ENCHANTMENT OF MAGENTO CMS SECURITY

Nikola Pavlović*,
Marko Šarac,
Saša Adamović,
Miloš Mravik

Singidunum University,
Belgrade, Serbia

Abstract:
Magento is a CMS (Content Management System) platform used to create e-commerce websites and online stores. With it, users can fully manage their online store, display specific products, provide customers with different methods of payment and delivery, as well as many other options. Magento uses an authentication system based on the knowledge of the user name and password. In addition to increasing this problem, there is no mechanism that prevents an attacker from brute forcing passwords. The proposed solution is replacing default Magento authentication with OAuth.

Keywords:
Magento, OAuth, WordPress.

1. INTRODUCTION

E-commerce refers to the buying and selling of goods or services using the internet. One of the most popular CMS platforms for E-commerce is Magento. Magento is open-source platform written in PHP. Thanks to its great scalability, flexibility and optimization Magento has become a tool that is most used by developers, designers and web agencies to create E-commerce websites.

One of the biggest problems of Magento CMS is security. To keep Admin Dashboard and entire back-end secure most of the time just changing admin dashboard and back-end URL is not enough. This is because default Controllers still stays at same URL and anyone can brute force them.

The proposed solution for this problem is to replace default Magento password-based authentication with OAuth and add Captcha to prevent brute forcing open controllers. Furthermore, the generic scheme of the suggested solution and its implementation will be explained.

2. MAGENTO SECURITY

In Hivemind's E-commerce Survey report Magento holds 24.6% after studying 56,793 E-commerce websites inside the Alexa Top 1 Million. It is clear that Magento is the most popular CMS.

Correspondence:
Nikola Pavlović

e-mail:
nikola.pavlovic.141@singimail.rs

DOI: 10.15308/Sinteza-2019-223-228

E-commerce websites are attractive targets to hackers because of payment and personal information required to properly use them. A compromised web store can have long-term consequences for both customer and owner. Owner can suffer lawsuits, damage to their reputation, revoked privileges with financial institutions, higher processing fees. Customer can suffer financial loss and identity theft.

Trust plays an important part when one has an online store. To improve Magento security it is important to have latest version of Magento implemented.

As of July of 2018. Google started to force HTTPS. This is very important because no store owners want to get their data being intercepted. It is also one of the key elements in making store compliant with the PCI data security standard and in securing online transactions. SSL certification can be obtained from Let's Encrypt. It will also help in becoming PCI compliant.

The most popular way for hackers to get access to server is that by guessing FTP password. Hosting providers nowadays always recommend using SFTP (Secured File Transfer Protocol) which uses private key for decryption or authenticating a user.

Disabling directory indexing is another way to increase security of Magento site.

E-mails used for Magento store needs to be protected with two-factor authentication and not publicly known.

Magento developers recommend using password with uppercase, numbers and special characters.

Another alternative for security is using static IP addresses and black listing all that shouldn't have access to admin dashboard.

The most popular two factor authentication method is Google Authenticator. With Google Authenticator you also need to know two-factor code. First, install application on your smart phone. In order to add new verification code into your Google Authenticator mobile application, you just need to scan QR code of the desired website (the best way) which you want to add. Each two-factor code changes every 30 seconds meaning every time you access the site this code will be different (the same way when you enter CAPTCHA incorrectly). You can use Google Authenticator only on sites who supports Google Authenticator. This application is most commonly used on sites which primary job is to buy or sell goods online.

3. STATE OF WORDPRESS SECURITY

While Magento is built to be enterprise level E-commerce platform, WordPress is built to be more developer friendly. This comes with community solutions for many security problems.

It is important to mention that Wordpress comes with Captcha support for admin dashboard out of box.

There are many community plugins [1] that improve Wordpress security but most important is iThemes Security.

iThemes Security (formerly Better WP Security) adds two factor authentification using Google Authenticator. Out of box generate stronger passwords and adds expiration date so users can be forced to update them. All users actions are logged. Forces usage of SSL. Prevents brute force attacks by banning hosts and users with too many invalid login attempts. Detects and blocks numerous attacks to your filesystem and database.

Magento is ideal for building online business and sales. On the other hand WordPress revolves around marketing and digital publishing. However, WordPress can become a robust online store with the help of plugins such as WooCommerce. Big companies choose Magento as default platform for online sales while smaller WordPress (WooCommerce).

4. STATE OF THE ART

Authors [2] suggest using the OAuth to separate the role of the Client from the resource owner. In OAuth, the client request access to resources owned by the resource owner and is issued a different set of credentials than those of the resource owner. The client obtains access token by the authorization server and then uses is to access the protected resources hosted by the resource owner. Authors also suggest using OAuth because it defines four roles: resource owner, an entity capable of granting access to a protected resource, resource server, the server hosting protected resource, client, an application making request to protected resources and authorization server, the server issuing access tokens.

In the paper [3] the authors researched some of the most popular web security exploits such as SQL injection and XSS. Their suggestion to improve Web Application security is to avoid unsafe APIs, always sanitize user inputs, apply strong cryptography and security protocols. One of the most important suggestion is to follow the principle of least privilege.

Authors [4] review existing CAPTCHA schemes to protect various Web services. Authors explains that text-based CAPTCHAs have become sufficiently hard for humans to solve and thus their usability has decreased at least for an ordinary user. Often ordinary users fail to solve hard text-based CAPTCHAs in their first attempt. Also breaking CAPTCHA challenge is difficult and it is hard to find 100% success rate. Authors have implied that several CAPTCHA implementations have been broken and provided to be inefficient. With the requirement to make web more user-friendly new CAPTCHA techniques are designed. They require less server processing and offer improved security control against bots.

The popularity of content management software (CMS) is growing vastly. The authors of the paper [5] have implemented tool SAISAN to conduct examination of the 176 WordPress websites. From their research 32% of the WordPress websites are on version 4.7.0 or 4.7.1 and have content injection vulnerability.

Millions of users use Google to log in to websites supporting OAuth 2.0 or OpenID Connect. The authors [6] used OAuthGuard to survey 1000 most popular websites that supports Google-sign in. From the research they have concluded that of the 137 sites in our study that employ Google Sign-in, 69 were found to suffer from at least one serious vulnerability

Authors [7] did analysis of the OAuth 2.0. The aim of their security analysis is establishing strong authorization, authentication and session integrity guarantees. They have covered authorization code grant, implicit grant, resource owner password credentials grant, and the client credentials grant. Four security breaks have been discovered. Authors have proposed solutions to patch security exploits.

5. MAGENTO SECURITY GENERAL GUIDE

The aim of this chapter is to describe simply main aspects that must be considered when securing Magento E-commerce platform.

Security testing is a major means for assuring software security [8]. Most popular way to test security exploits in Magento is by using online tool Magereport.

This web application will generate key that must be placed in content of the website. After placing the key scanner can be run and report for missing patches and possible vulnerabilities will be generated. Magereport do passive checks in read only mode. Also, Magereport tells only what is wrong, not how to exploit it.

Before starting with best practices to improve security of Magento CMS, latest security patches must be applied. Magento patches must be applied in order. Applying them out of order can break Magento since they are directly changing core code.

Magento patches can be downloaded from Magento Security Center as it is only verified source.

Most popular practice is to change path for the admin panel. In most cases access to admin panel can be granted by visiting “example.com/admin”. To prevent this, it is possible to change “/admin” to desired word or code. To change path, open “/app/etc/local.xml”, find <![CDATA[admin]]> and replace admin with desired word or code.

Set correct permissions on Magento Core files. Core Magento and directory files should be set to read only, including “app/etc/local.xml” files.

Setting access permissions for the various files in the Magento application is one of the most important security measures that should be applied in a production environment. Setting proper permissions cuts down the potential damage that an attacker might cause through some vulnerability or exploit in the system. It is also advisable to set privileges for reading and execution as shown below:

```
find . -type f -exec chmod 400 {} \;
find . -type d -exec chmod 500 {} \;
find var/ -type f -exec chmod 600 {} \;
find media/ -type f -exec chmod 600 {} \;
find var/ -type d -exec chmod 700 {} \;
find media/ -type d -exec chmod 700 {} \;
```

Another most popular way to protect Magento administrator dashboard is limiting number of allowed IP addresses that can access it. This can be achieved by changing “.htaccess” file. The following example would redirect to the main Magento address all requests received by the Apache server from any IP other than 192.168.56.102 when the URL contains the string “admin”.

```
<IfModule mod_rewrite.c>
   RewriteEngine On
   RewriteCond %{REMOTE_ADDR}
!^192\.168\.56\.102
    RewriteCond %{THE_REQUEST} ^.*
(admin).* [NC]
    RewriteRule (.*) /
</IfModule>
```

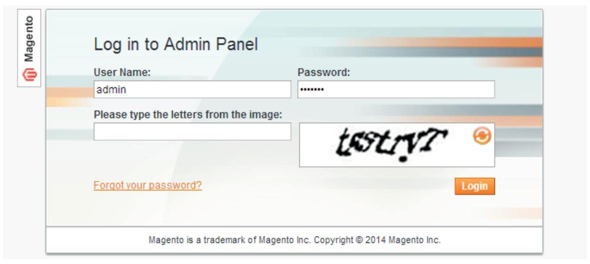
On nginx servers following example can be applied:

```
# nginx configuration

location / {
  if ($remote_addr !~
"^192\.168\.56\.102"){
    rewrite ^(.*)$ / ;
  }
}
```

6. PROPOSED SOLUTION

In this chapter, a developed solution will be presented from theory to practical implementation. Our solution has two different ways of implementation. First is adding CAPTCHA to admin dashboard login form to prevent password guessing.



admin

Magento is a trademark of Magento Inc. Copyright © 2014 Magento Inc.

Figure 1. Adding CAPTCHA to admin login form

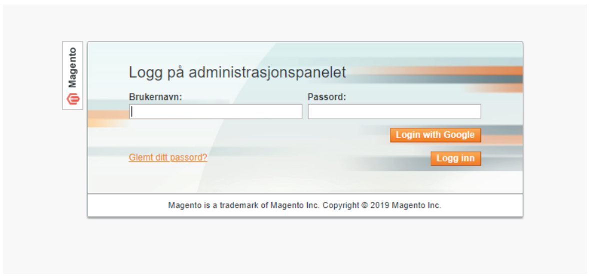
Using CAPTCHA is great from client-side point of view, but it is not perfect. To further improve security of store account lockdown is added. Account lockdown is very effective deterrent against brute force attacks. Account lockdown functionality is independent of the client side completely.

Since it is unrealistic for a human to send two or more requests within one second, adding limit the frequency of accepted connections is another techique to futher improve security and performance of Magento store. Next command can be added to iptables to disable multiple accepting multiple requests:

```
iptables -I INPUT -p tcp --dport 22 -i eth0
-m state --state NEW -m recent --set

iptables -I INPUT -p tcp --dport 22 -i eth0
-m state --state NEW -m recent --update
--seconds 60 --hitcount 4 -j DROP
```

Second implementation of the proposed solution is using two step authentification and OAuth 2.0 for Web Server Applications. OAuth 2.0 is the industry-standard protocol for authorization. Every E-commerce store owner uses Google mail and has Google account. Since Google account security natively supports two step authentication using phone number or another mail, adding this to Magento store is most secure approach.



Magento is a trademark of Magento Inc. Copyright © 2019 Magento Inc.

Figure 2. Using Google OAuth 2.0 to sign up

The next chapter will present the implementation of the proposed solution. The solution itself is developed PHP programming language in Zend Framework.

7. REALISATION OF THE GIVEN SOLUTION

In this chapter, the realisation of the given idea will be shown. On figure 3 is displayed how Magento store authenticate user based on his credentials using OAuth 2.0 Google API.

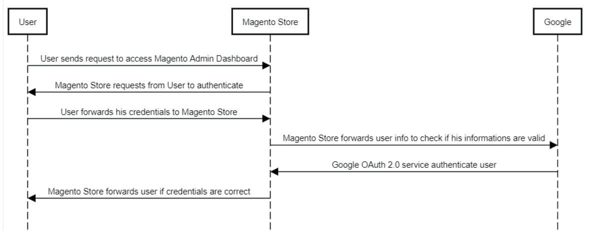
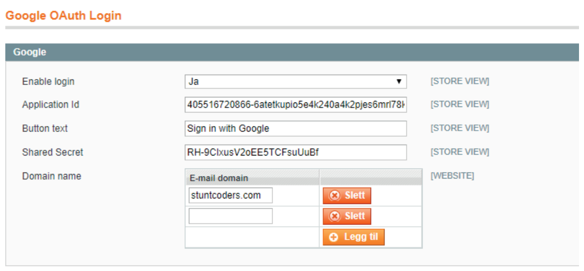

User
Magento Store
Google
User sends request to access Magento Admin Dashboard
Magento Store requests from User to authenticate
User forwards his credentials to Magento Store
Magento Store forwards user info to check if his informations are valid
Google OAuth 2.0 service authenticate user
Magento Store forwards user if credentials are correct

Figure 3. Authentication using OAuth 2.0

When user access Admin Login form Magento will request from him to authenticate. To successfully authenticate user will forward his Google account credentials to Magento store. After getting user credentials Magento will request from Google Authorization Service if user is valid. If user is valid Google Authorization Service will respond with user information. Our solution uses domain name as check if user is valid after getting information from Google Authorization Service. This feature can be improved further by using full email address. We're using domain name since most E-commerce owners have custom domain and they want that all their employees have access to Magento Admin dashboard. Application Id and Shared secret is what links Magento Store with Google OAuth 2.0 API.



Google OAuth Login

Google

Enable login	Ja ▾	[STORE VIEW]
Application Id	405516720866-6atetkupio5e4k240a4k2pjes6mrl78k	[STORE VIEW]
Button text	Sign in with Google	[STORE VIEW]
Shared Secret	RH-9ClxusV2oEE5TCFsuUuBf	[STORE VIEW]
Domain name	E-mail domain: stuntcoders.com	[WEBSITE]

Figure 4. Configuration for OAuth 2.0

On figure 4. Is displayed configuration for OAuth 2.0. To successfully use OAuth 2.0 on Magento Application id generated from Google OAuth API. Application Id is shared to public. On the other hand, Shared Secret must be protected and never shared to public since it is used by Google OAuth 2.0 API as verification key.

On figure 5 is displayed how Magento store authenticate user based on his credentials and Captcha guess.

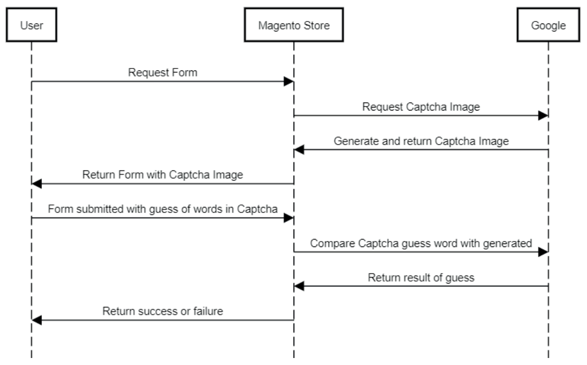

User
Magento Store
Google
Request Form
Request Captcha Image
Generate and return Captcha Image
Return Form with Captcha Image
Form submitted with guess of words in Captcha
Compare Captcha guess word with generated
Return result of guess
Return success or failure

Figure 5. Authentication using Captcha

When user visits Admin dashboard login page Magento will require from him to authenticate. By using Captcha on this form, we have added one additional step before serving Admin Login Form. Before generating HTML and displaying it to user, Magento will send request to Google Captcha API to generate challenge for user. After challenge is generated Google Captcha API will respond to Magento with challenge. After getting challenge Magento will display Form to user. After user guess challenge and submit form. Magento will forward challenge result to Google Captcha API. Google will compare result with generated and respond to Magento. Magento will then then validate username and password if Google response to challenge is positive.

8. CONCLUSION

The goal of this paper is to enchantment of Magento CMS security. By using OAuth 2.0 and Captcha we have removed possible password guessing, SQL injection and XSS attacks. Proposed solution has improved Magento performance since authentication is moved to third trusted party, in our case Google Authorization Server. By removing need to type email and password to access Magento Admin dashboard client experience is drastically improved. Also, by removing need to have hard guessed password to access store Administrator will never have to worry about losing access to store or being locked out of it.

By using Captcha, we have overcome Magento biggest problem of password guessing. This solution as well have improved Magento performance and decreased load on MySQL database since all the work is done by Google Authorization Server.

The implemented solution must be based on the trusted third party, i.e. on the authentication server. The benefit of this authentication method is to achieve high level security of user credentials. The disadvantages are if the authentication server is not active, Magento Admin dashboard is inaccessible, due to the inability of the Administrator to successfully authenticate.

It is important to mention that using OAuth 2.0 is only secure if the authorization code is not used more then once. If it is used more than once server must deny and revoke all previously issued tokens based on that authorization code. Also access token should never be passed as URI parameter.

REFERENCES

[1] H. Huang, Z. Zhang, H. Cheng and S. W. Shieh, "Web Application Security: Threats, Countermeasures, and Pitfalls, " in Computer, vol. 50, no. 6, pp. 81-85, 2017. doi: 10.1109/MC.2017.183

[2] M. Jones and D. Hardt, “The OAuth 2.0 Authorization Framework”, *RFC 6750, 2012.*

[3] M. T. Banday and N. A. Shah, “A Study of CAPTCHAs for Securing Web Services”, *International Journal of Secure Digital Information Age (IJSDIA)*, 2009.

[4] J. Ruohonen, “A Demand-Side Viewpoint to Software Vulnerabilities in WordPress Plugins”, *CoRR abs/1812.05293*, 2018.

[5] M. Hassan, K. Sarker, S. Biswas and H. Sharif, “Detection Of Wordpress Content Injection Vulnerability”, *International Journal on Cybernetics & Informatics (IJCI)*, 2017.

[6] W. Li, C. J. Mitchel and T. Chen, “OAuthGuard: Protecting User Security and Privacy with OAuth 2.0 and OpenID Connect”, *CoRR abs/1901.08960,* 2019.

[7] D. Fett, R. Küsters, “A Comprehensive Formal Security Analysis of OAuth 2.0”, *ACM Conference on Computer and Communications Security*, 2016.

[8] L. Thomas, W. Xu and D. Xu, "Mutation Analysis of Magento for Evaluating Threat Model-Based Security Testing", *Annual Computer Software and Applications Conference Workshops*, 2011.