# Command Injection

Command injections are among the most critical vulnerabilities in web services. They allow system command execution directly on the back-end server. If a web service uses user-controlled input to execute a system command on the back-end server, an attacker may be able to inject a malicious payload to subvert the intended command and execute his own.

Let us assess together a web service that is vulnerable to command injection.

You may have come across connectivity-checking web services in router admin panels or even websites that merely execute a ping command towards a website of your choosing.

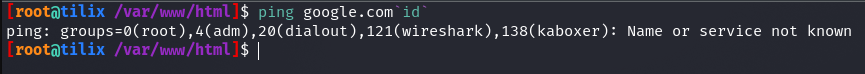
Proceed to the end of this section and click on Click here to spawn the target system! or the Reset Target icon. Use the provided Pwnbox or a local VM with the supplied VPN key to reach the target service and follow along.

Suppose we are assessing such a connectivity-checking service residing in http://<TARGET IP>:3003/ping-server.php/ping. Suppose we have also been provided with the source code of the service.

**Note**: The web service we are about to assess does not follow the web service architectural designs/approaches we covered. It is quite close to a normal web service, though, as it provides its functionality in a programmatic way, and different clients can use it for connectivity-checking purposes.

Code: php

<?php  
function ping($host\_url\_ip, $packets) {  
 if (!in\_array($packets, array(1, 2, 3, 4))) {  
 die('Only 1-4 packets!');  
 }  
 $cmd = "ping -c" . $packets . " " . escapeshellarg($host\_url);  
 $delimiter = "\n" . str\_repeat('-', 50) . "\n";  
 echo $delimiter . implode($delimiter, array("Command:", $cmd, "Returned:", shell\_exec($cmd)));  
}  
  
if ($\_SERVER['REQUEST\_METHOD'] === 'GET') {  
 $prt = explode('/', $\_SERVER['PATH\_INFO']);  
 call\_user\_func\_array($prt[1], array\_slice($prt, 2));  
}  
?>

* A function called *ping* is defined, which takes two arguments *host\_url\_ip* and *packets*. The request should look similar to the following. http://<TARGET IP>:3003/ping-server.php/ping/<VPN/TUN Adapter IP>/3. To check that the web service is sending ping requests, execute the below in your attacking machine and then issue the request.
  + yovecio@htb[/htb]$ sudo tcpdump -i tun0 icmp  
     tcpdump: verbose output suppressed, use -v[v]... for full protocol decode  
     listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes  
     11:10:22.521853 IP 10.129.202.133 > 10.10.14.222: ICMP echo request, id 1, seq 1, length 64  
     11:10:22.521885 IP 10.10.14.222 > 10.129.202.133: ICMP echo reply, id 1, seq 1, length 64  
     11:10:23.522744 IP 10.129.202.133 > 10.10.14.222: ICMP echo request, id 1, seq 2, length 64  
     11:10:23.522781 IP 10.10.14.222 > 10.129.202.133: ICMP echo reply, id 1, seq 2, length 64  
     11:10:24.523726 IP 10.129.202.133 > 10.10.14.222: ICMP echo request, id 1, seq 3, length 64  
     11:10:24.523758 IP 10.10.14.222 > 10.129.202.133: ICMP echo reply, id 1, seq 3, length 64
* The code also checks if the *packets*'s value is more than 4, and it does that via an array. So if we issue a request such as http://<TARGET IP>:3003/ping-server.php/ping/<VPN/TUN Adapter IP>/3333, we're going to get an *Only 1-4 packets!* error.
* A variable called *cmd* is then created, which forms the ping command to be executed. Two values are "parsed", *packets* and *host\_url*. [escapeshellarg()](https://www.php.net/manual/en/function.escapeshellarg.php) is used to escape the *host\_url*'s value. According to PHP's function reference, *escapeshellarg() adds single quotes around a string and quotes/escapes any existing single quotes allowing you to pass a string directly to a shell function and having it be treated as a single safe argument. This function should be used to escape individual arguments to shell functions coming from user input. The shell functions include exec(), system() shell\_exec() and the backtick operator.* If the *host\_url*'s value was not escaped, the below could happen. 
* The command specified by the *cmd* parameter is executed with the help of the *shell\_exec()* PHP function.
* If the request method is GET, an existing function can be called with the help of [call\_user\_func\_array()](https://www.php.net/manual/en/function.call-user-func-array.php). The *call\_user\_func\_array()* function is a special way to call an existing PHP function. It takes a function to call as its first parameter, then takes an array of parameters as its second parameter. This means that instead of http://<TARGET IP>:3003/ping-server.php/ping/www.example.com/3 an attacker could issue a request as follows. http://<TARGET IP>:3003/ping-server.php/system/ls. This constitutes a command injection vulnerability!

You can test the command injection vulnerability as follows.

yovecio@htb[/htb]$ curl http://<TARGET IP>:3003/ping-server.php/system/ls  
index.php  
ping-server.php