References

[1] ”The ‘digital world’. What does it mean?”, Teaching in a digital world, 2015. [Online]. Available: https://donnakeenan.wordpress.com/2015/03/24/the-digital-world-what-does-it-mean/ [Accessed: 28- Jan- 2021].

[2] M. Uma and G. Padmavathi, ”A Survey on Various Cyber Attacks and Their Classification”, International Journal of Network Security, Vol.15, No.5, PP.390-396, 2013. [Online]. Available: http://ijns.jalaxy.com.tw/contents/ijns-v15-n5/ijns-2013-v15-n5-p390-396.pdf [Accessed: 28- Jan- 2021]

[3] J. Salter, ”Sourcegraph: Devs are managing 100x more code now than they did in 2010”, Ars Technica, 2021. [Online]. Available: https://arstechnica.com/gadgets/2020/10/sourcegraph-devs-are-managing-100x-more-code-now-than-they-did-in-2010/ [Accessed: 08- Jan- 2021].

[4] ”OWASP Top Ten Web Application Security Risks”, Owasp.org, 2021. [Online]. Available: https://owasp.org/www-project-top-ten/ [Accessed: 10- Jan- 2021].

[5] ”CWE - Common Weakness Enumeration”, Cwe.mitre.org, 2021. [Online]. Available: https://cwe.mitre.org/ [Accessed: 10- Jan- 2021].

[6] N. Alexopoulos, S. Mahbub Habib, S. Schulz and M. Mühlhäuser, ”The Tip of the Iceberg: On the Merits of Finding Security Bugs”, ACM Transactions on Privacy and Security: Vol 24, No 1, 2020. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3406112 [Accessed: 05- Feb- 2021].

[7] Synopsys Team, ”What is the secure software development life cycle (SDLC)?”, Software Integrity Blog, 2020. [Online]. Available: https://www.synopsys.com/blogs/software-security/secure-sdlc/ [Accessed: 12- Feb- 2021].

[8] A. E. Hassan, S. McIntosh, Y. Kamei and B. Adams, ”The impact of code review coverage and code review participation on software quality: a case study of the Qt, VTK, and ITK projects”, Proceedings of the 11th Working Conference on Mining Software Repositories, 2014. [Online]. Available: https://dl.acm.org/doi/10.1145/2597073.2597076 [Accessed: 15- Feb- 2021].

[9] ”Sample Secure Code Review Report”, The MITRE Corporation, 2014. [Online]. Available: https://www.mitre.org/publications/all/sample-secure-code-review-report [Accessed: 22- Feb- 2021].

[10] A. Bacchelli, C. Bird, “Expectations, outcomes, and challenges of modern code review”, Proc. Int. Conf. on Soft. Eng.p, ICSE ’13, pages 712–721, 2013. [Online]. Available: https://sback.it/publications/icse2013.pdf [Accessed: 22- Feb- 2021].

[11] J. Czerwonka, M. Greiler and J. Tilford, ”Code reviews do not find bugs: how the current code review best practice slows us downs”, Proc. Int. Conf. on Soft. Eng.p, ICSE ’15, Volume 2, pages 28–29, 2015. [Online]. Available: https://dl.acm.org/doi/10.5555/2819009.2819015 [Accessed: 24- Feb- 2021].

[12] A. Buttner, R. Piazza, A. Summers and R. Purohit, ”A Secure Code Review Retrospective”, IEEE Secure Development (SecDev), pages 31-21, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9230284 [Accessed: 24- Feb- 2021].

[13] A. Buttner, ”Using Automated Static Analysis Tools for Code Reviews”, The MITRE Corporation, 2013. [Online]. Available: https://www.mitre.org/capabilities/cybersecurity/overview/cybersecurit

y-blog/using-automated-static-analysis-tools-for. [Accessed: 25- Feb- 2021]

[14] A. Buttner, ”The Importance of Manual Secure Code Review”, The MITRE Corporation, 2014. [Online]. Available: https://www.mitre.org/capabilities/cybersecurity/overview/cybersecurity-blog/the-importance-of-manual-secure-code-review [Accessed: 25- Feb- 2021].

[15] Microsoft Security Team, ”Microsoft open sources CodeQL queries used to hunt for Solorigate activity”, Microsoft Security, 2021. [Online]. Available: https://www.microsoft.com/security/blog/2021/02/25/microsoft-open-sources-codeql-queries-used-to-hunt-for-solorigate-activity/. [Accessed: 04- Mar- 2021].

[16] E. Woollacott, ”Semgrep: Static code analysis tool helps ‘eliminate entire classes of vulnerabilities’”, The Daily Swig Cybersecurity news and views, 2020. [Online]. Available: https://portswigger.net/daily-swig/semgrep-static-code-analysis-tool-helps-eliminate-entire-classes-of-vulnerabilities [Accessed: 04- Mar- 2021].

[17] ”Semgrep A Practical Introduction”, NotSoSecure part of claranet cyber security, 2020. [Online]. Available: https://notsosecure.com/semgrep-a-practical-introduction/ [Accessed: 04- Mar- 2021].

[18] ”semgrep”, PyPI, 2020. [Online]. Available: https://pypi.org/project/semgrep/0.15.0b1/ [Accessed: 05- Mar- 2021].

[19] M. Goodrich and R. Tamassia, Introduction to Computer Security: Pearson New International Edition. Harlow: Pearson Education Limited, chapter 1 page 2, 2014. [20] G. Mcgraw, ”Automated Code Review Tools for Security”, IEEE Computer, Volume: 41, Issue: 12, pages 108-111, 2008. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4712512. [Accessed: 08- Mar- 2021].

[21] ”About CodeQL — CodeQL”, Codeql.github.com, 2021. [Online]. Avail-

able: https://codeql.github.com/docs/codeql-overview/about-codeql/. [Accessed: 08- Mar- 2021].

[22] ”QL language reference — CodeQL”, Codeql.github.com, 2021. [Online]. Available: https://codeql.github.com/docs/ql-language-reference/. [Accessed: 09- Mar- 2021].

[23] ”Home - Semgrep Docs”, Semgrep.dev, 2021. [Online]. Available: https://semgrep.dev/docs. [Accessed: 10- Mar- 2021].

[24] R. Kent, ”Introduction to variant analysis with QL and LGTM (part 2)”, Blog.semmle.com, 2019. [Online]. Available: https://blog.semmle.com/introduction-to-variant-analysis-part-2/ [Accessed: 12- Mar- 2021].

[25] V. Gite, ”How To Use grep Command In Linux / UNIX”, Cybercity.biz, 2021. [Online]. Available: https://www.cyberciti.biz/faq/howto-use-grep-command-in-linux-unix/ [Accessed: 12- Mar- 2021].

[26] J. Jones, “Abstract Syntax Tree Implementation Idioms,” Pattern Languages of Program Design, 2003. [Online]. Available: https://hillside.net/plop/plop2003/Papers/Jones-ImplementingASTs.pdf. [Accessed: 12- Mar- 2021].

[27] MDN contributors, ”JavaScript”, Developer.mozilla.org, 2020. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/JavaScript. [Accessed: 21- Mar- 2021].

[28] MDN contributors, ”A re-introduction to JavaScript”, Developer.mozilla.org, 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript. [Accessed: 21- Mar- 2021].

[29] D. Baca, B. Carlsson and L. Lundberg, ”Evaluating the cost reduction of static code analysis for software security”, Proceedings of the third ACM SIGPLAN workshop on Programming languages and analysis for security, pages 79-88, 2008.

[Online]. Available: https://dl.acm.org/doi/pdf/10.1145/1375696.1375707 [Accessed: 01- Apr- 2021].

[30] ”Quantitative Data Collection Methods”, Research-Methodology. [Online]. Available: https://research-methodology.net/research-methods/quantitative-research/ [Accessed: 01- Apr- 2021].

[31] ”Qualitative Data Collection Methods”, Research-Methodology. [Online]. Available: https://research-methodology.net/research-methods/qualitative-research/ [Accessed: 02- Apr- 2021].

[32] A. Håkansson, ‘Portal of Research Methods and Methodologies for Research Projects and Degree Projects’, Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering FECS’13, pp. 67–73, 2013. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:677684/FULLTEXT02.pdf [Accessed: 02- Apr- 2021].

[33] ”Inductive Approach (Inductive Reasoning)”, Research-Methodology. [Online]. Available: https://research-methodology.net/research-methodology/research-approach/inductive-approach-2/ [Accessed: 06- Apr- 2021].

[34] ”Deductive Approach (Deductive Reasoning)”, Research-Methodology. [Online]. Available: https://research-methodology.net/research-methodology/research-approach/inductive-approach-2/ [Accessed: 06- Apr- 2021].

[35] ”Getting started with the CodeQL CLI”, Codeql.github.com, 2021. [Online]. Available: https://codeql.github.com/docs/codeql-cli/getting-started-with-the-codeql-cli/#getting-started-with-the-codeql-cli [Accessed: 22- Apr- 2021].

[36] ”github/codeql”, GitHub. [Online]. Available: https://github.com/github/codeql/tree/main/javascript/ql/src/Security [Accessed: 20- Apr- 2021].

[37] ”appsecco/dvna”, GitHub. [Online]. Available: https://github.com/app

secco/dvna [Accessed: 06- Apr- 2021].

[38] ”OWASP/NodeGoat”, GitHub. [Online]. Available: https://github.com/OWASP/NodeGoat [Accessed: 10- Apr- 2021].

[39] N. Golafshani, ”Understanding reliability and validity in qualitative research”, The Qualitative Report, 8(4): pages 597–606, 2003. [Online]. Available: http://www.nova.edu/ssss/QR/QR8-4/golafshani.pdf [Accessed: 20-May-2021]

[40] ”Abstract syntax tree - Wikipedia”, En.wikipedia.org. [Online]. Available: https://en.wikipedia.org/wiki/Abstract_syntax_tree#/media/File:Abstract_syntax_tree_for_Euclidean_algorithm.svg [Accessed: 20- Feb- 2021].

[41] ”Introduction to variant analysis with QL and LGTM (part 2)”, Blog.semmle.com. [Online]. Available: https://blog.semmle.com/introduction-to-variant-analysis-part-2/ [Accessed: 20- Feb- 2021].

Appendix A

Questionnaire

The purpose of this questionnaire is to collect data. Your participation in this study is completely voluntary and your answers will be anonymous.
Please do not use any automated code review tools or Google, in order to help us with an unbiased research.

Top 10 Web Application Security Risks according to OWASP

The following are the security vulnerability risks that may or may not be present in this questionnaire:

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring

INSTRUCTIONS

For every question, starting from question 3, make sure to write down the name of the vulnerability risk and state where in the code it occurs. Write the row number or the range.

An answer can look like this: (Insufficient Logging & Monitoring, row 105-107).

If you think that the code is OK and there are no security vulnerabilities, write "none" as an answer.

PS! This applies to all the following questions.

1. Please choose one of the following:

- [ ] Student
- [ ] Developer
- [ ] Security professional

2. Which organization do you belong to? (optional)

3. Question 1

```
const http = require('http');
const app = require('./app');
var fs = require('fs'),
    url = require('url');

var server = http.createServer(function(req, res) {
  let path = url.parse(req.url, true).query.path;
  res.write(fs.readFileSync(path));
  res.write(fs.readFileSync("/home/user/" + path));
});
```

4. Question 2

```
22
23  var express = require('express');
24  var app = express();
25  var cryptoRandomString = require('crypto-random-string');
26  app.use(session({
27    secret: cryptoRandomString(30),
28    resave: true,
29    saveUninitialized: true,
30    cookie: { secure: false }
31  }))
32
```

5. Question 3

```
12  const app = require("express");
13  const router = app.Router();
14  const nodemailer = require('nodemailer');
15  const fs = require('fs');
16  let config = JSON.parse(fs.readFileSync('config.json', 'utf8'));
17  router.post('/resetpass', (req, res) => {
18    let email = req.query.email;
19    let transport = nodemailer.createTransport(config.smtp);
20    let token = backend.getUserSecretResetToken(email);
21    transport.sendMail({
22      from: 'webmaster@example.com',
23      to: email,
24      subject: 'Forgot password',
25      text: `Click to reset password: https://${req.host}/resettoken/${token}`,
26    });
27  });
```

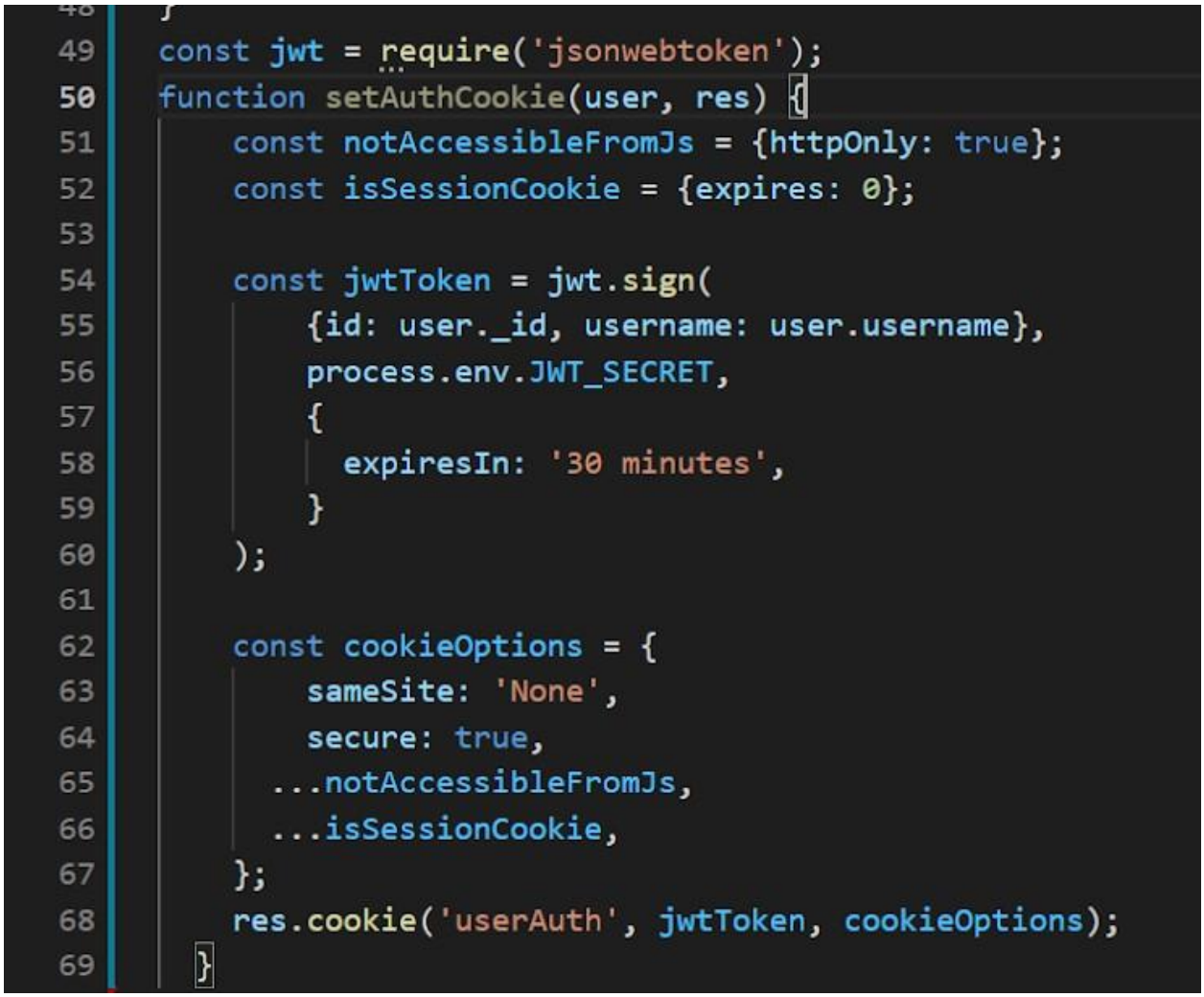
6. Question 4

```
var db = require('../models')
module.exports.userSearch = function (req, res) {
    var query = "SELECT name,id FROM Users WHERE login='" + req.body.login + "'";
    db.sequelize.query(query, {
        model: db.User
    }).then(user => {
        if (user.length) {
            var output = {
                user: {
                    name: user[0].name,
                    id: user[0].id
                }
            }
            res.render('app/usersearch', {
                output: output
            })
        } else {
            req.flash('warning', 'User not found')
            res.render('app/usersearch', {
                output: null
            })
        }
    }).catch(err => {
        req.flash('danger', 'Internal Error')
        res.render('app/usersearch', {
            output: null
        })
    })
}
```

7. Question 5

```
60  const app = require("express");
61  const router = app.Router();
62  router.get('/user/:id', function(req, res) {
63      if (!isValidUserId(req.params.id))
64        res.send("Unknown user: " + req.params.id);
65      else
66        // TODO: do something exciting
67        ;
68    });
```

8. Question 6



```
49  const jwt = require('jsonwebtoken');
50  function setAuthCookie(user, res) {
51      const notAccessibleFromJs = {httpOnly: true};
52      const isSessionCookie = {expires: 0};
53
54      const jwtToken = jwt.sign(
55          {id: user._id, username: user.username},
56          process.env.JWT_SECRET,
57          {
58            expiresIn: '30 minutes',
59          }
60      );
61
62      const cookieOptions = {
63          sameSite: 'None',
64          secure: true,
65        ...notAccessibleFromJs,
66        ...isSessionCookie,
67      };
68      res.cookie('userAuth', jwtToken, cookieOptions);
69    }
```

9. Question 7

```
const http = require('http');
const app = require('./app');
http.createServer(function onRequest(req, res) {
    var body;
    try {
      body = handleRequest(req);
    }
    catch (err) {
      res.statusCode = 500;
      res.setHeader("Content-Type", "text/plain");
      res.end(err.stack);
      return;
    }
    res.statusCode = 200;
    res.setHeader("Content-Type", "application/json");
    res.setHeader("Content-Length", body.length);
    res.end(body);
  }).listen(3000);

const port = process.env.PORT || 5000;
server = http.createServer(app);


server.listen(port, ()=>console.log(`listening on port ${port}`));
```

10. Question 8

```
const app = require("express");
const router = app.Router();
router.get('/list-directory', function(req, res) {
    fs.readdir('/public', function (error, fileNames) {
        var list = '<ul>';
        fileNames.forEach(fileName => {
            list += '<li>' + fileName + '</li>';
        });
        list += '</ul>'
        res.send(list);
    });
});
```

11. Question 9

```
const exec = require('child_process').exec;
module.exports.ping = function (req, res) {
    exec('ping -c 2 ' + req.body.address, function (err, stdout, stderr) {
        output = stdout + stderr
        res.render('app/ping', {
            output: output
        })
    })
}
```

12. Question 10

```
2   const app = require("express");
3   const router = app.Router();
4   const jsyaml = require("js-yaml");
5
6   router.get("load", function(req, res) {
7     let data = jsyaml.load(req.params.data);
8   });
```

13. Question 11

```
27  router.get('/remember-password', function (req, res) {
28      let pw = req.param("current_password");
29      res.cookie("password", pw);
30  });
```

14. Question 12

```
54  const libxml = require("libxmljs");
55  const app = require("express");
56  const router = app.Router();
57  router.post("upload", (req, res) => {
58    let xmlSrc = req.body,
59      doc = libxml.parseXml(xmlSrc, { noent: true });
60  });
```

15. Question 13

```
233
234 var libxmljs = require("libxmljs");
235 module.exports.bulkProducts =  function(req, res) {
236     if (req.files.products && req.files.products.mimetype=='text/xml'){
237         var products = libxmljs.parseXmlString(req.files.products.data.toString('utf8'), {noent:false,noblanks:true})
238         products.root().childNodes().forEach( product => {
239             var newProduct = new db.Product()
240             newProduct.name = product.childNodes()[0].text()
241             newProduct.code = product.childNodes()[1].text()
242             newProduct.tags = product.childNodes()[2].text()
243             newProduct.description = product.childNodes()[3].text()
244             newProduct.save()
245         })
246         res.redirect('/app/products')
247     }else{
248         res.render('app/bulkproducts',{messages:{danger:'Invalid file'},legacy:false})
249     }
250 }
251
```

16. Question 14

```
const webpack = require("webpack");
module.exports = [{
    plugins: [
        new webpack.DefinePlugin({
            "process.env": JSON.stringify(process.env)
        })
    ]
}];
```

17. Question 15

```
const app = require("express");
const router = app.Router();
expat = require("node-expat");
router.post("upload", (req, res) => {
  let xmlSrc = req.body,
    parser = new expat.Parser();
  parser.on("startElement", handleStart);
  parser.on("text", handleText);
  parser.write(xmlSrc);
});
```

18. What are your thoughts about these questions? Which bug was easiest/most difficult to detect?

Appendix B

Comments about the questionnaire

Volunteer Number	Occupation	Thoughts about the questionnaire and which bug was easiest/most difficult to detect?
1	Security Professional	Easiest: 4. Hardest: 6, 7. I've never used Node.JS so that makes it more difficult.
2	Security Professional	Question 9 was easiest, and question 2, 6, 13 and 14 was most difficult.
3	Security Professional	Don't have deep knowledge about node, not using google to lookup the modules/methods means I don't know the details of what they do...
4	Security Professional	—
5	Security Professional	Not using google to lookup the modules/methods means I don't know the details of what they actually do
6	Developer	sql injection was the easiest. in my opinion this might be because I feel sql injections is the most known vulnerability. I dont really know much about "insufficient loggnig and monitoring" as none of these examples include neither logging nor monitoring so I guess they all have that problem :D
7	Developer	If using new libraries or parameters, I always do research (googling) about how to use them. A large portion of the time developing is information gathering. Therefor it can be hard to answer questions about use cases not seen before. And I must admit that I did some searches during this questionnaire too.
8	Security Professionals	Most of the bugs are situational and all depends on who can access the parameters. Also, not having the full code and not being able to look up suspected code/modules limits my accuracy
9	Student	I think it would have been helpful with a short text explaining what the intended behavior of the code is. For example, the first code segment seems to allow anyone to read any file on the host using both row 27 and 28. But I can only assume that it was a vulnerability and not it's intended function. All bugs were quite difficult to detect, but I'll add that I am not very fluent in Javascript.
10	Security Professional	Anything relating to nodejs-specific libraries is difficult for me as it's not my area
11	Security professional	Some bugs are too obvious to spot because the code snippets are very small. Bugs get harder to detect later on. SQLi was the easiest to spot (Q4).
12	Student	It was pretty hard to find the vulnerabilities, a lot of the code was hard to understand when you don't know the specific libraries.
13	Student	The code was hard to understand when the whole picture was not available which led to harder understanding and detecting of the bugs.
14	Student	Difficult questions, 4 was easiest.
15	Student	As a student with limited experience in the web security, most bugs were difficult to detect. The easiest bugs to detect were those that exposed sensitive data such as questions 11. My general thoughts on the questions are that they are well thought-out, i.e the bugs weren't super obvious and also varied in security risks.
16	Student	A bit difficult
17	Developer	a little bit difficult.
18	Developer	Had some difficulties as I don't actually know what some of these libraries do.
19	Developer	Difficult since I don't know what the code does.
20	Developer	question 4 was the easiest, but difficult to know the functionality of the code.