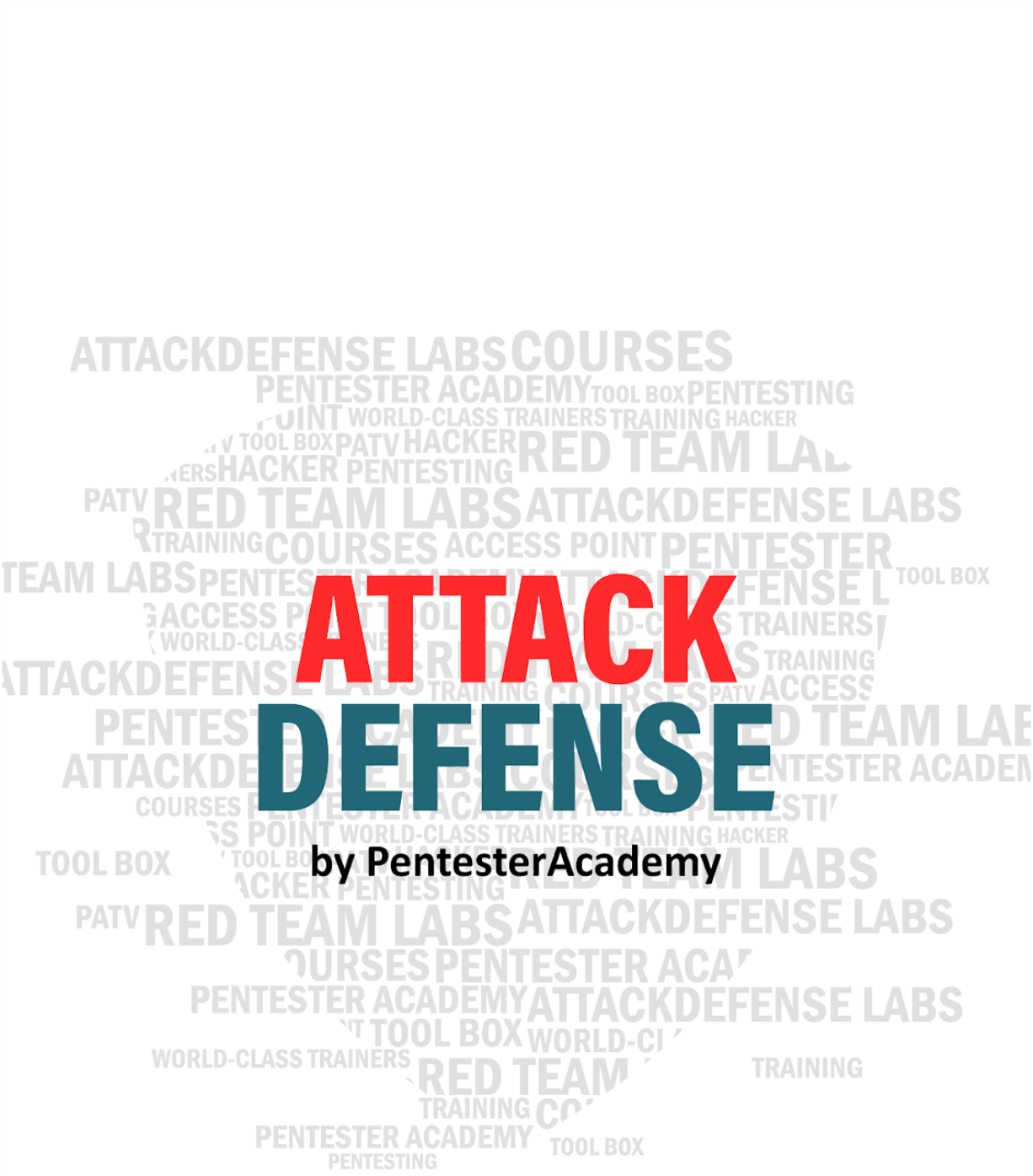
ATTACK
DEFENSE
by PentesterAcademy

Name	NodeJs Scan: Finding Bugs in NodeJs Code
URL	https://www.attackdefense.com/challengedetails?cid=2156
Type	DevSecOps Basics: Static Application Security Testing

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Challenge Description

The NodeJs Scan tool is used to find vulnerabilities on Node applications.

A Kali CLI machine (kali-cli) is provided to the user with NodeJs Scan (njsscan) installed on it. The source code for three sample applications is provided in the home directory of the root user.

Objective: Use the njsscan utility to find vulnerabilities in the applications!

Instructions:

- The source code of applications is provided at /root/github-repos

Solution

Step 1: Check the provided applications.

Command: ls -l github-repos/

```
root@attackdefense:~# ls -l github-repos/
total 12
drwxrwxr-x 7 root root 4096 Nov 17 04:58 node-app-template
drwxrwxr-x 8 root root 4096 Nov 17 04:59 nodejs-simple-contact-webapp
drwxrwxr-x 6 root root 4096 Nov 17 04:59 pollarboy
root@attackdefense:~#
```

We will take one example at a time and run the tool on that.

Example 1: node-app-template

Step 1: Navigate to the node-app-template directory.

Commands:
cd ~/github-repos/node-app-template
ls

```
root@attackdefense:~# cd ~/github-repos/node-app-template
root@attackdefense:~/github-repos/node-app-template# ls
app.js  bin  config.js  LICENSE.md  package.json  public  README.md  routes  views
root@attackdefense:~/github-repos/node-app-template#
```

Step 2: Run the njsscan tool in order to identify vulnerabilities in the application.

Command: njsscan .

```
root@attackdefense:~/github-repos/node-app-template# njsscan .
- Pattern Match ████████████████████████████████████████ 38
- Semantic Grep ██████ 268

================================================================================================
RULE ID: ejs_ect_template
DESCRIPTION: The EJS/ECT template has an unescaped variable. Untrusted user input passed to this variable results in Cross S
ite Scripting (XSS).
TYPE: Regex
PATTERN: <%-(?![ ]*include\().*%>
SEVERITY: ERROR
INPUTCASE: exact
CWE: CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
OWASP: A1: Injection
================================================================================================
```

```
File: views/layout.ejs
Match Position: 238 - 249
Line Number(s): 16
Match String: <%- body %>
root@attackdefense:~/github-repos/node-app-template#
```

The Njsscan has identified that on line 16 of the layout.ejs is vulnerable to Cross-site scripting.

Issues Detected:

- Cross-Site Scripting (CWE-79)

Example 2: nodejs-simple-contact-webapp

Step 1: Navigate to the nodejs-simple-contact-webapp directory.

Commands:
cd ~/github-repos/nodejs-simple-contact-webapp
ls

```
root@attackdefense:~/github-repos/node-app-template#
root@attackdefense:~/github-repos/node-app-template#
root@attackdefense:~/github-repos/node-app-template# cd ~/github-repos/nodejs-simple-contact-webapp
root@attackdefense:~/github-repos/nodejs-simple-contact-webapp# ls
api  app.js  config  LICENSE  package.json  public  README.md  test  views
root@attackdefense:~/github-repos/nodejs-simple-contact-webapp#
```

Step 2: Run the Njsscan tool in order to identify vulnerabilities in the application.

Command: njsscan .

```
root@attackdefense:~/github-repos/nodejs-simple-contact-webapp# njsscan .
- Pattern Match ████████████████████████████████████████████████ 46
- Semantic Grep ▌ 187

================================================================================================================
RULE ID: node_insecure_random_generator
OWASP: A9: Using Components with Known Vulnerabilities
CWE: CWE-327: Use of a Broken or Risky Cryptographic Algorithm
DESCRIPTION: crypto.pseudoRandomBytes()/Math.random() is a cryptographically weak random number generator.
SEVERITY: WARNING
================================================================================================================
```

```
_________________FILES_____________________


File: test/api/controllers/product.js
Match Position: 14 - 27
Line Number(s): 9
Match String: var random = Math.random();
```

```
================================================================================================================
RULE ID: node_nosqli_injection
OWASP: A1: Injection
CWE: CWE-943: Improper Neutralization of Special Elements in Data Query Logic
DESCRIPTION: Untrusted user input in findOne() function can result in NoSQL Injection.
SEVERITY: ERROR
================================================================================================================

_________________FILES_____________________


File: api/controllers/auth.js
Match Position: 28 - 23
Line Number(s): 19: 22
Match String:                       return UserModel.findOne({
                        'name': req.body.username,
                        'password': req.body.password
                    })
```

```
================================================================================================================
RULE ID: node_username
OWASP: A3: Sensitive Data Exposure
CWE: CWE-798: Use of Hard-coded Credentials
DESCRIPTION: A hardcoded username in plain text is identified. Store it properly in an environment variable.
SEVERITY: ERROR
================================================================================================================

_________________FILES_____________________


File: public/script/UserController.js
Match Position: 11 - 32
Line Number(s): 113
Match String:           user.avatar = 'svg-1';


File: public/script/UserController.js
Match Position: 17 - 38
Line Number(s): 224
Match String:                 user.avatar = 'svg-1';
root@attackdefense:~/github-repos/nodejs-simple-contact-webapp#
```

The Njsscan has identified 2 issues in the application, SQL Injection in 'auth.js' at findOne() function and sensitive data exposure (Hardcoded credentials) which is a false positive in this case.

Issues Detected

- NoSQL Injection (CWE-943)
- Sensitive Data Exposure (CWE-798)

Example 3: pollarboy

Step 1: Navigate to the pollarboy directory.

Commands:
cd ~/github-repos/pollarboy
ls

```
root@attackdefense:~/github-repos/nodejs-simple-contact-webapp# cd ~/github-repos/pollarboy
root@attackdefense:~/github-repos/pollarboy# ls
db.js    package.json       public     routes     shrinkwrap.yaml  yarn.lock
LICENSE  package-lock.json  README.md  server.js  views
root@attackdefense:~/github-repos/pollarboy#
root@attackdefense:~/github-repos/pollarboy#
```

Step 2: Run the Njsscan tool in order to identify vulnerabilities in the application and print the output in JSON format.

Command: njsscan . --json

```
root@attackdefense:~/github-repos/pollarboy# njsscan . --json
{
  "errors": [
    {
      "code": 3,
      "help": "If the code appears to be valid, this may be a semgrep bug.",
      "level": "warn",
      "long_msg": "Could not parse LICENSE as javascript",
      "short_msg": "parse error",
      "spans": [
```

```
        {
          "context_end": null,
          "context_start": null,
          "end": {
            "col": 4,
            "line": 1
          },
          "file": "LICENSE",
          "source_hash": "8ca142cc05d84c4a98b98bb8a7a270137c0cb2107f6019fb63ab852c80ad9232",
          "start": {
            "col": 1,
            "line": 1
          }
        }
      ],
```

```
  "nodejs": {
    "express_open_redirect": {
      "files": [
        {
          "file_path": "routes/poll.js",
          "match_lines": [
            74,
            108
          ],
          "match_position": [
            5,
            6
          ],
          "match_string": "    let pollid = req.params.slug;\n    let itemid = req.params.item;\n    if (pollid && itemid) {\n        if (db.get(\"polls\").find({\n                slug: pollid\n            }).get(\"options\").find({\n  slug: itemid\n            }).value() != undefined) {\n\n            //Checks if the person can vote again or not\n    // if( MultipleVotes(req,res,pollid,itemid)){\n          if(canVote(req , pollid)){\n            db.get(\"polls\").find({\n                slug: pollid\n            }).get(\"options\").find({\n                slug: itemid\n            }).update('count', n => n + 1).write();\n          }else{\n            res.redirect(\"/sorry\");\n          }\n            \n    // }\n        } else {\n            res.redirect(\"/err\");\n        }\n\n        // console.log(poll);\n        // console.log(\"UP2\");\n        res.cookie(pollid, 1, { expires: new Date(Date.now() + 30 * 24 * 60 * 60 * 1000), httpOnly: false }).redirect(\"/thankyou/?id=\" + pollid);\n\n\n\n    } else {\n        res.redirect(\"/\");\n    }"
        }
```

```
      ],
      "metadata": {
        "cwe": "CWE-601: URL Redirection to Untrusted Site ('Open Redirect')",
        "description": "Untrusted user input in redirect() can result in Open Redirect vulnerability.",
        "owasp": "A1: Injection",
        "severity": "ERROR"
      }
    },
    "node_insecure_random_generator": {
      "files": [
        {
          "file_path": "routes/pollmaker.js",
          "match_lines": [
```

```
          "match_position": [
            57,
            70
          ],
          "match_string": "    return Math.random().toString(36).substring(2, m) + Math.random().toString(36).substring(2, m
);"
        }
      ],
      "metadata": {
        "cwe": "CWE-327: Use of a Broken or Risky Cryptographic Algorithm",
        "description": "crypto.pseudoRandomBytes()/Math.random() is a cryptographically weak random number generator.",
        "owasp": "A9: Using Components with Known Vulnerabilities",
        "severity": "WARNING"
      }
    }
  },
  "templates": {}
}
root@attackdefense:~/github-repos/pollarboy#
```


Issues Detected

- Open Redirect (CWE-601)
- Broken or Risky Cryptographic Algorithm (CWE-327)

Learnings

Perform Static Code Analysis using the NodeJs Scan tool.

References:

- Node-app-template (https://github.com/lresende/node-app-template.git)
- Pollarboy (https://github.com/bauripalash/pollarboy.git)
- Nodejs-simple-contact-webapp (https://github.com/mazipan/nodejs-simple-contact-webapp.git)