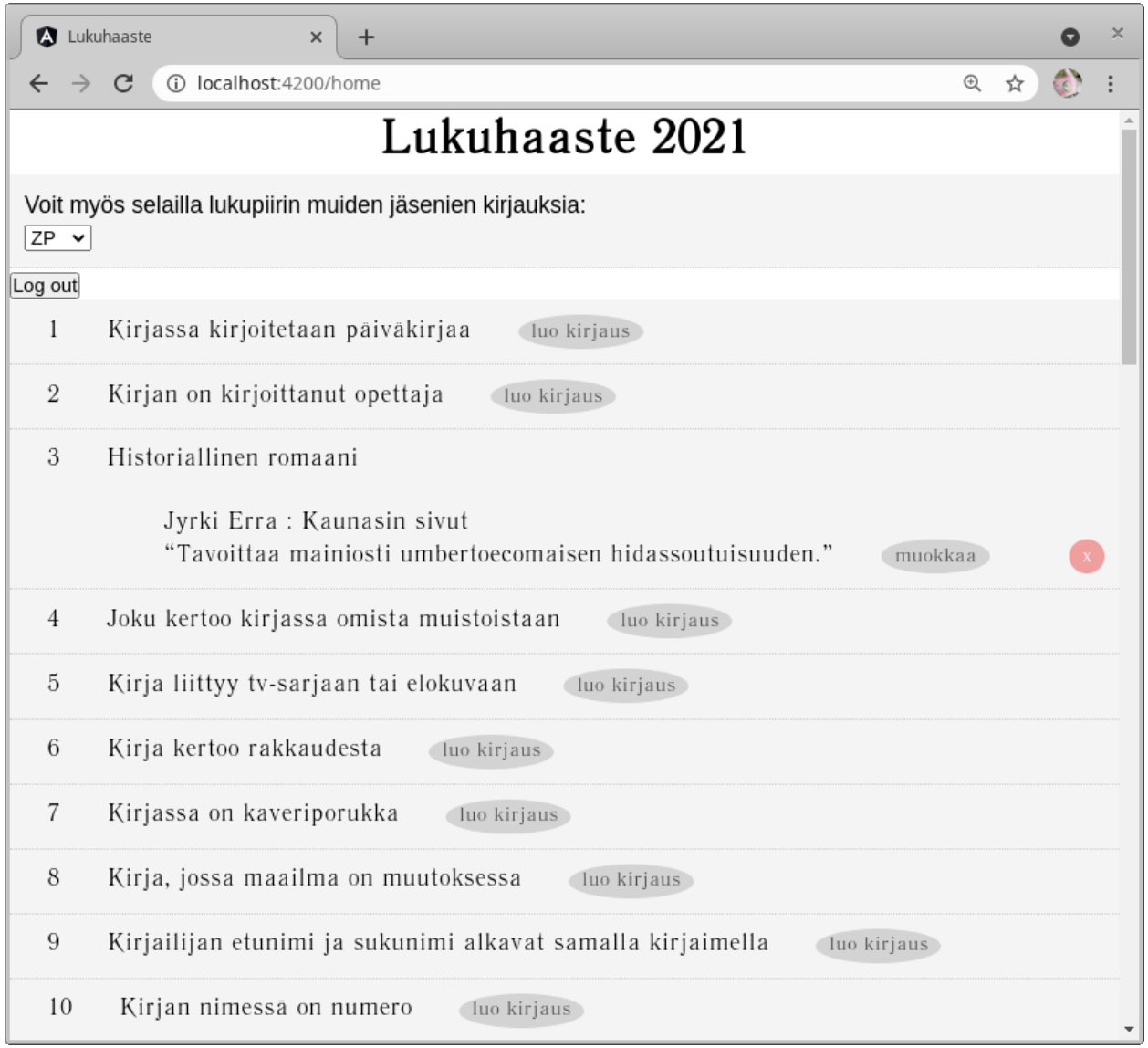
Improving the security of "Lukuhaaste" web application

JP Paalassalo 21.5.2021

Scope

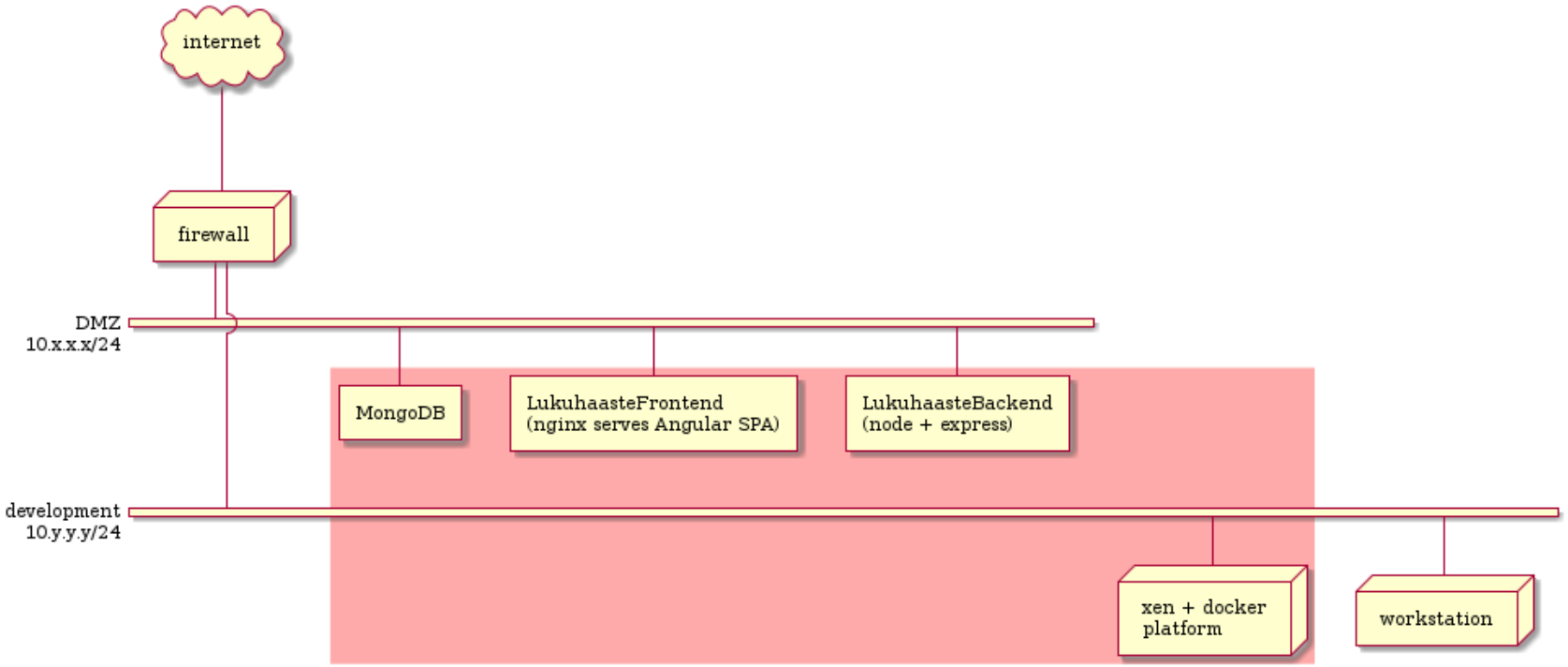
- Existing single-user MEAN stack web application:
  - Perform threat analysis
  - New feature: implement user login and access tokens

App Functionality



- App presents list of reading challenges
- User can post achievements including freetext comments

App Deployment

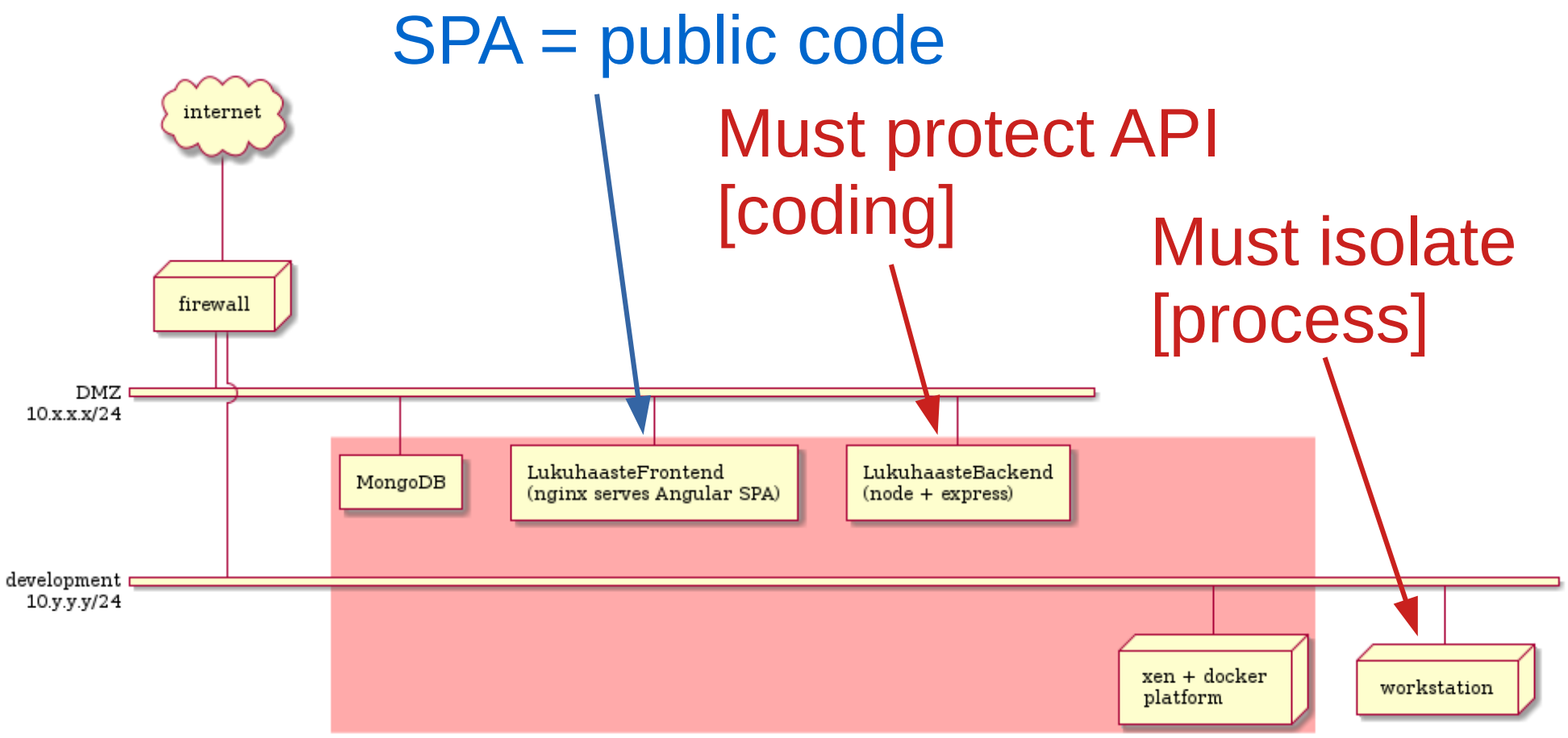


- Attack surface contains not only app interfaces, but sysadmin/developer layer too

Detailed Threat Analysis

Scope	Vulnerability	Risk	Mitigation
firewall	configuration error	2	Utilize external testing services
firewall	unauthorized access	2	Keep FW updated. Strong admin password. Restrict access from internal network too.
frontend	nginx vulnerabilities	3	Set up watchtower to rebuild frontend as nginx upgrades are available.
docker platform	docker API vulnerabilities exposed to containers	1	Keep upgrading platform. All public containers must run with limited non-root privileges.
xen platform	virtualisation network stack has vulnerability	1	Keep xen up-to-date.
backend/frontend node platform	node library vulnerabilities	2	Check and fix vulnerabilities using snyk.
development	Development workstation gets infected and provides admin access to system	5	Isolate development environment to a separate workstation or on Qubes OS.
frontend code	Frontend provides unauthorised access to backend	1	(Frontend code is public and fully modifiable by attackers; not much can be done) Implement user authentication and authorization. Sanitize data sent to backend.
frontend code	Frontend identity is compromised enabling MITM attack and possibly stealing user credentials	4	Use proper certificates and force using HTTPS, use external authentication.
frontend code	Lost user credentials allow stealing the account	2	Enable multi-factor authentication and anomaly detection
backend code	unauthorised access	5	Implement user authentication and jwt sessions
backend code	injection attacks	4	Sanitize all incoming data
backend code	access tokens are used from malicious context	4	Implement CORS. Set SameSite cookie attribute to Strict.
backend code	unnecessary server information is leaked	2	Set up helmet to minimise header data

Key Attack Vectors



Implementing authentication

- Decided to use external authentication service
  - Separates authentication security issues from app
  - Provides MFA if needed
- Selected auth0
  - Supports google/github etc credentials
  - Supports standard authentication protocols: OAuth2, JWT
- In SPA context use "Authorization Code Flow with PKCE" (proof Key for Code Exchange).

Backend user auth: /api/users

```
const jwtAuthz = require('express-jwt-authz');
// require jwt token with authorization to read user data
const checkScopes = jwtAuthz([ 'read:users' ]);

// get all users
// Todo: User should have access only to users that are members of same reading circle
//       Current implementation has one global reading circle
router.get('/', checkJwt, checkScopes, (req, res) => {
    User.find({}).exec(function(err, docs) {
        if (!err) {
            res.send(docs);
        }
        else {
            throw err;
        }
    });
});
```

...backend index.js

```
// add routes
const router = express.Router();
app.use('/api/users', require('./routes/api/users'));
app.use('/api/challenges', require('./routes/api/challenges'));
app.use('/api/books', require('./routes/api/books'));

// last function in stack: none of the previous functions has processed the request
app.use(function (req, res) {
  res.status(404);
  res.json({"message" : "Requested route does not exist" });
} );

//global error handler; if any of the functions in middleware stack throws exception it ends up here
app.use(function (err, req, res, next) {
  console.log("Error message" + err.name + ": " + err.message);
  if (err.name === 'UnauthorizedError') {
    res.status(401);
    res.json({"message" : err.name + ": " + err.message});
  } else {
    res.status(500);
    res.json({"message" : err.name + ": " + err.message});
  }
});
```

Testing

- Backend: nodejs library vulnerabilities were checked with snyk
  - Will send alert emails when new vulnerabilities are found
- App testing with OWASP ZAP
  - Is a proxy server between browser and app

Dev environment isolation

- Tested Qubes OS: enables running isolated VM’s on xen with configurable firewalls for each VM
  - Problem: not enough memory on test laptop to run Windows VM’s in order to use Teams efficiently…
  - Will use this with next laptop
- Set up isolated VM for development, no browsing or unnecessary downloads from that