WAS 2015: TP 4

1. (File: api.js, mashup.html) The mashup provides an api for b.js to creat 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.

2. (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.

3. (File: no file) Let adapi.js be the code for some external gadget. Which of the following integrator codes is more secure in order to protect "secret"? Justify and discuss your answer for each of the proposals.

   Version 1

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

   Version 2

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

Version 3

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

Version 4

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

Version 5

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

Version 6

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

4. (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);
```

5. (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];
  }
```