TP

1. See code for boot.js, trusted.js, mashup1.js. Without changing this code, write code for attacker.js in order to make trusted.js execute unwanted code.
2. Change boot.js (and if need be trusted.js) to avoid the attack

TP

3. 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?

TP

V1

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

TP

V2

< script >

function integrator(){var secret = 42; return this;}

</script >

< script src = http : //adserver:com=adapi.js >

TP

V3

< script >

(function (){secret = 42; return this;})()

</script >

< script src = http : //adserver:com=adapi.js >

TP

4. 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.

TP

V1

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

TP

V2

```
lookup2 =
function(o, prop){
var goodprop = {
‘publicproperty’: ‘publicproperty’,
‘secretproperty’: ‘publicproperty’}[prop];
return o[goodprop]; }
```