

Lab 2.3 - Command Injection

Vulnerabilities - Grepper



Command injection allows a user to enter, prepend or append an executable command to unsanitized input. It happens all the time.

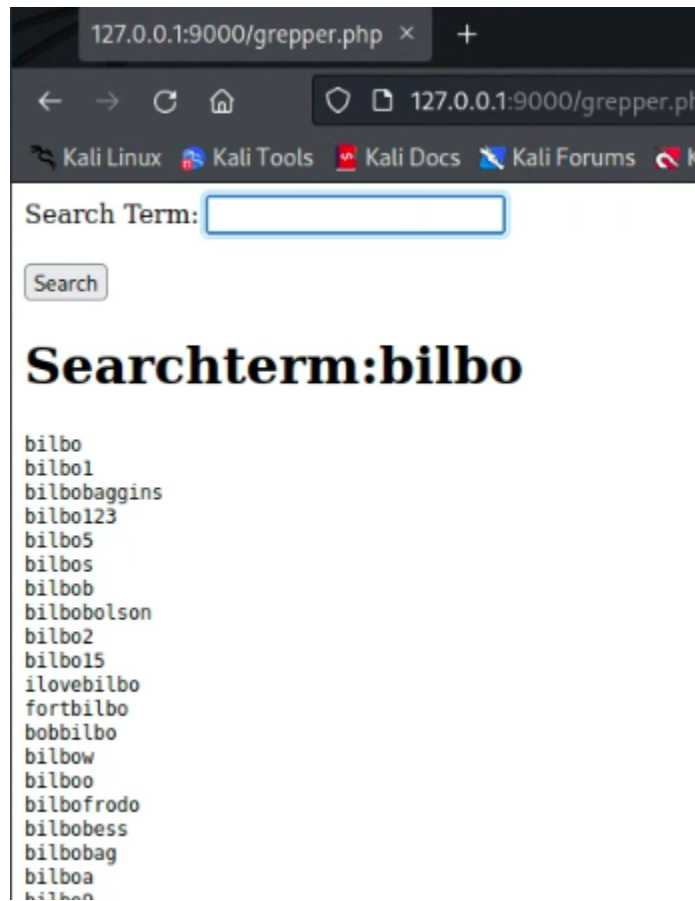
grepper.php

Create a new php file called grepper.php with the following contents. Adjust the source code as necessary to point to your version of rockyou.txt. This application will allow you to search it for dictionary items.

```
<form id="logform" method="post">
<div>Search Term: <input type="text" name="search"><div>
</select>
<div class="full-width"></br>
  <button type="submit">Search</button>
</div>
</form>
<?php
if(isset($_POST['search'])) {
  $searchterm=$_POST['search'];
  echo "<div>";
  echo "<h1>Searchterm:" . $searchterm . "</h1>";
  echo "</div>";

  echo "<pre>";
  passthru("cat /usr/share/wordlists/rockyou.txt | grep " . $searchterm);
  echo "</pre>";
}
?>
```

Deliverable 1. Try the application out and search for a string of interest.



Note that the http parameters are not visible in the url, this is because we are using POST and form data to make this happen.

💡 Though semicolon usage in prose may be underappreciated and perhaps baffling to you, it is critical in command injection. The ";" is in essence a command separator. By including it in the right place in an vulnerable application you can inject a series of your own commands.

Deliverable 2. Figure out how to run commands of your choosing. Provide a screenshot similar to the one below that shows your application output as well as commands you've snuck in.

```
127.0.0.1:9000/grepper.php x +
127.0.0.1:9000/grepper.php
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB
Search Term: 
Search
Searchterm
balrog
balrog_99
balrogvsgandalf!
balrogue
balrog81
balrog33
balrog27
balrog24
balrog21
balrog2
balrog12
balrog1117
balrog07
balrog000000
balrog00000
balrog!
eth0: flags=4163 mtu 1500
    inet 10.0.17.50 netmask 255.255.255.0 broadcast 10.0.17.255
    inet6 fe80::250:56ff:fe01:8c60 prefixlen 64 scopeid 0x20
    ether 00:50:56:a1:8c:60 txqueuelen 1000 (Ethernet)
    RX packets 32762715 bytes 5201532278 (4.8 GiB)
    RX errors 0 dropped 1731 overruns 0 frame 0
    TX packets 162804902 bytes 118430706033 (110.2 GiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73 mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10
    loop txqueuelen 1000 (Local Loopback)
    RX packets 649560 bytes 247692393 (236.2 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 649560 bytes 247692393 (236.2 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wg0: flags=209 mtu 1420
    inet 10.0.99.10 netmask 255.255.255.0 destination 10.0.99.10
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 1000 (UNSPEC)
    RX packets 123623 bytes 52412124 (49.9 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 132809 bytes 21527376 (20.5 MiB)
    TX errors 25686 dropped 0 overruns 0 carrier 0 collisions 0

champuser
```

Challenge, see if you can use this technique to invoke a reverse shell, you can catch it on another local port, alternatively you can work with a partner to exploit their version of grepper

and invoke a shell on the remote system.

Deliverable 3. Write a technical article on command injection. If you remember sec335, [shellshock](#) was an example of this type of vulnerability.