**Academic Year:** Winter Semester 2021/22

# 6090: Security of Computer and Embedded Systems

Problem Sheet 5

***Secure Programming***

In this homework, we will deepen our knowledge in software/application security in general and in particular the knowledge on how software vulnerabilities are classified, analyzed, and fixed.

## 1. Injection Attacks

In the following exercises, we will learn how/ to exploit and prevent different injection attacks.

### Exercise 1: *XSS*

Suppose you know that a particular web site offers a bulletin board feature (i.e., users can post comments that can be read by other users). How could you steal the session cookie of another user?

### Exercise 2: *SQL Injection*

Suppose you know that a particular web site uses a backend database to implement authentication. Given a login page with username and password fields, what would you type into these fields to try to perform SQL injection to bypass proper authentication? Briefly explain why your approach would work.

## Exercise 3: *Analyzing Source Code*

1. What vulnerability exists in the following code?

```
def readFile(request):
  result=HttpResponse()
  file = request.GET.get(’currentFile’)
  f = open(file, ’w’)
  for index,line in enumerate(f):
    result.write(str(index)+"␣"+line+"")
  return result
```

2. What vulnerability exists in the following code?

```
function processUrl()
{
    var pos = url.indexOf("?");
    url = pos != -1 ? url.substr(pos + 1) : "";
    if (!parent._ie_firstload) {
        parent.BrowserHistory.setBrowserURL(url);
        try {
            parent.BrowserHistory.browserURLChange(url);
        } catch(e) { }
    } else {
        parent._ie_firstload = false;
    }
}

var url = document.location.href;
processUrl();
document.write(url);
```

## 2. Preventing Attacks

In this section, you can deepen your understanding on attack prevention mechanisms.

### Exercise 4: *Input Sanitization*

The recommendation for preventing injection attacks is to *sanitize* (filter) user input. In general, there are two approaches for filtering input: *Blacklisting* and *whitelisting*.

- Briefly explain the concepts blacklisting and whitelisting.
- Which approach is usually recommended? Briefly explain your answer.

### Exercise 5: *Input Sanitization*

To protect an application against XSS attacks, a programmer writes the following code for removing script-tags.

```
function sanitize_xss(input)
{
   return input.split("<script>").join("").split("<\script>").join("");
}
```

Does this work?

### Exercise 6: *Input Sanitization*

For preventing SQL injections, web application developers implement a *client-side* whitelisting in JavaScript. You can assume that the whitelisting is strict, e.g., only allowing lower-case and upper-case characters to be passed to the backend.

Does this approach prevent SQL injection? Explain briefly your answer.

## References

1. Ross J. Anderson. Security Engineering: A Guide to Building Dependable Distributed Systems. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2001. The complete book is available at: http://www.cl.cam.ac.uk/~rja14/book.html

2. OWASP Top 10 – 2017, 2017.