NIST SPECIAL PUBLICATION 1800-17C

Multifactor Authentication for E-Commerce

Risk-Based, FIDO Universal Second Factor Implementations for Purchasers

Volume C:
How-To Guides

William Newhouse
Information Technology Laboratory
National Institute of Standards and Technology

Brian Johnson
Sarah Kinling
Jason Kuruvilla
Blaine Mulugeta
Kenneth Sandlin
The MITRE Corporation
McLean, Virginia

July 2019

This publication is available free of charge from https://doi.org/10.6028/NIST.SP.1800-17

The first draft of this publication is available free of charge from
https://www.nccoe.nist.gov/sites/default/files/library/sp1800/cr-mfa-nist-sp1800-17.pdf

National Institute of Standards and Technology
U.S. Department of Commerce

NATIONAL CYBERSECURITY CENTER OF EXCELLENCE

DISCLAIMER

National Institute of Standards and Technology Special Publication 1800-17C, Natl. Inst. Stand. Technol. Spec. Publ. 1800-17C, 166 pages, (July 2019), CODEN: NSPUE2

FEEDBACK

As a public-private partnership, we are always seeking feedback on our practice guides. We are particularly interested in seeing how businesses apply NCCoE reference designs in the real world. If you have implemented the reference design, or have a question about applying it in your environment, please email us at consumer-nccoe@nist.gov.

National Cybersecurity Center of Excellence
National Institute of Standards and Technology
100 Bureau Drive
Mailstop 2002
Gaithersburg, MD 20899
Email: nccoe@nist.gov

NATIONAL CYBERSECURITY CENTER OF EXCELLENCE

NIST CYBERSECURITY PRACTICE GUIDES

ABSTRACT

As retailers in the United States have adopted chip-and-signature and chip-and-PIN (personal identification number) point-of-sale (POS) security measures, there have been increases in fraudulent online card-not-present electronic commerce (e-commerce) transactions. The risk of increased fraudulent online shopping became more widely known following the adoption of chip-and-PIN technology that increased security at the POS in Europe.

The NCCoE at NIST built a laboratory environment to explore methods to implement multifactor authentication (MFA) for online retail environments for the consumer and the e-commerce platform

administrator. The NCCoE also implemented logging and reporting to display authentication-related system activity.

This NIST Cybersecurity Practice Guide demonstrates to online retailers that it is possible to implement open standards-based technologies to enable Universal Second Factor (U2F) authentication at the time of purchase when risk thresholds are exceeded.

The example implementations outlined in this guide encourage online retailers to adopt effective MFA implementations by using standard components and custom applications that are composed of open-source and commercially available components.

KEYWORDS

electronic commerce (e-commerce) security; internet shopping security; multifactor authentication (MFA)

ACKNOWLEDGMENTS

We are grateful to the following individuals for their generous contributions of expertise and time.

Name	Organization
Greg Dicovitsky	RSA
Leonardo Andrade	RSA
Adam Cohn	Splunk
Arshad Noor	StrongKey
Kamil Kreiser	TokenOne
Derek Hanson	Yubico
David Coley	Dominion Tea
Brian Abe	The MITRE Corporation
Lorrayne Auld	The MITRE Corporation

Name	Organization
Lura Danley	The MITRE Corporation
Sallie Edwards	The MITRE Corporation
Charles Jones, Jr.	The MITRE Corporation
Joshua Klosterman	The MITRE Corporation
Jay Vora	The MITRE Corporation
Mary Yang	The MITRE Corporation

The Technology Partners/Collaborators who participated in this build submitted their capabilities in response to a notice in the Federal Register. Respondents with relevant capabilities or product components were invited to sign a Cooperative Research and Development Agreement (CRADA) with NIST, allowing them to participate in a consortium to build these example implementations. We worked with:

Technology Partner/Collaborator	Build Involvement
RSA	RSA Adaptive Authentication (Cloud) Version 13.1
Splunk	• Splunk Enterprise Version 6.6.1 • Splunk DB Connect Version 3.1.2 • Splunk Universal Forwarder Version 7.0.1
StrongKey	• StrongKey CryptoEngine Version 2.0 Open Source Fast IDentity Online (FIDO) U2F server • MagentoFIDO (magfido) 1st Edition Module
TokenOne	TokenOne cloud-based Authentication Version 2.8.5
Yubico	Yubico YubiKey NEO Security Key

Contents

List of Figures

List of Tables

1 Introduction

The following volume of this guide shows information technology (IT) professionals and security engineers how we implemented the two example implementations. We cover all of the products employed in these reference designs. We do not re-create the product manufacturers' documentation, which is presumed to be widely available. Rather, this volume shows how we incorporated the products together in our environment.

Note: These are not comprehensive tutorials. There are many possible service and security configurations for these products that are out of scope for these reference designs.

1.1 Practice Guide Structure

This National Institute of Standards and Technology (NIST) Cybersecurity Practice Guide demonstrates standards-based reference designs and provides retailers with the information they need to replicate the multifactor authentication (MFA) for electronic commerce (e-commerce) example implementations. These reference designs are modular and can be deployed in whole or in parts.

This guide contains three volumes:

- NIST SP 1800-17A: *Executive Summary*
- NIST SP 1800-17B: *Approach, Architecture, and Security Characteristics* – what we built and why
- NIST SP 1800-17C: *How-To Guides* – instructions for building the example implementations **(you are here)**

Depending on your role in your organization, you might use this guide in different ways:

Business decision makers, including chief security and technology officers, will be interested in the *Executive Summary*, *NIST SP 1800-17A*, which describes the following topics:

- challenges that enterprises face in implementing MFA to reduce online fraud
- example implementations built at the NCCoE
- benefits of adopting one or more of these example implementations

Technology or security program managers who are concerned with how to identify, understand, assess, and mitigate risk will be interested in *NIST SP 1800-17B*, which describes what we did and why. The following sections of Volume B will be of particular interest:

- Section 3.4, Risk Assessment, provides a description of the risk analysis we performed.
- Appendix A, Mapping to Cybersecurity Framework, maps NIST and consensus security references to the Cybersecurity Framework Subcategories that are addressed in this practice guide. Additionally, work roles in NIST SP 800-181, *National Initiative for Cybersecurity Education*

(NICE) Cybersecurity Workforce Framework that perform the tasks necessary to implement those cybersecurity Functions and Subcategories were identified.

You might share the *Executive Summary, NIST SP 1800-17A*, with your leadership team members to help them understand the importance of adopting standards-based solutions when implementing MFA that can increase assurance of who is using the purchaser's credit card and account information.

IT professionals who want to implement approaches like these will find the whole practice guide useful. You can use the How-To portion of the guide, *NIST SP 1800-17C*, to replicate all or parts of the build created in our lab. This How-To portion of the guide provides specific product installation, configuration, and integration instructions for deploying the example implementations. We do not recreate the product manufacturers' documentation, which is generally widely available. Rather, we show how we incorporated the products together in our environment to create example implementations.

This guide assumes that IT professionals have experience implementing security products within the enterprise. While we have used a suite of commercial products to address this challenge, this guide does not endorse these particular products. Your organization can adopt these example implementations or one that adheres to these guidelines in whole, or you can use this guide as a starting point for tailoring and implementing parts of these e-commerce fraud-reducing capabilities. Your organization's security experts should identify the products that will best integrate with your existing tools and IT system infrastructure. We hope that you will seek products that are congruent with applicable standards and best practices. Volume B, Section 3.5, Technologies, lists the products that we used and maps them to the cybersecurity controls provided by the reference implementations.

1.2 Example Builds Overview

The NCCoE at NIST built two example laboratory environments to explore MFA options available to online retailers, which are described in this section.

1.2.1 Usage Scenarios

The example implementations fulfill the use cases of a returning purchaser with established login account credentials with the retailer, and who possesses a Fast IDentity Online (FIDO) Universal Second Factor (U2F) authenticator [1], [2]. The purchaser's U2F authenticator is used when the retailer system requests additional authentication. This gives the retailer additional assurance that the purchaser is a returning customer when the checkout process occurs in circumstances that exceed the retailer's risk thresholds. In these NCCoE reference architectures, the risk thresholds that initiate MFA requests are based on the total cost of the shopping-cart transaction or upon input received from the risk engine.

The NCCoE worked with members of the NCCoE Retail Community of Interest to develop a set of use case scenarios to help design and test the reference implementations. For a detailed description of the example builds' architectures and the use cases that they are based upon, reference Sections 4 and 5 in Volume B.

1.2.2 Architectural Overview

The MFA for e-commerce high-level reference architectures illustrated in Figure 1-1 and Figure 1-2 show the *cost threshold* and *risk engine* example implementations, respectively. The high-level reference architectures display the data communication among the returning purchaser, retailer e-commerce platform, risk assessment/MFA module and risk engine, MFA mechanisms, and logging and reporting dashboard.

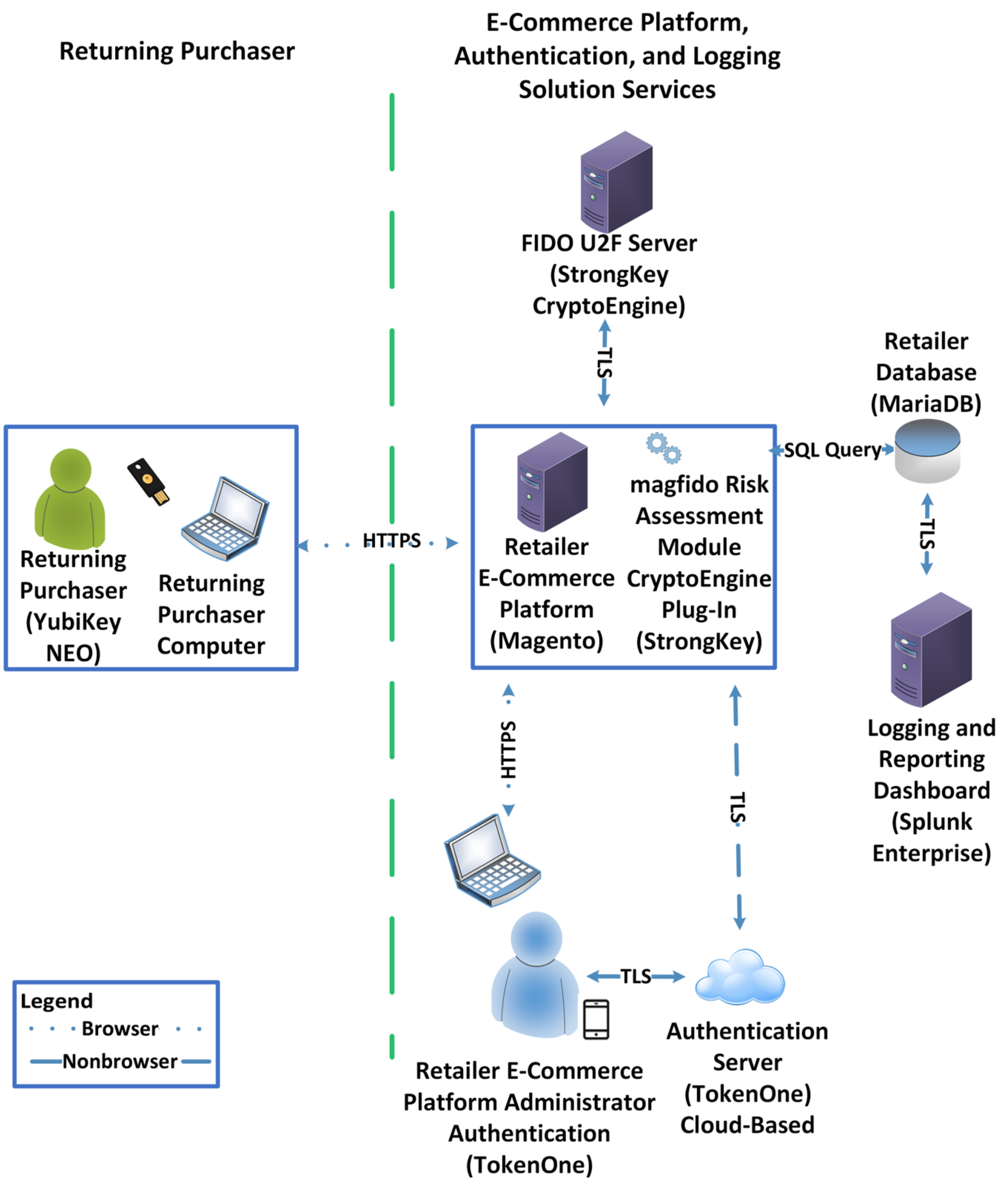
The *cost threshold* example implementation uses a predetermined shopping-cart price threshold to require use of MFA by the returning purchaser. The *risk engine* example implementation uses analytics to determine when MFA is required by the returning purchaser. The two example implementations include e-commerce platform capabilities, risk assessment and MFA, and logging and display capabilities.

The example implementations were constructed on the NCCoE's VMware vSphere virtualization operating environment. Internet access was used to connect to remote cloud-based components, while software components were installed as virtual servers within the vSphere environment.

TokenOne's authentication capability authenticates the Magento e-commerce platform administrator before any administration modifications are made to the e-commerce platform. It is based upon TokenOne's cloud-based authentication infrastructure and a smartphone application on either an Android or iPhone device. This helps secure the overall e-commerce organization's infrastructure.

The lab network that was used to build and configure the example implementations is not connected to the NIST enterprise network.

Figure 1-1 MFA for E-Commerce High-Level Cost Threshold Reference Architecture



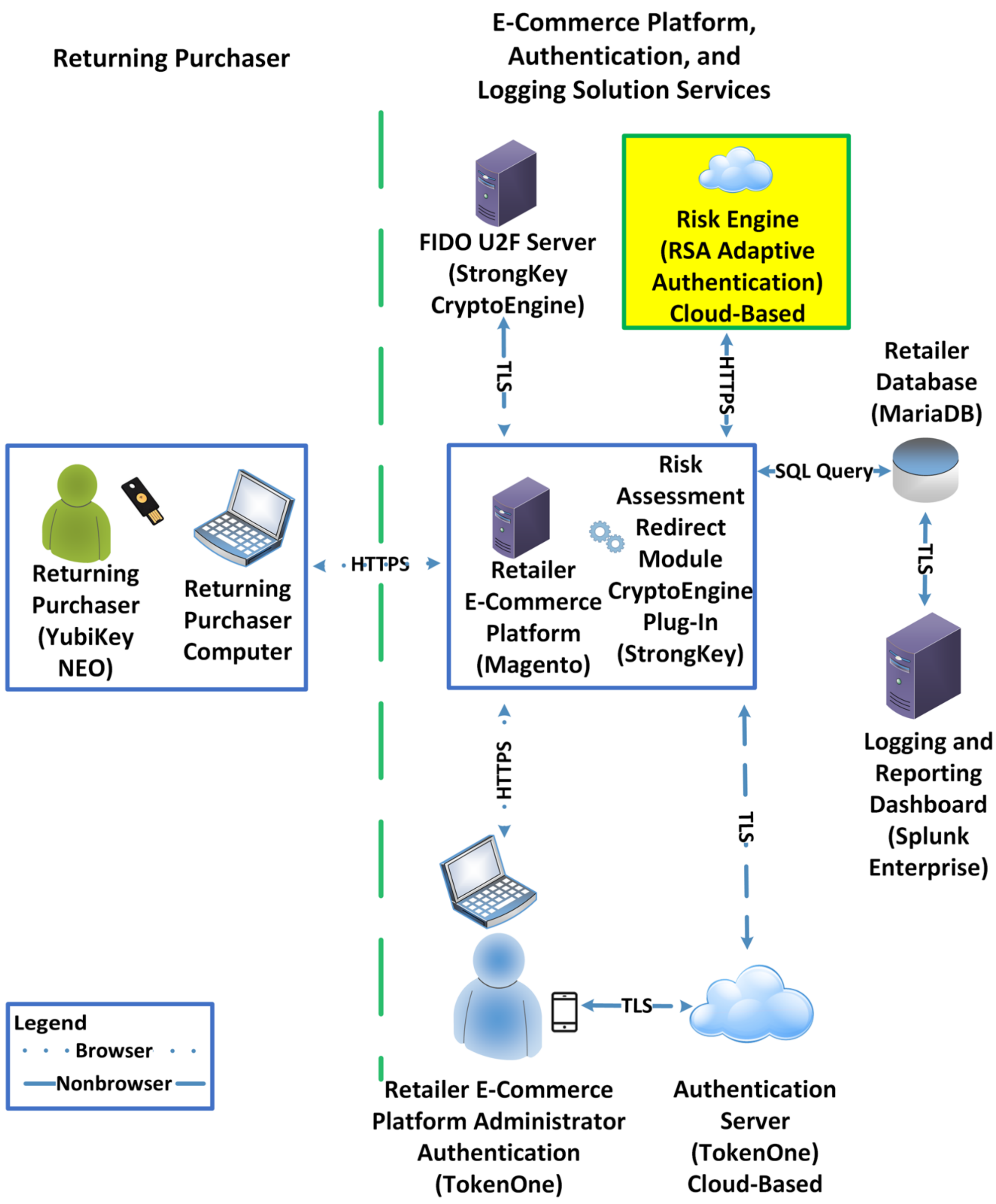
Returning Purchaser
E-Commerce Platform, Authentication, and Logging Solution Services
FIDO U2F Server (StrongKey CryptoEngine)
TLS
Retailer Database (MariaDB)
SQL Query
Returning Purchaser (YubiKey NEO)
Returning Purchaser Computer
HTTPS
Retailer E-Commerce Platform (Magento)
magfido Risk Assessment Module CryptoEngine Plug-In (StrongKey)
TLS
Logging and Reporting Dashboard (Splunk Enterprise)
HTTPS
TLS
TLS
Retailer E-Commerce Platform Administrator Authentication (TokenOne)
Authentication Server (TokenOne) Cloud-Based
Legend
Browser
Nonbrowser

The *cost threshold* example build illustrated in Figure 1-1 uses the components listed in Table 1-1.

Table 1-1 Cost Threshold Architecture List of Components

Components	Installation Guidance
StrongKey CryptoEngine (SKCE) FIDO U2F server and CryptoEngine plug-in	Section 2.1
Magento Open Source e-commerce platform	Section 2.2
StrongKey Magento magfido risk assessment module	Section 2.3
TokenOne Authentication	Section 2.5
Splunk Enterprise logging/reporting dashboard	Section 2.6
Yubico YubiKey NEO Security Key	Section 2.7

Figure 1-2 MFA for E-Commerce High-Level Risk Engine Reference Architecture



Returning Purchaser
E-Commerce Platform, Authentication, and Logging Solution Services
FIDO U2F Server (StrongKey CryptoEngine)
Risk Engine (RSA Adaptive Authentication) Cloud-Based
Retailer Database (MariaDB)
TLS
HTTPS
SQL Query
Returning Purchaser (YubiKey NEO)
Returning Purchaser Computer
HTTPS
Retailer E-Commerce Platform (Magento)
Risk Assessment Redirect Module CryptoEngine Plug-In (StrongKey)
TLS
Logging and Reporting Dashboard (Splunk Enterprise)
HTTPS
TLS
TLS
Legend
Browser
Nonbrowser
Retailer E-Commerce Platform Administrator Authentication (TokenOne)
Authentication Server (TokenOne) Cloud-Based

The *risk engine* example build illustrated in Figure 1-2 uses the components listed in Table 1-2. The Risk Engine Adaptive Authentication component is highlighted in yellow within the green box.

Table 1-2 Risk Engine Architecture List of Components

Components	Installation Guidance
SKCE FIDO U2F server and CryptoEngine plug-in	Section 2.1
Magento Open Source e-commerce platform	Section 2.2
RSA Adaptive Authentication	Section 2.4
TokenOne Authentication	Section 2.5
Splunk Enterprise logging/reporting dashboard	Section 2.6
Yubico YubiKey NEO Security Key	Section 2.7

1.2.3 General Infrastructure Details and Requirements

The lab network architecture is shown in Figure 1-3, where the relationship among the MFA example implementation components, firewalls, and network design are illustrated. The installation and configuration for many of the components shown in Figure 1-3 will be referenced in this volume of the guide.

Figure 1-3 MFA for E-Commerce Lab Network Architecture

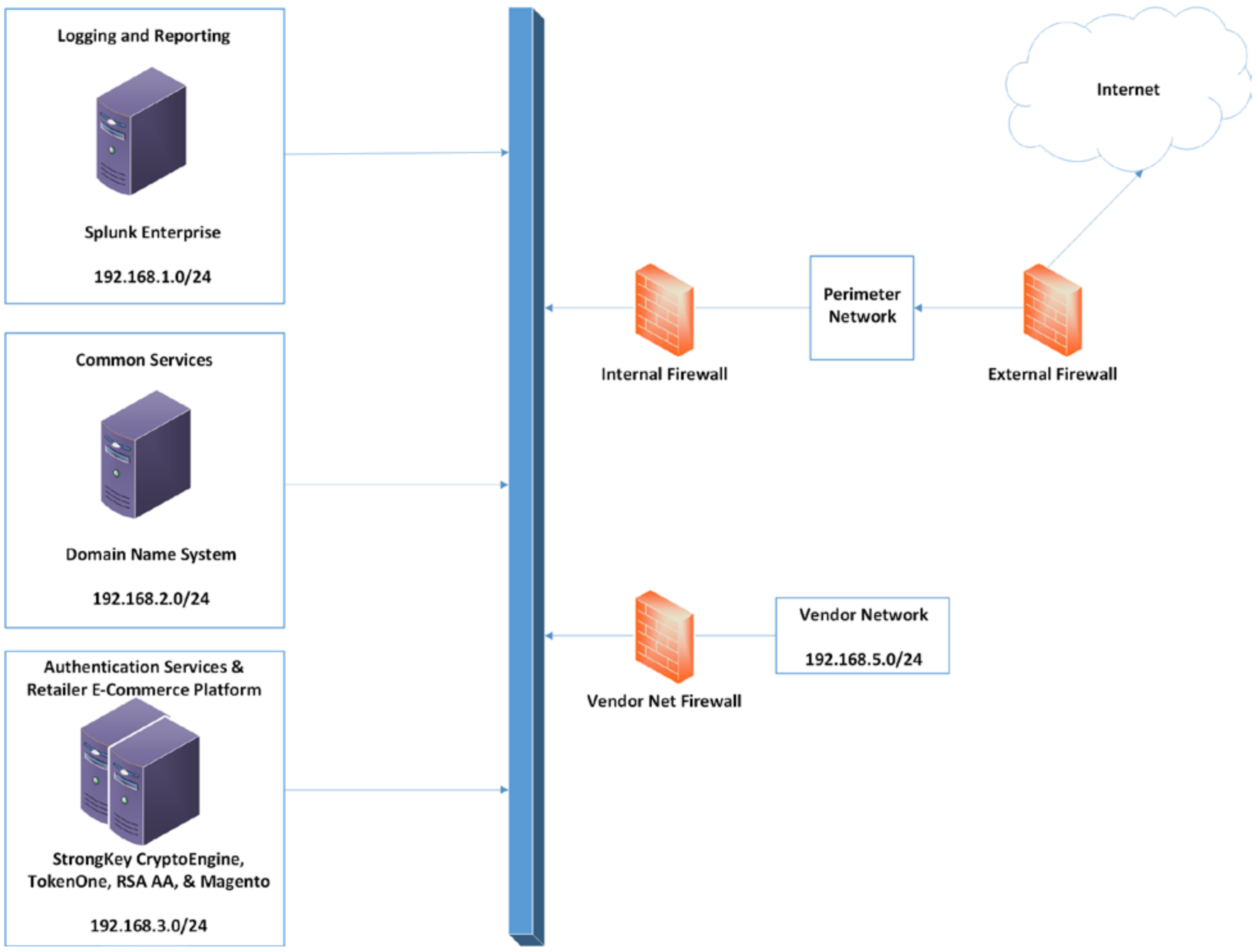


Table 1-3 lists the MFA example lab build's network internet protocol (IP) address range, system, and associated IP addresses. These network addresses were used in the example implementation builds and will be modified to reflect actual network architectures when deployed into a retailer's information system network.

Table 1-3 MFA Example Lab Build Network Details

Network	System	IP Address
192.168.1.0/24	Splunk Enterprise server logging and reporting	192.168.1.10
192.168.2.0/24	domain name system (DNS) common services	192.168.2.10
192.168.3.0/24	SKCE FIDO U2F server authentication services	192.168.3.30
192.168.3.0/24	RSA Adaptive Authentication connectivity, TokenOne, Magento Open Source authentication services and retailer e-commerce platform	192.168.3.155
192.168.5.0/24	Optional future services for vendor network	As assigned

There are both prerequisite infrastructure and example implementation components, whose installation and configuration are described below.

1.2.3.1 Domain Name System

DNS was configured within the lab to facilitate data communication among the example implementation components. The domain names and IP address ranges will be modified to reflect actual network architectures when deployed into an online retailer's information system network.

The name of the domain used for this example build is mfa.local. Create the following host records in the mfa.local forward lookup zone by using the host names, fully qualified domain names (FQDNs), and IP addresses listed in Table 1-4.

Table 1-4 Lab Network Host Record Information

Host Name	FQDN	IP Address
Splunk	Splunk.mfa.local	192.168.1.10
DNS	DNS.mfa.local	192.168.2.10
Magento	Magento.mfa.local	192.168.3.30
Magento2	Magento2.mfa.local	192.168.3.155

The network adapter configuration for the DNS server is as follows:

- Network Configuration (Interface 1)
  - IPv4 Manual
  - IPv6 Disabled

    - IP Address: 192.168.2.10
    - Netmask: 255.255.255.0
    - Gateway: 192.168.2.1
    - DNS Name Servers: 192.168.2.10
  - DNS-Search Domains: mfa.local

1.3 Typographic Conventions

The following table presents typographic conventions used in this volume.

Typeface/Symbol	Meaning	Example
Italics	file names and path names; references to documents that are not hyperlinks; new terms; and placeholders	For detailed definitions of terms, see the *NCCoE Glossary.*
Bold	names of menus, options, command buttons, and fields	Choose **File > Edit.**
`Monospace`	command-line input, onscreen computer output, sample code examples, and status codes	`mkdir`
`Monospace Bold`	command-line user input contrasted with computer output	**`service sshd start`**
blue text	link to other parts of the document, a web URL, or an email address	All publications from NIST's NCCoE are available at https://www.nccoe.nist.gov.

2 How to Install and Configure

This section of the practice guide contains detailed instructions for installing and configuring the products used to build the example implementations.

2.1 StrongKey CryptoEngine FIDO U2F Server

This section of the guide provides installation and configuration guidance for the SKCE, which provides FIDO authentication services.

2.1.1 StrongKey CryptoEngine Overview

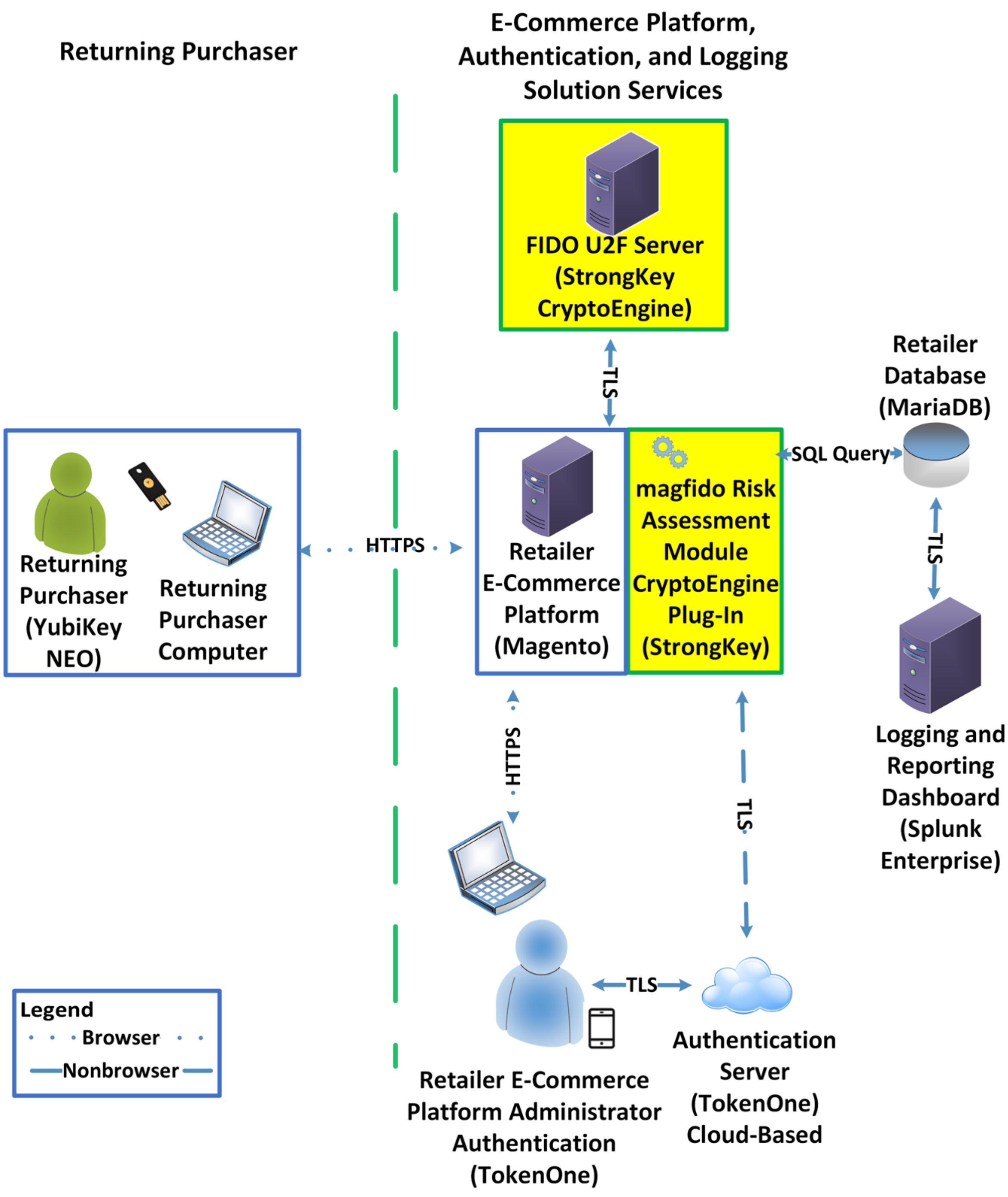
The SKCE 2.0 Build 163 from StrongKey [3] performs the FIDO U2F [1], [2] server functionality in the build architecture.

SKCE is provided in the StrongKey Key Appliance, but the company also distributes some of its software under the Lesser General Public License, published by the Free Software Foundation. SKCE was downloaded from the StrongKey repository on SourceForge and was used in this build.

The CryptoEngine plug-in enables Magento to communicate with the SKCE when the returning purchasers require MFA.

Both the *cost threshold* and *risk engine* example implementations use the SKCE's capabilities. The components that are installed by using the instructions in this section are illustrated in Figure 2-1 and are highlighted in yellow within the green box.

Figure 2-1 StrongKey CryptoEngine Components



Returning Purchaser
E-Commerce Platform, Authentication, and Logging Solution Services
FIDO U2F Server (StrongKey CryptoEngine)
TLS
Retailer Database (MariaDB)
SQL Query
Returning Purchaser (YubiKey NEO)
Returning Purchaser Computer
HTTPS
Retailer E-Commerce Platform (Magento)
magfido Risk Assessment Module CryptoEngine Plug-In (StrongKey)
TLS
Logging and Reporting Dashboard (Splunk Enterprise)
HTTPS
TLS
TLS
Legend
Browser
Nonbrowser
Retailer E-Commerce Platform Administrator Authentication (TokenOne)
Authentication Server (TokenOne) Cloud-Based

Installation instructions and the product download site for StrongKey's FIDO U2F server, SKCE, can be found at https://sourceforge.net/projects/skce/. For this example implementation, we installed and configured a local copy of SKCE by using the SKCE installation instructions documented below in Section 2.1.2.

2.1.2 SKCE Requirements

The following subsections document the software, hardware, and network requirements for SKCE Version 2.0.

2.1.2.1 SKCE Software Requirements

For this build, SKCE was installed on a Community Enterprise Operating System (CentOS) 7.4 64-bit server.

Because SKCE is a Java application, it is compatible with operating systems that support a compatible version of Java and the other required software. The application was built with the Oracle Java Development Kit (JDK) Version 8, Update 72. Instructions for obtaining Oracle JDK and the other necessary components are provided in this section.

SKCE can be installed manually or with an installation script included in the download. SKCE depends on other software components, including a Structured Query Language (SQL) database, a lightweight directory access protocol (LDAP) directory server, and the Glassfish Java application server. By default, the script will install MariaDB, OpenDJ, and Glassfish all on a single server.

For this build, the scripted installation was used with the default software components. The required software components listed below must be downloaded prior to running the installation script:

- Glassfish 4.1 2010
- Java Cryptography Extension Unlimited Strength Jurisdiction Policy Files 8 2011
- JDK 8, Update 121 2012
- OpenDJ 3.0.0 2013
- MariaDB 10.1.22 2014
- MariaDB Java Client 2015

Reference StrongKey's scripted installation instructions for details and preinstallation software download links (StrongKey, n.d.).

Note: To download OpenDJ, the user must register for a free account for ForgeRock BackStage.

2.1.2.2 Hardware Requirements

StrongKey recommends installing SKCE on a server with at least 10 gigabytes (GB) of available disk space and 4 GB of random access memory (RAM).

2.1.2.3 Network Requirements

The SKCE application programming interface (API) uses transmission control protocol (TCP) Port 8181 (Table 2-1). Any applications that request U2F registration, authentication, or deregistration actions from the SKCE need to be able to connect on this port. Glassfish runs a hypertext transfer protocol secure (https) service on this port. Use firewall-cmd, iptables, or any other system utility for manipulating the firewall to open this port.

Table 2-1 Network Ports to Be Enabled

Port	Use
TCP 8181	U2F Application Access

Other network services listen on the ports listed in Table 2-2. For the scripted installation, where all of these services are installed on a single server, there is no need to adjust firewall rules for these services when they are only accessed from localhost.

Table 2-2 Local Ports

Port	Use
TCP 3306	MariaDB listener
TCP 4848	Glassfish administrative console
TCP 1389	OpenDJ LDAP service

2.1.3 Install SKCE, the FIDO U2F Authentication Server

The installation procedure consists of the following steps:

- Download the software dependencies to the server where SKCE will be installed.
- Make any required changes to the installation script.
- Run the script as root/administrator.
- Perform post-installation configuration.
- Reference StrongKey's scripted installation instructions for details and preinstallation software download links (StrongKey, n.d.).

The installation script creates a "strongauth" Linux user and installs all software under */usr/local/strongauth*. Rather than reproduce the installation steps here, this section provides some notes on the installation procedure:

1. Download the software. Download and unzip the SKCE build to a directory on the server where SKCE is being installed. Download all installers as directed in the SKCE instructions to the same directory.

2. Change software versions as required in the installation script. If different versions of any of the software dependencies were downloaded, update the file names in the installation script *(install-skce.sh)*. Using different versions of the dependencies, apart from minor point-release versions, is not recommended. For the lab build, JDK Version 8u151 was used instead of the version referenced in the instructions. This required updating the JDK and JDKVER settings in the file.

3. Change passwords in the installation script. Changing the default passwords in the delivered script is strongly recommended. The defaults are readily discoverable, as they are distributed with the software. Passwords should be stored in a password vault or other agency-approved secure storage. Once the installation script has been run successfully, the script should be deleted or sanitized to remove passwords. The following lines in the installation script contain passwords:

   ```
   LINUX_PASSWORD=ShaZam123          # For 'strongauth' account
   GLASSFISH_PASSWORD=adminadmin     # Glassfish Admin password
   MYSQL_ROOT_PASSWORD=BigKahuna     # MySQL 'root' password
   MYSQL_PASSWORD=AbracaDabra        # MySQL 'skles' password
   SKCE_SERVICE_PASS=Abcd1234!       # Webservice user 'service-cc-ce' password
   SAKA_PASS=Abcd1234!
   SERVICE_LDAP_BIND_PASS=Abcd1234!
   SEARCH_LDAP_BIND_PASS=Abcd1234!
   ```

4. Set the App ID (identifier) uniform resource locator (URL): The App ID setting in *install-skce.sh* should point to a URL that will be accessible to clients where the *app.json* file can be downloaded. The default location is a URL on the SKCE server, but the SKCE would not be exposed to mobile clients in a typical production deployment. In the lab, *app.json* was hosted on the following SKCE server:

   /usr/local/strongauth/payara41/glassfish/domains/domain1/docroot/app.json

   This enables the file to be accessed by clients at the following URL: *https://magento.mfa.local:8181/app.json*.

5. Run the script. *install-skce.sh* must be run as the root user. If the installation script terminates with an error, then troubleshoot and correct any problems before continuing.

6. (For CentOS 7) create the firewall rule. The installation script attempts to open the required port by using iptables, which does not work on CentOS 7. In that case, the following commands will open the port:

```
# firewall-cmd --permanent --add-port 8181/tcp
success
# firewall-cmd --reload

success
```

7. Restart Glassfish. On CentOS 7, run the following command:

```
$ sudo systemctl restart glassfishd
```

8. Complete Step 3b in the SKCE installation instructions to activate the cryptographic module.

9. Complete Step 3c in the SKCE installation instructions to create the domain signing key. When prompted for the App ID, use the URL referenced above in the App ID setting of the *install-skce.sh* script.

10. Complete Step 4 in the SKCE installation instructions if secondary SKCE instances are being installed; this was not done for this build but is recommended for a production installation.

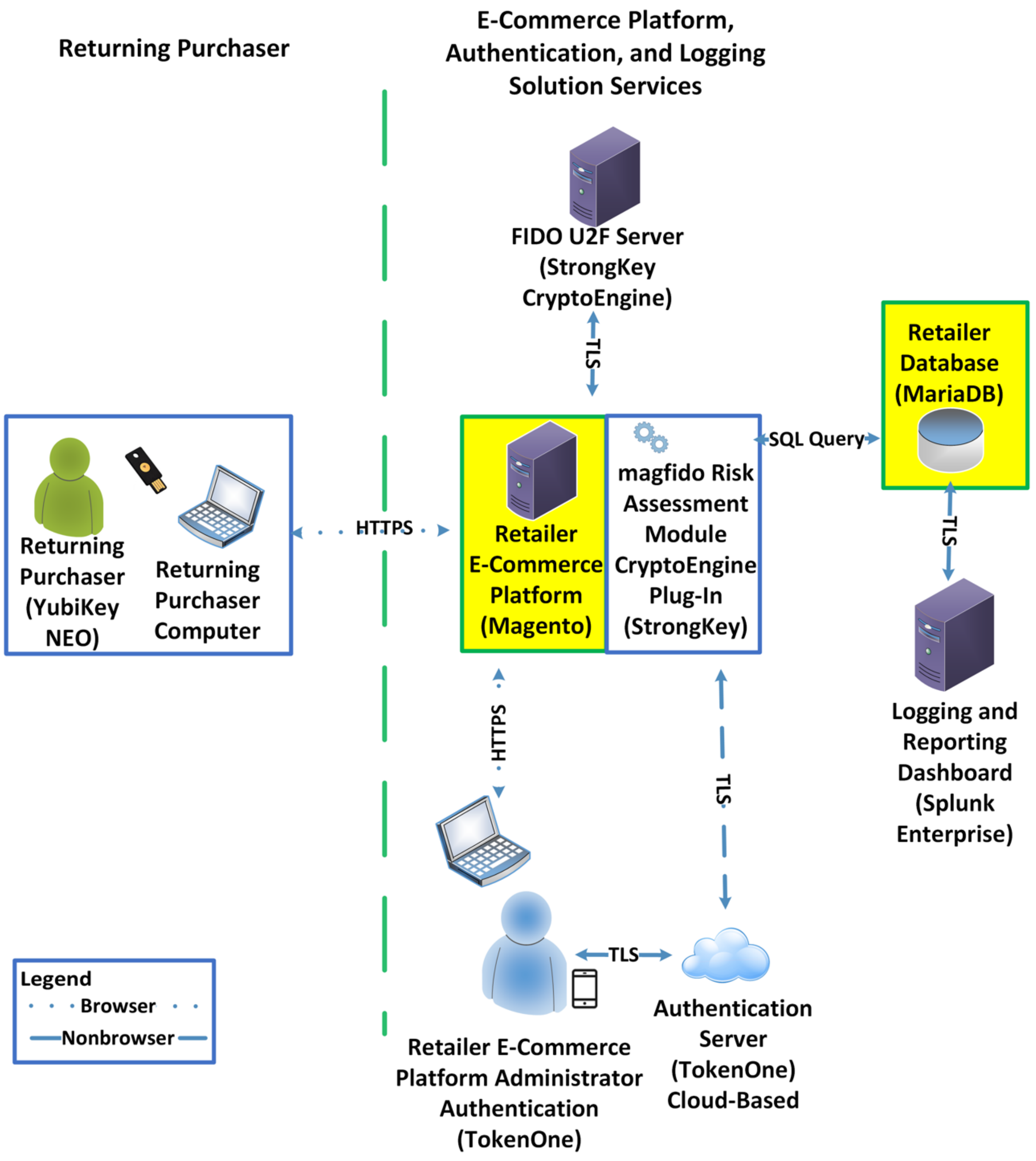
11. Test the FIDO Engine. Follow the testing instructions under Step D at the following URL: https://sourceforge.net/p/skce/wiki/Test%20SKCE%202.0%20Using%20a%20Client%20Program%20%28Build%20163%29/.

    There are additional tests on that web page to test the other cryptographic functions of the SKCE; however, only the FIDO Engine tests are critical for this build.

2.2 Magento Open Source Electronic Commerce Platform

This section provides installation and configuration guidance for the Magento Open Source e-commerce platform. The Magento platform provides connectivity to most of the example implementations' components. Both example implementation builds use Magento. The location of the Magento components that are installed using the instructions in this section are illustrated in Figure 2-2 and are highlighted in yellow within the green boxes.

Figure 2-2 Magento Open Source E-Commerce Platform Components



Returning Purchaser
E-Commerce Platform, Authentication, and Logging Solution Services
FIDO U2F Server (StrongKey CryptoEngine)
TLS
Retailer Database (MariaDB)
SQL Query
Returning Purchaser (YubiKey NEO)
Returning Purchaser Computer
HTTPS
Retailer E-Commerce Platform (Magento)
magfido Risk Assessment Module CryptoEngine Plug-In (StrongKey)
TLS
Logging and Reporting Dashboard (Splunk Enterprise)
HTTPS
TLS
TLS
Legend
Browser
Nonbrowser
Retailer E-Commerce Platform Administrator Authentication (TokenOne)
Authentication Server (TokenOne) Cloud-Based

2.2.1 Magento Overview

Magento is an e-commerce platform that offers on-premises and cloud solutions to retailers. For this lab implementation, we leveraged the Magento Open Source version of this platform, which was hosted on-premises. This section describes how to install and configure Magento Open Source [4], [5] and how to configure it with StrongKey's SKCE FIDO U2F server capabilities. For the e-commerce platform, Magento Open Source Version 2.1.8 was used in the example implementation.

The installation procedure consists of the following steps:

- Download the Magento software to the server where it will be installed.
- Download the software dependencies to the server where Magento will be installed.
- Execute commands as root/administrator.
- Perform post-installation configuration.

2.2.2 Magento Requirements

The following subsections document the software, hardware, and network requirements for Magento Open Source 2.1.X.

2.2.2.1 Software Requirements

For this implementation, Magento was installed on a CentOS 7.0 server.

Magento Open Source developer's documentation states that Magento can operate on Linux operating systems, such as these:

- Red Hat Enterprise Linux
- CentOS
- Ubuntu
- Debian

Magento Open Source 2.1.X requires the following installations:

- Web Server: Apache 2.2 or 2.4 or NGINX 1.X
- Database: MySQL 5.6, MariaDB, Percona, or other binary-compatible MySQL technologies
- Hypertext Preprocessor (PHP): 7.0.2, 7.0.4, 7.0.6-7.0.X, or 7.1.X
- Secure Sockets Layer (SSL)
- Mail Server: Redis 3.0, Varnish 3.5, memcached

See Magento's developer's documentation for additional details and download links: https://devdocs.magento.com/guides/v2.1/install-gde/system-requirements-tech.html.

2.2.2.2 Hardware Requirements

Magento requires installing Magento Open Source on a server with at least 2 GB of RAM.

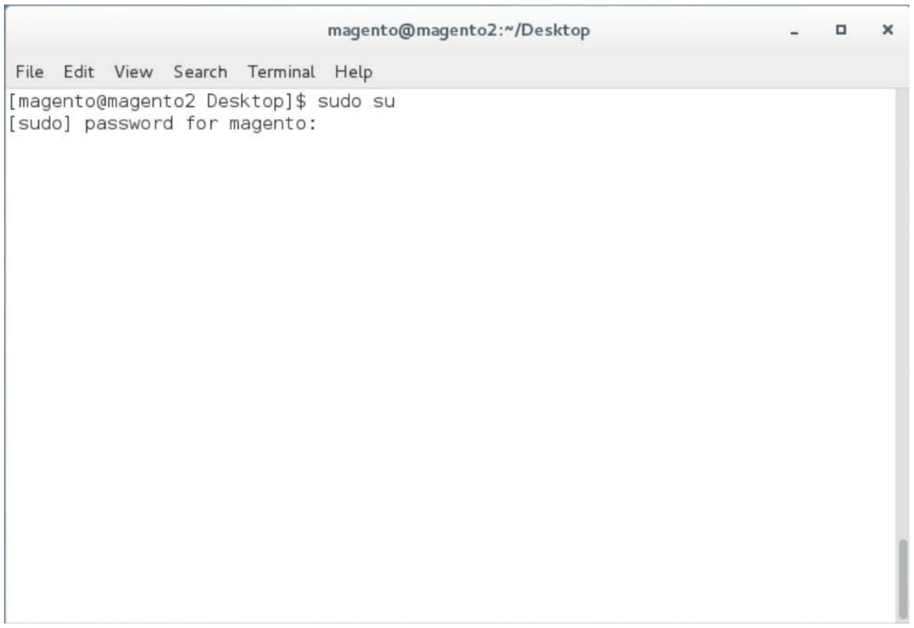
2.2.3 Magento Preinstallation

Magento requires the Linux, Apache, MySQL, PHP (LAMP) software stack. This section describes the process of installing and configuring the LAMP software stack that uses versions compatible with Magento.

1. Open a terminal window, and enter the following command to log in as root:

   ```
   sudo su
   ```

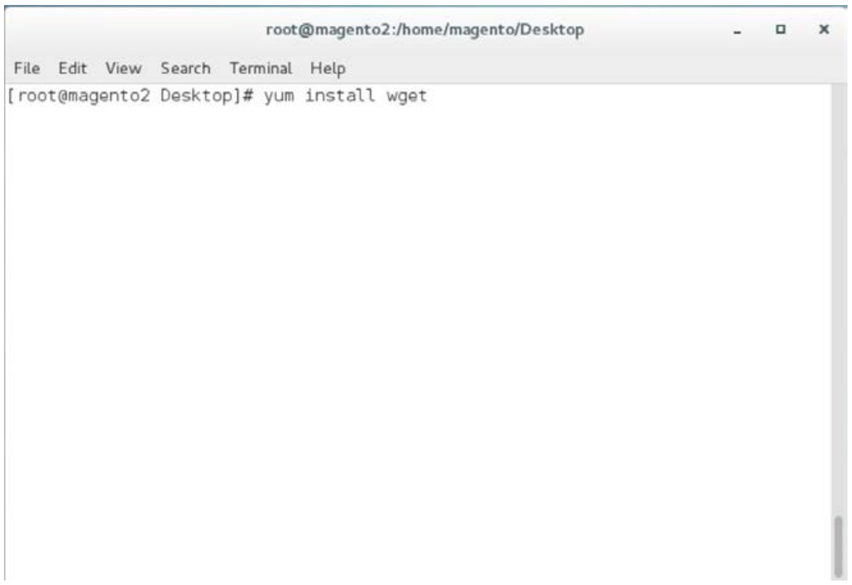
   a. After entering the command, you will be prompted to enter the password for the current user.


magento@magento2:~/Desktop
File Edit View Search Terminal Help

```
[magento@magento2 Desktop]$ sudo su
[sudo] password for magento:
```

2. To install wget from the terminal, enter the following command:

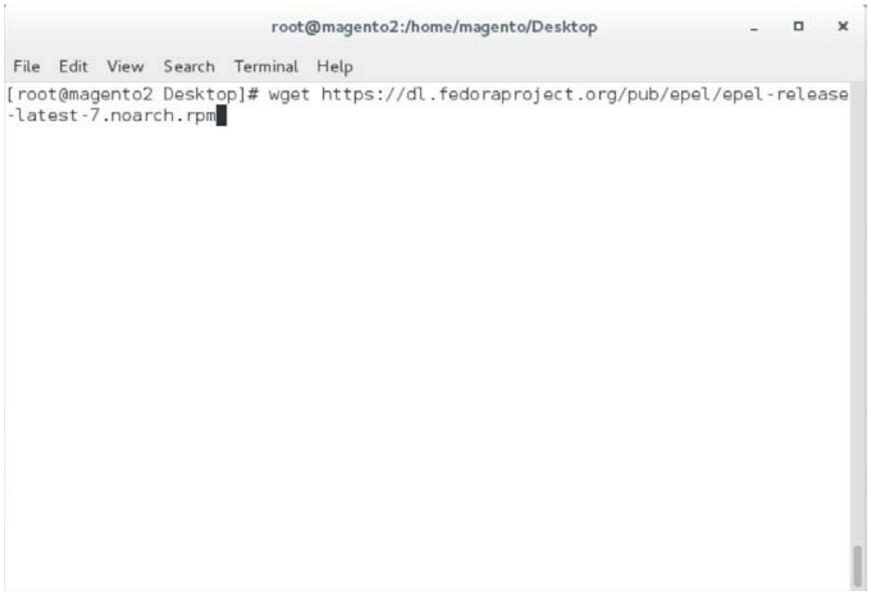
   ```
   yum install wget
   ```


root@magento2:/home/magento/Desktop

```
[root@magento2 Desktop]# yum install wget
```

3. Download the Extra Packages for Enterprise Linux repository by entering the following command:

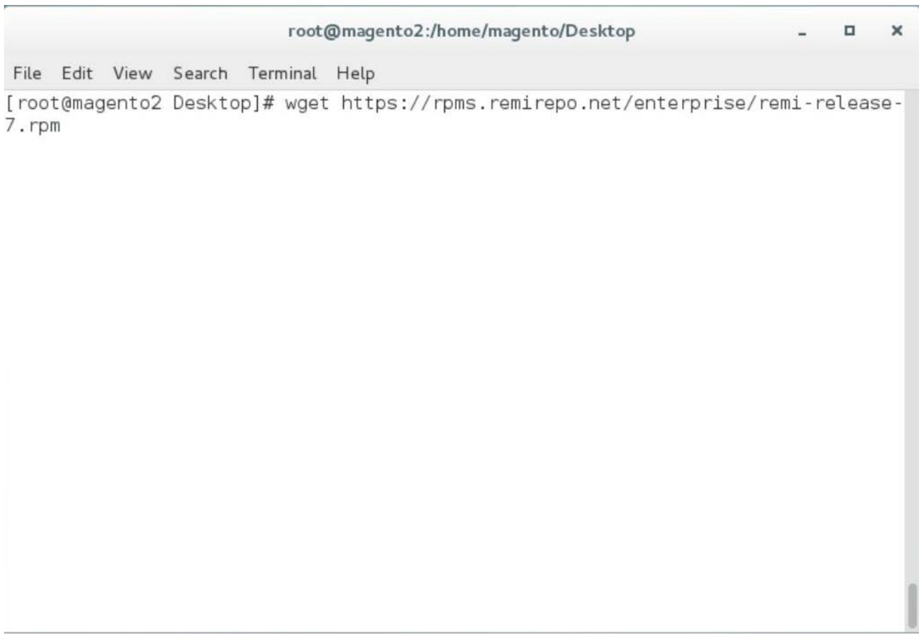
   ```
   wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
   ```


root@magento2:/home/magento/Desktop

```
[root@magento2 Desktop]# wget https://dl.fedoraproject.org/pub/epel/epel-release
-latest-7.noarch.rpm
```

4. Download the Remi repository by entering the following command:

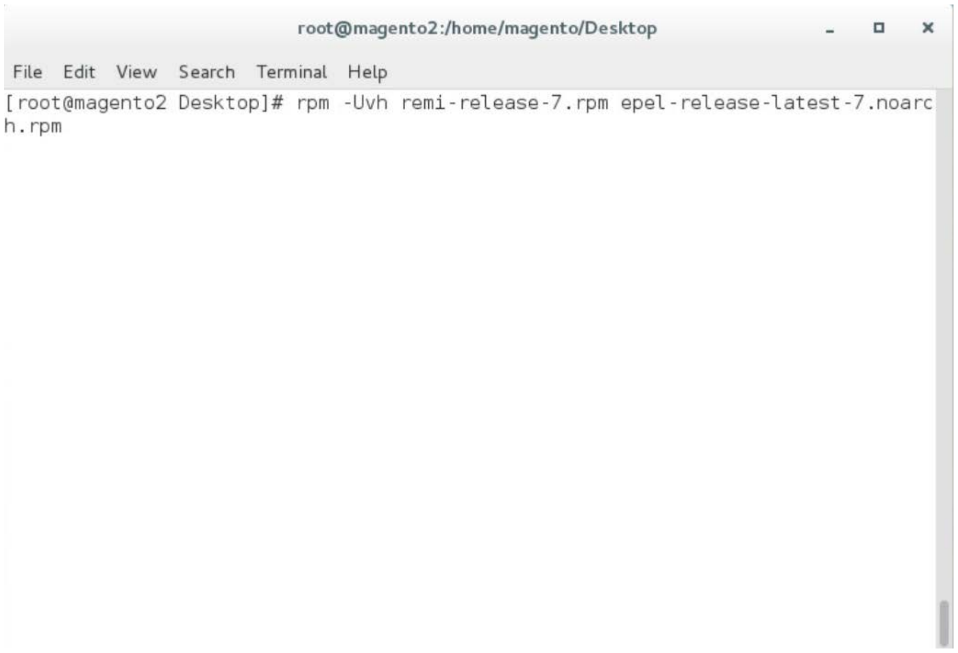
   ```
   wget http://rpms.remirepo.net/enterprise/remi-release-7.rpm
   ```


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
[root@magento2 Desktop]# wget https://rpms.remirepo.net/enterprise/remi-release-
7.rpm
```

5. Add the two repositories—so that Yellowdog Updater Modified (YUM) can locate them when needed—by entering the following command:

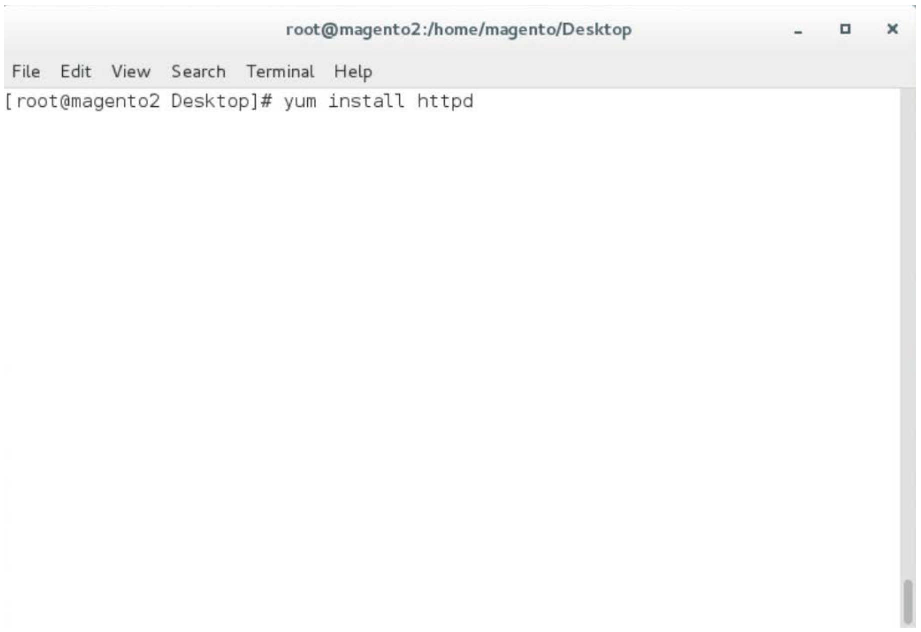
   ```
   rpm -Uvh remi-release-7.rpm epel-release-latest-7.noarch.rpm
   ```


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
[root@magento2 Desktop]# rpm -Uvh remi-release-7.rpm epel-release-latest-7.noarc
h.rpm
```

6. Install the Apache server by entering the following command:

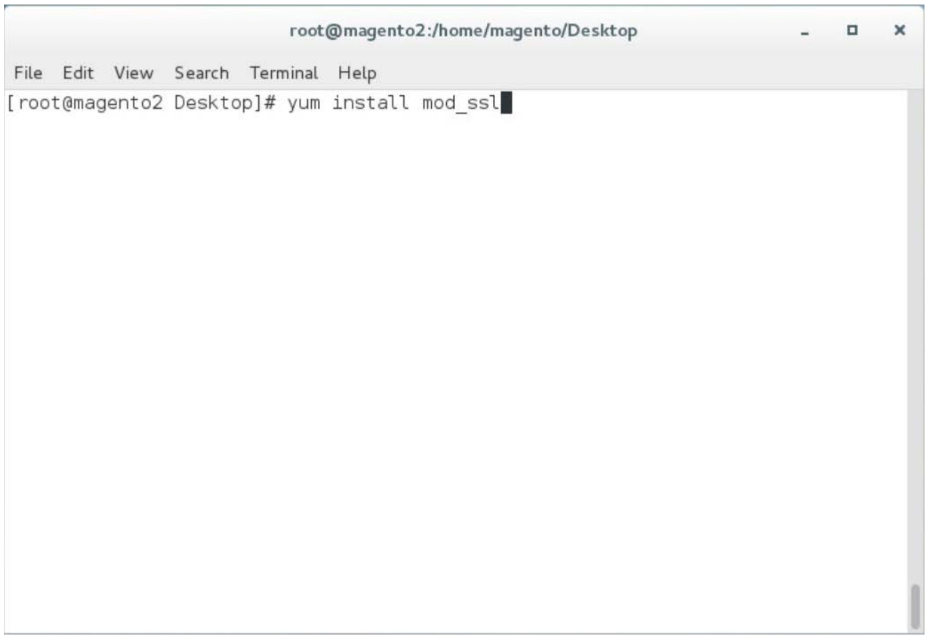
   ```
   yum install httpd
   ```


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
[root@magento2 Desktop]# yum install httpd
```

7. Install Transport Layer Security (TLS)/SSL support for Hypertext Transfer Protocol Daemon (HTTPD) by entering the following command:

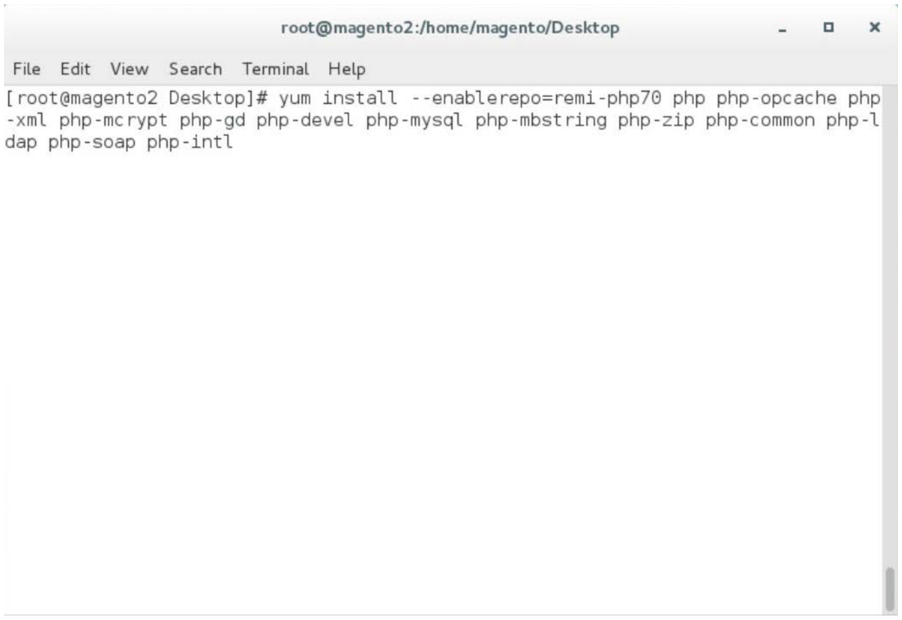
   ```
   yum install mod_ssl
   ```


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
[root@magento2 Desktop]# yum install mod_ssl
```

8. Install PHP by entering the following command:

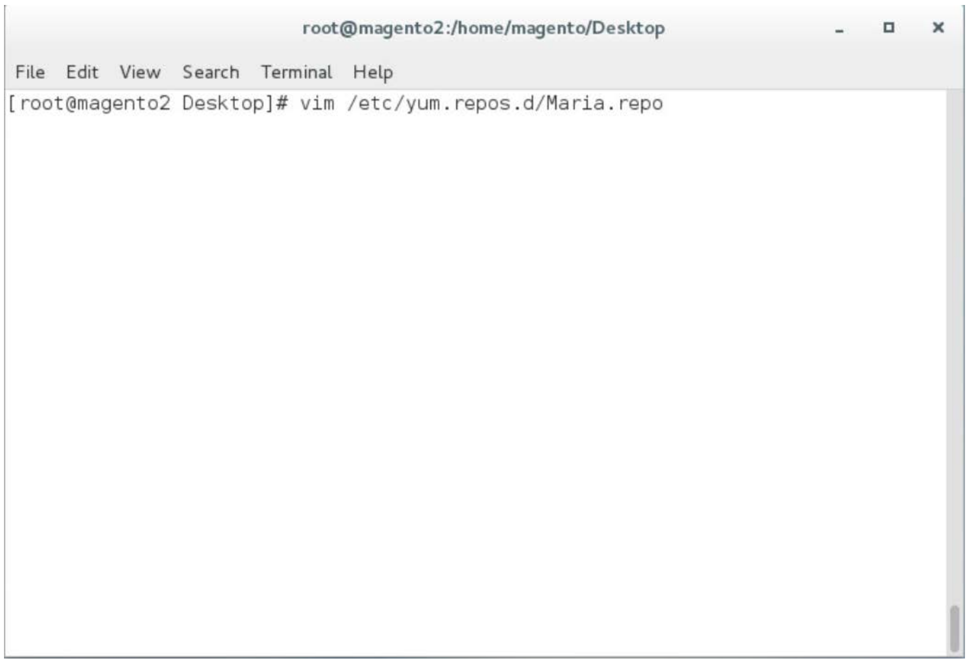
   ```
   yum install --enablerepo=remi-php70 php php-opcache php-xml php-mcrypt php-gd
   php-devel php-mysql php-mbstring php-zip phpcommon php-ldap php-soap php-intl
   ```


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
[root@magento2 Desktop]# yum install --enablerepo=remi-php70 php php-opcache php
-xml php-mcrypt php-gd php-devel php-mysql php-mbstring php-zip php-common php-l
dap php-soap php-intl
```

9. Create a file named *Maria.repo* in the */etc/yum.repos.d* by entering the following command:

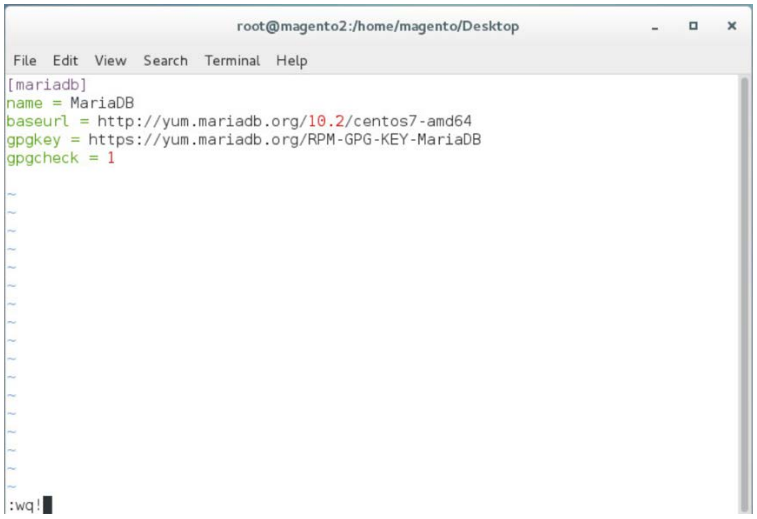
   ```
   vim /etc/yum.repos.d/Maria.repo
   ```


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
[root@magento2 Desktop]# vim /etc/yum.repos.d/Maria.repo
```

10. In the text editor, enter the following contents:

```
[mariadb]

name = MariaDB

baseurl = http://yum.mariadb.org/10.2/centos7-amd64

gpgkey = https://yum.mariadb.org/RPM-GPG-KEY-MariaDB

gpgcheck = 1
```


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

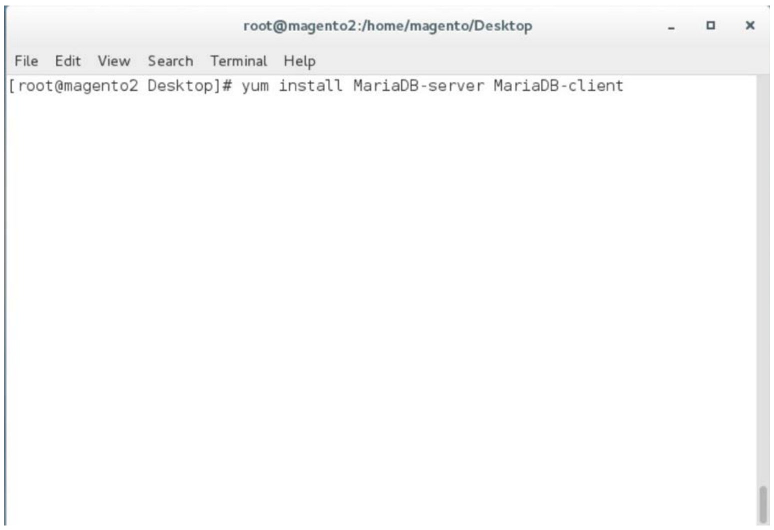
```
[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/10.2/centos7-amd64
gpgkey = https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck = 1

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:wq!
```

11. Save the file, and exit by entering the following command:

    ```
    :wq!
    ```

12. Install MariaDB by entering the following command:

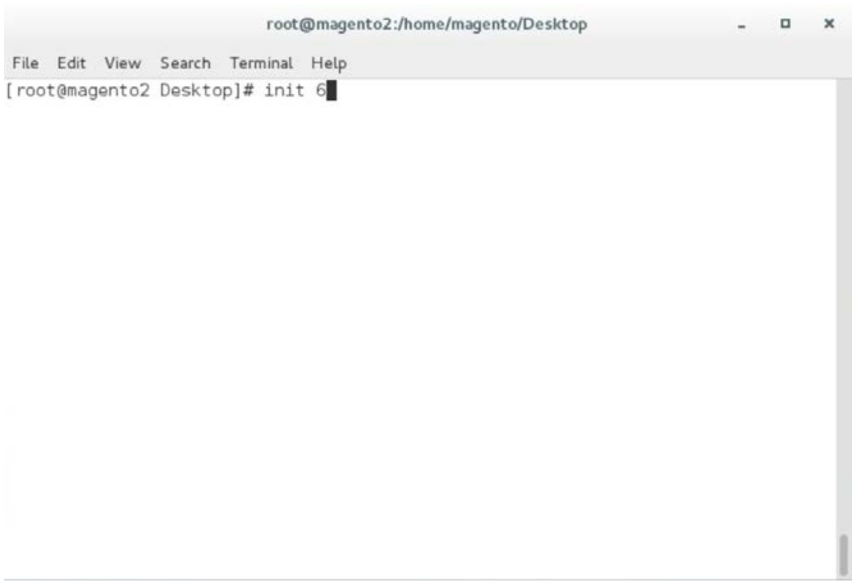
    ```
    yum install mariadb-server mariadb-client
    ```


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
[root@magento2 Desktop]# yum install MariaDB-server MariaDB-client
```

13. Restart the computer system by entering the following command:

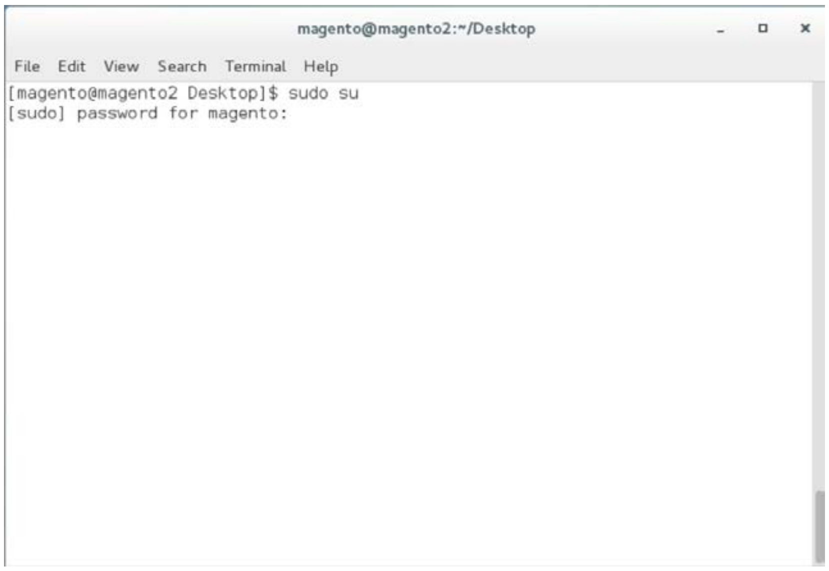
    ```
    init 6
    ```


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
[root@magento2 Desktop]# init 6
```

14. Open a terminal window, and enter the following command to log in as root:

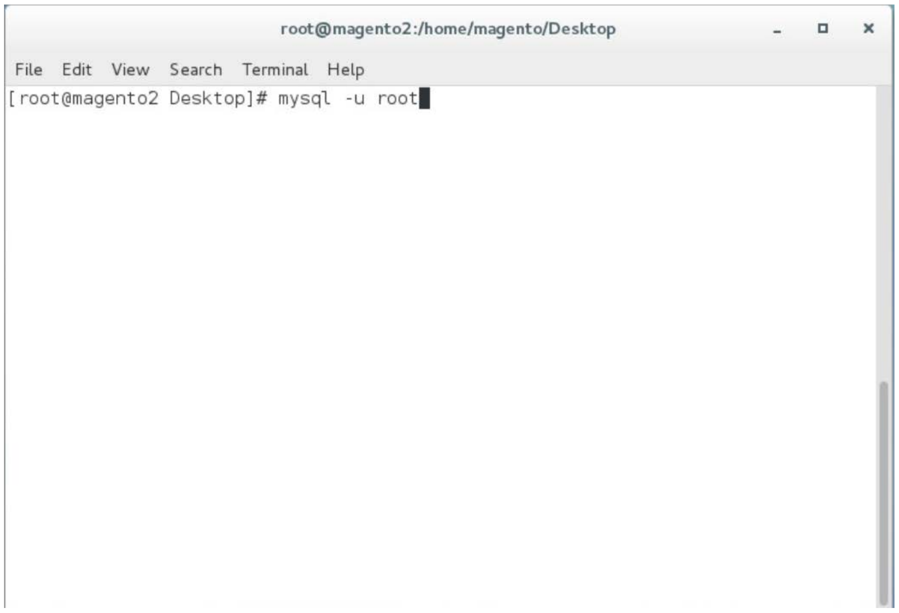
    ```
    sudo su
    ```


magento@magento2:~/Desktop
File Edit View Search Terminal Help

```
[magento@magento2 Desktop]$ sudo su
[sudo] password for magento:
```

15. Log in to MariaDB as root by entering the following command (Note: Even though the MariaDB relational database is being used, it uses the same tools as the MySQL database.):

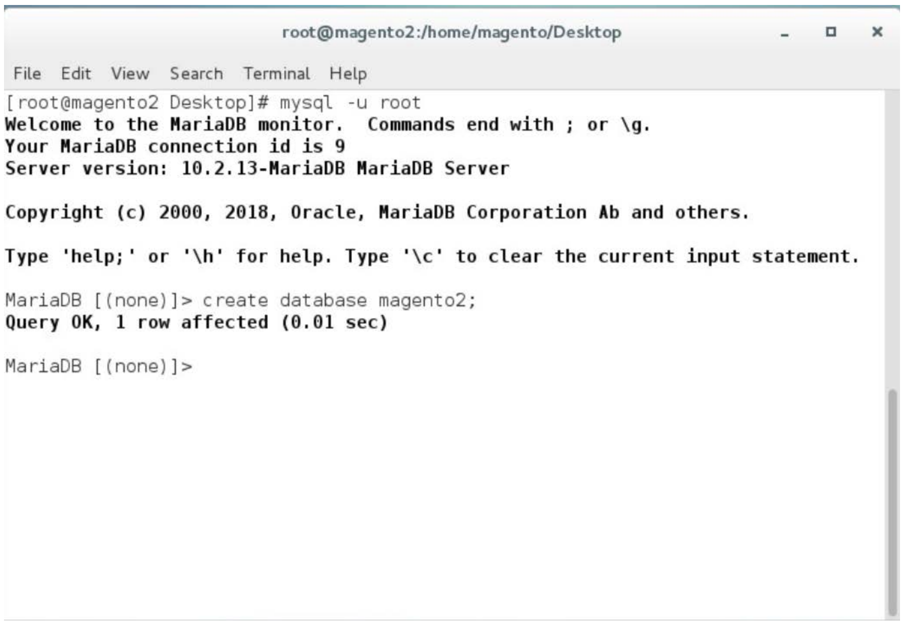
    ```
    mysql -u root
    ```


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
[root@magento2 Desktop]# mysql -u root
```

16. Create the Magento database by entering the following SQL command:

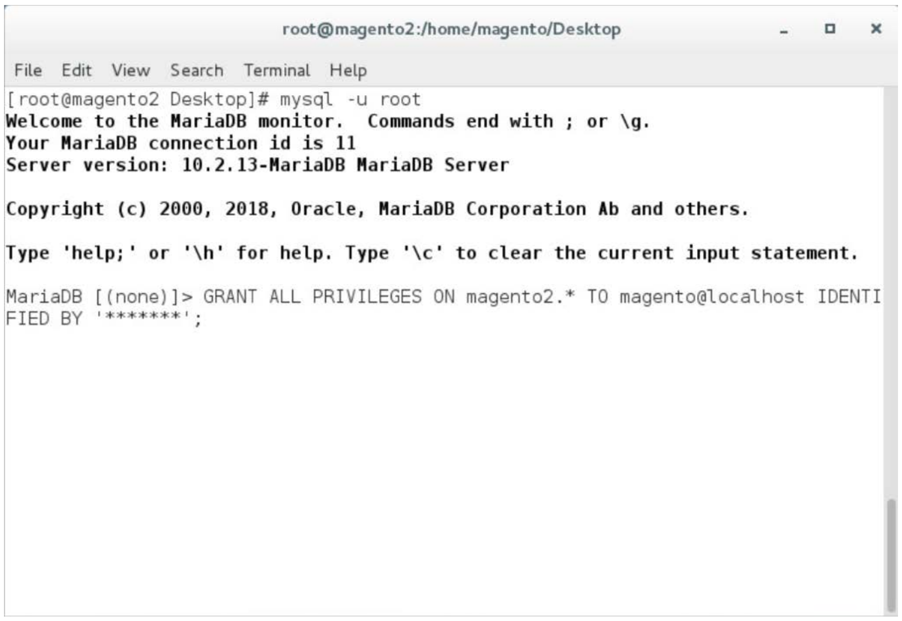
```
create database magento2;
```


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
[root@magento2 Desktop]# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.2.13-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database magento2;
Query OK, 1 row affected (0.01 sec)

MariaDB [(none)]>
```

17. Create the Magento user by entering the following command, replacing parameters in <> with values appropriate for your installation:

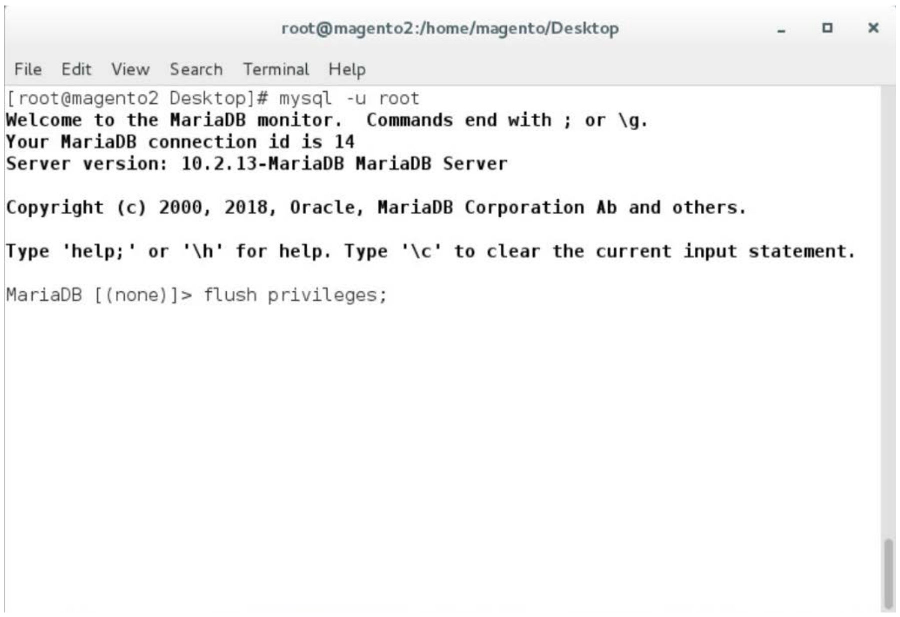
```
GRANT ALL PRIVILEGES ON magento2.* TO magento@localhost IDENTIFIED BY '<db
password>';
```


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
[root@magento2 Desktop]# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 11
Server version: 10.2.13-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> GRANT ALL PRIVILEGES ON magento2.* TO magento@localhost IDENTI
FIED BY '*******';
```

18. Flush the database privileges by entering the following SQL command:

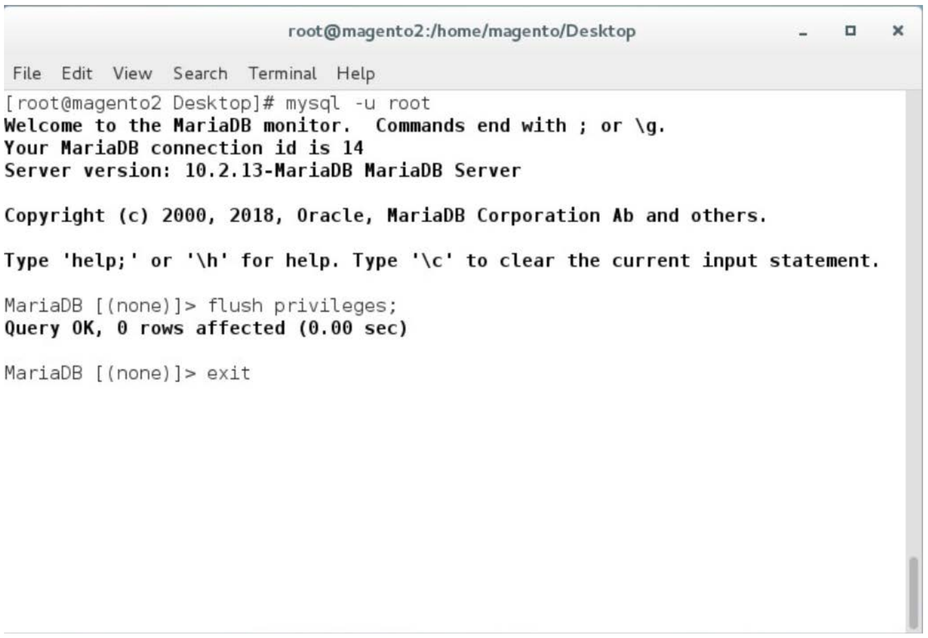
    ```
    flush privileges;
    ```


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
[root@magento2 Desktop]# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 14
Server version: 10.2.13-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> flush privileges;
```

19. Exit the MariaDB shell by entering the following command:

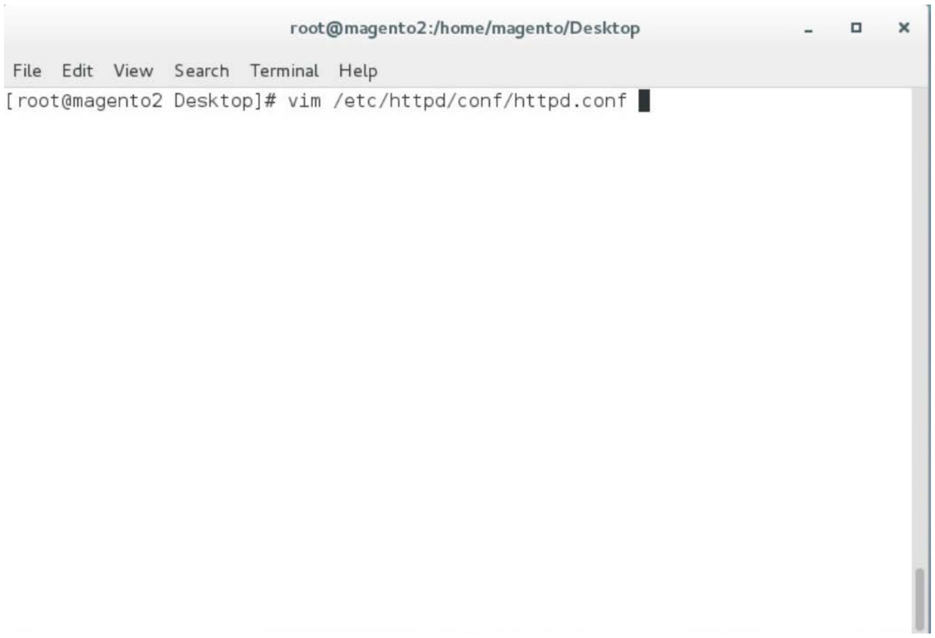
    ```
    exit
    ```


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
[root@magento2 Desktop]# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 14
Server version: 10.2.13-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> exit
```

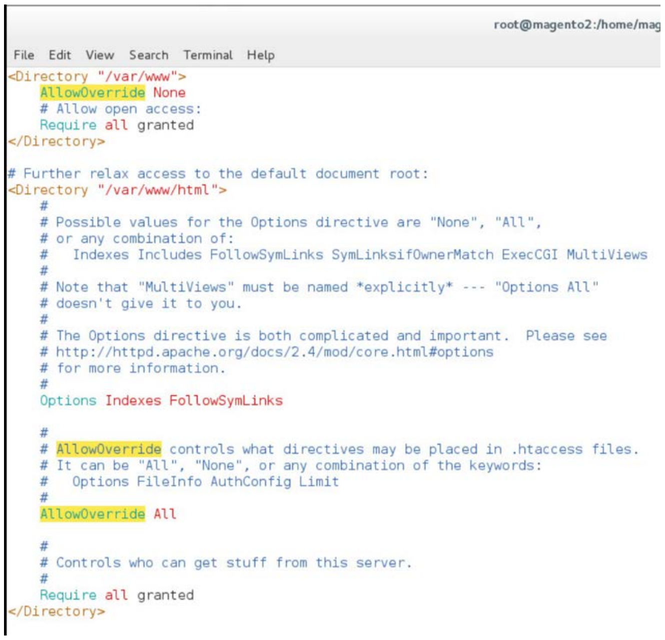
20. Open *httpd.conf* to modify Apache settings by entering the following command:

    ```
    vim /etc/httpd/conf/httpd.conf
    ```


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
[root@magento2 Desktop]# vim /etc/httpd/conf/httpd.conf
```

21. Locate the `<Directory "/var/www/html">` section, and change `"AllowOverride None"` to `"AllowOverride All"`.



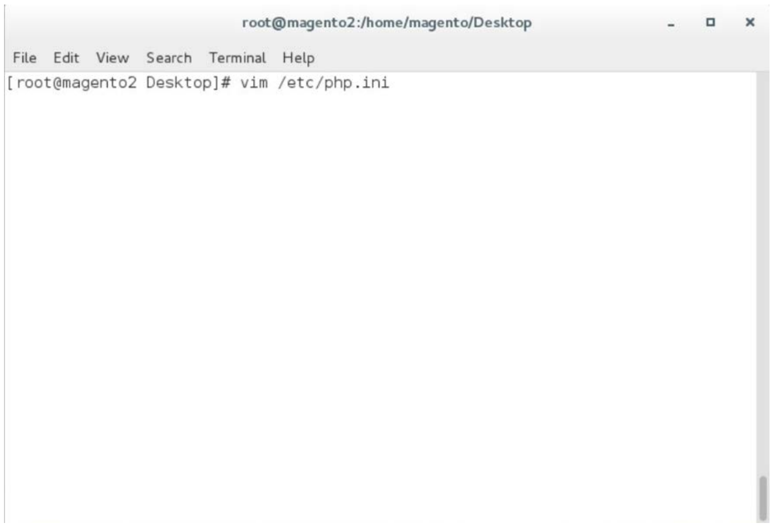
```
<Directory "/var/www">
    AllowOverride None
    # Allow open access:
    Require all granted
</Directory>

# Further relax access to the default document root:
<Directory "/var/www/html">
    #
    # Possible values for the Options directive are "None", "All",
    # or any combination of:
    #   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
    #
    # Note that "MultiViews" must be named *explicitly* --- "Options All"
    # doesn't give it to you.
    #
    # The Options directive is both complicated and important.  Please see
    # http://httpd.apache.org/docs/2.4/mod/core.html#options
    # for more information.
    #
    Options Indexes FollowSymLinks

    #
    # AllowOverride controls what directives may be placed in .htaccess files.
    # It can be "All", "None", or any combination of the keywords:
    #   Options FileInfo AuthConfig Limit
    #
    AllowOverride All

    #
    # Controls who can get stuff from this server.
    #
    Require all granted
</Directory>
```

22. Save, and exit by entering the following command:

    ```
    :wq!
    ```

23. Open *php.ini* to modify PHP settings by entering the following command:

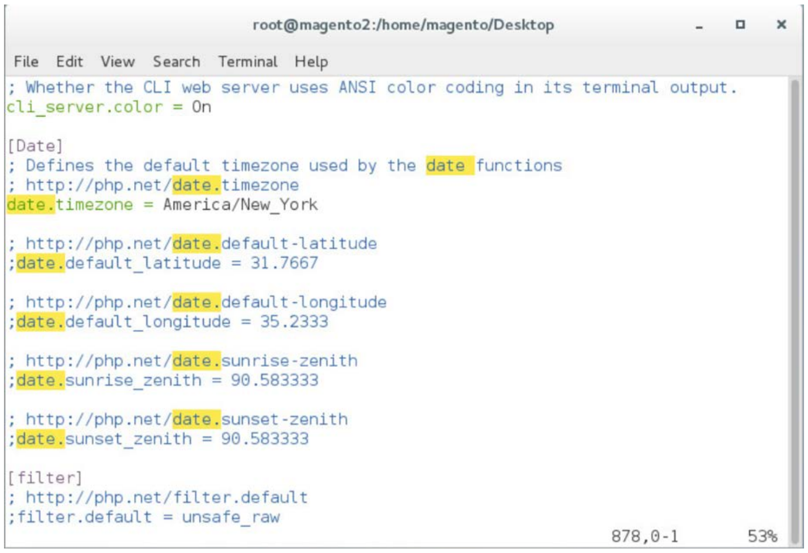
    ```
    vim /etc/php.ini
    ```


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
[root@magento2 Desktop]# vim /etc/php.ini
```

24. Uncomment the line containing `date.timezone` by removing the “;” character preceding the text, and enter your time zone as shown below (this example is for the eastern United States).

```
date.timezone = America/New_York
```

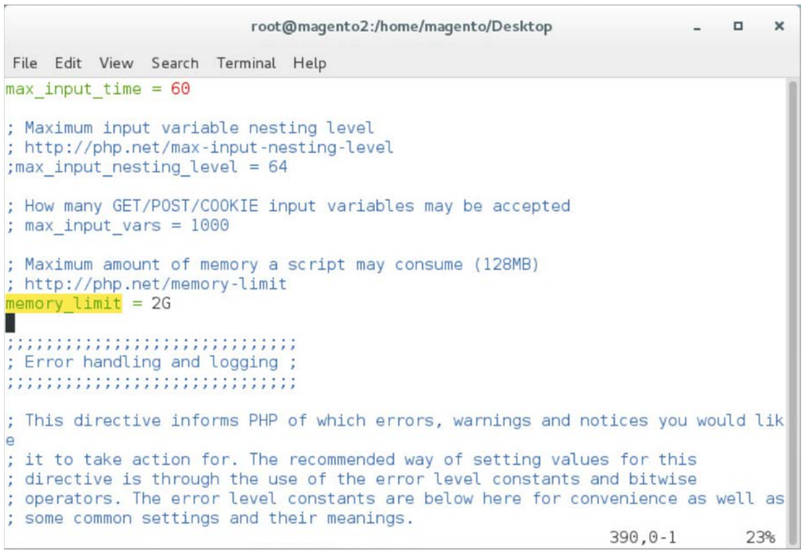


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
; Whether the CLI web server uses ANSI color coding in its terminal output.
cli_server.color = On

[Date]
; Defines the default timezone used by the date functions
; http://php.net/date.timezone
date.timezone = America/New_York

; http://php.net/date.default-latitude
;date.default_latitude = 31.7667

; http://php.net/date.default-longitude
;date.default_longitude = 35.2333

; http://php.net/date.sunrise-zenith
;date.sunrise_zenith = 90.583333

; http://php.net/date.sunset-zenith
;date.sunset_zenith = 90.583333

[filter]
; http://php.net/filter.default
;filter.default = unsafe_raw
                                                            878,0-1       53%
```

25. Uncomment the line containing `memory_limit` by removing the “;” character preceding the text, and enter 2G as the value, as shown below.

```
memory_limit = 2G
```

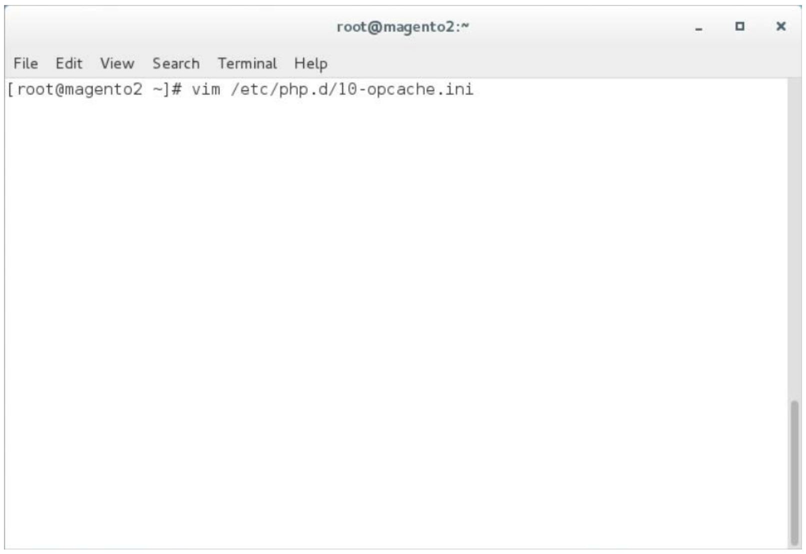


root@magento2:/home/magento/Desktop
File Edit View Search Terminal Help

```
max_input_time = 60

; Maximum input variable nesting level
; http://php.net/max-input-nesting-level
;max_input_nesting_level = 64

; How many GET/POST/COOKIE input variables may be accepted
; max_input_vars = 1000

; Maximum amount of memory a script may consume (128MB)
; http://php.net/memory-limit
memory_limit = 2G

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Error handling and logging ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; This directive informs PHP of which errors, warnings and notices you would lik
e
; it to take action for. The recommended way of setting values for this
; directive is through the use of the error level constants and bitwise
; operators. The error level constants are below here for convenience as well as
; some common settings and their meanings.
                                                            390,0-1       23%
```

26. Open *10-opcache.ini* to modify PHP settings by entering the following command:

```
vim /etc/php.d/10-opcache.ini
```



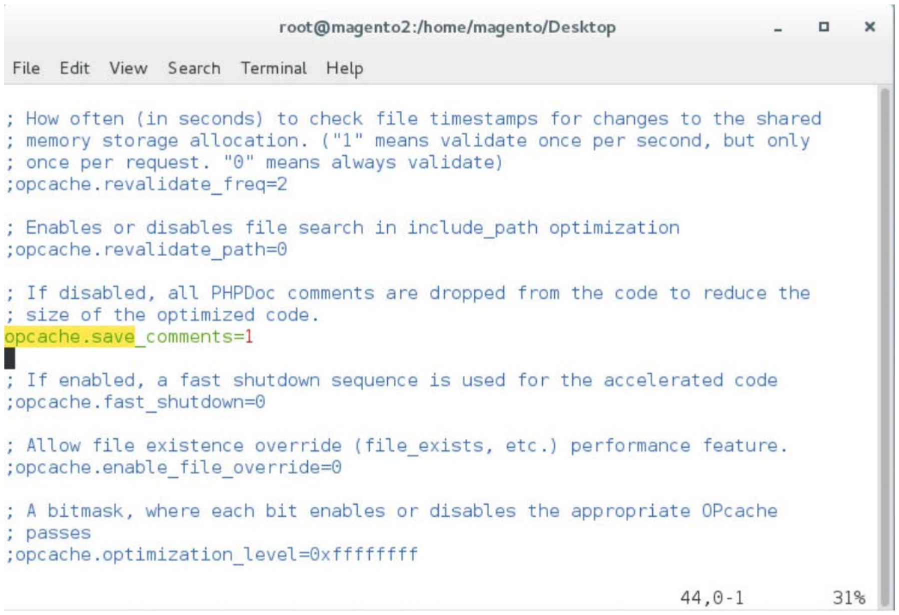
root@magento2:~

File Edit View Search Terminal Help

```
[root@magento2 ~]# vim /etc/php.d/10-opcache.ini
```

27. Uncomment the line containing `opcache.save_comments` by removing the ";" character preceding the text. The line should then read as shown below.

    ```
    opcache.save_comments=1
    ```



root@magento2:/home/magento/Desktop

File Edit View Search Terminal Help

```
; How often (in seconds) to check file timestamps for changes to the shared
; memory storage allocation. ("1" means validate once per second, but only
; once per request. "0" means always validate)
;opcache.revalidate_freq=2

; Enables or disables file search in include_path optimization
;opcache.revalidate_path=0

; If disabled, all PHPDoc comments are dropped from the code to reduce the
; size of the optimized code.
opcache.save_comments=1

; If enabled, a fast shutdown sequence is used for the accelerated code
;opcache.fast_shutdown=0

; Allow file existence override (file_exists, etc.) performance feature.
;opcache.enable_file_override=0

; A bitmask, where each bit enables or disables the appropriate OPcache
; passes
;opcache.optimization_level=0xffffffff
                                                              44,0-1        31%
```

2.2.4 Magento Installation

For the e-commerce platform, Magento Open Source Version 2.1.8 [5] was used in the example implementation.

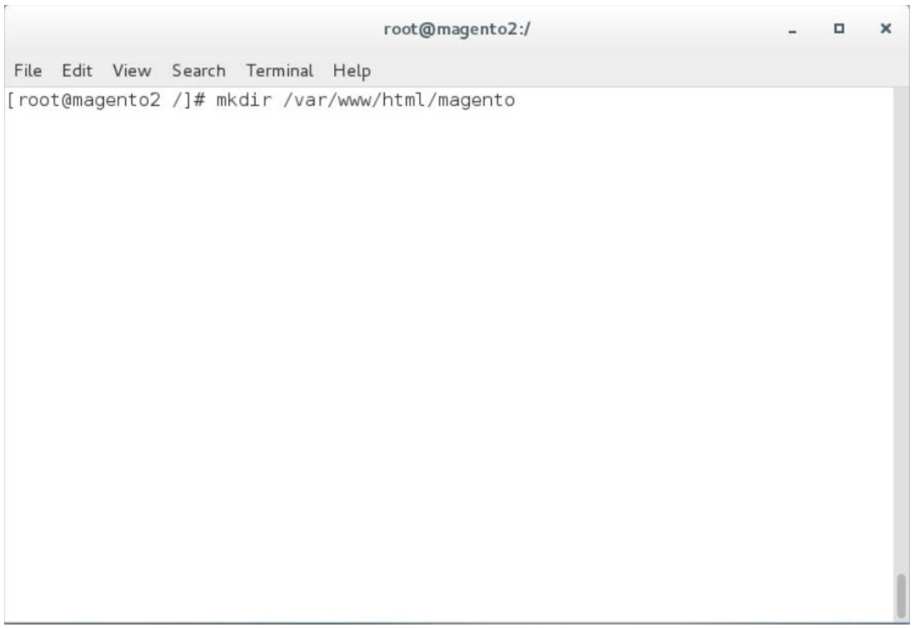
To download the open-source copy of Magento, navigate to the site: https://magento.com/products/open-source.

When redirected to the resource page, specify the download format. In the example implementation, we installed Magento on CentOS by selecting a file that ends in `.gz`, as shown in the example below.

```
Magento-Community-Edition-2.1.8.tar.gz
```

1. Create a Magento directory inside HTTPD's DocumentRoot folder by entering the following command:

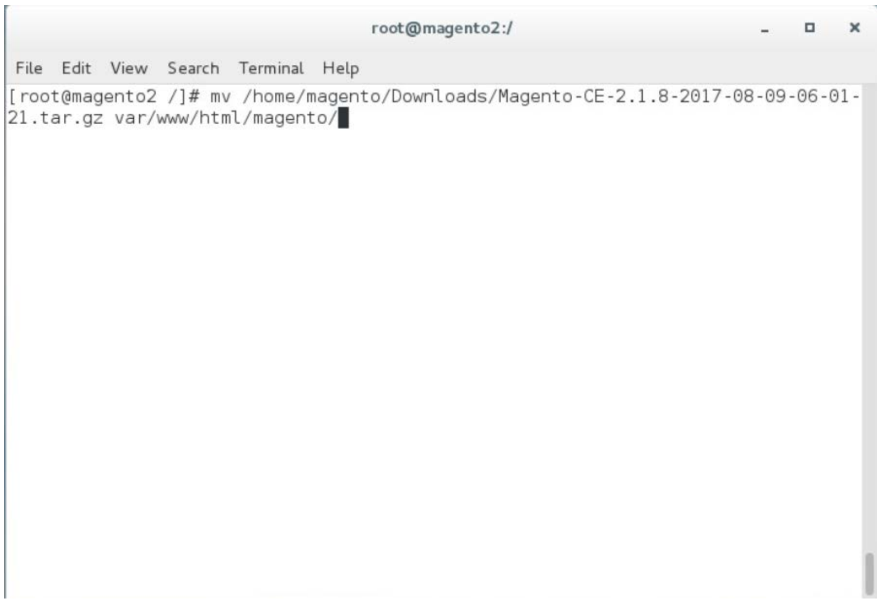
   ```
   mkdir /var/www/html/magento
   ```



```
[root@magento2 /]# mkdir /var/www/html/magento
```

2. Move the *Magento-CE-2.1.8.tar.gz* into the Magento directory with the following command:

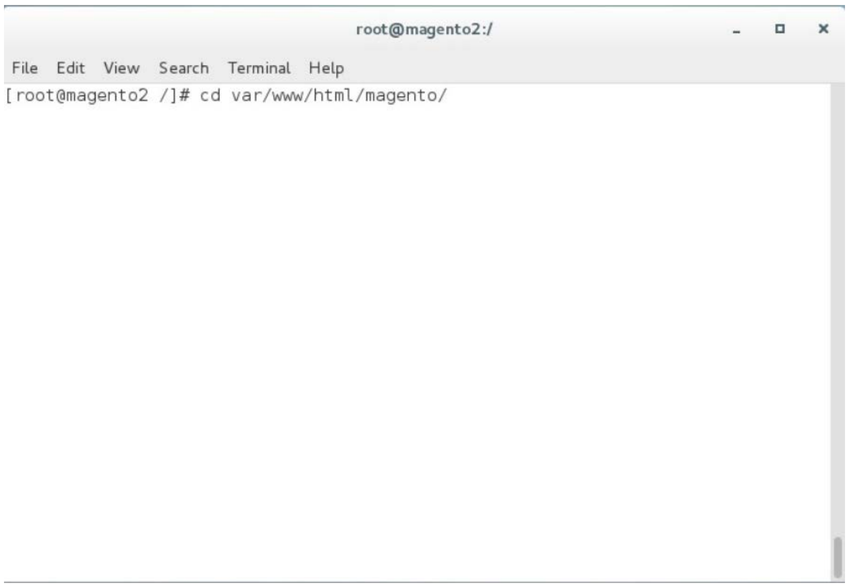
   ```
   mv <download location>/Magento-CE-2.1.8-2017-08-09-96-91-21.tar.gz
   /var/www/html/magento
   ```



```
[root@magento2 /]# mv /home/magento/Downloads/Magento-CE-2.1.8-2017-08-09-06-01-
21.tar.gz var/www/html/magento/
```

3. Change the directory to the Magento directory by entering the following command (all commands following this step should be run from this directory):

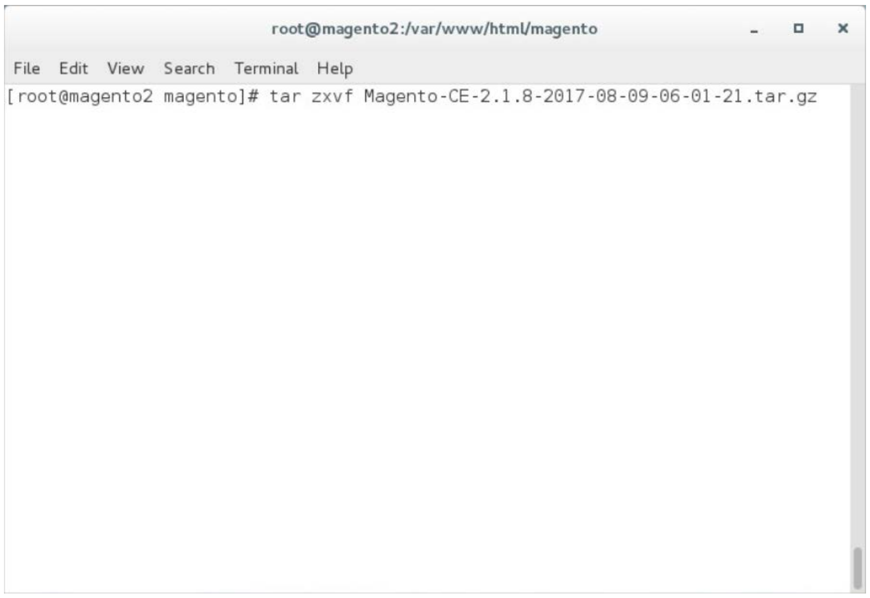
   ```
   cd /var/www/html/magento
   ```



```
[root@magento2 /]# cd var/www/html/magento/
```

4. Extract the Magento distribution from *Magento-CE-2.1.8.tar.gz* by entering the following command:

   ```
   tar zxvf Magento-CE-2.1.8-2017-08-09-96-91-21.tar.gz
   ```

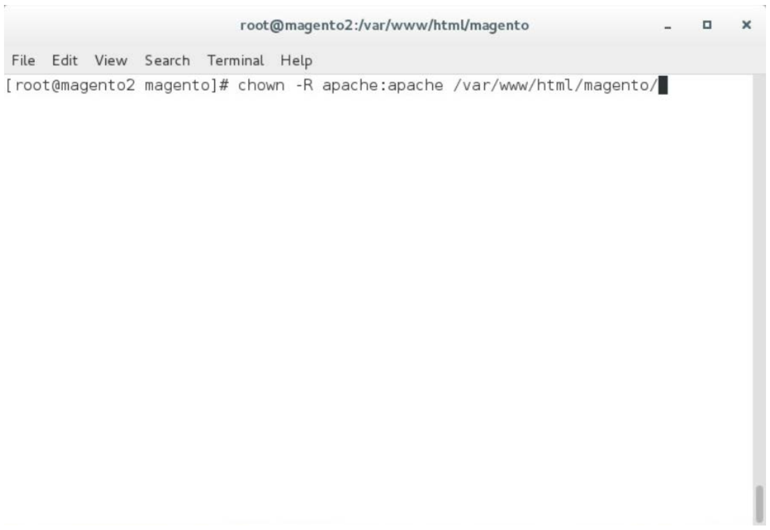


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# tar zxvf Magento-CE-2.1.8-2017-08-09-06-01-21.tar.gz
```

5. Change ownership of the extracted files to the Apache user by entering the following command:

   ```
   chown -R apache:apache /var/www/html/magento
   ```

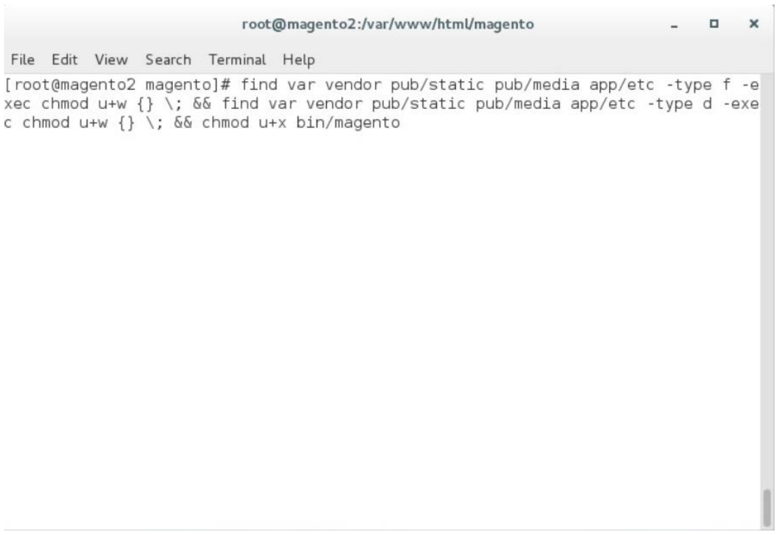


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# chown -R apache:apache /var/www/html/magento/
```

6. Change file permissions by entering the following command (Note: This is a single command that must be executed on a single line.):

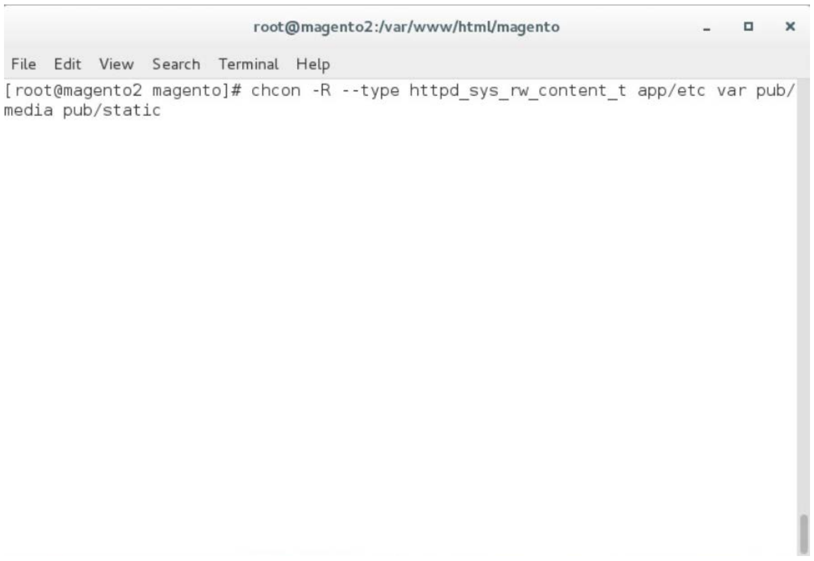
   ```
   find var vendor pub/static pub/media app/etc -type f -exec chmod u+w {} \; &&
   find var vendor pub/static pub/media app/etc -type d -exec chmod u+w {} \; &&
   chmod u+x bin/magento
   ```


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# find var vendor pub/static pub/media app/etc -type f -e
xec chmod u+w {} \; && find var vendor pub/static pub/media app/etc -type d -exe
c chmod u+w {} \; && chmod u+x bin/magento
```

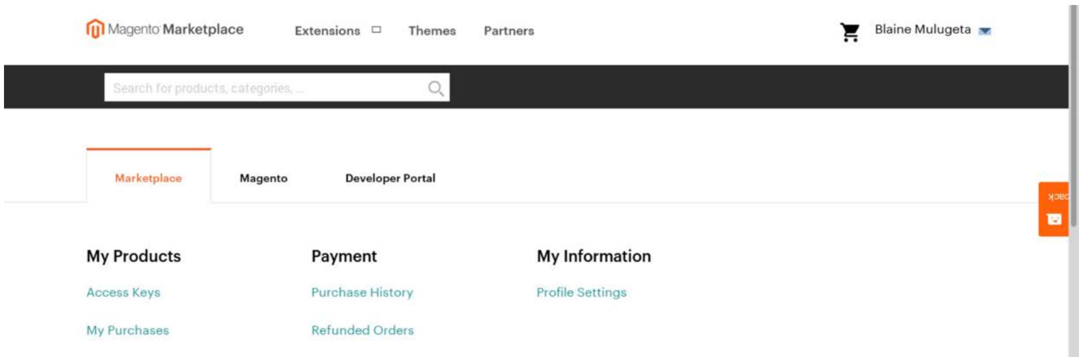
7. Change the Security-Enhanced Linux (SELinux) context permissions to allow the Apache user to have read/write access to specific directories within the Magento directory by entering the following command:

   ```
   chcon -R --type httpd_sys_rw_content_t app/etc var pub/media pub/static
   ```

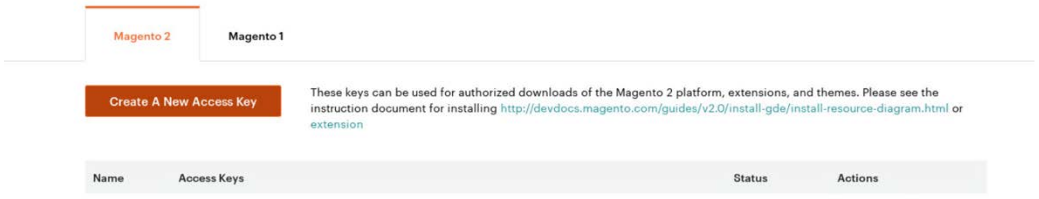

root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# chcon -R --type httpd_sys_rw_content_t app/etc var pub/
media pub/static
```

8. Open the web browser to log in to https://marketplace.magento.com and access your account. Click **Access Keys.**



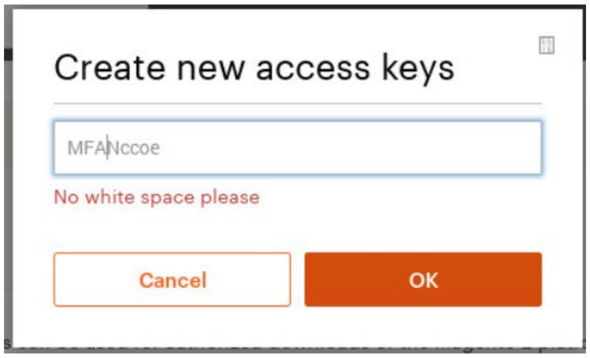
9. In the Magento tab, click **Create A New Access Key.**



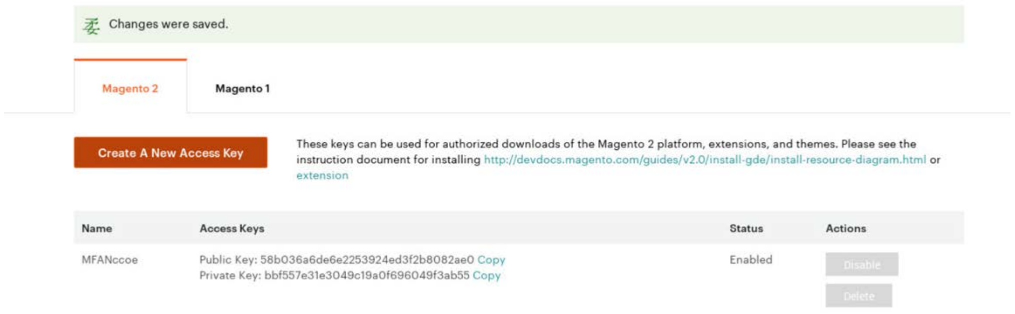
These keys can be used for authorized downloads of the Magento 2 platform, extensions, and themes. Please see the instruction document for installing http://devdocs.magento.com/guides/v2.0/install-gde/install-resource-diagram.html or extension

Name	Access Keys	Status	Actions

10. Enter a name for your new access key, and click **OK.**



11. The new access keys will be displayed in the menu with the **Status** of **Enabled.**

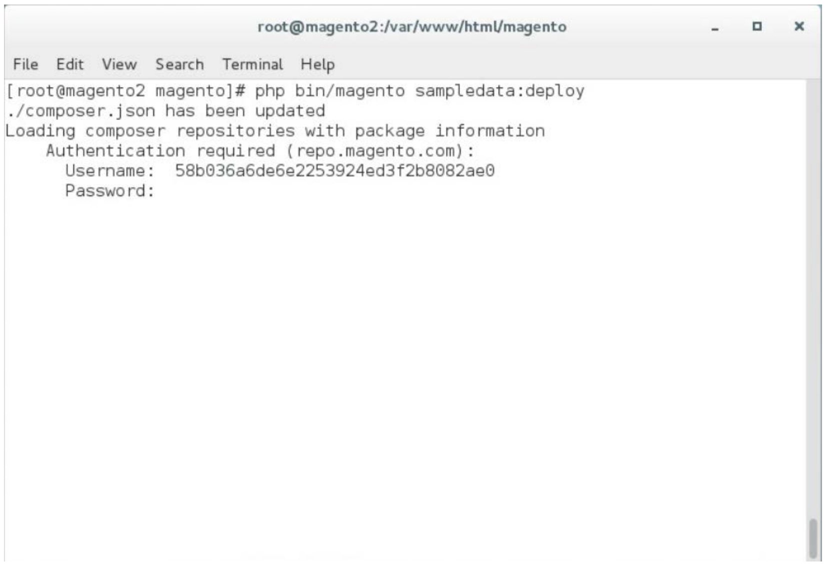


These keys can be used for authorized downloads of the Magento 2 platform, extensions, and themes. Please see the instruction document for installing http://devdocs.magento.com/guides/v2.0/install-gde/install-resource-diagram.html or extension

Name	Access Keys	Status	Actions
MFANccoe	Public Key: 58b036a6de6e2253924ed3f2b8082ae0 Copy Private Key: bbf557e31e3049c19a0f696049f3ab55 Copy	Enabled	Disable Delete

12. Install Magento's sample data by entering the following command and then providing <public key> when a **Username** is requested and <private key> as the **Password** when prompted:

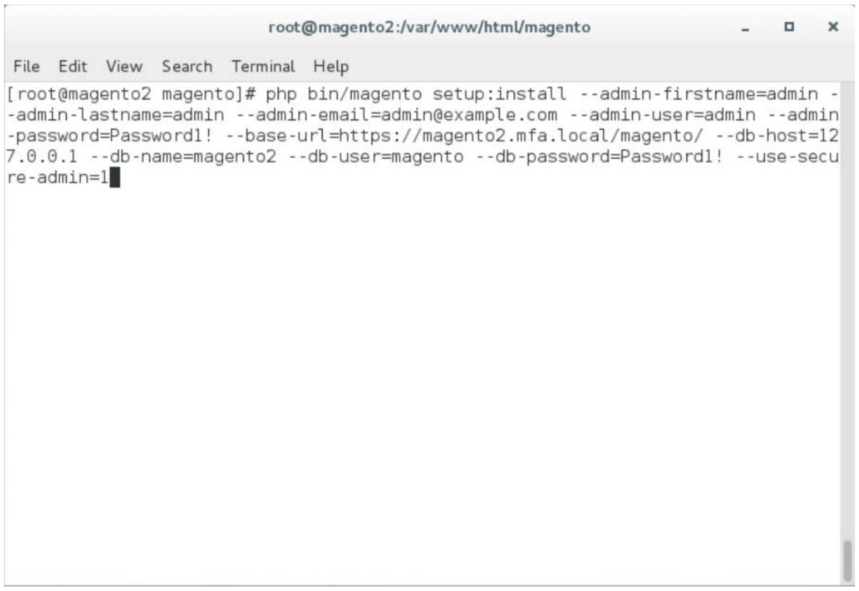
```
php bin/magento sampledata:deploy
```


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# php bin/magento sampledata:deploy
./composer.json has been updated
Loading composer repositories with package information
    Authentication required (repo.magento.com):
      Username:  58b036a6de6e2253924ed3f2b8082ae0
      Password:
```

13. Install the Magento software distribution by issuing the following command, replacing parameters in <> with values appropriate for your installation (Note: This is a single command that must be executed on a single line.):

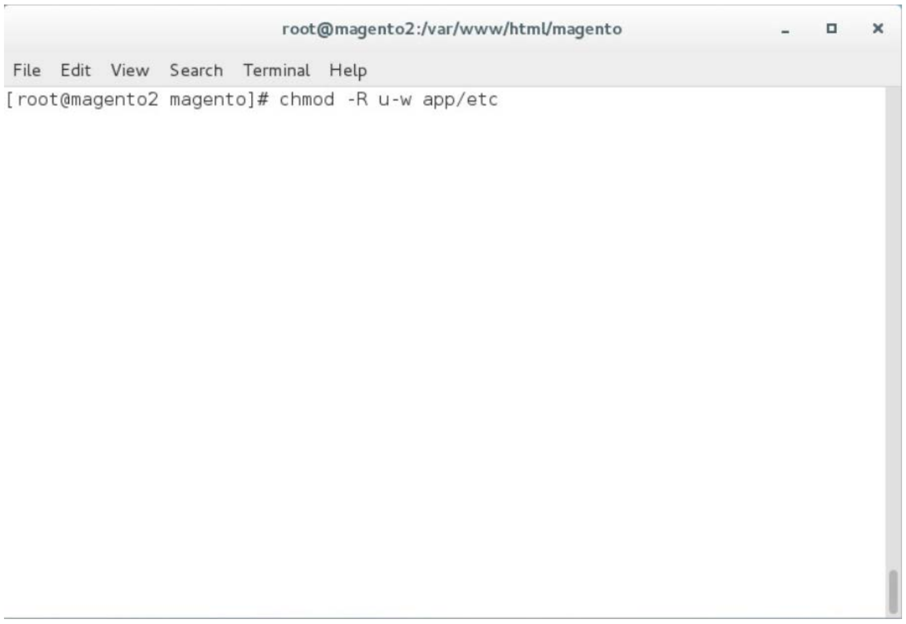
```
php bin/magento setup:install --admin-firstname=<First Name> --admin-
lastname=<Last Name> --admin-email=<email> --admin-user=strongauth --admin-
password=<password> --base-url=https://<fully-qualified-domainname>/magento/ --
db-host=127.0.01 --db-name=magento2 --db-user=magento --db-password=<db
password> --use-secure-admin=1
```


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# php bin/magento setup:install --admin-firstname=admin -
-admin-lastname=admin --admin-email=admin@example.com --admin-user=admin --admin
-password=Password1! --base-url=https://magento2.mfa.local/magento/ --db-host=12
7.0.0.1 --db-name=magento2 --db-user=magento --db-password=Password1! --use-secu
re-admin=1
```

14. Modify compiled file permissions by issuing the following command:

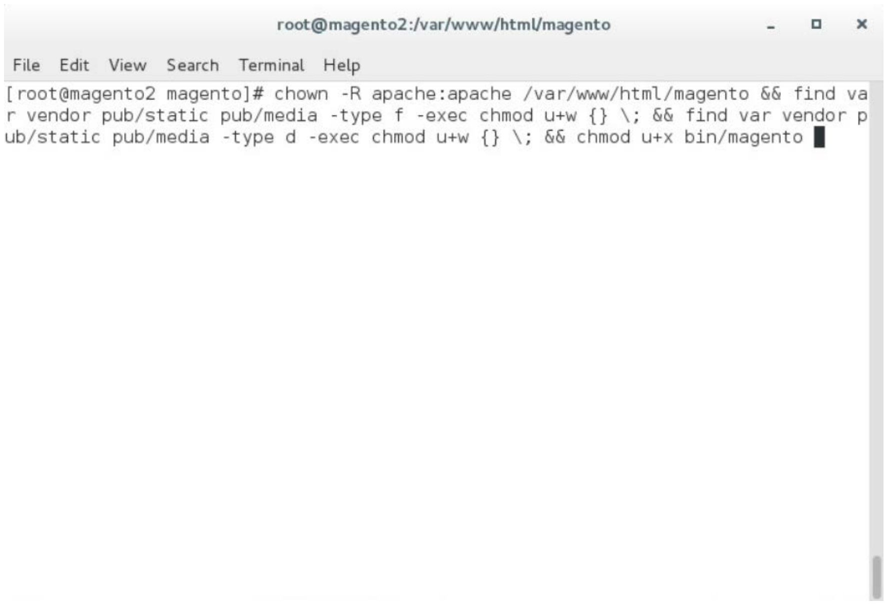
    ```
    chmod -R u-w app/etc
    ```


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# chmod -R u-w app/etc
```

15. Modify compiled file permissions by issuing the following command:

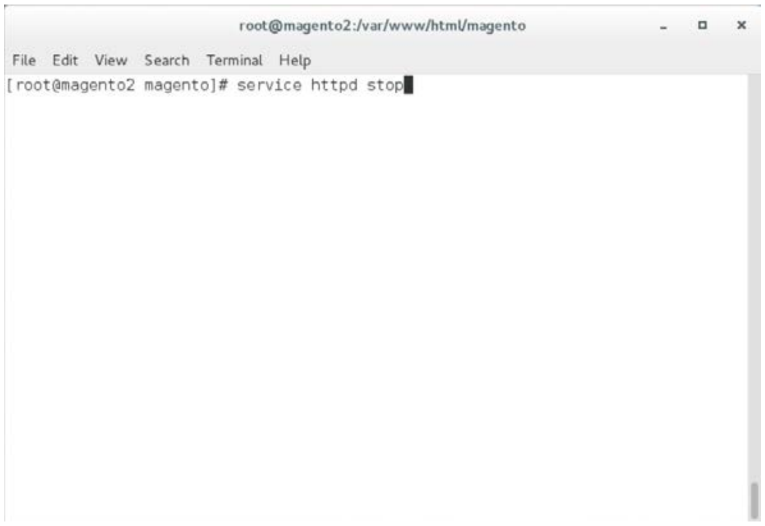
    ```
    chown -R apache:apache /var/www/html/magento && find var vendor pub/static
    pub/media -type f -exec chmod u+w {} \; && find var vendor pub/static pub/media
    -type d -exec chmod u+w {} \; && chmod u+x bin/magento
    ```


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# chown -R apache:apache /var/www/html/magento && find va
r vendor pub/static pub/media -type f -exec chmod u+w {} \; && find var vendor p
ub/static pub/media -type d -exec chmod u+w {} \; && chmod u+x bin/magento
```

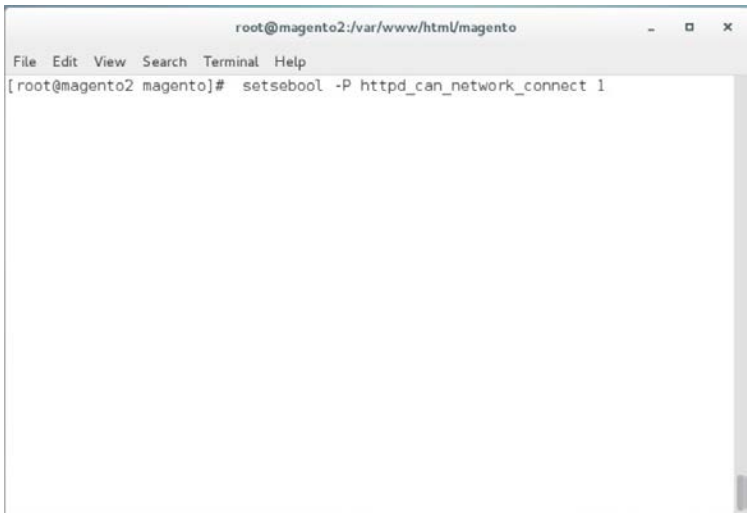
16. Modify SELinux permissions, to enable HTTPD to access the database, by executing the following commands:

    a. `service httpd stop`


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

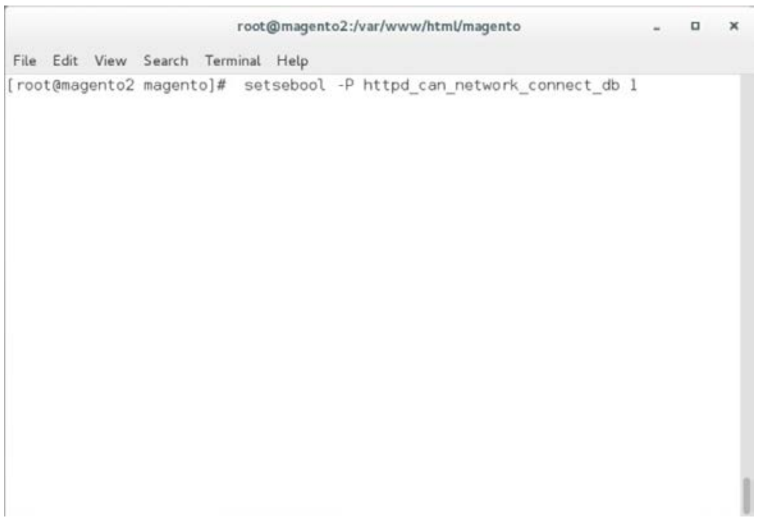
```
[root@magento2 magento]# service httpd stop
```

    b. `setsebool -P httpd_can_network_connect 1`


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

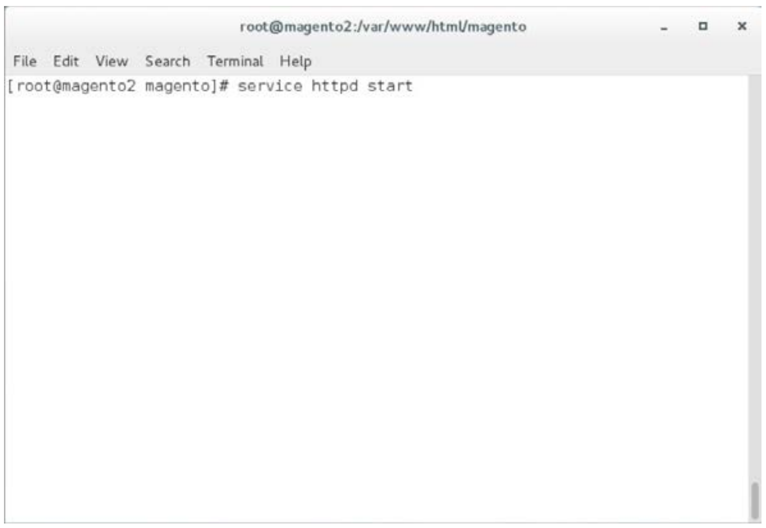
```
[root@magento2 magento]#  setsebool -P httpd_can_network_connect 1
```

c. `setsebool -P httpd_can_network_connect_db 1`


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

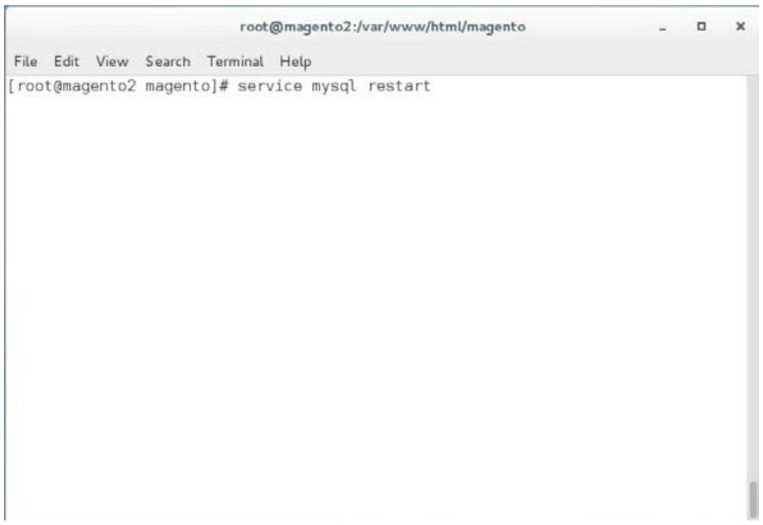
```
[root@magento2 magento]#  setsebool -P httpd_can_network_connect_db 1
```

d. `service httpd start`


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

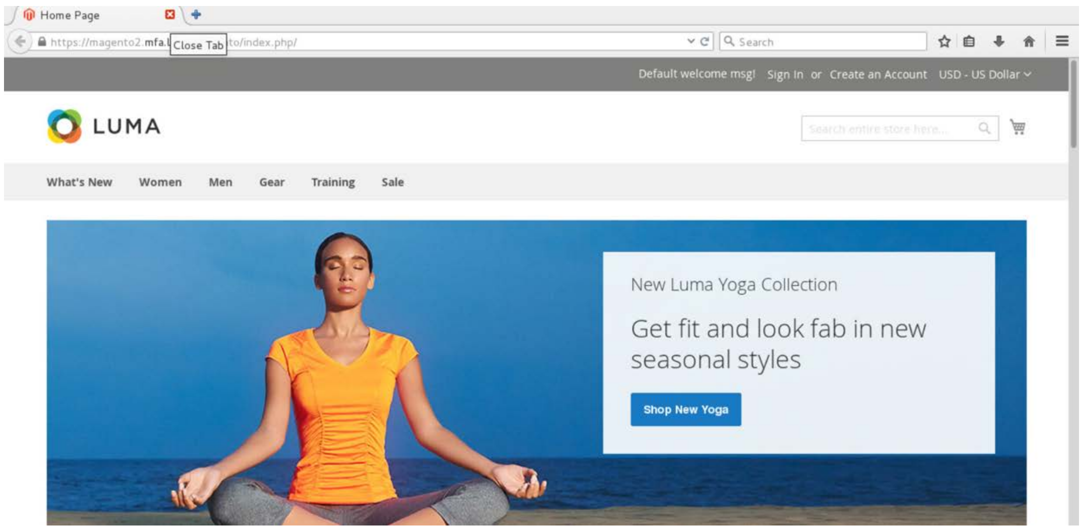
```
[root@magento2 magento]# service httpd start
```

e. `service mysql restart`


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# service mysql restart
```

17. Verify the installation by navigating in the browser to the store URL, which was set up in Section 2.2.4, Step 13 (*https://magento2.mfa.local/magento*).

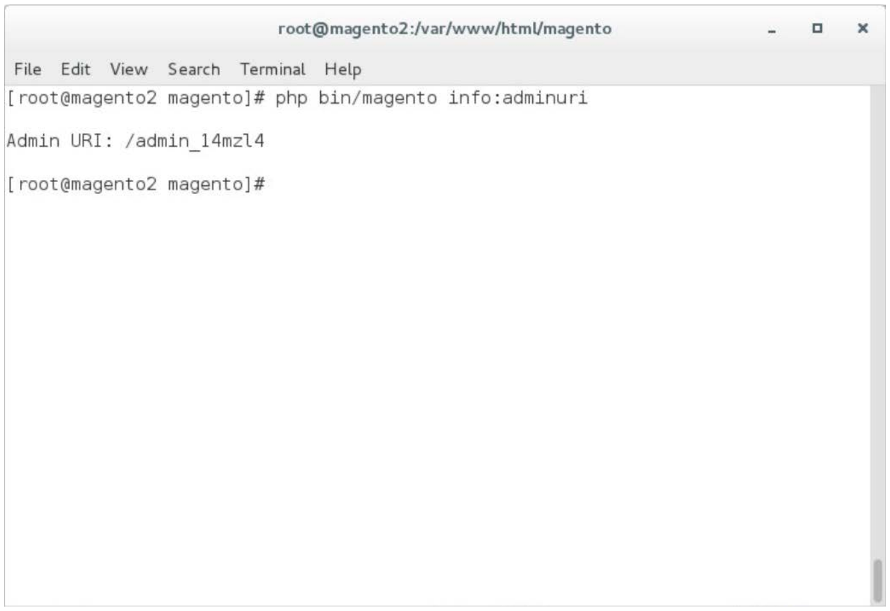

Home Page
https://magento2.mfa.l Close Tab to/index.php/
Search
Default welcome msg! Sign In or Create an Account USD - US Dollar
LUMA
Search entire store here...
What's New
Women
Men
Gear
Training
Sale
New Luma Yoga Collection
Get fit and look fab in new seasonal styles
Shop New Yoga

2.2.5 Configuring the Magento Account Lockout Feature

This section describes the steps required to configure account lockouts after a specified number of failed login attempts. For our example implementation, we specified five as the maximum number of login-attempt failures before temporarily disabling the account, and 20 minutes as the lockout time. These parameters can be adjusted, and the administrator of the Magento site has the information system privileges to set these values based on the implementer’s preference.

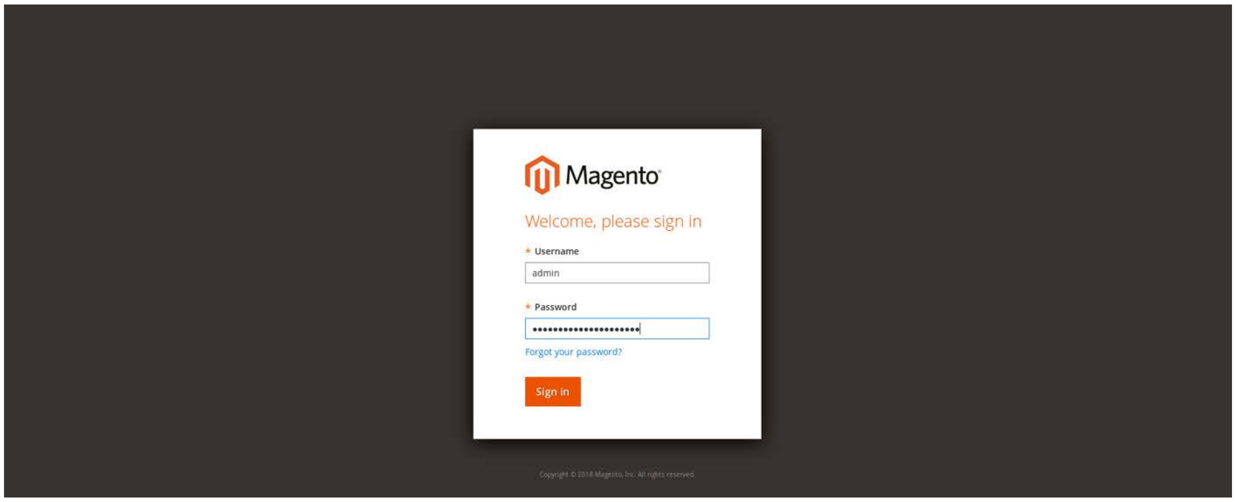
1. Determine the admin uniform resource identifier (URI) by running the following command:

   ```
   php bin/magento info:adminuri
   ```

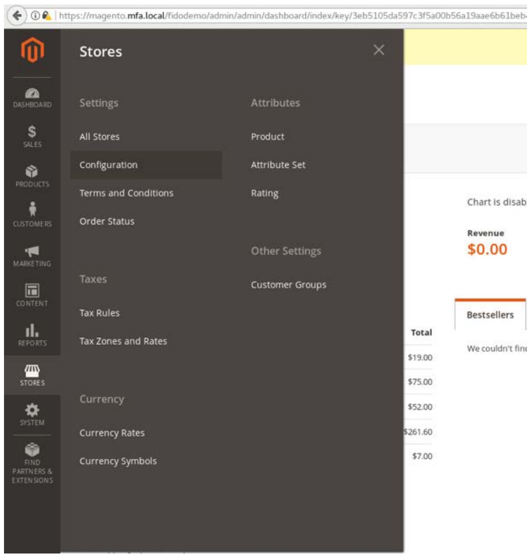


```
[root@magento2 magento]# php bin/magento info:adminuri

Admin URI: /admin_14mzl4

[root@magento2 magento]#
```

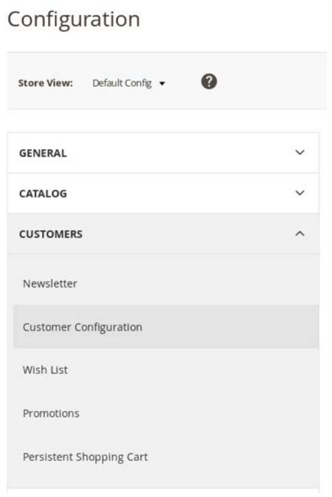
2. Navigate to the admin URI identified in Section 2.2.5, Step 1, and sign in with the Magento **Username** and **Password** created in Section 2.2.4, Step 13 (the example implementation URI is *https://magento2.mfa.local/admin_14mzl4*).



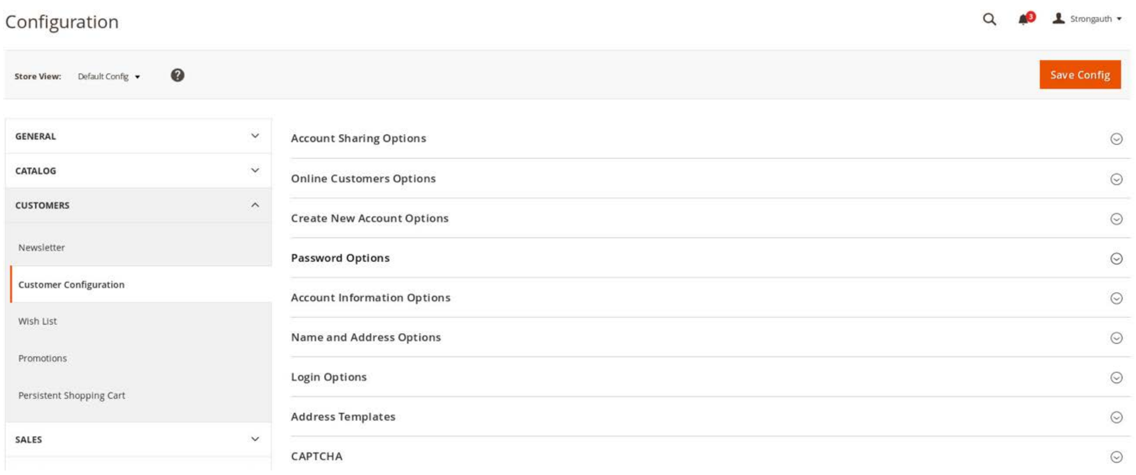
3. Proceed to the Configuration page: **STORES > Configuration.**



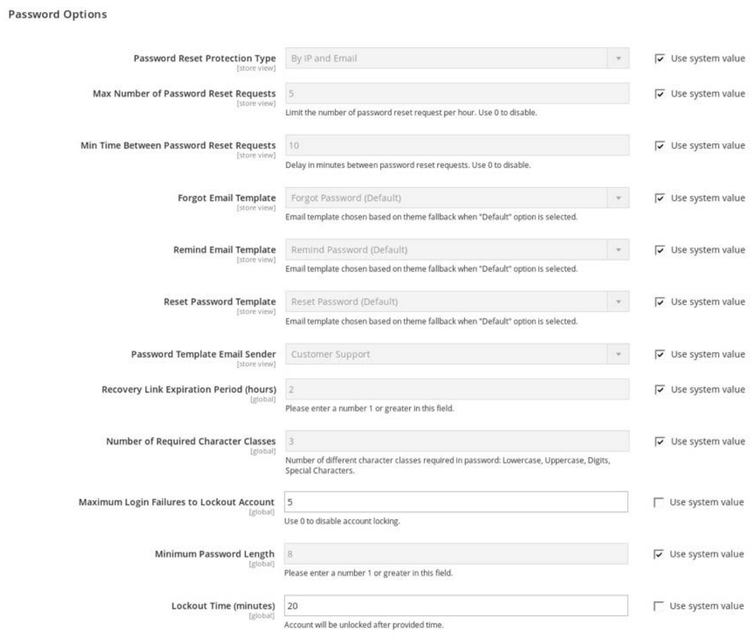
4. Click the **CUSTOMERS** drop-down from the menu in the **Configuration** page, and select **Customer Configuration.**



5. Click the **Password Options** drop-down.

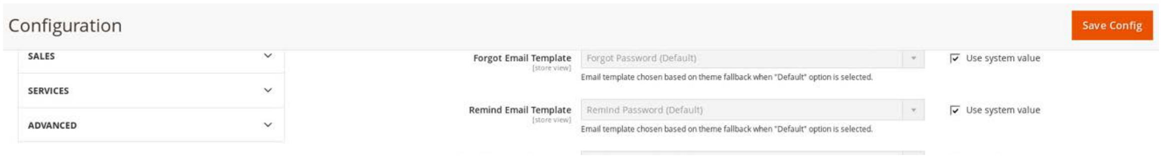


6. Uncheck the **Use system value** fields for the **Maximum Login Failures to Lockout Account** and **Lockout Time (minutes)** to modify the settings for the **Password Options.**


Password Options

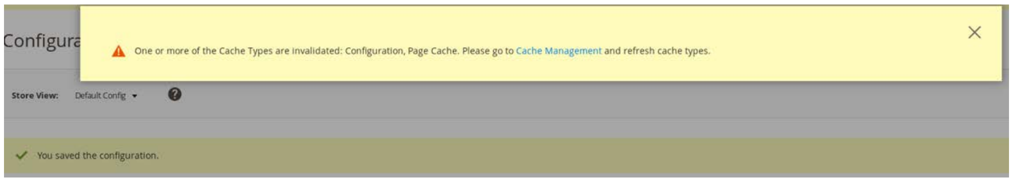
Password Reset Protection Type [store view]	By IP and Email	☑ Use system value
Max Number of Password Reset Requests [store view]	5 Limit the number of password reset request per hour. Use 0 to disable.	☑ Use system value
Min Time Between Password Reset Requests [store view]	10 Delay in minutes between password reset requests. Use 0 to disable.	☑ Use system value
Forgot Email Template [store view]	Forgot Password (Default) Email template chosen based on theme fallback when "Default" option is selected.	☑ Use system value
Remind Email Template [store view]	Remind Password (Default) Email template chosen based on theme fallback when "Default" option is selected.	☑ Use system value
Reset Password Template [store view]	Reset Password (Default) Email template chosen based on theme fallback when "Default" option is selected.	☑ Use system value
Password Template Email Sender [store view]	Customer Support	☑ Use system value
Recovery Link Expiration Period (hours) [global]	2 Please enter a number 1 or greater in this field.	☑ Use system value
Number of Required Character Classes [global]	3 Number of different character classes required in password: Lowercase, Uppercase, Digits, Special Characters.	☑ Use system value
Maximum Login Failures to Lockout Account [global]	5 Use 0 to disable account locking.	☐ Use system value
Minimum Password Length [global]	8 Please enter a number 1 or greater in this field.	☑ Use system value
Lockout Time (minutes) [global]	20 Account will be unlocked after provided time.	☐ Use system value

7. Click **Save Config** to save the changes made.



Forgot Email Template [store view]	Forgot Password (Default) Email template chosen based on theme fallback when "Default" option is selected.	☑ Use system value
Remind Email Template [store view]	Remind Password (Default) Email template chosen based on theme fallback when "Default" option is selected.	☑ Use system value

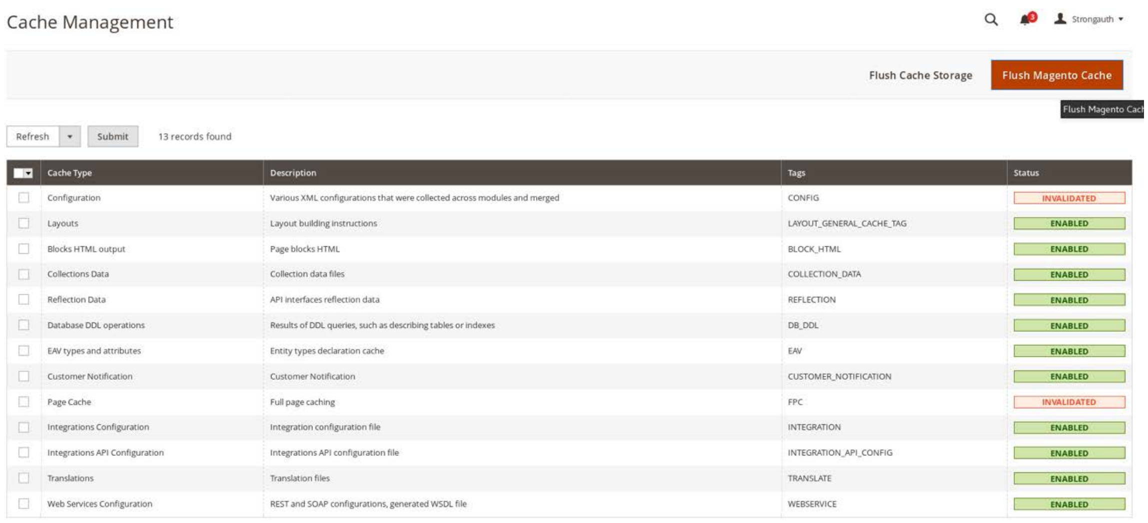
8. The following pop-up will appear, notifying you to refresh Cache Types. Click the **Cache Management** link in the message.



One or more of the Cache Types are invalidated: Configuration, Page Cache. Please go to Cache Management and refresh cache types.

You saved the configuration.

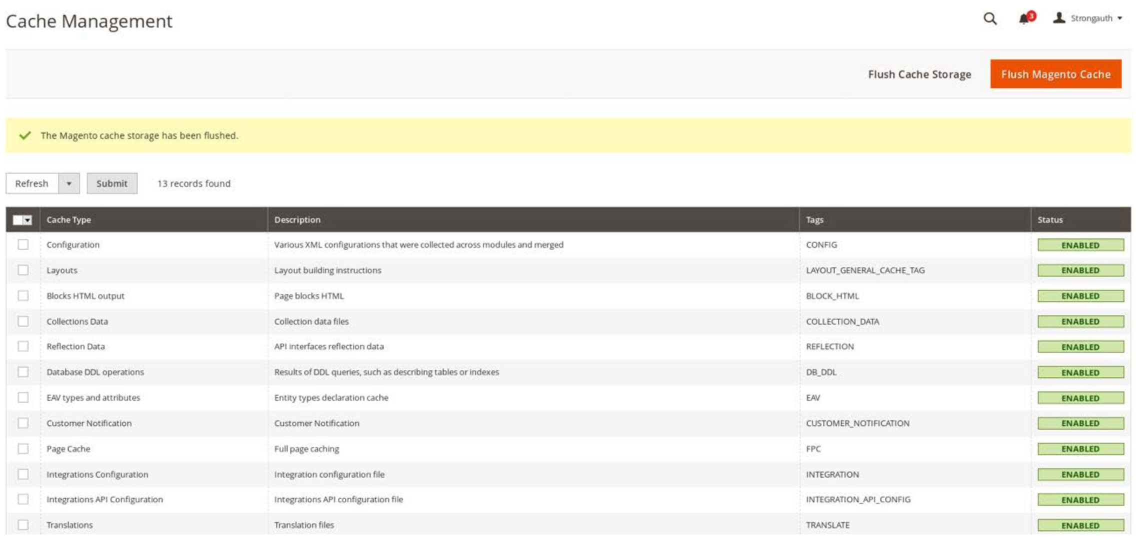
9. You will be redirected to the **Cache Management** page. Click **Flush Magento Cache** to resolve the **INVALIDATED** Cache Types.



13 records found

Cache Type	Description	Tags	Status
Configuration	Various XML configurations that were collected across modules and merged	CONFIG	INVALIDATED
Layouts	Layout building instructions	LAYOUT_GENERAL_CACHE_TAG	ENABLED
Blocks HTML output	Page blocks HTML	BLOCK_HTML	ENABLED
Collections Data	Collection data files	COLLECTION_DATA	ENABLED
Reflection Data	API interfaces reflection data	REFLECTION	ENABLED
Database DDL operations	Results of DDL queries, such as describing tables or indexes	DB_DDL	ENABLED
EAV types and attributes	Entity types declaration cache	EAV	ENABLED
Customer Notification	Customer Notification	CUSTOMER_NOTIFICATION	ENABLED
Page Cache	Full page caching	FPC	INVALIDATED
Integrations Configuration	Integration configuration file	INTEGRATION	ENABLED
Integrations API Configuration	Integrations API configuration file	INTEGRATION_API_CONFIG	ENABLED
Translations	Translation files	TRANSLATE	ENABLED
Web Services Configuration	REST and SOAP configurations, generated WSDL file	WEBSERVICE	ENABLED

10. Upon completion of the flush, the page will reflect the changes.



The Magento cache storage has been flushed.

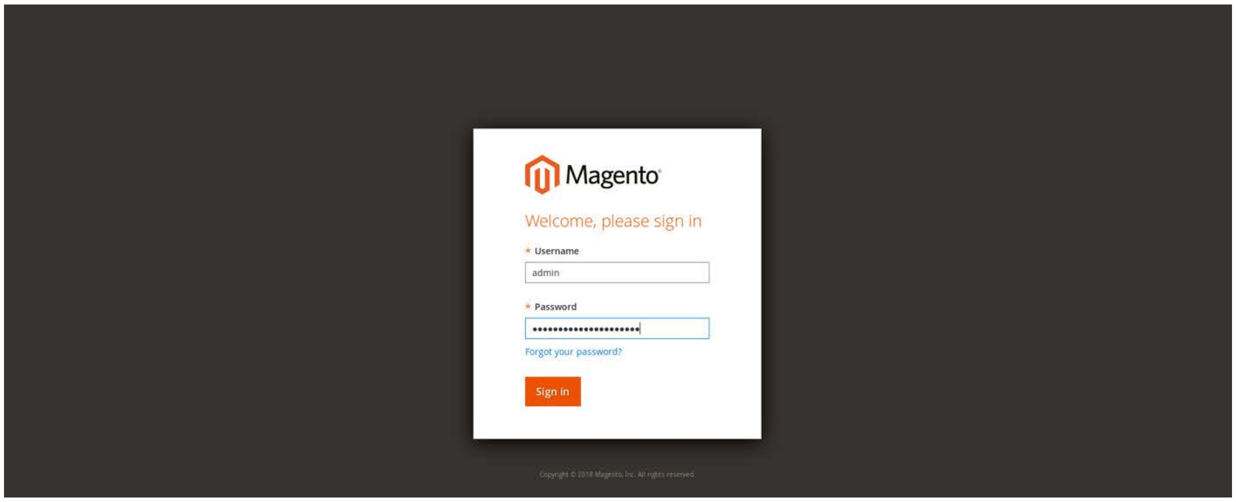
13 records found

Cache Type	Description	Tags	Status
Configuration	Various XML configurations that were collected across modules and merged	CONFIG	ENABLED
Layouts	Layout building instructions	LAYOUT_GENERAL_CACHE_TAG	ENABLED
Blocks HTML output	Page blocks HTML	BLOCK_HTML	ENABLED
Collections Data	Collection data files	COLLECTION_DATA	ENABLED
Reflection Data	API interfaces reflection data	REFLECTION	ENABLED
Database DDL operations	Results of DDL queries, such as describing tables or indexes	DB_DDL	ENABLED
EAV types and attributes	Entity types declaration cache	EAV	ENABLED
Customer Notification	Customer Notification	CUSTOMER_NOTIFICATION	ENABLED
Page Cache	Full page caching	FPC	ENABLED
Integrations Configuration	Integration configuration file	INTEGRATION	ENABLED
Integrations API Configuration	Integrations API configuration file	INTEGRATION_API_CONFIG	ENABLED
Translations	Translation files	TRANSLATE	ENABLED

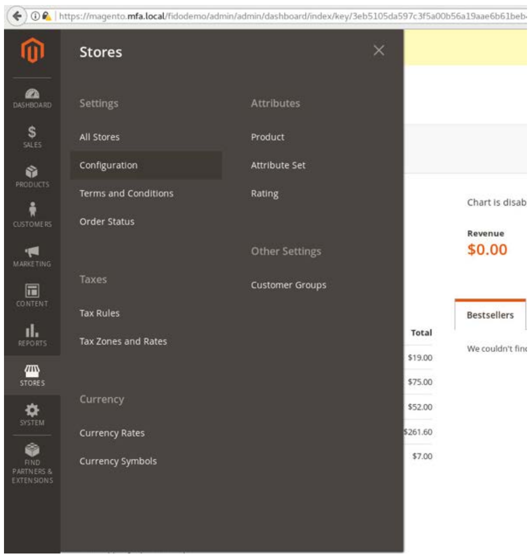
2.2.6 Disabling Magento Guest Checkout

This section describes steps to disable Magento's guest checkout feature to ensure that purchasers cannot choose to check out as a guest.

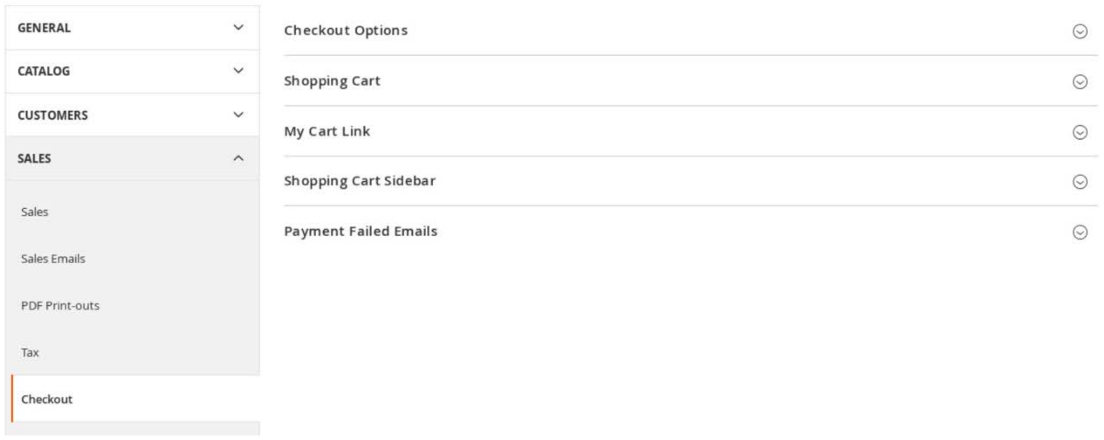
1. Navigate to the admin URI identified in Section 2.2.5, Step 1 (*https://magento2.mfa.local/admin_14mzl4*), and sign in with the **Username** and **Password** created in Section 2.2.4, Step 13.



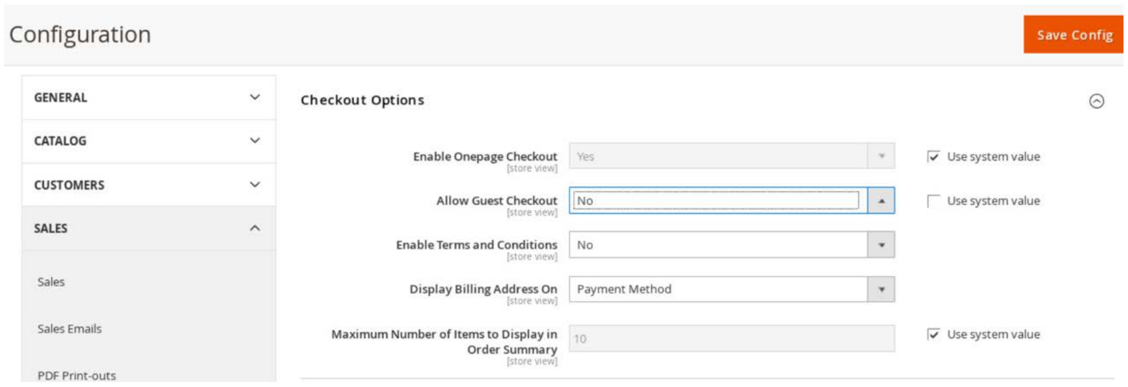
2. Proceed to the **Configuration** page: **STORES > Configuration.**



3. Click the **SALES** drop-down from the menu on the **Configuration** page, select **Checkout**, and expand the **Checkout Options.**



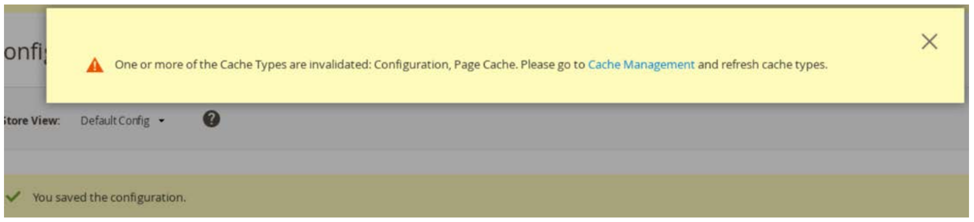
4. Uncheck the **Use system value** fields for the **Allow Guest Checkout** setting, and modify the settings to **No** for the **Checkout Options.**



Enable Onepage Checkout [store view]	Yes	☑ Use system value
Allow Guest Checkout [store view]	No	☐ Use system value
Enable Terms and Conditions [store view]	No	
Display Billing Address On [store view]	Payment Method	
Maximum Number of Items to Display in Order Summary [store view]	10	☑ Use system value

5. Click **Save Config.**

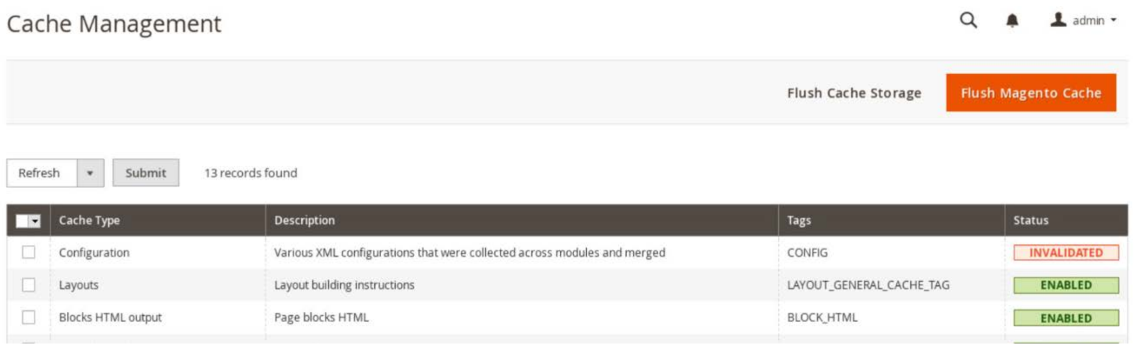
6. The following pop-up will appear, notifying you to refresh Cache Types. Click the **Cache Management** link in the message.



One or more of the Cache Types are invalidated: Configuration, Page Cache. Please go to Cache Management and refresh cache types.

You saved the configuration.

7. You will be redirected to the **Cache Management** page. Click **Flush Magento Cache** to resolve the **INVALIDATED** Cache Types.


Cache Management

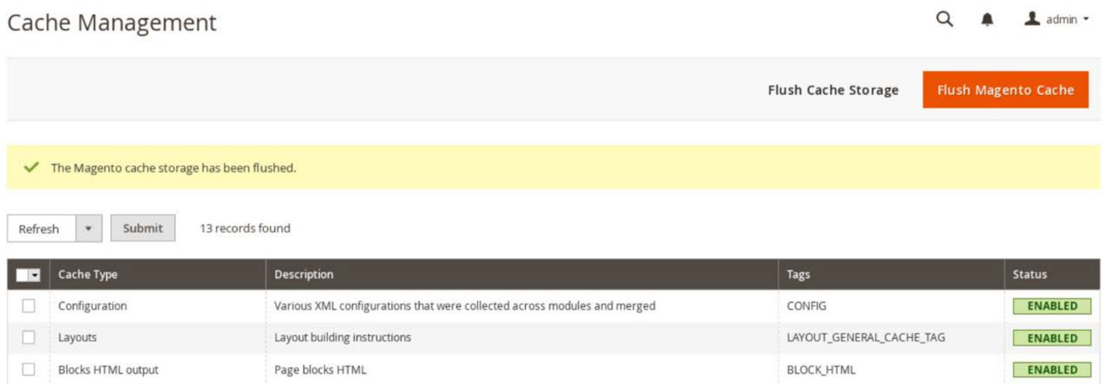
admin

Flush Cache Storage | Flush Magento Cache

Refresh | Submit | 13 records found

Cache Type	Description	Tags	Status
Configuration	Various XML configurations that were collected across modules and merged	CONFIG	INVALIDATED
Layouts	Layout building instructions	LAYOUT_GENERAL_CACHE_TAG	ENABLED
Blocks HTML output	Page blocks HTML	BLOCK_HTML	ENABLED

8. Upon completion of the flush, the page will reflect the changes.


Cache Management

admin

Flush Cache Storage | Flush Magento Cache

✓ The Magento cache storage has been flushed.

Refresh | Submit | 13 records found

Cache Type	Description	Tags	Status
Configuration	Various XML configurations that were collected across modules and merged	CONFIG	ENABLED
Layouts	Layout building instructions	LAYOUT_GENERAL_CACHE_TAG	ENABLED
Blocks HTML output	Page blocks HTML	BLOCK_HTML	ENABLED

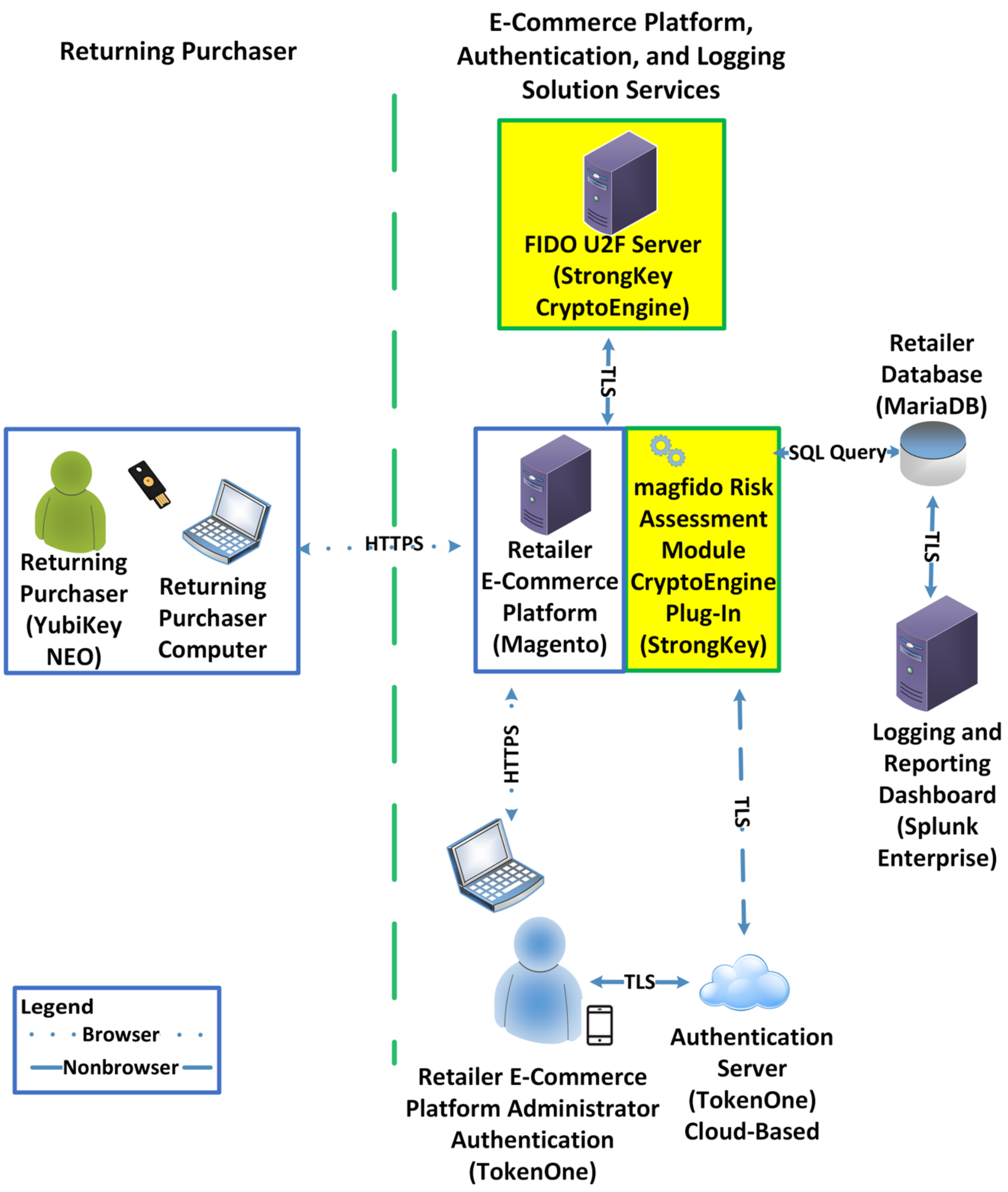
2.3 StrongKey magfido Module

This section of the guide provides installation and configuration guidance for the StrongKey magfido *FIDOU2FAuthenticator* module [6]. While the core feature of the magfido module is to enable U2F authentication, the magfido module also allows registration of FIDO U2F Security Keys. Additional information on magfido and how the registration feature works can be found in Appendix A.

2.3.1 StrongKey magfido Overview

The magfido module is used in the *cost threshold* example implementation build to examine the shopping cart's characteristics and to recommend whether MFA is required for the returning purchaser. The magfido module will modify the default behavior of Magento to register *FIDOU2FAuthenticators,* also known as FIDO Security Keys, and for FIDO authentication on purchases that exceed a total of $25. The StrongKey magfido components that are installed by using the instructions in this section are illustrated in Figure 2-3 and are highlighted in yellow within the green boxes.

Figure 2-3 StrongKey magfido Module Components


Returning Purchaser
E-Commerce Platform, Authentication, and Logging Solution Services
FIDO U2F Server (StrongKey CryptoEngine)
TLS
Retailer Database (MariaDB)
SQL Query
TLS
Logging and Reporting Dashboard (Splunk Enterprise)
Returning Purchaser (YubiKey NEO)
Returning Purchaser Computer
HTTPS
Retailer E-Commerce Platform (Magento)
magfido Risk Assessment Module CryptoEngine Plug-In (StrongKey)
HTTPS
TLS
TLS
Legend
Browser
Nonbrowser
Retailer E-Commerce Platform Administrator Authentication (TokenOne)
Authentication Server (TokenOne) Cloud-Based

2.3.2 StrongKey magfido Installation and Configuration

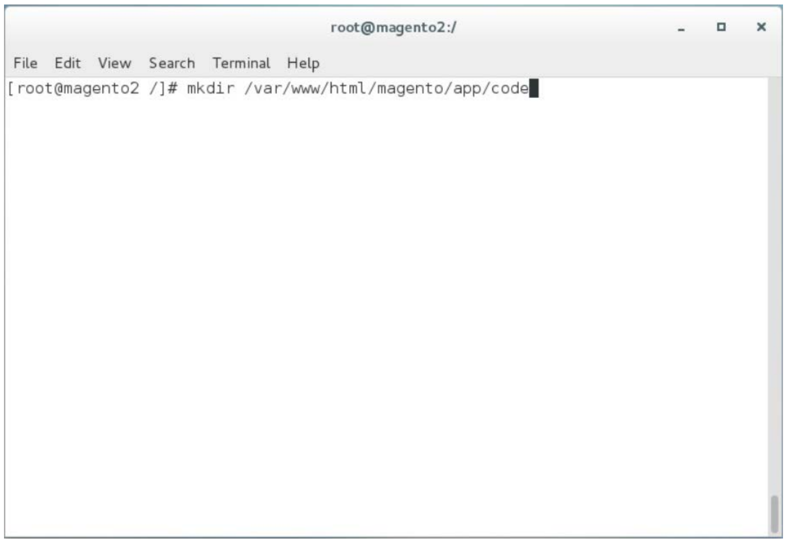
The installation procedure consists of the following steps.

- Download the software module to the Magento server where magfido will be installed.
- Execute commands as root/administrator.
- Perform post-installation configuration.

Navigate to the following site, and proceed to download the code: https://sourceforge.net/projects/magfido/.

1. Create a code directory inside Magento's app folder by entering the following command:

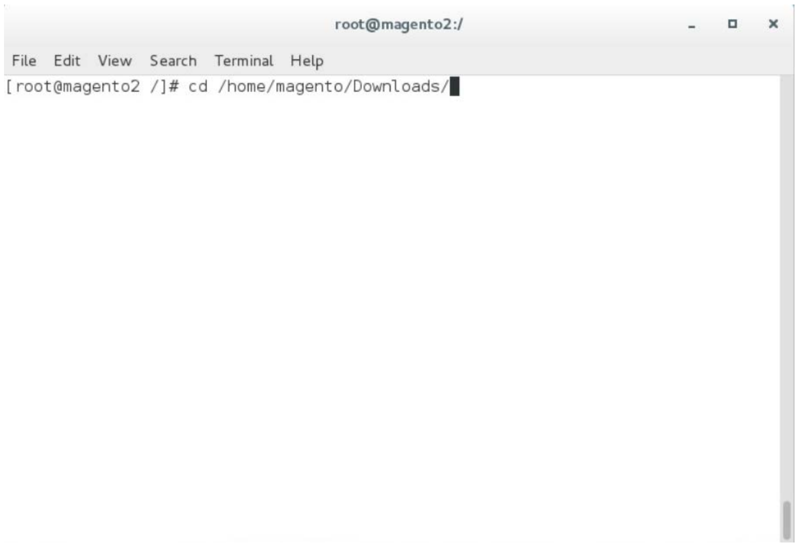
   ```
   mkdir /var/www/html/magento/app/code
   ```



```
[root@magento2 /]# mkdir /var/www/html/magento/app/code
```

2. Change your current directory to the Downloads directory by entering the following command:

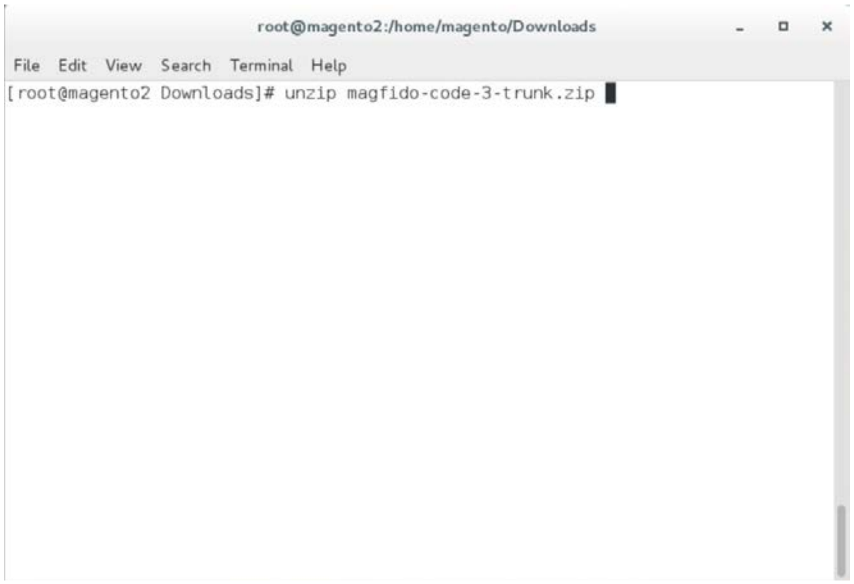
   ```
   cd /home/magento/Downloads/
   ```



```
[root@magento2 /]# cd /home/magento/Downloads/
```

3. Unzip the *magfido-code-3-trunk.zip* by entering the following command:

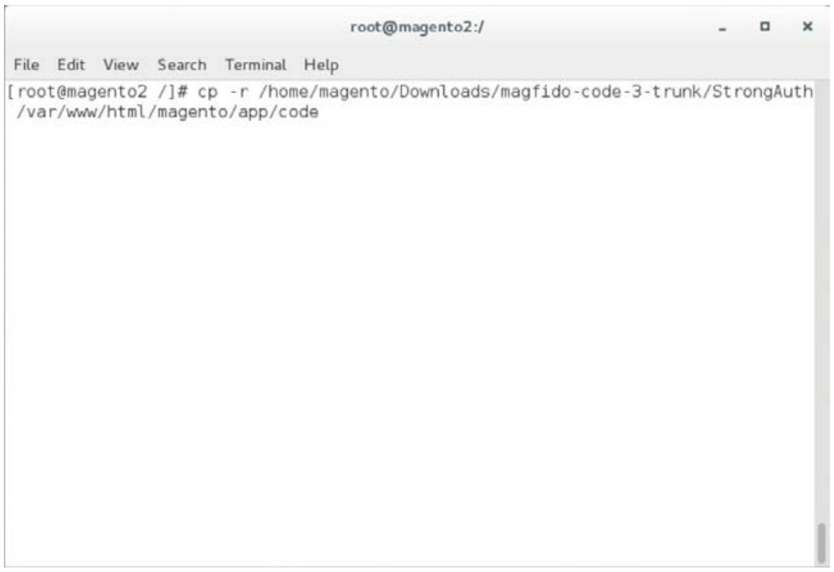
   ```
   unzip magfido-code-3-trunk.zip
   ```


root@magento2:/home/magento/Downloads
File Edit View Search Terminal Help

```
[root@magento2 Downloads]# unzip magfido-code-3-trunk.zip
```

4. Move the *StrongAuth_FIDOU2FAuthenticator* module to the code directory by entering the following command:

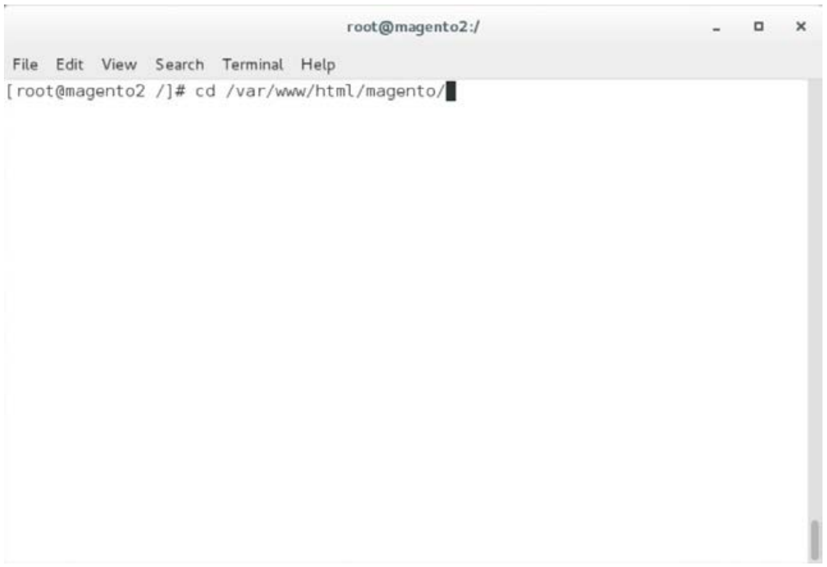
   ```
   cp -r home/magento/Downloads/magfido-code-3-trunk/StrongAuth
   /var/www/html/magento/app/code
   ```



```
[root@magento2 /]# cp -r /home/magento/Downloads/magfido-code-3-trunk/StrongAuth
 /var/www/html/magento/app/code
```

5. Change directories to the Magento directory by entering the following command:

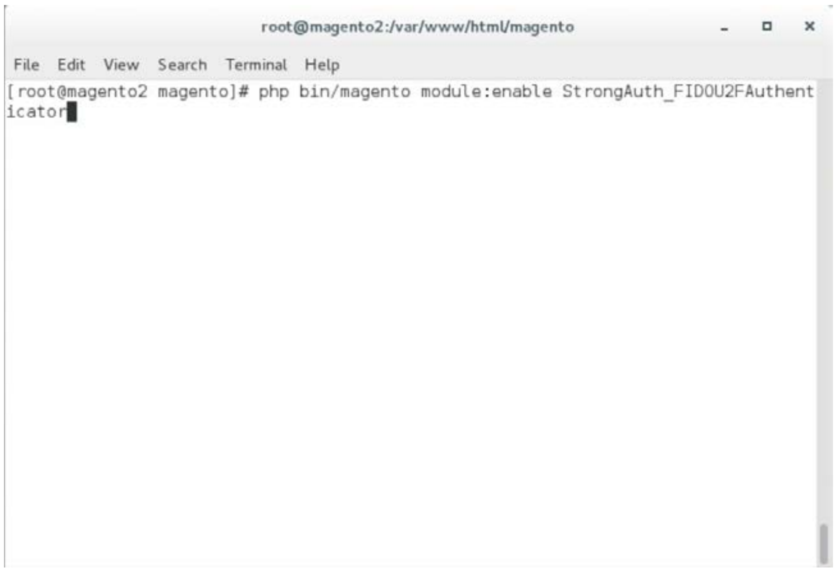
   ```
   cd /var/www/html/magento
   ```



```
[root@magento2 /]# cd /var/www/html/magento/
```

6. Enable the *StrongAuth_FIDOU2FAuthenticator* module by entering the following command:

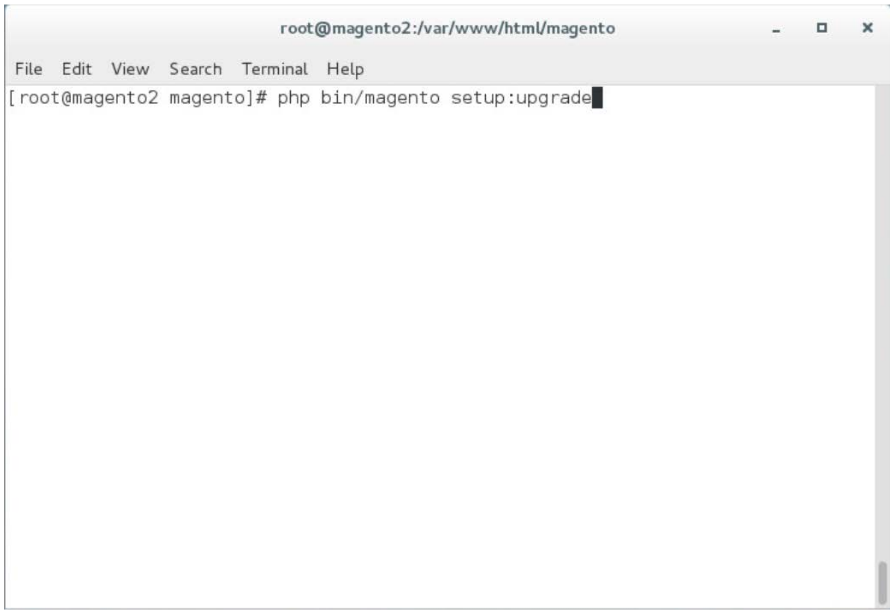
   ```
   php bin/magento module:enable StrongAuth_FIDOU2FAuthenticator
   ```


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# php bin/magento module:enable StrongAuth_FIDOU2FAuthent
icator
```

7. Register the *StrongAuth_FIDOU2FAuthenticator* module by entering the following command:

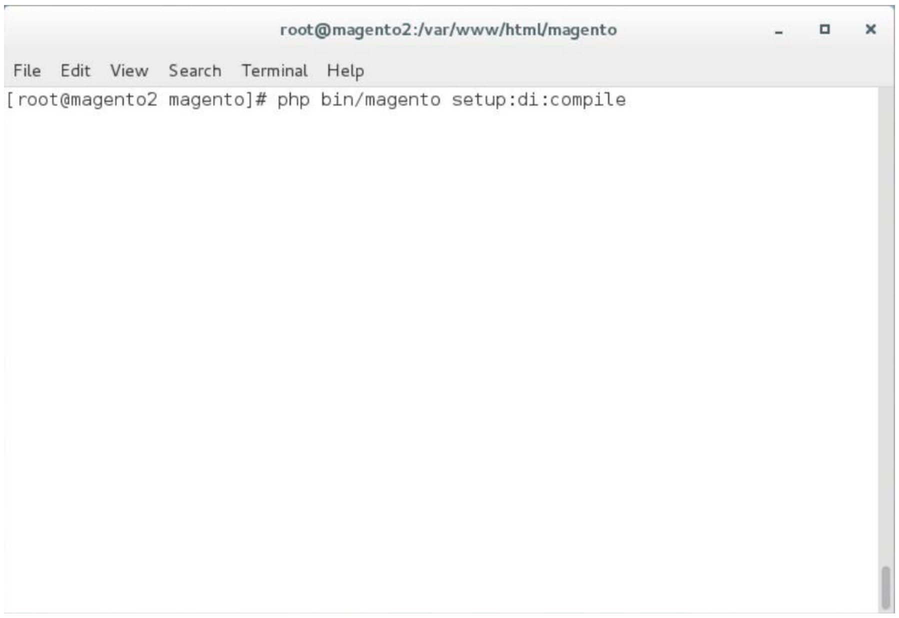
   `php bin/magento setup:upgrade`


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# php bin/magento setup:upgrade
```

8. Recompile dependencies by entering the following command:

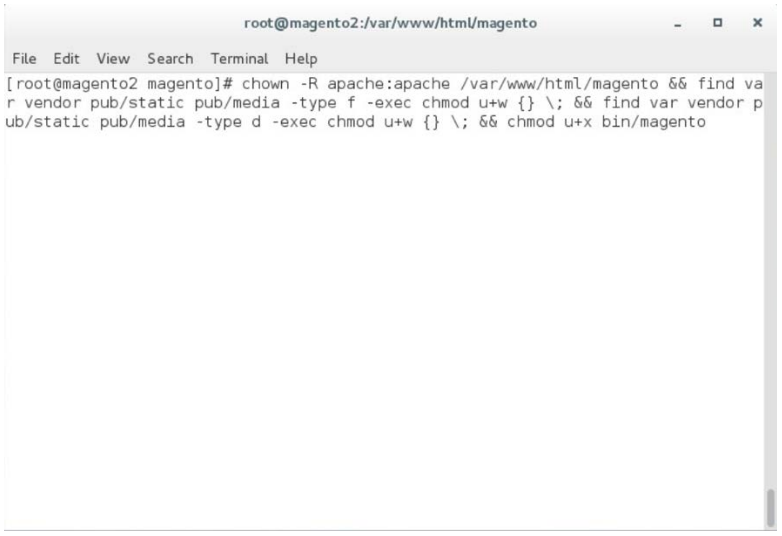
   `php bin/magento setup:di:compile`


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# php bin/magento setup:di:compile
```

9. Adjust the compiled file permissions by entering the following command:

   ```
   chown -R apache:apache /var/www/html/magento && find var vendor pub/static
   pub/media -type f -exec chmod u+w {} \; && find var vendor pub/static pub/media
   -type d -exec chmod u+w {} \; && chmod u+x bin/magento
   ```

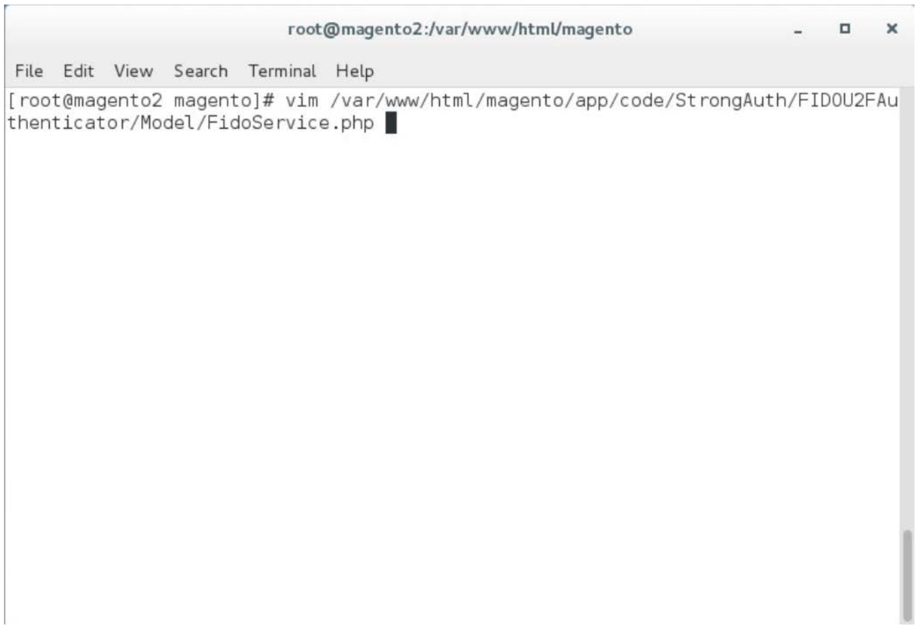

root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# chown -R apache:apache /var/www/html/magento && find va
r vendor pub/static pub/media -type f -exec chmod u+w {} \; && find var vendor p
ub/static pub/media -type d -exec chmod u+w {} \; && chmod u+x bin/magento
```

10. Configure the locally installed SKCE with the following steps:

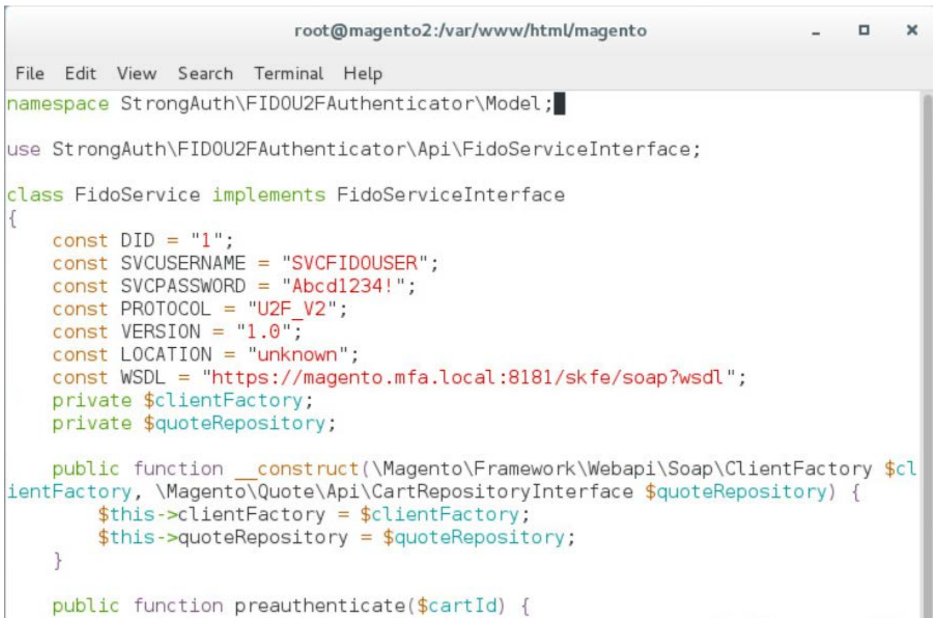
    a. Open *FidoService.php* by entering the following command:

    ```
    Vim
    /var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model/Fido
    Service.php
    ```


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# vim /var/www/html/magento/app/code/StrongAuth/FIDOU2FAu
thenticator/Model/FidoService.php
```

    b. Modify the file to include the following information:


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
namespace StrongAuth\FIDOU2FAuthenticator\Model;

use StrongAuth\FIDOU2FAuthenticator\Api\FidoServiceInterface;

class FidoService implements FidoServiceInterface
{
    const DID = "1";
    const SVCUSERNAME = "SVCFIDOUSER";
    const SVCPASSWORD = "Abcd1234!";
    const PROTOCOL = "U2F_V2";
    const VERSION = "1.0";
    const LOCATION = "unknown";
    const WSDL = "https://magento.mfa.local:8181/skfe/soap?wsdl";
    private $clientFactory;
    private $quoteRepository;

    public function __construct(\Magento\Framework\Webapi\Soap\ClientFactory $cl
ientFactory, \Magento\Quote\Api\CartRepositoryInterface $quoteRepository) {
        $this->clientFactory = $clientFactory;
        $this->quoteRepository = $quoteRepository;
    }

    public function preauthenticate($cartId) {
```

i. The **DID** parameter is the Domain ID of SKCE.

ii. The **SVCUSERNAME** parameter is the SKCE user responsible for authorizing requests to the FIDO server.

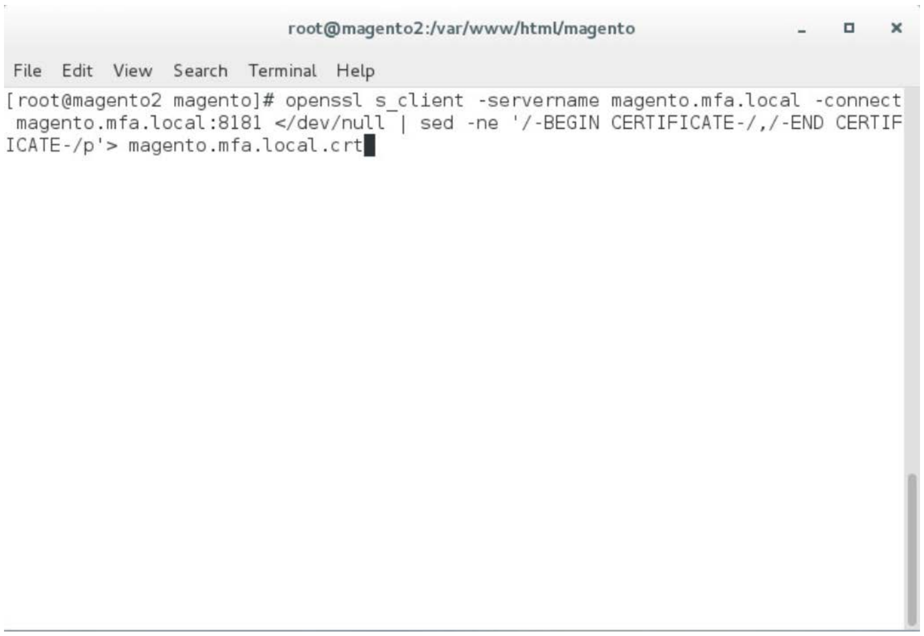
iii. The **SVCPASSWORD** parameter is the password of the SKCE user.

iv. The **PROTOCOL, VERSION,** and **LOCATION** are parameters used for reference for the FIDO server. They should be left as is.

v. The **WSDL** (Web Services Description Language) parameter specifies the web service end point with which the Magento server will communicate to send web-service requests to the FIDO server. The default SKCE install will have the WSDL as “https://<fully-qualified-domainname>:8181/skfe/soap?wsdl.”

c. Retrieve a copy of the FIDO server’s TLS digital certificate by entering the following command (Note: This is a single command that must be executed on a single line.):

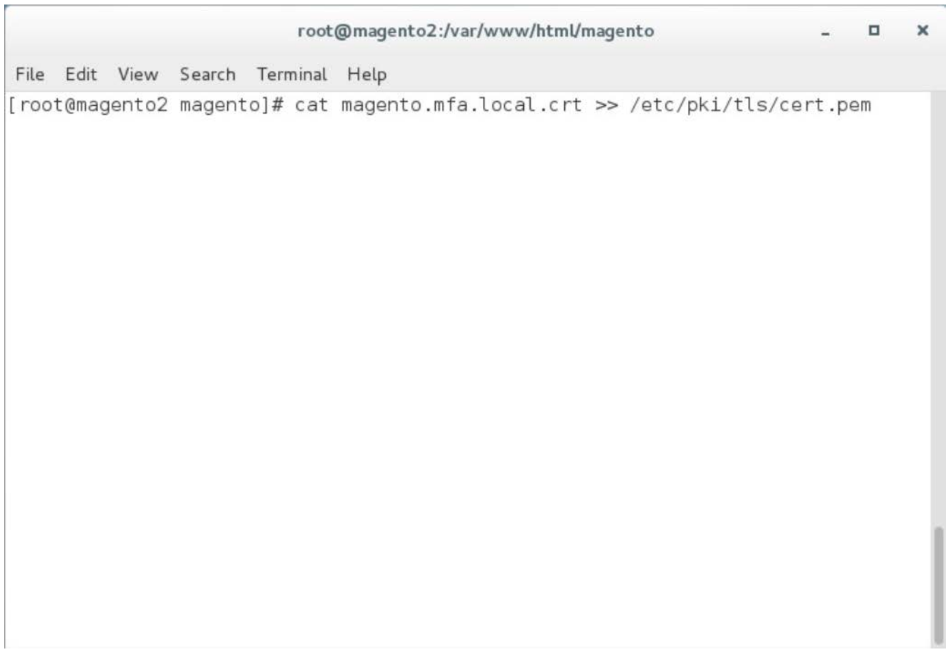
```
openssl s_client -servername <fully-qualified-domain-name> -connect
<fully-qualified-domain-name>:8181 </dev/null | sed -ne '/BEGIN
CERTIFICATE-/,/-END CERTIFICATE-/p' > <FQDN>.crt
```


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# openssl s_client -servername magento.mfa.local -connect
 magento.mfa.local:8181 </dev/null | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIF
ICATE-/p'> magento.mfa.local.crt
```

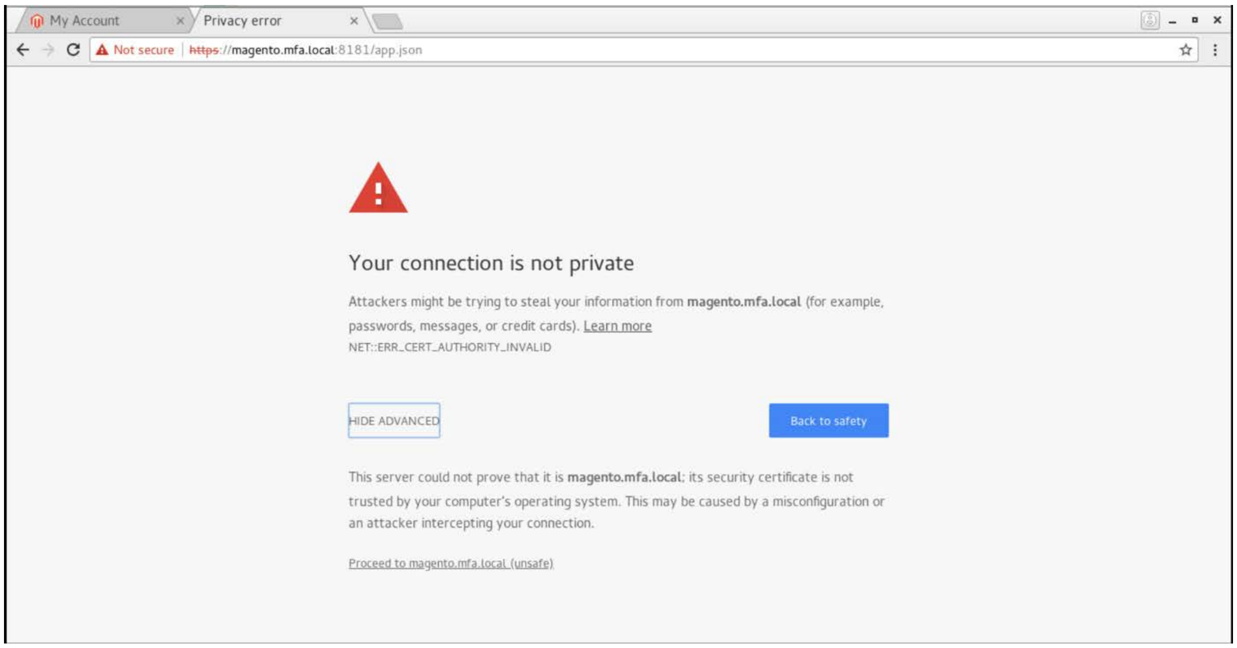
d. Add the certificate to the list of trusted certificates by entering the following command:

```
cat <fully-qualified-domain-name>.crt >> /etc/pki/tls/cert.pem
```


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# cat magento.mfa.local.crt >> /etc/pki/tls/cert.pem
```

e. Open the Chrome browser and navigate to *https://magento.mfa.local:8181/app.json*.


My Account
Privacy error
Not secure | ~~https~~://magento.mfa.local:8181/app.json

Your connection is not private

Attackers might be trying to steal your information from **magento.mfa.local** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_AUTHORITY_INVALID

HIDE ADVANCED

Back to safety

This server could not prove that it is **magento.mfa.local**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

Proceed to magento.mfa.local (unsafe)

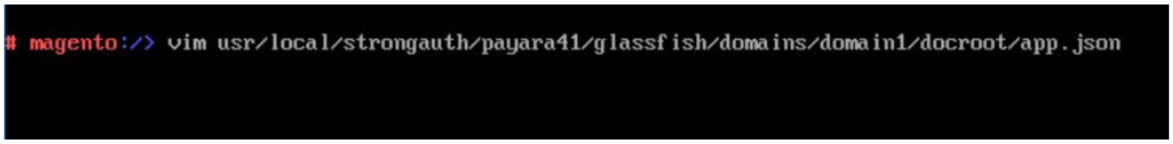
i. A warning will appear, stating that "Your connection is not private."

ii. Click **HIDE ADVANCED.**

iii. Click **Proceed to <fully-qualified-domain-name> (unsafe).**

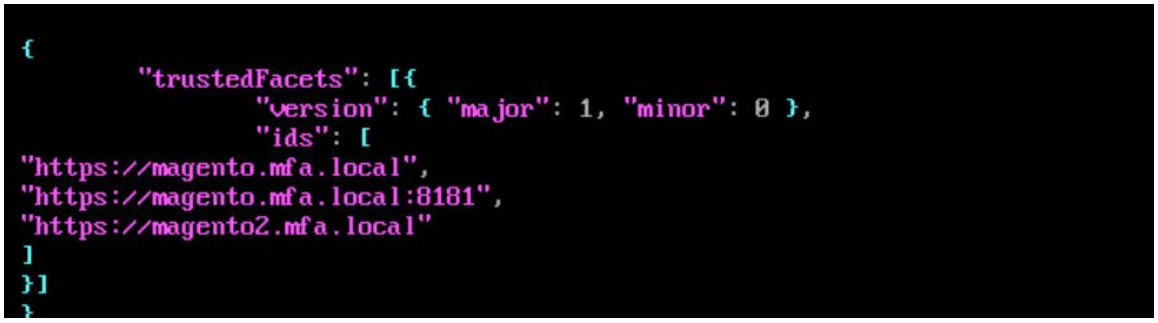
f. On your SKCE machine, edit the *app.json* file by entering the following command:

```
vim
usr/local/strongauth/payara41/glassfish/domains/domain1/docroot/app.json
```



```
# magento:/> vim usr/local/strongauth/payara41/glassfish/domains/domain1/docroot/app.json
```

g. Add the FQDN of the machine hosting the Magento application in the ids array, and save the file.

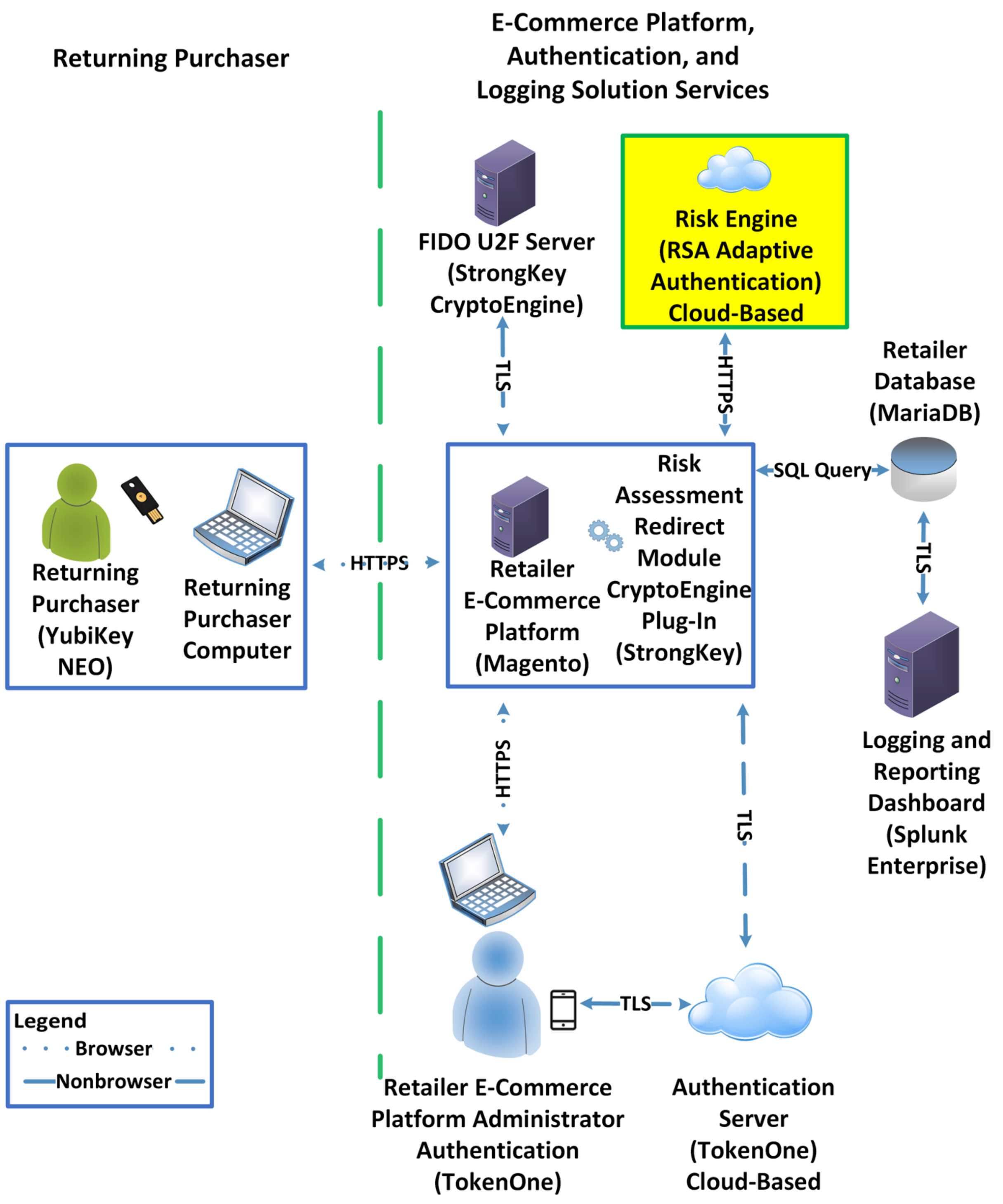


```
{
        "trustedFacets": [{
                "version": { "major": 1, "minor": 0 },
                "ids": [
"https://magento.mfa.local",
"https://magento.mfa.local:8181",
"https://magento2.mfa.local"
]
}]
}
```

2.4 RSA Adaptive Authentication

This section of the guide provides installation and configuration guidance for the RSA Adaptive Authentication risk engine. The RSA Adaptive Authentication product performs a risk analysis and then prompts the returning user to provide an MFA authenticator when required for the *risk engine* example implementation build. The purpose of the RSA Adaptive Authentication is to minimize fraud with a low-friction consumer experience. This example implementation uses the RSA Adaptive Authentication cloud offering. The components that integrate Magento with RSA Adaptive Authentication are installed by using the instructions in this section. The components are illustrated in Figure 2-4 and are highlighted in yellow within the green box.

Figure 2-4 RSA Adaptive Authentication Components


Returning Purchaser
E-Commerce Platform, Authentication, and Logging Solution Services
FIDO U2F Server (StrongKey CryptoEngine)
Risk Engine (RSA Adaptive Authentication) Cloud-Based
Retailer Database (MariaDB)
TLS
HTTPS
SQL Query
Returning Purchaser (YubiKey NEO)
Returning Purchaser Computer
HTTPS
Retailer E-Commerce Platform (Magento)
Risk Assessment Redirect Module CryptoEngine Plug-In (StrongKey)
TLS
Logging and Reporting Dashboard (Splunk Enterprise)
HTTPS
TLS
TLS
Retailer E-Commerce Platform Administrator Authentication (TokenOne)
Authentication Server (TokenOne) Cloud-Based
Legend
Browser
Nonbrowser

2.4.1 RSA Overview

RSA [7] offers an Adaptive Authentication [8] capability, which is part of the *risk engine* example implementation.

The installation procedure consists of the following steps:

- Preinstallation:
  - Download the RSA Project Library.
  - Configure Magento to accept additional extension attributes.
- Installation and configuration:
  - Integrate RSA files into Magento.
  - Create policy in RSA Back Office.

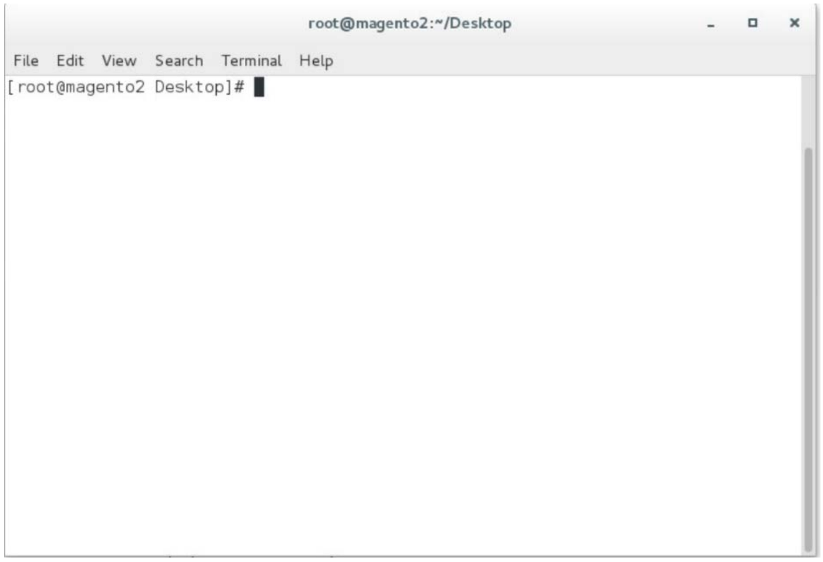
2.4.2 RSA Preinstallation Steps

Before beginning installation, perform the following steps.

- Contact your RSA representative regarding access to RSA project library files *(RSA.zip)* and *RSA.php* files. Download these files to the */home/magento/Downloads* directory.
- Configure Magento to accept additional extension attributes as outlined below.

This section will discuss how to add extension attributes to Magento to pass necessary information to RSA Adaptive Authentication.

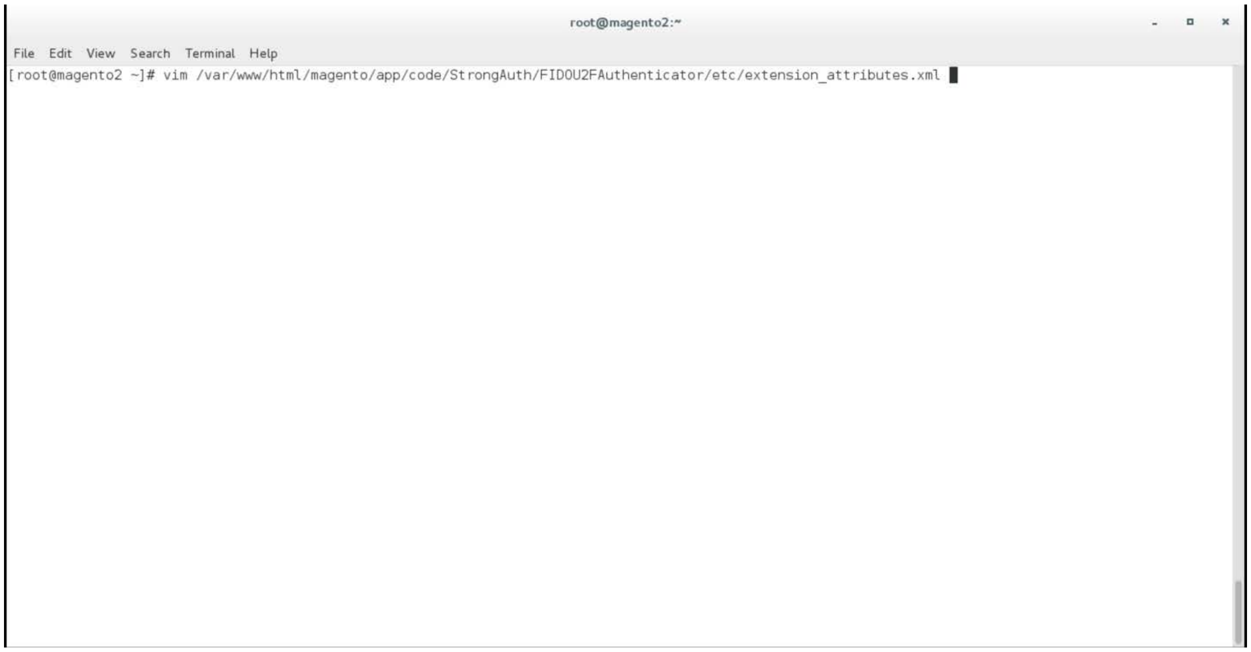
1. Open a terminal window.



```
[root@magento2 Desktop]#
```

2. To edit the file containing Magento's extension attributes, issue the following commands:

   a. `vim /var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/etc/extension_attributes.xml`

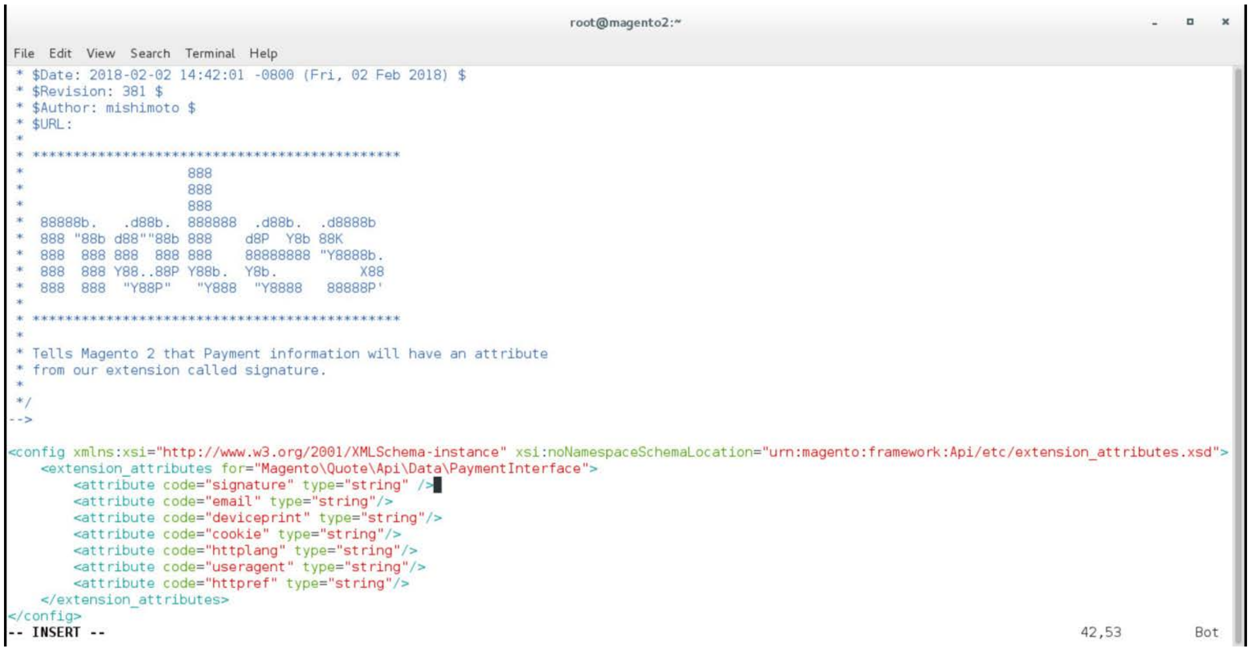


```
[root@magento2 ~]# vim /var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/etc/extension_attributes.xml
```

   b. Press **i** to enter insertion mode.

3. Following Line 53, which contains `<attribute code="signature" type="string" />`, insert the following lines (shown in the picture below):

```
<attribute code="email" type="string"/>

<attribute code="deviceprint" type="string"/>

<attribute code="cookie" type="string"/>

<attribute code="httplang" type="string"/>

<attribute code="useragent" type="string"/>

<attribute code="httpref" type="string"/>
```



```
 * $Date: 2018-02-02 14:42:01 -0800 (Fri, 02 Feb 2018) $
 * $Revision: 381 $
 * $Author: mishimoto $
 * $URL:
 *
 * *********************************************
 *                   888
 *                   888
 *                   888
 *  88888b.   .d88b.  888888  .d88b.   .d8888b
 *  888 "88b d88""88b 888    d8P  Y8b 88K
 *  888  888 888  888 888    88888888 "Y8888b.
 *  888  888 Y88..88P Y88b.  Y8b.          X88
 *  888  888  "Y88P"   "Y888  "Y8888   88888P'
 *
 * *********************************************
 *
 * Tells Magento 2 that Payment information will have an attribute
 * from our extension called signature.
 *
 */
-->

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:Api/etc/extension_attributes.xsd">
    <extension_attributes for="Magento\Quote\Api\Data\PaymentInterface">
        <attribute code="signature" type="string" />
        <attribute code="email" type="string"/>
        <attribute code="deviceprint" type="string"/>
        <attribute code="cookie" type="string"/>
        <attribute code="httplang" type="string"/>
        <attribute code="useragent" type="string"/>
        <attribute code="httpref" type="string"/>
    </extension_attributes>
</config>
-- INSERT --                                                                                    42,53         Bot
```

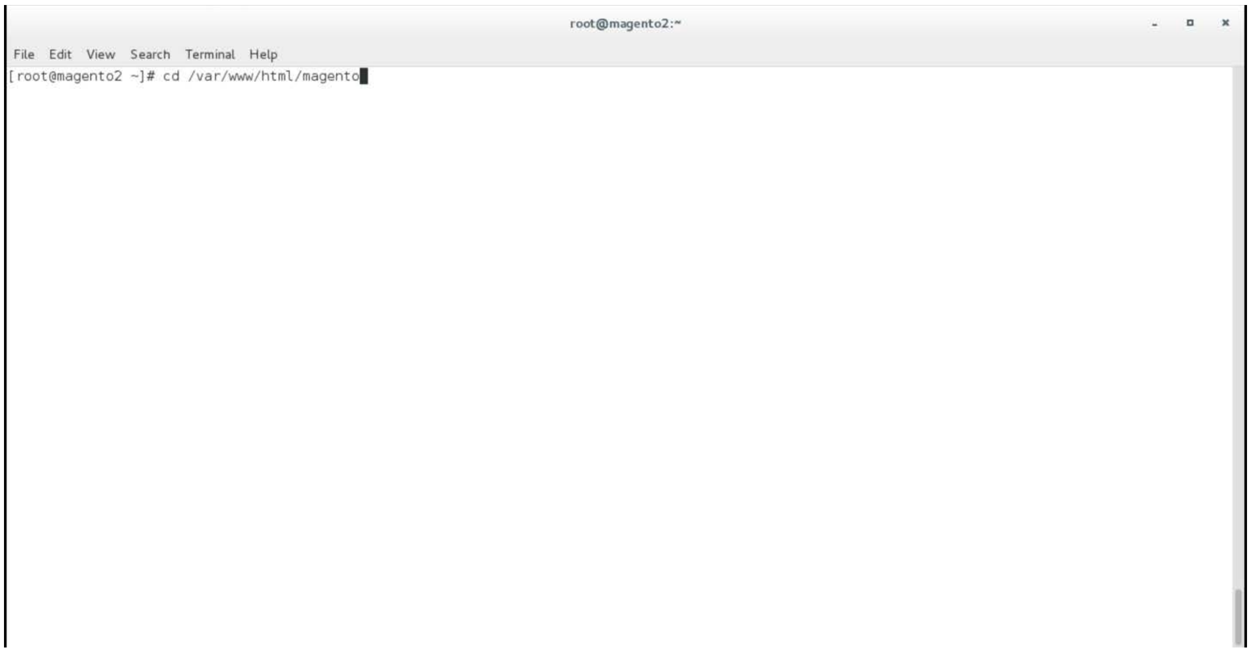
4. Press the Esc key to exit insert mode.

5. Save changes, and exit by entering the following command: `:wq`.

6. Return to the terminal window.

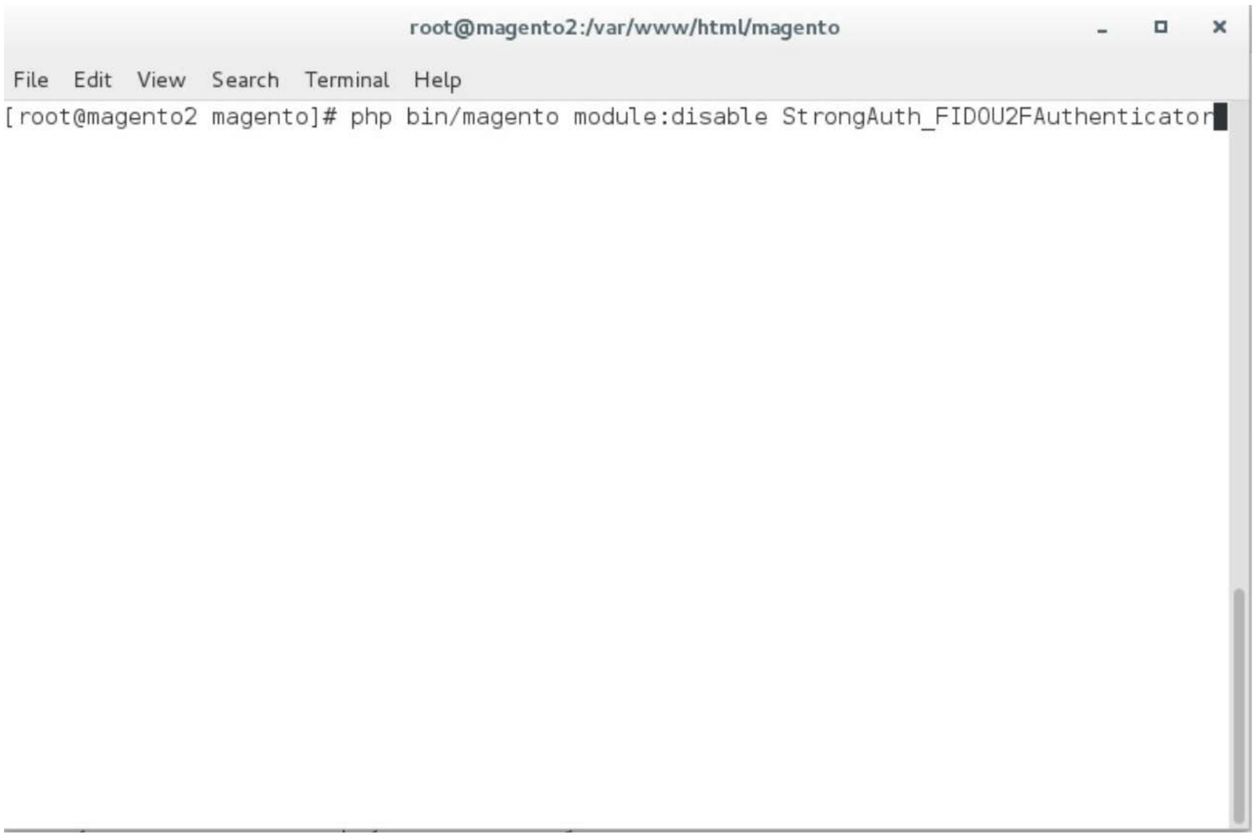
7. Change to the Magento folder by entering the following command:

   ```
   cd /var/www/html/magento
   ```



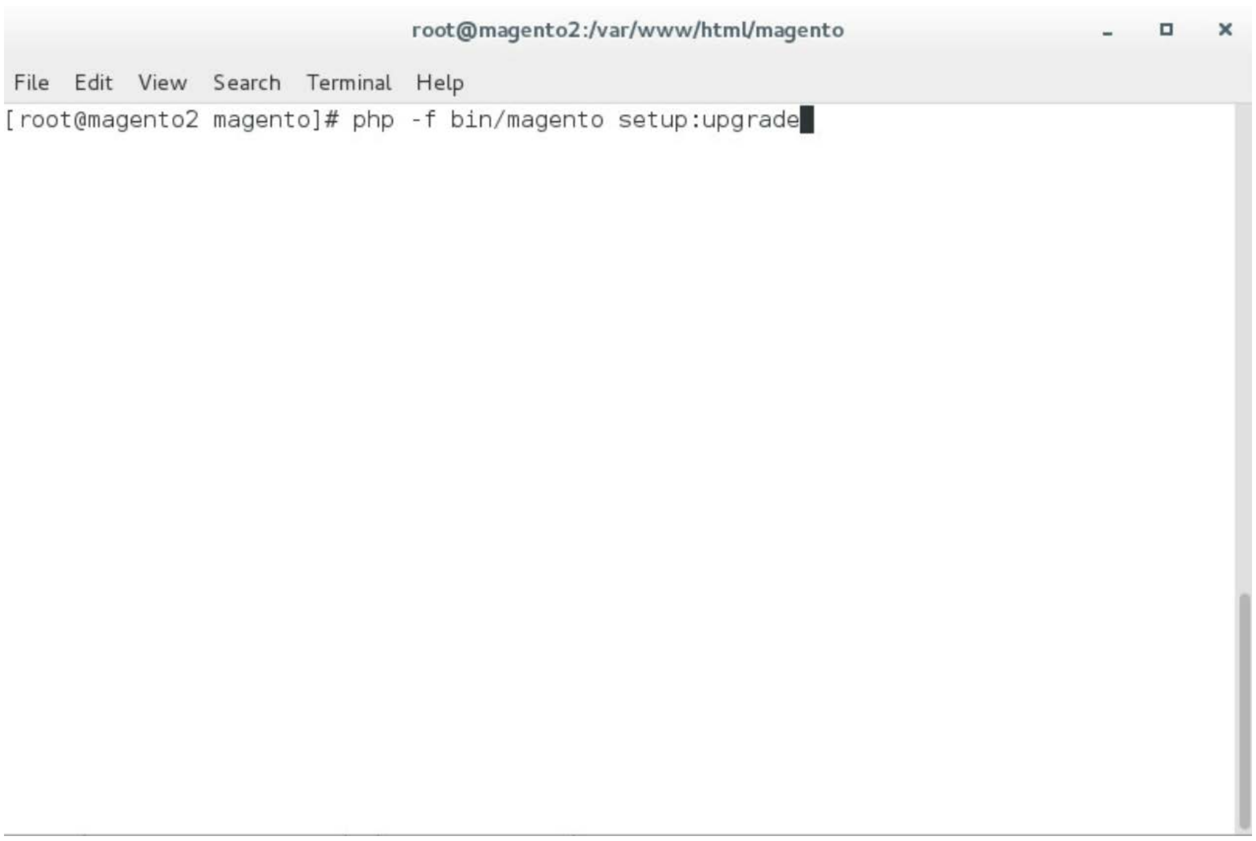
```
[root@magento2 ~]# cd /var/www/html/magento
```

8. To recompile Magento to reflect the changes made to the extension attributes file, issue the following commands:
    a. `php bin/magento module:disable StrongAuth_FIDOU2FAuthenticator`


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

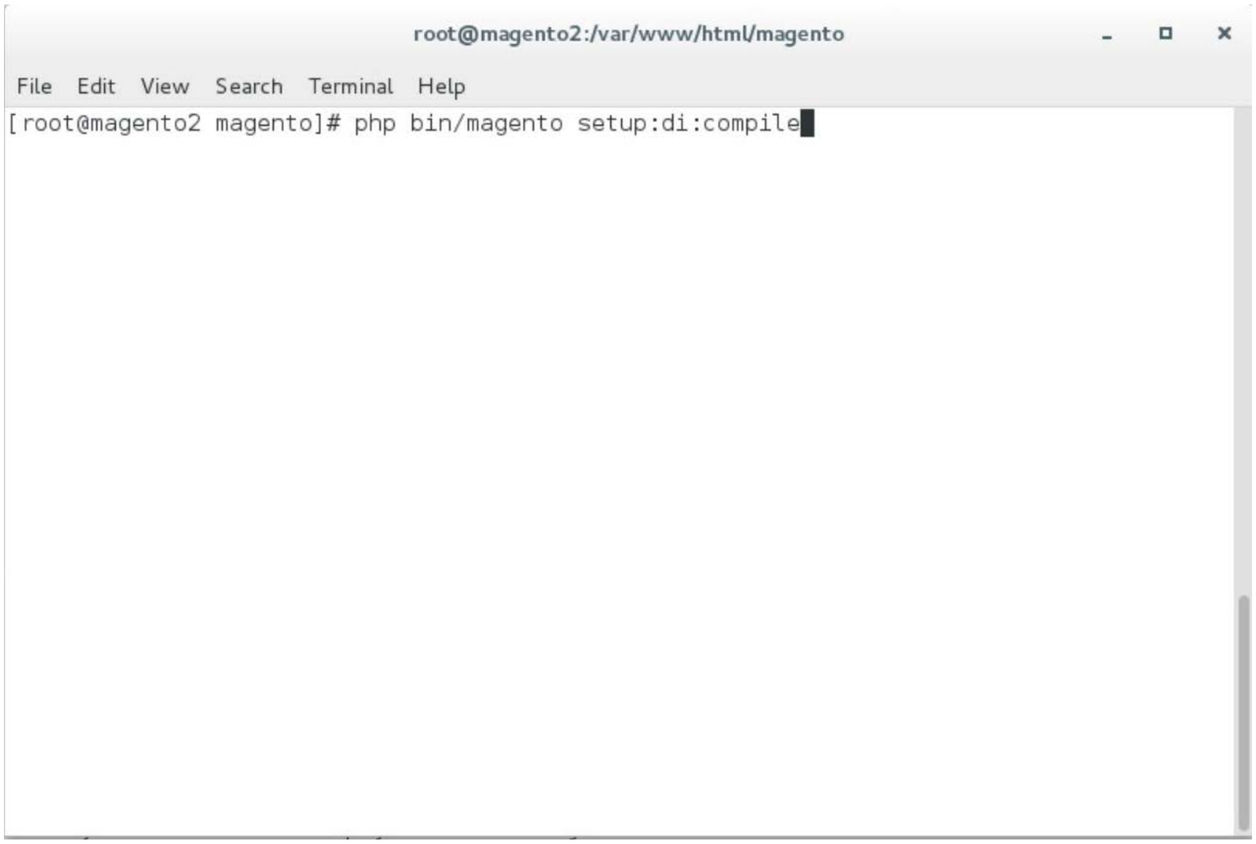
```
[root@magento2 magento]# php bin/magento module:disable StrongAuth_FIDOU2FAuthenticator
```

b. `php -f bin/magento setup:upgrade`


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

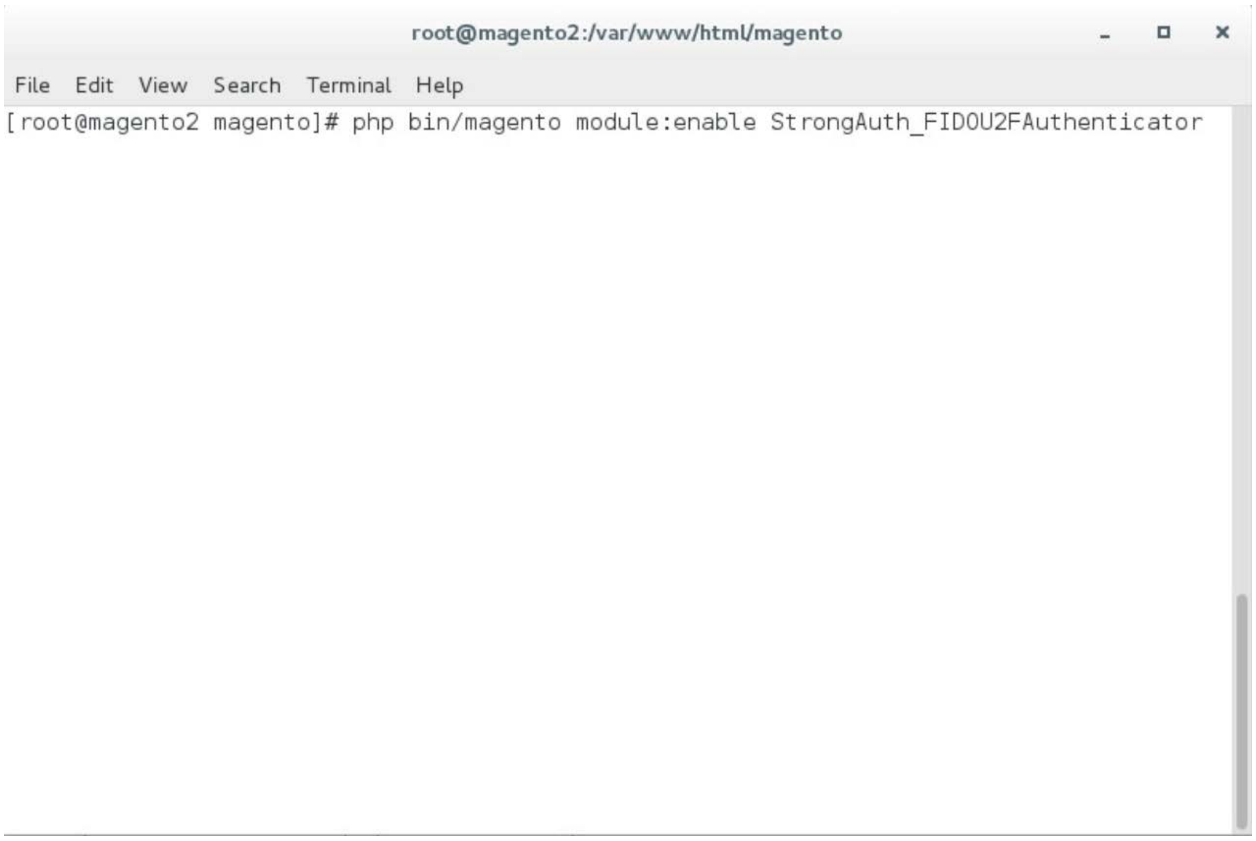
```
[root@magento2 magento]# php -f bin/magento setup:upgrade
```

c. `php bin/magento setup:di:compile`


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

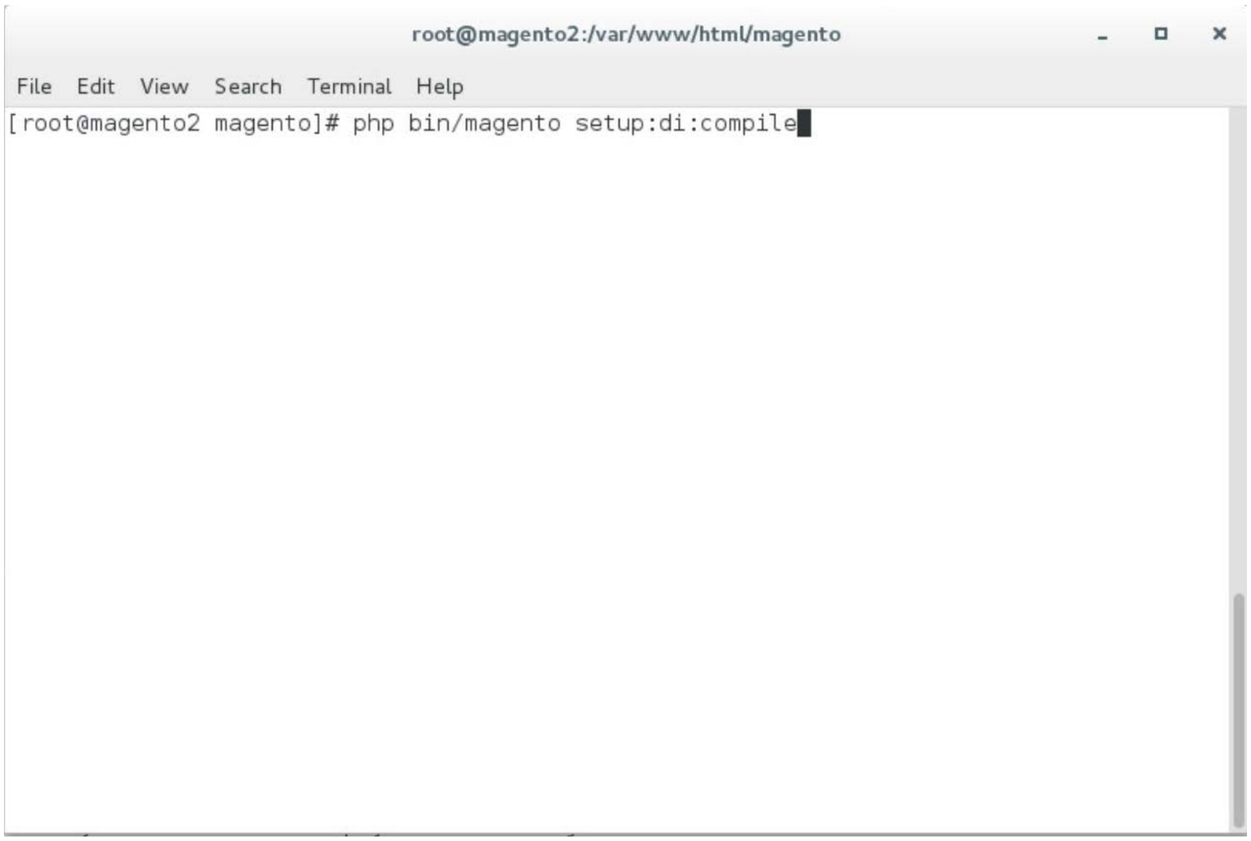
```
[root@magento2 magento]# php bin/magento setup:di:compile
```

d. `php bin/magento module:enable StrongAuth_FIDOU2FAuthenticator`


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# php bin/magento module:enable StrongAuth_FIDOU2FAuthenticator
```

e. `php bin/magento setup:di:compile`

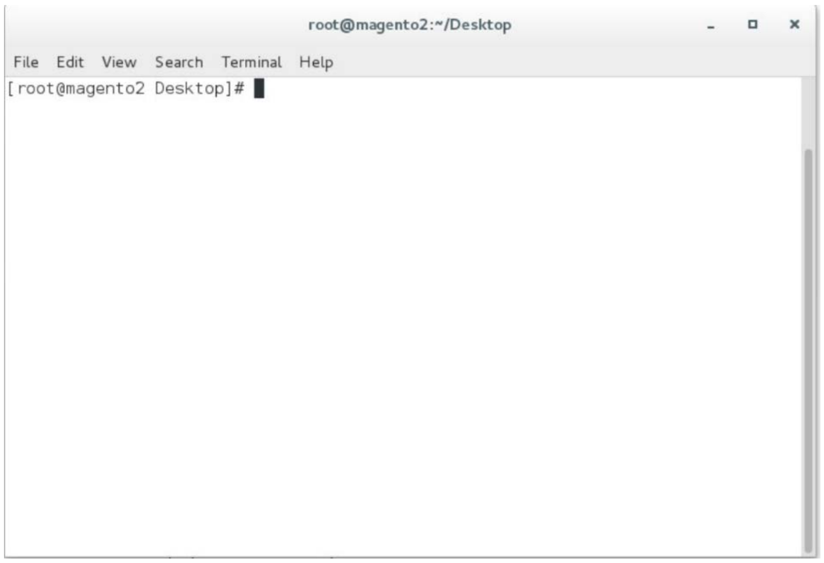

root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# php bin/magento setup:di:compile
```

2.4.3 Adaptive Authentication Installation and Configuration

This section provides a step-by-step installation guide for integrating RSA Adaptive Authentication. Before you begin, make sure that you have received your RSA project libraries from your RSA representative.

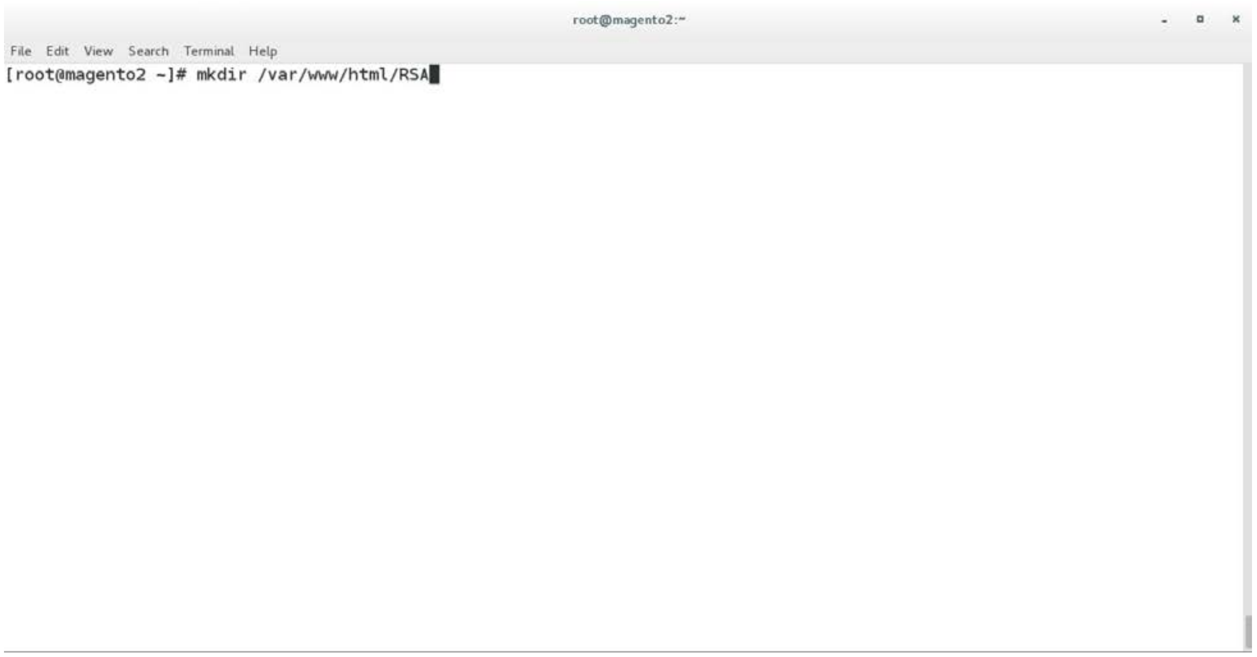
1. Open a terminal window.



```
[root@magento2 Desktop]#
```

2. Create a new directory by entering the following command:

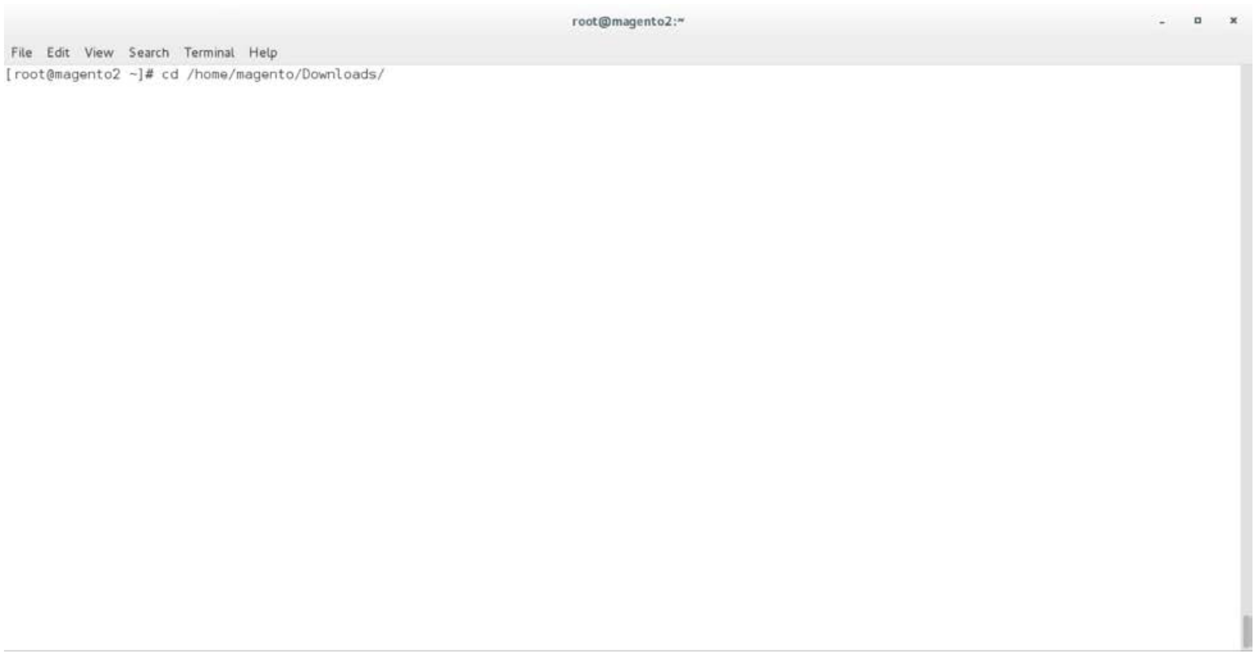
   ```
   Mkdir /var/www/html/RSA
   ```



```
[root@magento2 ~]# mkdir /var/www/html/RSA
```

3. Obtain the RSA zip file from your RSA representative.
4. Change to the Downloads directory by entering the following command:

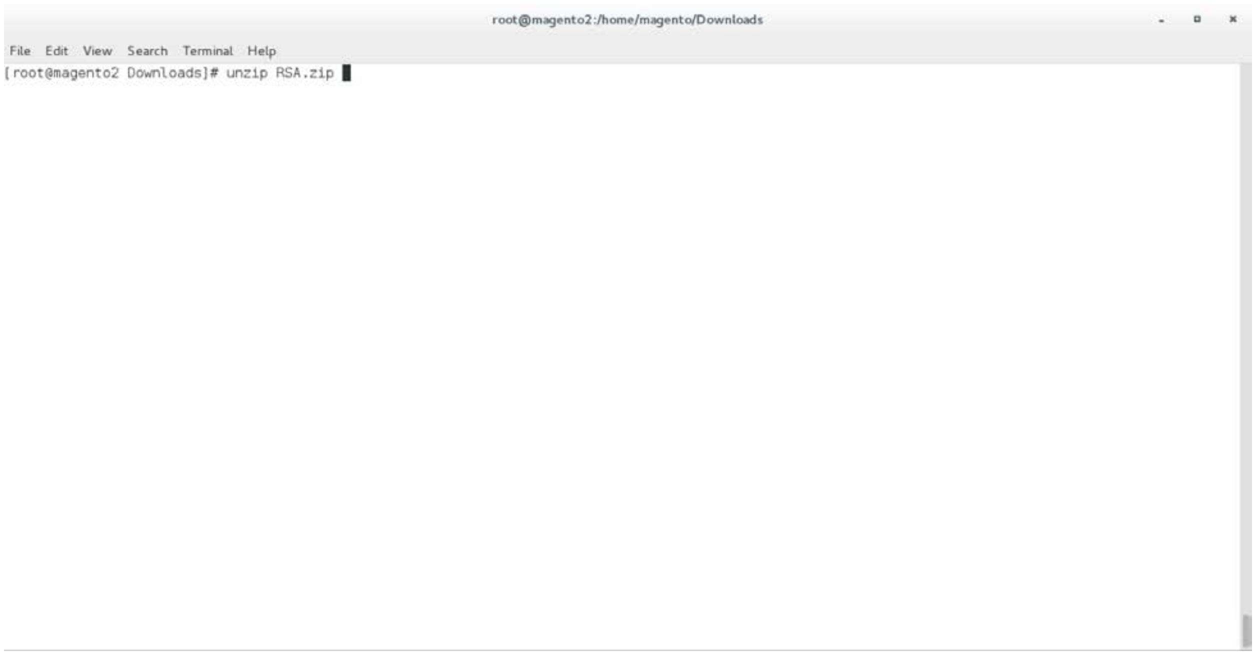
   ```
   cd /home/magento/Downloads
   ```



```
[root@magento2 ~]# cd /home/magento/Downloads/
```

5. Unzip the RSA directory by entering the following command:

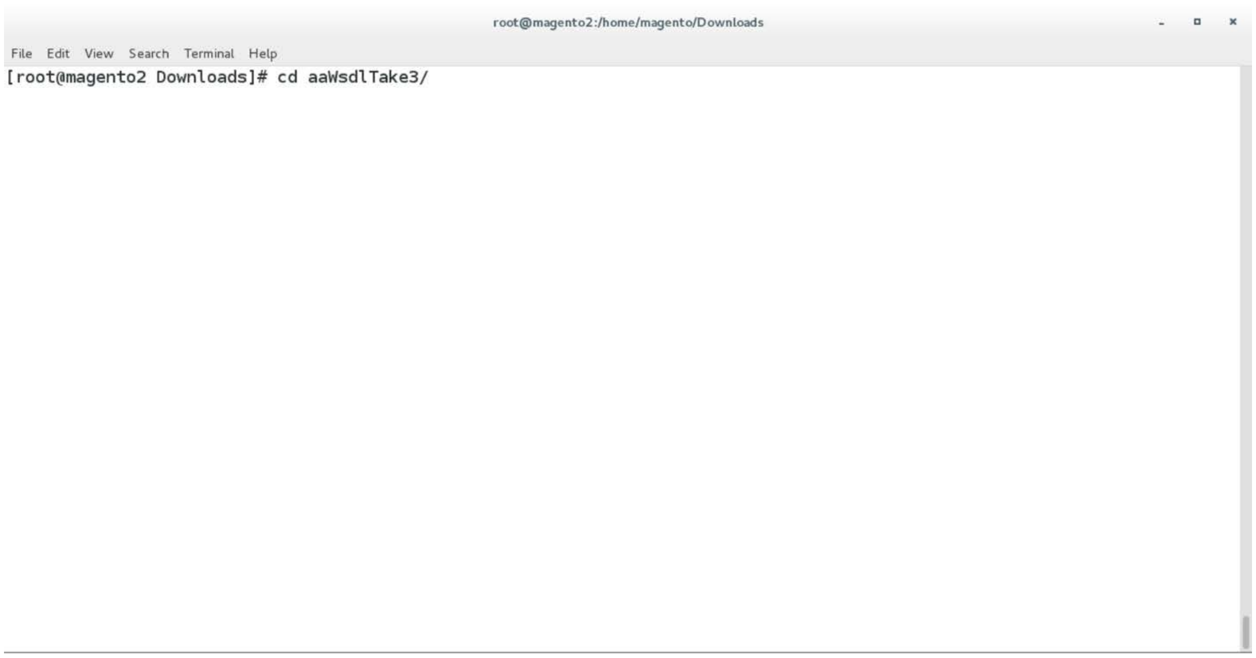
   `unzip RSA.zip`


root@magento2:/home/magento/Downloads
File Edit View Search Terminal Help

```
[root@magento2 Downloads]# unzip RSA.zip
```

6. Change to the newly unzipped directory by entering the following command:

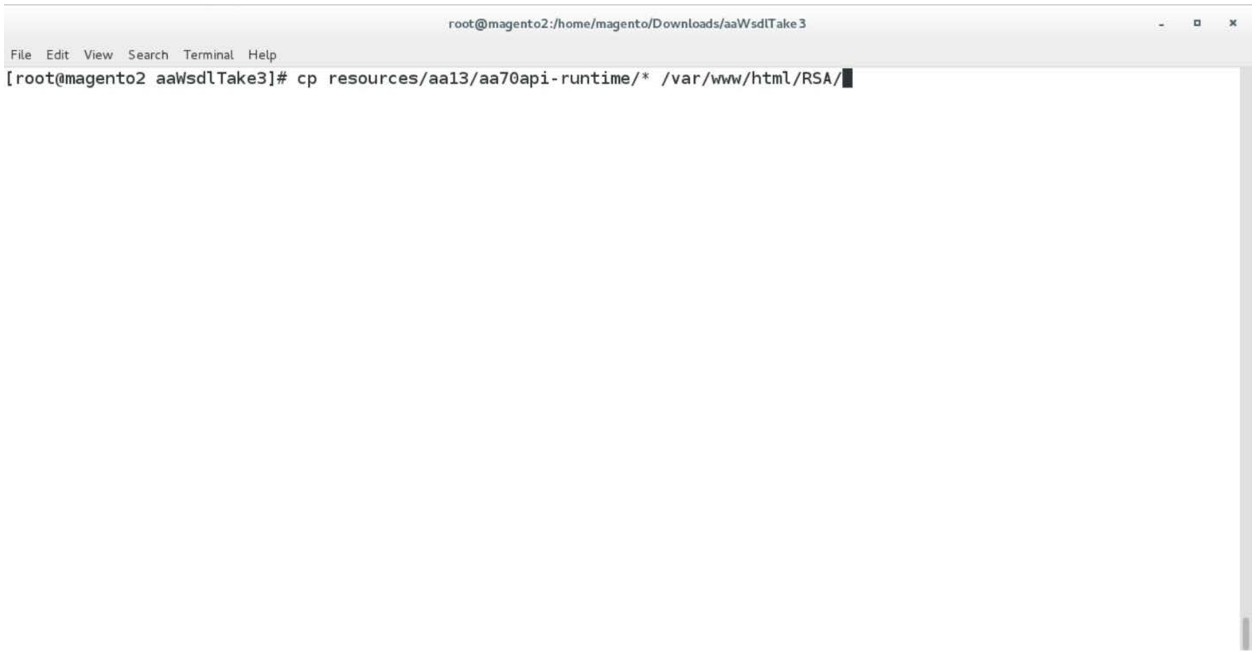
   ```
   cd aaWsdlTake3/
   ```


root@magento2:/home/magento/Downloads
File Edit View Search Terminal Help

```
[root@magento2 Downloads]# cd aaWsdlTake3/
```

7. Copy the contents of the API runtime directory to the RSA directory, which was created in Step 2, by entering the following command:

   ```
   cp resources/aa13/aa70api-runtime/* /var/www/html/RSA/
   ```

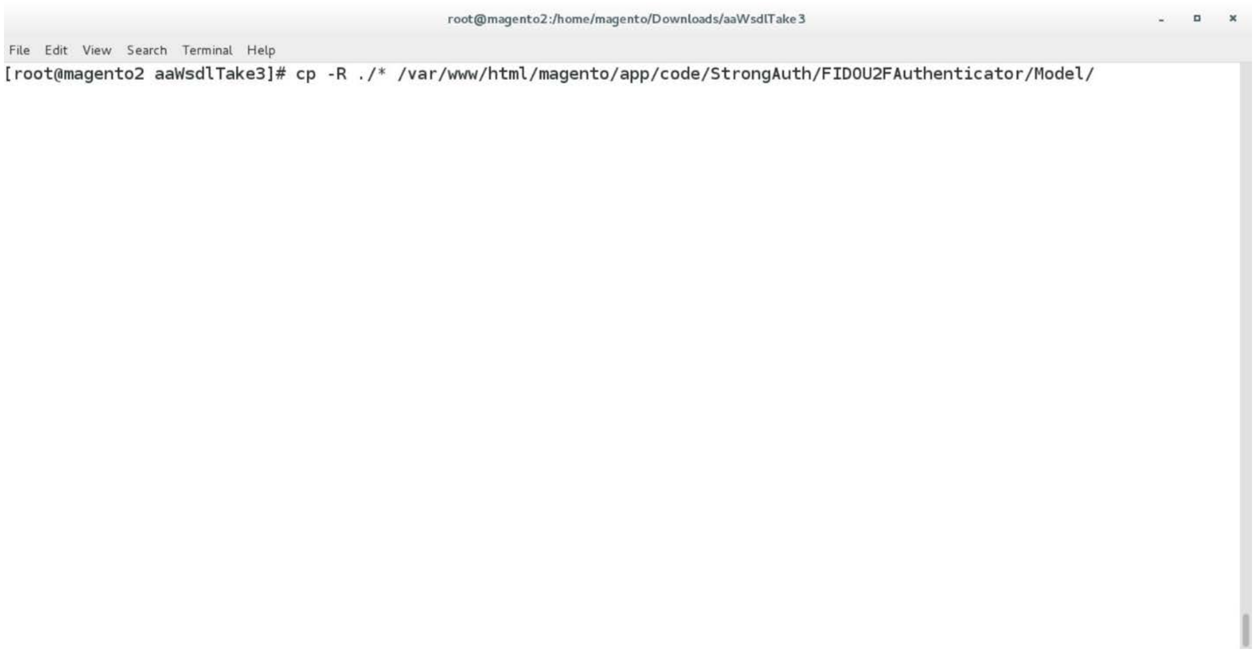


root@magento2:/home/magento/Downloads/aaWsdlTake3

```
[root@magento2 aaWsdlTake3]# cp resources/aa13/aa70api-runtime/* /var/www/html/RSA/
```

8. Copy the contents of the aaWsdlTake3 directory to the StrongAuth model directory by entering the following command:

   `cp -R ./* /var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model/`

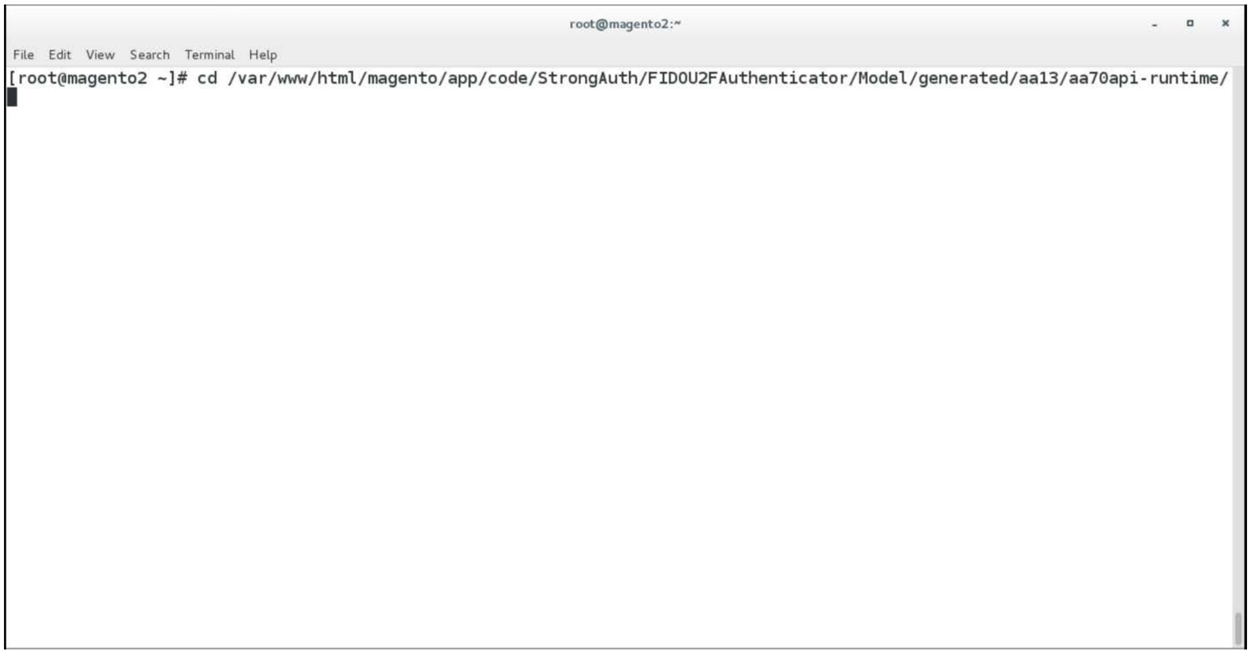


root@magento2:/home/magento/Downloads/aaWsdlTake3

```
[root@magento2 aaWsdlTake3]# cp -R ./* /var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model/
```

9. Change to the generated RSA API runtime folder by entering the following command:

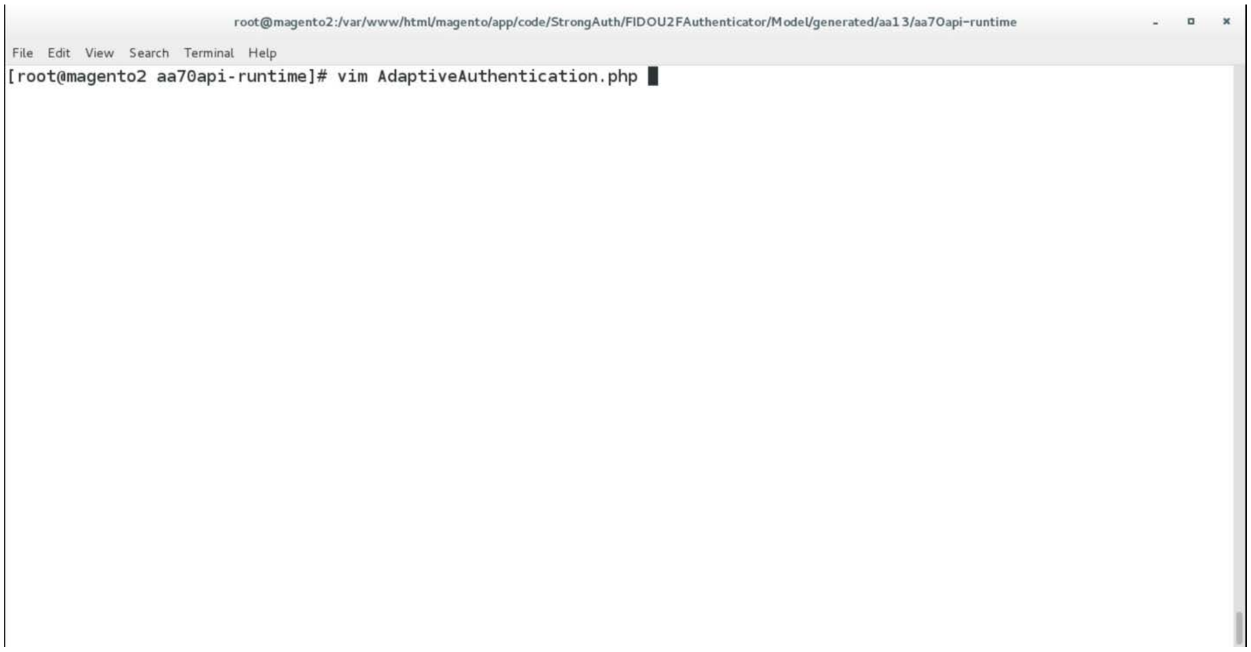
```
cd
/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model/generated/
aa13/aa70api-runtime/
```



```
[root@magento2 ~]# cd /var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model/generated/aa13/aa70api-runtime/
```

10. Edit the Adaptive Authentication file by entering the following command:

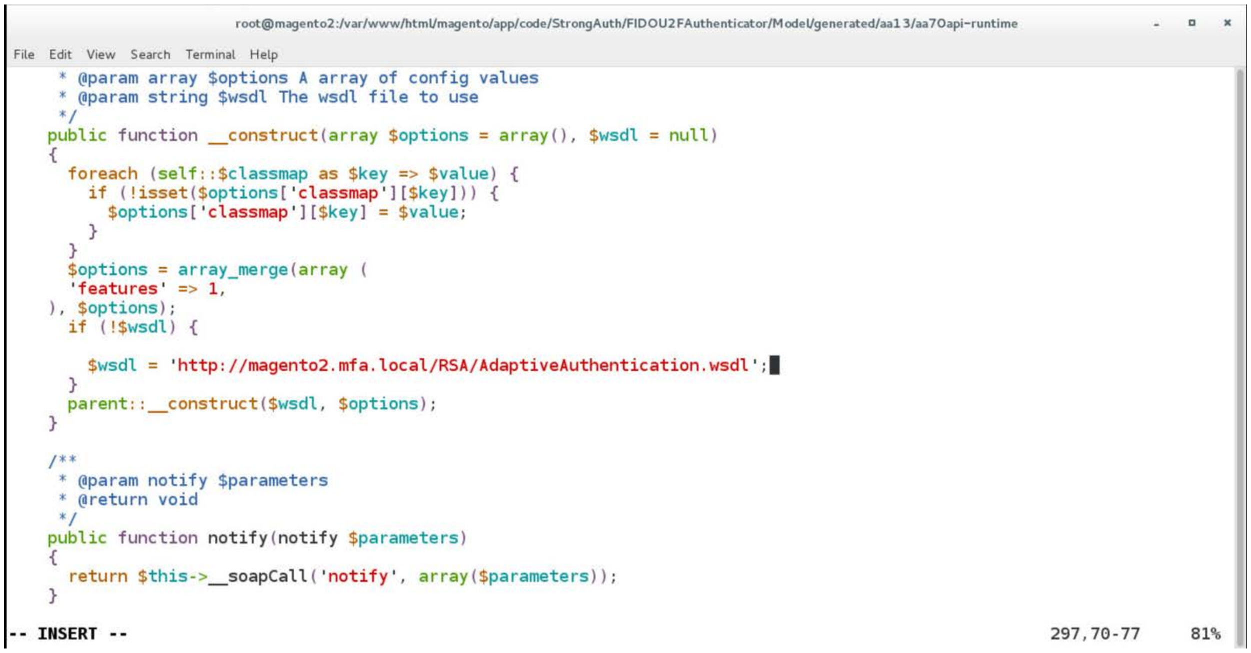
```
vim AdaptiveAuthentication.php
```


```
root@magento2:/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model/generated/aa13/aa70api-runtime
File  Edit  View  Search  Terminal  Help
[root@magento2 aa70api-runtime]# vim AdaptiveAuthentication.php
```

11. Make edits in the Adaptive Authentication file by pressing the **i** key to enter insert mode.

12. Change Line 297 of the document to the following line:

    ```
    $wsdl = 'http://magento2.mfa.local/RSA/AdaptiveAuthentication.wsdl';
    ```

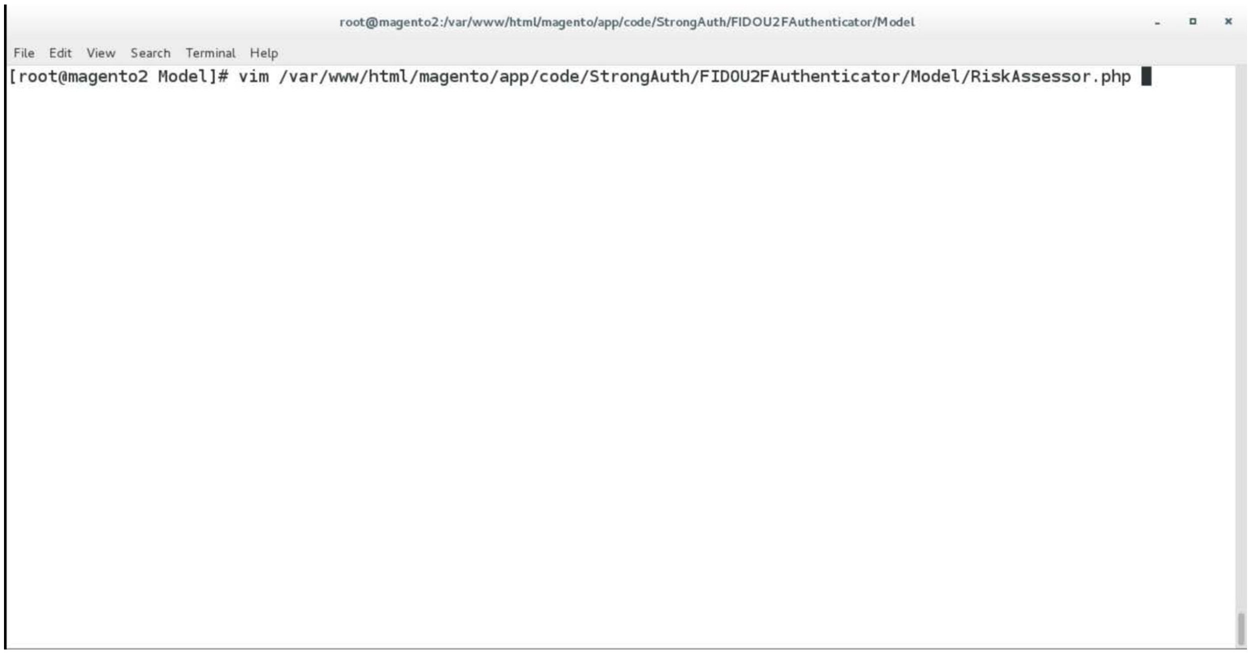

```
root@magento2:/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model/generated/aa13/aa70api-runtime
File  Edit  View  Search  Terminal  Help
     * @param array $options A array of config values
     * @param string $wsdl The wsdl file to use
     */
    public function __construct(array $options = array(), $wsdl = null)
    {
      foreach (self::$classmap as $key => $value) {
        if (!isset($options['classmap'][$key])) {
          $options['classmap'][$key] = $value;
        }
      }
      $options = array_merge(array (
      'features' => 1,
    ), $options);
      if (!$wsdl) {

        $wsdl = 'http://magento2.mfa.local/RSA/AdaptiveAuthentication.wsdl';
      }
      parent::__construct($wsdl, $options);
    }

    /**
     * @param notify $parameters
     * @return void
     */
    public function notify(notify $parameters)
    {
      return $this->__soapCall('notify', array($parameters));
    }
-- INSERT --                                                          297,70-77     81%
```

13. Press the **Esc** key to exit insert mode.

14. Save changes, and exit by entering the following command: `:wq`.

15. Edit the RSA Risk Assessor File by entering the following command:

```
vim
/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model/RiskAssess
or.php
```



root@magento2:/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model
File Edit View Search Terminal Help

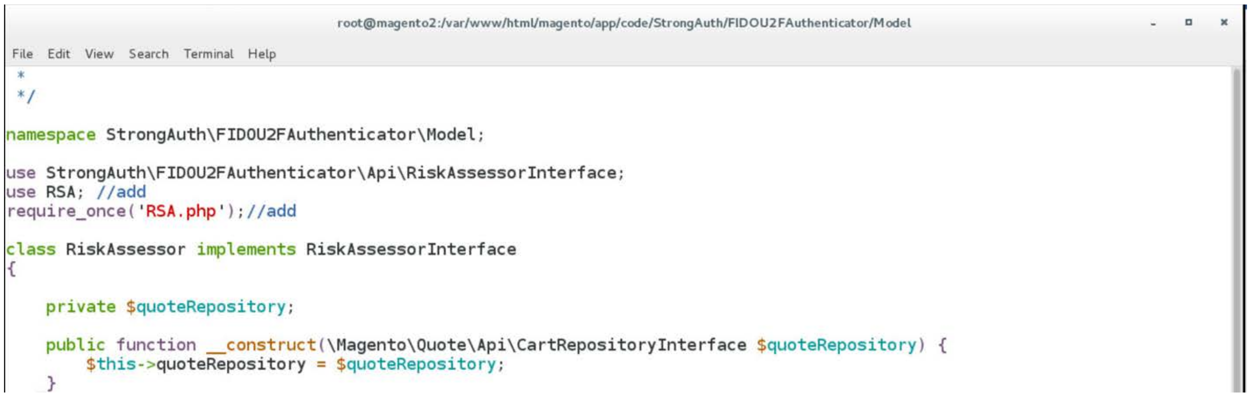
```
[root@magento2 Model]# vim /var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model/RiskAssessor.php
```

16. Press the **i** key to enter editor mode.

17. Make the following changes to the *RiskAssessor.php* file:

    a. After Line 41, add the following two lines:

    ```
    use RSA;
    require_once('RSA.php');
    ```

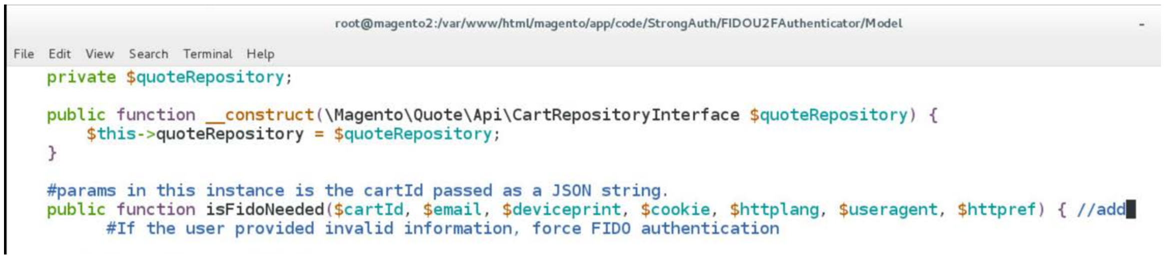


root@magento2:/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model
File Edit View Search Terminal Help

```
 *
 */

namespace StrongAuth\FIDOU2FAuthenticator\Model;

use StrongAuth\FIDOU2FAuthenticator\Api\RiskAssessorInterface;
use RSA; //add
require_once('RSA.php');//add

class RiskAssessor implements RiskAssessorInterface
{

    private $quoteRepository;

    public function __construct(\Magento\Quote\Api\CartRepositoryInterface $quoteRepository) {
        $this->quoteRepository = $quoteRepository;
    }
```

b. Change Line 55 to the following line:

```
Public function isFidoNeeded($cartId, $email, $deviceprint, $cookie,
$httplan, $useragent, $httpref)
```

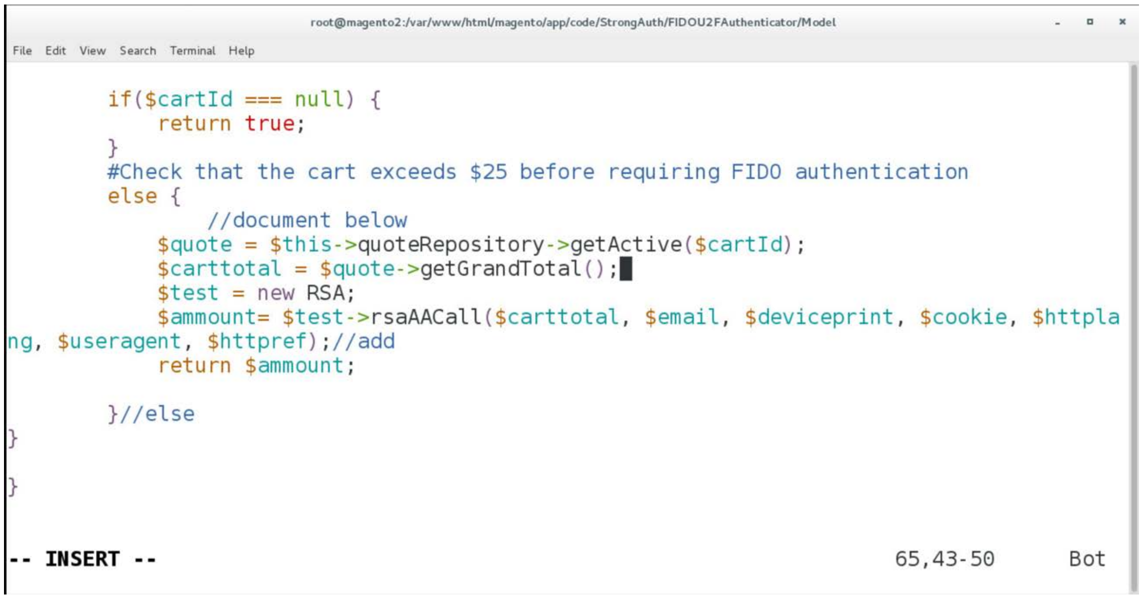


root@magento2:/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model
File Edit View Search Terminal Help

```
    private $quoteRepository;

    public function __construct(\Magento\Quote\Api\CartRepositoryInterface $quoteRepository) {
        $this->quoteRepository = $quoteRepository;
    }

    #params in this instance is the cartId passed as a JSON string.
    public function isFidoNeeded($cartId, $email, $deviceprint, $cookie, $httplang, $useragent, $httpref) { //add
        #If the user provided invalid information, force FIDO authentication
```

c. After Line 65, edit the following lines:

```
$test = new RSA;

$amount = $test->rsaAACall($cartId, $email, $deviceprint, $cookie,
$httplan, $useragent, $httpref);

return $amount;
```


root@magento2:/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model
File Edit View Search Terminal Help

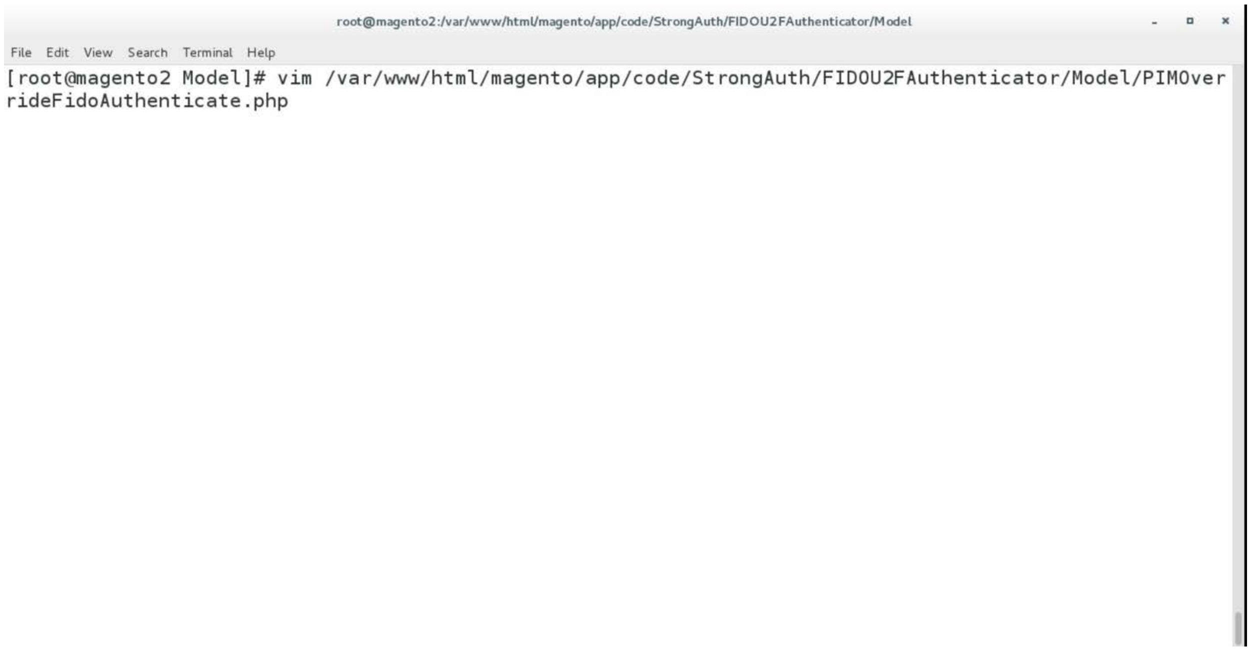
```
        if($cartId === null) {
            return true;
        }
        #Check that the cart exceeds $25 before requiring FIDO authentication
        else {
                //document below
            $quote = $this->quoteRepository->getActive($cartId);
            $carttotal = $quote->getGrandTotal();
            $test = new RSA;
            $ammount= $test->rsaAACall($carttotal, $email, $deviceprint, $cookie, $httpla
ng, $useragent, $httpref);//add
            return $ammount;

        }//else
}

}

-- INSERT --                                                            65,43-50      Bot
```

d. Press the **Esc** key to exit insert mode.

e. Save changes, and exit by entering the following command: `:wq`.

18. Open the *PIMOverrideFidoAuthenticate.php* file in the vim editor by entering the following command:

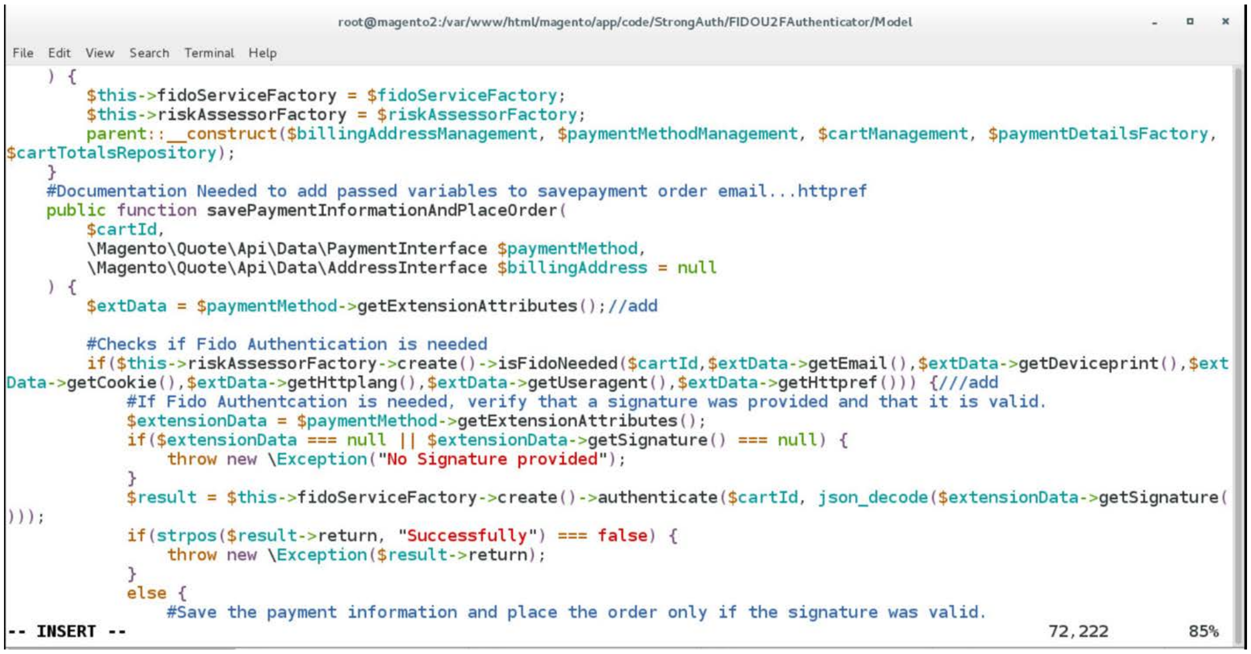
```
vim
/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model/PIMOverrid
eFidoAuthenticate.php
```


root@magento2:/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model
File Edit View Search Terminal Help

```
[root@magento2 Model]# vim /var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model/PIMOver
rideFidoAuthenticate.php
```

19. Press the **i** key to enter editor mode.
20. Make the following changes to the *PIMOverrideFidoAuthenticate.php* file:
    a. Between Lines 68 and 72, edit the following lines:

```
extData = $paymentMethod->getExtensionAttributes();

if($this->riskAssessorFactory->create()->isFidoNeeded($cartId,$extData-
>getEmail(),$extData->getDeviceprint(),$extData->getCookie,$extData-
>getHttplang(),$extData->getUseragent,$extData->getHttpref())) {
```


root@magento2:/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model
File Edit View Search Terminal Help

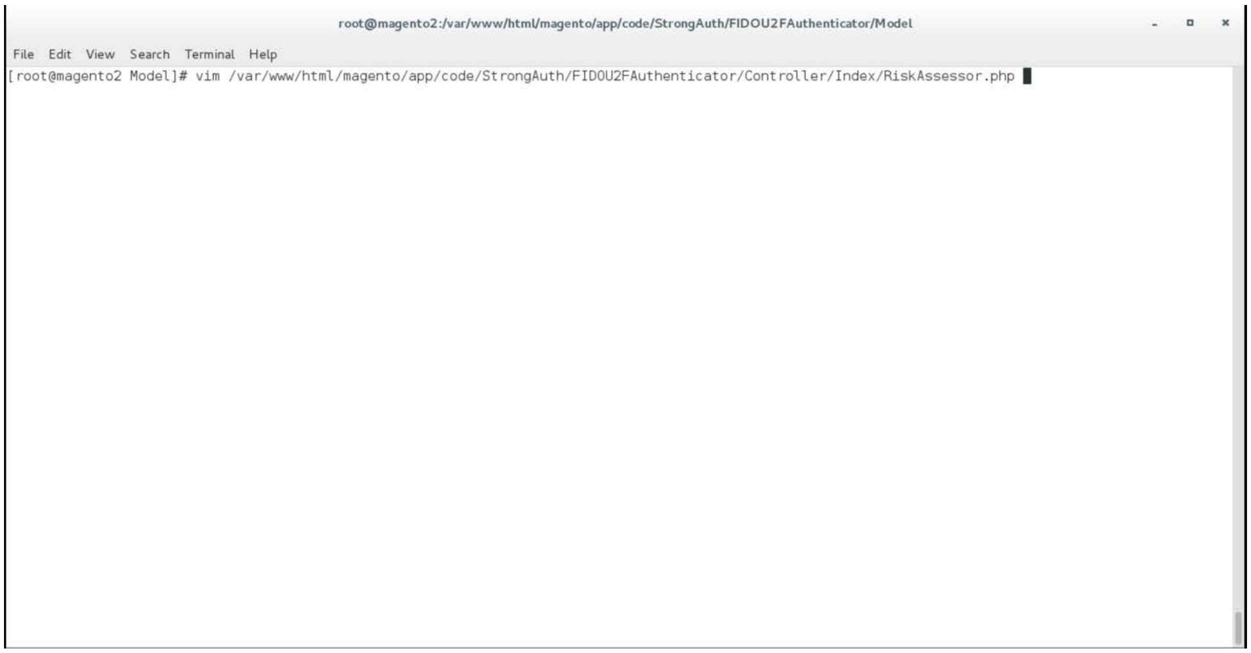
```
    ) {
        $this->fidoServiceFactory = $fidoServiceFactory;
        $this->riskAssessorFactory = $riskAssessorFactory;
        parent::__construct($billingAddressManagement, $paymentMethodManagement, $cartManagement, $paymentDetailsFactory,
$cartTotalsRepository);
    }
    #Documentation Needed to add passed variables to savepayment order email...httpref
    public function savePaymentInformationAndPlaceOrder(
        $cartId,
        \Magento\Quote\Api\Data\PaymentInterface $paymentMethod,
        \Magento\Quote\Api\Data\AddressInterface $billingAddress = null
    ) {
        $extData = $paymentMethod->getExtensionAttributes();//add

        #Checks if Fido Authentication is needed
        if($this->riskAssessorFactory->create()->isFidoNeeded($cartId,$extData->getEmail(),$extData->getDeviceprint(),$ext
Data->getCookie(),$extData->getHttplang(),$extData->getUseragent(),$extData->getHttpref())) {///add
            #If Fido Authentcation is needed, verify that a signature was provided and that it is valid.
            $extensionData = $paymentMethod->getExtensionAttributes();
            if($extensionData === null || $extensionData->getSignature() === null) {
                throw new \Exception("No Signature provided");
            }
            $result = $this->fidoServiceFactory->create()->authenticate($cartId, json_decode($extensionData->getSignature(
)));

            if(strpos($result->return, "Successfully") === false) {
                throw new \Exception($result->return);
            }
            else {
                #Save the payment information and place the order only if the signature was valid.
-- INSERT --                                                                                         72,222        85%
```

b. Press the **Esc** key to exit insert mode.

c. Save changes, and exit by entering the following command: `:wq`.

21. Open the RSA RiskAssessor Controller file by entering the following command:

```
vim
/var/www/html/magento/StrongAuth/FIDOU2FAuthenticator/Controller/Index/Riskasse
ssor.php
```


root@magento2:/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model
File Edit View Search Terminal Help

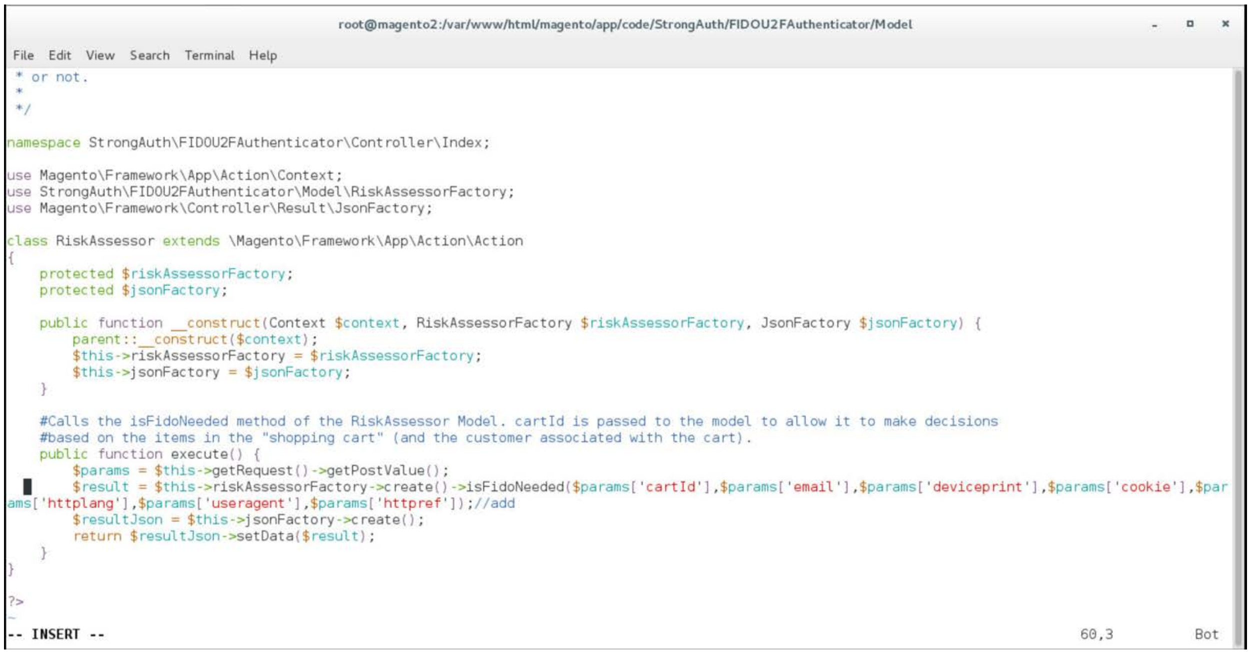
```
[root@magento2 Model]# vim /var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Controller/Index/RiskAssessor.php
```

22. Press the **i** key to enter editor mode.

23. Make the following changes to the *RiskAssessor.php* file:

    a. Change Line 60 to the following line:

    ```
    $result = $this->riskAssessorFactory->create()-
    >isFidoNeeded($params['cartId'], $params['email'],
    $params['deviceprint'], $params['cookie'], $params['httplang'],
    $params['useragent'], $params['httpref']);
    ```


root@magento2:/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/Model
File Edit View Search Terminal Help

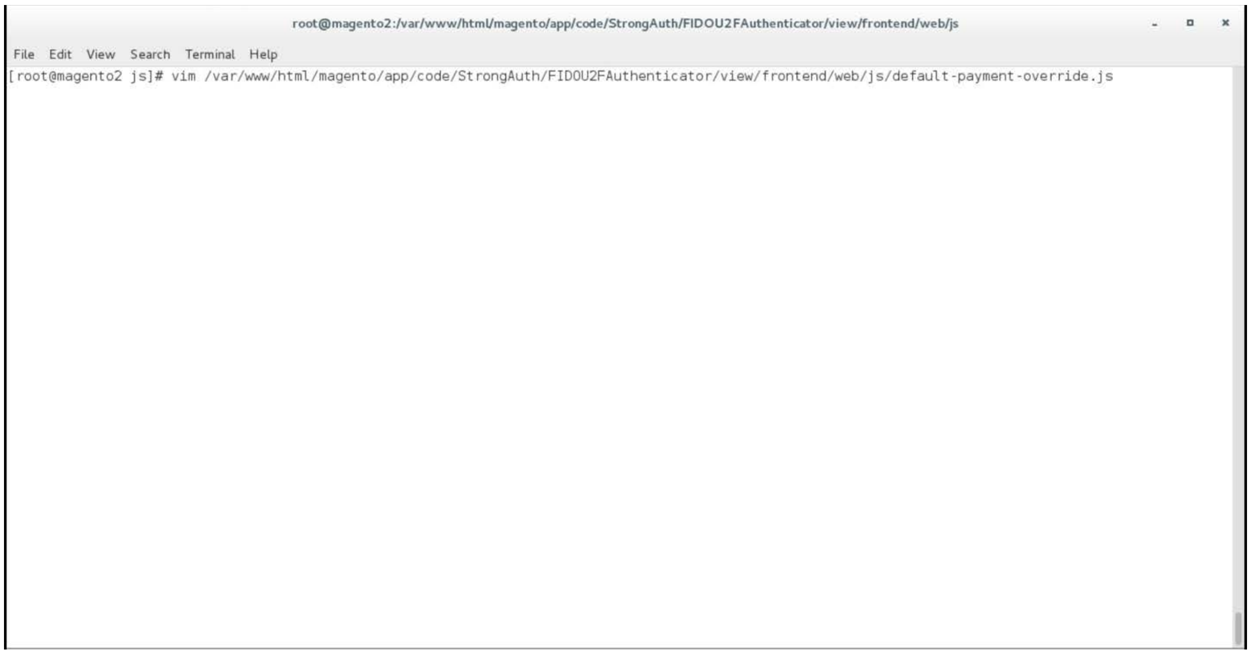
```
 * or not.
 *
 */

namespace StrongAuth\FIDOU2FAuthenticator\Controller\Index;

use Magento\Framework\App\Action\Context;
use StrongAuth\FIDOU2FAuthenticator\Model\RiskAssessorFactory;
use Magento\Framework\Controller\Result\JsonFactory;

class RiskAssessor extends \Magento\Framework\App\Action\Action
{
    protected $riskAssessorFactory;
    protected $jsonFactory;

    public function __construct(Context $context, RiskAssessorFactory $riskAssessorFactory, JsonFactory $jsonFactory) {
        parent::__construct($context);
        $this->riskAssessorFactory = $riskAssessorFactory;
        $this->jsonFactory = $jsonFactory;
    }

    #Calls the isFidoNeeded method of the RiskAssessor Model. cartId is passed to the model to allow it to make decisions
    #based on the items in the "shopping cart" (and the customer associated with the cart).
    public function execute() {
        $params = $this->getRequest()->getPostValue();
        $result = $this->riskAssessorFactory->create()->isFidoNeeded($params['cartId'],$params['email'],$params['deviceprint'],$params['cookie'],$par
ams['httplang'],$params['useragent'],$params['httpref']);//add
        $resultJson = $this->jsonFactory->create();
        return $resultJson->setData($result);
    }
}

?>
~
-- INSERT --                                                                                              60,3          Bot
```

b. Press the **Esc** key to exit insert mode.

c. Save changes, and exit by entering the following command: `:wq`.

24. Open the RSA JavaScript Override file by entering the following command:

```
vim
/var/www/html/magento/StrongAuth/FIDOU2FAuthenticator/view/frontend/web/js/defa
ult-payment-override.js
```


root@magento2:/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/view/frontend/web/js
File Edit View Search Terminal Help

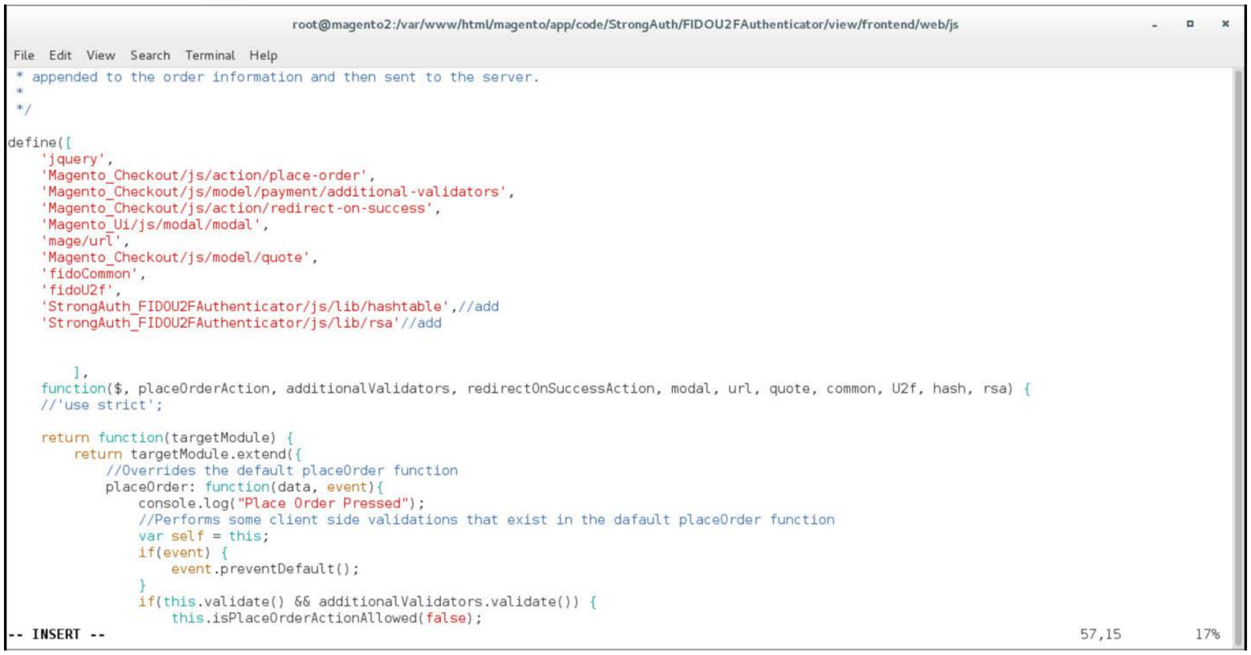
```
[root@magento2 js]# vim /var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/view/frontend/web/js/default-payment-override.js
```

25. Press the **i** key to enter editor mode.

26. Make the following changes to the *default-payment-override.js* file:

    a. Add the following two lines after Line 57:

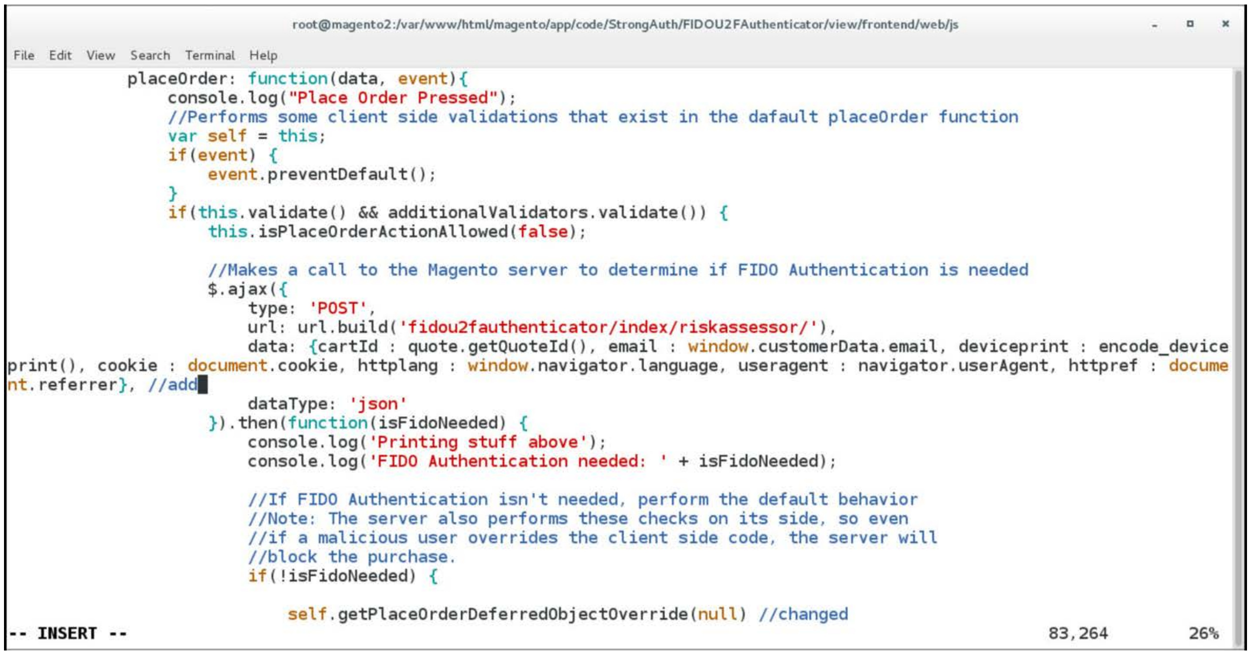
    ```
    'StrongAuth_FIDOU2FAuthenticator/js/lib/hashtable',
    'StrongAuth_FIDOU2FAuthenticator/js/lib/rsa'
    ```



```
root@magento2:/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/view/frontend/web/js
File Edit View Search Terminal Help
 * appended to the order information and then sent to the server.
 *
 */

define([
    'jquery',
    'Magento_Checkout/js/action/place-order',
    'Magento_Checkout/js/model/payment/additional-validators',
    'Magento_Checkout/js/action/redirect-on-success',
    'Magento_Ui/js/modal/modal',
    'mage/url',
    'Magento_Checkout/js/model/quote',
    'fidoCommon',
    'fidoU2f',
    'StrongAuth_FIDOU2FAuthenticator/js/lib/hashtable',//add
    'StrongAuth_FIDOU2FAuthenticator/js/lib/rsa'//add


        ],
    function($, placeOrderAction, additionalValidators, redirectOnSuccessAction, modal, url, quote, common, U2f, hash, rsa) {
    //'use strict';

    return function(targetModule) {
        return targetModule.extend({
            //Overrides the default placeOrder function
            placeOrder: function(data, event){
                console.log("Place Order Pressed");
                //Performs some client side validations that exist in the dafault placeOrder function
                var self = this;
                if(event) {
                    event.preventDefault();
                }
                if(this.validate() && additionalValidators.validate()) {
                    this.isPlaceOrderActionAllowed(false);
-- INSERT --                                                                                    57,15         17%
```

b. Change Line 83 to the following line:

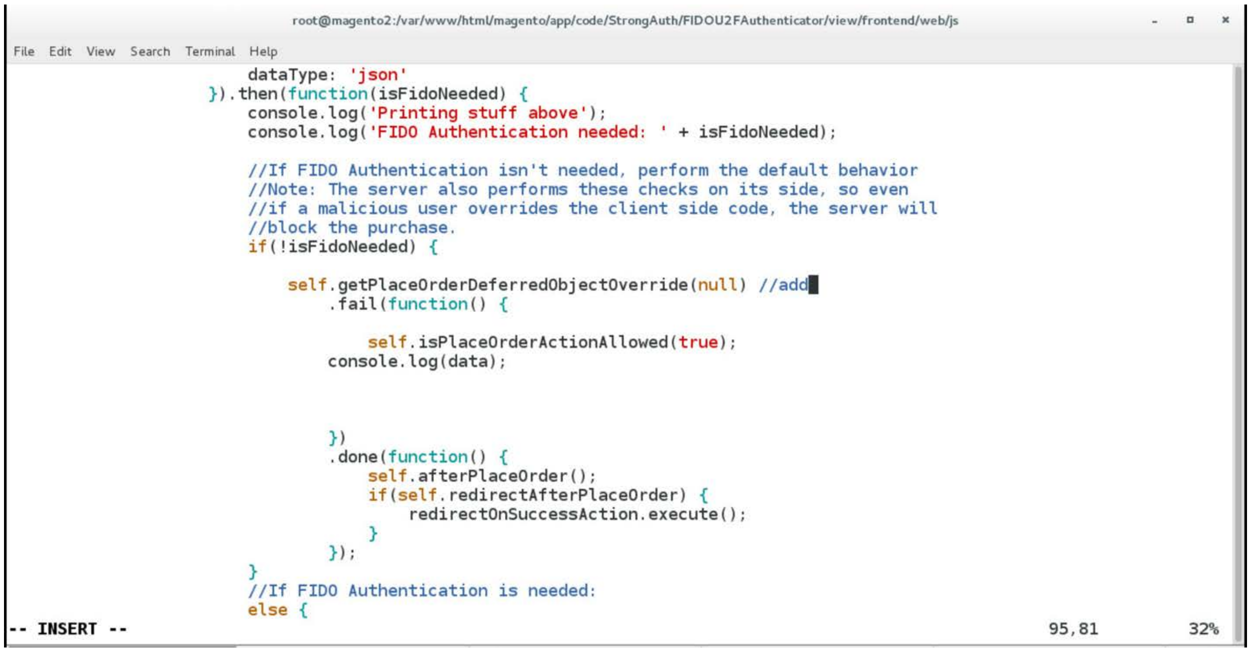
```
Data: {cartId: quote.getQuoteId(), email : window.customerData.email,
deviceprint : encode_deviceprint(), cookie: document.cookie, httplang :
window.navigator.language, useragent : navigator.userAgent, httpref :
document.referrer},
```



```
root@magento2:/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/view/frontend/web/js
File Edit View Search Terminal Help
            placeOrder: function(data, event){
                console.log("Place Order Pressed");
                //Performs some client side validations that exist in the dafault placeOrder function
                var self = this;
                if(event) {
                    event.preventDefault();
                }
                if(this.validate() && additionalValidators.validate()) {
                    this.isPlaceOrderActionAllowed(false);

                    //Makes a call to the Magento server to determine if FIDO Authentication is needed
                    $.ajax({
                        type: 'POST',
                        url: url.build('fidou2fauthenticator/index/riskassessor/'),
                        data: {cartId : quote.getQuoteId(), email : window.customerData.email, deviceprint : encode_device
print(), cookie : document.cookie, httplang : window.navigator.language, useragent : navigator.userAgent, httpref : docume
nt.referrer}, //add
                        dataType: 'json'
                    }).then(function(isFidoNeeded) {
                        console.log('Printing stuff above');
                        console.log('FIDO Authentication needed: ' + isFidoNeeded);

                        //If FIDO Authentication isn't needed, perform the default behavior
                        //Note: The server also performs these checks on its side, so even
                        //if a malicious user overrides the client side code, the server will
                        //block the purchase.
                        if(!isFidoNeeded) {

                            self.getPlaceOrderDeferredObjectOverride(null) //changed
-- INSERT --                                                                               83,264        26%
```

c. Change Line 95 to the following line:

```
self.getPlaceOrderDeferredObjectOverride(null)
```

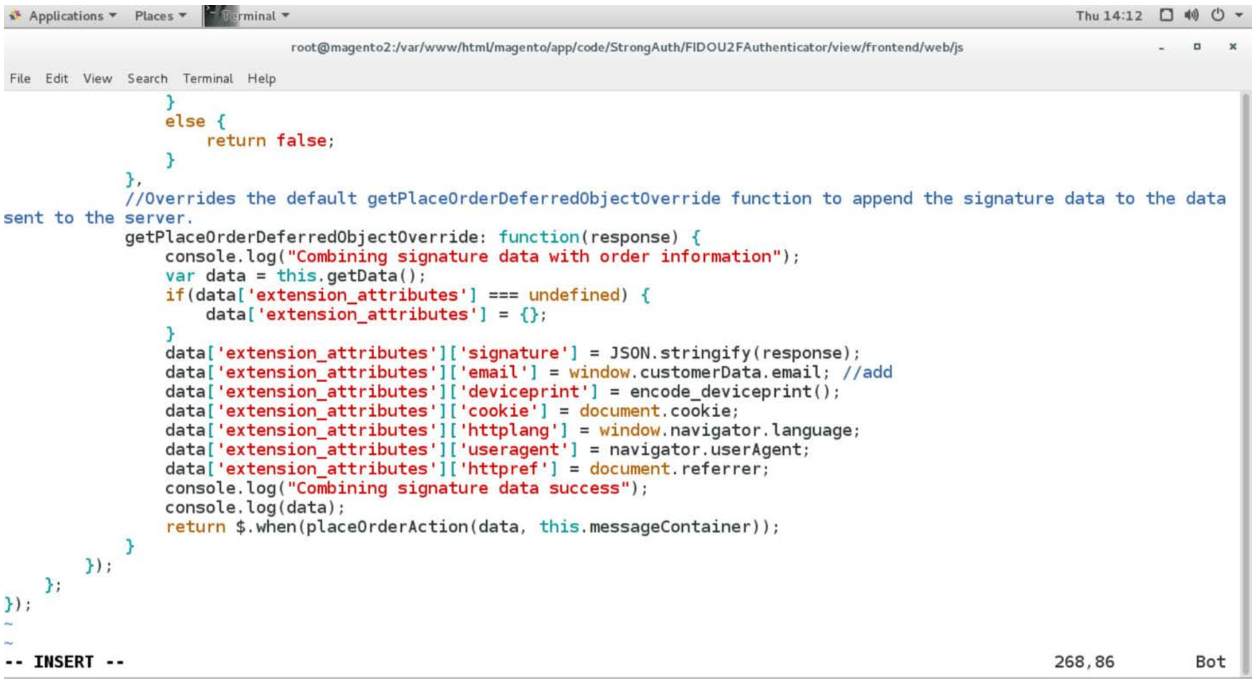


root@magento2:/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/view/frontend/web/js
File Edit View Search Terminal Help

```
                    dataType: 'json'
                }).then(function(isFidoNeeded) {
                    console.log('Printing stuff above');
                    console.log('FIDO Authentication needed: ' + isFidoNeeded);

                    //If FIDO Authentication isn't needed, perform the default behavior
                    //Note: The server also performs these checks on its side, so even
                    //if a malicious user overrides the client side code, the server will
                    //block the purchase.
                    if(!isFidoNeeded) {

                        self.getPlaceOrderDeferredObjectOverride(null) //add
                            .fail(function() {

                                self.isPlaceOrderActionAllowed(true);
                            console.log(data);



                            })
                            .done(function() {
                                self.afterPlaceOrder();
                                if(self.redirectAfterPlaceOrder) {
                                    redirectOnSuccessAction.execute();
                                }
                            });
                    }
                    //If FIDO Authentication is needed:
                    else {
-- INSERT --                                                                95,81         32%
```

d. After Line 268, add the following lines:

```
Data[‘extension_attributes’][‘email’] = window.customerData.email;
Data[‘extension_attributes’][‘deviceprint’] = encode_deviceprint();
Data[‘extension_attributes’][‘cookie’] = document.cookie;
Data[‘extension_attributes’][‘httplang’] = window.navigator.language;
Data[‘extension_attributes’][‘useragent’] = navigator.userAgent;
Data[‘extension_attributes’][‘httpref’] = document.referrer;
```


Applications
Places
Terminal
Thu 14:12
root@magento2:/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/view/frontend/web/js
File Edit View Search Terminal Help

```
                }
                else {
                    return false;
                }
            },
            //Overrides the default getPlaceOrderDeferredObjectOverride function to append the signature data to the data
sent to the server.
            getPlaceOrderDeferredObjectOverride: function(response) {
                console.log("Combining signature data with order information");
                var data = this.getData();
                if(data['extension_attributes'] === undefined) {
                    data['extension_attributes'] = {};
                }
                data['extension_attributes']['signature'] = JSON.stringify(response);
                data['extension_attributes']['email'] = window.customerData.email; //add
                data['extension_attributes']['deviceprint'] = encode_deviceprint();
                data['extension_attributes']['cookie'] = document.cookie;
                data['extension_attributes']['httplang'] = window.navigator.language;
                data['extension_attributes']['useragent'] = navigator.userAgent;
                data['extension_attributes']['httpref'] = document.referrer;
                console.log("Combining signature data success");
                console.log(data);
                return $.when(placeOrderAction(data, this.messageContainer));
            }
        });
    };
});
~
~
-- INSERT --                                                                                  268,86          Bot
```

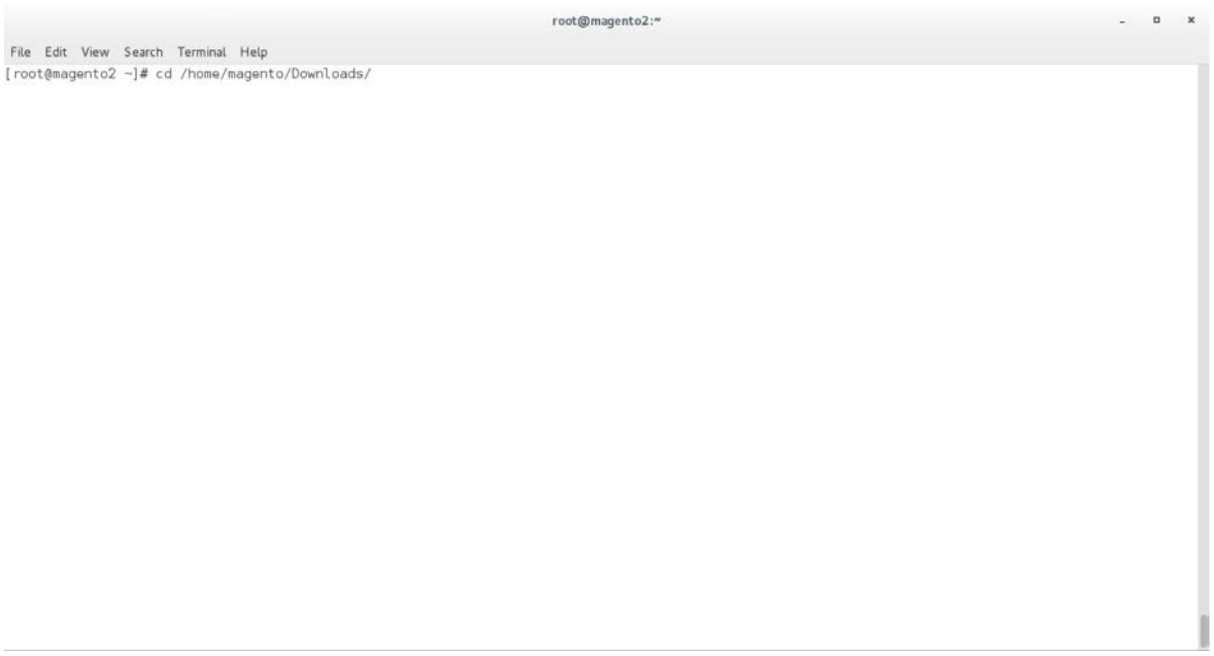
e. Press the **Esc** key to exit insert mode.

f. Save changes, and exit by entering the following command: `:wq`.

27. Download the RSA JavaScript files from your RSA representative.

28. Make the following change to the Downloads directory:

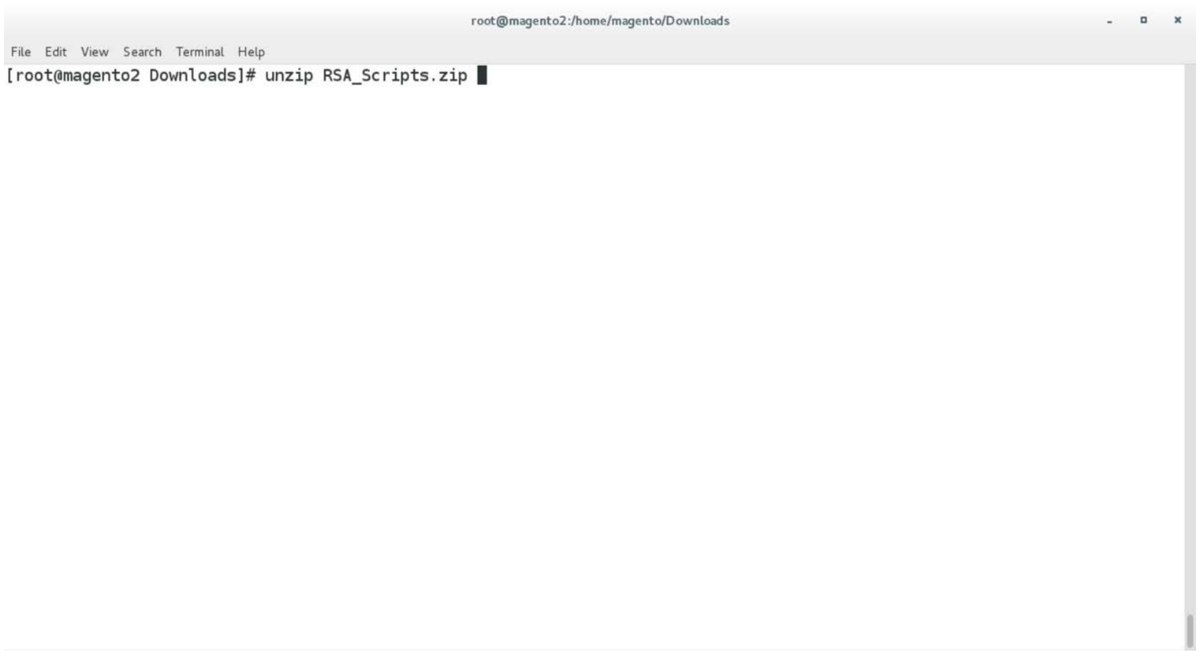
```
cd /home/magento/Downloads
```



```
[root@magento2 ~]# cd /home/magento/Downloads/
```

29. Unzip the contents of the RSA JavaScript folder by entering the following command:

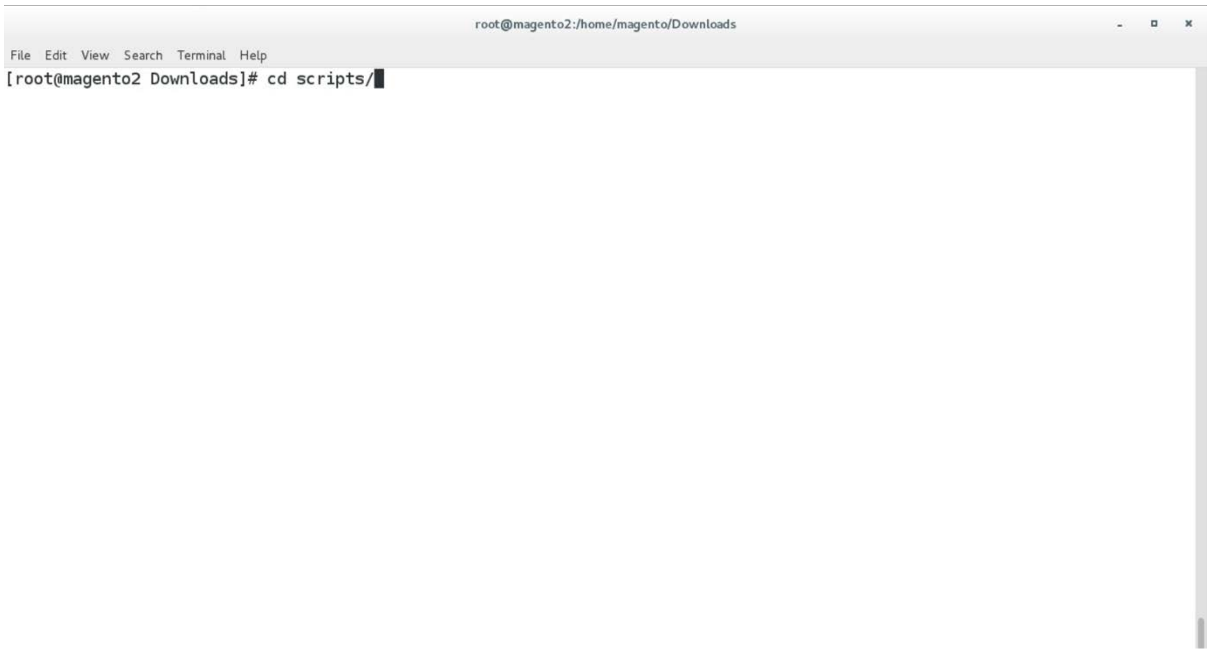
    ```
    unzip RSA_Scripts.zip
    ```


root@magento2:/home/magento/Downloads
File Edit View Search Terminal Help

```
[root@magento2 Downloads]# unzip RSA_Scripts.zip
```

30. Move to the newly unzipped scripts folder by entering the following command:

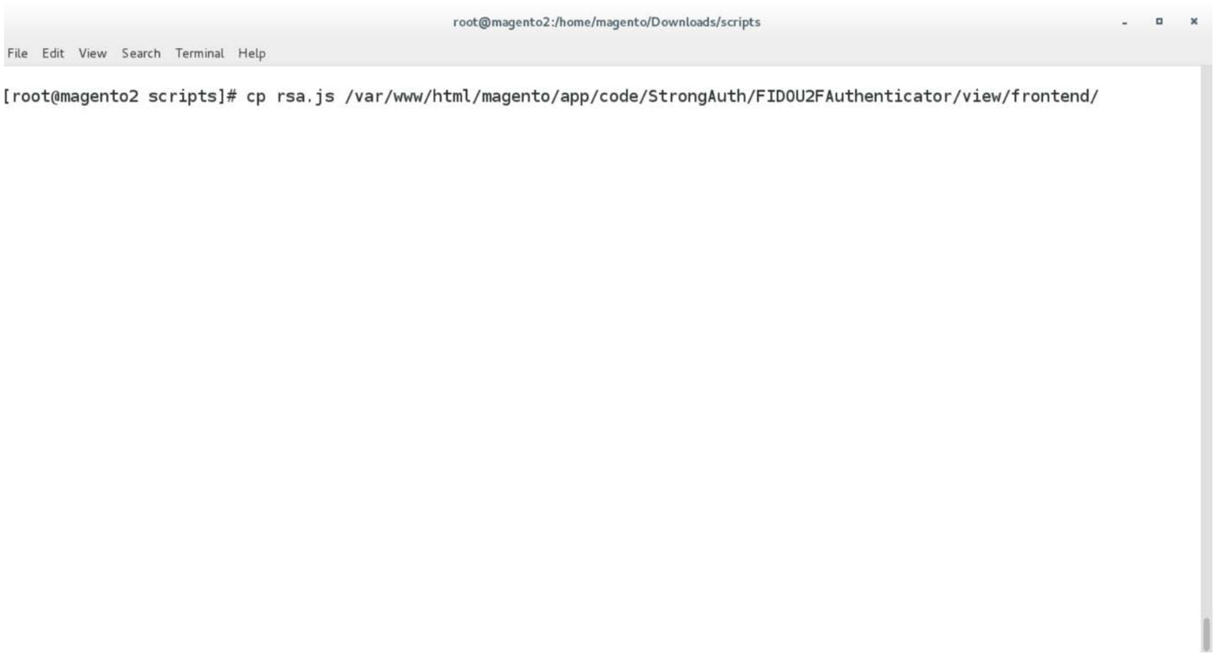
```
cd scripts/
```


root@magento2:/home/magento/Downloads
File Edit View Search Terminal Help

```
[root@magento2 Downloads]# cd scripts/
```

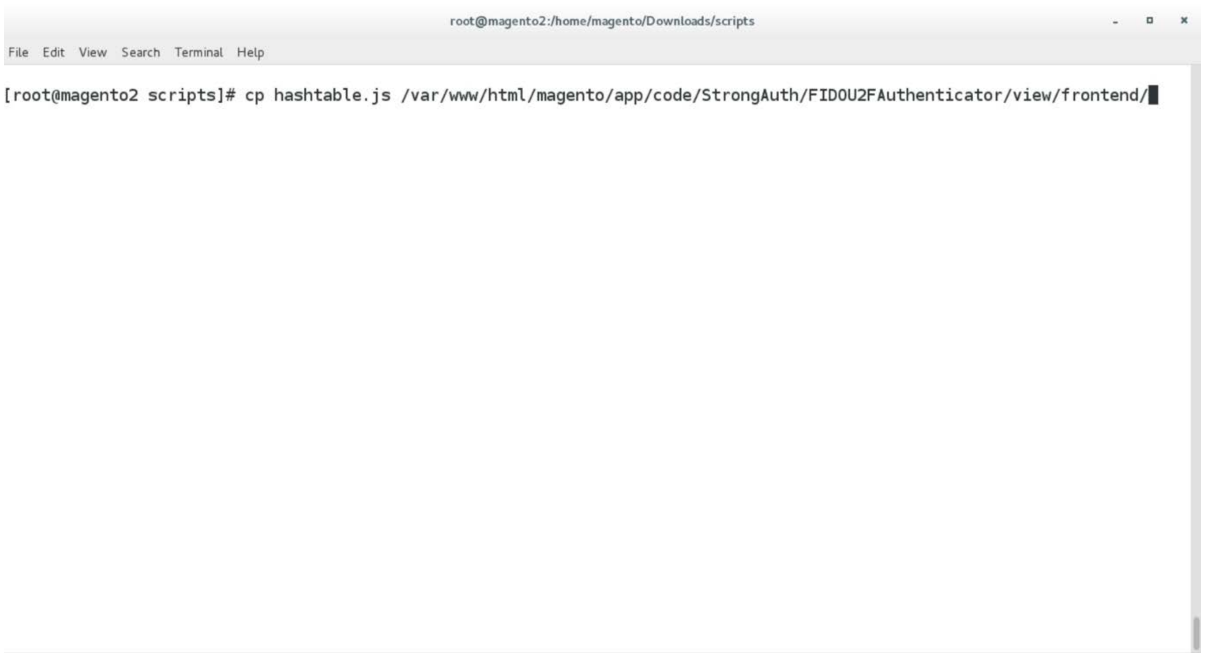
31. Copy the *rsa.js* and *hashtable.js* files to StrongAuth front-end JavaScript directory by entering the following commands:

    a.
    ```
    cp rsa.js
    /var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/view/front
    end/web/js/lib/
    ```


root@magento2:/home/magento/Downloads/scripts
File Edit View Search Terminal Help

```
[root@magento2 scripts]# cp rsa.js /var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/view/frontend/
```

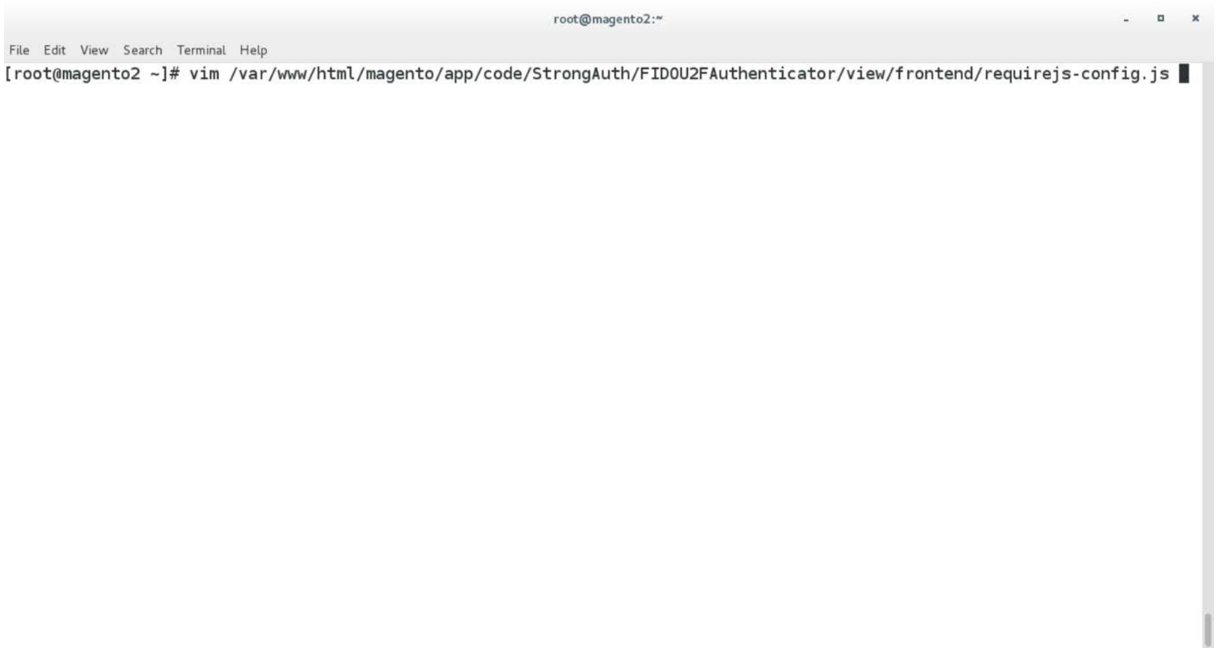
b. `cp hashtable.js /var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/view/frontend/web/js/lib/`


root@magento2:/home/magento/Downloads/scripts
File Edit View Search Terminal Help

```
[root@magento2 scripts]# cp hashtable.js /var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/view/frontend/
```

32. Open the StrongAuth JavaScript required file by entering the following command:

```
vim
/var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/view/frontendreq
uirejs-config.js
```



```
[root@magento2 ~]# vim /var/www/html/magento/app/code/StrongAuth/FIDOU2FAuthenticator/view/frontend/requirejs-config.js
```

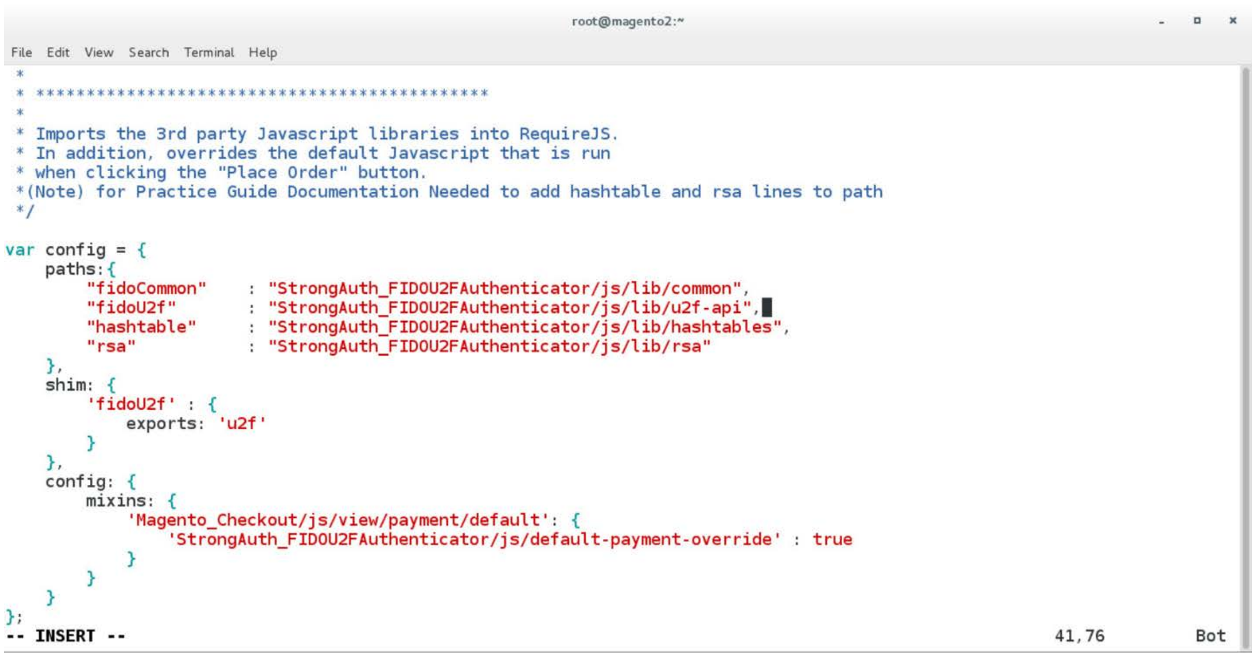
33. Press the **i** key to enter editor mode.

34. Make the following edits to the *requirejs-config.js* file:

    a. After Line 41, insert the following lines:

    ```
    “hashtable” : “StrongAuth_FIDOU2FAuthenticator/js/lib/hastables”,
    ```

    ```
    “rsa” : “StrongAuth_FIDOU2FAuthenticator/js/lib/rsa
    ```



```
 *
 * ********************************************
 *
 * Imports the 3rd party Javascript libraries into RequireJS.
 * In addition, overrides the default Javascript that is run
 * when clicking the "Place Order" button.
 *(Note) for Practice Guide Documentation Needed to add hashtable and rsa lines to path
 */

var config = {
    paths:{
        "fidoCommon"    : "StrongAuth_FIDOU2FAuthenticator/js/lib/common",
        "fidoU2f"       : "StrongAuth_FIDOU2FAuthenticator/js/lib/u2f-api",
        "hashtable"     : "StrongAuth_FIDOU2FAuthenticator/js/lib/hashtables",
        "rsa"           : "StrongAuth_FIDOU2FAuthenticator/js/lib/rsa"
    },
    shim: {
        'fidoU2f' : {
            exports: 'u2f'
        }
    },
    config: {
        mixins: {
            'Magento_Checkout/js/view/payment/default': {
                'StrongAuth_FIDOU2FAuthenticator/js/default-payment-override' : true
            }
        }
    }
};
-- INSERT --                                                              41,76         Bot
```

b. Press the **Esc** key to exit insert mode.

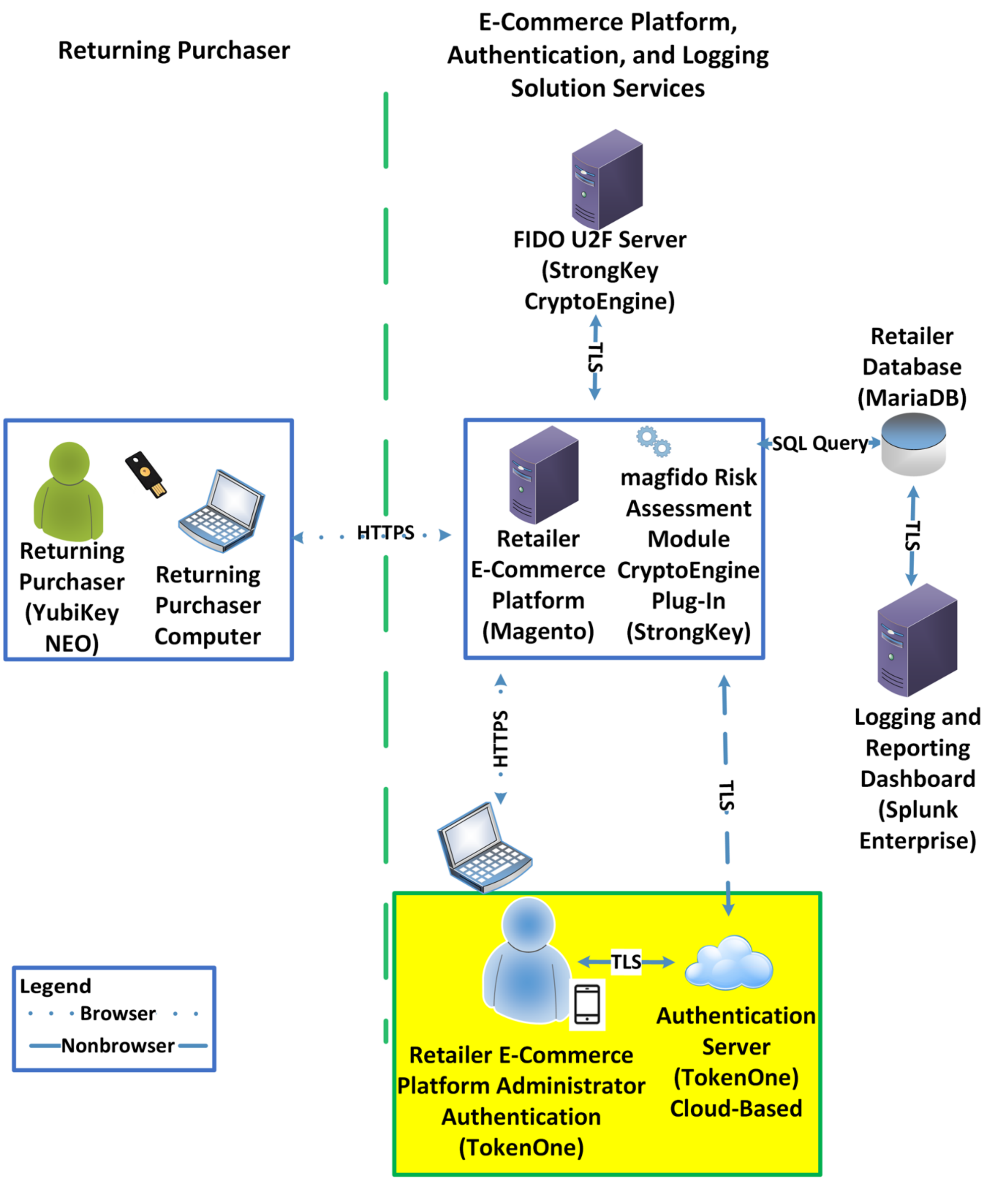
c. Save changes, and exit by entering the following command: `:wq`.

35. To create Adaptive Authentication policies, follow the product configuration instructions provided by your RSA representative.

2.5 TokenOne

This section provides installation and configuration guidance for TokenOne's authentication capability [9]. TokenOne's authentication product is used by the retailer e-commerce platform administrator when they are managing the Magento e-commerce platform. TokenOne developed a Magento connector that both the *cost threshold* and *risk engine* example implementations use. The TokenOne authentication components that are installed and configured in this section are illustrated in Figure 2-5 and are highlighted in yellow within the green box.

Figure 2-5 TokenOne Authentication Components


Returning Purchaser
E-Commerce Platform, Authentication, and Logging Solution Services
FIDO U2F Server (StrongKey CryptoEngine)
TLS
Retailer Database (MariaDB)
SQL Query
Returning Purchaser (YubiKey NEO)
Returning Purchaser Computer
HTTPS
Retailer E-Commerce Platform (Magento)
magfido Risk Assessment Module CryptoEngine Plug-In (StrongKey)
TLS
Logging and Reporting Dashboard (Splunk Enterprise)
HTTPS
TLS
TLS
Retailer E-Commerce Platform Administrator Authentication (TokenOne)
Authentication Server (TokenOne) Cloud-Based
Legend
Browser
Nonbrowser

2.5.1 TokenOne Overview

TokenOne allows software-based authentication through a onetime personal identification number (PIN). The Magento Admin URI portal has been configured to use Second Factor Authentication with TokenOne. When accessing Magento with TokenOne's authentication capability, the user's numeric PIN is not entered, transmitted, or stored, but the corresponding letter code—which is entered when accessing Magento—is different every time that the user accesses the system. The TokenOne smartphone application is not push-button. The user always enters the code in the Magento administration interface.

The installation procedure consists of the following steps:

- Preinstallation:
  - Download the TokenOne application.
  - Download the TokenOne module.
- Installation and configuration:
  - Integrate the TokenOne module into Magento.
  - Test connectivity and authentication.

2.5.2 Preinstallation Steps

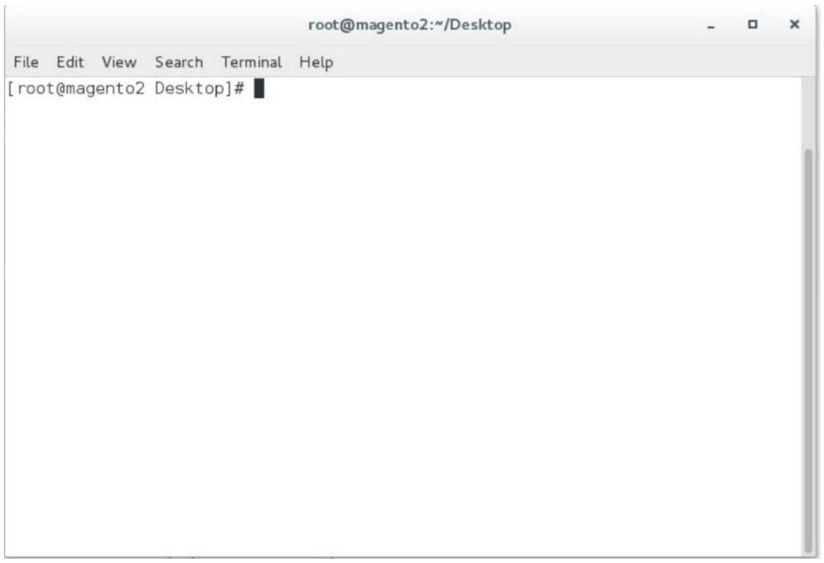
Before beginning installation, ensure that the following steps are completed:

- Download and install the TokenOne mobile application from either the Apple App Store or the Google Play Store.
- Speak with your TokenOne representative to receive the *TokenOne10.zip* file.
- Download the *TokenOne10.zip* file to the */home/magento/Downloads* directory.

2.5.3 TokenOne Installation and Configuration

To begin installation, perform the following steps:

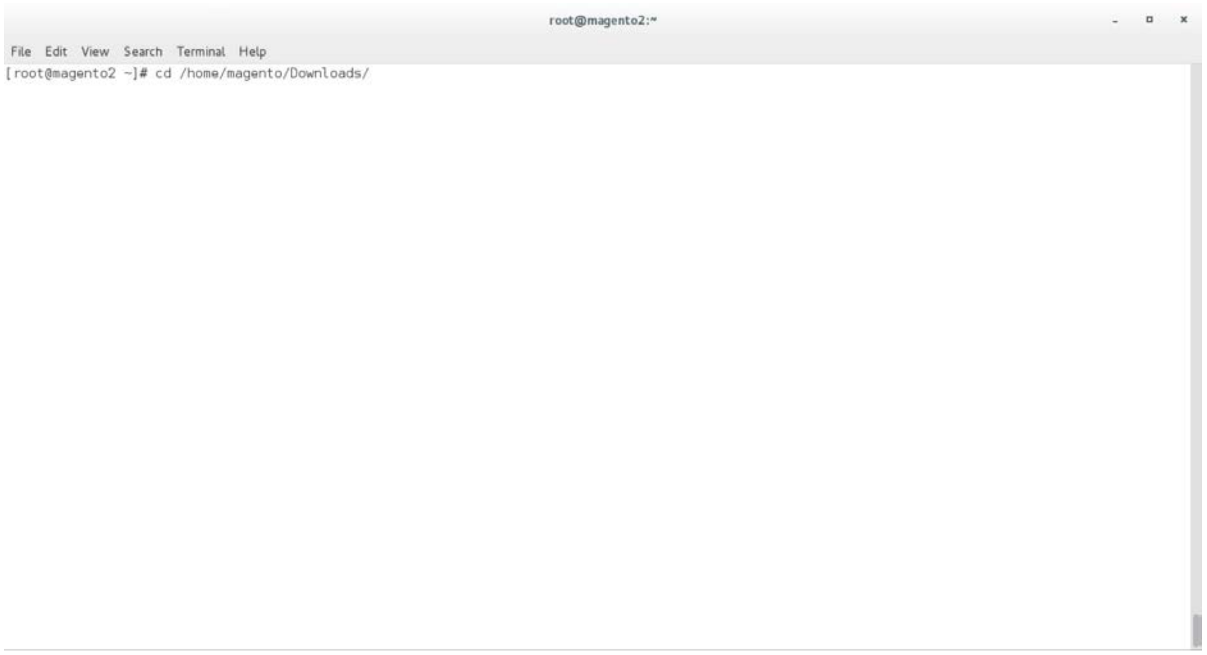
1. Open a terminal window.



```
[root@magento2 Desktop]#
```

2. Change to the Downloads directory by entering the following command:

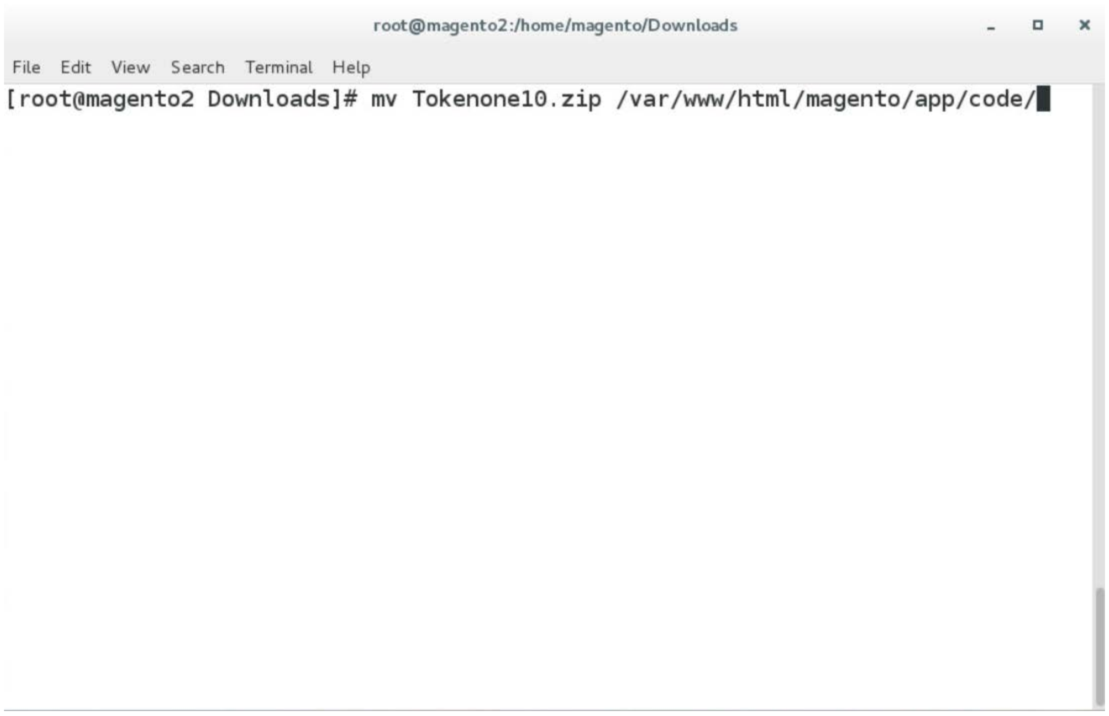
   `cd /home/magento/Downloads`



```
[root@magento2 ~]# cd /home/magento/Downloads/
```

3. Move the *Tokenone10.zip* file to the Magento application code directory by entering the following command:

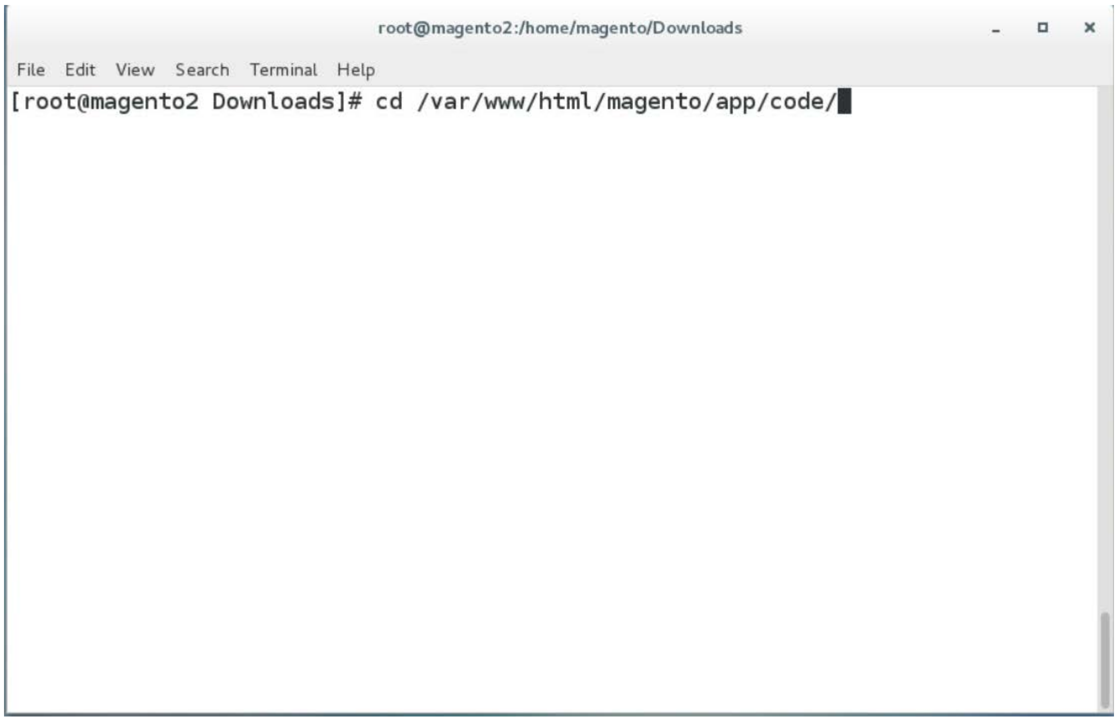
   `mv Tokenone10.zip /var/www/html/magento/app/code/`


root@magento2:/home/magento/Downloads
File Edit View Search Terminal Help

```
[root@magento2 Downloads]# mv Tokenone10.zip /var/www/html/magento/app/code/
```

4. Change to the Magento application directory by entering the following command:

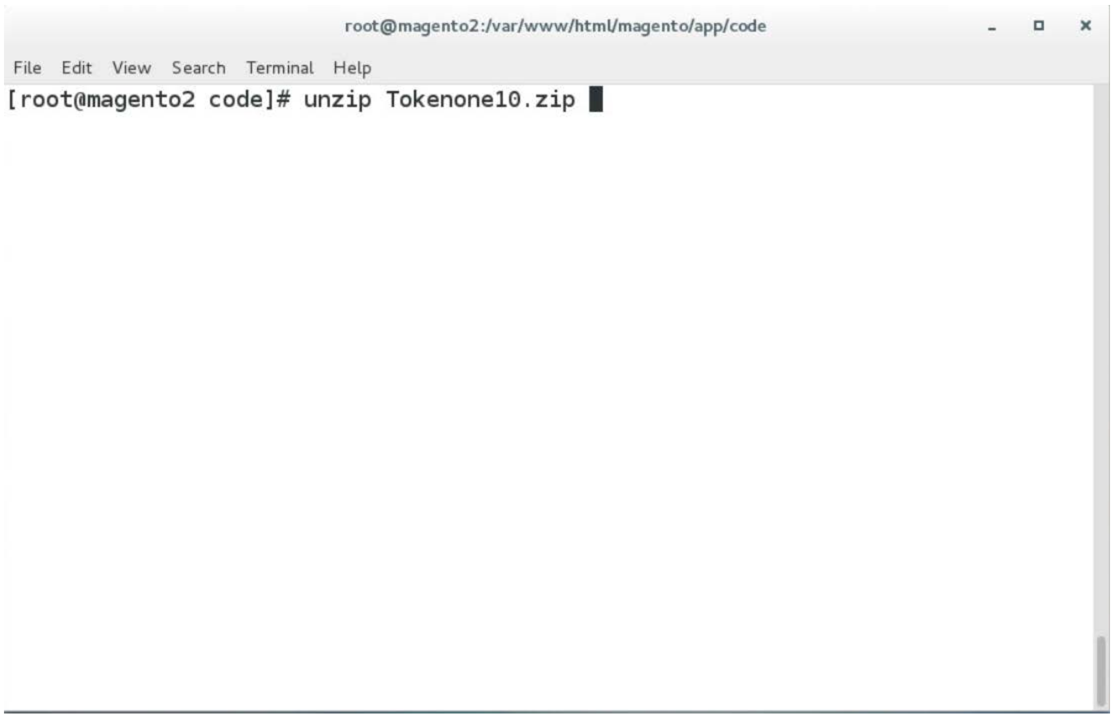
   ```
   cd /var/www/html/magento/app/code/
   ```


root@magento2:/home/magento/Downloads
File Edit View Search Terminal Help

```
[root@magento2 Downloads]# cd /var/www/html/magento/app/code/
```

5. Unzip the TokenOne zip file by entering the following command:

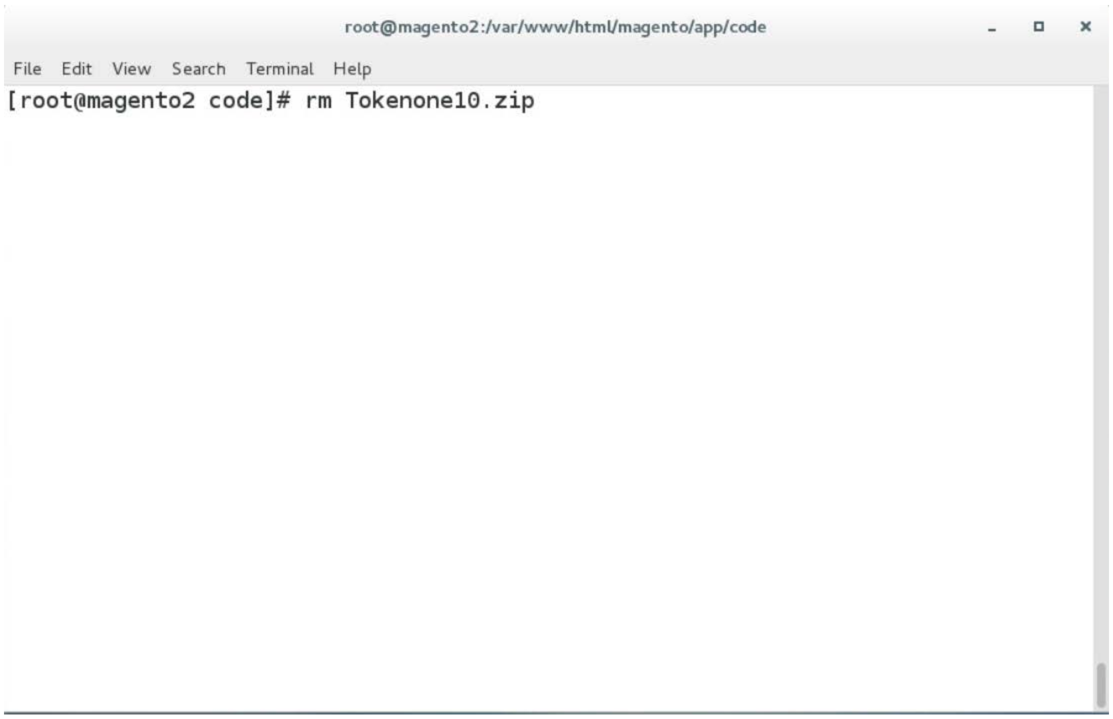
   ```
   unzip Tokenone10.zip
   ```


root@magento2:/var/www/html/magento/app/code
File Edit View Search Terminal Help

```
[root@magento2 code]# unzip Tokenone10.zip
```

6. Remove the zip file from the code directory by entering the following command:

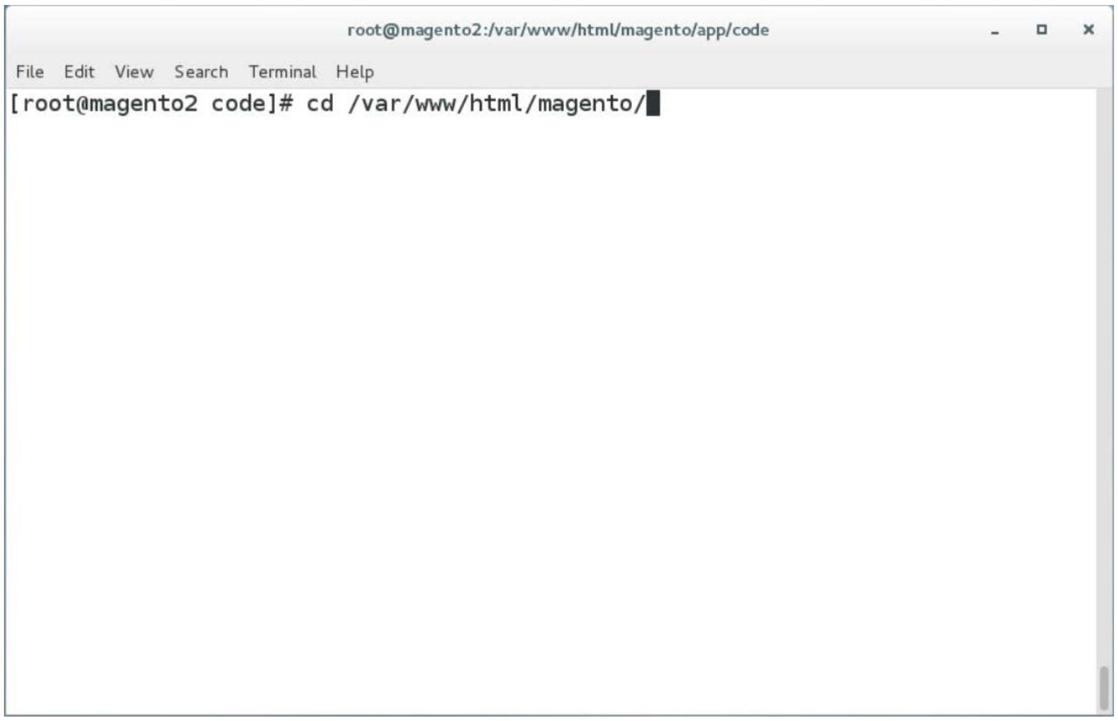
   ```
   rm Tokenone10.zip
   ```


root@magento2:/var/www/html/magento/app/code
File Edit View Search Terminal Help

```
[root@magento2 code]# rm Tokenone10.zip
```

7. Change to the Magento web server directory by entering the following command:

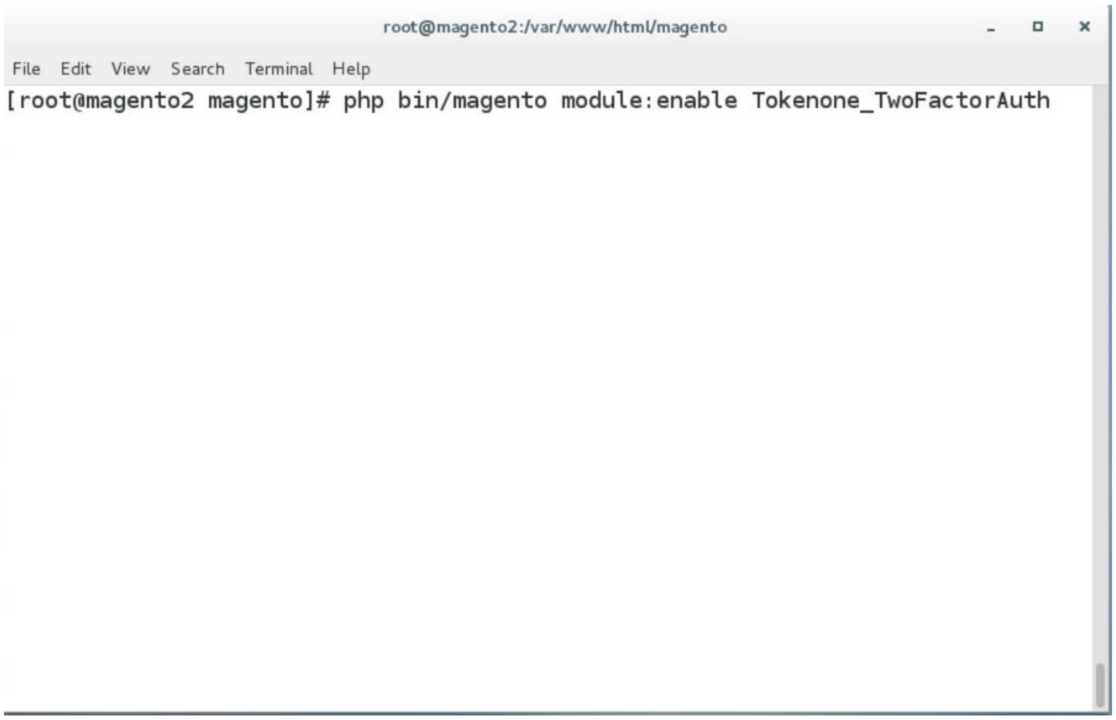
   ```
   cd /var/www/html/magento/
   ```


root@magento2:/var/www/html/magento/app/code
File Edit View Search Terminal Help

```
[root@magento2 code]# cd /var/www/html/magento/
```

8. Enable the TokenOne module by entering the following command:

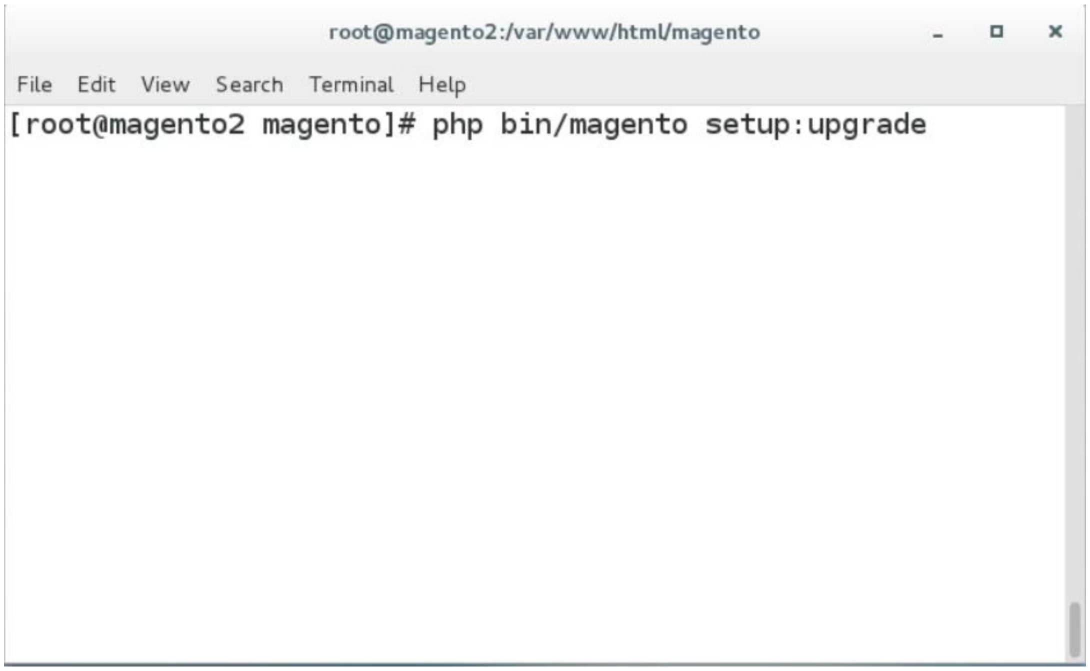
   ```
   php bin/magento module:enable Tokenone_TwoFactorAuth
   ```


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# php bin/magento module:enable Tokenone_TwoFactorAuth
```

9. To upgrade Magento to reflect the newly enabled module, enter the following command:

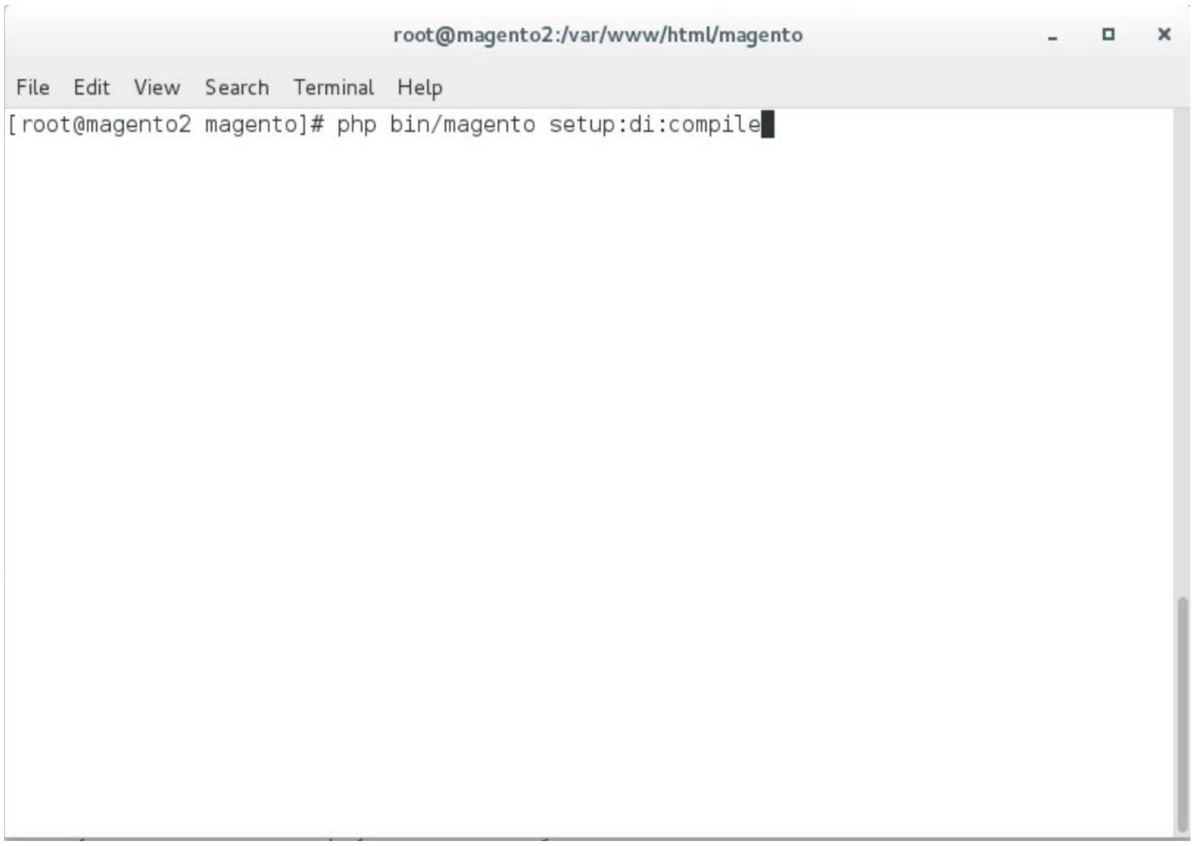
   `php bin/magento setup:upgrade`


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# php bin/magento setup:upgrade
```

10. Recompile Magento to reflect the changes by entering the following command:

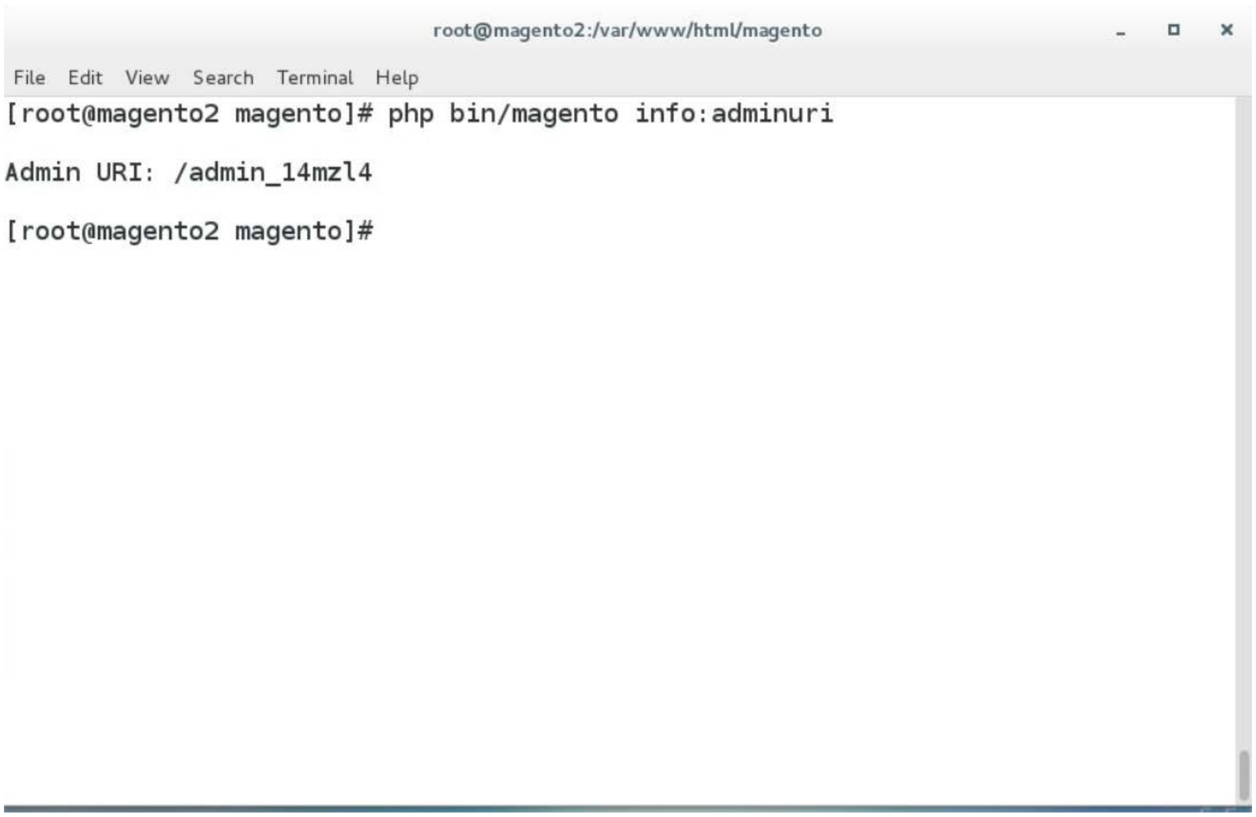
    ```
    php bin/magento setup:di:compile
    ```


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

```
[root@magento2 magento]# php bin/magento setup:di:compile
```

11. To find the Magento admin URI, enter the following command:

    ```
    php bin/magento info:adminuri
    ```


root@magento2:/var/www/html/magento
File Edit View Search Terminal Help

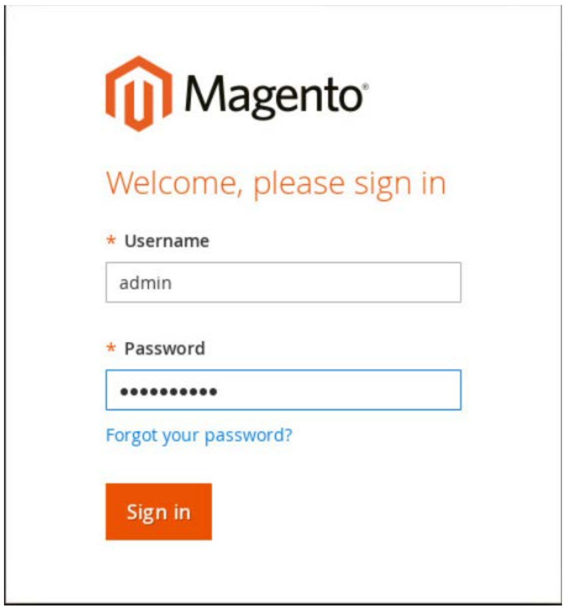
```
[root@magento2 magento]# php bin/magento info:adminuri

Admin URI: /admin_14mzl4

[root@magento2 magento]#
```

Note the URI that is output from the command. It will be used for TokenOne provisioning.

2.5.4 TokenOne Provisioning

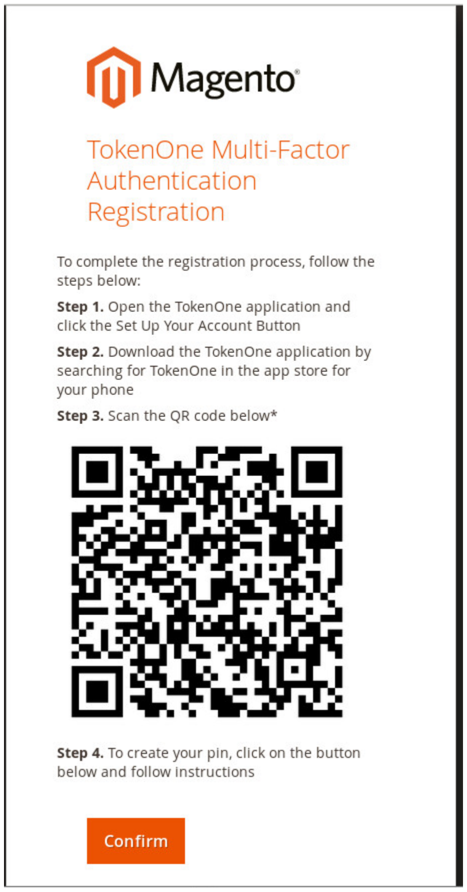
Once TokenOne has been installed, administrators will be required to use TokenOne to log in to the administration portal. The first time that an administrator logs into the portal, they will be required to provision and link their TokenOne authenticator with the system by using the following steps:

1. Open a web browser and navigate to *https://magento2.mfa.local/magento/admin_14mzl4*.
2. Sign in to the admin portal.



admin

3. Once the administrator has signed in to the Magento admin portal, a TokenOne splash screen will appear with steps to create an account.



TokenOne Multi-Factor Authentication Registration

To complete the registration process, follow the steps below:

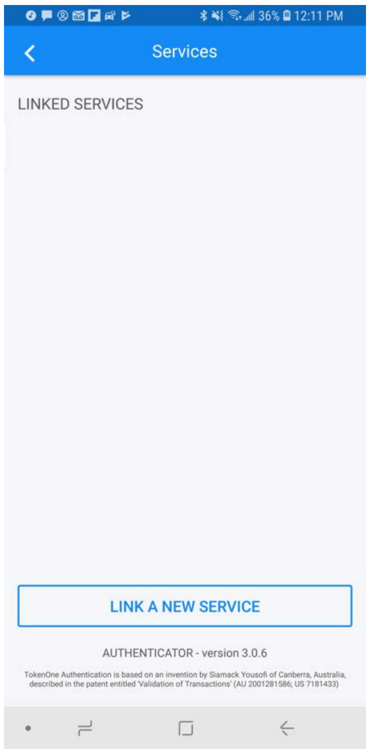
Step 1. Open the TokenOne application and click the Set Up Your Account Button

Step 2. Download the TokenOne application by searching for TokenOne in the app store for your phone

Step 3. Scan the QR code below*

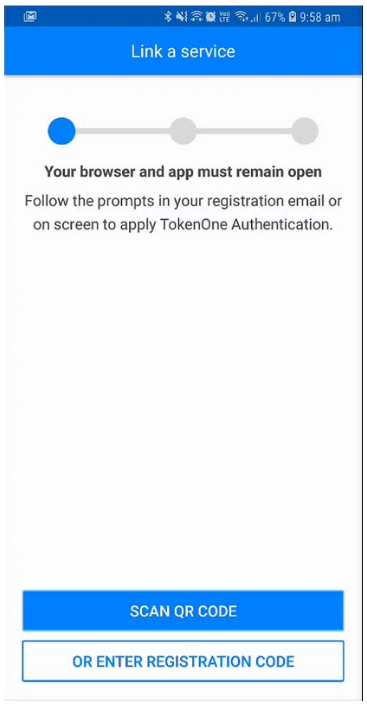
Step 4. To create your pin, click on the button below and follow instructions

4. Open the TokenOne mobile application and click **LINK A NEW SERVICE.**



TokenOne Authentication is based on an invention by Siamack Yousofi of Canberra, Australia, described in the patent entitled 'Validation of Transactions' (AU 2001281586; US 7181433)

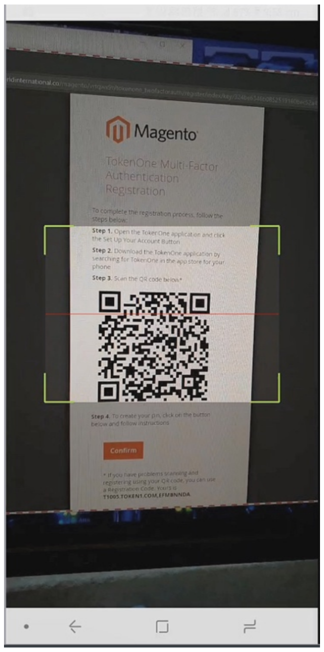
5. Click **SCAN QR CODE.**



Your browser and app must remain open

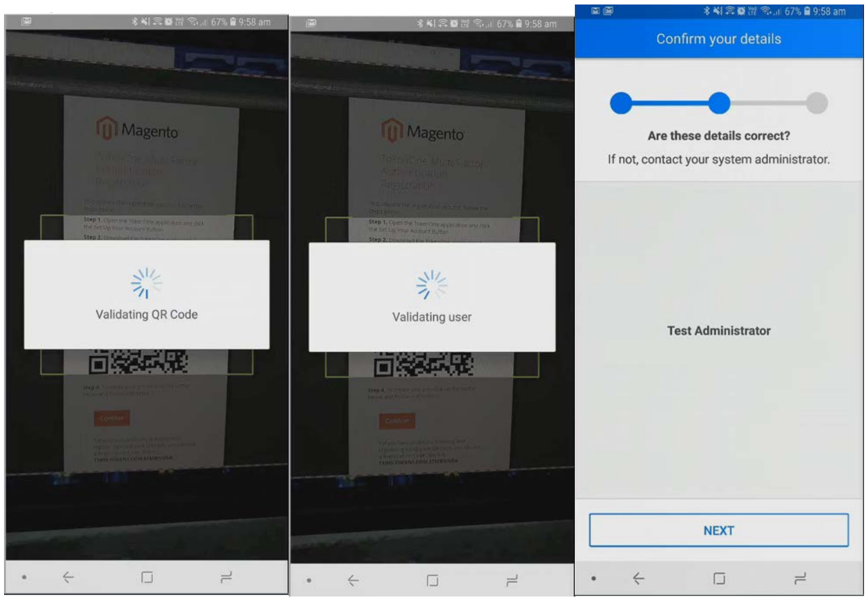
Follow the prompts in your registration email or on screen to apply TokenOne Authentication.

6. Capture the Quick Response (QR) code that is displayed on the Magento site.



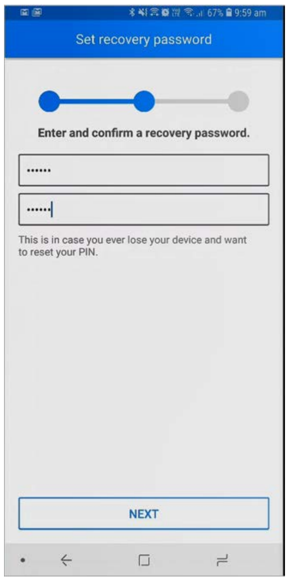
7. Upon scanning the QR code, the phone will then be profiled and registered.

8. Follow the prompts on the smartphone to complete the registration.



9. Click **NEXT.**

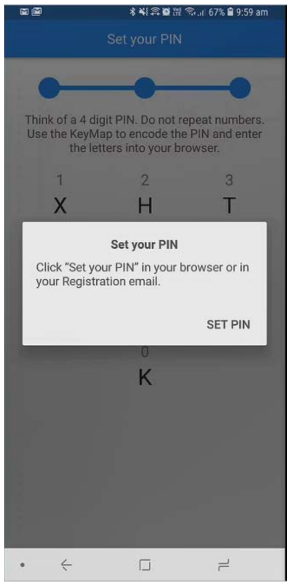
10. Create a recovery password for the account.



Enter and confirm a recovery password.

This is in case you ever lose your device and want to reset your PIN.

11. Click **NEXT.** Once the phone has been profiled and the account provisioned, you will be prompted to set your user PIN.

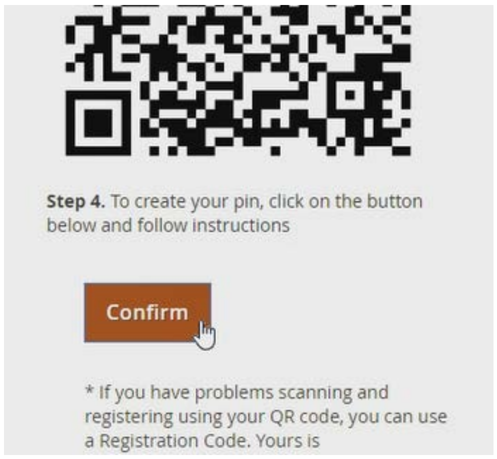


Think of a 4 digit PIN. Do not repeat numbers. Use the KeyMap to encode the PIN and enter the letters into your browser.

Set your PIN

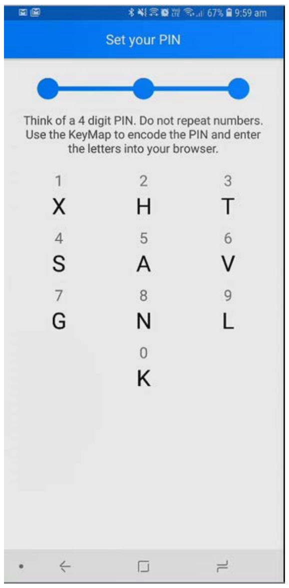
Click "Set your PIN" in your browser or in your Registration email.

12. Click **SET PIN** on the phone, and click **Confirm** on your computer.


Step 4. To create your pin, click on the button below and follow instructions

* If you have problems scanning and registering using your QR code, you can use a Registration Code. Yours is

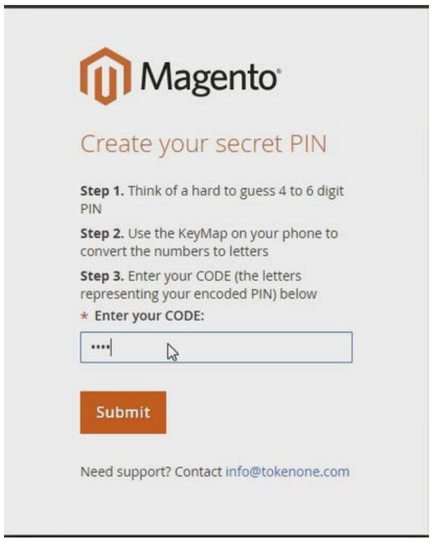
13. Use the KeyMap on the phone screen to encode your user PIN into a letter code. A KeyMap is simply a sheet of 10 letters, each with a corresponding number (0 to 9). Match the numbers of your PIN to the corresponding letters. This is your onetime letter code. For example, if your PIN is 2610, then your onetime letter code is HVXK.



Think of a 4 digit PIN. Do not repeat numbers. Use the KeyMap to encode the PIN and enter the letters into your browser.

1 X	2 H	3 T
4 S	5 A	6 V
7 G	8 N	9 L
	0 K	

14. Enter the letters corresponding to your PIN into the Magento admin panel, and click **Submit.** Repeat the process to confirm your PIN.


Magento

Create your secret PIN

Step 1. Think of a hard to guess 4 to 6 digit PIN

Step 2. Use the KeyMap on your phone to convert the numbers to letters

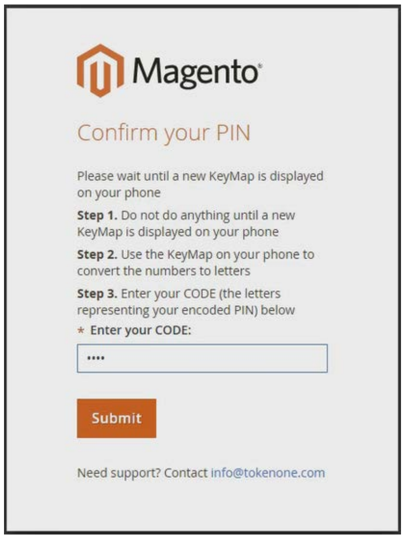
Step 3. Enter your CODE (the letters representing your encoded PIN) below

* **Enter your CODE:**

••••

Submit

Need support? Contact info@tokenone.com


Magento

Confirm your PIN

Please wait until a new KeyMap is displayed on your phone

Step 1. Do not do anything until a new KeyMap is displayed on your phone

Step 2. Use the KeyMap on your phone to convert the numbers to letters

Step 3. Enter your CODE (the letters representing your encoded PIN) below

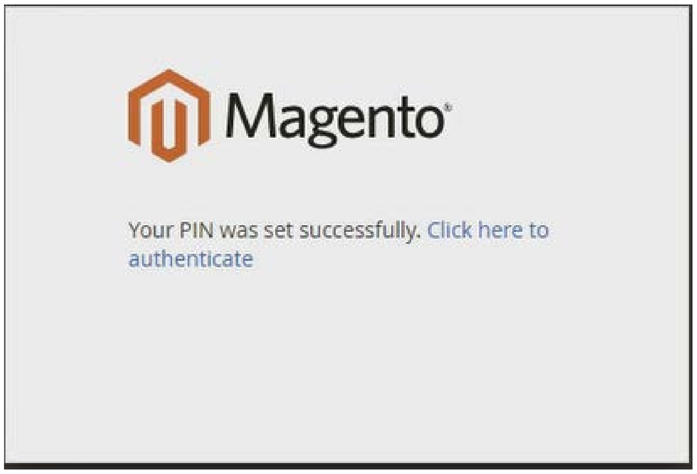
* **Enter your CODE:**

••••

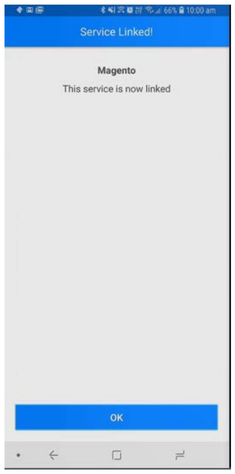
Submit

Need support? Contact info@tokenone.com

15. Do not turn off your phone during this process. Wait until the smartphone application indicates that the account has been registered.


Magento

Your PIN was set successfully. Click here to authenticate


Service Linked!

Magento

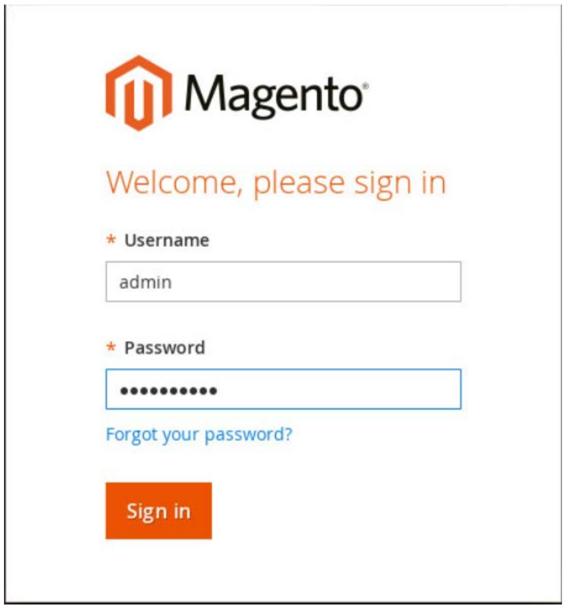
This service is now linked

OK

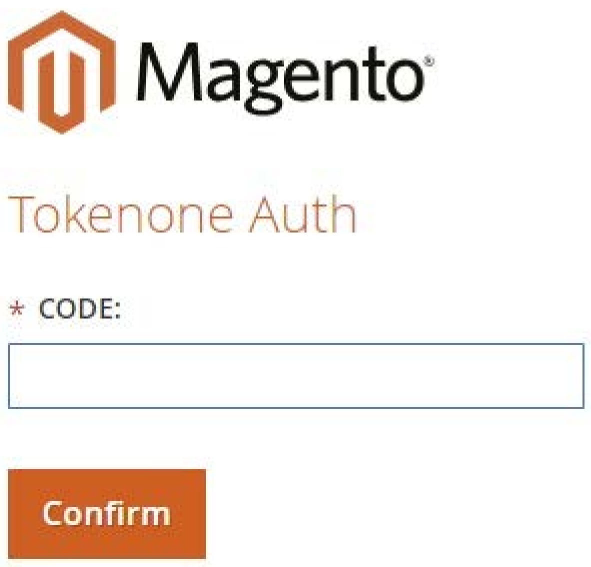
2.5.5 Administrator Login with TokenOne Authentication

To log in to the Magento administration portal by using TokenOne authentication, perform the following steps:

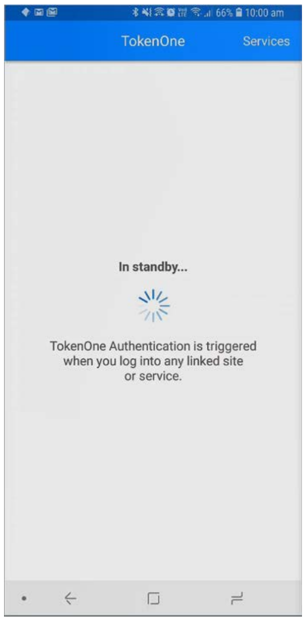
1. Open a web browser and navigate to *https://magento2.mfa.local/magento/admin_14mzl4*.
2. Sign in to the admin portal.



3. Magento will prompt for the TokenOne **CODE.**



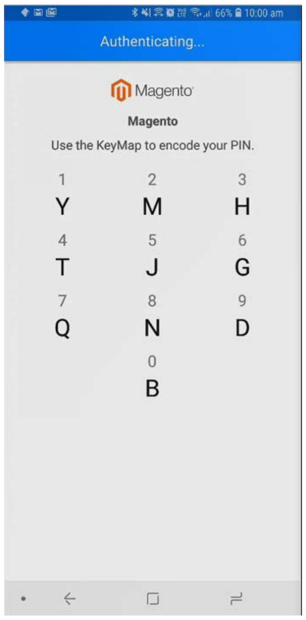
4. Open the TokenOne mobile application on your smartphone.
5. An **In standby…** screen will appear while the service verifies that you are using the correct registered device.



In standby...

TokenOne Authentication is triggered when you log into any linked site or service.

6. Once your device is verified, a unique KeyMap will appear.

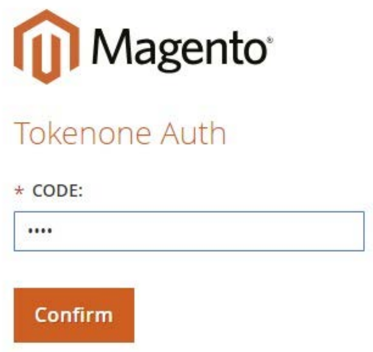


Magento

Use the KeyMap to encode your PIN.

1 Y	2 M	3 H
4 T	5 J	6 G
7 Q	8 N	9 D
	0 B	

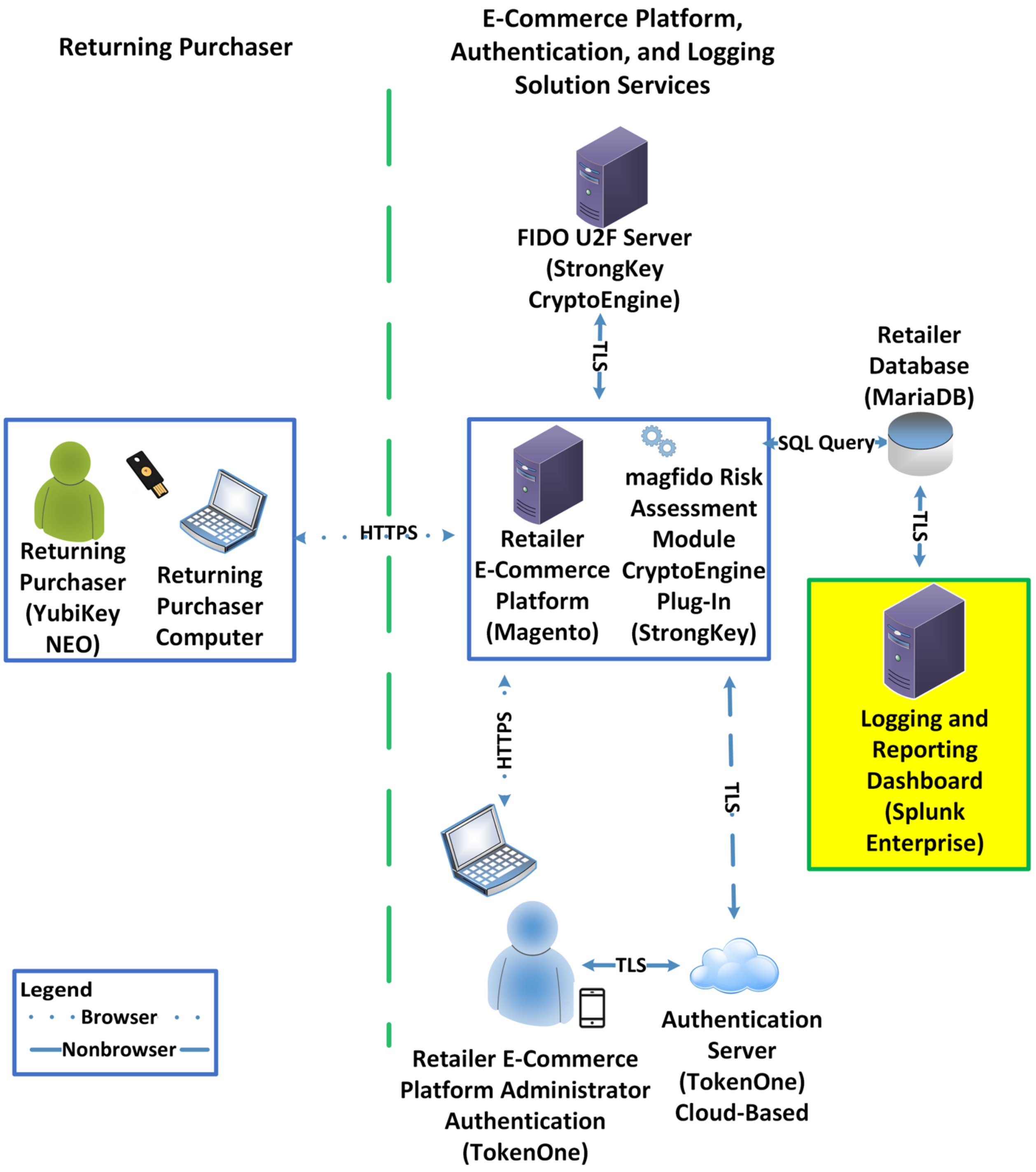
7. Match the numbers of your PIN to the corresponding letters. This is your onetime letter code. For example, if your PIN is **2610,** then your onetime letter code is **MGYB.**
8. Enter the letter code into the administration panel, and click **Confirm.**



2.6 Splunk Enterprise

This section provides installation and configuration guidance for Splunk's Enterprise product. Splunk Enterprise is used in both the *cost threshold* and *risk engine* example implementation builds to process and display authentication logging information. In addition to installing and configuring Splunk Enterprise and its supporting components, this section also provides step-by-step guidance on developing dashboard displays of the logged information. The locations of the Splunk components that are installed by using the instructions in this section are illustrated in Figure 2-6 and are highlighted in yellow within the green box.

Figure 2-6 Splunk Enterprise Components


Returning Purchaser
E-Commerce Platform, Authentication, and Logging Solution Services
FIDO U2F Server (StrongKey CryptoEngine)
TLS
Retailer Database (MariaDB)
SQL Query
TLS
Returning Purchaser (YubiKey NEO)
Returning Purchaser Computer
HTTPS
Retailer E-Commerce Platform (Magento)
magfido Risk Assessment Module CryptoEngine Plug-In (StrongKey)
Logging and Reporting Dashboard (Splunk Enterprise)
HTTPS
TLS
TLS
Legend
Browser
Nonbrowser
Retailer E-Commerce Platform Administrator Authentication (TokenOne)
Authentication Server (TokenOne) Cloud-Based

2.6.1 Splunk Technologies Overview

Splunk [10] technologies enable computer log and data collection, parsing, and display. Splunk Enterprise [11], along with two enabling capabilities, was used in both example implementations:

- Splunk Enterprise [11], where data was collected, parsed, and displayed by using dashboards
- Splunk Universal Forwarder [12], which was installed on systems from which we collected data, forwarding the information to Splunk Enterprise
- Splunk DB Connect [13], which was used to import structured data for analysis, indexing, and visualization into Splunk Enterprise in the example implementation

2.6.2 Splunk Enterprise

2.6.2.1 Overview

Splunk Enterprise [11] enables the monitoring and analyzing of data from multiple sources. Splunk Enterprise can receive data from many sources and then respond to data queries and provide dashboard displays of the data that has been provided to it.

For both example implementations, we used Splunk Enterprise to ingest a variety of log types from the retail e-commerce platform server. Once the data was collected by Splunk Enterprise, it could then be parsed and displayed by using prebuilt rules or custom criteria. For both example implementations, we displayed information as described in Section 2.6.5.

2.6.2.2 Splunk Enterprise: Requirements

System requirements required to support the use of Splunk Enterprise can be found here: http://docs.splunk.com/Documentation/Splunk/6.6.1/Installation/Systemrequirements.

2.6.2.3 Splunk Enterprise: Prepare for Installation

To prepare your environment for an on-premises installation, follow this guidance:

Windows:
http://docs.splunk.com/Documentation/Splunk/6.6.1/Installation/PrepareyourWindowsnetworkforaSplunkinstallation

2.6.2.4 Splunk Enterprise: Installation

You will need a Splunk account to download Splunk Enterprise. The account is free and can be set up at https://www.splunk.com/page/sign_up.

Download Splunk Enterprise from https://www.splunk.com/en_us/download/splunk-enterprise.html. Splunk Enterprise was installed on a Windows instance. The installation instructions can be found here: http://docs.splunk.com/Documentation/Splunk/6.6.1/Installation/InstallonWindows.

2.6.3 Splunk Universal Forwarder

2.6.3.1 Splunk Universal Forwarder: Overview

The Splunk Universal Forwarder collects data to be used by Splunk Enterprise. Splunk Universal Forwarder allows Splunk Enterprise to collect data from remote sources and send it for indexing. To use this capability, Splunk Universal Forwarder must be installed on each system from which you want to collect data.

We used Splunk Universal Forwarder to collect data from Magento and forward it to Splunk Enterprise. Once the data was delivered to Splunk Enterprise, the data provided by the Splunk Universal Forwarder was used to analyze purchaser authentication trends and to populate the dashboard displays.

2.6.3.2 Splunk Universal Forwarder: Requirements

System requirements required to support the use of Splunk Universal Forwarder can be found here: http://docs.splunk.com/Documentation/Forwarder/6.6.1/Forwarder/Systemrequirements.

2.6.3.3 Splunk Universal Forwarder: Prepare for Installation

Before you can forward data to Splunk Enterprise, you must enable forwarding and receiving on Splunk Enterprise. Instructions can be found here:
http://docs.splunk.com/Documentation/Forwarder/6.6.1/Forwarder/EnableaReceiver.

2.6.3.4 Splunk Universal Forwarder: Installation

The Splunk Universal Forwarder can be installed on different operating system platforms. The following subsections provide instructions for installing the Splunk Universal Forwarder on both Linux and Windows.

2.6.3.4.1 Installing Splunk Universal Forwarder on Linux

Detailed Splunk Universal Forwarder installation instructions can be found here:
http://docs.splunk.com/Documentation/Forwarder/6.6.1/Forwarder/Installanixuniversalforwarder#Install_the_universal_forwarder_on_Linux.

The following steps are an abridged version of the preceding installation link:

1. You will need a splunk.com account to download the Splunk Universal Forwarder on Linux. Account setup is free and can be done here: https://www.splunk.com/page/sign_up.

2. Once you have an account, the Splunk Universal Forwarder for Linux is free and can be downloaded from here: http://www.splunk.com/en_us/download/universal-forwarder.html.

3. Having the latest operating system version is recommended for installations. For both example implementations, we used the latest CentOS version 2.6+ kernel Linux distributions (64-bit). For the example implementation, we installed on CentOS by selecting the file that ends in .tgz and placed it on the target Linux machine. This is an example:

   splunkforwader-7.0.1-2b5b15c4ee89-linux-x86_64.tgz

4. Untar the file downloaded to the opt/ directory:

   ```
   tar zxvf  <splunk_package_name.tgz> -C /opt
   ```

5. Change to the /opt/splunkforwarder/bin directory:

   ```
   cd /opt/splunkforwarder/bin
   ```

6. Start the universal forwarder:

   ```
   ./splunk start
   ```

7. Enable boot start of the universal forwarder:

   ```
   ./splunk enable boot-start
   ```

2.6.3.4.2 Configure Splunk Forwarder on Linux

More information about adding a forwarder can be found at http://docs.splunk.com/Documentation/Forwarder/6.6.1/Forwarder/Configuretheuniversalforwarder.

1. Change to the /opt/splunkforwarder/bin directory:

   ```
   cd /opt/splunkforwarder/bin
   ```

2. Run script to configure the forwarder to connect to the Splunk Enterprise server:

   ```
   ./splunk add forward-server loghost:7777 -auth admin:change
   ```

2.6.3.4.3 Installing Splunk Universal Forwarder on Windows

1. You will need a splunk.com account to download the Splunk Universal Forwarder on Windows. An account is free and can be set up here: https://www.splunk.com/page/sign_up.

2. Once you have an account, the Splunk Universal Forwarder for Windows is free and can be downloaded from here: http://www.splunk.com/en_us/download/universal-forwarder.html.

3. You want the latest version for operating system version Windows (64-bit). Because this download will be installed on Windows, select the file that ends in .msi and follow the instructions for installation and configuration. This is an example of the file name that will be downloaded:

   ```
   spunkforwarder-7.0.0-00f5bb3fa822-x64-release.msi
   ```

2.6.4 Splunk DB Connect

Splunk DB Connect facilitates database information imports, exports, lookups, and multiple data source combinations [13], [14].

2.6.4.1 Overview

Splunk DB Connect provides a solution for integrating database information with Splunk Enterprise queries and reports. It allows for structured data-collection from databases, which can be leveraged in analysis.

Splunk DB Connect was used to import structured data from Magento's MySQL database instance. This enabled us to leverage information in the database within the Splunk Enterprise deployment.

2.6.4.2 Splunk DB Connect Requirements

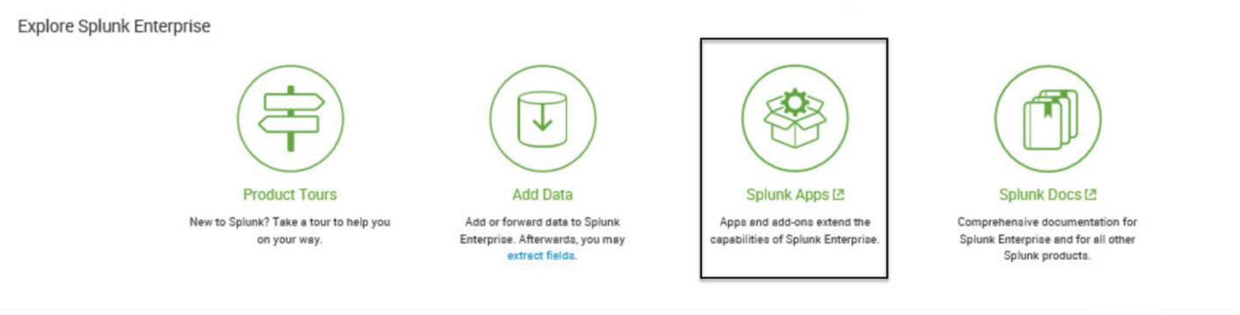
Splunk DB Connect requires that the Java Runtime Environment (JRE) is installed on the Splunk Enterprise search head. The JRE can be installed from here: http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html.

You must install a driver for the database that you are planning to connect to the Splunk DB Connect application. Splunk DB Connect supports a list of drivers that can define other databases. MariaDB is not included in the list of predefined databases. As MariaDB is a branch of MySQL, we downloaded the MySQL Java Connector from the following location (Section 2.6.4.4, Step 6 provides installation directions for the Java Connector.): https://dev.mysql.com/downloads/connector/j/.

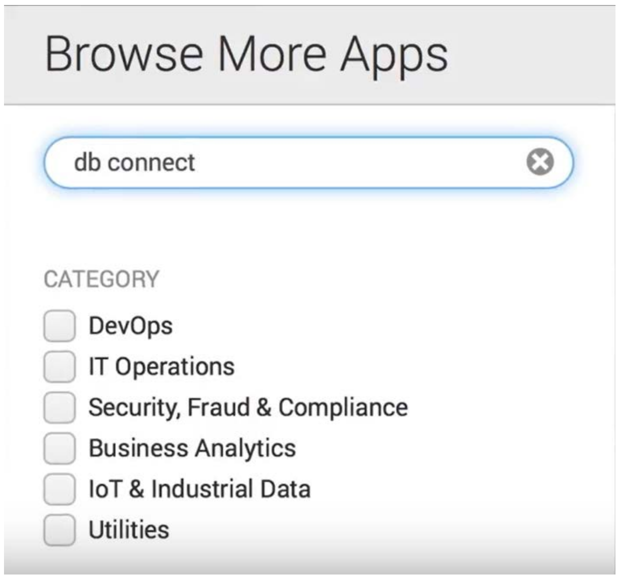
2.6.4.3 Splunk DB Connect Installation

This section describes the steps required to install the Splunk DB Connect application onto your single-instance deployment of Splunk. Additional guidance can be found here: https://docs.splunk.com/Documentation/DBX/3.1.2/DeployDBX/AboutSplunkDBConnect.

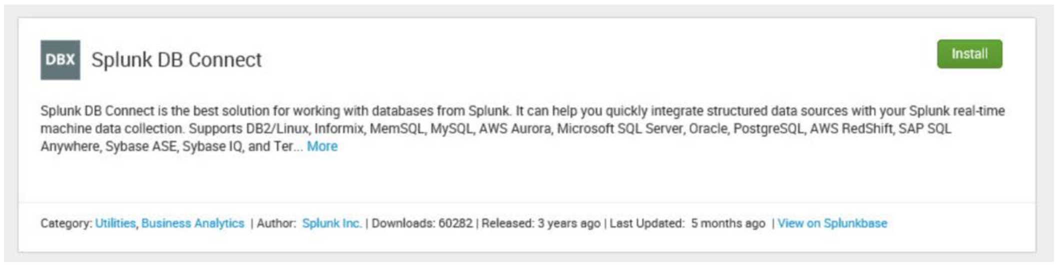
1. Navigate to the Splunk Enterprise home page, and click the **Splunk Apps** icon.


Explore Splunk Enterprise
Product Tours
New to Splunk? Take a tour to help you on your way.
Add Data
Add or forward data to Splunk Enterprise. Afterwards, you may extract fields.
Splunk Apps
Apps and add-ons extend the capabilities of Splunk Enterprise.
Splunk Docs
Comprehensive documentation for Splunk Enterprise and for all other Splunk products.

2. Type “db connect” into the search bar to locate the Splunk DB Connect application.



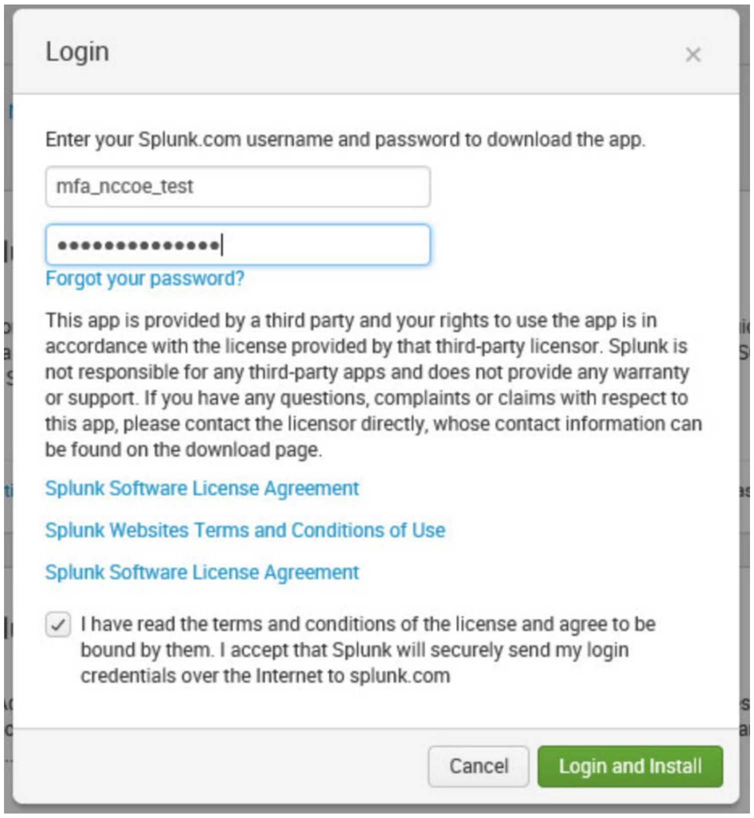
3. Once the **Splunk DB Connect** application is located, click **Install.**



Splunk DB Connect is the best solution for working with databases from Splunk. It can help you quickly integrate structured data sources with your Splunk real-time machine data collection. Supports DB2/Linux, Informix, MemSQL, MySQL, AWS Aurora, Microsoft SQL Server, Oracle, PostgreSQL, AWS RedShift, SAP SQL Anywhere, Sybase ASE, Sybase IQ, and Ter... More

Category: Utilities, Business Analytics | Author: Splunk Inc. | Downloads: 60282 | Released: 3 years ago | Last Updated: 5 months ago | View on Splunkbase

4. Log in and accept the terms and conditions by using your splunk.com user account and credentials (not the Splunk Enterprise instance credentials) and then by clicking **Login and Install.**


Login

Enter your Splunk.com username and password to download the app.

mfa_nccoe_test

••••••••••••••

Forgot your password?

This app is provided by a third party and your rights to use the app is in accordance with the license provided by that third-party licensor. Splunk is not responsible for any third-party apps and does not provide any warranty or support. If you have any questions, complaints or claims with respect to this app, please contact the licensor directly, whose contact information can be found on the download page.

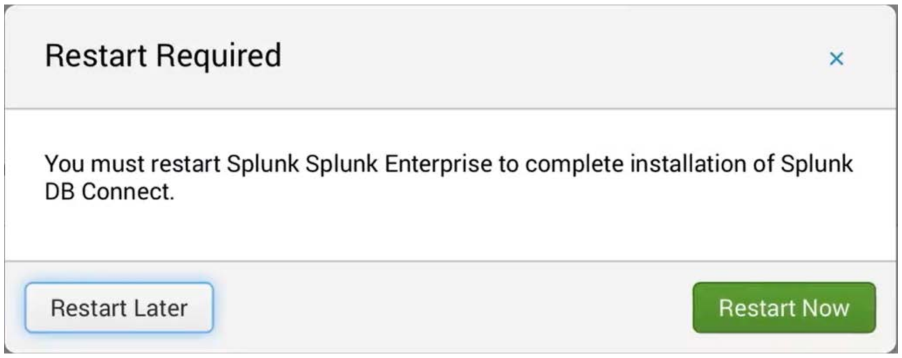
Splunk Software License Agreement

Splunk Websites Terms and Conditions of Use

Splunk Software License Agreement

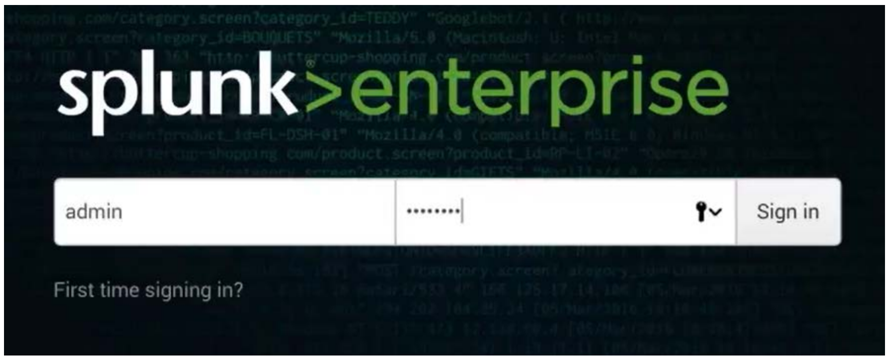
✓ I have read the terms and conditions of the license and agree to be bound by them. I accept that Splunk will securely send my login credentials over the Internet to splunk.com

5. Click **Restart Now.**


Restart Required

You must restart Splunk Splunk Enterprise to complete installation of Splunk DB Connect.

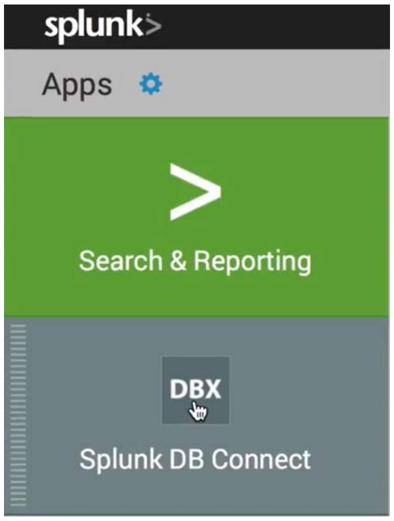
6. Log in after reboot with the Splunk Enterprise instance credentials that were created during installation of Splunk Enterprise.



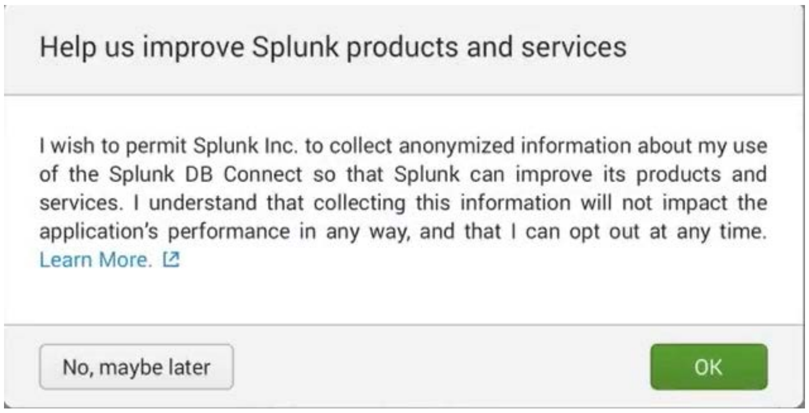
2.6.4.4 Setup

This section describes the initial setup process that will follow installation of Splunk DB Connect.

1. On the home page, navigate to **Splunk DB Connect** in the **Apps** sidebar.



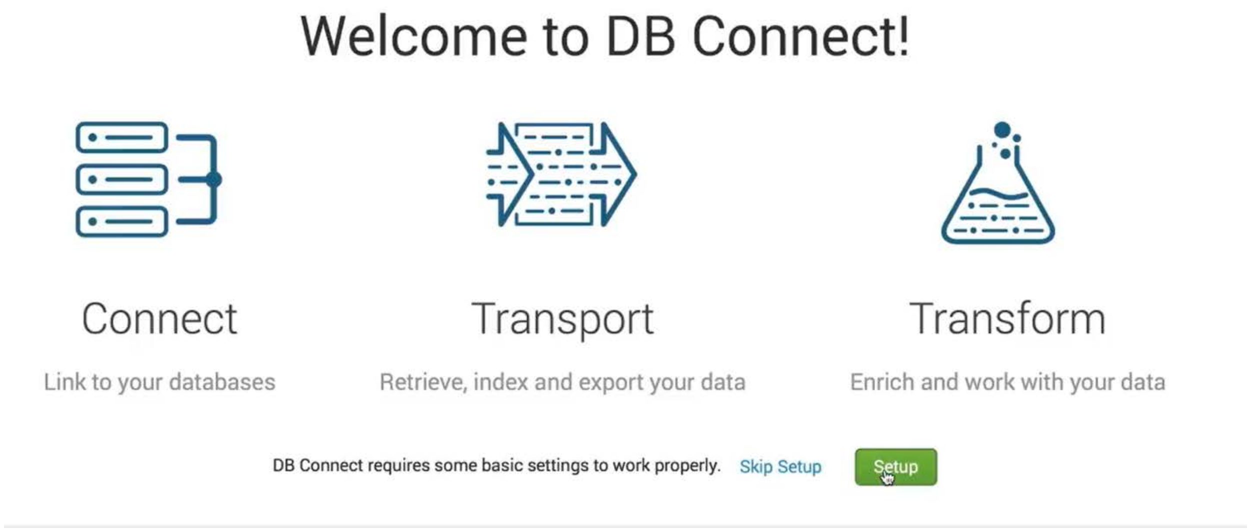
2. Select whether to send Splunk information about your use of Splunk DB Connect.



Help us improve Splunk products and services

I wish to permit Splunk Inc. to collect anonymized information about my use of the Splunk DB Connect so that Splunk can improve its products and services. I understand that collecting this information will not impact the application's performance in any way, and that I can opt out at any time. Learn More.

3. Click **Setup** to begin the configuration process.



Welcome to DB Connect!

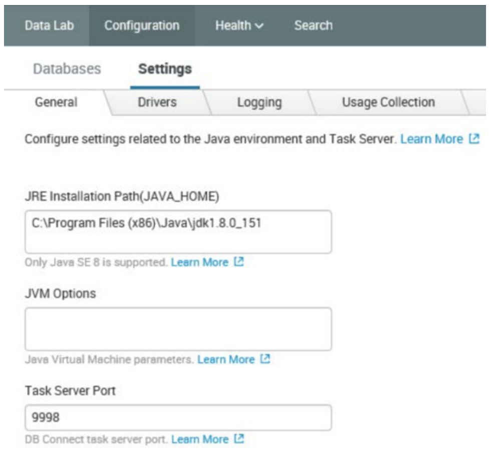
Connect
Link to your databases

Transport
Retrieve, index and export your data

Transform
Enrich and work with your data

DB Connect requires some basic settings to work properly.

4. Specify the **JRE Installation Path (JAVA_HOME).**



Configure settings related to the Java environment and Task Server. Learn More

JRE Installation Path(JAVA_HOME)

C:\Program Files (x86)\Java\jdk1.8.0_151

Only Java SE 8 is supported. Learn More

JVM Options

Java Virtual Machine parameters. Learn More

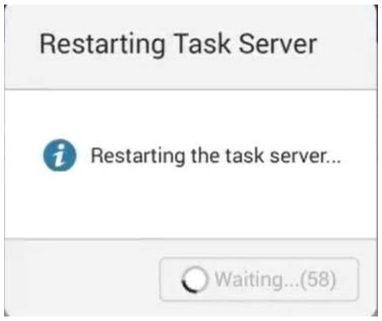
Task Server Port

9998

DB Connect task server port. Learn More

a. Click **Save** to confirm general configurations.

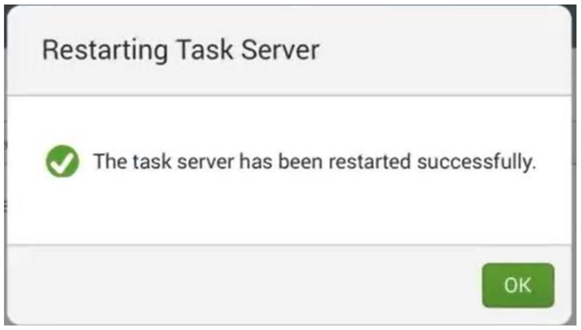
b. Task server restart will occur.



Restarting Task Server

Restarting the task server...

c. Once the restart completes, click **OK.**

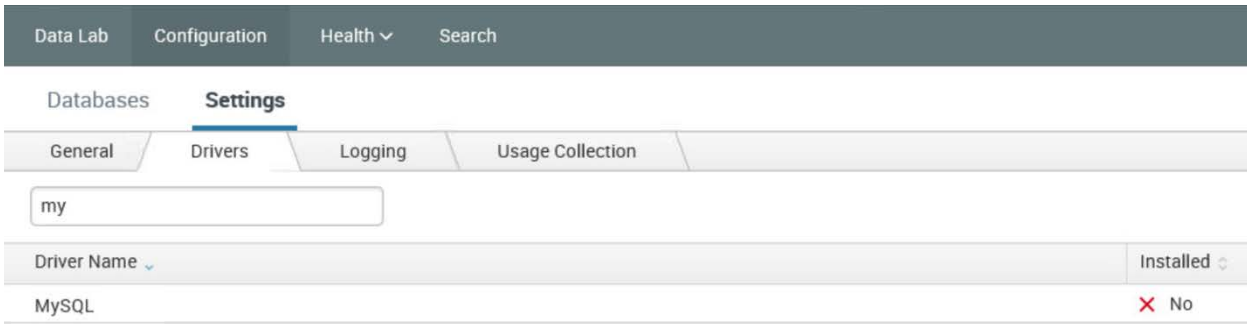


Restarting Task Server

The task server has been restarted successfully.

5. Proceed to set up drivers for the database in the **Drivers** tab: **Configuration > Settings > Drivers.**

6. Search for the database that you are using.



Driver Name	Installed
MySQL	✗ No

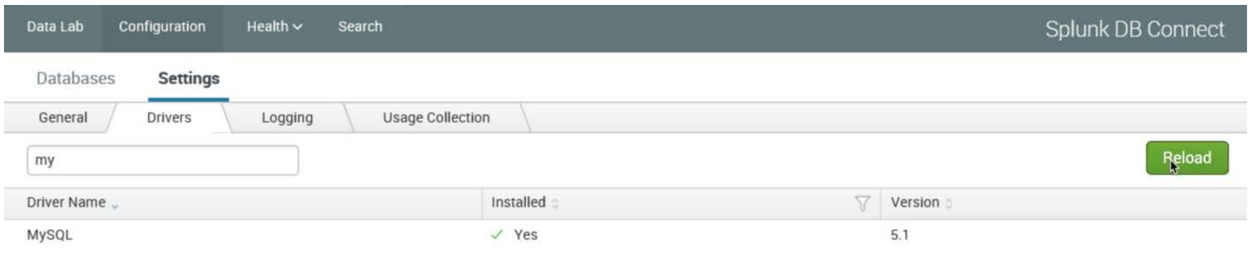
a. If your driver is not installed, Splunk DB Connect will show **No** for **Installed.** If that is the case, perform Step i below to move the connector into a new directory to enable configuring Splunk DB Connect.

i. Move the MySQL Java Connector downloaded in Section 2.6.4.2 to the following directory:

C:\Program Files\Splunk\etc\apps\splunk_app_db_connect\drivers

b. To specify a database that is not predefined, follow the Splunk documentation located here: https://docs.splunk.com/Documentation/DBX/3.1.2/DeployDBX/AboutSplunkDBConnect.

7. Click **Reload.** The status of the driver should reflect that it was installed.



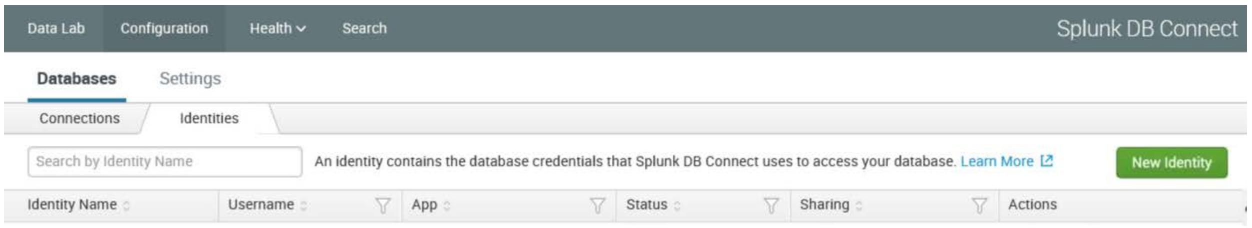
Driver Name	Installed	Version
MySQL	✓ Yes	5.1

2.6.4.5 Creating Identities

Before connecting Splunk DB Connect to your database, an identity is needed to establish the connection. This section details creating an identity that leverages database credentials, which will be used by Splunk DB Connect to access your database.

1. Navigate to the **Identities** tab: **Configuration > Databases > Identities.**

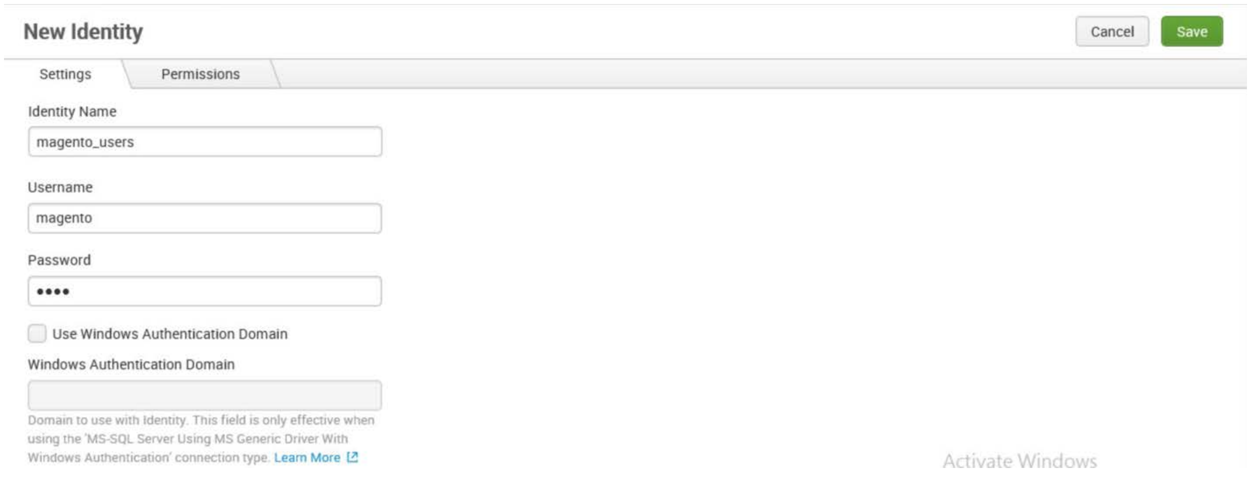
2. Click **New Identity.**



An identity contains the database credentials that Splunk DB Connect uses to access your database. Learn More

Identity Name	Username	App	Status	Sharing	Actions

3. Configure the **Settings** for your **New Identity.**



Identity Name: magento_users

Username: magento

Password: ••••

☐ Use Windows Authentication Domain

Windows Authentication Domain:

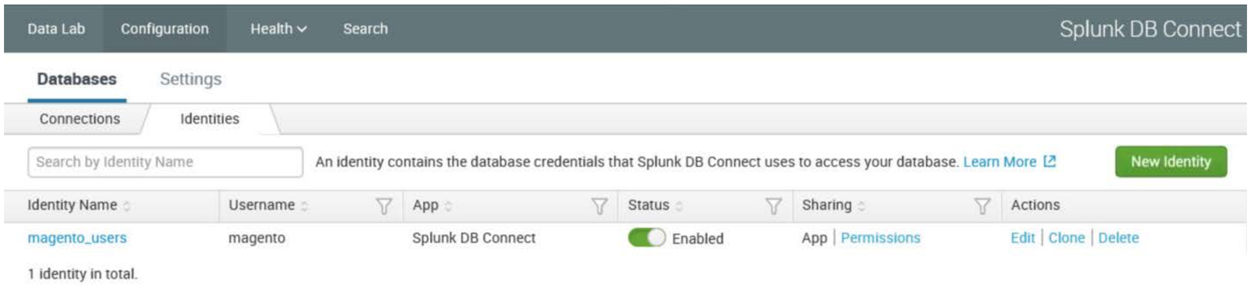
Domain to use with Identity. This field is only effective when using the 'MS-SQL Server Using MS Generic Driver With Windows Authentication' connection type. Learn More

a. Specify a unique **Identity Name.**

b. Enter the **Username** and **Password** that are used to access your database.

c. Click **Save.**

4. You will now see the new identity that you created, listed in the table of identities.



An identity contains the database credentials that Splunk DB Connect uses to access your database. Learn More

Identity Name	Username	App	Status	Sharing	Actions
magento_users	magento	Splunk DB Connect	Enabled	App | Permissions	Edit | Clone | Delete

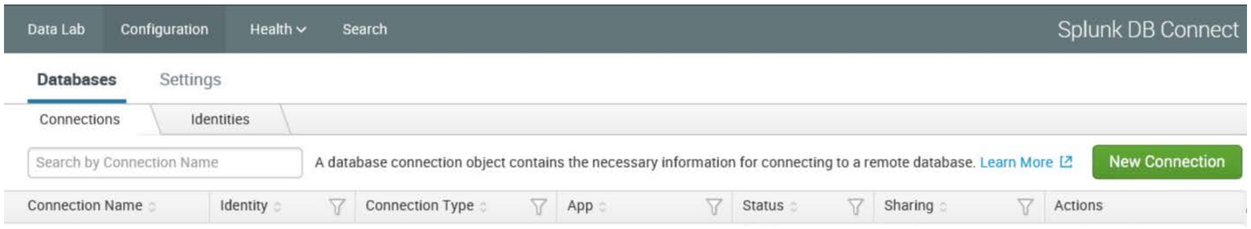
1 identity in total.

2.6.4.6 Creating Connections

This section details how to create a database connection for Splunk DB Connect to use. This provides the information that the software needs to connect to your remote database.

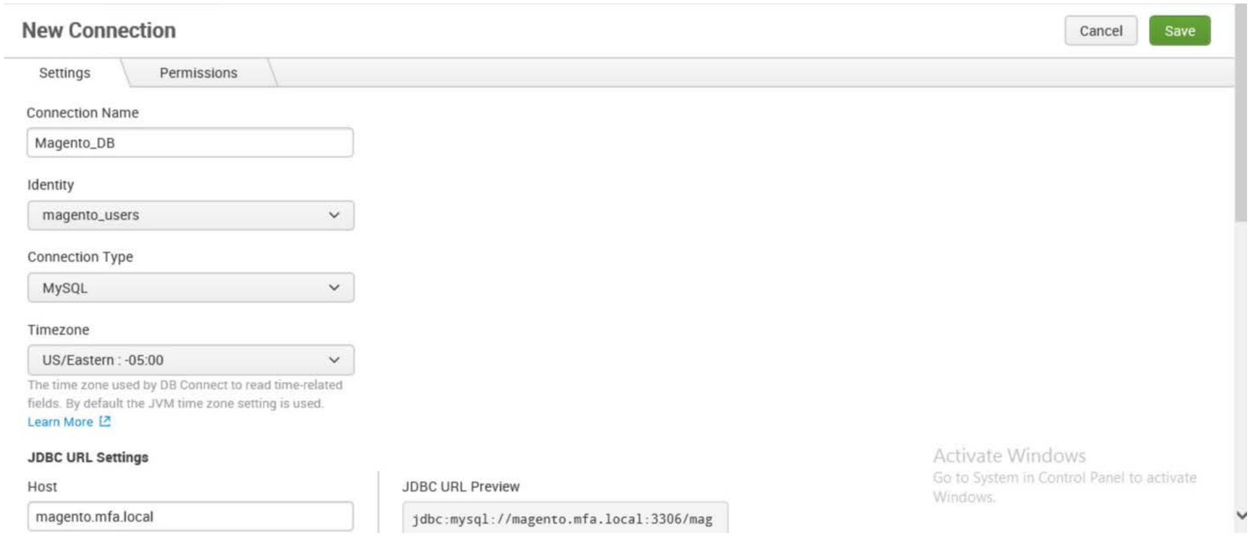
1. Navigate to the **Connections** tab: **Configuration > Databases > Connections.**

2. Click **New Connection.**



A database connection object contains the necessary information for connecting to a remote database. Learn More

3. Configure the **Settings** for your **New Connection.**



Connection Name: Magento_DB

Identity: magento_users

Connection Type: MySQL

Timezone: US/Eastern : -05:00

The time zone used by DB Connect to read time-related fields. By default the JVM time zone setting is used. Learn More

Host: magento.mfa.local

JDBC URL Preview

```
jdbc:mysql://magento.mfa.local:3306/mag
```

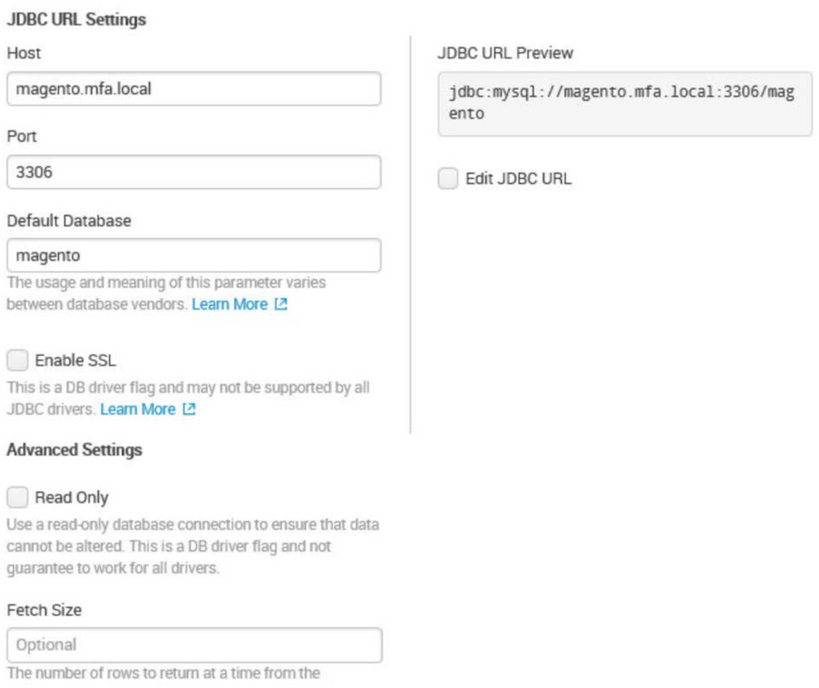
   a. Uniquely name your connection in the **Connection Name** field.

   b. Select the **Identity** created in Section 2.6.4.5.

   c. In the **Connection Type** field, select the type of database being connected.

   d. Specify the **Timezone.**

4. Configure the **JDBC URL Settings.**


JDBC URL Settings

Host: magento.mfa.local

Port: 3306

Default Database: magento

The usage and meaning of this parameter varies between database vendors. Learn More

☐ Enable SSL

This is a DB driver flag and may not be supported by all JDBC drivers. Learn More

Advanced Settings

☐ Read Only

Use a read-only database connection to ensure that data cannot be altered. This is a DB driver flag and not guarantee to work for all drivers.

Fetch Size: Optional

The number of rows to return at a time from the

JDBC URL Preview

```
jdbc:mysql://magento.mfa.local:3306/magento
```

☐ Edit JDBC URL

   a. Enter the database's host name in the **Host** field.

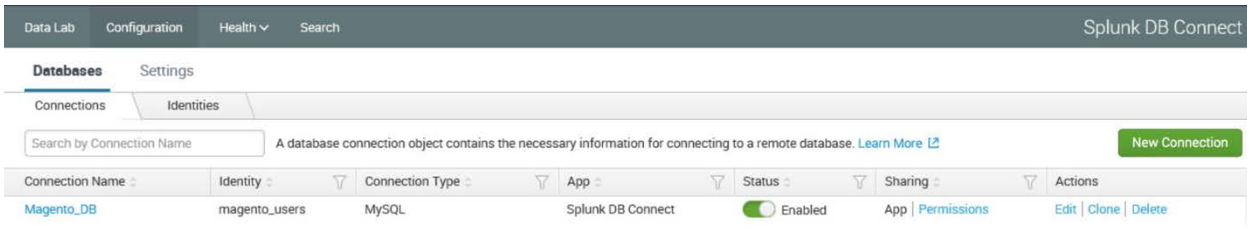
   b. Specify the **Port** that your database uses for remote connections.

   c. Specify the **Default Database** to be used.

   d. Click **Save.**

      Note: If you receive an error when attempting to save the connection, be sure to check that the database to which you are attempting to connect is configured for remote connections.

5. You will now see the new connection that you created, listed in the table of connections.



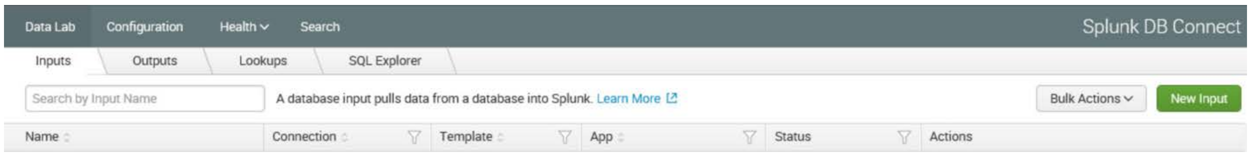
A database connection object contains the necessary information for connecting to a remote database. Learn More

Connection Name	Identity	Connection Type	App	Status	Sharing	Actions
Magento_DB	magento_users	MySQL	Splunk DB Connect	Enabled	App | Permissions	Edit | Clone | Delete

2.6.4.7 Creating Inputs

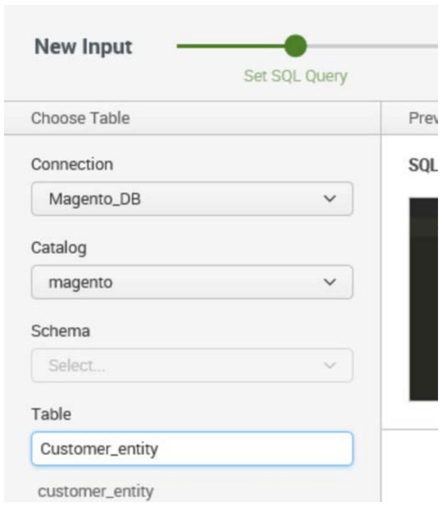
This section details how to ingest data from your database by using inputs. We demonstrated creation of an input that pulled customer account information from the Magento database.

1. Navigate to the **Inputs** tab: **Data Lab > Inputs.**
2. Click **New Input.**



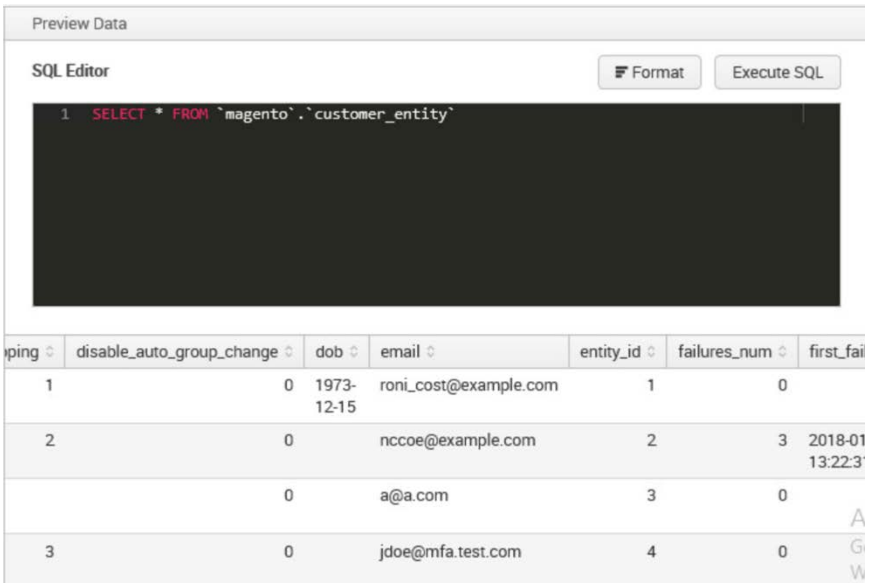
A database input pulls data from a database into Splunk. Learn More

3. Choose the table for your **New Input.**



Connection: Magento_DB
Catalog: magento
Schema: Select...
Table: Customer_entity
customer_entity

   a. Select the **Connection** created in Section 2.6.4.6.
   b. Select the Default Database created in Section 2.6.4.6, Step 4c, as the **Catalog.**
   c. Search for and select the **Table** from which input is to pull data. We selected the **Customer_entity** table.
4. Preview the data.

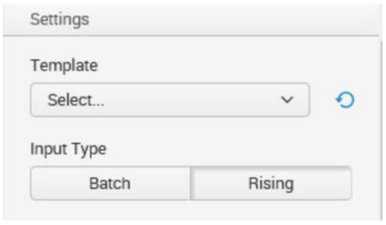


```
SELECT * FROM `magento`.`customer_entity`
```

ping	disable_auto_group_change	dob	email	entity_id	failures_num	first_fai
1	0	1973-12-15	roni_cost@example.com	1	0	
2	0		nccoe@example.com	2	3	2018-01 13:22:3
	0		a@a.com	3	0	
3	0		jdoe@mfa.test.com	4	0	

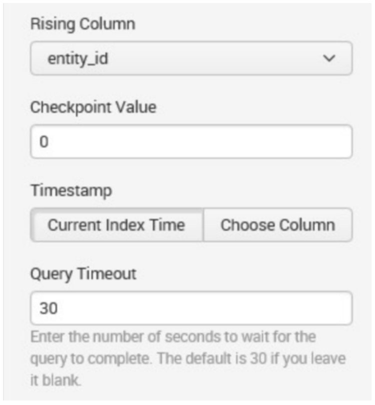
5. Click **Execute SQL** to review the results of the query.

6. Select the **Input Type.**



Batch or **Rising: Batch** indexes all of the table's data every time that it runs, whereas **Rising** uses a checkpoint to update the data that it collects from the table. We selected **Rising.**

7. Configure the settings for the Rising input type.



Rising Column: entity_id

Checkpoint Value: 0

Timestamp: Current Index Time | Choose Column

Query Timeout: 30

Enter the number of seconds to wait for the query to complete. The default is 30 if you leave it blank.

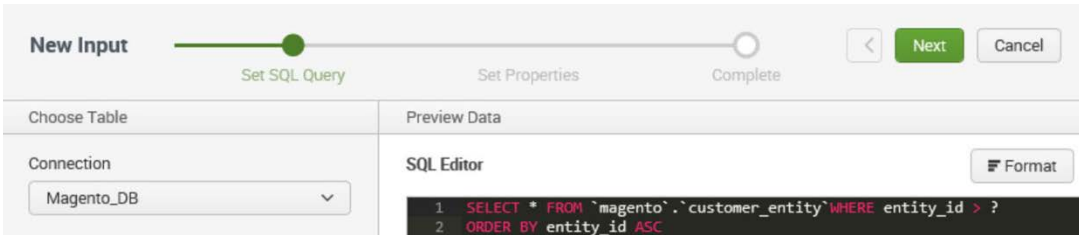
a. Specify the column of your table to be used as the **Rising Column.** We selected **entity_id.**

b. Enter the **Checkpoint Value** of the entry where you want your Rising Input to begin updating. This will dynamically update as the query is executed over time. We entered **0** to begin input at the first entity created.

c. Select the **Timestamp** for Splunk to index this data. We selected **Current Index Time.**

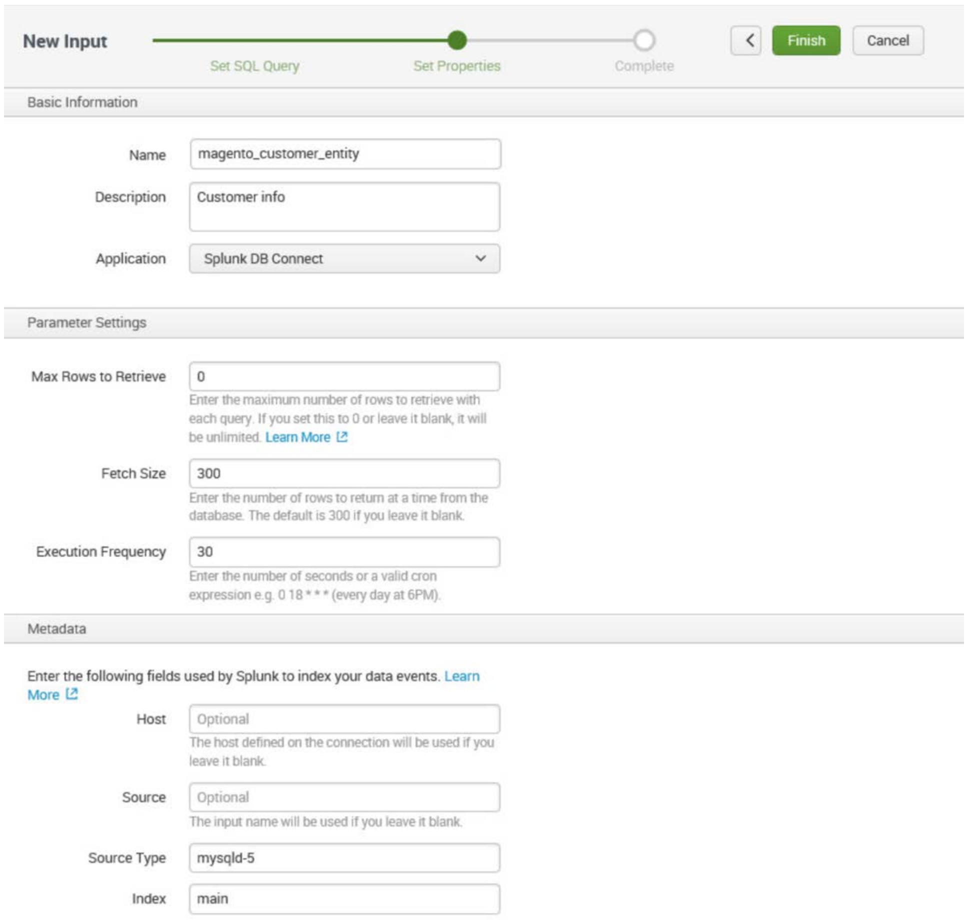
d. **Query Timeout:** Enter the number of seconds to wait for the query to complete. We entered **30.**

8. Click **Next.**



```
SELECT * FROM `magento`.`customer_entity`WHERE entity_id > ?
ORDER BY entity_id ASC
```

9. **Set Properties** for the **New Input.**



Basic Information

Name	magento_customer_entity
Description	Customer info
Application	Splunk DB Connect

Parameter Settings

Max Rows to Retrieve	0 Enter the maximum number of rows to retrieve with each query. If you set this to 0 or leave it blank, it will be unlimited. Learn More
Fetch Size	300 Enter the number of rows to return at a time from the database. The default is 300 if you leave it blank.
Execution Frequency	30 Enter the number of seconds or a valid cron expression e.g. 0 18 * * * (every day at 6PM).

Metadata

Enter the following fields used by Splunk to index your data events. Learn More

Host	Optional The host defined on the connection will be used if you leave it blank.
Source	Optional The input name will be used if you leave it blank.
Source Type	mysqld-5
Index	main

a. Enter a unique **Name** for the input. We named our instance **magento_customer_entity.**

b. Enter a **Description** for the type of data being input from the table.

c. Select the **Application** context. We selected **Splunk DB Connect.**

d. Enter the **Max Rows to Retrieve** with each query. We entered the default, **0.**

e. Enter the **Fetch Size.** This specifies the number of rows to be returned with each input query. We entered the default, **300.**

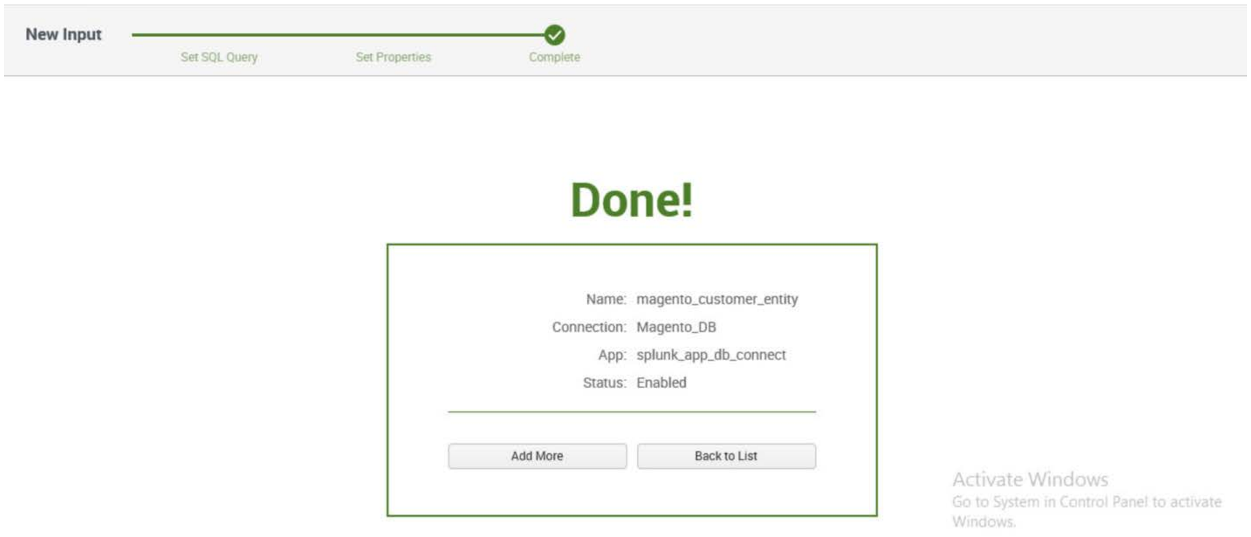
f. Enter the **Execution Frequency.** This specifies how frequently, in seconds, to execute the query for this input. We entered **30.**

g. Enter a **Source Type** for the data being queried by this input. Note: This can be predefined, or a new type can be created in this field. We entered the predefined **mysqld-5.**

h. Select the **Index** field, and enter **main.**

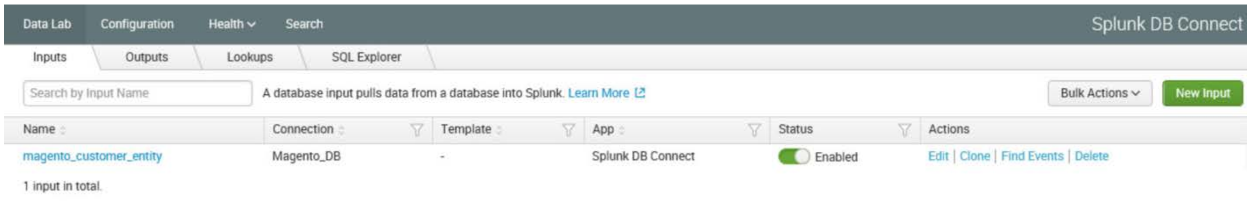
i. Click **Finish.**

10. The following screen will appear upon completion. Click **Back to List.**



Name: magento_customer_entity
Connection: Magento_DB
App: splunk_app_db_connect
Status: Enabled

11. You will now see the new input that you created, listed in the table of inputs.



A database input pulls data from a database into Splunk. Learn More

Name	Connection	Template	App	Status	Actions
magento_customer_entity	Magento_DB	-	Splunk DB Connect	Enabled	Edit | Clone | Find Events | Delete

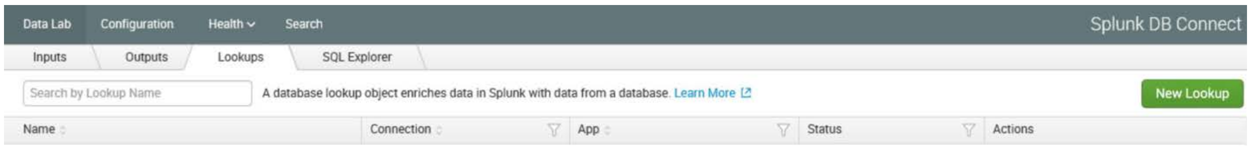
1 input in total.

2.6.4.8 *Creating Database Lookups*

This section describes creating a new database lookup. Database lookups allow you to extend the data being input from your external database into the Splunk Search Processing Language (SPL) queries. It allows events gathered from logs to be correlated with the information pulled from your database. This example correlates the entity_id returned in SPL queries to user emails stored in the database.

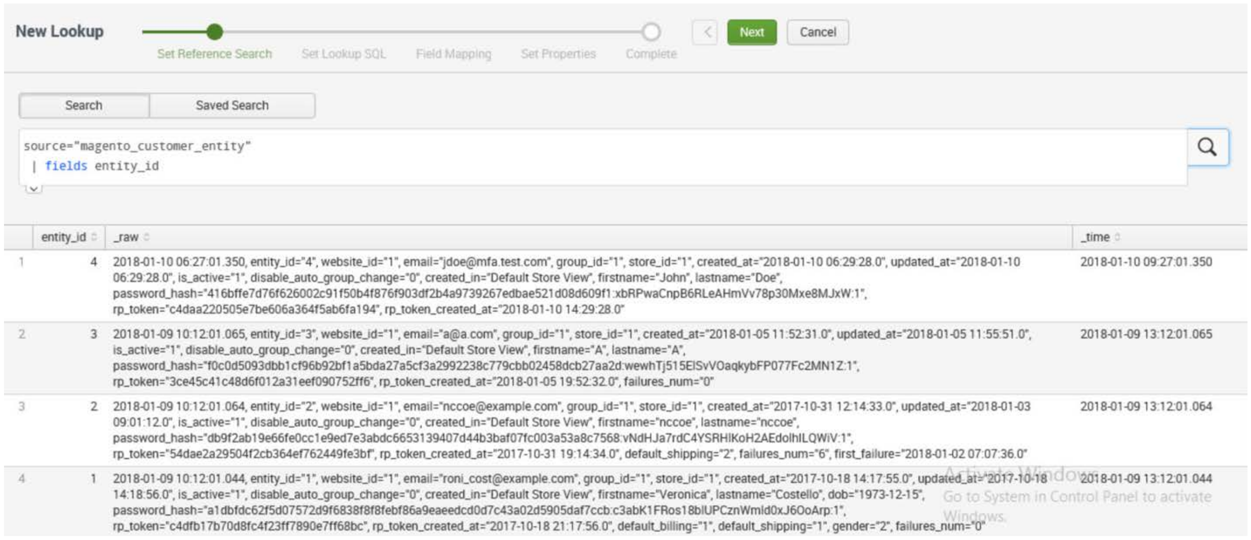
1. Navigate to the **Lookups** tab: **Data Lab > Lookups.**

2. Click **New Lookup.**



A database lookup object enriches data in Splunk with data from a database.

3. Navigate to **Set Reference Search,** and select the field of interest to be mapped to the lookup.



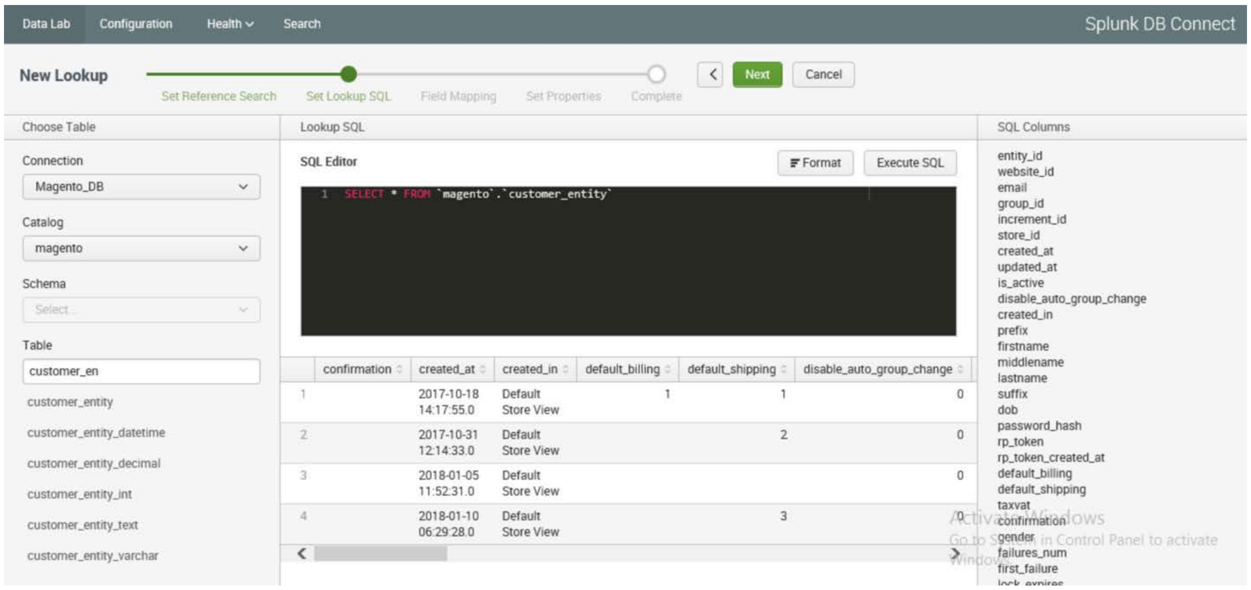
```
source="magento_customer_entity"
 | fields entity_id
```

	entity_id	_raw	_time
1	4	2018-01-10 06:27:01.350, entity_id="4", website_id="1", email="jdoe@mfa.test.com", group_id="1", store_id="1", created_at="2018-01-10 06:29:28.0", updated_at="2018-01-10 06:29:28.0", is_active="1", disable_auto_group_change="0", created_in="Default Store View", firstname="John", lastname="Doe", password_hash="416bffe7d76f626002c91f50b4f876f903df2b4a9739267edbae521d08d609f1:xbRPwaCnpB6RLeAHmVv78p30Mxe8MJxW:1", rp_token="c4daa220505e7be606a364f5ab6fa194", rp_token_created_at="2018-01-10 14:29:28.0"	2018-01-10 09:27:01.350
2	3	2018-01-09 10:12:01.065, entity_id="3", website_id="1", email="a@a.com", group_id="1", store_id="1", created_at="2018-01-05 11:52:31.0", updated_at="2018-01-05 11:55:51.0", is_active="1", disable_auto_group_change="0", created_in="Default Store View", firstname="A", lastname="A", password_hash="f0c0d5093dbb1cf96b92bf1a5bda27a5cf3a2992238c779cbb02458dcb27aa2d:wewhTj515ElSvVOaqkybFP077Fc2MN1Z:1", rp_token="3ce45c41c48d6f012a31eef090752ff6", rp_token_created_at="2018-01-05 19:52:32.0", failures_num="0"	2018-01-09 13:12:01.065
3	2	2018-01-09 10:12:01.064, entity_id="2", website_id="1", email="nccoe@example.com", group_id="1", store_id="1", created_at="2017-10-31 12:14:33.0", updated_at="2018-01-03 09:01:12.0", is_active="1", disable_auto_group_change="0", created_in="Default Store View", firstname="nccoe", lastname="nccoe", password_hash="db9f2ab19e66fe0cc1e9ed7e3abdc6653139407d44b3baf07fc003a53a8c7568:vNdHJa7rdC4YSRHIKoH2AEdolhILQWiV:1", rp_token="54dae2a29504f2cb364ef762449fe3bf", rp_token_created_at="2017-10-31 19:14:34.0", default_shipping="2", failures_num="6", first_failure="2018-01-02 07:07:36.0"	2018-01-09 13:12:01.064
4	1	2018-01-09 10:12:01.044, entity_id="1", website_id="1", email="roni_cost@example.com", group_id="1", store_id="1", created_at="2017-10-18 14:17:55.0", updated_at="2017-10-18 14:18:56.0", is_active="1", disable_auto_group_change="0", created_in="Default Store View", firstname="Veronica", lastname="Costello", dob="1973-12-15", password_hash="a1dbfdc62f5d07572d9f6838f8f8febf86a9eaeedcd0d7c43a02d5905daf7ccb:c3abK1FRos18blUPCznWmld0xJ6OoArp:1", rp_token="c4dfb17b70d8fc4f23ff7890e7ff68bc", rp_token_created_at="2017-10-18 21:17:56.0", default_billing="1", default_shipping="1", gender="2", failures_num="0"	2018-01-09 13:12:01.044

a. We entered a new **Search.**

b. Click **Next.**

4. Navigate to **Set Lookup SQL.**



Connection: Magento_DB

Catalog: magento

Schema: Select...

Table: customer_en

customer_entity
customer_entity_datetime
customer_entity_decimal
customer_entity_int
customer_entity_text
customer_entity_varchar

```
SELECT * FROM `magento`.`customer_entity`
```

	confirmation	created_at	created_in	default_billing	default_shipping	disable_auto_group_change
1		2017-10-18 14:17:55.0	Default Store View	1	1	0
2		2017-10-31 12:14:33.0	Default Store View		2	0
3		2018-01-05 11:52:31.0	Default Store View			0
4		2018-01-10 06:29:28.0	Default Store View		3	0

entity_id
website_id
email
group_id
increment_id
store_id
created_at
updated_at
is_active
disable_auto_group_change
created_in
prefix
firstname
middlename
lastname
suffix
dob
password_hash
rp_token
rp_token_created_at
default_billing
default_shipping
taxvat
confirmation
gender
failures_num
first_failure
lock_expires

a. Specify a **Connection** by using information from the connection, which was created in Section 2.6.4.6.

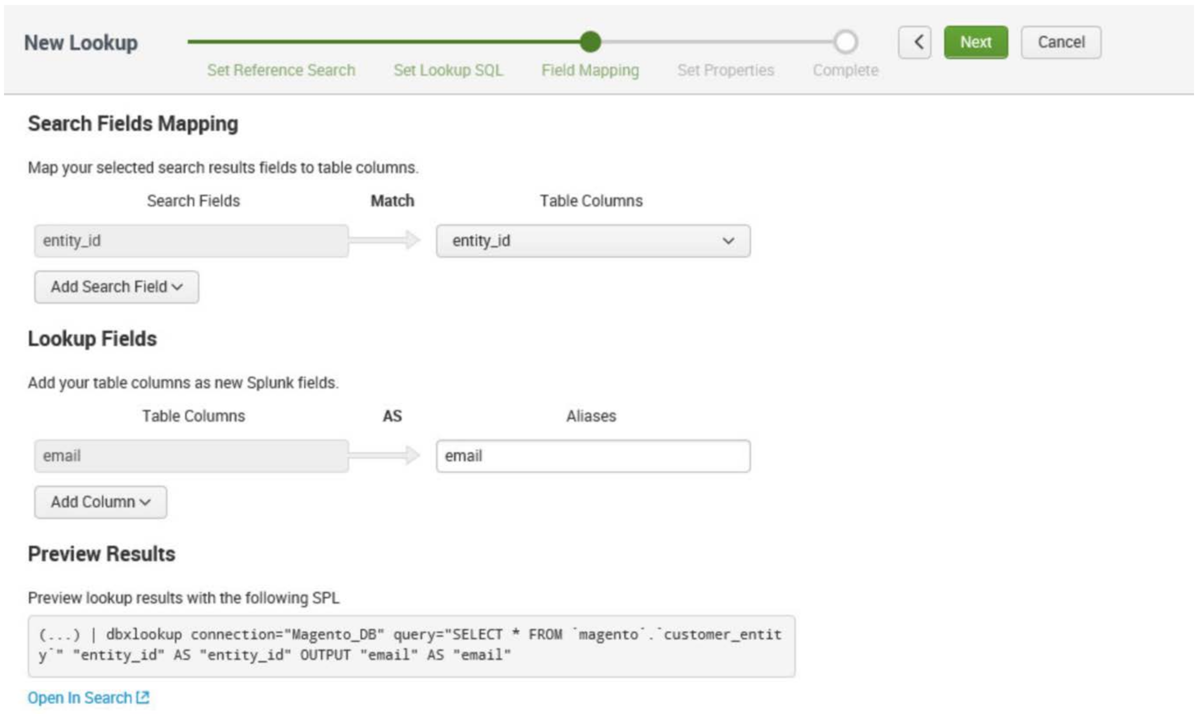
b. Specify the **Catalog.**

c. Enter the **Table.**

d. Click **Execute SQL** to view the results of the query created.

e. Click **Next.**

5. Navigate to **Field Mapping.**



Search Fields Mapping

Map your selected search results fields to table columns.

Search Fields	Match	Table Columns
entity_id	→	entity_id

Add Search Field

Lookup Fields

Add your table columns as new Splunk fields.

Table Columns	AS	Aliases
email	→	email

Add Column

Preview Results

Preview lookup results with the following SPL

```
(...) | dbxlookup connection="Magento_DB" query="SELECT * FROM `magento`.`customer_entit
y`" "entity_id" AS "entity_id" OUTPUT "email" AS "email"
```

Open In Search

a. Click **Add Search Field.**

b. Select the **Search Fields** to be mapped to the database. We selected **entity_id.**

c. Select the **Table Columns** to which the field maps in the database. We selected **entity_id.**

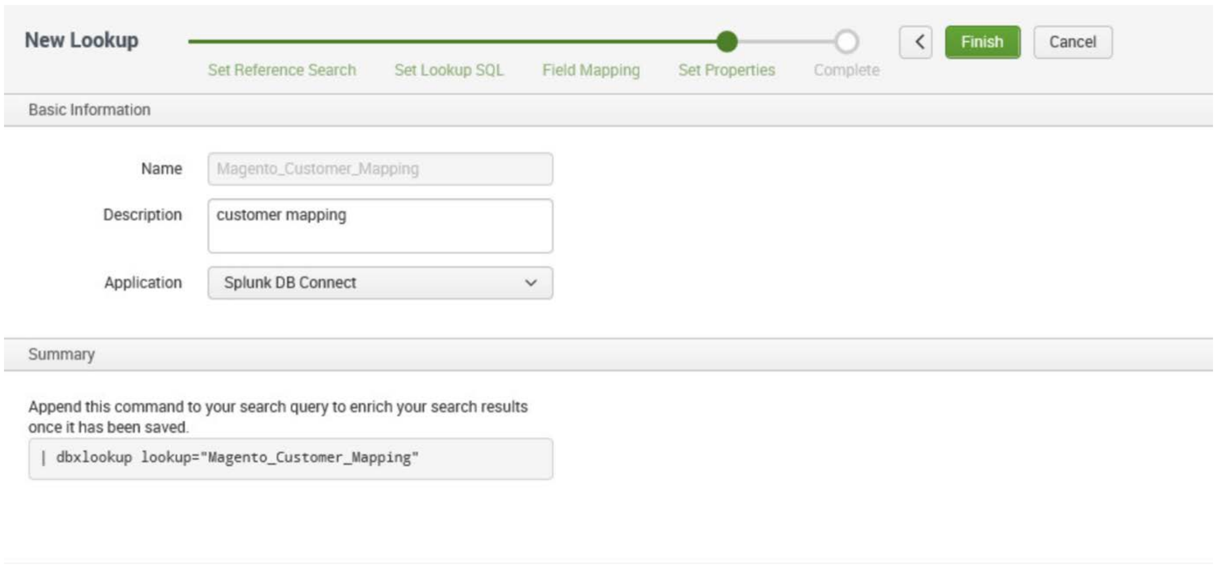
d. Click **Add Column.**

e. Select the **Table Columns** to be returned as Splunk fields. We selected **email.**

f. Enter an **Alias** for the field. We chose to leave the name of the field as **email.**

g. Click **Next.**

6. Navigate to **Set Properties.**



Name: Magento_Customer_Mapping

Description: customer mapping

Application: Splunk DB Connect

Append this command to your search query to enrich your search results once it has been saved.

```
| dbxlookup lookup="Magento_Customer_Mapping"
```

a. Enter a unique **Name** for the lookup. We named our instance **Magento_Customer_Mapping.**

b. Enter a **Description** for the type of new lookup being created.

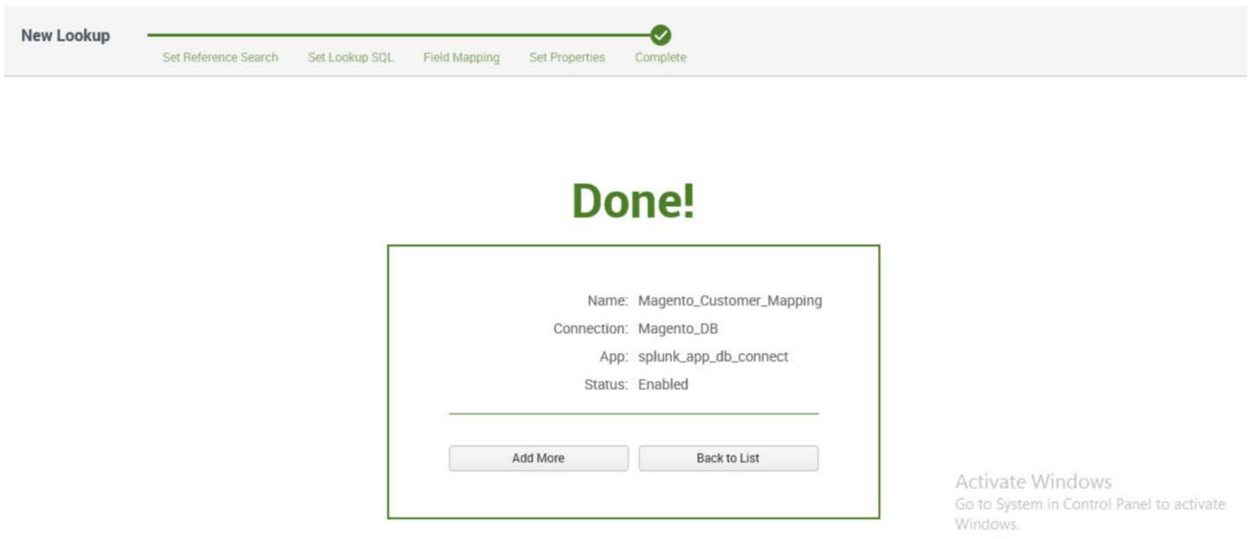
c. Select the **Application** context. We selected **Splunk DB Connect.**

d. The **Summary** contains the command to be appended to your SPL searches to leverage the lookup:

```
| dbxlookup lookup="Magento_Customer_Mapping"
```

e. Click **Finish.**

7. The following screen will appear upon completion. Click **Back to List.**


New Lookup

Set Reference Search | Set Lookup SQL | Field Mapping | Set Properties | Complete

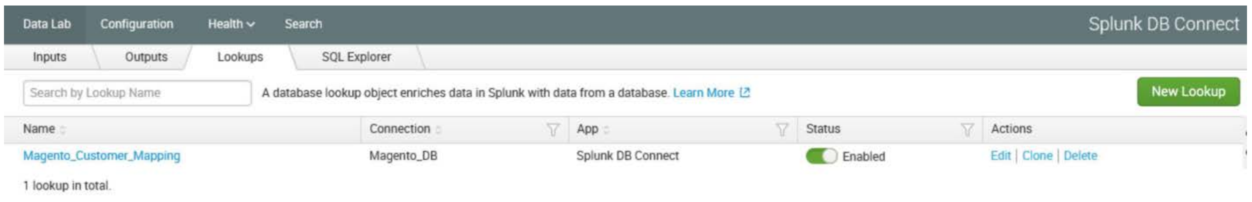
Done!

Name: Magento_Customer_Mapping
Connection: Magento_DB
App: splunk_app_db_connect
Status: Enabled

Add More | Back to List

Activate Windows
Go to System in Control Panel to activate Windows.

8. You will now see the new lookup that you created, listed in the table of lookups.



A database lookup object enriches data in Splunk with data from a database. Learn More

Name	Connection	App	Status	Actions
Magento_Customer_Mapping	Magento_DB	Splunk DB Connect	Enabled	Edit | Clone | Delete

1 lookup in total.

2.6.5 Splunk Enterprise Queries and Dashboards

Splunk Enterprise reports, alerts, and dashboards are powered by queries written in the Splunk SPL. These queries are used to perform the analytics responsible for capturing events, identifying trends, and detecting anomalies. Once a query is written, it can be saved as a report, an alert, or a dashboard panel. The following queries were developed for both example implementations and were also saved as Splunk Enterprise dashboards to provide a central viewing location.

2.6.5.1 Query: Total Attempted Single-Factor Authentications

The following search query traverses the logs aggregated from the Magento server. The query uses multiple data sources relating to the same access log to detect when access to a customer account is attempted via single-factor credentials. The output of the query shows the total events per hour.

```
host="magento.mfa.local"  source ="/var/log/httpd/*" sourcetype=access_common 302
"/fidodemo/customer/account/loginPost"  earliest=1 latest=now | stats count by
date_hour
```

2.6.5.2 Query: Failed Single-Factor Authentications Within Past Five Minutes

The following search query traverses the logs aggregated from the Magento server, specifically the database logs. This log returns information, including failed login attempts per entity ID. With the database lookup created in Section 2.6.4.8, the query below maps the entity ID to the respective email address reporting when a customer account has failed to be logged in via single-factor credentials. The output of the query shows failed logins, per email address, within a five-minute interval.

```
source="/usr/local/strongauth/mariadb-10.1.22/log/mysqld.log" failures_num!="'0'" |
rex field=entity_id "\'?(?<entity_id>[\d\.]+)\'?" | dbxlookup
lookup="Magento_Customer_Mapping" earliest=-5m latest=now | eventstats |  stats count
by email
```

2.6.5.3 Query: Attempted Single-Factor Authentications in Past Five Minutes

The following search query traverses the logs aggregated from the Magento server. The query uses multiple data sources relating to the same access log to detect when access to a customer account is attempted via single-factor credentials. The output of the query shows the failed login, per IP address, within a five-minute interval.

```
host="magento.mfa.local"  source ="/var/log/httpd/*" sourcetype=access_common 302
"/fidodemo/customer/account/loginPost"  earliest=-5m latest=now | stats count by IP
```

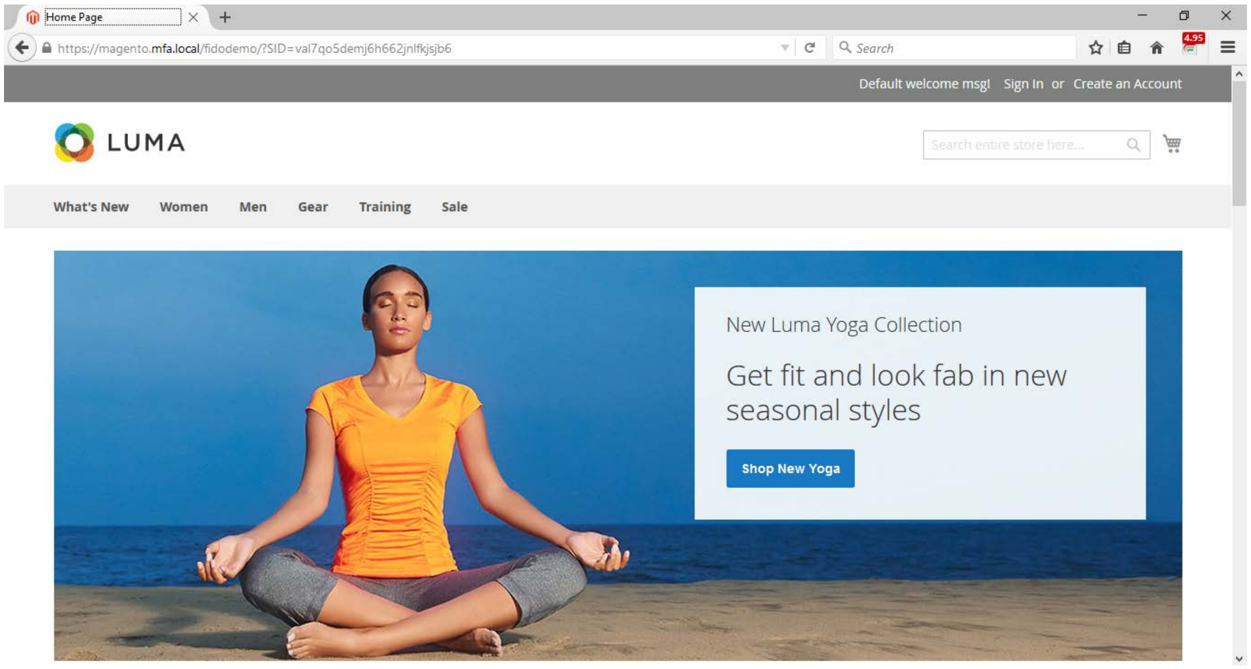
2.7 Testing FIDO Key Registration and Checkout

Once installed and configured, the example implementation can configure accounts, and the build can be tested. To test the implementation, an example customer account was created. Example processes for customer account creation, FIDO key registration, and FIDO checkout are detailed in the following subsections.

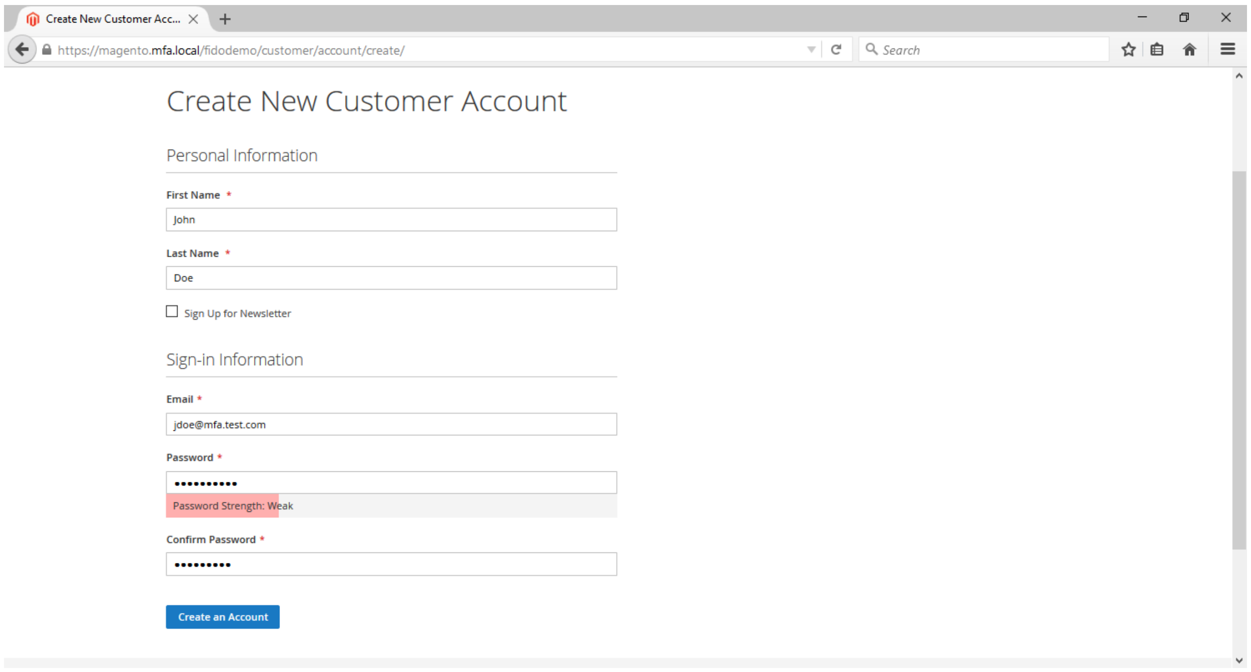
2.7.1 Creating an Example Magento Customer Account

This section outlines how to create example customer accounts. The accounts are created using a web browser interface.

1. To begin, **open a web browser** and navigate to *https://magento.mfa.local/fidodemo*.


Home Page
https://magento.mfa.local/fidodemo/?SID=val7qo5demj6h662jnlfkjsjb6
Search
Default welcome msg! Sign In or Create an Account
LUMA
Search entire store here...
What's New
Women
Men
Gear
Training
Sale
New Luma Yoga Collection
Get fit and look fab in new seasonal styles
Shop New Yoga

2. Click **Create an Account.**
3. Fill out the form as shown in the example below.
    a. **First Name:** John
    b. **Last Name:** Doe
    c. **Email:** jdoe@mfa.test.com
    d. **Password:** <password>



Create New Customer Account

https://magento.mfa.local/fidodemo/customer/account/create/

Create New Customer Account

Personal Information

First Name *

John

Last Name *

Doe

Sign Up for Newsletter

Sign-in Information

Email *

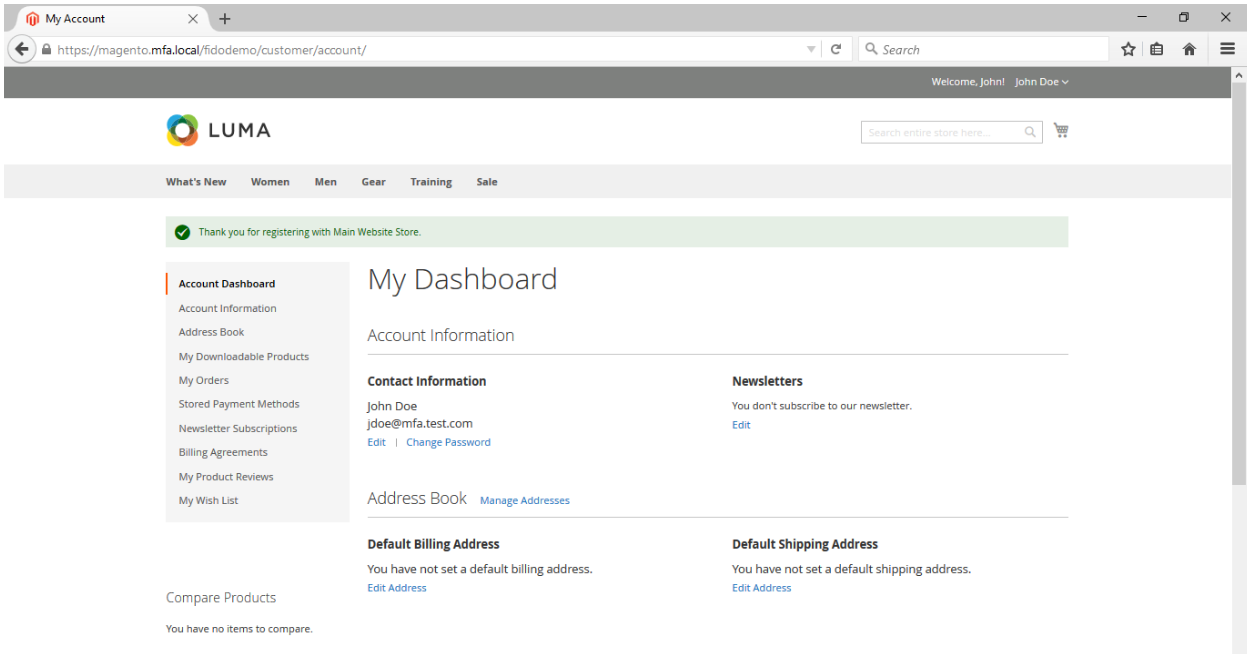
jdoe@mfa.test.com

Password *

Password Strength: Weak

Confirm Password *

4. After entering the required information, click **Create an Account.**
5. Upon successful account creation, you will be taken to the **Account Dashboard** page, where details of the account that was created are visible.



My Account

https://magento.mfa.local/fidodemo/customer/account/

Welcome, John! John Doe

Thank you for registering with Main Website Store.

My Dashboard

Account Information

Contact Information

John Doe
jdoe@mfa.test.com
Edit | Change Password

Newsletters

You don't subscribe to our newsletter.
Edit

Address Book Manage Addresses

Default Billing Address

You have not set a default billing address.
Edit Address

Default Shipping Address

You have not set a default shipping address.
Edit Address

Compare Products

You have no items to compare.

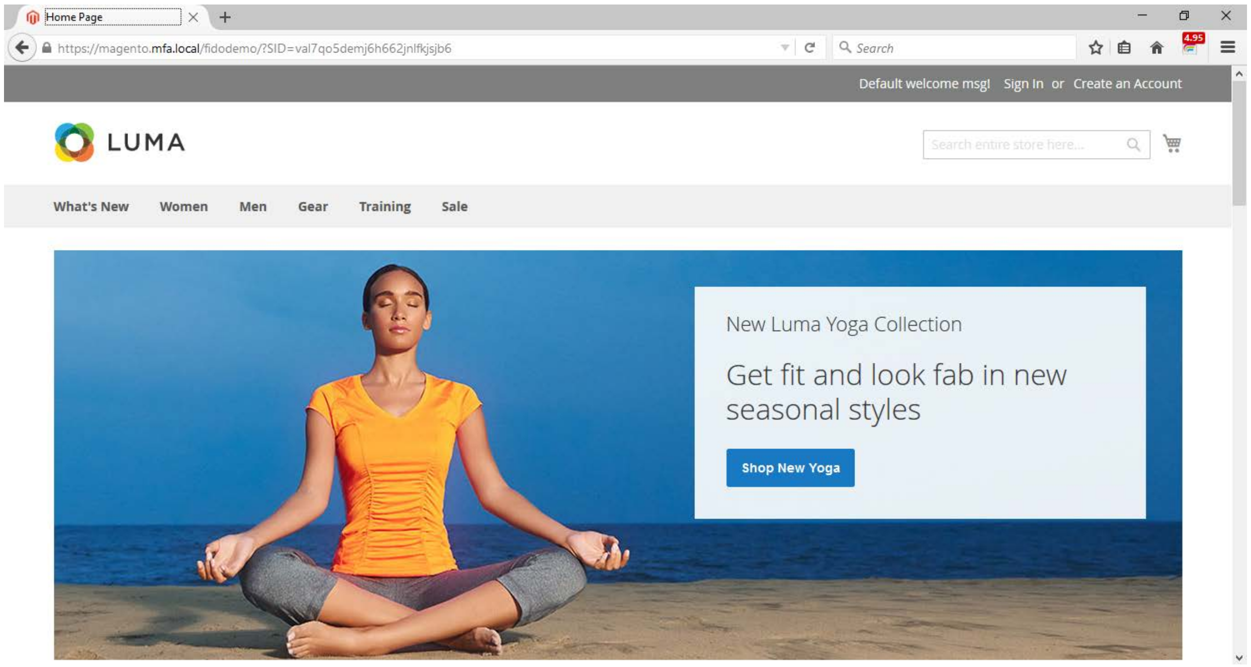
2.7.2 FIDO Key Registration

This section provides information for associating the FIDO key with the purchaser's account that was created in Section 2.7.1. The account holder will need their FIDO key to complete the registration process.

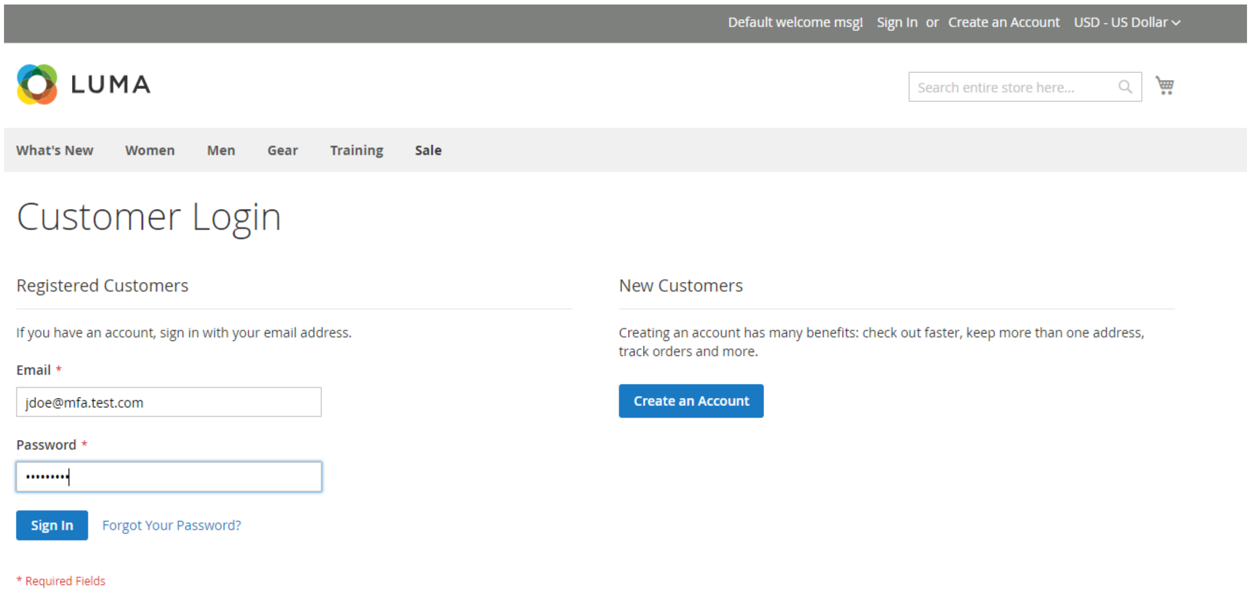
1. To begin, open a web browser and navigate to *https://magento.mfa.local/fidodemo*.

   Note: You need to have already created a Magento Example Customer Account. If you have not done so, please refer to Section 2.7.1.

2. Click **Sign In.**


Home Page
https://magento.mfa.local/fidodemo/?SID=val7qo5demj6h662jnlfkjsjb6
Search
Default welcome msg! Sign In or Create an Account
LUMA
Search entire store here...
What's New
Women
Men
Gear
Training
Sale
New Luma Yoga Collection
Get fit and look fab in new seasonal styles
Shop New Yoga

3. Fill out the **Email** and **Password** for the example customer account that was created in Section 2.7.1.


Default welcome msg! Sign In or Create an Account USD - US Dollar
LUMA
Search entire store here...
What's New
Women
Men
Gear
Training
Sale

Customer Login

Registered Customers

If you have an account, sign in with your email address.

Email *

jdoe@mfa.test.com

Password *

•••••••••

Sign In
Forgot Your Password?

* Required Fields

New Customers

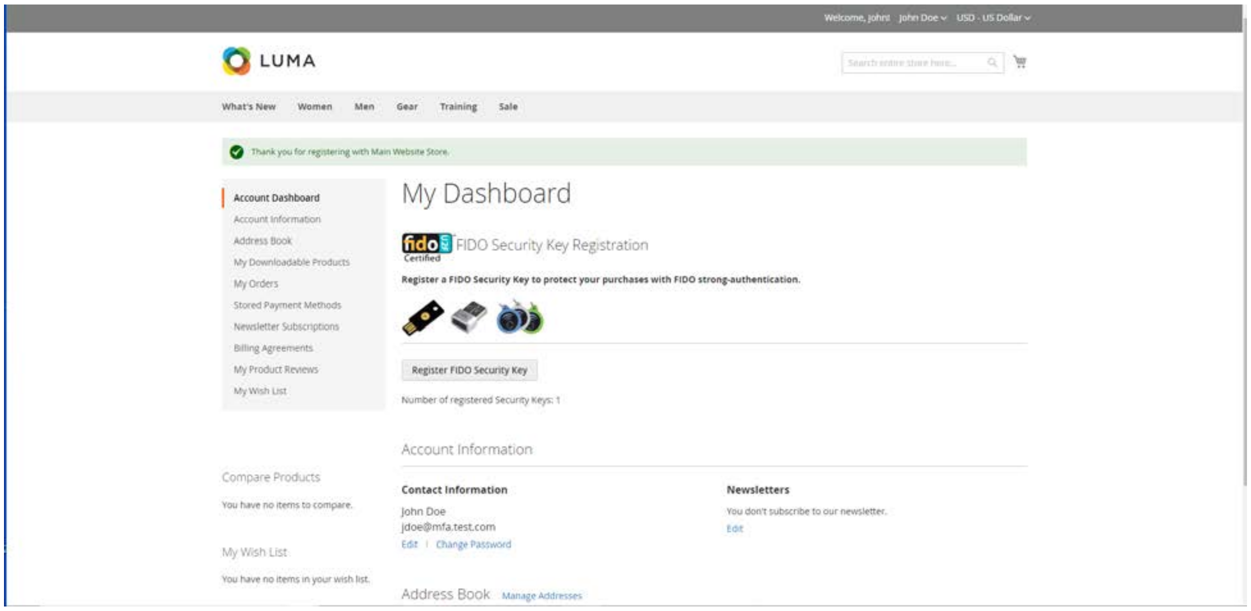
Creating an account has many benefits: check out faster, keep more than one address, track orders and more.

a. **Email:** jdoe@mfa.test.com

b. **Password:** <password>

4. Click **Sign In.**

5. On the **Account Dashboard** page, click **Register FIDO Security Key.**



Thank you for registering with Main Website Store.

My Dashboard

FIDO Security Key Registration

Register a FIDO Security Key to protect your purchases with FIDO strong-authentication.

Number of registered Security Keys: 1

Account Information

Contact Information

John Doe
jdoe@mfa.test.com

Newsletters

You don't subscribe to our newsletter.

Compare Products

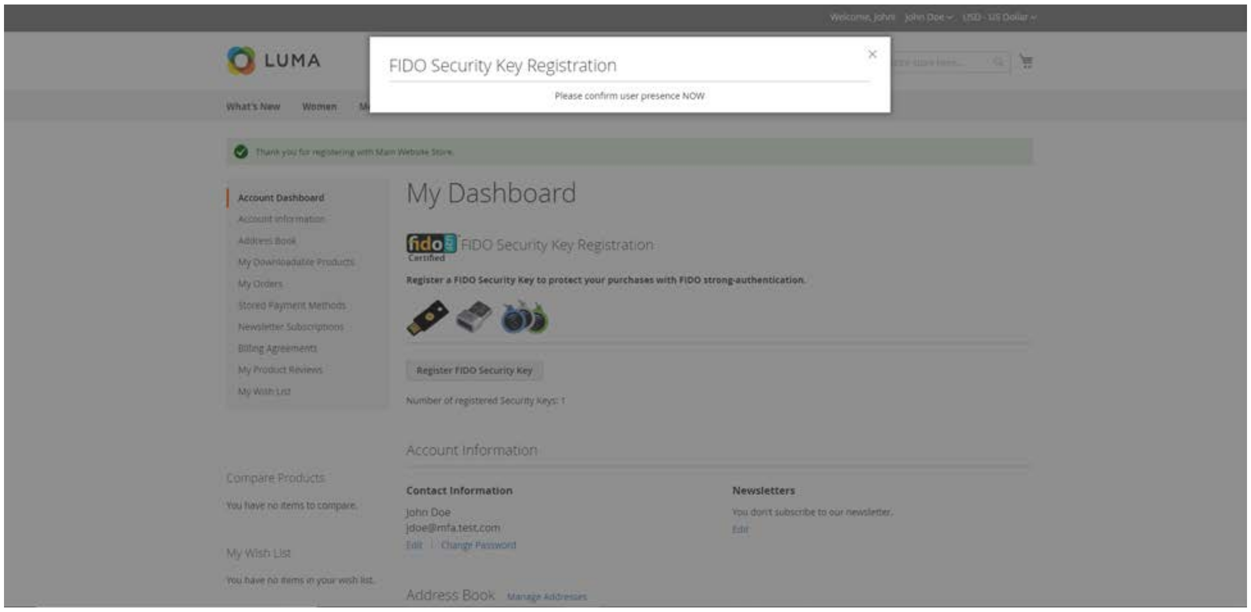
You have no items to compare.

My Wish List

You have no items in your wish list.

Address Book

6. The FIDO Authentication Engine will prompt “Please confirm user presence NOW.”



Welcome, John! John Doe USD - US Dollar

LUMA

What's New Women

FIDO Security Key Registration

Please confirm user presence NOW

Thank you for registering with Main Website Store.

Account Dashboard
Account Information
Address Book
My Downloadable Products
My Orders
Stored Payment Methods
Newsletter Subscriptions
Billing Agreements
My Product Reviews
My Wish List

My Dashboard

fido Certified FIDO Security Key Registration

Register a FIDO Security Key to protect your purchases with FIDO strong-authentication.

Register FIDO Security Key

Number of registered Security Keys: 1

Account Information

Contact Information
John Doe
jdoe@mfa.test.com
Edit | Change Password

Newsletters
You don't subscribe to our newsletter.
Edit

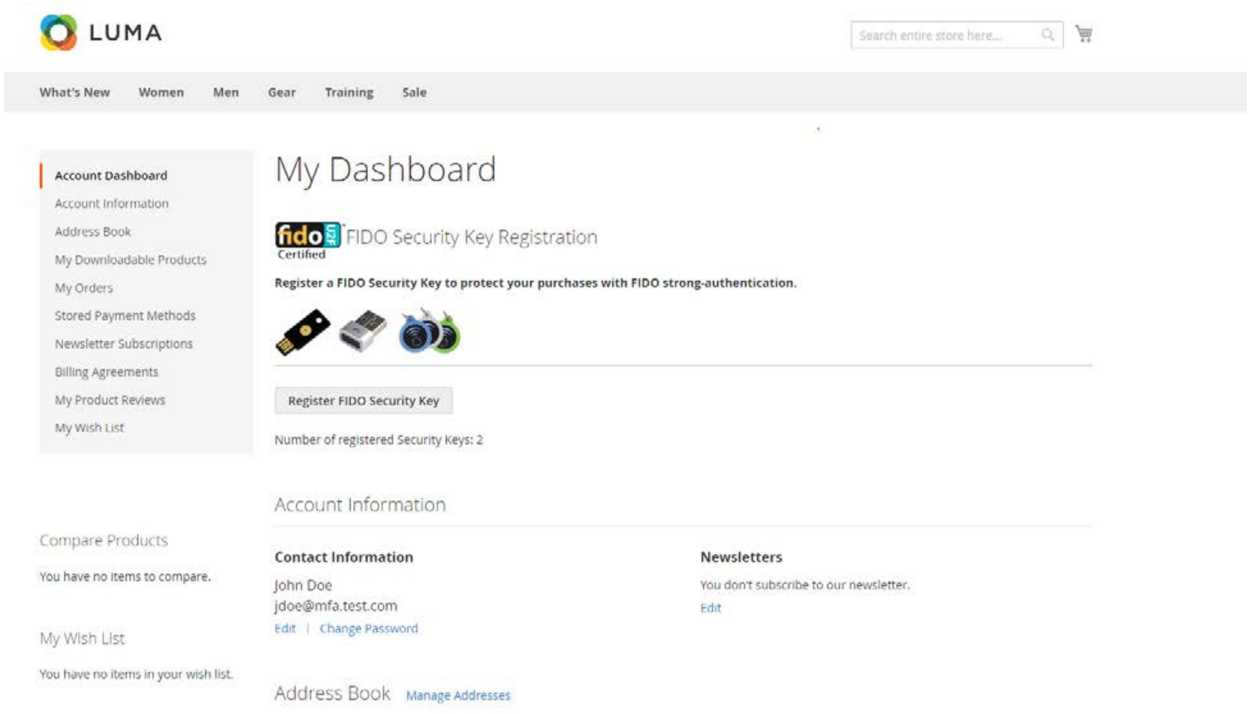
Compare Products
You have no items to compare.

My Wish List
You have no items in your wish list.

Address Book Manage Addresses

Insert the Yubico YubiKey NEO Security Key [15], [16] into an available Universal Serial Bus (USB) slot on the computer, and then place a finger on the gold contact pad.

7. Successful key registration will result in returning to the **Account Dashboard** page.



My Dashboard

FIDO Security Key Registration

Register a FIDO Security Key to protect your purchases with FIDO strong-authentication.

Number of registered Security Keys: 2

Account Information

Contact Information

John Doe
jdoe@mfa.test.com

Newsletters

You don't subscribe to our newsletter.

You have no items to compare.

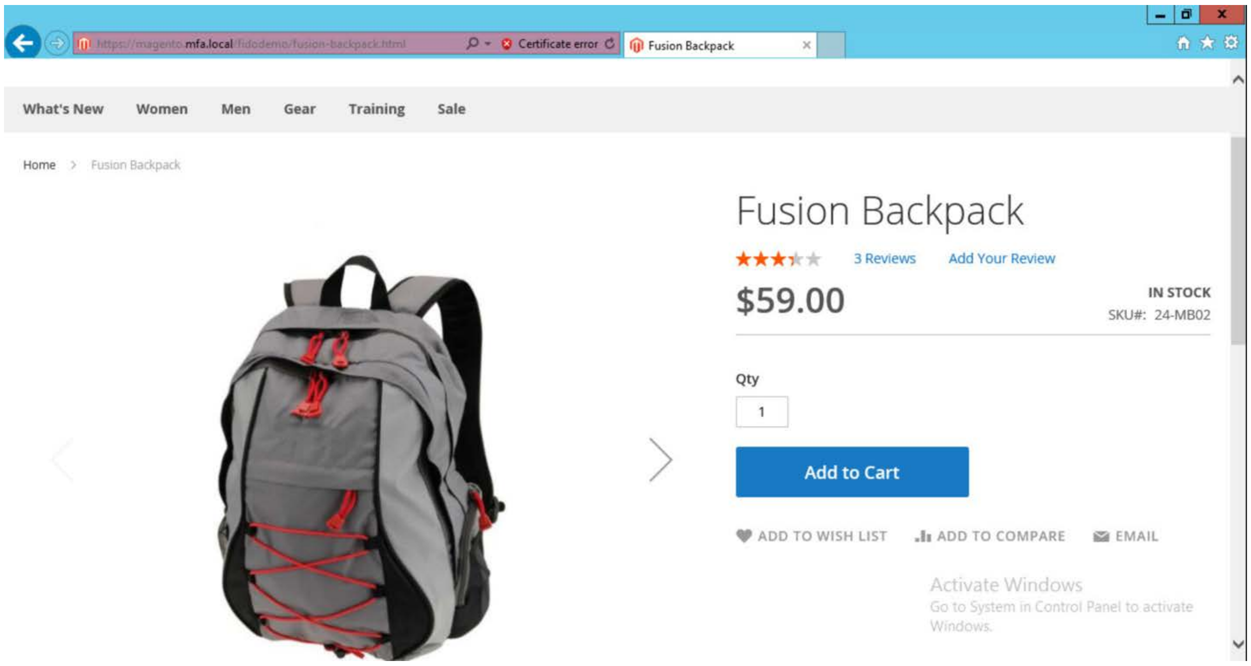
You have no items in your wish list.

Address Book

2.7.3 Testing Customer Checkout

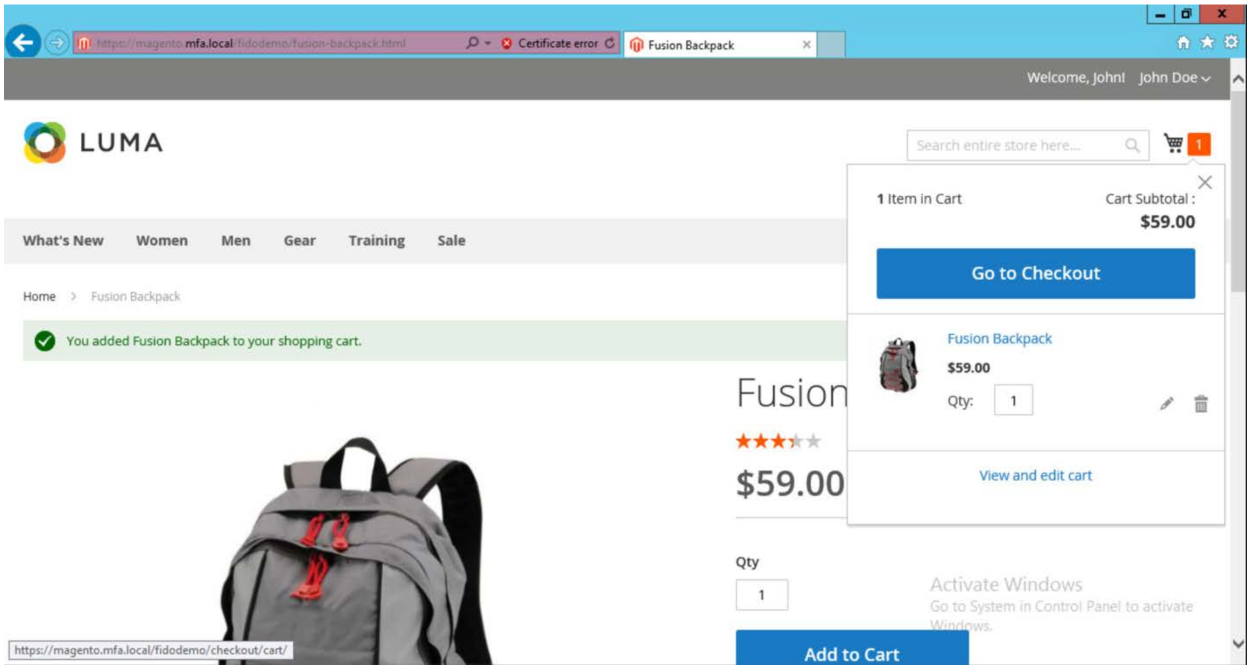
This section provides information for testing that the FIDO server is prompting for a second form of authentication for purchases above $25. This section assumes that an example customer account has been created with a registered FIDO Security Key (Section 2.7.1 and Section 2.7.2).

1. Open a web browser and navigate to *https://magento.mfa.local/fidodemo*.
2. If not already logged in to an example customer account, select **Sign In** from the Magento home page and log in with the following credentials:
    a. **Email:** jdoe@mfa.test.com
    b. **Password:** <password>
3. You will be taken to the **Account Dashboard** page.
4. From there, navigate back to *https://magento.mfa.local/fidodemo*.
5. Scroll down the page and select any item over $25. For our demonstration, we have selected the Fusion Backpack.

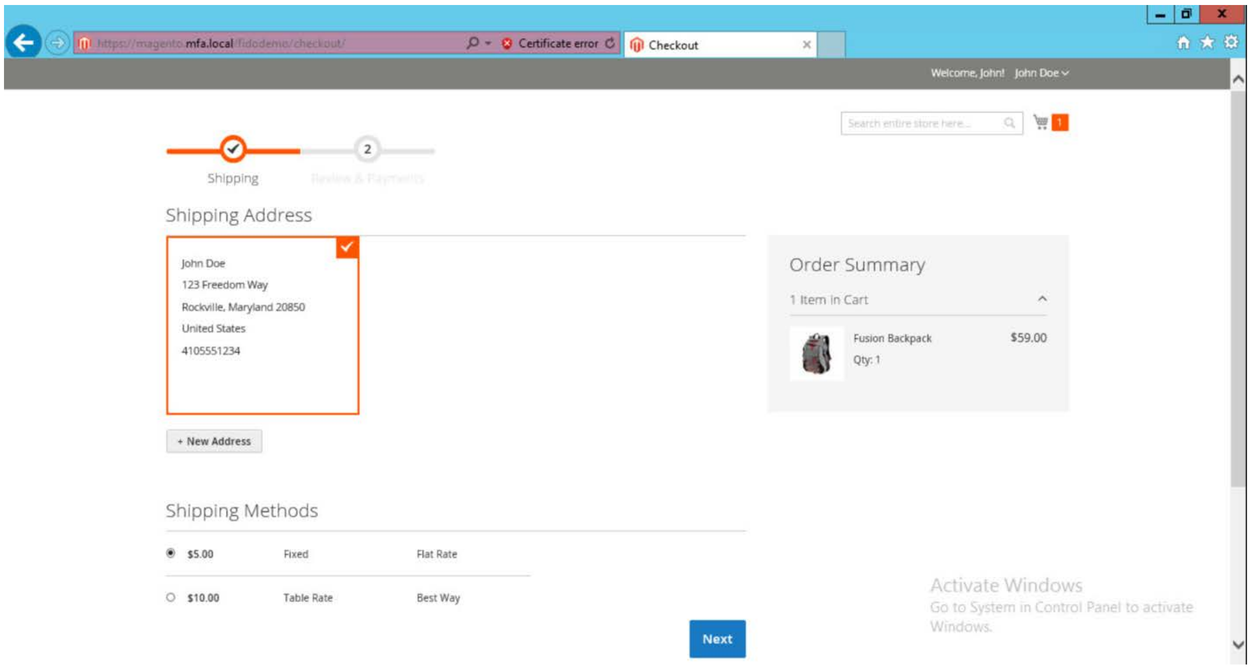


6. Click **Add to Cart.**

7. Click the shopping-basket icon, and then click **Go to Checkout.**

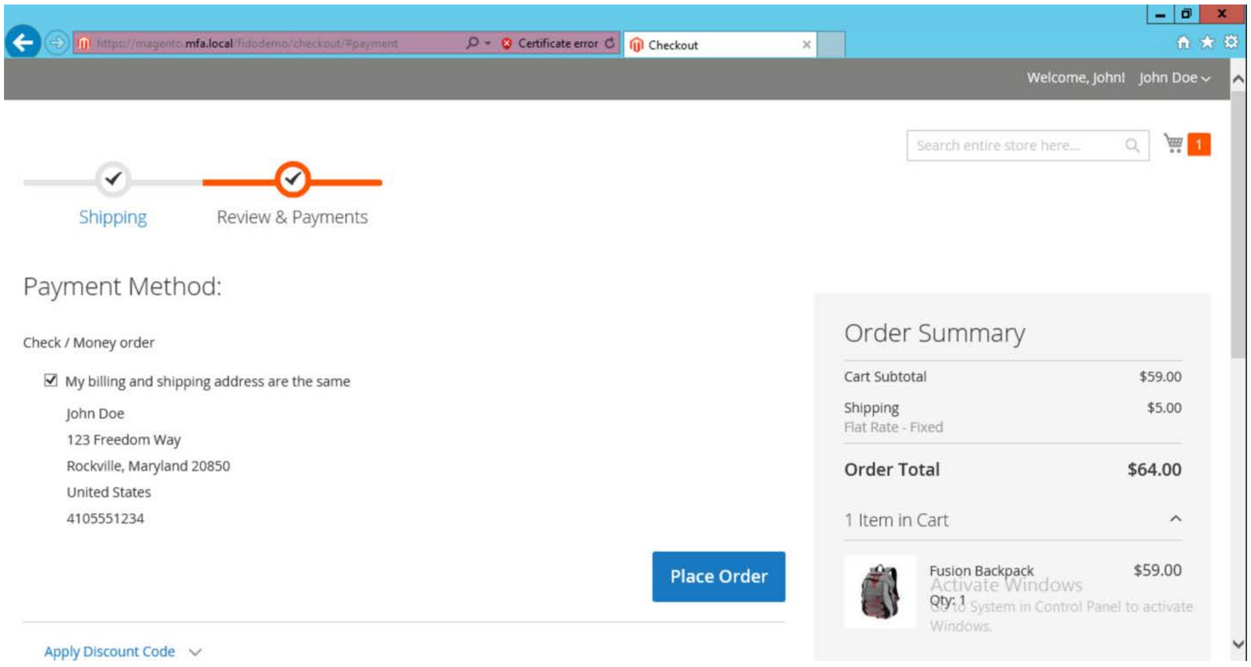


8. Under **Shipping Methods,** select the **Fixed–Flat Rate** radio bubble.

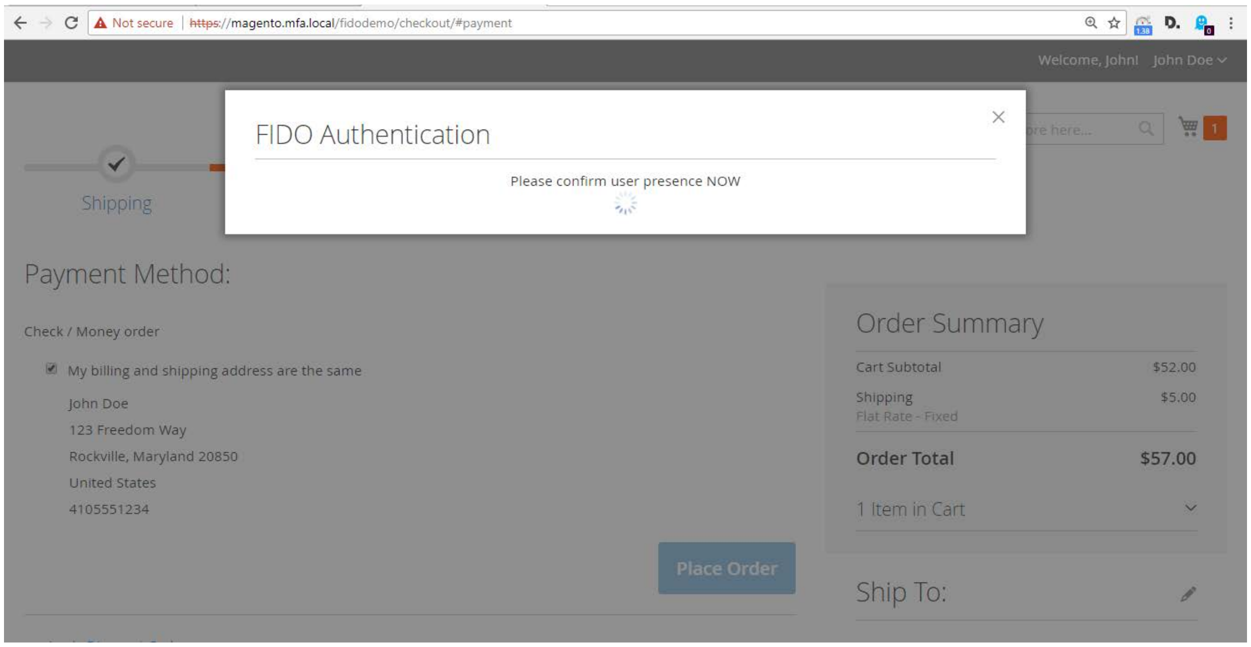


9. Click **Next.**

10. On the following page, select **Place Order.**



11. The FIDO Authentication Engine will prompt "Please confirm user presence NOW."


Not secure | https://magento.mfa.local/fidodemo/checkout/#payment

Welcome, John! John Doe

FIDO Authentication

Please confirm user presence NOW

Shipping

Payment Method:

Check / Money order

My billing and shipping address are the same

John Doe
123 Freedom Way
Rockville, Maryland 20850
United States
4105551234

Place Order

Order Summary

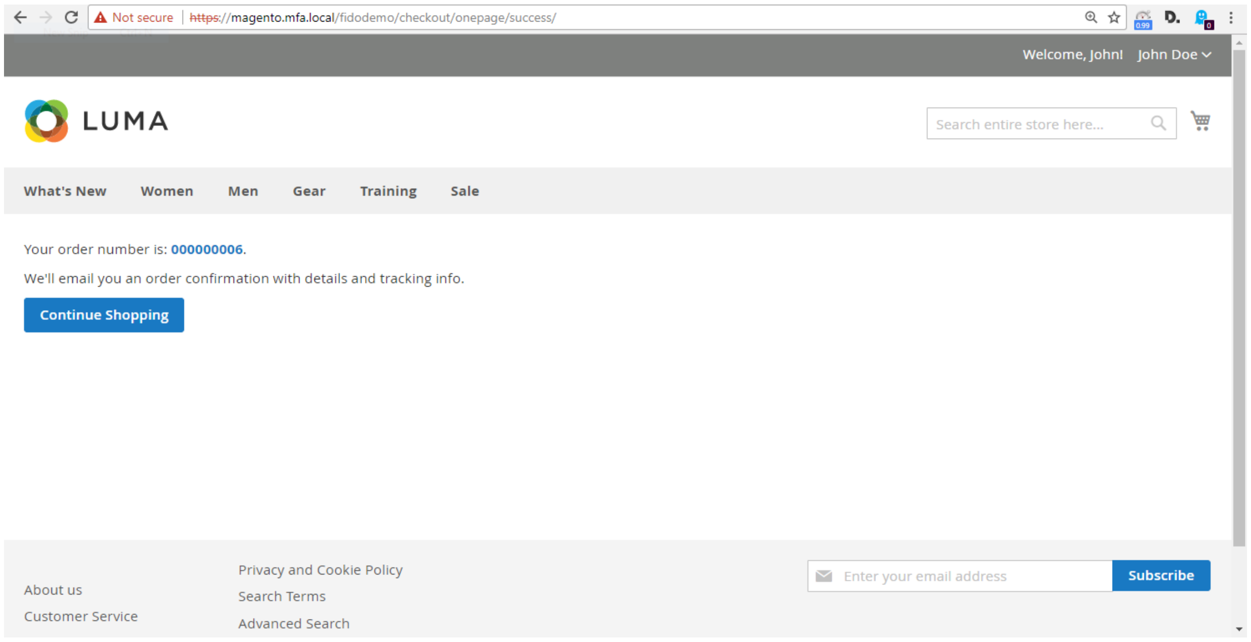
Cart Subtotal	$52.00
Shipping Flat Rate - Fixed	$5.00
Order Total	$57.00

1 Item in Cart

Ship To:

12. Insert the Yubico YubiKey NEO Security Key into an available USB slot on the computer, and then place a finger on the gold contact pad.

13. Successfully activating the FIDO token will result in the order confirmation page.


Not secure | https://magento.mfa.local/fidodemo/checkout/onepage/success/

Welcome, John! John Doe

LUMA

Search entire store here...

What's New Women Men Gear Training Sale

Your order number is: 000000006.

We'll email you an order confirmation with details and tracking info.

Continue Shopping

About us
Customer Service

Privacy and Cookie Policy
Search Terms
Advanced Search

Enter your email address Subscribe

Appendix A FIDO U2F Security Key Registration

Fast IDentity Online (FIDO) authentication requires registering one or more *FIDOU2FAuthenticators,* also known as FIDO Universal Second Factor (U2F) Security Keys, or security keys. Security keys can be used for authentication on multiple information systems or websites. If the purchaser already has a U2F, then they can use that same U2F as their multifactor authenticator for the electronic commerce (e-commerce) example implementations depicted in this guide.

FIDO authentication in these example implementations is accomplished by using the magfido *FIDOU2FAuthenticator* module created by StrongKey for the Magento Open Source platform. When deploying the example implementations, there are three parts to the process. While these three parts all execute in sequence without the purchaser being aware of each part, it is helpful to explain each part so that developers understand the workflow.

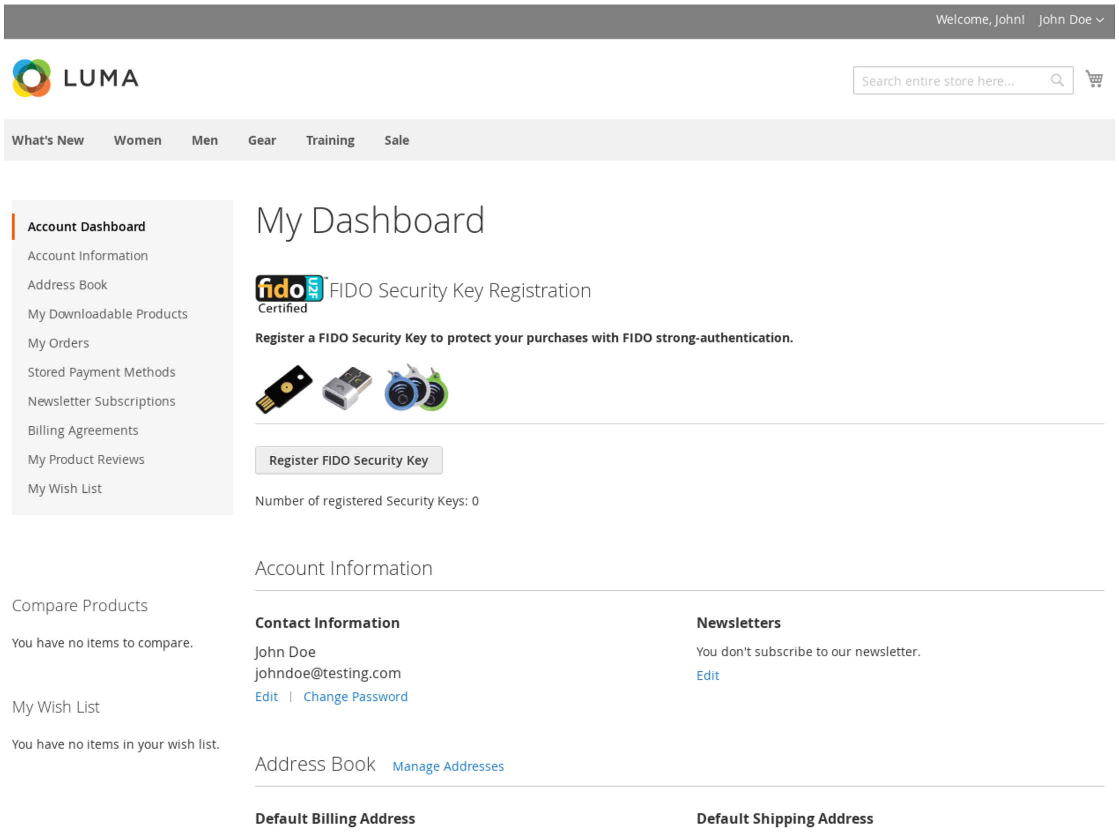
A.1 Display Function

In this part of the process, the Magento layout file *customer_account_index.xml* loads code from the *fido_register.phtml* file on the server side to perform these two functions:

1. Generate Hypertext Markup Language (HTML) that displays FIDO registration purchaser-interface components in the browser, along with summary information of the number of security keys that a purchaser may have registered. The summary information on registered keys is shown above the Recent (Magento) Orders section, which normally appears at the top of the dashboard.
2. Execute the FIDO registration process to register a new FIDO Security Key, using JavaScript embedded in the *fido_register.phtml* file.

If a purchaser has not yet registered a FIDO Security Key within Magento, then the HTML displays a zero (0) value for the number of registered keys, and a button to register a new security key (Figure A-1).

Figure A-1 Browser Display Without Any Security Keys Registered



My Dashboard

FIDO Security Key Registration

Register a FIDO Security Key to protect your purchases with FIDO strong-authentication.

Register FIDO Security Key

Number of registered Security Keys: 0

Account Information

Contact Information

John Doe
johndoe@testing.com
Edit | Change Password

Newsletters

You don't subscribe to our newsletter.
Edit

Compare Products

You have no items to compare.

My Wish List

You have no items in your wish list.

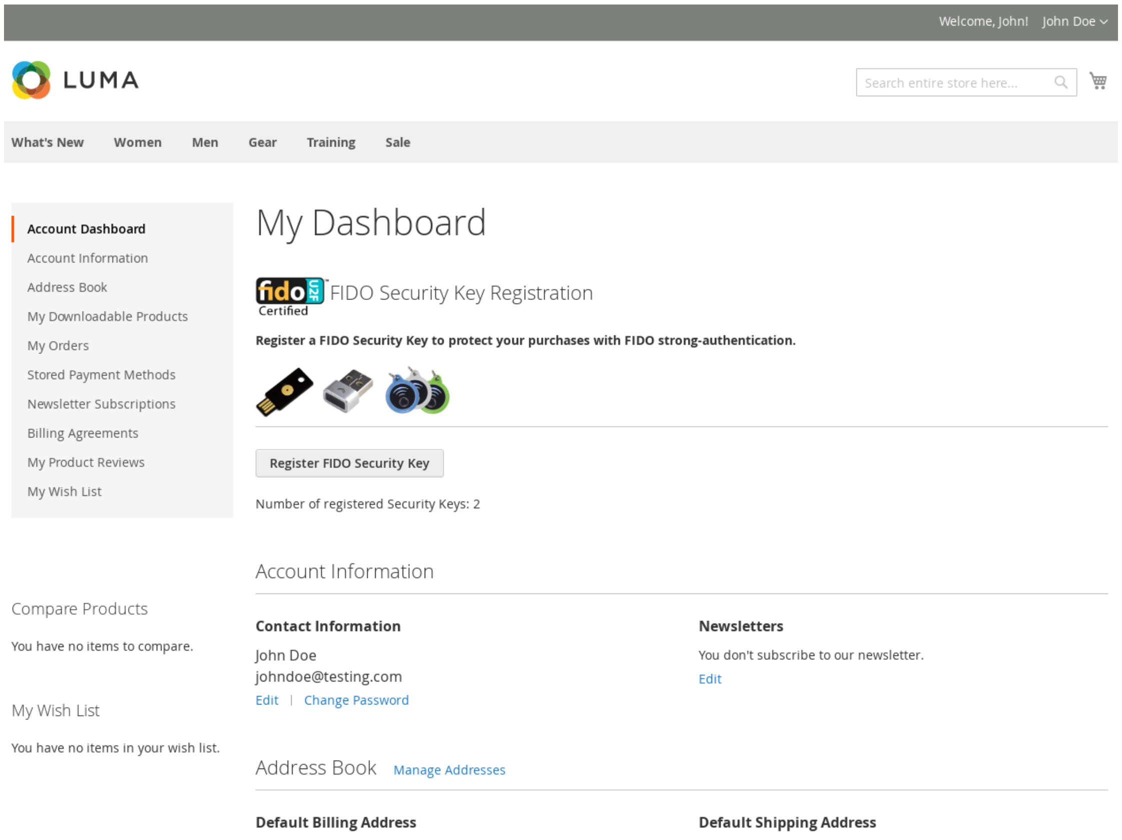
Address Book Manage Addresses

Default Billing Address

Default Shipping Address

If a purchaser has registered one or more security keys to their account—which the FIDO U2F protocol allows—then the *FIDOU2FAuthenticator* module displays the number of security keys registered by the purchaser. Otherwise, it displays 0. The HTML display for such a purchaser's registered keys resembles the depiction shown in Figure A-2.

Figure A-2 Browser Display with Two Security Keys Registered


Welcome, John! John Doe

LUMA

Search entire store here...

What's New Women Men Gear Training Sale

- Account Dashboard
- Account Information
- Address Book
- My Downloadable Products
- My Orders
- Stored Payment Methods
- Newsletter Subscriptions
- Billing Agreements
- My Product Reviews
- My Wish List

Compare Products

You have no items to compare.

My Wish List

You have no items in your wish list.

My Dashboard

fido U2F Certified FIDO Security Key Registration

Register a FIDO Security Key to protect your purchases with FIDO strong-authentication.

Register FIDO Security Key

Number of registered Security Keys: 2

Account Information

Contact Information

John Doe
johndoe@testing.com
Edit | Change Password

Newsletters

You don't subscribe to our newsletter.
Edit

Address Book Manage Addresses

Default Billing Address

Default Shipping Address

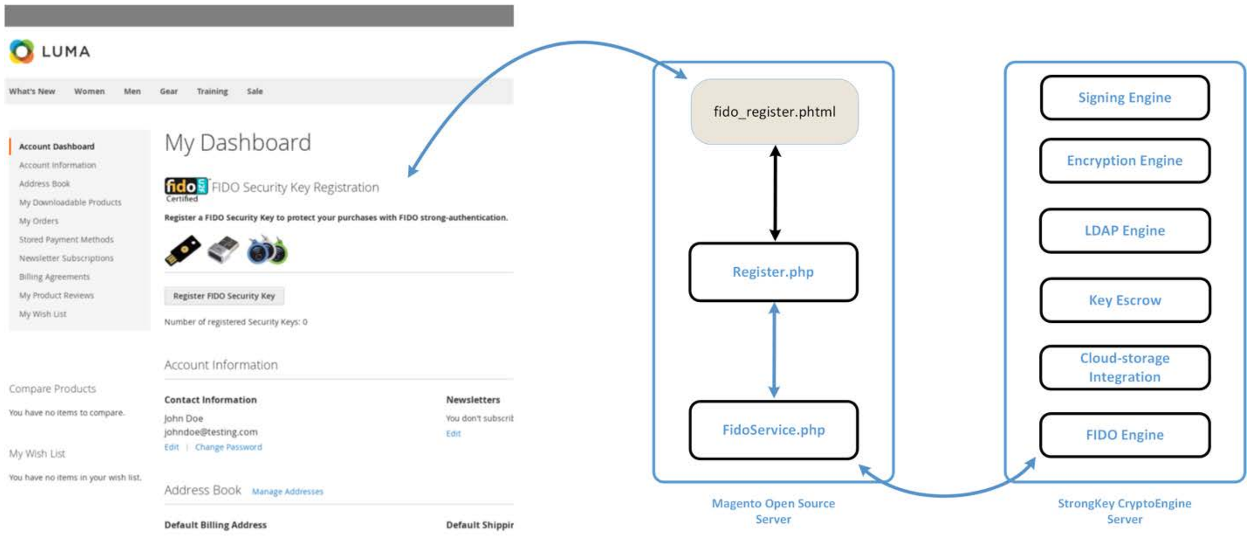
To determine the number of FIDO Security Keys registered by a purchaser within their account, the server code in *fido_register.phtml* calls the "block" file, *Register.php*. This Hypertext Preprocessor (PHP) file, in turn, invokes *FidoService.php* to call a web service (also sometimes known as "consume a web service") on a previously configured FIDO U2F server (implemented in StrongKey CryptoEngine [SKCE]) known to the Magento instance. The web-service request retrieves security-key-related information for the specific purchaser from the FIDO server.

FidoService.php parses the retrieved number of registered keys and returns the value to *Register.php*, which, in turn, returns the number to *fido_register.phtml* that generates HTML for the browser to display.

Note: In this example implementation, *Register.php* is executed only when the purchaser navigates to their purchaser-dashboard page. If a new security key is registered while on that page, then the page is automatically refreshed upon completion of the transaction to display the correct number of registered security keys.

An overview diagram of the first part of the registration process—that displays the current number of registered security keys, if any—is shown in Figure A-3.

Figure A-3 Display Function Part of the FIDO Registration Process



Register a FIDO Security Key to protect your purchases with FIDO strong-authentication.

Number of registered Security Keys: 0

You have no items to compare.

You have no items in your wish list.

John Doe
johndoe@testing.com

You don't subscrit

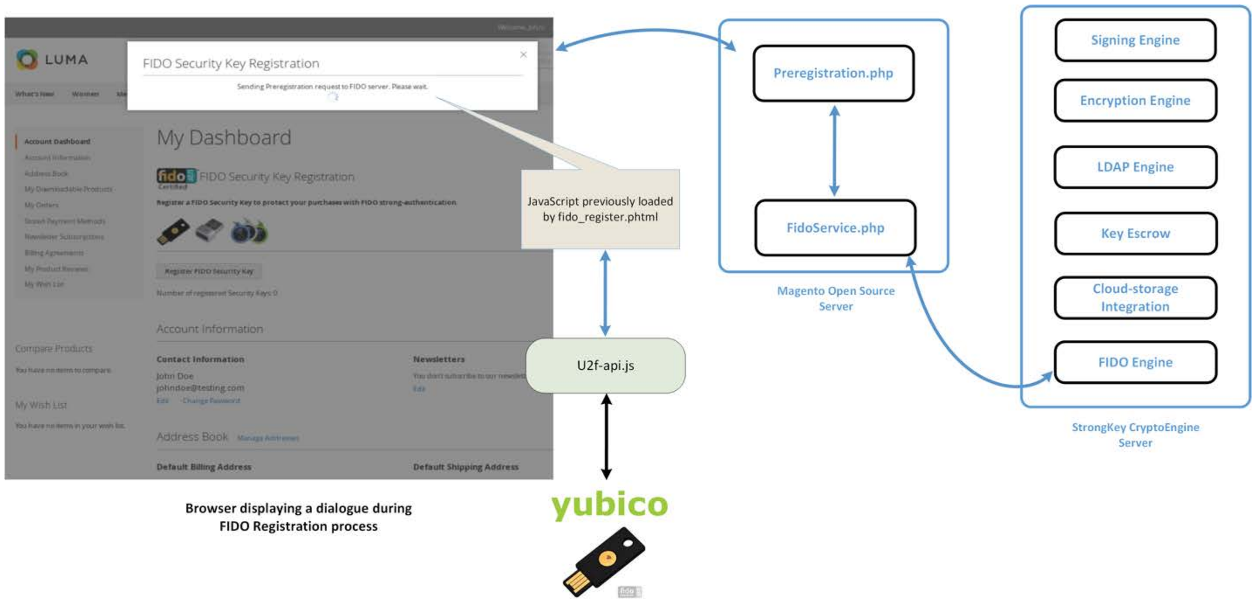
A.2 Preregister Function

The second part of the FIDO registration process acquires a challenge from the FIDO U2F server (SKCE) for processing within the purchaser's FIDO Security Key (Figure A-4).

When the **Register FIDO Security Key** button on the browser is clicked by the purchaser, JavaScript that was loaded earlier in the web page (by *fido_register.phtml*) makes an Asynchronous JavaScript and XML (Extensible Markup Language) (AJAX) call to *Preregistration.php* on the Magento server, which, in turn, invokes *FidoService.php* to call the ***preregister*** web-service operation on the SKCE. SKCE returns a nonce, along with a list of previously registered FIDO Security Keys, if any. If this is the first security key being registered, then this list is empty.

Note: In the FIDO U2F protocol, currently registered security keys, if any, are returned by the FIDO server to safeguard that security keys do not attempt to generate a duplicate key for purchasers on the same device. This implies that manufacturers of FIDO Security Keys must implement logic to ensure that they check for an existing key pair for a purchaser for the specific website. A FIDO Certified Authenticator will always have this logic implemented because it is part of the protocol-conformance testing to achieve the FIDO Certified label.

Figure A-4 Preregistration Part of the FIDO Registration Process


FIDO Security Key Registration
Sending Preregistration request to FIDO server. Please wait.
My Dashboard
JavaScript previously loaded by fido_register.phtml
U2f-api.js
yubico
Browser displaying a dialogue during FIDO Registration process
Preregistration.php
FidoService.php
Magento Open Source Server
Signing Engine
Encryption Engine
LDAP Engine
Key Escrow
Cloud-storage Integration
FIDO Engine
StrongKey CryptoEngine Server

Upon receiving the challenge, the browser and the security key interact with each other by using the *u2f-api.js* library to perform FIDO U2F-specified protocol functions. If the security key does not already have a cryptographic key pair for this specific website domain, then it requires the purchaser to perform an action to prove their presence in front of the computer. Upon the purchaser doing so, it generates a new Elliptic Curve Digital Signature Algorithm (ECDSA) key pair.

The "purchaser action" may be something chosen by the manufacturer of the security key, such as these actions:

- touching a metallic component or pressing a button that has a blinking light-emitting diode
- removing and reinserting a USB-based security key
- bringing a near field communication (NFC)-based security key near the NFC-enabled computer/mobile device
- scanning their finger or iris on a mobile device enabled with biometric capabilities
- additional manufacturer choices

FIDO protocols do not mandate any specific user/purchaser action for testing human presence. Manufacturers are at liberty to choose whatever complies with the protocol.

A.3 Register Function

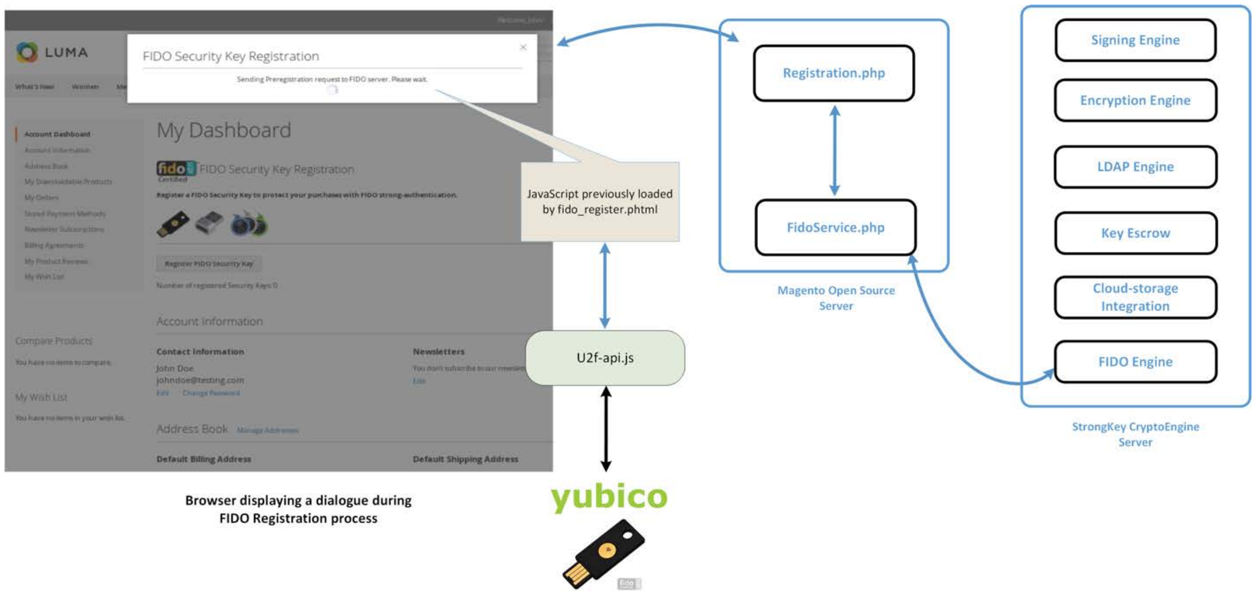
The third and last part of the FIDO registration process generates a new key pair for the purchaser for the specific website domain on the purchaser's FIDO Security Key, digitally signs the challenge from the FIDO U2F server (SKCE), and then submits a package of the response to SKCE for processing.

When the purchaser has "activated" their FIDO Security Key by using the mechanism that the manufacturer designed into the process, the security key generates a new ECDSA key pair, uses the newly generated private key from the key pair to digitally sign the nonce, and assembles a package of information to return to the browser. The browser sends the package to *Registration.php*, which, in turn, sends the package to *FidoService.php*, which finally calls the *register* web-service operation on SKCE to register the newly generated public key with the FIDO server.

During this process, *fido_register.phtml* displays a modal dialogue to notify purchasers of progress and/or error messages, should something go wrong. Any interaction with the modal dialogue, such as closing it, does not affect the operation. The operation continues until it succeeds or fails.

This last step of the registration process is shown in Figure A-5.

Figure A-5 Third and Final Step of the FIDO Registration Process



FIDO Security Key Registration
Sending Preregistration request to FIDO server. Please wait.
LUMA
My Dashboard
FIDO Security Key Registration
Register FIDO Security Key
Account Information
JavaScript previously loaded by fido_register.phtml
U2f-api.js
Browser displaying a dialogue during FIDO Registration process
yubico
Registration.php
FidoService.php
Magento Open Source Server
Signing Engine
Encryption Engine
LDAP Engine
Key Escrow
Cloud-storage Integration
FIDO Engine
StrongKey CryptoEngine Server

A.3.1 The Checkout Process

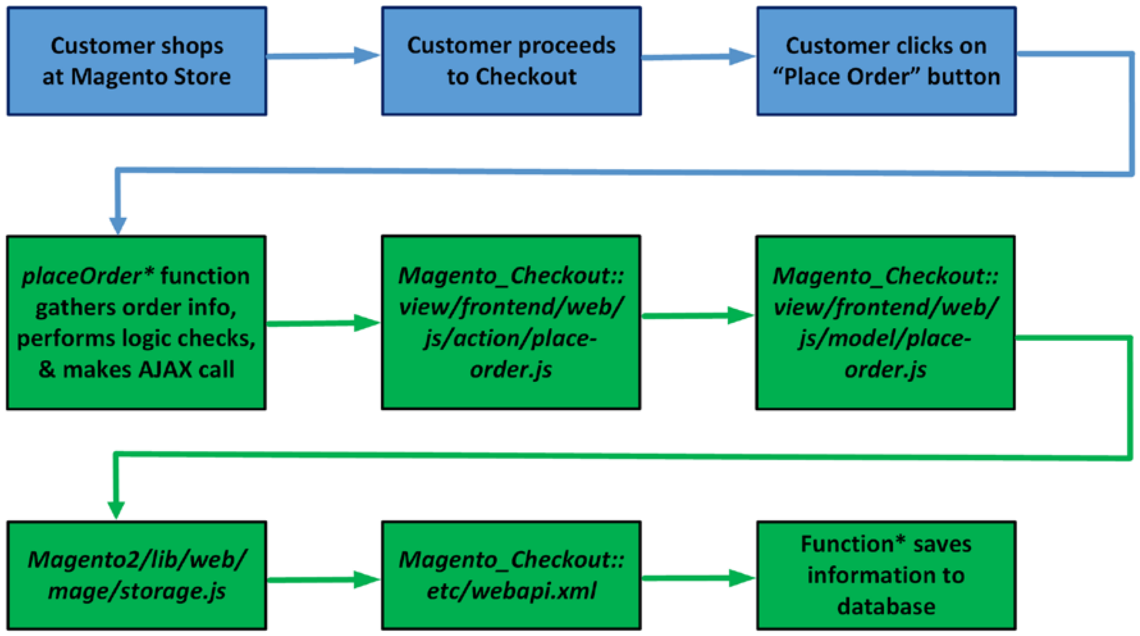
The *FIDOU2FAuthenticator* module must integrate with Magento's default checkout workflow.

Before describing the FIDO authentication process, a brief background of the default checkout workflow is presented below.

1. Purchasers browse the e-commerce website to purchase one or more items.
2. Purchasers place and remove items in and out of their shopping cart until they decide to purchase the items in their shopping cart.
3. Purchasers click **Proceed to Checkout.**
4. At this point, the checkout process requires the purchaser to fill out billing and shipping information and then to click **Place Order.**
5. This causes the browser to run JavaScript code, which makes an AJAX call to submit the shopping cart, billing address, and payment information to the Magento server.
6. The Magento server processes the information and saves it to its database—or returns an error if there is an exception—confirming the conclusion of the transaction.

The checkout workflow is displayed in Figure A-6.

Figure A-6 Magento Checkout Workflow



Customer shops at Magento Store
Customer proceeds to Checkout
Customer clicks on "Place Order" button
*placeOrder** function gathers order info, performs logic checks, & makes AJAX call
Magento_Checkout:: view/frontend/web/ js/action/place-order.js
Magento_Checkout:: view/frontend/web/ js/model/place-order.js
Magento2/lib/web/ mage/storage.js
Magento_Checkout:: etc/webapi.xml
Function* saves information to database

Note: In Figure A-6,

* placeOrder is in Magento_Checkout::view/frontend/web/js/view/payment/default.js

# savePaymentInformationAndPlaceOrder is in Magento_Checkout::PaymentInformationManagement

By understanding the above Magento default checkout workflow, you can better understand how the example implementations' FIDO authentication flow is implemented.

A.3.2 The FIDO Authentication Flow for the Example Implementations

The *FIDOU2FAuthenticator* module, when installed, will inject itself into the workflow described above. The primary modification that FIDO authentication makes to the checkout process is to override the *placeOrder* function of *Magento_Checkout/view/payment/default.js.*

1. The new *placeOrder* function makes an AJAX call to the *RiskAssessor.php* on the Magento server to determine whether FIDO authentication is required (based on this example implementation's rule to check whether the total order is greater than $25).

2. If the total is $25 or less, then the checkout data is sent to the Magento server to be persisted directly without any FIDO actions. However, if the order total exceeds $25, then another AJAX call is made to *FidoService.php* to request a FIDO challenge from SKCE. This is accomplished by *FidoService.php* making a *preauthenticate* web-service request to SKCE, the FIDO U2F server. *FidoService.php* returns the challenge nonce to the calling JavaScript in the customer's browser.

3. Upon receiving the challenge, the browser interacts with *u2f-api.js* to prompt the customer to digitally sign the challenge by using their FIDO Security Key.

4. Once the challenge nonce has been signed by using the FIDO Security Key, the digital signature is appended to checkout data that is normally sent to the Magento server.

5. On the server, where the *Magento_Checkout/Model/PaymentInformationManagement savePaymentInformationAndPlaceOrder* function has been overridden, Magento receives the checkout data and checks again if FIDO authentication is required. This is to ensure that web-service requests to the back-end services are not manipulated to bypass FIDO strong authentication.

6. If FIDO strong authentication is not required, then Magento goes through the standard checkout flow and persists the transaction. If FIDO strong authentication is required, then the overridden code in *PIMOverrideFidoAuthenticate.php* checks for the digital signature bytes appended to the checkout data.

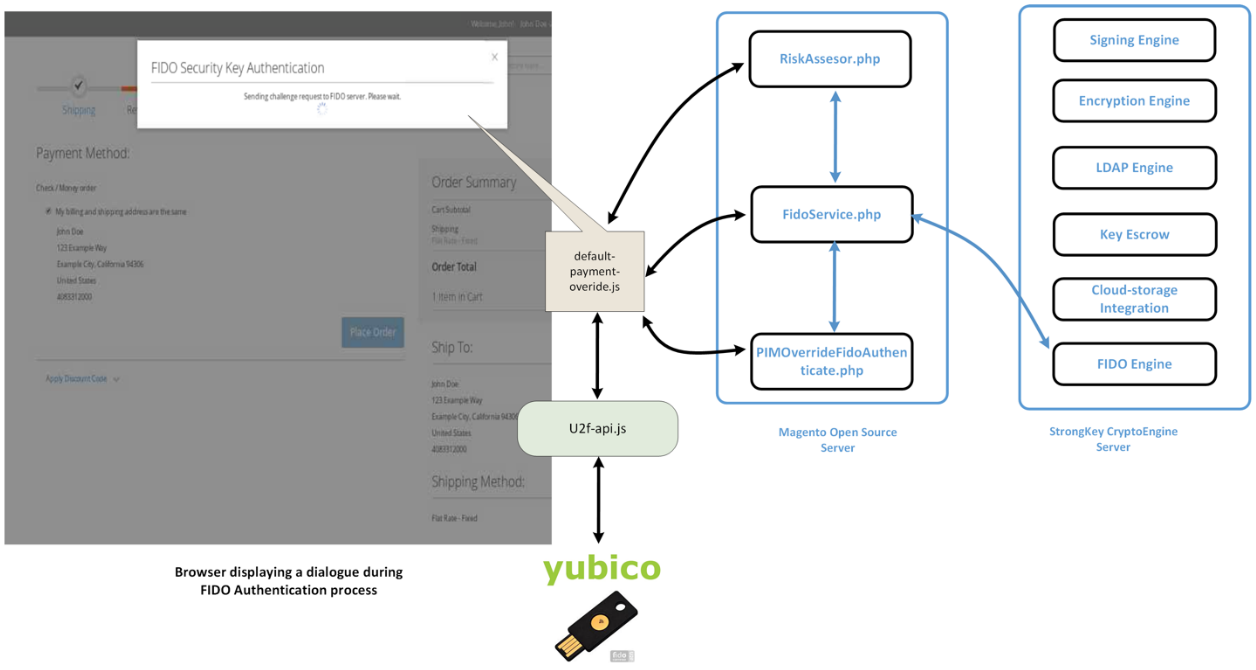
7. If the signature bytes are present, then *PIMOverrideFidoAuthenticate.php* calls the *authenticate* web-service operation (by using *FidoService.php*) on SKCE with the signature bytes.

8. If the *authenticate* web service returns successfully, then *PIMOverrideFidoAuthenticate.php* continues with the checkout process, persists transaction data to the database, and confirms the transaction to the customer. A failed response to the *authenticate* web service returns an error to the customer, and the checkout fails.

In the browser, a modal dialogue provides status messages on the FIDO strong-authentication process executing in the background (if FIDO strong authentication is determined to be necessary); otherwise, the FIDO dialogue does not display itself. As in the FIDO registration workflow, closing the modal dialogue does not stop the FIDO authentication process, and interacting with the browser window in any way does not change the behavior.

Figure A-7 provides an overview of the FIDO authentication process at a high level.

Figure A-7 Overview of the FIDO Authentication Process


FIDO Security Key Authentication
Sending challenge request to FIDO server. Please wait.
Browser displaying a dialogue during FIDO Authentication process
default-payment-overide.js
U2f-api.js
yubico
RiskAssesor.php
FidoService.php
PIMOverrideFidoAuthenticate.php
Magento Open Source Server
Signing Engine
Encryption Engine
LDAP Engine
Key Escrow
Cloud-storage Integration
FIDO Engine
StrongKey CryptoEngine Server

A.3.3 Information About the magfido Files and Directories

This section provides additional information regarding files referenced and/or modified by StrongKey to implement FIDO U2F MFA for these example implementations. If you are familiar with Magento, then you may skip this section; others may find this section to be helpful in understanding what must be done to integrate FIDO U2F into their Magento instance in a production environment.

Magento includes several boilerplate/configuration files: *composer.json* and *registration.php* are those that must be included in every Magento module because they identify the module to the Magento system.

The *etc* folder contains configuration files:

- *module.xml* is a boilerplate file.

- *di.xml* tells Magento to override the default *PaymentInformationManagement.php* file with StrongKey's custom version (named *PIMOverrideFidoAuthenticate.php*).
- *extension_attributes.xml* tells Magento that purchase-transaction data sent to the server may have signature data appended to it, which can be identified by the attribute name *signature*.
- *etc/frontend/di.xml* adds an *AdditionalConfigProvider* that supplies the MFA modal dialogue with the file name *loading.gif*.
- *routes.xml* tells Magento that this module defines controllers that will handle uniform resource locator (URL) requests to fidou2fauthenticator.

The *api* folder contains interface files describing valid functions of the models *FidoService* and *RiskAssessor*. The interface files are named *FidoServiceInterface.php* and *RiskAssessorInterface.php*.

The *block* folder contains server-side logic to generate views displayed by the browser. Specifically, it contains the file *Register.php* that provides the base URL for AJAX calls in the registration workflow and returns the number of security keys registered to the online customer.

The *controller* folder contains controllers to handle AJAX calls from the browser. The controllers map to SKCE web services, such as *preregistration*, *registration*, and *preauthentication*. Because FIDO authentication is part of the checkout process and is performed in conjunction with payment data, an explicit controller for FIDO authentication is not defined here but is included as part of *PIMOverrideFidoAuthentication*. It also contains the *RiskAssessor.php* controller to call the *RiskAssessor.php* code in the *model* folder (see below), which performs the actual risk assessment.

The *model* folder contains the following server-side logic files:

- *AdditionalConfigProvider.php* retrieves the static URL of the *loading.gif* image and adds it to variables for the browser client to deliver a better user experience.
- *FidoService.php* makes the actual web-service calls to the FIDO U2F server, SKCE.
- *RiskAssessor.php* makes the risk decision in this example implementation—to check if the order's total value is greater than $25—and returns a *Boolean* value indicating if FIDO multifactor authentication (MFA) is necessary.
- *PIMOverrideFidoAuthentication.php* implements the server-side logic to check, once again, if FIDO MFA is necessary, checking if signature bytes are appended to payment data, verifying if the supplied digital signature is valid (through *FidoService.php*), and persisting the order transaction.

The *view* folder contains the client-side logic. Because all FIDO-related workflows in this example implementation are intended for customer interaction only, there is a *frontend* folder inside the *view* folder (as opposed to an *adminhtml* folder, which would normally define views for administrators). Within the *frontend* folder, there are four groups of files:

- The first group contains files related to the registration workflow: *layout/customer_account_index.xml* directs Magento to load *templates/fido_register.phtml* above the Recent Orders section of the Customer dashboard in the browser. *fido_register.phtml* coordinates the entire FIDO registration workflow.
- The second group contains files related to the modal dialogue: *layout/checkout_index_index.xml* appends JavaScript from *web/js/view/checkout-modal.js* to JavaScript normally loaded on checkout pages. *checkout-modal.js*, in turn, loads *web/template/checkout-modal.html* with HTML that is rendered on the checkout page.
- The third group of files provides client-side logic to perform FIDO authentication. *requirejs-config.js* is a configuration file to load JavaScript libraries found in *web/js/lib*—including *u2f.js* and *common.js*, which are part of the standard distribution for FIDO U2F from Google for use with the Chrome browser—and overrides the default JavaScript in *Magento_Checkout/js/view/payment/default.js* with *web/js/default-override.js*. The latter file—*default-override.js*—provides client-side logic, including requesting the challenge nonce, getting the challenge nonce digitally signed by the FIDO Security Key, returning the digital signature, and updating the modal dialogue with progress information.
- The last group of files found in the *view/frontend* folder contains image files found in *web/images/*.

A.3.4 Solutions to Common Challenges When Configuring Magento and magfido

The following subsections provide solutions to common challenges when the magfido module is configured with Magento.

A.3.4.1 Code Was Modified but Change Did Not Take Effect

The most common reason for this issue is that Magento's cache was not cleared. Clear the browser cache from the browser's admin console, or open a terminal, change to the Magento directory (*/var/www/html/fidodemo),* and run this command:

```
php bin/magento cache:flush
```

A.3.4.2 Magento Is Unable to Read the WSDL of the FIDO Server

Possible reasons for Magento being unable to read the FIDO server's Web Services Description Language (WSDL), and thus being unable to complete the action, are explained below.

- The Fully Qualified Domain Name of the FIDO server was defined incorrectly. This can be fixed by modifying the WSDL constant in *StrongAuth_FidoValidator/Model/FidoService.php*.
- The FIDO server has a self-signed certificate that Hypertext Transfer Protocol Daemon (HTTPD) does not trust. This can be fixed by adding the self-signed certificate to the trusted certificate store located in */etc/pki/tls/certs/ca-bundle.crt*.

- The Security-Enhanced Linux (SELinux) security policy is blocking the outbound port used by HTTPD to connect to the FIDO server. This can be fixed by disabling SELinux for testing purposes. In production environments, it is recommended that SELinux rules be modified to permit HTTPD to connect to the FIDO server.

A.3.4.3 Error 500 When Attempting to Access the Home Page

This is not a FIDO-related issue but can manifest itself as a Magento-HTTPD misconfiguration. While there are many possible ways that this error can occur, the most common reason is incorrect file permissions. For testing purposes, running the following command should fix the problem to make the Magento home page accessible:

```
cd /var/www/html/fidodemo && find var vendor pub/static pub/media app/etc -type f -
exec chmod 777 {} \; && find var vendor pub/static pub/media app/etc -type d -exec
chmod 777 {} \; && chmod 777 bin/magento
```

In production environments, consider the security ramifications before adjusting permissions to the directory structure and files, and before making modifications. Please note that the command shown above is a concatenation of multiple commands executed as a single command, so either execute them in a single command (as shown above) or execute them as multiple commands in sequence:

```
cd /var/www/html/fidodemo

find var vendor pub/static pub/media app/etc -type f -exec chmod 777 {} \;

find var vendor pub/static pub/media app/etc -type d -exec chmod 777 {} \;

chmod 777 bin/magento
```

Appendix B List of Acronyms

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CentOS	Community Enterprise Operating System
DNS	Domain Name System
ECDSA	Elliptic Curve Digital Signature Algorithm
e-commerce	Electronic Commerce
FIDO	Fast IDentity Online
FQDN	Fully Qualified Domain Name
GB	Gigabyte(s)
HTML	Hypertext Markup Language
HTTPD	Hypertext Transfer Protocol Daemon
HTTPS	Hypertext Transfer Protocol Secure
ID	Identifier
IP	Internet Protocol
IT	Information Technology
JDK	Java Development Kit
JRE	Java Runtime Environment
LAMP	Linux, Apache, MySQL, PHP
LDAP	Lightweight Directory Access Protocol
MFA	Multifactor Authentication
NCCoE	National Cybersecurity Center of Excellence
NFC	Near Field Communication
NIST	National Institute of Standards and Technology
PHP	Hypertext Preprocessor
PIN	Personal Identification Number

QR	Quick Response
RAM	Random Access Memory
SELinux	Security-Enhanced Linux
SKCE	StrongKey CryptoEngine
SP	Special Publication
SPL	Splunk Search Processing Language
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
U2F	Universal Second Factor
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
WSDL	Web Services Description Language
XML	Extensible Markup Language
YUM	Yellowdog Updater Modified

Appendix C Glossary

Authentication	Verifying the identity of a user, process, or device, often as a prerequisite to allowing access to a system's resources [17]
Authenticator	Something the claimant possesses and controls (typically a cryptographic module or password) that is used to authenticate the claimant's identity [17]
Credential	An object or data structure that authoritatively binds an identity–via an identifier or identifiers–and (optionally) additional attributes to at least one authenticator possessed and controlled by a subscriber While common usage often assumes that the subscriber maintains the credential, these guidelines also use the term to refer to electronic records maintained by the Credential Service Providers that establish binding between the subscriber's authenticator(s) and identity [17].
Credential Service Provider	A trusted entity that issues or registers subscriber authenticators and issues electronic credentials to subscribers. A Credential Service Provider may be an independent third party or issue credentials for its own use [17].
Identity	An attribute, or set of attributes, that uniquely describes a subject within a given context [17]
Multifactor	A characteristic of an authentication system or an authenticator that requires more than one distinct authentication factor for successful authentication. MFA can be performed by using a single authenticator that provides more than one factor or by using a combination of authenticators that provide different factors. The three authentication factors are something you know, something you have, and something you are [17].
Multifactor Authentication (MFA)	An authentication system that requires more than one distinct authentication factor for successful authentication. Multifactor authentication can be performed by using a multifactor authenticator or by using a combination of authenticators that provide different factors. The three authentication factors are something you know, something you have, and something you are [17].
Personal Identification Number	A memorized secret typically consisting of only decimal digits [17]

Private Key	The secret part of an asymmetric key pair that is used to digitally sign or decrypt data [17]
Public Key	The public part of an asymmetric key pair that is used to verify signatures or encrypt data [17]
Public Key Certificate	A digital document issued and digitally signed by the private key of a certificate authority that binds an identifier to a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the private key. See also RFC 5280 [17].
Relying Party	An entity that relies upon the subscriber's authenticator(s) and credentials or a verifier's assertion of a claimant's identity, typically to process a transaction or grant access to information or a system [17]
Risk	The level of impact on organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals resulting from the operation of an information system, given the potential effect of a threat and the likelihood of that threat occurring [18]
Session	A persistent interaction between a subscriber and an endpoint, either a relying party or a Credential Service Provider. A session begins with an authentication event and ends with a session termination event. A session is bound by use of a session secret that the subscriber's software (a browser, application, or OS) can present to the relying party or the Credential Service Provider, in lieu of the subscriber's authentication credentials [17].
Single Factor	A characteristic of an authentication system or an authenticator that requires only one authentication factor (something you know, something you have, or something you are) for successful authentication [17]
Subscriber	A party who has received a credential or authenticator from a Credential Service Provider [17]
Token	See Authenticator [17].
Transaction	A discrete event between a user and a system that supports a business or programmatic purpose. A government digital system may have multiple categories or types of transactions, which may require separate analysis within the overall digital identity risk assessment [17].

Appendix D References

[1] FIDO Alliance. (n.d.). *Specifications Overview* [Online]. Available: https://fidoalliance.org/specifications/overview/.

[2] FIDO Alliance. (n.d.). FIDO Alliance [Online]. Available: https://fidoalliance.org/.

[3] StrongKey. (n.d.). Home–StrongKey [Online]. Available: https://www.strongkey.com/.

[4] Magento, Inc. (n.d.). *eCommerce Platform|Best eCommerce Software for Selling Online* [Online]. Available: https://magento.com/.

[5] Magento, Inc. (n.d.). *Magento Open Source* [Online]. Available: https://magento.com/products/open-source.

[6] A. Noor and A. de Leon. (Feb. 20, 2018). *FIDO U2F Integration for Magento 2* [Online]. Available: https://sourceforge.net/projects/magfido/?source=navbar.

[7] RSA. (n.d.). *RSA|Security Solutions to Address Cyber Threats* [Online]. Available: https://www.rsa.com/.

[8] RSA Security LLC. (n.d.). *Adaptive Authentication|Fraud Detection–RSA* [Online]. Available: https://www.rsa.com/en-us/products/fraud-prevention/secure-consumer-access.

[9] TokenOne. (n.d.). *TokenOne|Secure Authentication|Sydney* [Online]. Available: https://www.tokenone.com.

[10] Splunk Inc. (n.d.). Splunk [Online]. Available: https://www.splunk.com/.

[11] Splunk Inc. (n.d.). *Splunk® Enterprise* [Online]. Available: https://www.splunk.com/en_us/products/splunk-enterprise.html.

[12] Splunk Inc. (n.d.). *Splunk® Universal Forwarder: Forwarder Manual* [Online]. Available: http://docs.splunk.com/Documentation/Forwarder/7.0.2/Forwarder/Abouttheuniversalforwarder.

[13] Splunk Inc. (n.d.). *Splunk DB Connect* [Online]. Available: https://splunkbase.splunk.com/app/2686/.

[14] Splunk Inc. (n.d.). *Splunk DB Connect Details* [Online]. Available: https://splunkbase.splunk.com/app/2686/#/details.

[15] Yubico. (n.d.). *Yubico NEO* [Online]. Available: https://www.yubico.com/products/yubikey-hardware/.

[16] Yubico. (n.d.). *Yubico|YubiKey Strong Two Factor Authentication for Business and Individual Use* [Online]. Available: https://www.yubico.com/.

[17] National Institute of Standards and Technology (NIST) Special Publication 800-63-3, *Digital Identity Guidelines.* (June 2017). [Online]. Available: https://pages.nist.gov/800-63-3/.

[18] NIST Interagency/Internal Report 7298, Rev. 2, *Glossary of Key Information Security Terms.* (May 2013). [Online]. Available: https://www.nist.gov/publications/glossary-key-information-security-terms-1.