OWASP
Open Web Application Security Project

Getting started with ModSecurity

Lightning Training – AppSec USA 2015

Luca Carettoni – lcarettoni@linkedin.com
Mukul Khullar – mkhullar@linkedin.com

Training Goal

- After this training, you will be able to tune and deploy ModSecurity, use available rule packages and write simple custom rules
- From there, you will be equipped to explore additional advanced ModSecurity topics such as:
  - Persistent storage
  - Lua rules engine
  - XML parsing
  - ….and more

What is Required

- Oracle VM VirtualBox
- Download the VM from the Lab Wifi Network

  Network Name: **owaspmodsec**

  Network Password: **nointernetheregoaway**

  http://192.168.1.2/owaspmodsec.ova

  MD5 (owaspmodsec.ova) =
  6f838d64b5946a6e7c6d7e0a25653465

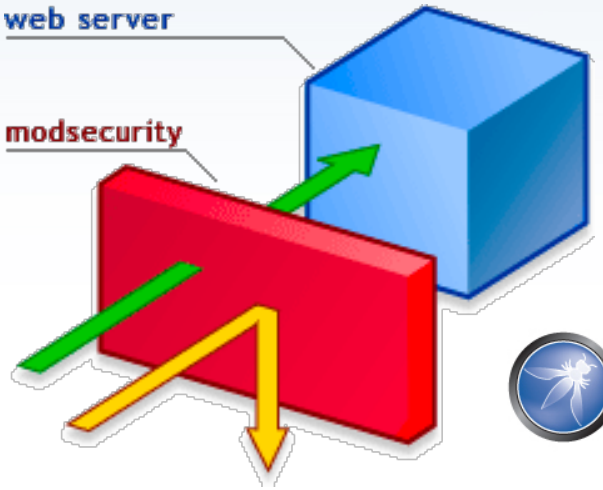
What We Use

- Apache HTTPd
- ModSecurity 2.9
- OWASP ModSecurity CRS
- Damn Vulnerable Web Application

Introduction

- The OpenSource Web Application Firewall
  https://www.modsecurity.org/

- "ModSecurity is a toolkit for real-time web application monitoring, logging and access control" – *Ivan Ristic*



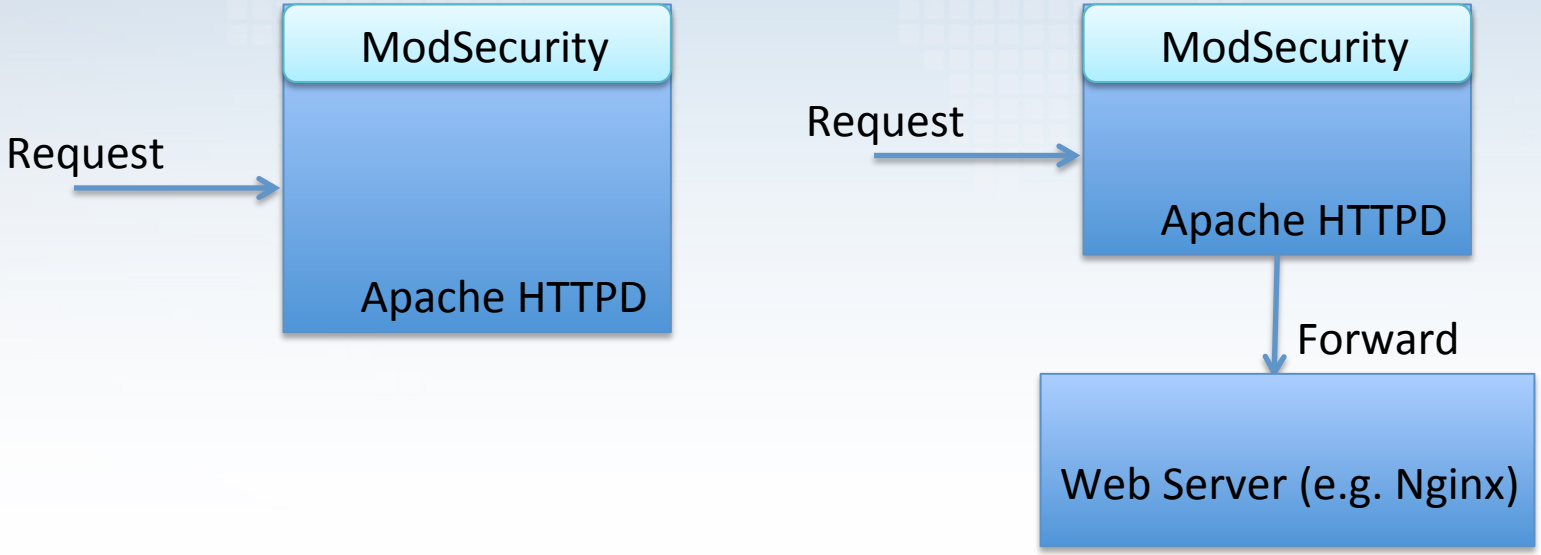
Deployment Usage

- **Logging and monitoring**
  - ModSecurity gives you the ability to log raw HTTP traffic
- **Virtual patching**
  - ModSecurity can be used to block HTTP requests based on flexible and extensible rules set
- **Data leakage safety net**
  - ModSecurity can also block HTTP responses to prevent data leakage
- **Hardening**
  - ModSecurity can reduce the attack surface by limiting specific HTTP types, headers and parameters

Deployment Scenarios

Embedded with the WS

Reverse Proxy ← We use this mode



OWASP
Open Web Application Security Project

Preparation For Labs 1/3

Is your virtual machine up and running?

Login with:
User: *owasp*
Password: *owasp*

Preparation For Labs 2/3

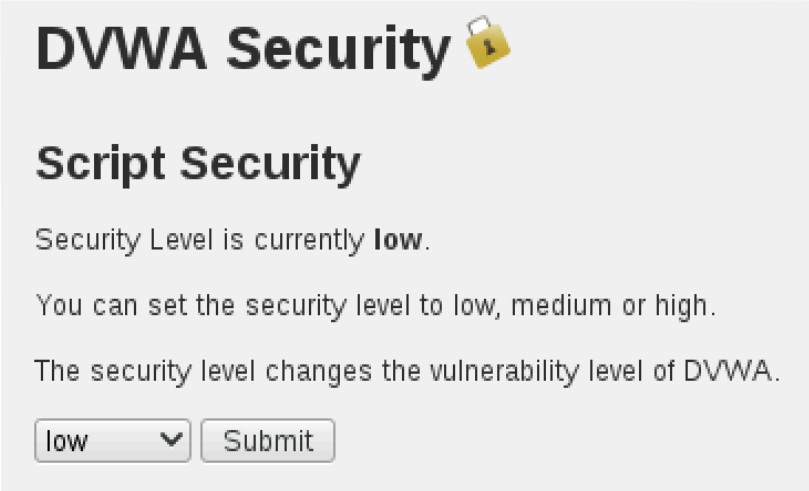
We have already installed:

- Nginx on port 8888/tcp, localhost only
- Apache (reverse proxy) on port 80/tcp and 443/tcp

Preparation For Labs 3/3

Nginx is hosting a broken web app (DVWA)

- Open Firefox and visit **https://owaspmodsec/dvwa/**
- Username: *admin*, Password: *password*
- Verify DVWA Security setting



Lab1 - Install

ModSecurity can be compiled from source code, or simply installed via OS packages

- It's 2015. Let's use the binary package

```
$ sudo dnf install mod_security
```

Lab2 – Exploring the default config

Let's have a look at the default configuration files and directories:

```
$ less /etc/httpd/conf.d/mod_security.conf
$ sudo nano /etc/httpd/conf.d/mod_security.conf

    SecDebugLogLevel 0 → SecDebugLogLevel 3
…
SecDebugLog /var/log/httpd/modsec_debug.log
SecAuditLog /var/log/httpd/modsec_audit.log

$ ls /etc/httpd/modsecurity.d/
```

Lab2 – Exploring the default config

After restarting Apache HTTPd, let's verify that ModSecurity is properly enabled…

- It works, but there are no rules yet!

```
$ sudo systemctl restart httpd
$ sudo tail -f /var/log/httpd/error_log

…
[Sat Jun 13 21:46:20 2015] [notice] ModSecurity for Apache/
2.7.3 (http://www.modsecurity.org/) configured
…
```

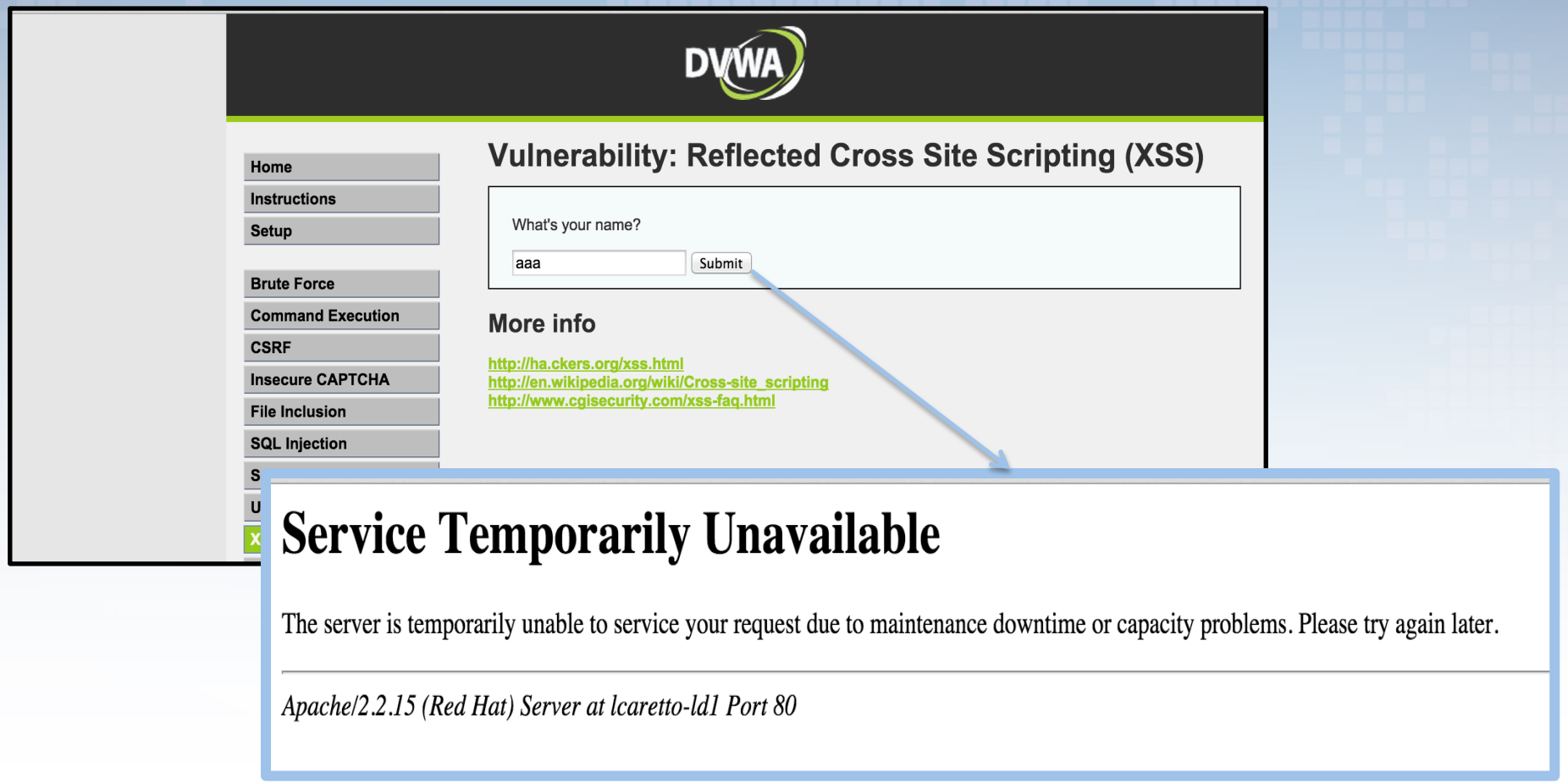
Lab3 – Write your first rule

We want to create a rule to block pages containing the HTTP GET parameter "name"

- In DVWA, visit the XSS Reflect exercise
- What do you expect to see?

```
$ sudo nano /etc/httpd/modsecurity.d/activated_rules/
firstrule.conf

SecRule ARGS_GET_NAMES "name" "id:'1',phase:
2,log,deny,status:503"
```

Lab3 – Write your first rule



OWASP
Open Web Application Security Project

Lab3 – Write your first rule

Let's experiment a bit more with our rule

- Enable detection, instead of blocking
- Change the HTTP Status Error
- Use ARGS and a regexp definition

```
$ sudo nano /etc/httpd/conf.d/mod_security.conf

  SecRuleEngine On → SecRuleEngine DetectionOnly

$ sudo systemctl reload httpd
$ sudo tail -f /var/log/httpd/error_log
$ sudo tail -f /var/log/httpd/modsec_audit.log
```

Lab4 – Install OWASP ModSecurity CRS

The project provides pluggable rules to detect common web vulnerabilities

- Let's see how to use them:

```
$ git clone https://github.com/SpiderLabs/owasp-modsecurity-crs
$ sudo mv owasp-modsecurity-crs/ /etc/httpd/modsecurity.d/
$ sudo su -

# cd /etc/httpd/modsecurity.d/
# cp owasp-modsecurity-crs/
modsecurity_crs_10_setup.conf.example
modsecurity_crs_10_setup.conf
```

Lab4 – Install OWASP ModSecurity CRS

```
# for f in `ls owasp-modsecurity-crs/base_rules/` ; do ln -s /
etc/httpd/modsecurity.d/owasp-modsecurity-crs/base_rules/
$f activated_rules/$f ; done

# for f in `ls owasp-modsecurity-crs/optional_rules/` ; do ln -
s /etc/httpd/modsecurity.d/owasp-modsecurity-crs/
optional_rules/$f activated_rules/$f ; done

# ls -auhl activated_rules
# systemctl reload httpd
# tail -f /var/log/httpd/error_log
```

Lab4 – Install OWASP ModSecurity CRS

Restart your browser, and login in DVWA

- What's happening?
- Look at the logs. Which rules are triggered? Why?

```
$ sudo nano /etc/httpd/conf.d/mod_security.conf

        SecRuleEngine DetectionOnly → SecRuleEngine On

$ sudo systemctl reload httpd
```

Lab5 – Tune OWASP ModSecurity CRS

It's time to tune the rules for our web app

```
$ sudo nano /etc/httpd/conf.d/mod_security.conf

        Include modsecurity.d/disabled.conf

$ sudo nano /etc/httpd/modsecurity.d/disabled.conf

        SecRuleRemoveById 900046

$ sudo systemctl reload httpd
```

Lab5 – Tune OWASP ModSecurity CRS

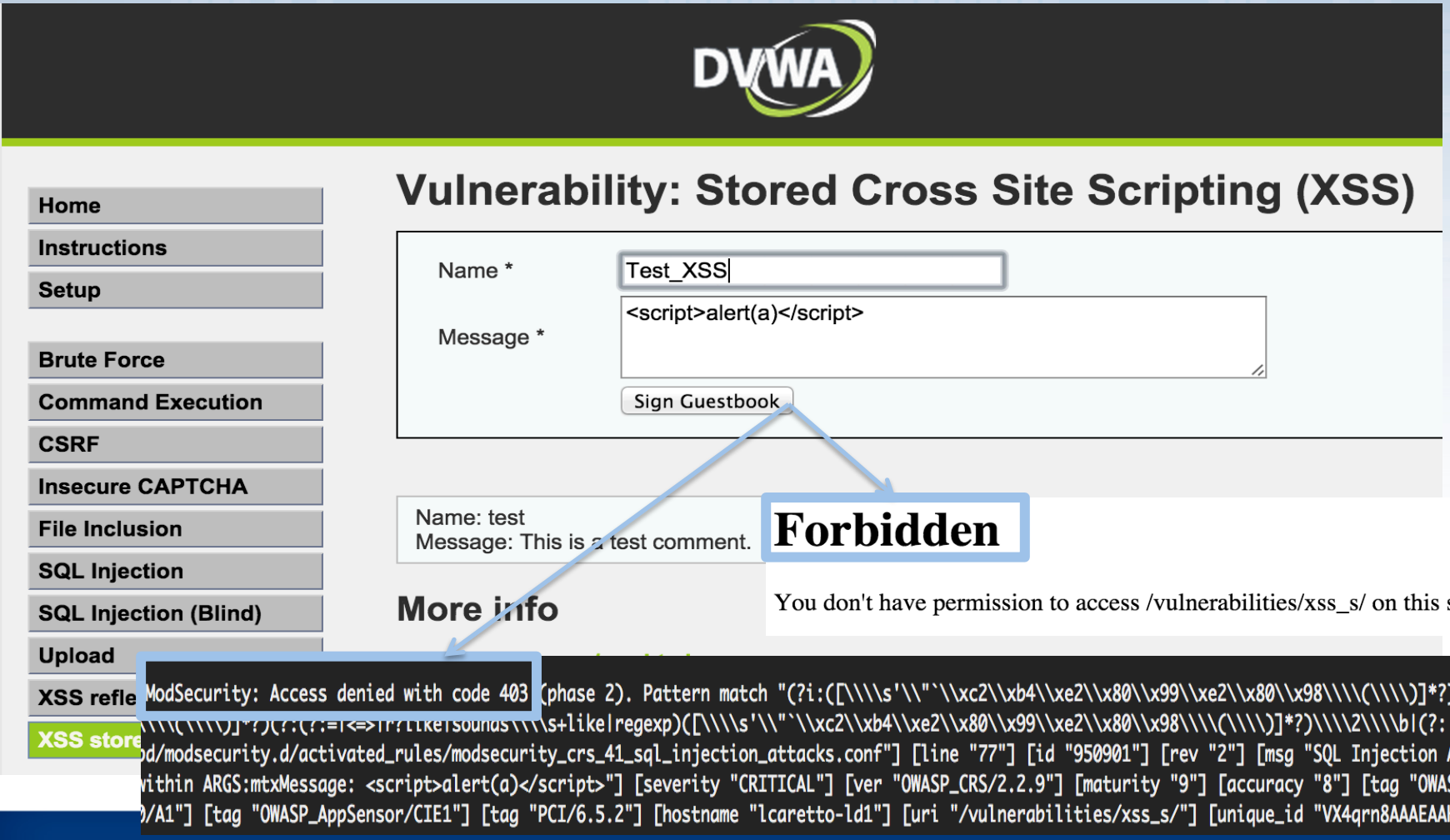
Check the logs again. Still see a warning?

```
$ sudo nano /etc/httpd/modsecurity.d/disabled.conf

    <LocationMatch /vulnerabilities/xss_r/>
        SecRuleRemoveById 981143
     </LocationMatch>

$ sudo systemctl reload httpd
```

Now, try again…

Lab5 – Tune OWASP ModSecurity CRS

Time to experiment with real vulnerabilities



Lab5 – Tune OWASP ModSecurity CRS

It's working!

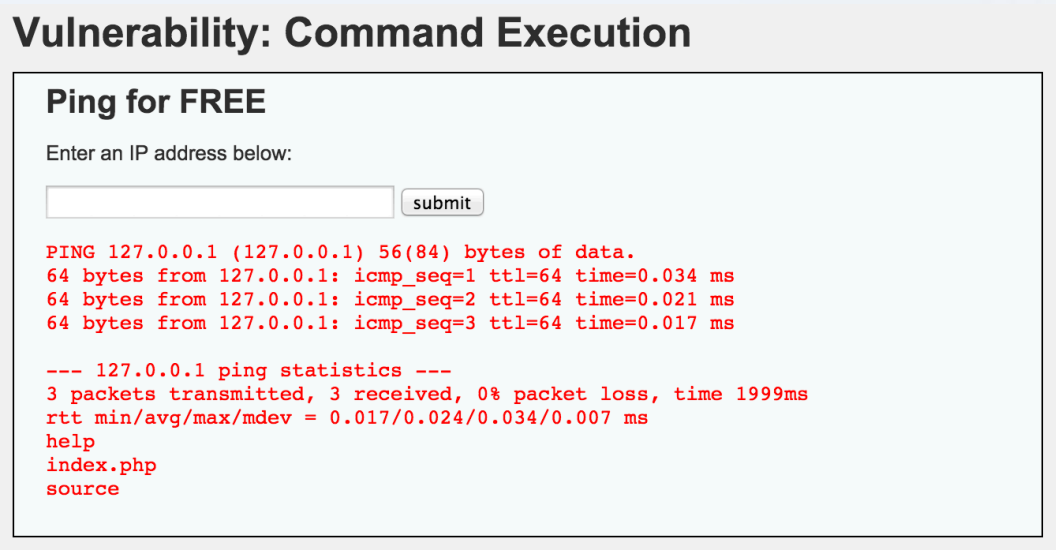
Let's experiment a bit more with the following exercises:

- SQL Injection
- Cross-Site Request Forgery
  - Is it blocking? Why?
- Command Execution
  - Does it work?

Lab5 – Tune OWASP ModSecurity CRS

Command Execution exercise

- Attack payload: *127.0.0.1 && ls*
  - *Returns 403*
- Attack payload: *127.0.0.1; ls*
  - *Woot! Bypass?*



Lab5 – Tune OWASP ModSecurity CRS

Let's write a rule to allow IP addresses only

- Positive security model (whitelisting)

```
$ cd /etc/httpd/modsecurity.d/
$ sudo nano activated_rules/whitelist.conf

SecRule ARGS:ip "!^[\d\.]+$" "id:'2',phase:2,block,msg:'Host is not IP address'"

$sudo systemctl reload httpd
```

Try again. What's happening?

Lab6 – Setup a custom Error Page

Default error pages are boring. Be creative!

```
$ sudo nano  /etc/httpd/conf/httpd.conf

    ErrorDocument 403 "<html><body bgcolor=
    \"#FF0000\"><h1>ModSecurity is here! Go away...</h1></
    body></html>”

$ sudo /etc/init.d/httpd restart
```

Lab7 – Preventing data leakage

Let's see how to block PHP's phpinfo() output

- Attackers can leverage this page for info gathering
- We're building a system to prevent data leakage

```
$ sudo nano mod_security.conf

    SecResponseBodyAccess Off → SecResponseBodyAccess On

$ sudo nano activated_rules/custom_info_leak_block.conf

    SecRule RESPONSE_BODY "@rx <title>phpinfo\(\)</title>"  "id:'1234',phase:
    4,log,block,t:none,msg:'PHP phpinfo() output detected'"

$ sudo systemctl reload httpd
```

Lab8 – Implementing Rate Limiting

$ sudo nano activated_rules/global_rate_limiting.conf

```
<LocationMatch "^/">
  SecAction initcol:ip=%{REMOTE_ADDR},pass,nolog,id:12345
  SecAction ”id:12346,phase:5,deprecatevar:ip.somepathcounter=1/1,pass,nolog"
  SecRule IP:SOMEPATHCOUNTER "@gt 60" ”id:12347,phase:2,pause:300,deny,status:
509,setenv:RATELIMITED,skip:1,nolog"
  SecAction ”id:12348,phase:2,pass,setvar:ip.somepathcounter=+1,nolog"
  Header always set Retry-After "10" env=RATELIMITED
</LocationMatch>
```

$ sudo systemctl reload httpd

http://johnleach.co.uk/words/1073/rate-limiting-with-apache-and-mod-security

Lab8 – Implementing Rate Limiting

Let's test this rule…

```
$ sudo rm /etc/httpd/modsecurity.d/activated_rules/modsecurity_*
$ sudo systemctl reload httpd

$ while(true); do time curl -s -k https://owaspmodsec/dvwa/ ; done

real    0m0.113suser 0m0.018ssys    0m0.030s
…
real    0m0.411suser 0m0.017ssys    0m0.028s
```

That's all for today

From here, you can learn something new everyday

As an example:

- Use ModSecurity to identify vulnerabilities or misconfigurations in your web application
  - Detect missing CSRF tokens, X-Frame-Options header, Improper Content-Type, …
  - Identify pages with exception stack traces
  - Setup email alert once a rule is triggered (using *exec*)

References

- ModSecurity Reference Manual
  - https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual
- ModSecurity Handbook
  - https://www.feistyduck.com/books/modsecurity-handbook/