ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA E SCIENZE INFORMATICHE

ISOLAMENTO E CONTROLLO DEGLI STUDENTI DURANTE UN'ESAME MEDIANTE FIREWALL

Relazione finale in

SYSTEMS INTEGRATION

Relatore

Prof. VITTORIO GHINI

Presentata da

MATTIA PANZAVOLTA

Co-relatore

Dott. CIRO BARBONE

Prima Sessione di Laurea

Anno Accademico 2019-2020

Indice

Indice delle figure

Capitolo 1
Introduzione

1.1 Scenario

Il mondo al giorno d'oggi si trova all'interno del periodo della digitalizzazione, che sta lentamente acquisendo un ruolo da protagonista nella vita di tutti noi, sia in ambito lavorativo che privato; ciò è evidente osservando come quotidianamente vengano utilizzati computer, telefoni e tablet dalla maggior parte delle persone, creando una vera e propria dipendenza. Molti settori si sono dovuti adattare per rimanere al passo coi tempi, come per esempio il commercio che con l'ausilio di Internet è divenuto un commercio di tipo elettronico; lo stesso vale anche per l'ambiente scolastico, dove vengono messe a disposizione diverse tecnologie per l'utilizzo didattico da parte di studenti e insegnanti, come per esempio lavagne interattive multimediali, videoproiettori o connessione Wi-Fi. Si è arrivati a un punto dove anche il cartaceo sta iniziando a scomparire per lasciare spazio al digitale, ad esempio il passaggio dai libri alle slide caricate dai docenti sul Web (o ai libri in formato elettronico) o dallo svolgimento di esami scritti su fogli protocollo a esami svolti al computer; focalizzandoci su quest'ultimo punto, una delle maggiori preoccupazioni da parte dei docenti è sempre stata quella di impedire che gli studenti copiassero durante lo svolgimento di una prova scritta. Essi, infatti, si sono sempre dimostrati molti creativi nello sviluppo di nuovi metodi per copiare, senza contare l'utilizzo di alcuni classici come i bigliettini. Durante un esame al computer gli viene dato un metodo aggiuntivo per imbrogliare: Internet; ciò non è tuttavia una novità, infatti già da tempo gli studenti fanno uso di cellulari per poter copiare. All'interno dei laboratori del nuovo Campus universitario di Cesena la soluzione è stata quella di utilizzare un firewall per impedire l'accesso al web per un determinato periodo di tempo; l'unica nota negativa è che ogni volta che deve essere svolto un esame è necessario l'intervento di un tecnico per la configurazione del firewall. Questa tesi si concentra sullo sviluppo di una soluzione a tale problema.

1.2 Obiettivi

I computer all'interno dei laboratori informatici sono connessi tramite cavo di rete alle porte degli switch, a cui poi vengono assegnate una specifica VLAN; ad ogni laboratorio viene associata una differente VLAN; mediante l'apposito uso di switch (livello 2 ISO/OSI) e router (livello 3 ISO/OSI) il traffico viene fatta passare attraverso l'unità FortiGate, connessa alla rete esterna, che si occupa di filtrare le richieste in entrata e in uscita mediante l'utilizzo di un firewall. L'obbiettivo di questa tesi si focalizza sullo sviluppo di un'applicazione tramite Rest API che permetta di collegarsi a FortiGate e abilitare il firewall da remoto mediante l'uso di un'interfaccia Web semplice, comoda e intuibile collocata all'interno dell'applicazione PADFY, così che possa essere utilizzata dai docenti, senza dover richiedere l'intervento da parte dei tecnici informatici. Così facendo si ha la impedire il compimento di alcune azioni fraudolente da parte degli studenti durante una prova d'esame svolta all'interno dei laboratori informatici, quali:

- Accesso alla rete Internet.

- Comunicazione tra due computer all'interno della stessa rete.

Capitolo 2
Tecnologie utilizzate

2.1 Laravel

Laravel è un framework open source di tipo MVC (Model, View, Controller) scritto in linguaggio PHP per lo sviluppo di applicazioni web. Qui di seguito vengono elencate alcune delle sue caratteristiche:

- **Eloquent ORM (Object Relation Mapping):** ogni tabella all'interno del database possiede una corrispondente classe Model con la quale interagisce; tali classi permettono di richiedere dati dalle tabelle e di inserirne di nuovi.

- **Query Builder:** fornisce un'interfaccia conveniente per creare ed eseguire le query sulle tabelle di un database.

- **Migrations**: permettono di creare, modificare e condividere le tabelle di un database.

- **Views:** contengono la parte in HTML e separano la logica del controller e dell'applicazione da quella di presentazione.

- **Blade:** template che permette di utilizzare codice PHP all'interno delle view; esse sono compilate in un semplice codice PHP e memorizzate nella cache fino a quando non vengono modificate, il che significa che aggiungono sostanzialmente zero overhead all' applicazione.

- **Authorization**: sono i modi per autorizzare le possibili azioni degli utenti rispetto ad una data risorsa tramite gates e policies: le prime permettono di definire chi può fare cosa, andando a registrare le varie abilità dell'utente; le seconde, invece, hanno la responsabilità di controllare le abilità sul model, infatti per ogni classe del model si avrà una corrispondente classe policy.

- **Authentication**: definisce la logica per il login, la registrazione e il recupero della password dell'utente.

- **Routing**: gestione delle URL, mappatura tra esse e le funzionalità offerte dall'applicazione stessa.

- **Middleware**: forniscono un meccanismo conveniente per filtrare le richieste HTTP che vengono inviate all'applicazione (per esempio, per verificare se un'utente è loggato).

2.2 cURL

cURL (Client URL) è un progetto software che fornisce una libreria (libcurl) e uno strumento a linea di comando (curl) che permette di connettersi, comunicare e trasferire dati a diversi tipi di server mediante l'uso di vari protocolli. cURL, se viene specificato un protocollo sicuro (come per esempio HTTPS), connettendosi ad un server remoto ne ottiene il certificato e confrontandolo con il suo archivio di certificati CA (certification authority) ne verifica la sua validità per assicurarsi che il server sia chi affermi di essere; se invece utilizza un certificato autofirmato o che non è stato firmato da una CA valida, allora cURL restituirà un messaggio d'errore.

2.3 REST

Representational State Transfer (REST) è uno stile architetturale software per i sistemi distribuiti che definisce un insieme di vincoli per la creazione di servizi Web. Un sistema RESTful viene definito da sei vincoli che limitano i modi in cui il server può processare e rispondere alle richieste del server, ottenendo alcune proprietà vantaggiose, quali performance, scalabilità, portabilità, semplicità, visibilità e affidabilità.

- **Uniform interface**. Un'interfaccia di comunicazione omogenea tra client e server permette di semplificare e disaccoppiare l'architettura, permettendo ad ogni sua parte di potersi evolvere indipendentemente.

- **Architettura client–server:** Un insieme di interfacce uniformi separa i client dai server. Questa separazione di ruoli significa che, per esempio, il client non si deve preoccupare del salvataggio delle informazioni, che rimangono all'interno di ogni singolo server, in questo modo la portabilità del codice del client ne trae vantaggio. I server non si devono preoccupare dell'interfaccia grafica o dello stato dell'utente, in questo modo i server sono più semplici e maggiormente scalabili. Server e client possono essere sostituiti e sviluppati indipendentemente fintanto che l'interfaccia non viene modificata.

- **Stateless:** La comunicazione client–server è ulteriormente vincolata in modo che nessun contesto client venga memorizzato sul server tra le richieste. Ogni richiesta da ogni client contiene tutte le informazioni necessarie per richiedere il servizio, e lo stato della sessione è contenuto sul client. Lo stato della sessione può anche essere trasferito al server attraverso un altro servizio posto a persistere, ad esempio un database.

- **Cacheable:** Come nel World Wide Web, i client possono fare caching delle risposte, dove devono essere definite implicitamente o esplicitamente cacheable o no, in modo da prevenire che i client possano riusare stati vecchi o dati errati. Una gestione ben fatta della cache può ridurre o parzialmente eliminare le comunicazioni client server, migliorando scalabilità e performance.

- **Code on demand:** I server possono temporaneamente estendere o personalizzare le funzionalità del client trasferendo del codice eseguibile; ad esempio, questo può includere componenti compilati come Applet Java o linguaggi di scripting client side come JavaScript. "Code on demand" è l'unico vincolo opzionale per la definizione di un'architettura REST.

- **Layered system:** Un client non può dire se è connesso direttamente ad un server di livello più basso od intermedio; i server intermedi possono migliorare la scalabilità del sistema con load-balancing o con cache distribuite e possono anche offrire politiche di sicurezza.

Le API di un servizio Web, implementato utilizzando HTTP e i principi REST, che aderiscono a tali vincoli vengono chiamate RESTful API e sono definite da:

- Un URL base.

- Un tipo di formato per la rappresentazione dei dati (JSON, XML…).

- Un metodo HTTP:
    - **POST:** crea un nuovo elemento.
    - **GET:** restituisce l'elemento specificato.
    - **PUT:** modifica o sostituisce un elemento completamente.
    - **DELETE:** elimina un elemento.

2.4 PADFY

PADFY (Port Access Data network For You) è un'applicazione Web sviluppata con l'ausilio del framework Laravel che si occupa della gestione dei vari apparati di rete all'interno del nuovo campus dell'università di Cesena, accessibile sia da docenti che da tecnici.

2.5 FortiGate

Fortinet è una multinazionale americana che si occupa dello sviluppo e commercializzazione di prodotti e servizi per la sicurezza informatica, come firewall e antivirus. Il primo prodotto sviluppato dalla compagnia fu FortiGate, un firewall disponibile in varie forme e dimensioni per adattarsi a qualunque ambiente e fornire un'ampia gamma di funzioni di interfacciamento alla rete e di sicurezza all'avanguardia. Qui di seguito vengono riportate alcune sue caratteristiche e funzionalità:

- Creazione di reti basate sulla sicurezza e consolidano le funzionalità di sicurezza leader del settore come il sistema di prevenzione dalle intrusioni (IPS), il filtro Web, l'ispezione SSL (Secure Socket Layer) e la protezione automatica dalle minacce.

- Utilizzo dei laboratori FortiGuard basati sull'intelligenza artificiale (AI) per offrire una protezione proattiva contro le minacce tramite l'ispezione ad alte prestazioni del traffico di rete sia in chiaro che crittografato.

- Ispezione del traffico in entrata e in uscita dalla rete. Queste ispezioni avvengono a velocità, scala e prestazioni senza pari e impediscono qualsiasi cosa, dai ransomware agli attacchi DDoS, senza degradare l'esperienza dell'utente o creare costosi tempi di inattività.

Alcuni dei componenti che compongono FortiGate sono:

- **Policies:** sono uno dei maggiori aspetti del sistema; ci sono molti blocchi e configurazioni coinvolti nell'impostazione di un firewall e all'interno delle policies molti di questi componenti si uniscono per formare un'unità coesiva per svolgerne la funzione principale, ovvero di analizzare il traffico di rete e rispondere in modo appropriato ai risultati di quell'analisi. Ci sono differenti tipi di policies, ma nella maggior parte dei casi vengono suddivise in versione IPv4 e versione IPv6, come riportato qui di seguito:

    - **IPv4 Policy:** usato per gestire il traffico che passa attraverso il dispositivo usando i protocolli IPv4.

    - **IPv6 Policy:** usato per gestire il traffico che passa attraverso il dispositivo usando i protocolli IPv6.

    - **IPv4 access control list:** usato per filtrare pacchetti basandosi su specifici parametri IPv4.

    - **IPv6 access control list:** usato per filtrare pacchetti basandosi su specifici parametri IPv6.

    - **IPv4 DoS policy:** utilizzato per impedire a pacchetti dannosi o difettosi di negare l'accesso agli utenti su un'interfaccia IPv4.

    - **IPv6 DoS policy:** utilizzato per impedire a pacchetti dannosi o difettosi di negare l'accesso agli utenti su un'interfaccia IPv6.

- **Schedules**: controllano quando le policies sono attive; quando si aggiunge una policy di sicurezza su un'unità FortiGate, è necessario impostare una pianificazione per determinare l'intervallo di tempo in cui tale policy funzionerà; nel caso non venisse impostata rimarrebbe sempre in funzione a controllare il traffico che attraversa FortiGate; la componente temporale del programma si basa su una notazione di 24 ore. È inoltre possibile organizzare più schedules in un unico gruppo in modo da ridurre la lista delle policies, questo perché non è possibile associare la stessa policy a più schedules. Esistono due tipi di pianificazioni:

    - **One-Time:** sono in vigore una sola volta per il periodo di tempo specificato nella pianificazione. Ciò può essere utile durante i test di prova per limitare la durata della policy nel caso in cui non venga rimossa, oppure può essere utilizzata per eventi isolati come una conferenza o un esame in cui è necessario modificare temporaneamente l'infrastruttura. L'intervallo di tempo per una pianificazione singola viene configurato utilizzando un'ora di inizio che include Anno, Mese, Giorno, Ora, Minuti e un tempo di arresto che include le stesse variabili. Quindi, mentre la frequenza del programma è solo una volta, la durata può variare da 1 minuto a più anni.

- **Recurring:** sono in vigore ripetutamente a orari specifici di determinati giorni della settimana. Il programma ricorrente si basa su un ciclo ripetuto dei giorni della settimana anziché ogni x giorni o giorni del mese, ciò significa che è possibile configurare la pianificazione in modo che diventi effettiva martedì, giovedì e sabato, ma non ogni due giorni o in giorni dispari del mese. Se una pianificazione ricorrente ha un'ora di fine precedente all'ora di inizio, la pianificazione avrà effetto all'ora di inizio ma terminerà all'ora di arresto il giorno successivo; è possibile utilizzare questa tecnica per creare pianificazioni ricorrenti che vanno da un giorno all'altro.

2.6 phpMyAdmin

phpMyAdmin è un’applicazione Web scritta in linguaggio PHP che supporta un’ampia gamma di operazioni per la gestione di un database di tipo MySQL o MariaDB.

Capitolo 3
VLAN e FortiGate

3.1 LAN / VLAN

Una LAN (Local Area Network) è un insieme di computer e dispositivi collegati tra di loro all'interno di un dominio di broadcast nella quale si scambiano pacchetti tra di loro. Le VLAN (Virtual LAN) utilizzano un ID per separare logicamente una LAN in domini di broadcast più piccoli, ottenendo i seguenti vantaggi:

- Riduzione del traffico di rete, in quanto i pacchetti vengono ricevuti da un numero minore di dispositivi.

- Aumento della sicurezza della rete; se per esempio un hacker dovesse riuscire ad ottenere l'accesso ad una VLAN, riuscirebbe a leggere soltalto una piccola parte del traffico di rete e non di tutta.

Gli switch di livello 2 e i router di livello 3 aggiungono un tag ID VLAN ai pacchetti che man mano arrivano e li rimuovono prima che vengano consegnati alla loro destinazione finale; i numeri ID VLAN validi sono compresi tra 1 e 4094, mentre lo 0 viene utilizzato per i pacchetti ad alta priorità e il 4095 è riservato.

L'unità FortiGate può operare in due diverse modalità:

- **Transparent:** il dispositivo si comporta come un bridge di livello 2, ma può comunque fornire servizi come scansione antivirus, filtro web, filtro spam e protezione dalle intrusioni al traffico di rete. Esistono alcune limitazioni in questa modalità in quanto non è possibile utilizzare SSL VPN, PPTP / L2TP VPN, DHCP o eseguire NAT.

- **NAT (Network Address Translation):** l'unità funziona come un dispositivo di livello 3. In questa modalità viene controllato il flusso di pacchetti tra varie VLAN, ma può anche rimuovere i tag VLAN dai pacchetti VLAN in arrivo. L'unità FortiGate può anche inoltrare pacchetti senza tag ad altre reti, come Internet.

3.2 Configurazione di criteri di sicurezza e routing

Le policies di sicurezza consentono la comunicazione tra le interfacce di rete dell'unità FortiGate in base agli indirizzi IP di origine e di destinazione; le interfacce che comunicano con la VLAN richiedono criteri di sicurezza per consentire il corretto passaggio del traffico tra di loro. Ogni VLAN necessita di una policy di sicurezza per ciascuna delle seguenti connessioni che essa andrà ad utilizzare:

- Da questa VLAN a una rete esterna.
- Da una rete esterna a questa VLAN.
- Da questa VLAN a un'altra VLAN nello stesso dominio virtuale sull'unità FortiGate.
- Da un'altra VLAN a questa VLAN nello stesso dominio virtuale sull'unità FortiGate.

I pacchetti su ciascuna VLAN sono soggetti a scansioni antivirus mentre attraversano l'unità FortiGate.

.

Capitolo 4
Analisi dei prerequisiti

4.1 Modalità classica di attivazione del Firewall

In questa prima parte del paragrafo viene illustrato il modo convenzionale di accedere a FortiGate per abilitare il firewall durante una prova di esame scritta in laboratorio:

- Collegamento a FortiGate tramite indirizzo IP e inserimento delle credenziali d'accesso.

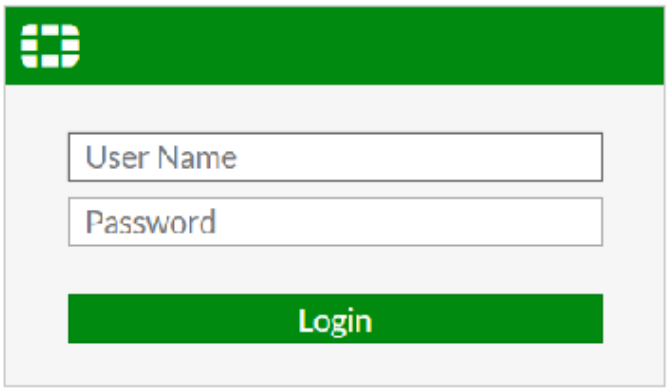

Figura 4.1: Schermata di login di FortiGate.

- Dal menu a tendina si seleziona "Schedules".

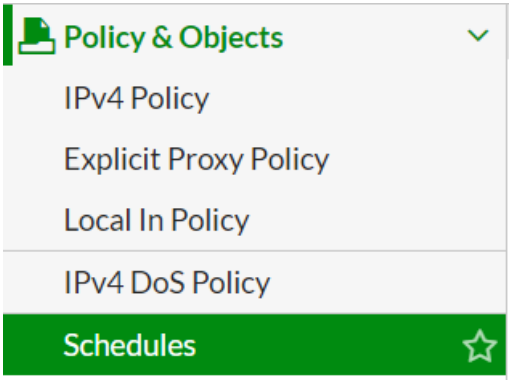

Figura 4.2: Menu di selezione "Policy & Objects".

- Di seguito compare una lista con tutti i nomi degli schedules raggruppati in Recurring e One-Time, con le rispettive data di inizio, data di fine e indice di riferimento al loro gruppo di appartenenza; da qui si seleziona quello di interesse.

Name	Days/Members	Start
Recurring (2)		
always	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	00:00
none	None	00:00
One-Time (50)		
2017-6-05-15-17-anvur-psice (expired)		2017/06/05 15:00
2017-6-06-10-12-anvur-psice (expired)		2017/06/06 10:00
Lab-Virt01-Rete73-Esami (expired)		2017/06/30 09:00
Lab-Virt01-Rete76-Esami (expired)		2014/09/15 10:30
Lab-arc-esami-cantar (expired)		2018/03/28 13:00
Lab2.2-manuale (expired)		2020/02/10 11:21
Lab3.1-manuale (expired)		2019/06/11 00:00
Lab3.3-manuale (expired)		2020/01/15 11:54
Lab4.2-manuale (expired)		2020/01/15 13:13
LabSA-manuale (expired)		2020/04/08 15:53
LabSATesi-manuale (expired)		2020/04/22 11:08
agraria-fabbri-2-9-2020 (expired)		2019/09/02 09:50
agraria-fabbri-4-6-2020 (expired)		2019/06/04 09:50
agraria-fabbri-7-7-2020 (expired)		2019/07/07 09:50
agraria-fabbri-17-9-2020 (expired)		2019/09/17 09:50
agraria-fabbri-19-6-2020 (expired)		2019/06/19 09:50
agraria-fabbri-20-9-2019 (expired)		2019/09/20 09:50
agraria-fabbri-21-2-2020 (expired)		2020/02/21 09:50
agraria-fabbri-22-7-2020 (expired)		2019/07/22 09:50
agraria-ragni-contesi-18-2-2020 (expired)		2020/02/18 08:40
danesi-agraria-14-02-2020 (expired)		2020/02/14 09:30
lab2.2-chiari-18-02-2020 (expired)		2020/02/18 13:55

Figura 4.3: Lista degli schedules.

- Dalla seguente schermata è possibile modificare le impostazioni dello schedule e una volta terminato dare l'OK; le policies entreranno in vigore al momento della data di inizio fino a quella di fine senza ulteriore interventi esterni (a meno che non si decida di terminarlo prima, in quel caso basterà modificare la data di fine in modo antecedente a quella attuale).

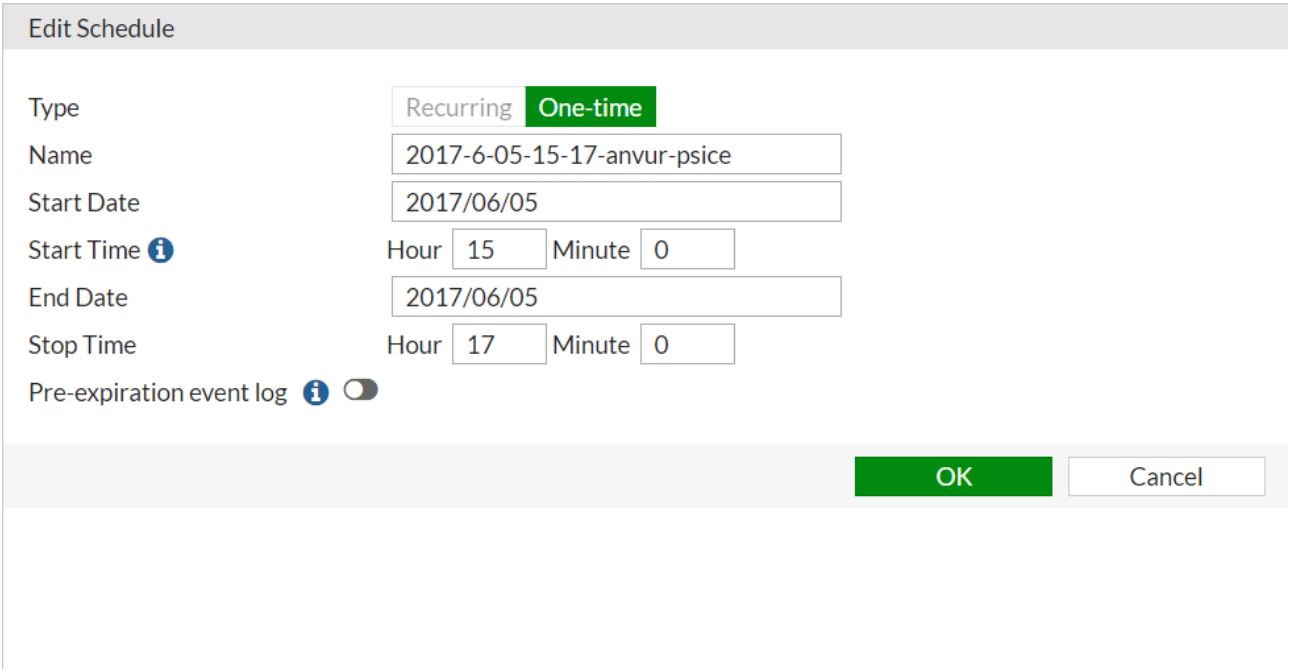

Figura 4.4: Impostazioni dello schedule.

4.2 Analisi delle richieste HTTP

Nel paragrafo precedente sono stati mostrati i passi che vengono eseguiti per l'attivazione del firewall; per la creazione della nostra applicazione è necessario convertirli in codice PHP, andando ad analizzare le richieste HTTP che vengono inviate al server; il modo più semplice per farlo è quello di utilizzare il Chrome DevTools, un set di strumenti per sviluppatori web integrato all'interno del browser Google Chrome; per accedervi è possibile cliccare col tasto destro del mouse su una pagina e selezionare "Ispeziona" (o alternativamente cliccare il tasto F12 della tastiera). L'interfaccia si compone di diversi pannelli:

- **Device:** simula dispositivi mobile.
- **Elements:** visualizza il codice HTML e le regole di stile (CSS) associati ad un determinato elemento.
- **Consoles:** visualizza messaggi ed esegui JavaScript dalla Console.
- **Sources**: esamina i dati memorizzati nei vari file, come i dati contenuti in sessione e nei cookie.
- **Network:** visualizza ed esegui il debug delle attività di rete.
- **Performance**: visualizza le prestazioni durante la fase di caricamento e durante l'esecuzione.
- **Memory**: utilizzo della memoria e rintracciamento delle perdite.
- **Application**: Ispeziona tutte le risorse caricate, inclusi database IndexedDB o Web SQL, archiviazione locale e di sessione, cookie, cache dell'applicazione, immagini, caratteri e fogli di stile.

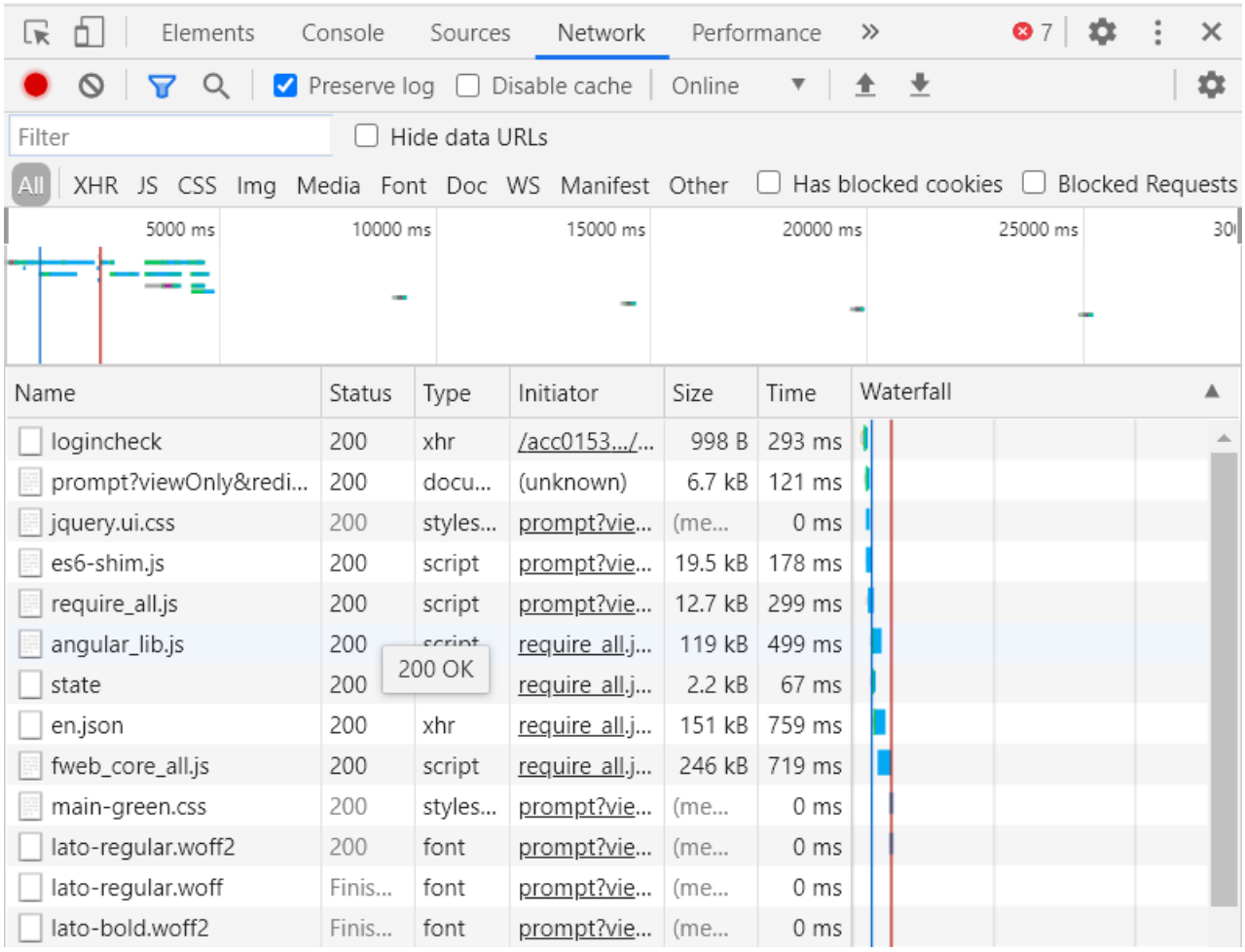

Figura 4.5: Pannello network del Chrome DevTools

Tramite il pannello Network possiamo analizzare tutte le risorse di rete che vengono scaricate quando una pagina Web viene caricata, elencate in ordine cronologico all'interno del registro di rete; generalmente la risorsa principale è la prima dell'elenco e aprendola ci vengono forniti diversi tipi di informazione utili all'interno del pannello Headers.

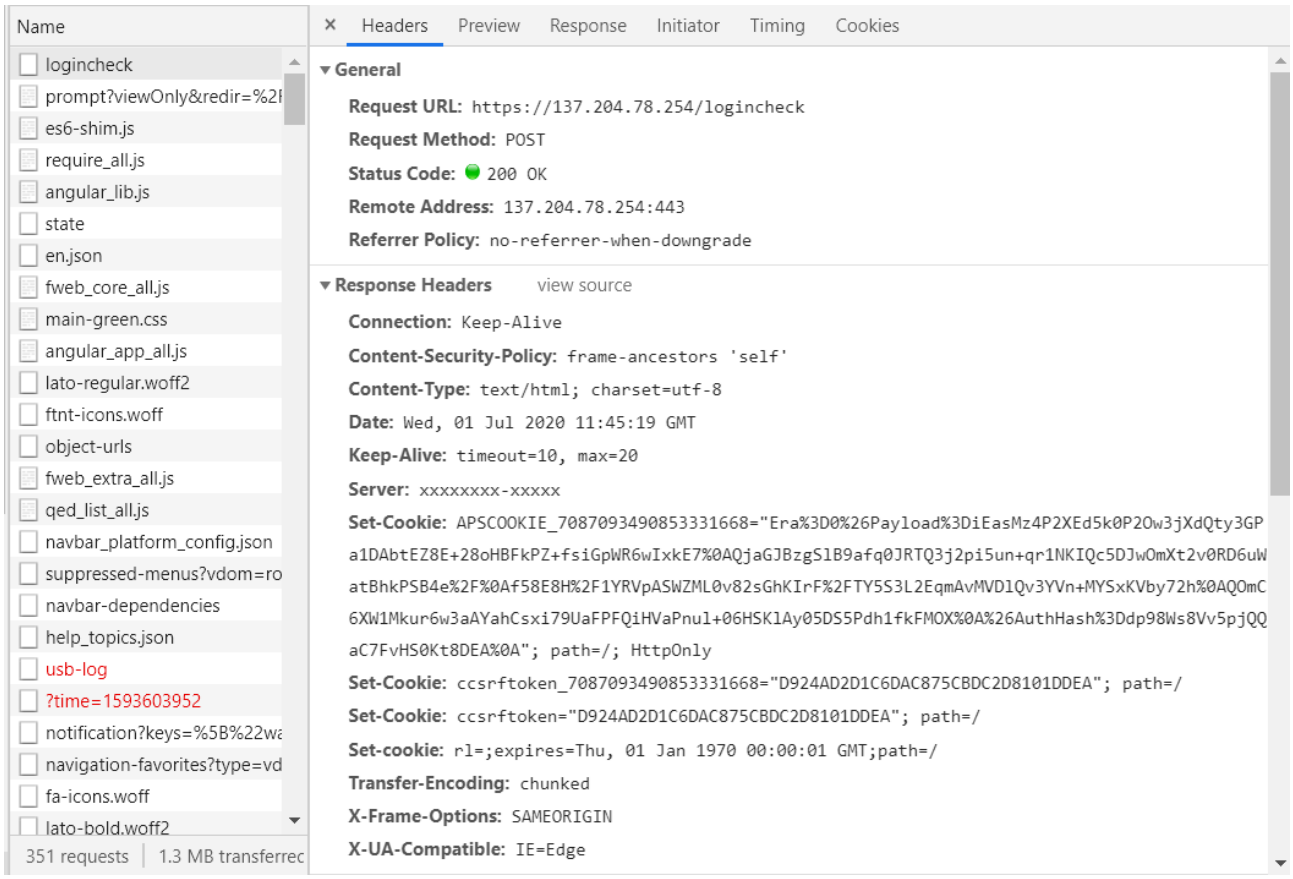

Figura 4.6: Pannello Headers della singola risorsa di rete

Tradizionalmente, per accedere da remoto a FortiGate si utilizza SSH (considerata durante lo sviluppo dell'applicazione come soluzione alternativa nel caso si fossero riscontrati problemi durante l'utilizzo di cURL), ma oltre ad essa viene messa a disposizione dei programmatori un'altra interfaccia, REST API.

Capitolo 5
Sviluppo dell’applicazione

5.1 Implementazione REST API

La prima parte dello sviluppo dell’applicazione consiste nella creazione di un metodo che permetta di connettersi da remoto a FortiGate con cURL tramite REST API, andando ad implementare i quattro metodi HTTP (GET, POST, PUT, DELETE), ma solamente i primi due sono stati effettivamente realizzati, in quanto i due restanti non si sono rivelati necessari ai fini del progetto.

Per gestire una sessione cURL verso un server remoto si utilizzano le funzioni riportate qui di seguito, chiamate nel seguente ordine:

- **curl_init ([string $url = NULL]): resource** inizializza una sessione cURL; se la funzione va a buon fine restituisce un handle (variabile puntatore), in grado di mantenere lo stato tra le varie richieste e utilizzato dai metodi successivi, mentre in caso di errore ritorna FALSE.

- **curl_setopt (resource $ch , int $option , mixed $value) : bool** permette di impostare un’opzione per la sessione indicata.

- **curl_exec (resource $ch) : mixed** esegue la sessione cURL specificata; questa funzione dovrebbe essere chiamata dopo averne impostato tutte le opzioni.

- **curl_close (resource $ch) : void** termina la sessione e libera tutte le risorse, inoltre l’handle viene eliminato.

La parte più delicata riguarda curl_setopt, dove le opzioni utilizzate per la connessione da remoto a FortiGate devono rispecchiare il più possibile le richieste HTTP che vengono inviate durante la normale navigazione all’interno del browser:

- **CURLOPT_USERAGENT:** permette ai server di identificare il sistema operativo, il fornitore e la versione dell’user-agent (applicazione) richiedente.

- **CURLOPT_ENCODING:** consente la decodifica della risposta. Se viene usata una stringa vuota "", allora viene inviata un'intestazione contenente tutti i tipi di codifica supportati.

- **CURLOPT_SSL_VERIFYHOST:** se impostato a TRUE, verifica che l'host con cui stai parlando sia l'host indicato dal certificato.

- **CURLOPT_SSL_VERIFYPEER:** se impostato a TRUE, verifica se il certificato SSL

- **CURLOPT_AUTOREFERER:** se viene impostato a TRUE, allora si occupa automaticamente di impostare il referer (indirizzo URL che viene controllato quando si accede ad una nuova pagina Web per vedere da dove si è originata la richiesta) mentre viene eseguito un reindirizzamento.

- **CURLOPT_HEADER:** viene incluso l'header nel risultato di output.

- **CURLOPT_RETURNTRANSFER:** se impostato a TRUE restituire il trasferimento come stringa del valore restituito di curl_exec().

- **CURLOPT FOLLOWLOCATION:** il server più eseguire uno o più reindirizzamenti verso altre pagine; impostando questo parametro a TRUE gli diciamo di seguirli invece di fermarci.

- **CURLOPT COOKIESESSION:** se impostato a TRUE va a sovrascrivere i cookie salvati all'interno del file ad ogni richiesta cURL.

- **CURLOPT COOKIEFILE:** utilizza i cookies salvati all'interno del file per la sessione cURL.

- **CURLOPT COOKIEJAR:** salva i cookies all'interno di un file una volta terminata la sessione cURL

```php
curl_setopt($curl_connection, CURLOPT_HTTPHEADER, $headers);
curl_setopt($curl_connection, CURLOPT_USERAGENT, 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
curl_setopt($curl_connection, CURLOPT_ENCODING, '');
//FALSE to stop cURL from verifying the peer and host certificate
curl_setopt($curl_connection, CURLOPT_SSL_VERIFYHOST, FALSE);
curl_setopt($curl_connection, CURLOPT_SSL_VERIFYPEER, FALSE);
curl_setopt($curl_connection, CURLOPT_AUTOREFERER, TRUE);
//TRUE to include the header in the output.
curl_setopt($curl_connection, CURLOPT_HEADER, TRUE);
curl_setopt($curl_connection, CURLOPT_RETURNTRANSFER, TRUE);
//TRUE to follow all the redirect
curl_setopt($curl_connection, CURLOPT_FOLLOWLOCATION, TRUE);
//If set to TRUE it will rewrite the cookie file at every curl request
curl_setopt($curl_connection, CURLOPT_COOKIESESSION, FALSE);
//Cookiefile and cookiejar save and load cookies in a single file
curl_setopt($curl_connection, CURLOPT_COOKIEFILE, dirname(__FILE__) . '/cookie.txt');
curl_setopt($curl_connection, CURLOPT_COOKIEJAR, dirname(__FILE__) . '/cookie.txt');

$result = curl_exec($curl_connection);
if (curl_errno($curl_connection)) {
    echo 'Error:' . curl_error($curl_connection);
}

//print_r(curl_getinfo($curl_connection));
//print_r($result);

curl_close($curl_connection);

return $result;
```

Figura 5.1: Opzioni per la sessione cURL a FortiGate

- **CURLOPT_HTTPHEADER:** un array di campi di intestazione

```php
$headers = array();
$headers[] = 'Content-Type: application/json';
$headers[] = 'Pragma: no-cache';
$headers[] = 'Accept: application/json, text/plain';
$headers[] = 'Host: 137.204.78.254';
$headers[] = 'Origin: https://137.204.78.254';
$headers[] = 'Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7';
$headers[] = 'Content-Type: text/plain;charset=UTF-8';
$headers[] = 'Accept: */*';
$headers[] = 'Cache-Control: max-age=0, no-store, no-cache, must-revalidate';
$headers[] = 'Connection: keep-alive';
$headers[] = 'If-Modified-Since: Sat, 1 Jan 2000 00:00:00 GMT';
$headers[] = 'Content-type: application/x-www-form-urlencoded';
$headers[] = 'X-Requested-With: XMLHttpRequest';
$headers[] = 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;
if ($referer) {
    $headers[] = 'Referer: '. $referer .'';
}
```

Figura 5.2: Lista di Headers utilizzati durante la sessione cURL con FortiGate

A seconda del metodo HTTP usato per la richiesta a FortiGate si utilizzano delle opzioni di curl_setopt specifiche ad esso:

- **GET:** recupera informazioni dal server; non sono necessaria ulteriori aggiunte alla opzioni per la sessione cURL.

- **POST:** invia i dati al server, ma non prima di essere stati convertiti in una query con codifica URL dal metodo **http_build_query**. Utilizza:

    - **CURLOPT_POST:** esegue una regolare richiesta HTTP POST.

    - **CURLOPT_POSTFIELDS:** invia i dati al server come array con il nome del campo come chiave e i dati del campo come valore.

```php
$curl_connection = curl_init($url);

switch ($method)
{
    case "GET":
        curl_setopt($curl_connection, CURLOPT_NOBODY, TRUE);
        break;

    case "POST":
        curl_setopt($curl_connection, CURLOPT_POST, TRUE);
        if ($data) {
            $post_data = http_build_query($data);
            curl_setopt($curl_connection, CURLOPT_POSTFIELDS, $post_data);
        }
        curl_setopt($curl_connection, CURLOPT_NOBODY, FALSE);
        break;

    case "PUT":
        curl_setopt($curl_connection, CURLOPT_PUT, TRUE);
        break;
}
```

Figura 5.3: Implementazione REST API

La funzione così ottenuta, denominata **call_fortigate_api($method, $url, $data, $referer)**, accetta come parametri un metodo HTTP per indicare il tipo di richiesta da inviare a FortiGate, un URL che designa il server alla quale ci si vuole connettere, un array di dati da inviare all'interno della richiesta nel caso si utilizzi il metodo POST e il referer

5.2 Login da remoto

La funzione di login, denominata **fortigate_login(),** fa uso del metodo **call_fortigate_api($method, $url, $data, $referer)** appena creato passandogli il metodo HTTP POST, il corrispettivo indirizzo URL e le credenziali d'accesso (username e password); eseguendo la funzione con i seguenti parametri il server restituisce il codice di errore 401 Unauthorized, ovvero che l'autenticazione è possibile ma è fallita; andando poi ad analizzare le richieste HTTP durante una sessione col browser si è notato che oltre alle due credenziali d'accesso ne veniva richiesta anche una terza (ajax=1). Una volta corretto l'errore, la richiesta di autenticazione veniva accettata con conseguente restituzione da parte del server del codice di stato 200 OK e di alcuni cookies da essere utilizzati per successive richieste a FortiGate all'interno della stessa sessione cURL.

```php
public function fortigate_login()
{
    $fg_host = "137.204.78.254";
    $ajax = '1';
    $fg_user = 'testrestapi';
    $fg_pass = 't3$tR3st@p!';
    $logincheck_url = 'https://'.$fg_host.'/logincheck';

    $data = array('ajax' => $ajax, 'username' => $fg_user, 'secretkey' => $fg_pass);
    $logincheck_result = $this->call_fortigate_api("POST", $logincheck_url, $data, FALSE);

    return $logincheck_result;
}
```

Figura 5.4: Funzione per il login al server remoto FortiGate

5.3 Tabella degli schedules

Le successive funzioni da implementare riguardano l'attivazione e la disattivazione del firewall da remoto, ma prima è necessario creare una tabella (accessibile tramite phpMyAdmin) nella quale viene aggiunto un nuovo record ogni volta che un docente decide di abilitare il firewall tramite l'apposita interfaccia all'interno dell'applicazione Web PADFY. La tabella si compone dei seguenti campi:

- **Id:** chiave primaria per identificare il record.
- **User_id:** identificativo del docente che ha abilitato il firewall.
- **IP:** indirizzo IP della VLAN.
- **Name:** nome della stanza corrispondente all'indirizzo IP della VLAN.
- **Date_time_start:** data di attivazione del firewall.
- **Date_time_end:** data di disattivazione del firewall.
- **Status:** lo stato del firewall, che può essere attivo o inattivo
- **Created_at:** data di creazione del record.
- **Updated_at:** data di aggiornamento del record.

id	user_id	ip	name	date_time_start	date_time_end	status	created_at	updated_at
3	5	137.204.79.93	Lab4.2-manuale	2019-06-04 09:58:00	2019-06-04 09:59:00	Off	2019-06-04 09:58:54	2019-06-04 09:59:58
4	5	137.204.79.93	Lab4.2-manuale	2019-06-04 10:21:00	2019-06-04 10:49:00	Off	2019-06-04 10:21:25	2019-06-04 10:49:23
5	5	137.204.79.93	Lab4.2-manuale	2019-06-04 10:52:00	2019-06-04 10:52:00	Off	2019-06-04 10:52:08	2019-06-04 10:52:28
6	5	137.204.79.93	Lab4.2-manuale	2019-06-04 10:58:00	2019-06-04 11:01:00	Off	2019-06-04 10:58:53	2019-06-04 11:01:03
7	5	137.204.79.93	Lab4.2-manuale	2019-06-04 11:04:00	2019-06-04 11:04:00	Off	2019-06-04 11:04:08	2019-06-04 11:05:17
8	5	137.204.79.93	Lab4.2-manuale	2019-06-04 11:14:00	2019-06-04 11:16:00	Off	2019-06-04 11:14:05	2019-06-04 11:16:23
9	5	137.204.79.93	Lab4.2-manuale	2019-06-04 11:16:00	2019-06-04 11:24:00	Off	2019-06-04 11:16:43	2019-06-04 11:24:25
10	5	137.204.79.93	Lab4.2-manuale	2019-06-05 12:26:00	2019-06-05 12:30:00	Off	2019-06-05 12:26:30	2019-06-05 10:50:33
11	5	137.204.79.93	Lab4.2-manuale	2019-06-05 12:51:00	2019-06-05 12:52:00	Off	2019-06-05 12:51:01	2019-06-05 10:53:20
12	5	137.204.79.93	Lab4.2-manuale	2019-06-06 09:33:00	2019-06-06 09:35:00	Off	2019-06-06 09:33:42	2019-06-06 09:35:02
13	5	137.204.79.93	Lab4.2-manuale	2019-06-06 09:35:00	2019-06-06 09:35:00	Off	2019-06-06 09:35:48	2019-06-06 09:35:55
14	5	137.204.79.93	Lab4.2-manuale	2019-06-06 10:17:00	2019-06-06 10:17:00	Off	2019-06-06 10:17:11	2019-06-06 10:18:01
15	5	137.204.79.93	Lab4.2-manuale	2019-06-06 10:40:00	2019-06-06 10:41:00	Off	2019-06-06 10:40:59	2019-06-06 10:41:01
16	5	137.204.79.93	Lab4.2-manuale	2019-06-06 10:41:00	2019-06-06 10:41:00	Off	2019-06-06 10:41:15	2019-06-06 10:41:28
17	5	137.204.79.93	Lab4.2-manuale	2019-06-06 10:41:00	2019-06-06 10:41:00	Off	2019-06-06 10:41:36	2019-06-06 10:41:49
18	5	137.204.79.93	Lab4.2-manuale	2019-06-06 10:41:00	2019-06-06 10:42:00	Off	2019-06-06 10:41:57	2019-06-06 10:42:01
19	5	137.204.79.93	Lab4.2-manuale	2019-06-06 10:47:00	2019-06-06 10:48:00	Off	2019-06-06 10:47:22	2019-06-06 10:48:01
20	5	137.204.79.93	Lab4.2-manuale	2019-06-06 10:48:00	2019-06-06 10:48:00	Off	2019-06-06 10:48:06	2019-06-06 10:48:44
21	5	137.204.79.93	Lab4.2-manuale	2019-06-06 10:49:00	2019-06-06 10:50:00	Off	2019-06-06 10:49:29	2019-06-06 10:50:01
22	5	137.204.79.93	Lab4.2-manuale	2019-06-06 10:50:00	2019-06-06 10:50:00	Off	2019-06-06 10:50:05	2019-06-06 10:50:42

Figura 5.5: Tabella fortigate_schedules.

5.4 Interfaccia utente

L'interfaccia utente per il controllo del firewall è stata creata all'interno dell'applicazione PADFY mediante l'uso di un file Blade; aprendo lo specifico pannello viene visualizzato un menu a tendina in cui vengono mostrate solo i nomi delle stanze aventi lo stesso indirizzo IP del client da cui si accede, ottenuto mediante la funzione **get_source_ip()**.

```php
public function get_source_ip()
{
    // Function to get the client ip address
    $ip_address = $_SERVER['REMOTE_ADDR'];
    return $ip_address;
}
```

Figura 5.6: Funzione per ottenere l'indirizzo del client.

In una prima query vengono selezionati dalla tabella "groups" gli indirizzi IP validi (non nulli); successivamente vengono controllati a uno a uno con quello ottenuto mediante la funzione **get_source_ip()** e in caso di corrispondenza positiva viene salvato l'identificativo della stanza all'interno di un array; infine, tramite una seconda query si ricavano i nomi delle stanze a cui abbiamo accesso dal client per l'abilitazione del firewall.

```php
$list_valid_ip = DB::table('groups')
                    ->select('id', 'ip')
                    ->where('ip', '<>', '')
                    ->get();

foreach($list_valid_ip as $valid_ip)
{
    $ips = explode(",", $valid_ip->ip);
    foreach($ips as $ip)
    {
        if($ip == $my_ip)
        {
            $list_id[] = $valid_ip->id;
        }
    }
}
if (count($list_id)>0)
{
    $list_schedule_names = DB::table('groups')
                    ->select('id', 'schedule_name')
                    ->where('schedule_name', '<>', '')
                    ->whereIn('id', $list_id)
                    ->get();
```

Figura 5.7: Query di selezione dei nomi

Una volta selezionato un nome dal menu a tendina la pagina si aggiorna, mostrando due caselle per l'impostazione della durata del firewall: una per le ore e una per i minuti. Stabilito un tempo, è possibile premere il pulsante "Attiva", abilitando di conseguenza il firewall; contemporaneamente compaiono sulla pagina un timer, implementato tramite uno script, che indica il tempo rimanente alla disattivazione automatica del firewall (gestito da FortiGate) e un pulsante "Disattiva".

```
<script>
    // Set the date we're counting down to
    var countDownDate = new Date("<?php echo $date_time_end ?>").getTime();

    // Update the count down every 1 second
    var x = setInterval(function() {

        // Get todays date and time
        var now = new Date().getTime();

        // Find the distance between now and the count down date
        var distance = countDownDate - now;

        // Time calculations for days, hours, minutes and seconds
        var days = Math.floor(distance / (1000 * 60 * 60 * 24));
        var hours = Math.floor((distance % (1000 * 60 * 60 * 24)) / (1000 * 60 * 60));
        var minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 * 60));
        var seconds = Math.floor((distance % (1000 * 60)) / 1000);

        // Output the result in an element with id="timer"
        document.getElementById("timer").innerHTML = days + "d " + hours + "h " + minutes + "m " + seconds + "s ";

        // If the count down is over, write some text
        if (distance < 0) {
            clearInterval(x);
            document.getElementById("timer").innerHTML = "EXPIRED";
            document.getElementById('disable_form').action = "{{ url('timeout_firewall/') }}";
            document.getElementById('disable_button').textContent = 'Esci';
        }
    }, 1000);
</script>
```

Figura 5.8: Script utilizzato per il timer

Mentre il firewall è attivo sono possibili due azioni:

- Premere il pulsante "Disattiva" per disabilitarlo.

- Lasciare che scada il tempo, in quel caso il timer verrà sostituito dalla scritta "EXPIRED" e comparirà il bottone "Esci" per ritornare alla pagina precedente.

5.5 Attivazione e disattivazione firewall da remoto

Nel paragrafo precedente abbiamo visto come funziona l'interfaccia quando vengono premuti i pulsanti "Attiva" e "Disattiva", qui invece vengono spiegate le funzioni invocate alla pressione dei due bottoni. Il primo chiama il metodo **enable_firewall (Request $request)** a cui vengono passati il nome della stanza in cui vogliamo attivare il firewall e la durata in ore e minuti; per prima cosa viene controllato che FortiGate non sia già attivo andando a controllare lo stato all'interno della tabella degli schedules tramite una query.

```
$schedule = DB::table('fortigate_schedules')
                ->select('status', 'date_time_end')
                ->where('ip','=',$client_ip)
                ->where('name','=',$fg_name)
                ->first();
```

Figura 5.9: Query per il controllo dello stato dello schedule

In caso positivo vengono eseguite le seguenti funzioni in ordine per inviare i dati allo schedule e abilitare il firewall:

- **fortigate_login():** invio delle credenziali d'accesso alla pagina di login (Figura 4.1) e restituzione dei cookies da parte del server, che vengono salvati all'interno di un file di testo tramite l'opzione cURL CURLOPT COOKIEJAR; è importante che CURLOPT_COOKIESESSION abbia come valore FALSE, questo per impedire che i cookies vengano sovrascritti alla successiva richiesta cURL, in quanto vengono richiesti da FortiGate per poter accedere alla prossima pagina web; in caso contrario verrebbe restituito come codice di stato HTTP 403 Forbidden, questo perché pur avendo inviato dei dati corretti, il server rifiuta la nostra richiesta.

- **call_fortigate_api($method, $url, $data, $referer):** si usa come metodo HTTP GET e come URL quello in riferimento alla pagina con la lista degli schedules (Figura 4.3); in questo caso, una volta che l'esecuzione della sessione cURL è andata a buon fine, è necessario estrarre dal risultato il csrftoken (token utilizzato come misura di sicurezza contro gli attacchi CSFR) e salvarlo all'interno di una variabile, in quanto verrà utilizzato per la successiva richiesta cURL.

- **call_fortigate_api($method, $url, $data, $referer):** si usa come metodo HTTP POST per inviare i dati all'URL in riferimento alla pagina con le impostazioni dello schedule (Figura 4.4); se la richiesta cURL va a buon fine, allora il firewall viene attivato e ne viene salvato un record all'interno della tabella fortigate_schedules (Figura 5.4) con status attivo.

```
$login_result = $this->fortigate_login();
$csrf_token_result = $this->call_fortigate_api("GET", $scheduler_list_url, FALSE, FALSE);

preg_match_all('|Set-Cookie: csrftoken.+=(.*);|U', $csrf_token_result, $csrf_token_value);
$csrf_token_value = implode(';', $csrf_token_value[1]);

$data = array('csrfmiddlewaretoken' => $csrf_token_value, 'type' => $scheduler_type, 'name' => $fg_name,
              'color' => $scheduler_color, 'start_date' => $current_date, 'start_hour' => $current_hour,
              'start_min' => $current_min, 'stop_date' => $end_date, 'stop_hour' => $end_hour,
              'stop_min' => $end_min, 'expiration-days' => $scheduler_expiration, 'mkey' => $fg_name);
$enable_firewall_result = $this->call_fortigate_api("POST", $scheduler_target_url, $data, $scheduler_target_url);

DB::table('fortigate_schedules')->insert(
    ['user_id' => $user_id, 'ip' => $client_ip, 'name' => $fg_name, 'date_time_start' => $current_time,
       'date_time_end' => $end_time, 'status' => 'On',  'created_at' => \Carbon\Carbon::now(),
       'updated_at' =>  \Carbon\Carbon::now()]
);
```

Figura 5.10: Funzioni per l'attivazione del firewall

Il metodo per la disattivazione del firewall, **disable_firewall (Request $request)**, segue più o meno la stessa procedura, l'unica differenza sostanziale sta nell'ultima richiesta cURL, nella quale viene spedito come data di fine l'ora attuale e come data di inizio l'ora attuale meno un minuto, andando poi ad aggiornare lo stato e la data di fine all'interno della tabella fortigate_schedules.

5.6 Task Scheduling

Uno dei problemi sorti durante lo sviluppo dell'applicazione riguardava l'aggiornamento della tabella del database contenente i record degli schedules, infatti poteva accadere che lo stato del firewall non corrispondesse a quello attuale; la soluzione è stata quella di utilizzare Task Scheduling fornita da Laravel, nella quale è possibile definire un task che verrà eseguito periodicamente sul server. Nel nostro caso prendiamo in considerazione tutti gli schedules aventi status "On" (firewall attivo) e si va a confrontare la data corrente con quella di fine dello schedule, andando ad aggiornare lo status nel caso il tempo di attività del firewall sia già scaduto.

```php
protected function schedule(Schedule $schedule)
{
    // $schedule->command('inspire')
    //          ->hourly();

    $schedule->call(function() {
        date_default_timezone_set('Europe/Rome');
        $current_time = date('Y/m/d H:i');
        $active_schedules = DB::table('fortigate_schedules')
                                ->select('id', 'date_time_end')
                                ->where('status', '=', 'On')
                                ->get();

        foreach($active_schedules as $active_schedule)
        {
            if (strtotime($active_schedule->date_time_end) - strtotime($current_time) <=0)
            {
                DB::table('fortigate_schedules')
                        ->where('id', $active_schedule->id)
                        ->update(['status' => 'Off', 'updated_at' =>  \Carbon\Carbon::now()]
                );
            }
        }
    })->everyMinute();
}
```

Figura 5.11: Task Scheduling per l'aggiornamento del database

Capitolo 6
Conclusioni

Gli obbiettivi inizialmente prefissati per questa tesi sono stati portati a termine, rendendo accessibile ai docenti l'attivazione e la disattivazione del firewall dell'unità FortiGate da remoto tramite una comoda interfaccia web e isolando gli studenti durante una prova d'esame all'interno dei laboratori informatici, impendendo l'accesso alla rete Internet e la comunicazione tra computer all'interno della stessa rete; le persone che hanno potuto testare questa nuova modalità si sono dette soddisfatte, rendendo a tutti gli effetti lo sviluppo di questa applicazione un successo. Molte delle operazioni che normalmente verrebbero svolte manualmente da un tecnico sono effettuate automaticamente dal software tramite la pressione di pochi tasti, con l'ulteriore aggiunta di alcune funzionalità extra, come per esempio un timer per poter tenere sempre sott'occhio il tempo rimanente alla disattivazione automatica del firewall. Ogni volta che qualcuno abilita il firewall viene subito creato un record all'interno di una tabella (a cui è possibile accedervi tramite phpMyAdmin), così in caso di problemi è possibile rintracciarne la fonte d'origine e dare modo al tecnico di risolverlo prontamente; lo stato di uno schedule(attivo/disattivo) viene continuamente controllato, in modo da avere un database sempre aggiornato.

Capitolo 7
Bibliografia

[1] Documentazione Laravel: https://laravel.com/

[2] Documentazione PHP: https://www.php.net/

[3] Documentazione cURL: https://curl.haxx.se/

[4] Documentazione Fortinet: https://help.fortinet.com/fos60hlp/60/Content/FortiOS/fortiOS-HTML5-v2/Home.htm#

[5] Documentazione Chrome Web Tools: https://developers.google.com/web/tools

[6] Documentazione REST API: https://restfulapi.net/

[7] Gli impatti della digitalizzazione: https://www.innexhub.it/gli-impatti-della-digitalizzazione/

[8] La digitalizzazione nella scuola: http://anderswinst.it/la-digitalizzazione-nella-scuola/