Magento® An Adobe Company | U

# Exercise Solutions

Magento Security for Developers

https://u.magento.com/

v2.3

training@magento.com

u.magento.com

u.magento.com/certification

# Contents

# Unit 1. Introduction to Magento Security

## 1.2 Security Best Practices

### Exercise 1.2.1

Securing the Admin panel:

1. Change the Admin URL
   a. What is the CLI Command?
   b. How do you change it from the Magento Admin?
   c. How do you change from the env.php file?
2. Update Admin account security
3. ReCAPTCHA (see Unit 3.2
4. Set up Google authentication for 2FA and enable 2FA

**Solution**

Renaming the Admin panel is considered by many as a ”security by obscurity” approach that is plain wrong.

However, based on our experience we clearly see the attackers target first the easiest targets - and to automate large scale attacks they try typical names like /admin, /control, /administrator and similar. Changing the Admin panel URL won’t stop a sophisticated attacker targeting specifically your store but will significantly reduce the number of automated opportunistic attacks.

**Part 1**

1. **Command:** *bin/Magento setup:config:set --backend-frontname=<value>*
2. **Change from the Magento Admin:**
   - On the *Admin* sidebar, go to Stores > *Settings* > Configuration.
   - In the left panel, expand *Advanced* and choose Admin.
   - Expand the Admin Base URL section.
   - Set the configuration options for the custom URL:
     - If needed, clear the Use system value checkbox to change the setting:
       - Set Use Custom Admin URL to Yes.
       - Enter the Custom Admin URL: http://yourdomain.com/magento/
         *The Admin URL must be in the same Magento installation and have the same document root as the storefront.
       - Set Custom Admin Path to Yes.
       - Enter the Custom Admin Path.
         The path that you enter is appended to the Custom Admin URL after the last forward slash.
   - When complete, click Save Config.
   - After the changes are saved, **Sign Out** of the Admin. Then, log back in using the new Admin URL and path.
3. **Change from the env.php file**
   - Open the **app/etc/env.php** file in a text editor and change the name of the [admin] path. Make sure to use only lowercase characters. Then, save the file.

On the server, the *admin path* is located in the app/etc/env.php file. Look for the <adminhtml> argument in the <admin> section:

  - **Default Admin Path:** # <frontName><![CDATA[admin]]></frontName>
  - **New Admin Path:** # <frontName><![CDATA[backend]]></frontName>

- Use one of the following methods to clear the Magento cache:
  - On the *Admin* sidebar, go to **System** > *Tools* > **Cache Management**. Then, click **Flush Magento Cache**.
  - On the server, navigate to the var/cache folder and delete the contents of the cache folder.

**Part 2**

Log into the Admin panel and go to Stores > Configuration > Advanced > Admin > Security

1. Limit password reset
2. Set maximum number of login failures

**Part 3**

**<u>ReCAPTCHA exercise to be done in unit 3.2</u>**

**Part 4**

**Supported providers:** U2F Key, Google Authenticator, Authy & Duo

**Recommendation for this lab:** While other options exist, we will use Google Authenticator. Google Authenticator is not necessarily the best option for production.

Follow the steps below to enable 2FA:

1. Download and open Google authenticator app on your mobile
2. On the *Admin* sidebar, go to **Stores** > *Settings* > **Configuration**.
3. In the left panel, expand *Security* and choose **2FA**.
4. Expand () the **General** section, if necessary, and set **Enable Two Factor Auth** to Yes.
5. Select **Google Authenticator** for Force providers
6. Expand Google Authenticator
   a. Enable this provider — Set to Yes.
   b. (Optional) **Enable “trust this device” option** — Set to one of the following:
      i. Yes — The user does not have to enter their authenticator code for every login per device.
      ii. No — Forces authentication for every login **(Strongly Recommended)**
7. When complete, click Save Config.
8. On admin panel enter your username and password
9. A QR code will be displayed, scan the QR code displayed on admin portal using your mobile phone to sync authenticator to Magento, a new entry is created in your app

10. 6-digit OTP is generated, enter the OTP and confirm
11. You will be logged in to the admin panel with 2FA enabled. Every future login to admin panel will now require a password and OTP

# Unit 2. Secure Programming

## 2.3 Defensive PHP Programming

### Exercise 2.3.1

1. Review the provided customer facing form for security vulnerabilities

   > exercise_1.php

2. You can run this form on your local machine with the following command

   > php -S localhost:8080 ./example_1.php

3. Then visit:

   > http://localhost:8080

**Solution:**

- Line 10 – XSS Vulnerability – Unescaped User Input to HTML Output
- Line 15 – XSS Vulnerability – Unescaped User Input to HTML Output
- Line 22 – XSS Vulnerability – Unescaped User Input to HTML Output
- Line 26 – XSS Vulnerability – Unescaped User Input to HTML Output
- Line 30 – XSS Vulnerability – Unescaped User Input to HTML Output
- Line 48 – RCE Vulnerability – Unescaped User Input to Shell Command

### Exercise 2.3.2

1. Review the provided admin facing form for security vulnerabilities

   > exercise_2.php

2. If you have php installed on your machine, you can run this form on your local machine with the following command

   > php -S localhost:8080 ./example_2.php

3. Then visit:

   > http://localhost:8080

**Solution**

- Line 22 – XSS Vulnerability – Unescaped User Input to HTML Output
- Line 41 – XSS Vulnerability – Unescaped User Input to HTML Output
- Line 42 – XSS Vulnerability – Unescaped User Input to HTML Output

## 2.4 Magento Specific Secure Coding

### Exercise 2.4.1

Spot the vulnerability

```
<!-- app/code/Vendor/Example/view/adminhtml/customer/recent_searches.phtml -->
<ul>
<?php foreach ($this->getRecentSearches() as $search): ?>
    <li><?php echo $search; ?></li>
<?php endforeach; ?>
</ul>
```

**Solution**

This code is vulnerable to XSS as $search is not escaped.

### Exercise 2.4.2

Spot the vulnerability

```
// app/code/Vendor/Example/Block/ContentSuggestions.php
function getSuggestions() {
    $connection = $this->_resource->getConnection();
    $q = $this->request()->getParam('q');
    $query = 'SELECT content FROM cms_page WHERE title LIKE \'%' . $q '%\'';
    $results = $connection->fetchAll($$query);
    return $results;
}
```

**Solution**

The code is vulnerable to SQL injection as $q is passed directly to the query.

### Exercise 2.4.3

Spot the vulnerability

```
// app/code/Vendor/Example/Controller/Cart/Reclaim
public function execute()
{
    $cartContents = unserialize(base64_decode($this->getRequest()->getParam('cart')));

    $this->updateCartFromRequest($cartContents);

    $resultRedirect = $this->resultRedirect->create(ResultFactory::TYPE_REDIRECT);
    $resultRedirect->setUrl($this->_redirect->getRefererUrl());

    return $resultRedirect;
}
```

**Solution**

The code is vulnerable to PHP object injection as it uses the unserialize function on untrusted input.

## 2.5 Restricting Access in Magento

### Exercise 2.5.1

You are tasked to create a new admin page. Create the ACL rule, the controller and add an item in the menu to this page.

**TIP: TEMPORARILY** disable the option *Add Secret Key to URLs* in Stores > Configuration > Advanced > Admin.

Doing so will allow you to directly access the controller URL in the browser and test different user privileges to make sure the page is not loaded for an unauthorized admin.

**Solution**

Refer to the code zip provided:  2.5 Exercise.Training-AclExample-module.zip

**Step 1**

Register your module. See Training/AclExample/etc/module.xml and registration.php for a reference.

**Step 2**

Add Store configuration and a link item to the admin sidebar menu – see etc/adminhtml/system.xml and etc/adminhtml/menu.xml respectively for reference.

**Step 3**

Add an admin route and controller action – see etc/adminhtml/routes.xml and Controller/Adminhtml/Index for the reference. The controller action will be executed when admin users access the link introduced in step 2.

**Step 4**

Create etc/acl.xml file to add resources added in step 2 and 3 - Training_AclExample::config and Training_AclExample::controller to the acl resource tree. See etc/acl.xml for reference.

## 2.6 Secure Programming

### Exercise 2.6.1

OWASP ZAP Exercise (20 minutes)

1. Install Java JRE 8 or later
2. Install OWASP ZAP
3. Initiate browser
4. Proxy a request to your Magento store
5. Begin Tamper of Search Input Box
6. Begin fuzzing

**Solution**

Refer to slides of section 2.6 for a step-by-step solution.

# Unit 3. Operations

## 3.2 Restricting Access in Magento

### Exercise 3.2.1

Perform a log analysis

1. Import logs into Loggly
2. Count visits by IP address
3. Show results

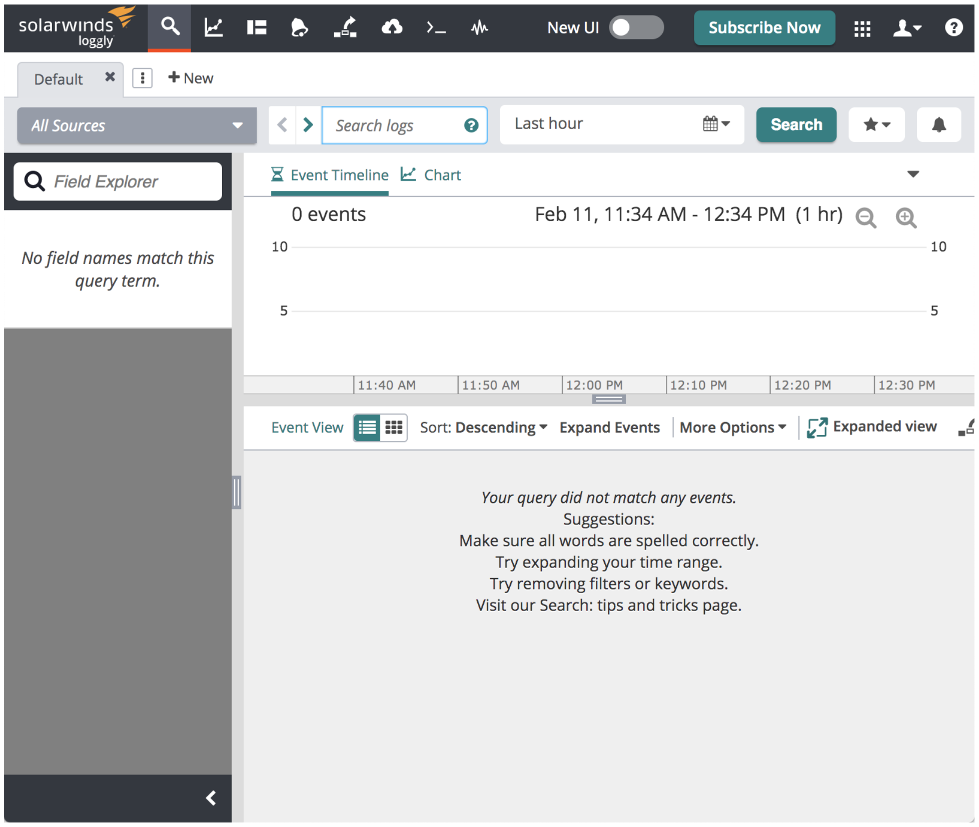
**Solution**

**Step 1**

Download Nginx access logs to local machine

**Step 2**

Create a new account at loggly.com



No field names match this query term.

0 events Feb 11, 11:34 AM - 12:34 PM (1 hr)

*Your query did not match any events.*
Suggestions:
Make sure all words are spelled correctly.
Try expanding your time range.
Try removing filters or keywords.
Visit our Search: tips and tricks page.

**Step 3**

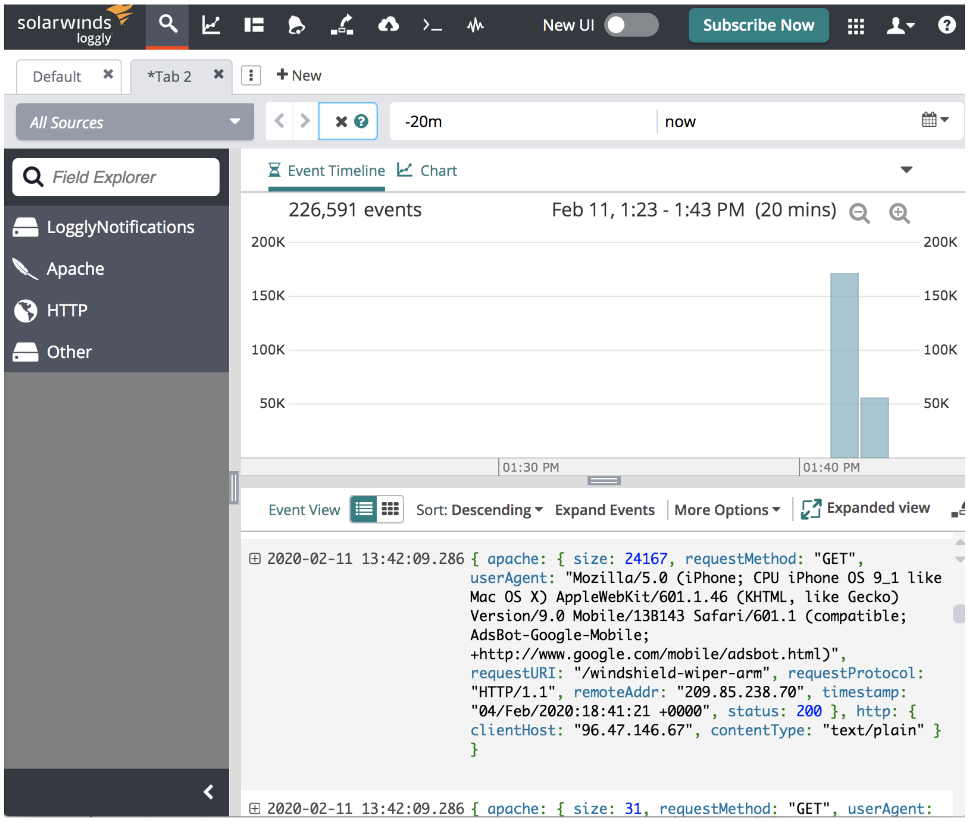
Click source setup (cloud with arrow icon) on the top

**Step 4**

Click Log File and select Browser File Upload

## Step 5

After the file uploads, click Show me my logs link and you should start seeing a list of events and timeline chart

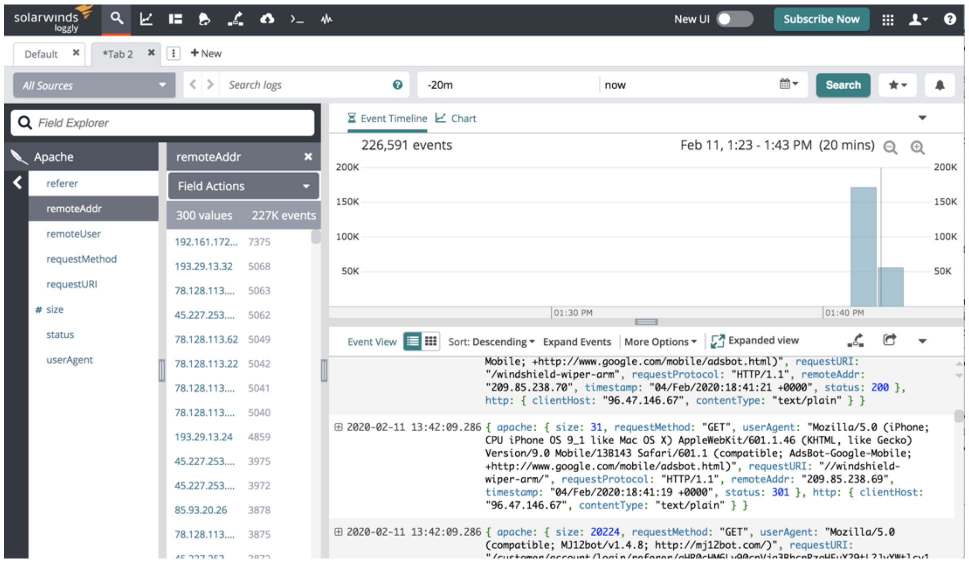


```
⊞ 2020-02-11 13:42:09.286 { apache: { size: 24167, requestMethod: "GET",
                          userAgent: "Mozilla/5.0 (iPhone; CPU iPhone OS 9_1 like
                          Mac OS X) AppleWebKit/601.1.46 (KHTML, like Gecko)
                          Version/9.0 Mobile/13B143 Safari/601.1 (compatible;
                          AdsBot-Google-Mobile;
                          +http://www.google.com/mobile/adsbot.html)",
                          requestURI: "/windshield-wiper-arm", requestProtocol:
                          "HTTP/1.1", remoteAddr: "209.85.238.70", timestamp:
                          "04/Feb/2020:18:41:21 +0000", status: 200 }, http: {
                          clientHost: "96.47.146.67", contentType: "text/plain" }
                          }

⊞ 2020-02-11 13:42:09.286 { apache: { size: 31, requestMethod: "GET", userAgent:
```

## Step 6

In the left menu, Field Explorer, choose Apache and then remoteAddr and you will see the following



| 300 values | 227K events |
|---|---|
| 192.161.172.... | 7375 |
| 193.29.13.32 | 5068 |
| 78.128.113.... | 5063 |
| 45.227.253.... | 5062 |
| 78.128.113.62 | 5049 |
| 78.128.113.22 | 5042 |
| 78.128.113.... | 5041 |
| 78.128.113.... | 5040 |
| 193.29.13.24 | 4859 |
| 45.227.253.... | 3975 |
| 45.227.253.... | 3972 |
| 85.93.20.26 | 3878 |
| 78.128.113.... | 3875 |

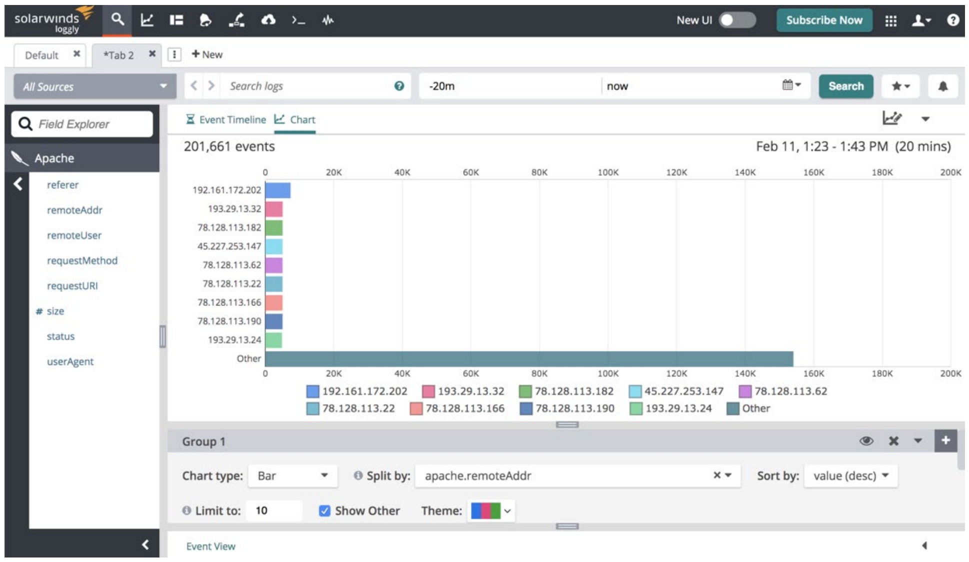
```
                          Mobile; +http://www.google.com/mobile/adsbot.html)", requestURI:
                          "/windshield-wiper-arm", requestProtocol: "HTTP/1.1", remoteAddr:
                          "209.85.238.70", timestamp: "04/Feb/2020:18:41:21 +0000", status: 200 },
                          http: { clientHost: "96.47.146.67", contentType: "text/plain" } }

⊞ 2020-02-11 13:42:09.286 { apache: { size: 31, requestMethod: "GET", userAgent: "Mozilla/5.0 (iPhone;
                          CPU iPhone OS 9_1 like Mac OS X) AppleWebKit/601.1.46 (KHTML, like Gecko)
                          Version/9.0 Mobile/13B143 Safari/601.1 (compatible; AdsBot-Google-Mobile;
                          +http://www.google.com/mobile/adsbot.html)", requestURI: "//windshield-
                          wiper-arm/", requestProtocol: "HTTP/1.1", remoteAddr: "209.85.238.69",
                          timestamp: "04/Feb/2020:18:41:19 +0000", status: 301 }, http: { clientHost:
                          "96.47.146.67", contentType: "text/plain" } }

⊞ 2020-02-11 13:42:09.286 { apache: { size: 20224, requestMethod: "GET", userAgent: "Mozilla/5.0
                          (compatible; MJ12bot/v1.4.8; http://mj12bot.com/)", requestURI:
```

**Step 7**

In the Field Actions menu choose View as Bar Chart and you should see top 10 IP addresses that visited the site over last 20 minutes. This can help identify for example a denial of service attack.



-20m now

201,661 events

Feb 11, 1:23 - 1:43 PM (20 mins)

Chart type: Bar
Split by: apache.remoteAddr
Sort by: value (desc)
Limit to: 10
Show Other ✓
Theme:

## Exercise 3.2.2

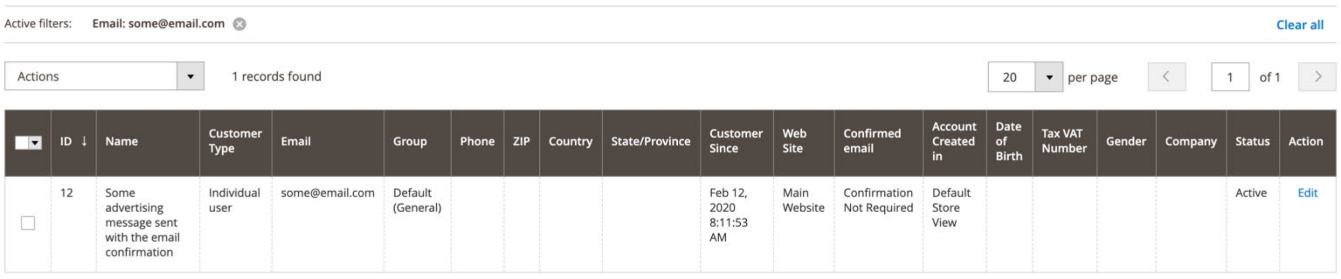
Investigate an automated script attack on the website frontend and protect against it.

1. Identify from which IP address the customer create request was made from (sample access.log file is available in the exercise materials for this investigation);
2. Enable the captcha solution for the customer registration page to prevent frontend customers creation by the automated scripts (complete it on your Magento 2 instance);

While reviewing the customer grid in the Magento admin panel, a business user identified suspicious user signups.

An example of a suspicious customer record is pictured below (pic 2.1).

Current admin locale timezone is set to “Pacific Standard Time (America/Los_Angeles)” (UTC -8).



Active filters: Email: some@email.com    Clear all

Actions    1 records found    20 per page    1 of 1

| | ID ↓ | Name | Customer Type | Email | Group | Phone | ZIP | Country | State/Province | Customer Since | Web Site | Confirmed email | Account Created in | Date of Birth | Tax VAT Number | Gender | Company | Status | Action |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ☐ | 12 | Some advertising message sent with the email confirmation | Individual user | some@email.com | Default (General) | | | | | Feb 12, 2020 8:11:53 AM | Main Website | Confirmation Not Required | Default Store View | | | | | Active | Edit |

Pic. 2.1. Customer Since column has the value “Feb 12, 2020 8:11:53 AM”

**Solution**

**Step 1**

Find out the customer creation date from the customer record info.

1.1 Can be found in the Magento admin panel at Customers > All Customers grid. Find the appropriate customer record and check the “Customer Since” column value (datetime in the current admin locale timezone)

1.2 Can be found in the database in the customer_entity table, created_at field (datetime in UTC timezone).

**Step 2**

Find the IP address information from the access.log for the request which created the account.

1. Customer account is created with a POST request /customer/account/createpost
2. Access.log contains datetime info in the UTC timezone. To find an account creation datetime in a UTC format a conversion must be completed from the admin locale timezone. In this particular case:
   i. `Feb 12, 2020 8:11:53 AM PST  =>  Feb 12, 2020 16:11:53 UTC`
3. Search for the mentioned POST request around the target datetime period in the access.log file. Keep in mind that couple seconds delay is possible between actual record creation in the DB and the web request to be completely processed with the record in the access.log.
4. In this case, the following record can be found in the access.log:
   a. 
   ```
   185.214.164.10 - - [12/Feb/2020:16:11:54 +0000] "POST
   /customer/account/createpost/ HTTP/1.1" 302 31
   "https://www.demostore.com/customer/account/create/" "Opera/9.80 (Windows NT
   6.1; U; Distribution 00) Presto/2.10.229 Version/11.60"
   ```
5. That IP can be added to the WAF blacklist to prevent automatic scripts execution on the website.

**Step 3**

Enable captcha for the customers registration form. Follow steps described in the documentation: https://docs.magento.com/m2/ee/user_guide/stores/security-google-recaptcha.html

## 3.3 Evaluating Extension Providers

### Exercise 3.3.2

Configure use of Magento Coding Standards for module evaluation.

The “Magento2” standards should be available to phpcs. Evaluate a core module to see the sort of output that may be expected. For example:

```
vendor/bin/phpcs --standard=Magento2

vendor/magento/module-cms
```

**Solution**

See documentation page for solution:

https://github.com/magento/magento-coding-standard/

The solution outlined on the README of the MCS repository demonstrates how to install, verify and use the MCS tools on a Magento codebase. Set these up and then run the noted “phpcs” command to evaluate the code in the Magento_Cms module.

# Unit 4. Incident Response

## 4.2 Incident Response Planning

### Exercise 4.2.1

Fill out the IRT Contact List in the worksheet provided.

See Exercise Worksheet

Define your "Rejection/Acceptance criteria" in the worksheet.

See Exercise Worksheet

#### Solution

There is a Microsoft Word Exercise sheet included with this module. When you see this icon, the exercise portion of the module involves switching to the Incident Response Worksheet and filling in the details for your organization. Attempt to make this as realistic as possible since the goal is that you take this worksheet and use it within your organization for Incidence response. If you work at a Magento Partner Agency then choose a single merchant you are familiar with and complete the worksheet for that Merchant. When this course is over, you should create an incident response plan for each merchant since their teams and processes will most likely change.

This is applicable to all exercises of module 4.2.

### Exercise 4.2.2

Fill out the IRT Contact List in the worksheet provided.

See Exercise Worksheet

#### Solution

Try to make your acceptance and rejection criteria as realistic as possible. It is ok if you need to change them based on further discussion with your colleagues in revision.

## Exercise 4.2.3

Complete “Breach and Severity Classification” section in the worksheet

See Exercise Worksheet

**Solution**

Sample Severity classification loosely based on GDPR is included as an example in the worksheet. This is included to help guide your classification in a realistic manner for severity, however, feel free to edit these based on the volume and impact on your merchant.

## Exercise 4.2.4

Add “Immediate Action” activities to the responsibility matrix in the worksheet

See Exercise Worksheet

**Solution**

Fill in the worksheet as applicable to your case.

## Exercise 4.2.5

Add “Establish Approximate Timeline” activities to the responsibility matrix in the worksheet

See Exercise Worksheet

**Solution**

When filling this section of the worksheet in, attempt to use the information you know that is already available to your team for establishing a timeline, but also be aware that you may have access to support from your hosting partner to help identify timelines.

## Exercise 4.2.6

Add “Identify Attack Vector” activities to the responsibility matrix in the worksheet

See Exercise Worksheet

**Solution**

Try to fill in as many possible avenues for determining the attack vector. However recognize that sometimes due to the state of logs it is almost impossible to determine the initial attack vector, or even worse you may determine separate successful initial attacks on your system.

## Exercise 4.2.7

Add “Establish Precise Timeline” activities to the responsibility matrix in the worksheet

See Exercise Worksheet

**Solution**

Fill in the worksheet as applicable to your case.

## Exercise 4.2.8

Add “Remove Persistent Threats” activities to the responsibility matrix in the worksheet

See Exercise Worksheet

**Solution**

Fill in the worksheet as applicable to your case.

## Exercise 4.2.9

Define communications requirements in the worksheet

See Exercise Worksheet

**Solution**

Fill in the worksheet as applicable to your case.