------------------------------  
| INFRASTRUTTURA |  
------------------------------

L’infrastruttura pensata per il nostro progetto è avere tre macchine una con la Dashboard, una con CTFd e una con i Laboratori. La dashboard tramite un Tunnel SSH va a collegarsi con la macchina dei Laboratori, con all’interno di essa Docker installato. Quando un nuovo utente verrà creato sulla dashboard parte uno script che crea sulla macchina dei Laboratori una nuova network docker per l’utente ed inoltre su di essa istanzia un server VPN( OpenVPN) creando il certificato che poi l’utente lo potrà scaricare dalla dashboard nell’apposita sezione. Quando un utente avvia un laboratorio dalla dashboard sarà avviato sulla sua rete docker privata e solo lui tramite la sua VPN potrà accederci. Così facendo ogni utente ha un “ambiente vulnerabile” privato e non può interferire con gli ambienti degli altri utenti. Con CTFd la dashboard interagisce aggiungendo in automatico gli utenti, i laboratori, le flag e prendendo tutta la parte di statistiche, tutto questo avviene tramite le api.

Setup  
------------------------------  
| MACCHINA DOCKER |  
------------------------------  
  
Per tutto il progetto abbiamo utilizzato delle macchina EC2 AWS con ubuntu 18,  
qualsiasi macchina può essere utilizzata, l'importante è avere Docker, ssh installato  
e aprire le porte: (22 per ssh) (Aprire il range di porte 34444 - 36962 su UDP, queste sono tutte le potenziali porte che servono alle VPN per esporsi su internet, il range è modificabile nel file vpn\_manage.py sulla dashboard)  
  
  
- Installazione docker (<https://docs.docker.com/engine/install/)>  
- Dopo aver installato, bisogna far startare il demone Docker su una porta TCP (2375) (Porta di default comunque configurabile dal pannello admin della dashboard):

**sudo rm /var/run/docker.pid && sudo systemctl stop docker && sudo dockerd -H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock**

Questo comando va dato ad ogni riavvio di questa macchina  
  
  
------------------------------  
| CTFD |  
------------------------------

Si è deciso di utilizzare come piattaforma per le CTF, CTFd. Il link ufficiale della repo utilizzata è: <https://github.com/CTFd/CTF>, è stata messa su una macchina, con docker e docker-compose installato, facendola startare con il docker-compose. Abbiamo dovuto modificare quest’ultimo poiché dalla repo non lo buildava più dopo l’utltimo aggiornamento( settembre 2020), il problema è qualche conflitto con il container di python inserito, in allegato ci sarà il docker-compose alternativo modificato da noi. Il comando dato è:

**docker-compose.yml up -d**

il comando fa partire tre container, uno per il database, uno con redis e uno con la piattaforma sulla porta 8000.

Al primo avvio bisogna fare la configurazione iniziale, inserire la modalità se a Team o ad Utenti e creare l’utente admin. Successivamente bisogna generare il token per l’admin che andrà inserito sulla dashboard. Per generare il token bisogna andare in settings e poi su access token, è raccomandato inserire una data di scadenza molto lunga e una volta generato questo token bisogna salvarlo poiché viene visualizzato una volta sola.

FUNZIONAMENTO API

Per l’interazione tra la dashboard e la piattaforma di Ctfd abbiamo usato le API di CTFd costruendo con esse una parte del nostro applicativo middleware, il link alla documentazione di quest’ultime è: <https://docs.ctfd.io/docs/api/>

Queste API sono poco documentate e non tutte funzionano, il nostro lavoro è stato principalmente capire come poterle sfruttare e cosa passargli. Per scrivere una funzione GET bisogna scrivere un payload ={} e un

headers = {

'Authorization': 'Token token\_generato\_in\_precedenza',

'Content-Type': 'application/json' }

Poi va scritta la funzione che si vuole richiamare usando la seguente sintassi.

def funzione():

url = <http://url_ctfd:8000/api/v1/api_da_chiamare>

response = requests.request("GET", url, headers=headers, data = payload)

result = json.loads(response.text)

Mentre per le funzioni POST e PATCH la sintassi per la funzione e l’headers saranno sempre gli stessi, cambiando solo il tipo da GET a POST/PATCH in response, ma nel payload andranno inserti i parametri da passare che vengono richiesti dalle API, per un esempio di queste funzioni vedere il codice middleware.py della nostra dashboard.  
  
------------------------------  
| DASHBOARD |  
------------------------------  
  
Installazione  
  
La dashboard è fatta in python 3.6.9 e Django 2.2  
  
- Clone della repo github (<https://github.com/mapoetto/group2_CTFLab)>  
- pip3 install -r requirements.txt (Installazione requisiti delle libreria python)  
- installazione dell'enviroment docker (per le API)  
Nella repo è già fornito un database con i nostri laboratori già inseriti e con solo l’utente admin creato, usr: **admin** pass: **tryfindme** ,la password di questo utente può essere cambiata dal pannello admin nella sezione UTENTI, solo l’account di admin sarà lì mentre tutti gli altri account saranno nella sezione USERS. Se si utilizza il database con i nostri laboratori la configurazione a CTFd è stata già aggiunta per poter aggiungere i laboratori, quindi bisogna modificare quella esistente mentre quella per la connessione alla macchina Docker bisogna aggiungerla. Una volta aggiunte le due configurazioni andare su ogni laboratorio e cambiare la Categoria e cliccare su salva così che vengano aggiunti su CTFd.  
  
-se è necessario creare un database da capo e senza alcun dato:  
-- bisogna cancellare il database (rm db.sqlite3)  
-- cancellare tutti i file di migrazione in app/migrations (tranne \_\_init\_\_.py)  
-- commentare tutto il file config\_const.py (Durante il development è stato perso un file di migrazione, quindi nel rigenerarle successivamente da errore se non si commenta questo file)  
-- ricreare le migrazioni (python3 manage.py makemigrations)  
-- applicare le migrazioni (python3 manage.py migrate)  
-- cancellare i commenti aggiunti dal file config\_const.py  
  
- Creazione utente admin (python3 manage.py createsuperuser)  
- Avvio della dashboard (python3 manage.py runserver IP PORT) (nel file setting.py ci sono gli IP whitelisted su cui il server può startare, se ci saranno nuovo IP basta aggiunger nel file)  
- Configurare le tabelle per la connessione alla macchina docker nel pannello ADMIN (IP delle macchine Docker e CTFD, porte locali e remote, full path key della chiave per il Tunnel SSH e TOKEN già generato per l'API CTFd )  
  
Al primo login di un utente normale (non admin) la dashboard prova a fare il tunnel SSH verso la macchina Docker, se l'host docker  
non è aggiunto come HOST affidabile nella configurazione di SSH, sarà richiesta la conferma sul terminale per effettuare il tunnel  
Una volta effettuato il tunnel, refreshare la pagina di login.  
  
  
POSSIBILI PROBLEMI ED OTTIMIZZAZIONI  
  
- La macchina Docker è stata testata e sembra reggere almeno fino a due utenti collegati che eseguono rispettivamente 1 laboratorio  
(molto può dipendere dalla grandezza dell'immagine del laboratorio) (La macchina riesce a reggere comunque almeno 5 serverVPN e 1 laboratorio in contemporanea)  
  
- Ottimizzare il multithread:  
Quando un utente effettua il primo login, viene istanziato un nuovo thread che si occupa di parlare con la macchina docker per effettuare il setup della VPN, siccome è un operazione  
bloccante, si è scelto di fare quest'operazione su un nuovo thread. Problema: se l'utente effettua subito il logout, e poi di nuovo il login, viene istanziato un nuovo thread che  
va in concorrenza con quello precedente, poichè provano entrambi ad istanziare da 0 un serverVPN con lo stesso nome, entrambi ricevono errore dall'API di docker, e non avviene la  
creazione del serverVPN. Possibile soluzione: implementare un pool di thread gestito in base all'utente. Un thread viene istanziato, per lo stesso motivo, anche per la creazione  
dell'utente sulla piattaforma di CTFd, anche per questo thread vale lo stesso discorso.  
  
- La statistica degli argomenti studiati non è affidabile, perchè non è stato implementato un sistema per capire quando un argomento è realmente studiato. Infatti ora appena si  
clicka un argomento, il contatore aumenta sempre  
  
- Il grafico che mostra l'andamento delle submission, non mette in ordine cronologico i giorni. Bisognerebbe convertire la data in TIME (numero assoluto in secondi) e poi ordinare  
il dizionario.  
  
- Migliorare l'interfaccia admin inserendo maggiori spiegazioni dei relativi campi

- Migliorare la logica che stoppa in automatico i laboratori, poiché adesso allo scadere del tempo anche se l’utente sta svolgendo il laboratorio gli viene chiuso e deve ricominciare dall’inizio.

- Per inserire laboratori che prevedono reverse shell bisogna dare un comando all'interno del laboratorio per aggiungere una route ip  
comando:

**ip route add 192.168.255.0/24 via 172.19.0.2** (possibili ip dei client collegati in vpn VIA ip del serverVPN sulla rete docker dell’utente)

il comando è scriptabile ed inseribile all'avvio del container. script:

**export ipServerVPN=$(hostname -I | sed 's:[^.]\*$:2:');  
 ip route add 192.168.255.0/24 via $ipServerVPN**

- Attualmente il progetto di django è impostato in development mode, e non production. Per andare in production si deve configurare django per partire con un webserver come nginx o apache.

-Possibilità per ogni utente di poter rigenerare la password di CTFd generata all’iscrizione direttamente dalla dashboard, non è molto complicato bisogna aggiungere il pulsante lato frontend e lato backend bisogna scrivere una funzione POST passando nel payload l’id dell’utente e la nuova password richiamando la funzione già scritta get\_random\_password().