1. **Product extensibility :**

Product extensibility describes how easy it is to expand a product's feature set. An extensible product has been designed from its earliest stages for customization and enhancement.

**Magento extensibility** describes the product’s built-in ability for developers and merchants to routinely extend their storefront’s capabilities as their business grows.

Product extensibility has always been taken into account from Magento’s earliest design stages. Magento 2 uses automatic dependency injection and service contracts to facilitate new implementations of existing functionality.

In software engineering, dependency injection is a design pattern in which an object or function receives other objects or functions that it depends on. A form of inversion of control, dependency injection aims to separate the concerns of constructing objects and using them, leading to loosely coupled programs. The pattern ensures that an object or function which wants to use a given service should not have to know how to construct those services. Instead, the receiving 'client' (object or function) is provided with its dependencies by external code (an 'injector'), which it is not aware of. Dependency injection helps by making implicit dependencies explicit and helps solve the following problems. The Dependency injection design pattern creates an external environment where you can inject dependencies into an object. Thanks to that, you will no longer have to create the objects manually. Namely, as when object A calls object or value B, this means B is a dependency of A.A contract for services is a formal, legally binding agreement before a business and a self-employment individual. It differs between an employment contract – known as a contract of service – which is between an employer and an individual who then becomes employed by the company.

**Injection types :** Injection types used Magento 2 Dependency Injection includes two types: Constructor Injection and Method Injection. You can see the following code snippet to learn more about both of them.

namespace Magento\Backend\Model\Menu; class Builder { /\*\* \* @param \Magento\Backend\Model\Menu\Item\Factory $menuItemFactory \* @param \Magento\Backend\Model\Menu $menu \*/ public function \_\_construct( Magento\Backend\Model\Menu\Item\Factory $menuItemFactory, // Service dependency Magento\Backend\Model\Menu $menu // Service dependency ) { $this->\_itemFactory = $menuItemFactory; $this->\_menu = $menu; }

public function processCommand(\Magento\Backend\Model\Menu\Builder\CommandAbstract $command) // API param

{

// processCommand Code

}

} Constructor injection As the above example, $menuItemFactory and $menu are the dependencies that will be added to an object’s class through the constructor injection. Besides, remember that the constructor injection is required to declare all optional and required of an object.

Method injection About Method Injection, you will use it when an object makes clear a dependency in one of its methods. As if tracking in the referred instance, $command is the dependency passed into the class through the process Command method

Links:

<https://delftswa.gitbooks.io/desosa-2017/content/magento/chapter.html#context-view>

<https://en.wikipedia.org/wiki/Dependency_injection>

<https://www.mageplaza.com/devdocs/magento-2-dependency-injection.html>

<https://www.mageplaza.com/devdocs/magento-2-dependency-injection.html#compiling-dependencies>

-----------------------------------------------------------------------------------------------------------------------------------------

1. **ARCHIETECTURE:**

The Magento architecture comes with Models, Views and Controllers.

**User Request** − The user sends a request to a server in the form of request message where web browsers, search engines, etc. act like clients.

**View** − View represents the data in particular format. It is the user interface which is responsible for displaying the response for user request. It specifies an idea behind the presentation of the model's data to the user. Views are used to reflect "how your data should look like". The must be easy to learn and adapt.

**Controller** - The controller is responsible for responding to user input and perform interactions on the data model objects. It uses models to process the data and send responses back to the view.

**Model** − The model is responsible for managing the data of the application. It contains logic of the data and represents basic data object in the framework. It responds to request from the view and to the instructions from the controller to update itself.

**Database** − Database contains the information which is requested from the user. When the user requests data, view sends requests to the controller, the controller requests from the model and the model fetches the required information from the database and responds to the user.

WSDL − WSDL stands for Web Services Description Language. It is used for describing web services and how to access them.

**Magento-architecture :**

Magento is built on top of the **Zend Framework** , that makes sure the code base will be secure and scalable. There are many reasons for choosing the Zend Framework, but the main one is that Zend provides an object-oriented library of code with a committed company standing behind it.

Using this framework, Magento was built with **3 central tenets** in mind.

**1) Flexibility:**

We believe each solution should be as unique as the business behind it. Magento’s code allows for seamless customizations.

**2) Upgradeable:**

By separating the core code from community and local customizations, Magento can be easily customized without losing the ability to upgrade.

**3) Speed and Security:**

The coding standards used by the developers follow best practices to maximize the efficiency of the software and provide a secure online storefront.

**Does Magento follow MVC architecture :**

Magento’s architecture does follows the well known MVC architecture but it actually does have some of its own additions which do help large scale web developments. Like we all know MVCs (Model, View and Controller) architecture is where you have a set of modules which comes with Models, Views and Controllers to split up your codes and make code management easier and simpler. In the conventional MVC, one would request the controller for a service and the controller would use models to get processed data and put forward the data to the view to give user the response and take another request which would be carried out in the same fashion. Magento’s architecture has added a lot more sub blocks to the above MVC architecture in order to handle bigger e-commerce system which can handle multiple sites / stores from the same back-end. You have controllers and helpers also helpers are module specific.



**Splitting of Views in Magento:**

We can see that Views have been split into three parts. The templates are the plainly html codes usually saved as phtmls with php tags to prints data and do some basic loops and some javascript calls like our usual view would look like. Next comes blocks which is a new concept to MVC. Blocks are simply used to lower the burden on central controller and make different views in a module more independent. This is important these days and any websites these days stands on a number of different blocks and some blocks may be AJAX loaded and provide different services. Therefore, Views have their own controllers to ask or request processed data from Models and provide graphical ‘view’ through templates. Blocks holds all the data and functions that can be called from the view template and Block can have nested blocks. Therefore, our website will have a root block, a header block, a navigation block and contents and footer block therefore they can be nested like that.

**Central Controller :**

The role of the central controller is if every view comes with their own controller which can interact with models and helpers. That is where layout comes into play actually. In most of the definitions I read previously they say that in order to say which blocks goes with which template and in order to define which block is nested in which we need the layout. This is true but there is a little bit more to it. Layouts can only be called by a central controller name, therefore the central controller has the layout which defines the sub controllers (blocks) and the templates it contains. The central controller along with the helper provide service to the overall block set while the individual blocks provide service in email. Layout therefore simply provides a way to tell which is the super block and which are nested and how they are nested.

Magento’s architecture is mainly built on concepts that allow for maximum flexibility and extensibility of its software:

* **OOP Principles**

As a PHP framework, Magento takes advantage of Object-Oriented Programming (OOP) principles that provide simplicity and extensibility. The concepts of data encapsulation and inheritance are especially useful for Magento’s front-end components

* **Modularity**

The concept of modularity lies at the heart of Magento. At the highest level, Magento’s architecture is composed of core components and optional independent modules. These modules are organized by feature and can be used to modify the appearance and behaviour of a webshop without altering other parts of the code.

* **Extensibility**

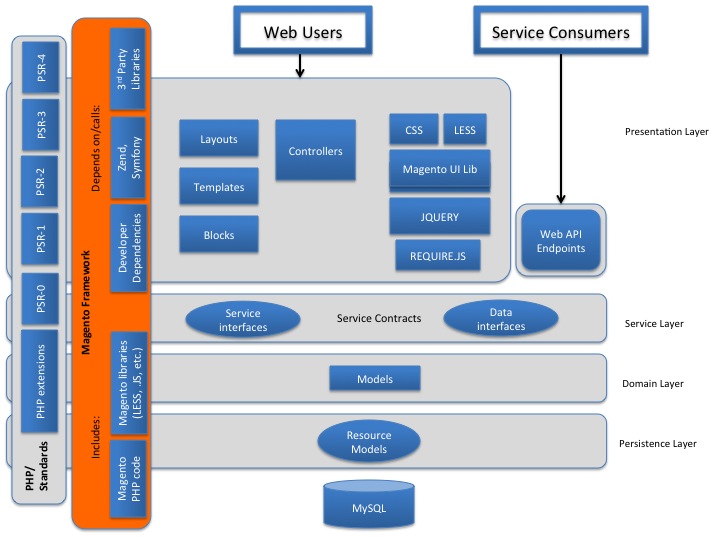
Product extensibility has always been taken into account from Magento’s earliest design stages . Magento 2 uses automatic dependency injection and service contracts to facilitate new implementations of existing functionality .

* **Stack of open-source technologies**

The Magento stack contains many [open-source technologies](http://devdocs.magento.com/guides/v2.0/architecture/tech-stack.html) for deployment and customization of storefronts.

Magento’s core product code has a layered design. For Magento’s customers specifically, a layered architecture provides the benefit of separating presentation logic from business logic. This simplifies the divided customization of store appearance on the one hand and store behaviour on the other. Architectural layers also provide developers a simplified model for the ideal placement of features and code in the system.

The design of Magento can be decomposed into clusters of components with a similar functionality. These clusters, or layers, enable the logic separation of different responsibilities. Figure [5](https://delftswa.gitbooks.io/desosa-2017/content/magento/chapter.html#fig5) illustrates the layered architecture of Magento and shows the components of each layer. The diagram also demonstrates the connections between the four layers and the Magento framework, third party libraries, the supported database, and other technologies.



**The REFERENCES**:

https://www.tutorialspoint.com/magento/magento\_architecture.htm#:~:text=The%20Magento%20architecture%20comes%20with,the%20data%20in%20particular%20format.

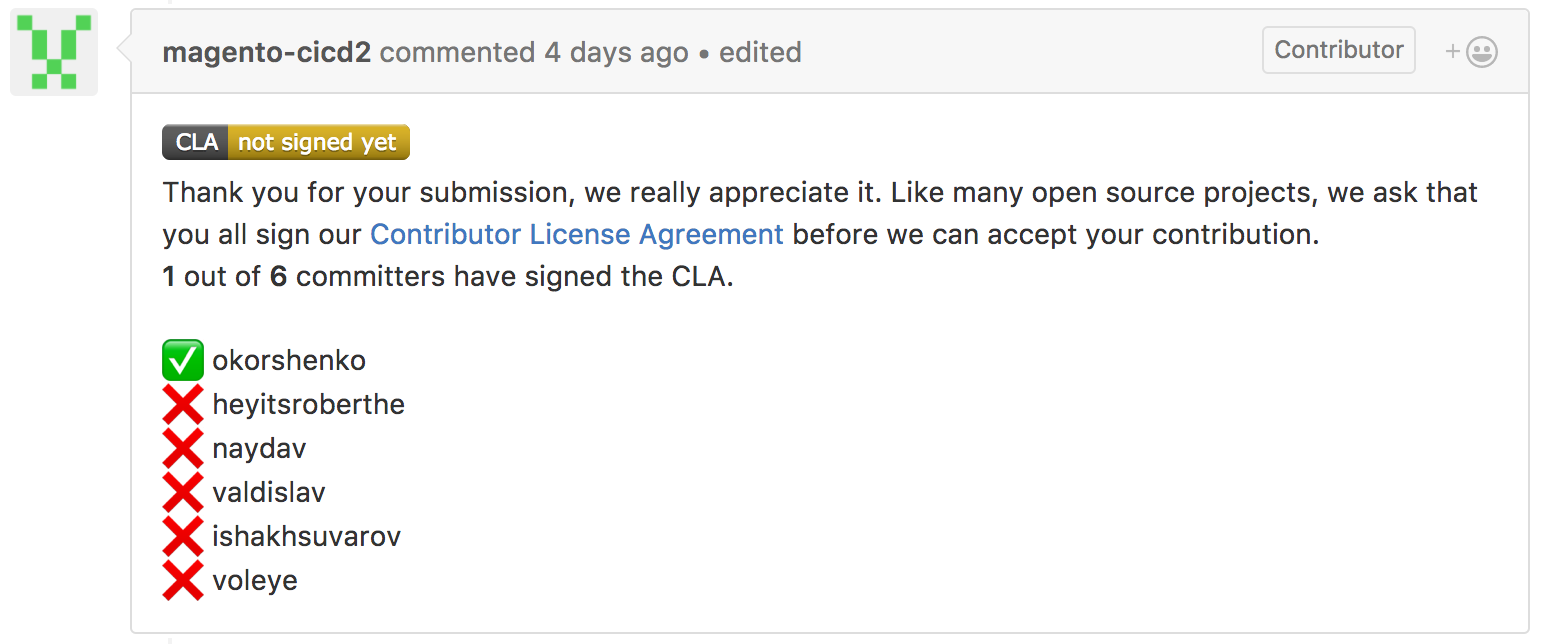
<https://blog.magestore.com/lesson-2-magento-architecture/>

<https://delftswa.gitbooks.io/desosa-2017/content/magento/chapter.html#context-view/>

-----------------------------------------------------------------------------------------------------------------------------------------

1. **Micro Services :**

All contributions are done via the fork and pull model with the integrators reviewing pull-requests on a first-in-first-out basis. One’s code only has the chance of being merged if it adheres to the rules in the developers guide:

* Each code extension needs to adhere to the [coding standards](http://devdocs.magento.com/guides/v2.0/coding-standards/bk-coding-standards.html).
* The Magento project has its own Definition of Done (DoD) defining a set of acceptance criteria that is applied to any changes in the code base. These criteria revolve around readability, sufficient code coverage and solid documentation.
* Each commit needs to pass the automated tests that are in place.Everyone who wants to contribute to Magento’s source code is required to accept the terms of the contributor agreement.
* 

The **Achievements** that are done using micro services include:

Using a Magento microservices architecture leveraging AWS’s Lambda we achieved :

**1)** **Keep infrastructure costs down for our clients**

We were able to keep infrastructure costs for our clients as low as possible. We work with a lot of early-stage eCommerce startups and every dollar is precious. The pay per use model with zero upfront costs helps startups save costs and deploy those resources in building features and responding to customer needs rather than worry about paying their monthly infra bills.

**2) Save Time and Money**

on DevOps tasks like rolling out EC2 instances, setting up auto-scaling, monitoring, looking at budget overruns, and let our DevOps team focus on adding value by Continuous Integration.

**3) High-Level Code Reuse**

Lambda functions take code reuse to a new level — now if we needed to say implement a feature to resize an image or create a pdf from an Invoice, Punchout or any other type of Magento — 3d party system integration we just called an API that we had created for an earlier project. Add to that, the microservices architecture makes this technology agnostic, so I could have a team working on a Magento PHP project called a Lambda function that was created on NodeJS or Go language without worrying about deploying to another server, etc.

**DrawBacks:**

If we look into Magento 1 the micro services were not a the part of it but by the time passes they introduced new versions or different products.In Magento 2, every new feature gets added to the existing application using extensions making it more and more complex and slower. Scaling becomes hard and resource-wasting since everything has to be scaled together. Deployment turns into a nightmare thanks to the 5 million PHP lines of code waiting to be pushed into production every time, di compile, setup, upgrade, static content deploy, cache clean.

**HOW IT CAN BE FIXED :**

In a microservice-oriented architecture, Magento features can be built up from multiple smaller pieces (cart, order, product, category, search, review) that work together but can be developed separately. By defining technology-agnostic communication channels, any language can be used PHP, NodeJS, Python, Rust, GO. This means that developers are not restricted to legacy technologies anymore, and they can freely just develop. If done properly, a microservice should be small enough to be rewritable in about two weeks by a single person.

REFERENCES: <https://www.metabase.com/blog/microservices-considered-harmful>

<https://delftswa.gitbooks.io/desosa-2017/content/magento/chapter.html#context-view/>

1. **Magento Performance Optimization:**

Before looking into the performance what is its definition

1. The process of making something, especially a computer system, work as effectively as possible (Cambridge English Dictionary).
2. Performance optimization is the process of modifying a system to amplify its functionality, thus making it more efficient and effective.

**Optimization using Cache :**

How Magento uses cache to optimize performance? Magento allows full-page cache enabling and disabling facility.

When a user is accessing your store a request is sent to the server. PHP handles this request by performing specific operations and database queries before returning the corresponding HTML to the user. Full-page cache stores that HTML response, so that the next identical request will directly return it skipping all the back-end processing and database queries. This significantly improves the Using full-page cache as part of your Magento optimization efforts can tremendously increase your website speed. Instead of running all the queries for each request, this will create cached versions of your pages and deliver them to the user. Only those pages which are used by all the users i.e. has same data for each user are cached. The pages like cart which have dynamic data or are user-specific are not cached.

**Def of cache**: In computing, a cache is a hardware or software component that stores data so that future requests for that data can be served faster; the data stored in a cache might be the result of an earlier computation or a copy of data stored elsewhere. Correctly using a cache can increase website performance, and a faster website can lead to higher conversion rates.

**CSS/JS Minification:**

Minifying the CSS and JS files is an important element of Magento 2 speed optimization. By minifying them we remove all the spaces, tabs, and newlines in the files. The resulting files will have fewer characters and thus a smaller size, so they will download faster.Magento has this feature built-in and you can enable CSS/JS minification in Admin.

**Using Lazy loading in Magento :**

What is lazy loading?

Lazy loading is the practice of delaying load or initialization of resources or objects until they're actually needed to improve performance and save system resources. Lazy loading keeps the number of loaded elements at once to a minimum, decreasing the page size and speeding up the overall loading.

**Advantages of Lazy Loading**

1) Bandwidth conversation

2) Reduce perceived and real loading time for your pages

3) Save system resource

4) Improve the customer’s shopping experience

Lazy loading in Magento can be enabled by installing extensions.

**Performance using CONTENT DELIVERY NETWORK:**

One of the advanced solutions for improving site loading performance is Content Delivery Network (CDN). The greater the distance between your main server and the customer, the longer they have to wait for your site to load. Because e-commerce is rapidly expanding and customers have less time and patience, your web loading time is an important factor in capturing their attention and thus has a significant impact on your SEO results.

Working of CONTENT DELIVERY NETWORK:

1. Browser requests media - A page from the store opens in the customer’s browser, and the browser requests the media that is specified in the HTML.

2. Request sent to CDN; images found and served - The request is sent first to the CDN. If the CDN has the images in storage, it serves the media files to the customer’s browser.

3. Media not found, request sent to Commerce web server - If the CDN does not have the media files, the request is sent to the Commerce web server. If the media files are found in the file system, the web server sends them to the customer’s browser.

CONTENT DELIVERY NETWORK IN MAGENTO:

In contrast to full-page cache, Content Delivery Network (CDN) is not fully integrated into Magento 2 Store, so you must choose and configure your own CDN.

**Progressive web app (PWA):**

A progressive web app (PWA) is a website that looks and behaves as if it is a mobile app. PWAs are built to take advantage of native mobile device features, without requiring the end user to visit an app store, make a purchase and download software locally.

A progressive web application (PWA), commonly known as a progressive web app, is a type of application software delivered through the web, built using common web technologies including HTML, CSS, JavaScript, and web assembly. It is intended to work on any platform with a standards-compliant browser, including desktop and mobile devices.

**What Is Magento PWA?**

A progressive web app (PWA) roughly means a mix of a website and a native app. It’s a web app that provides a native-like experience with enhanced UI/UX, rapid loading, and discoverability by search engines. And it supports push notifications like native apps. At the same time, as PWAs are basically website shortcuts, memory consumption is minimal.

How Magento PWA Improves the Magento Store:

A Magento store can benefit from the conversion into a PWA in the following ways:

1. The blazing speed and ease of installation can lower the bounce rate and increase mobile conversions.
2. The native-like experience will add to user-friendliness. It will give the store a much-needed leg-up in the overcrowded market and encourage purchasers to return to the website.
3. Other website metrics like session duration, pages visited per session, and cart abandonment rate will also improve.

As PWAs remain websites, you can promote the store in search results, gain backlinks, and drive visitors from other resources.

We can develop a Magento PWA site in following ways

1. Through Magento 2 PWA Extension
2. Through Magento PWA Frameworks
3. Through a Magento PWA Theme

Service workers are specialized JavaScript assets that act as proxies between web browsers and web servers. They aim to improve reliability by providing offline access, as well as boost page performance.

Service workers function work as a network proxy. They intercept all outgoing HTTP requests made by the application and can choose how to respond to them. For example, they can query a local cache and deliver a cached response if one is available

Service workers are a fundamental part of a PWA. They enable fast loading (regardless of the network), offline access, push notifications, and other capabilities.

-----------------------------------------------------------------------------------------------------------------------------------------

**5)Reliability in Magento**

This section describes the reliability, scalability and cloud usage of Magento. Magento uses AWS cloud services.

**Magento cloud** provides following benefits:

- Rapid Development

- Continuous Deployment

- Customizable Environment

- High Scalability

- Enhanced Security

**High Scalability:**

1) the store merchant does not have to plan ahead in terms of needed resources, timing, or anything other than to simply indicate that, “yes,” they want auto-scaling capability

2) that the amount of additional resources employed should not be charged in addition to the client’s plan, as long as their hosting is priced on a

resource-based plan. At Webscale, we charge based on the number of user sessions, regardless of how many resources are used to deliver the traffic. We also do not charge extra for “auto-scaling setup” or other configurations. With Auto Scaling, you can ensure that the number of servers you are using increases seamlessly during demand spikes to maintain performance and decreases automatically during demand lulls to minimize costs. To make sure your Magento shop can handle increased traffic, Auto Scaling is not only convenient but a must-have preparation for you. Auto Scaling ensures your e-commerce shop’s availability, no matter how high the traffic volume is.

When it comes to scalability, Magento becomes highly scalable because of cloud. Because of cloud you can add a lot of audience and cloud supports advanced scalability features such as order archiving, multiple master servers, scalable backend product management, and MySQL customer support.

Some of the Magento scaling tasks include:

1. Installing new services
2. Upscale & downscale resources
3. Server provisioning to avoid resource wastage
4. Auto Scaling solutions
5. Optimizing usage for low costs

**FAULT TOLERANCE**

Fault tolerance refers to the ability of a system (computer, network, cloud cluster, etc.) to continue operating without interruption when one or more of its components fail. The objective of creating a fault-tolerant system is to prevent disruptions arising from a single point of failure, ensuring the high availability and business continuity of mission-critical applications or systems.

**Continuous Data Backup**

Magento can do continues data backup for it users. Data can be lost due to human error or a cyber attack. Human error can be like the user not saves the work timely or closes without saving. Sometimes if he is aware of the saving, electricity failure can lead to data loss so data saving continuously is mandatory. Sometime hackers black mail us by getting our sensitive data. If there is backup available there is no need to worry or pay extra money get your own data back. The backup tasks can be managed by the Magento support staff.

Routine backups are done when you use the Magento hosting support. For your Magento system and databases, they create backups. In any unavoidable circumstances data can be retrieved quickly.

**Load Balancing**

As strain increases on a website or business application, eventually, a single server cannot support the full workload. To meet demand, organizations spread the workload

over multiple servers. Called "load balancing," this practice prevents a single server from becoming overworked, which could cause it to slow down, drop requests, and even crash. Magento Support team provides the services of Load Balancing. With this service they take care of all the aspects through which a website can become un accessible.

Magento does it by :

- Operating system patches

- Incremental and full backups

- Continues support on Application level

References

<https://www.mgt-commerce.com/astatic/assets/images/article/2021/10/bdad2e62730631f5859c19b5c92174b3.png/>

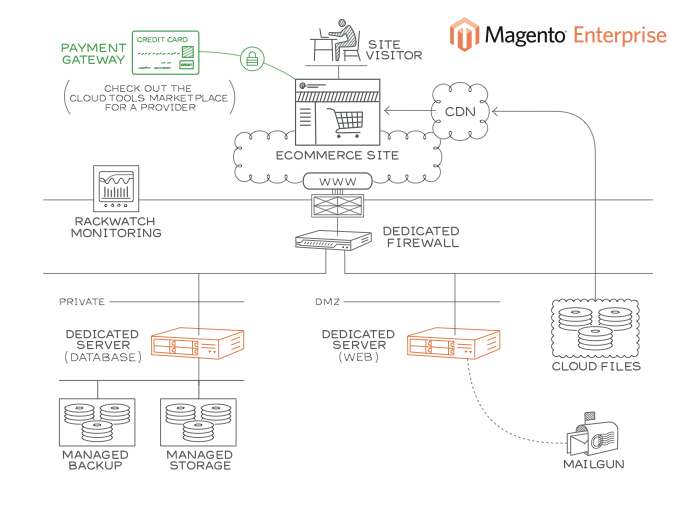
-----------------------------------------------------------------------------------------------------------------------------------------

**6)Security Magento**

This section describes the different technologies used by Magento to maintain security of Magento . Moreover this section deals with description of different compliances of specific privacy legislation followed by Magento.

According to TrustWave’s Global Security Report 2016, "Magento was the ecommerce target of choice for hackers, with Magento installations accounting for 85% of compromised ecommerce systems" . This was mostly caused by weaknesses in the server environment. A Magento infrastructure contains several potential gateways for hackers. Figure for illustration.

rustWave's report also notes that most software on Magento users’ servers were outdated and not fully patched. Therefore, securing the server infrastructure is the most critical aspect of securing a Magento website. In order to do this properly the Magento team published a [security guide](https://magento.com/security/best-practices/security-best-practices) which describes the best practices for server administrators to setup the third-party software correctly and securely.



Reference : <https://www.adobe.com/content/dam/cc/en/trust-center/ungated/whitepapers/experience-cloud/adobe-commerce-best-practices-guide.pdf/>

<https://delftswa.gitbooks.io/desosa-2017/content/magento/chapter.html#context-view/>

**Documentation**

Magento applies a Content Security policy (CSP). With this policy it provides an additional layer of security against different data injection attacks. This common attack vector works by injecting malicious content that falsely claims to originate from the website. Once this content is loaded it begins the transfer of sensitive data.

In all this scenario, CSP is useful when it tells the browser to which resource source it should trust and which should be blocked. Using carefully defined policies, CSP can restrict browser content to allow only whitelisted resources to appear.

**CSP has two modes:**

1. report-only
2. restrict mode

**Report-only:**

In report only mode which is by default mode in Magento. It reports all the resources which violated the CSP. In this mode it does not enforce the compliance with CSP. Once all the valid resources are whitelisted then Magento can apply or enforce CSP(restrict mode).

**Restrict mode:**

In restrict mode, the browser is instructed to enforce all content policies and limit publication to whitelisted resources.

* **Prevention of cross-site scripting (XSS) attacks**:

The Magento Framework follows protocols that manage the escape of data in output. These protocols allow clients to escape output for HTML pages (HTML, JSON, and JavaScript) and email. More information on measures against XSS attacks can be found in the [Magento documentation](http://devdocs.magento.com/guides/v2.0/frontend-dev-guide/templates/template-security.html).

* **Flexible file system ownership and permissions**: Since Magento 2.0.6., file system permissions are held in certain files that are writable in a development environment and read-only in a production environment. These permissions (particularly for production) can be further restricted using a umask, as explained in the [Magento documentation](http://devdocs.magento.com/guides/v2.0/install-gde/prereq/file-sys-perms-over.html).
* **Use of non-default Magento Admin URL**: To prevent large-scale attacks that use automated password guessing and target default admin URL's like admin or backend, Magento creates a random Admin URI when you install the product. This URI can be changed through the provided CLI. More information can be found in the [Magento documentation](http://devdocs.magento.com/guides/v2.0/install-gde/install/cli/install-cli-adminurl.html).
* **Prevention from ClickJacking attacks:**

ClickJacking is a from of attack in which hackers display a dummy web page to users . This can cause users to unwittingly download malware,visit malicious web pages, provide credentials or sensitive information, transfer money, or purchase products online. Magento protects it users from ClickJacking byadopting the mechanism of X-Frame Option.X-Frame Options becomes the part of HTTP header and it directs the browser either to load the webpage in frame or not.

* **Enhanced Password Management :**

Magento has improved the hashing algorithms it uses to store the passwords. Magento now uses the Argon2ID13 algorithms to store Password. Argon2ID13 is an algorithm for storing password

- It resists the GPU cracking attacks

- Argon2i is optimized to resist side-channel attacks

References:

[Security Documentation of Magento](https://experienceleague.adobe.com/docs/commerce-operations/security-and-compliance/overview.html)/

**Compliance**

Magento follows two Compliances

- California Consumer Privacy Act (CCPA)

- General Data Protection Regulation (GDRP)

1) CCPA

The California Consumer Privacy Act is a state statute intended to enhance privacy rights and consumer protection for residents of California, United States.

You can read more about CCPA from here:

[CCPA docs](<https://experienceleague.adobe.com/docs/commerce-operations/security-and-compliance/privacy/ccpa.html?lang=en)/>

2)GDRP

The General Data Protection Regulation is a regulation in EU law on data protection and privacy in the European Union and the European Economic Area

[GDRP docs](<https://experienceleague.adobe.com/docs/commerce-operations/security-and-compliance/privacy/gdpr.html?lang=en)\>

### Magento Vulnerabilities

Based on the published patches and compared to competitors, Magento is quite secure According to CVE Details, an online security vulnerability data source, there was only one vulnerability reported in 2016, five in 2015 and two so far in 2017. Figure for illustration

