Version 4

```
<script>
integrator = function(){
  var secret = 42;
  return this;
}
</script>
<script src=http://adserver.com/adapi.js></script>
```

11. (File: no file) Consider the following JavaScript code and answer if it is possible for an attacker that can inject any code into function *toto* to learn the secret value

stored in variable secret.

```
var x = 0;
var secret = 42;
functionfoo(z){
  var x = 1;
  var bar = function(){
    var x = 2;
    var toto = function(){
      this.y = z;
      document.writeln(x);
eval(attackercode);     }
    return toto;
  }
  return bar;
}
x = new(new(newfoo(0)));
```

Answer the same question for the following code:

```
var x = 0;
var secret = 42;
functionfoo(z){
  var x = 1;
  var bar = function(){
    var x = 2;
    var toto = function(){
      this.y = z;
      document.writeln(x);
eval(attackercode);     }
    return toto;
  }
  return bar;
}
newfoo(0);
```

12. (File: no file) Assume you have a function lookup that will replace any access to a property of the form $o[prop]$ in attacker code by $lookup(o, prop)$. The goal of lookup is to prevent any access to a special property "$secretproperty$". Which of

the following 2 implementations of lookup satisfy this goal? Justify your answer.

```
lookup1 =
function(o, prop){
  if(prop === "secretproperty"){
    return"unsafe!"; }
  else{
    return o[prop]; }}


lookup2 =
function(o, prop){
  var goodproperty = {"publicproperty" : "publicproperty", "secretproperty" : "publicproperty"}[p
  return o[goodproperty];
  }
```