[주식회사 닦다] 웹 구축 대책 보고서

실전 웹서버 해킹과 대응 2조

조

2022. 06. 23

< 목 차 >

I. 개요

1. 대책 개요

❍ 목 적

– 주식회사 닦다에서 운영중인 웹 서비스를 구축하는 과정에서 모의해킹을 통해 웹 서비스 이용에 공격에 노출된 취약점이 있음을 도출하고 행정안전부 홈페이지 SW(웹) 개발보안 가이드와 웹서버 보안 강화 안내서를 참고하여 웹 서비스를 보안하는데 있음.

2. 웹 서비스 구축 환경

❍ 구성도

대상	설명	환경
Apache 2	Web Application Server	Version: 2.4.46 Built: 2022-04-12T19:46:16
Mysql	Web Database	Version: 8.0.25
Nodejs	Web Framework	NPM Version: 6.2.0 Cookie-paser Version: 1.4.6 Doteny Version: 16.0.0 Ejs Version: 3.1.6 Nodemailer Version 6.7.3
express	Web Framework	Version: 4.17.3 express-fileupload Version: 1.3.1 express-session Version: 1.17.2

II. 시큐어 코딩

1. 주식회사 닦다

세부점검항목	1. SQL 인젝션	점검결과	양호

점검 절차

1. 주식회사 닦다 사이트 내 로그인 기능이 SQL 인젝션에 취약함.

```
connection.query(
    "SELECT * FROM users WHERE id = ? AND password = " +
    `'${password}'`,
    [id, password],
    function (error, results, fields) {
        if (results == undefined) {
            res.send(error);
        } else if (results.length > 0) {
            req.session.loggedin = true;
            req.session.name = id;
            res.redirect("/");
        } else {
            res.send(
                "<script>alert('아이디나 비밀번호가 틀렸습니다!!');history.back();</script>"
            );
        }
        res.end();
    }
);
```

2. filterSQL 문자열을 지정하여 전달된 값에 문자열이 존재하면 경고 후 뒤로가게 인젝션을 방어함.

```
const filterSQL = [
    "CREATE",
    "INSERT",
    "UPDATE",
    "DELETE",
    "DROP",
    "ALTER",
    "create",
    "insert",
    "update",
    "delete",
    "drop",
    "alter",
];
```

```
for (var i = 0; i < filterSQL.length; i++) {
    if (password.includes(filterSQL[i])) {
        attack = 1;
    }
    if (id.includes(filterSQL[i])) {
        attack = 1;
    }
}
if (id && password) {

    if (attack == 1) {
        res.send(
            "<script>alert('공격하지마세요!!');history.back();</script>"
        );
    }
    else {
        connection.query(
            "SELECT * FROM users WHERE id = ? AND password = " +
            `'${password}'`,
            [id, password],
            function (error, results, fields) {
                if (results == undefined) {
                    res.send(error);
                } else if (results.length > 0) {
                    req.session.loggedin = true;
                    req.session.name = id;
                    res.redirect("/");
                } else {
                    res.send(
                        "<script>alert('아이디나 비밀번호가 틀렸습니다!!');history.back();</script>"
                    );
                }
                res.end();
            }
        );
    }
```

보안 권장	• 소스 코드에 SQL Query 입력 값을 받는 함수나 코드를 써야 할 경우, 임의의 SQL Query 입력에 대한 검증 로직을 구현하여 검증되지 않는 SQL Query가 인수 값으로 들어올 경우 에러 페이지가 아닌 정상 페이지가 반환되도록 필터링 처리하고 웹 방화벽을 운용할 경우 웹 방화벽에 SQL 인젝션 관련 룰셋을 적용하여 SQL 인젝션 공격 차단함

세부점검항목	2. XSS	점검결과	양호

점검 절차

1. 취약한 코드로 인해 ejs 탬플릿 기능중 <%= %>를 사용하면 렌더링이 되면서 코드가 실행됨 XSS 공격 가능성이 높아 위험함.

```
        <%=results.title%>
    </h2>
    <div class="subContent">
        <div>
            <div style="font-size: 15px; text-align: left; padding-left: 10%;">
                작성자: <%=results.author%> <br>등록일: <%= moment(results.time).format('YYYY년 MM월 DD일 HH시mm분'
            </div>

        </div>
    </div>
</div>
<div class="sub-container">

    <div class="content-wrap" style="margin: 10px;">
        <div style="width: 60%; height: 500px auto; background-color: #ffffff; margin: 0 auto; margin-botto
        border: 1px solid rgb(0, 0, 0);">

            <input type="hidden" name="idx" value="<%=results._id%>" />



            <h3 style="font-size: 20px; text-align: left; padding: 20px;">
                <%=results.content%>
            </h3>
            <hr
                style="border: solid 1px rgb(89, 89, 89); margin:30px;margin-top: 30px; margin-bottom: 15p
            <td><a style="font-size: 20px;" href='/board/download/uploads/images/<%= results.filename%>'>
                    <%= results.filename%>
                </a>
```

2. 입력하는 results 값을 <%- %> 형식으로 사용함으로써 XSS 방어함.

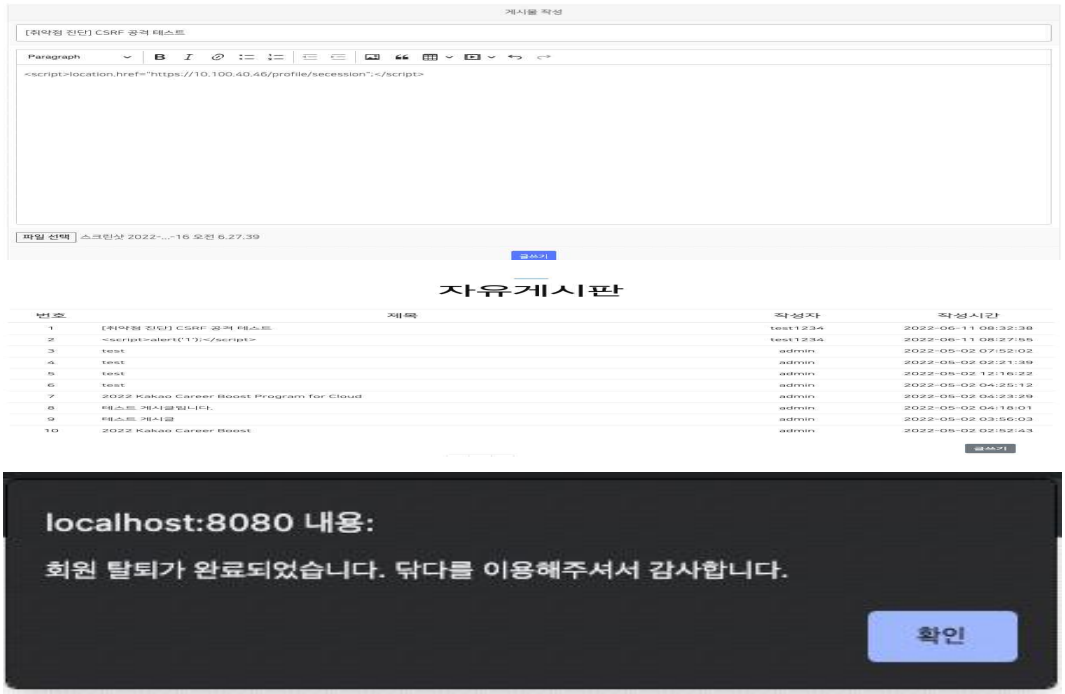
```
        <%-results.title%>
    </h2>
    <div class="subContent">
        <div>
            <div style="font-size: 15px; text-align: left; padding-left: 10%;">
                작성자: <%-results.author%> <br>등록일: <%= moment(results.time).format('YYYY년 MM월 D
            </div>

        </div>
    </div>
</div>
<div class="sub-container">

    <div class="content-wrap" style="margin: 10px;">
        <div style="width: 60%; height: 500px auto; background-color: #ffffff; margin: 0 auto;
        border: 1px solid rgb(0, 0, 0);">

            <input type="hidden" name="idx" value="<%=results._id%>" />



            <h3 style="font-size: 20px; text-align: left; padding: 20px;">
                <%- results.content%>
            </h3>
            <hr
                style="border: solid 1px rgb(89, 89, 89); margin:30px;margin-top: 30px; margin-
            <td><a style="font-size: 20px;" href='/board/download/uploads/images/<%= results.file
                    <%- results.filename%>
```

보안 권장	• 서버로 전달된 사용자 입력 값 검증 시 XSS에서 주로 사용하는 문자(< , > 등)들을 리스트로 지정하여 해당 문자가 존재할 시 삽입이 불가능하도록 설정해야 함. • URLDecoder 클래스에 존재하는 decode메소드를 통해 URL인코딩이 적용된 사용자 입력값을 디코딩함으로써 우회 공격을 차단할 수 있음.

세부점검항목	3. CSRF	점검결과	양호

점검 절차

1. XSS 공격과 같이 CSRF 공격구문을 삽입하여 회원탈퇴 코드를 실행하게함.



2. XSS 방어와 같이 공격구문 삽입시 렌더링 되지않도록 방어 구문을 작성하였음.

```
        <%-results.title%>
    </h2>
    <div class="subContent">
        <div>
            <div style="font-size: 15px; text-align: left; padding-left: 10%;">
                작성자: <%-results.author%> <br>등록일: <%= moment(results.time).format('YYYY년 MM월 [
            </div>

        </div>
    </div>
</div>
<div class="sub-container">

    <div class="content-wrap" style="margin: 10px;">
        <div style="width: 60%; height: 500px auto; background-color: ■#ffffff; margin: 0 auto;
        border: 1px solid □rgb(0, 0, 0);">

            <input type="hidden" name="idx" value="<%=results._id%>" />



            <h3 style="font-size: 20px; text-align: left; padding: 20px;">
                <%- results.content%>
            </h3>
            <hr
                style="border: solid 1px □rgb(89, 89, 89); margin:30px;margin-top: 30px; margin-
            <td><a style="font-size: 20px;" href='/board/download/uploads/images/<%= results.file
                    <%- results.filename%>
```

보안 권장	• 서버로 전달된 사용자 입력 값 검증 시 XSS에서 주로 사용하는 문자(< , > 등)들을 리스트로 지정하여 해당 문자가 존재할 시 삽입이 불가능하도록 설정해야 함. • URLDecoder 클래스에 존재하는 decode메소드를 통해 URL인코딩이 적용된 사용자 입력값을 디코딩함으로써 우회 공격을 차단할 수 있음.

세부점검항목	4. 버퍼 오버플로우	점검결과	양호

점검 절차

1. 게시판 글 작성 과정에서 제목 및 내용 글자 수에 관해 제한이 없어 버퍼 오버플로우 공격에 취약함.

```
if (id === undefined) {
    res.send(
        "<script>alert('로그인 후 게시글을 작성하세요.');history.back();</script>"
    );
} else if (title == "" || content == "") {
    res.send(
        "<script>alert('내용을 모두 작성하세요.');history.back();</script>"
    );
} else {
    connection.query(
        "insert into board (title, content, author, filename) VALUES ?",
        [[[title, content, id, req.file.filename]]],
        function (err, results, fields) {
            if (err) console.log(err);
            res.send(
                "<script>alert('글 작성이 완료되었습니다.');document.location.href='/board/page/1';</scrip
            );
        }
    );
}
```

2. else if(title.length >= 20 || content.length >= 2000), <script>alert('제목은 20자, 내용은 2000자를 넘기지 마세요.');history.back();</script> 이렇게 두가지 구문을 추가함으로서 제한적 작성 기능으로 방어함.

```
if (id === undefined) {
    res.send(
        "<script>alert('로그인 후 게시글을 작성하세요.');history.back();</script>"
    );
} else if (title == "" || content == "") {
    res.send(
        "<script>alert('내용을 모두 작성하세요.');history.back();</script>"
    );
} else if(title.length >= 20 || content.length >= 2000) {
    res.send(
        "<script>alert('제목은 20자, 내용은 2000자를 넘기지 마세요.');history.back();</script>"
    );
} else {
    connection.query(
        "insert into board (title, content, author, filename) VALUES ?",
        [[[title, content, id, req.file.filename]]],
        function (err, results, fields) {
            if (err) console.log(err);
            res.send(
                "<script>alert('글 작성이 완료되었습니다.');document.location.href='/board/page/1';</script
            );
        }
    );
}
```

보안 권장	• 외부 파라미터 입력 값을 할당하여 사용하는 경우 변수에 입력된 입력 값 범위를 검사하여 외부 파라미터 입력 값이 허용 범위를 벗어나는 경우 에러 페이지가 반환되지 않도록 조치하기를 권장함

세부점검항목	5. 파일 업로드	점검결과	양호

점검 절차

1. 파일명이 치환되지 않아 경로 추적이 가능함 또한 용량 및 확장자 제한 기능이 없어 시스템에 취약함.

```
var storage = multer.diskStorage({
    filename: function (req, file, cb) {
        cb(null, file.originalname);
    },
});

var upload = multer({
    storage: storage,
}).single("fileupload");
```

2. 파일명 치환, 업로드 파일 확장자 제한 및 용량 제한을 하여 파일업로드 공격에 방어함

```
var storage = multer.diskStorage({
    destination: function (req, file, cb) {
        //파일이 이미지 파일이면
        if (
            file.mimetype == "image/jpeg" ||
            file.mimetype == "image/jpg" ||
            file.mimetype == "image/png" ||
            file.mimetype == "image/gif"
        ) {
            console.log("이미지 파일이네요");
            cb(null, "uploads/images");
            //텍스트 파일이면
        }
    },
    //파일이름 설정
    filename: function (req, file, cb) {
        cb(null, Date.now() + "-" + file.originalname);
    },
});

const uploadFilter = function (req, file, cb) {
    if (!SUPPORTED_FILES.includes(file.mimetype)) {
        return cb(new TypeError("이미지 파일만 업로드 가능합니다."));
    }
    cb(null, true);
};

var upload = multer({
    storage: storage,
    limits: { fileSize: 10 * 1024 * 1024 }, //10MB

    fileFilter: uploadFilter,
}).single("fileupload");
```

보안 권장	• 화이트리스트 정책을 활용하여 허용 가능한 파일 확장자 목록을 만들어 허용된 확장자가 아닌 경우 업로드가 불가능 하게 설정하고, 파일 업로드 시 기존 파일명이 아닌 업로드 시간 또는 특수한 규칙에 따라 변경하여 경로 추적을 불가능 하게 할 것을 권장함

세부점검항목	6. 파일 다운로드	점검결과	양호

점검 절차

1. 파일명이 치환되지 않아 경로 추적이 가능함. 경로가 노출되어 다른 파일이나 디렉터리가 취약해짐.

```
var storage = multer.diskStorage({
    filename: function (req, file, cb) {
        cb(null, file.originalname);
    },
});

var upload = multer({
    storage: storage,
}).single("fileupload");
```

2. 파일명에 Date.now 솔트값을 추가해서 파일 경로 추적성을 방지하여 파일 다운로드 공격에 방어함. 또한 nodejs 프레임워크 use Router 함수를 사용하여 권한에 따른 경로접근을 부여함.

```
app.use("/", indexRouter);
app.use("/signup", signupRouter);
app.use("/login", loginRouter);
app.use("/resetpw", resetpwRouter);
app.use("/logout", logoutRouter);
app.use("/infor-overview", inforoverviewRouter);
app.use("/infor-history", inforhistoryRouter);
app.use("/infor-recruit", inforrecruitRouter);
app.use("/notice", noticeRouter);
app.use("/profile", profileRouter);
app.use("/board", pageRouter);
app.use("/service", serviceRouter);
app.use("/admin", adminRouter);
app.use("/findPW", findPWRouter);
app.use("/terms", termsRouter);
```

보안 권장	• 다운로드 파라미터를 대상으로 특수문자 필터링 규칙 적용하고 웹 어플리케이션 소스 시큐어 코딩 적용하여 파일 다운로드 시 우선적으로 웹 루트 상위 경로로 접근되지 않도록 권한 설정을 권장하며 파일 경로 및 파일 이름을 처리하는 파라미터 변수에 대해 null 여부를 체크 및 상대경로를 설정할 수 있는 문자(/, \\, &, . 등)가 존재할 경우 파일 다운로드가 불가능 하도록 구현하길 권장함

세부점검항목	7. 파라미터 변조 및 유효성 검증	점검결과	양호

점검 절차

1.read/:idx에서 idx가 파라미터이며 검증 없이 작동하는 부분이 취약함.

```
router.get("/read/:idx", function (req, res, next) {
    // board/read/idx숫자 형식으로 받을거
    var idx = req.params.idx; // :idx 로 맵핑할 req 값을 가져온다

    var sql = "SELECT * from board where _id=?";
    var comment = "SELECT * from comment where board_id=?";
    var path = __dirname + "/../" + "uploads/images/";

    connection.query(sql, [idx], function (err, results) {
        // 한개의 글만조회하기때문에 마지막idx에 매개변수를 받는다
        if (err) console.error("err : " + err);
        connection.query(comment, [idx], function (err, comment_results) {
            if (err) console.error("err : " + err);
            res.render("read", {
                title: "글 상세보기",
                results: results[0],
                user: req.session.name,
                comment_results: comment_results,
                length: comment_results.length - 1,
                moment,
            }); // 첫번째행 한개의데이터만 랜더링 요청
        });
    });
});
```

2. if(results==null)과 alert구문을 작성하여 파라미터에 대한 결과값이 존재하지 않으면 접근하지 못하도록 유효성 검증을 통해 파라미터 변조 공격을 방어함.

```
router.get("/read/:idx", function (req, res, next) {
    // board/read/idx숫자 형식으로 받을거
    var idx = req.params.idx; // :idx 로 맵핑할 req 값을 가져온다

    var sql = "SELECT * from board where _id=?";
    var comment = "SELECT * from comment where board_id=?";
    var path = __dirname + "/../" + "uploads/images/";

    connection.query(sql, [idx], function (err, results) {
        // 한개의 글만조회하기때문에 마지막idx에 매개변수를 받는다
        if (err) console.error("err : " + err);
        connection.query(comment, [idx], function (err, comment_results) {
            if (err) console.error("err : " + err);
            if(results==null){
                res.send(
                    "<script>alert('해당 값이 없습니다.');history.back();</script>"
                );
            } else{
            res.render("read", {
                title: "글 상세보기",
                results: results[0],
                user: req.session.name,
                comment_results: comment_results,
                length: comment_results.length - 1,
                moment,
            })}; // 첫번째행 한개의데이터만 랜더링 요청
        });
    });
});
```

보안 권장	• 웹 어플리케이션 소스코드 시큐어 코딩을 적용하여 웹 어플리케이션에서 제공하는 정보와 기능을 역할에 따라 배분하여 사용자가 변경할 수 있는 데이터의 노출을 최소화 하여야하고, 서버로 전송된 사용자 입력 값은 세션 및 DB와 비교하여 데이터 변조 유무를 검증할 것을 권장함

세부점검항목	8. 사용자 쿠키 변조	점검결과	양호

점검 절차

1. 사용자 Cookie 변조를 점검한 결과, 서버 세션을 사용하고 있으며 시크릿 키값을 이용한 암호화를 진행함. 쿠키 또한 60분의 세션타임아웃 설정이 되어있어 양호함

보안 권장	• 쿠키에 중요 정보가 포함되어 있으면 공격자가 사용자의 중요한 정보를 변조하거나 획득할 수 있는 위험이 있으므로 웹 어플리케이션 운영 시 가급적 쿠키 방식 대신 세션 방식을 사용하는 것을 권장함 • 부득이 쿠키 사용 시 SEED, 3DES, AES등 공인된 암호화 알고리즘을 사용하여 암호화를 적용시켜야 함

세부점검항목	9. 패스워드 복구시 취약점	점검결과	양호

점검 절차

1. 비밀번호 찾기 시 이메일로 인증코드가 전송되게 구성하여 중간 탈취 공격을 방어함.

```
router.post("/", (req, res) => {
  let id = req.body.id;
  let email = req.body.email;
  if (id && email) {
    connection.query(
      "SELECT * FROM users WHERE id = ? AND email = ?",
      [id, email],
      function (error, results, fields) {
        if (error) throw error;
        if (results.length > 0) {
          let password = results[0].password;
          res.send(
            "<script>alert('메일이 전송되었습니다.');history.go(-2);</script>"
          );
          const transporter = nodemailer.createTransport({
            service: "gmail",
            port: 465,
            secure: true,
            auth: {
              user: "n                l.com",
              pass: process.env.GMAIL_PW,
            },
          });
          const emailOptions = {
            from: "                .com",
            to: email,
            subject: "(주)닦다에서 비밀번호를 알려드립니다.",
            html:
              "<h1 >(주)닦다에서 " +
              id +
              "님의 비밀번호를 알려드립니다.</h1> <h2> 비밀번호 : " +
              password +
              "</h2>",
          };
          transporter.sendMail(emailOptions, res); //전송
        } else {
          res.send("Username이 맞지 않습니다.");
        }
        res.end();
      }
    );
  } else {
    res.send("Please enter Username and Email!!");
    res.end();
  }
```

보안 권장	• 웹 어플리케이션 소스코드 시큐어 코딩을 적용하여 비밀번호 찾기 시 데이터베이스에 저장되어 있는 사용자 정보와 입력한 정보가 일치하는지 비교하는 인증 구문을 적용해야 하며 인증 후 발급되는 임시 비밀번호의 경우 웹 페이지에 노출 시키지 않고 사용자의 이메일 또는 SMS를 통해 전송되도록 설정하기를 권장함

세부점검항목	10. 디렉터리 인덱싱	점검결과	양호

점검 절차

1. 유저 간 접속 권한을 개별 부여함으로서 nodejs 프레임워크의 use Router 함수를 사용해서 미리 정의해둔 경로만 접근가능하여 디렉터리 인덱싱을 방어함.

```
app.use("/", indexRouter);
app.use("/signup", signupRouter);
app.use("/login", loginRouter);
app.use("/resetpw", resetpwRouter);
app.use("/logout", logoutRouter);
app.use("/infor-overview", inforoverviewRouter);
app.use("/infor-history", inforhistoryRouter);
app.use("/infor-recruit", inforrecruitRouter);
app.use("/notice", noticeRouter);
app.use("/profile", profileRouter);
app.use("/board", pageRouter);
app.use("/service", serviceRouter);
app.use("/admin", adminRouter);
app.use("/findPW", findPWRouter);
app.use("/terms", termsRouter);
```

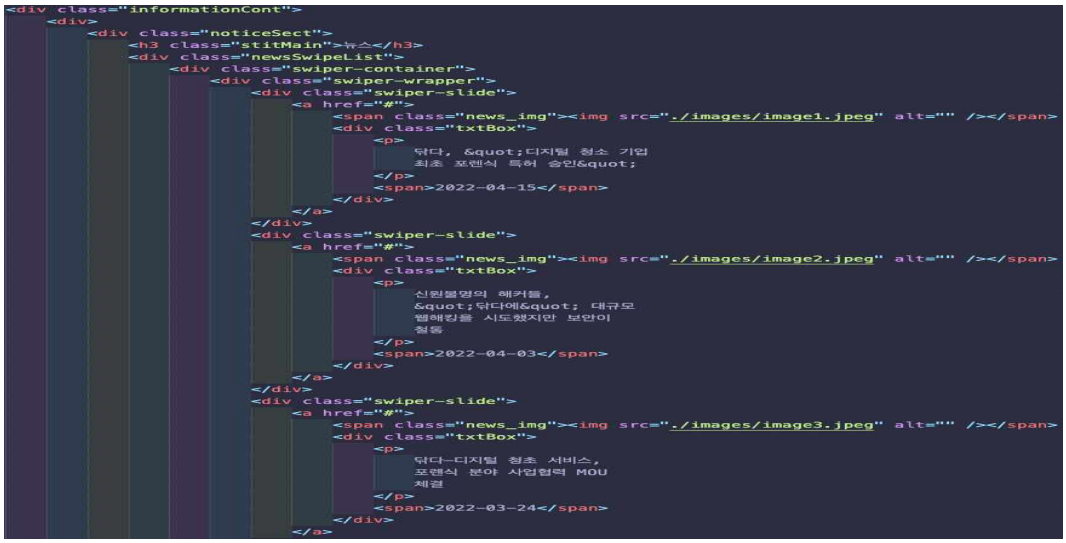
보안 권장	• 미흡한 웹 서버 보안 설정을 변경하여 웹 서버를 재시작하면 URL 강제 브라우징 시 디렉터리 및 파일 목록이 노출되지 않도록 하고 디렉터리 인덱싱 제어 이후 노출되는 에러 페이지는 '정보누출' 대응방안의 보안 설정 적용을 통해, "인덱싱 제어+ 정보누출방지"형태의 복합 대응을 하도록 권장함

세부점검항목	11. 백업 및 임시파일 노출	점검결과	양호

점검 절차

1. 홈페이지 소스코드에서 .bak, .old 와 같은 파일이 있는지 확인하고 그외에 주석처리된 정보가 있는지 확임함.

```
<!DOCTYPE html>
<html>

<head>
    <%- include ('./include/index_head.ejs') %>
</head>
<body>
    <div id="wrap">
        <header>
            <%- include ('./include/header.ejs') %>
        </header>
        <nav class="loc">
            <ol>
                <li><a href="./" title="홈 바로가기"></a></li>
                <li>메인페이지</li>
            </ol>
        </nav>
        <div class="businessCont">
            <h4 class="stitMain">주요 사업</h4>
            <ul>
                <li>
                    <a href="#">
                        <span><img src="./images/img_business_part1.jpg" alt="" /></span>
                        <p>디지털청소 서비스</p>
                        <strong>DigitalCleaning Service</strong>
                        <i title="자세히 보기"></i>
                    </a>
                </li>
                <li>
                    <a href="#">
                        <span><img src="./images/img_business_part2.jpg" alt="" /></span>
                        <p>개인정보 서비스</p>
                        <strong>Privacy Service </strong>
                        <i title="자세히 보기"></i>
                    </a>
                </li>
                <li>
                    <a href="#">
                        <span><img src="./images/img_business_part3.jpg" alt="" /></span>
                        <p>청소 솔루션</p>
                        <strong>Cleaning Solution</strong>
                        <i title="자세히 보기"></i>
                    </a>
                </li>
```



```
<div class="informationCont">
    <div>
        <div class="noticeSect">
            <h3 class="stitMain">뉴스</h3>
            <div class="newsSwipeList">
                <div class="swiper-container">
                    <div class="swiper-wrapper">
                        <div class="swiper-slide">
                            <a href="#">
                                <span class="news_img"><img src="./images/image1.jpeg" alt="" /></span>
                                <div class="txtBox">
                                    <p>
                                        닥다, &amp;quot;디지털 청소 기업
                                        최초 포렌식 특허 승인&amp;quot;
                                    </p>
                                    <span>2022-04-15</span>
                                </div>
                            </a>
                        </div>
                        <div class="swiper-slide">
                            <a href="#">
                                <span class="news_img"><img src="./images/image2.jpeg" alt="" /></span>
                                <div class="txtBox">
                                    <p>
                                        신원불명의 해커들,
                                        &amp;quot;닥다에&amp;quot; 대규모
                                        웹해킹을 시도했지만 보안이
                                        철통
                                    </p>
                                    <span>2022-04-03</span>
                                </div>
                            </a>
                        </div>
                        <div class="swiper-slide">
                            <a href="#">
                                <span class="news_img"><img src="./images/image3.jpeg" alt="" /></span>
                                <div class="txtBox">
                                    <p>
                                        닥다-디지털 청소 서비스,
                                        포렌식 분야 사업협력 MOU
                                        체결
                                    </p>
                                    <span>2022-03-24</span>
                                </div>
                            </a>
```

보안 권장	• 설치된 웹 서버에 대한 기본 페이지와 배너를 삭제하여 배너 검색에 의한 시스템 기본 정보 유출을 사전에 차단하고 톰캣, 아파치 등을 사용할 경우, 인스톨 혹은 기본 페이지 등을 삭제하고 샘플 파일 역시 삭제함 • 만일 관리 콘솔을 사용할 경우에는 반드시 접근 통제를 실시하여 임의의 위치 및 임의의 사용자가 접근할 수 없도록 하고, 관리자 계정이 유출되지 않도록 관리한다. • 관리자는 웹 디렉터리를 조사하여 *.jsp.bak 와 같은 백업파일을 모두 삭제하고 *.txt 같은 웹 페이지의 작업 도중 생성된 일반 텍스트 파일이나 그밖에 이미지 파일 등도 본래 파일 이외에는 제거하여야 한다. 백업 파일 같은 경우 언제 생성될 지 관리자의 측면에서는 알기가 매우 어려우므로 주기적으로 검사 및 삭제가 필요하며, 가급적이면 웹 서버를 실행하는데 필요한 파일을 제외하고는 모두 삭제할 것을 권장함

세부점검항목	12. 중요 데이터 암호화 전송 여부	점검결과	양호

점검 절차

1. 데어터 암호화 전송인 HTTPS 프로토콜을 적용하기 위해 SSL 인증키와 Cert 파일을 발급받아 적용함.

```
const options = {
  key: fs.readFileSync("fake_keys/key.pem"),
  cert: fs.readFileSync("fake_keys/cert.pem"),
};
```

```
https.createServer(options, app).listen(443);
```

보안 권장	• 웹 어플리케이션 소스코드 시큐어 코딩을 적용해, 중요 데이터를 암호화하여 송수신하고, 서버와 클라이언트 통신 시 중요정보가 사용되는 구간에 SSL등 암호화 통신을 적용할 것을 권장함

세부점검항목	13. 배너/포트 등 암호화 전송 여부	점검결과	양호

점검 절차

1. 포트의 경우 암호화 통신 설정과 동시에 포트번호를 지정하여 서버 설정에서 443 포트만을 오픈하게 설정함.

```
https.createServer(options, app).listen(443);
```

2. nmap 포트스캐닝을 통해 불필요한 포트가 열려 있는지 확인한 결과, 암호화 통신이 가능하여양호함.

```
┌──(leejiyu㉿leejiyu)-[~]
└─$ nmap -sV 10.100.40.46
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-12 16:16 KST
Nmap scan report for 27.nate.com (10.100.40.46)
Host is up (0.047s latency).
Not shown: 987 closed tcp ports (conn-refused)
PORT     STATE    SERVICE      VERSION
21/tcp   open     tcpwrapped
80/tcp   open     http         Apache httpd 2.4.29 ((Ubuntu))
139/tcp  filtered netbios-ssn
443/tcp  open     http         Apache httpd 2.4.29 ((Ubuntu))
445/tcp  filtered microsoft-ds
631/tcp  filtered ipp
1025/tcp filtered NFS-or-IIS
1723/tcp filtered pptp
2000/tcp filtered cisco-sccp
3000/tcp filtered ppp
3306/tcp filtered mysql


Service detection performed. Please report any incorrect results at https://n
map.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.15 seconds
```

보안 권장	• 웹 어플리케이션 소스코드 시큐어 코딩 적용을 하고 전체적인 통합 에러 페이지를 작성한 후 모든 에러코드에 대해 통합 에러 페이지로 리다이렉드 되도록 설정하여 공격자가 서버정보 및 에러코드를 수집할 수 없도록 설정하는 것을 권장함

세부점검항목	14. 관리자페이지 노출	점검결과	양호

점검 절차

1. 관리자 페이지 경우 관리자 계정으로만 접근이 가능하지만 2차 인증이 적용되어있지 않아 취약함.

```
router.get("/user/:page", function (req, res, next) {
    var id = req.session.name;
    var page = req.params.page; // 현재 페이지는 params 을 req 요청받아옴
    var sql = "select * from users order by _id desc"; // select 구절 그대로
    if (id !== "admin") {
        res.send(
            "<script>alert('관리자페이지는 관리자만 접근가능합니다.');history.back();</script>"
        );
    } else {
        connection.query(sql, function (err, rows) {
            if (err) console.log("err : " + err);
            res.render("admin_user", {
                rows: rows,
                page: page,
                length: rows.length - 1,
                page_num: 10,
                pass: true,
            });
        });
    }
});
```

2. 접속중인 IP와 관리자 지정 IP를 맵핑하여 일치해야 접근 가능하도록 2차인증을 적용시켜 방어함.

```
router.get("/user/:page", function (req, res, next) {
    var id = req.session.name;
    var ip = req.session.ip;
    var admin_ip="10.100.40.22";
    var page = req.params.page; // 현재 페이지는 params 을 req 요청받아옴
    var sql = "select * from users order by _id desc"; // select 구절 그대로
    if (id !== "admin" && ip==admin_ip) {
        res.send(
            "<script>alert('관리자페이지는 관리자만 접근가능합니다.');history.back();</script>"
        );
    } else {
        connection.query(sql, function (err, rows) {
            if (err) console.log("err : " + err);
            res.render("admin_user", {
                rows: rows,
                page: page,
                length: rows.length - 1,
                page_num: 10,
                pass: true,
            });
        });
    }
});
```

보안 권장	• 다운로드 파라미터를 대상으로 특수문자 필터링 규칙 적용하고 웹 어플리케이션 소스 시큐어 코딩 적용하여 파일 다운로드 시 우선적으로 웹 루트 상위 경로로 접근되지 않도록 권한 설정을 권장하며 파일 경로 및 파일 이름을 처리하는 파라미터 변수에 대해 null 여부를 체크 및 상대경로를 설정할 수 있는 문자(/, \\, &, . 등)가 존재할 경우 파일 다운로드가 불가능 하도록 구현하길 권장함

세부점검항목	15. 불필요한 메소드 차단 여부	점검결과	양호

점검 절차

1. nodejs 프레임워크에서 미리 정의해둔 메소드로만 접근이 가능하며, 불필요한 PUT, PATCH, DELETE 메소드는 사용하지 않았기에 불필요한 메소드는 존재하지않음.

```
router.get('/', function(req,res){ // 2
  res.render('main', {user: req.session.name});
  });
```

```
router.get("/form", function (req, res, next) {
    var id = req.session.name;

    if (id === undefined) {
        res.send(
            "<script>alert('로그인 후 게시글을 작성하세요.');history.back();</script>"
        );
    } else {
        res.render("form", { user: req.session.name });
    }
});
```

```
router.get("/download/uploads/images/:name", function (req, res) {
    var filename = req.params.name;
    console.log(req.params.name);
    console.log(filename);
    var file = __dirname + "/../uploads/images/" + decodeURIComponent(filename);
    console.log(file);
    res.download(file);
});
```

보안 권장	• 화이트리스트 정책을 활용하여 허용 가능한 파일 확장자 목록을 만들어 허용된 확장자가 아닌 경우 업로드가 불가능 하게 설정하고, 파일 업로드 시 기존 파일명이 아닌 업로드 시간 또는 특수한 규칙에 따라 변경하여 경로 추적을 불가능 하게 할 것을 권장함