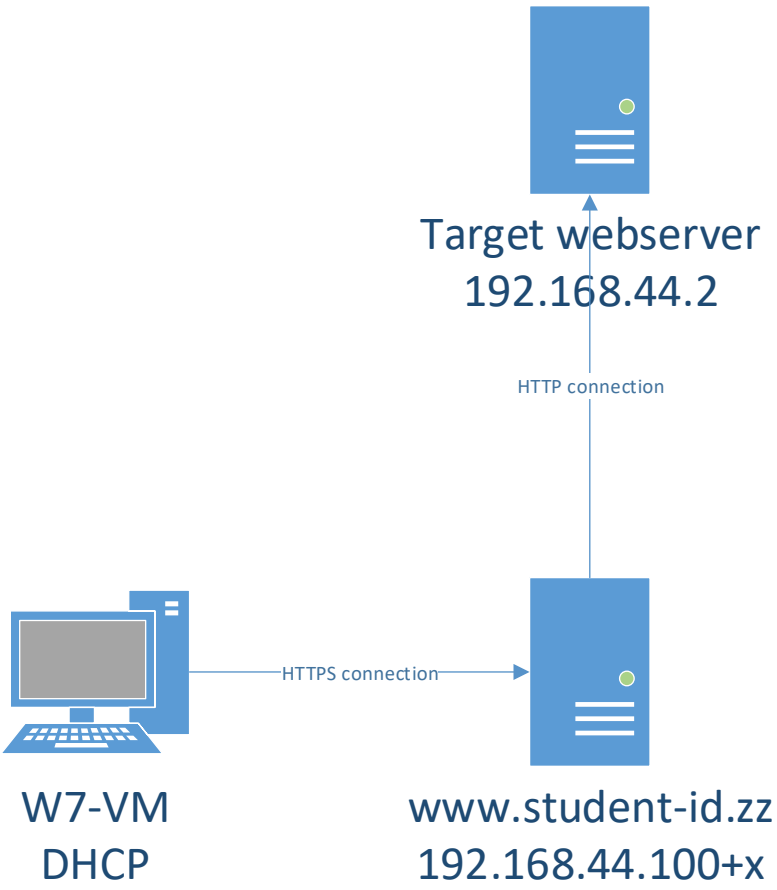
Lab4 – ModSecurity

You can use this lab manual for your personal documentation. Use screenshots for your own documentation, there will be questions later on that may point to this lab manual. Take care to check if you need to collect some information from the lab for the answers.

\ at the end of the line is used to mark that the command needs to be on one line. Replace **student-id** with your own student-id and **x** or **y** as your VMs correct IP in the labs.

The labs use the following topology:



All VMs in this lab are in VirtualBox ***Bridged*** network. The machines that have static IP need to have an offset, check the topology image for reference. USE YOUR WIN7 WORKSTATION IP ADDRESS AS **x**.

All templates for VMs can be found in \\ghost.labranet.jamk.fi\virtuaalikoneet\TTKS\

- ## Install mod_security

To enable the use of mod_security, install the module and CRS ruleset:

```
yum install mod_security mod_security_crs
```

The default ruleset for Centos7 version of the plugin are installed in **/usr/lib/modsecurity.d/base_rules**. You can enable these by creating a symbolic link to the rulesets in **/etc/httpd/modsecurity.d/activated_rules**, for example:

```
ln -s /usr/lib/modsecurity.d/base_rules/modsecurity_rulename.conf
   /etc/httpd/modsecurity.d/activated_rules/
```

You can also create new rulesets in that folder, like we will do in later steps.

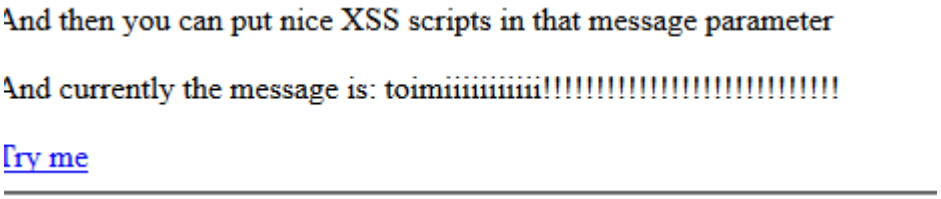
First, remove all currently enabled rules and restart apache (NOTE: Be careful with the rm command!):

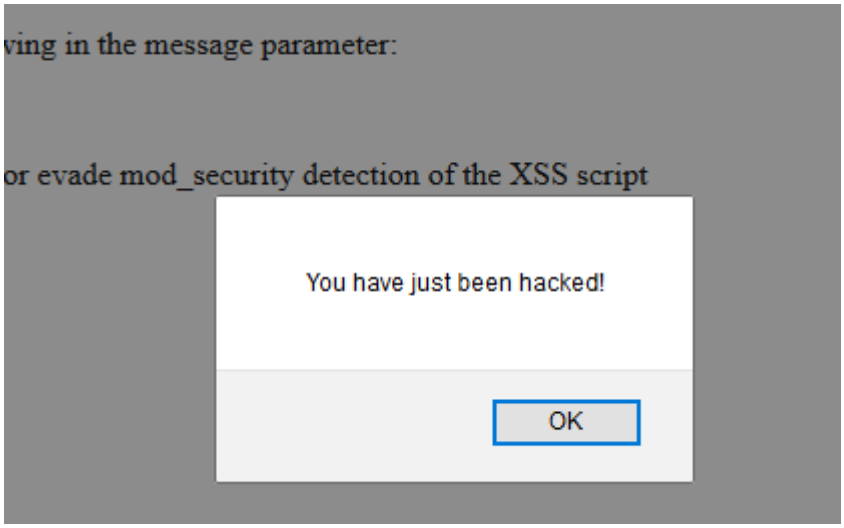
```
rm -rf /etc/httpd/modsecurity.d/activated_rules/*
systemctl restart httpd
```

QUESTIONNAIRE: How would you set mod_security to alert only and not block?

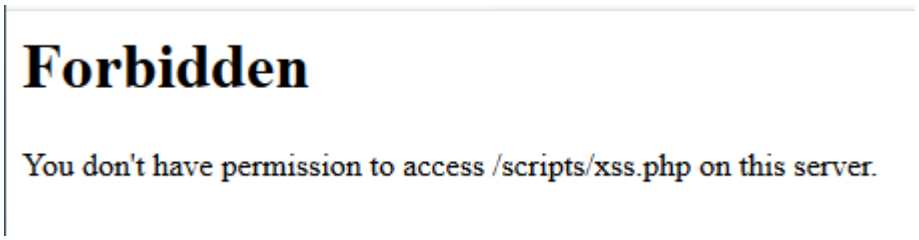
- ## Using preset rules

Using the proxy server, browse to /scripts –URI. There should be several different test cases. Start with the simple xss.php. It contains a sample of how to include a script in the page using XSS vulnerability. Test it first.





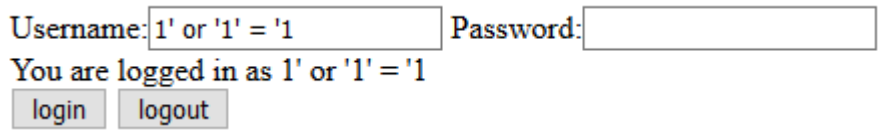
To prevent this, find out the correct CRS rule configuration file and link it to /etc/httpd/modsecurity.d/activated_rules/. You need to restart httpd after this.



Also check a simple login.php. This login prompt does not do any kind of input sanitizing. The correct account / password combination is **teppo / kissa123** which can be used for testing.

Now try a simple SQL login injection and log in with both username and password set as:

```
1' or '1' = '1
```



As you can see, you have just logged in as an imaginary account. As previously, find the correct ruleset and link it to use.

Forbidden

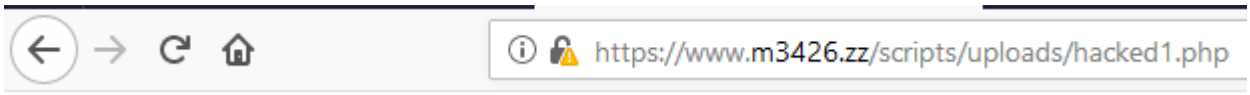
You don't have permission to access /scripts/login.php on this server.

- **Custom rules**

Next move on to **upload.php**. This script lets you upload files to an upload folder, which can be easily abused to upload malicious files, for example php scripts. Try to upload a script file (for example hacked.php) containing the following:

```
<?php
echo "Hacked!";
?>
```

Then access the script file in the uploads folder. This does nothing severe, but demonstrates how file extension/content check is important!



Hacked!

Now we can create a custom rule for mod_security to deny certain file extensions in uploads. Add this to etc/httpd/modsecurity.d/activated_rules/90_custom.conf:

```
SecRule REQUEST_FILENAME "upload.php"
   "id:'50000',chain,deny,log,msg:'Tried to upload a PHP file'"
SecRule FILES "@rx .*\.php$"
```

Forbidden

You don't have permission to access /scripts/upload.php on this server.

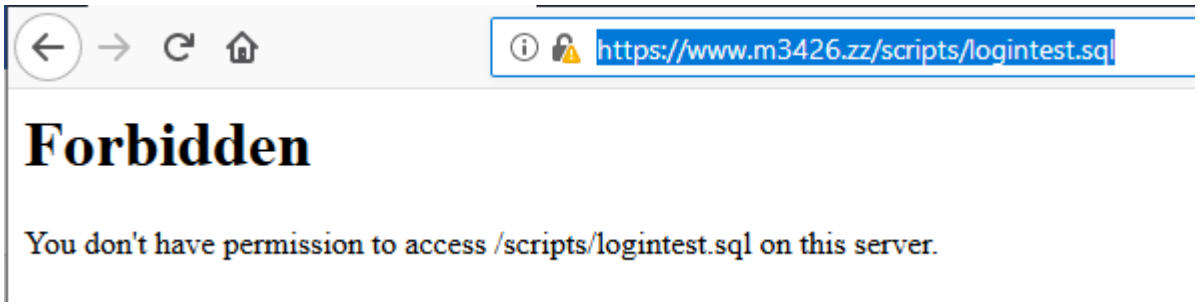
QUESTIONNAIRE: Explain the parts in the SecRule syntax

Next, prevent download of **logintest.sql** using a simple REQUEST_FILENAME rule:

- File name is \.sql$
- id is 50001
- Action is deny,log,msg like above
- Add some log message to warn about SQL Extraction

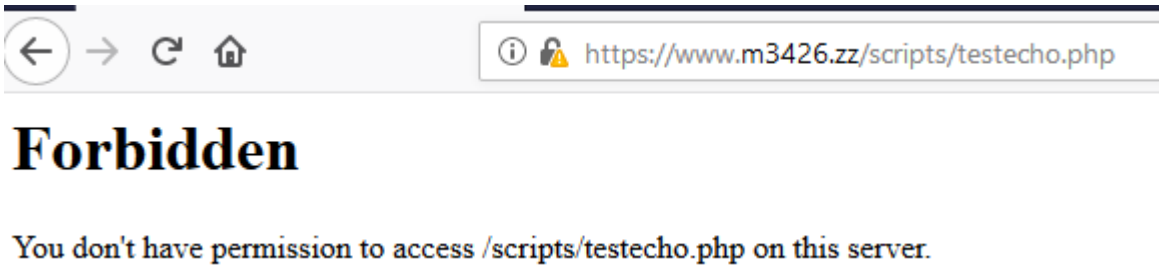
Test the rule.

```
SecRule REQUEST_FILENAME "logintest.sql" "id:'50001',deny,log,msg:'Tried to download sql file'"
#SecRule FILES "@rx .*\.sql$"
```



Lastly, create a rule for **testecho.php**, preventing the use of traditional finnish swear words such as perkele. Use id of 50002 and chain the rule like in the FILES example, but use the following style of rule for request body:

```
SecRule REQUEST_BODY "@rx (?i:(pills|insurance|rolex))"
```



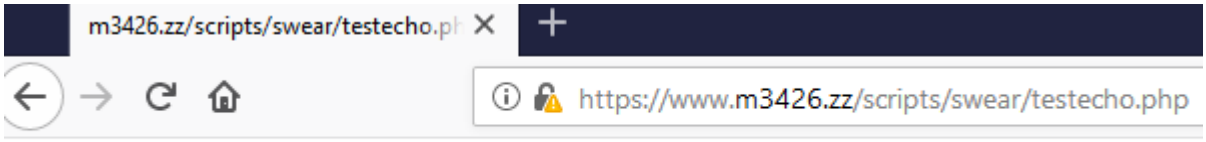
QUESTIONNAIRE: Explain the reason why the regex has an ?i: in it.

- **Exceptions**

In the folder **swear** exists another testecho.php. Make an exception for this location to turn off ONLY the swearword rule. XSS injections should still be blocked. You can remove a SecRule from an Apache **Location** with the following example:

```
<Location "/foldername/filename.php"
   <IfModule security2_module>
           SecRuleRemoveById <id-number-of-rule>
   </IfModule>
</Location>
```

```
<Location "/scripts/swear/testecho.php">
        <IfModule security2_module>
                SecRuleRemoveById 50002
        </IfModule>
</Location>
```


m3426.zz/scripts/swear/testecho.ph
https://www.m3426.zz/scripts/swear/testecho.php

perkele