Semabox

Version 2.0.0

Le Quadro

févr. 16, 2023

Contents:

1 install **1**
- 1.1 install_enterprise module . . . 1
- 1.2 install_single_user module . . . 2

2 Semabox **5**
- 2.1 Semabox module . . . 5
- 2.2 Semabox_CLI module . . . 7

3 SemaAPI **9**
- 3.1 API module . . . 9

4 SemaOS **11**
- 4.1 etat_server module . . . 11
- 4.2 generation_UID module . . . 11
- 4.3 info_server module . . . 12
- 4.4 materiel_server module . . . 13
- 4.5 ping module . . . 14
- 4.6 restart_server module . . . 14
- 4.7 scan_other_servers module . . . 14
- 4.8 scan_port_other_servers module . . . 14
- 4.9 scan_servers module . . . 15
- 4.10 server_speedtest module . . . 15
- 4.11 update_code module . . . 16

5 Indices and tables **19**

Index des modules Python **21**

Index **23**

CHAPITRE 1

install

1.1 install_enterprise module

Le script effectue les actions suivantes :

Importe les modules Python nécessaires :

Modules

[] dns.query dns.update sys mysql.connector subprocess

Ajoute le chemin vers le dossier “Application” afin de pouvoir importer nos modules Python personnalisés.

Importe nos modules Python personnalisés :

Fonctions :

get_ip_address sous le nom ip get_hostname sous le nom hostname get_dns sous le nom dns get_version_semabox sous le nom version_semabox lire_fichier sous le nom uid

Définit une fonction addDnsRecord() pour ajouter un enregistrement DNS :

Variables :

domain : le nom de domaine auquel ajouter l’enregistrement ip_dns : l’adresse IP du serveur DNS auquel envoyer la requête host : le nom de l’hôte à ajouter new_ip : l’adresse IP de l’hôte à ajouter enregistrement : le type d’enregistrement (A, AAAA, etc.) ttl : le temps de vie (en secondes) de l’enregistrement

Définit une fonction addBddRecord() pour ajouter un enregistrement dans la table “box” de la base de données “semabox” sur le serveur MariaDB à l’adresse 192.168.150.250 :

Variables :

sema_id : l’ID de la semabox sema_hostname : le nom d’hôte de la semabox sema_ip : l’adresse IP de la semabox sema_dns : le DNS de la semabox sema_version : la version de la semabox user : le nom d’utilisateur à utiliser pour la connexion à la base de données password : le mot de passe à utiliser pour la connexion à la base de données host : le nom d’hôte/IP du serveur de base de données database : la base de données à laquelle se connecter

Appelle la fonction addDnsRecord() pour ajouter un enregistrement DNS pour la semabox.

Appelle la fonction addBddRecord() pour ajouter un enregistrement pour la semabox dans la table “box” de la base de données “semabox”.

install_enterprise.**add_bdd_record**(*sema_id*, *sema_hostname*, *sema_ip*, *sema_ip_public*, *sema_dns*, *sema_version*, *user*, *password*, *host*, *database*)

Description :
Cette fonction ajoute un enregistrement à la table “box” de la base de données “semabox” sur le serveur MariaDB à l’adresse spécifiée.

Args :
sema_id (str) : L’identifiant unique (UID) de l’installation de SemaOS. sema_hostname (str) : Le nom d’hôte de l’installation de SemaOS. sema_ip (str) : L’adresse IP de l’installation de SemaOS. sema_dns (str) : Le nom de domaine de l’installation de SemaOS. sema_version (str) : La version de SemaOS utilisée. user (str) : Le nom d’utilisateur pour se connecter au serveur MariaDB. password (str) : Le mot de passe pour se connecter au serveur MariaDB. host (str) : L’adresse du serveur MariaDB. database (str) : Le nom de la base de données à utiliser.

install_enterprise.**add_dns_record**(*domain*, *ip_dns*, *host*, *new_ip*, *enregistrement*, *ttl*)

Description :
Cette fonction ajoute un enregistrement DNS pour un nouvel hôte avec une adresse IP spécifiée dans le domaine spécifié.

Args :
domain (str) : Le nom de domaine auquel ajouter l’enregistrement DNS. ip_dns (str) : L’adresse IP du serveur DNS qui gère le domaine. new_host (str) : Le nom de l’hôte à ajouter. new_ip (str) : L’adresse IP de l’hôte à ajouter. enregistrement (str) : Le type d’enregistrement DNS à ajouter (par exemple “A”, “AAAA”, “CNAME”, etc.). ttl (int) : Le temps de vie de l’enregistrement DNS en secondes.

install_enterprise.**main**()

Description : Fonction principale du script
— Appelle de la fonction define_os() : Permet de savoir sous quel système d’exploitation est installé le serveur et donc
— Appelle de la fonction version_python() : Vérifie la version de Python.
— Appelle de la fonction define_permissions_linux() : Définit les permissions Linux.
— Appelle de la fonction pre_installation() : Exécute le script de génération d’un identifiant unique (UID) pour l’installation de SemaOS.
— Appelle de la fonction add_dns_record() : Ajoute un enregistrement DNS pour la semabox
— Appelle de la fonction add_bdd_record() : Insertion des données dans la base de données

Paramètres :
— les valeurs définies dans les fonctions peuvent être modifiées (adresse IP, nom de domaine, etc.)

install_enterprise.**pre_installation**()

Description :
Cette fonction exécute le script de génération d’un identifiant unique (UID) pour l’installation de SemaOS.

1.2 install_single_user module

install_single_user.**define_os**() → None

Description : Cette fonction permet de définir l’OS sur lequel le script est exécuté.
— Si le script est exécuté sur Linux, on lance le script de creation du service SemaWEB.
— Si le script est exécuté sur Windows, on lance le script de creation du service SemaWEB.

Paramètres : Aucun Retour : Aucun

install_single_user.**define_permissions_linux**() → None

Description : Cette fonction permet de définir les permissions d'exécution du script.

— Si le script est exécuté sur Linux, on vérifie si l'utilisateur a les permissions d'exécution.

— Si l'utilisateur a les permissions d'exécution, le script continue son exécution(Windows)

Paramètres : Aucun Retour : Pour Linux

— Si l'utilisateur n'a pas les permissions d'exécution, on demande à l'utilisateur de saisir son mot de passe pour lui accorder les permissions d'exécution.

— Si l'utilisateur a les permissions d'exécution, le script continue son exécution.

install_single_user.**install_nmap**() → None

Description : Cette fonction permet d'installer Nmap sur Linux et windows. Paramètres : Aucun Retour : Aucun

install_single_user.**version_python**() → None

Description : Cette fonction permet de vérifier la version de Python utilisée.

— Si la version de Python est 3.11.1, le script continue son exécution.

Sinon

— Si la version de Python est différente de 3.11.1 alors on installe la version 3.11.1 (Linux) ou on demande à l'utilisateur d'installer la version 3.11.1 (Windows

Paramètres : Aucun Retour : Aucun

CHAPITRE 2

Semabox

2.1 Semabox module

Description : Fichier principal de l'application Semabox

Ce script importe plusieurs modules Python nécessaires pour son fonctionnement. Il importe également des fonctions de modules personnalisés se trouvant dans le répertoire "Semabox/SemaOS". Il définit une classe "App" qui représente la fenêtre principale de l'application. Le constructeur de cette classe initialise diverses variables telles que l'identifiant unique du système, le nom d'hôte, l'adresse IP et le nom de domaine associé. Il configure également la fenêtre en lui attribuant un titre, une taille et un alignement au centre de l'écran. Ensuite, il crée des widgets tels que des labels, des boutons et un menu déroulant. Ces widgets sont placés dans la fenêtre et sont configurés avec des polices de caractères, des couleurs de fond, du texte et d'autres paramètres. Le script termine en exécutant la boucle principale de Tkinter pour afficher la fenêtre et rendre possible l'interaction avec l'utilisateur.

`class` Semabox.**App**(*root*)

Bases : `object`

Description :

La classe App définit une application graphique utilisant l'interface utilisateur Tkinter. Le constructeur de la classe, __init__, prend en argument root, qui est la fenêtre principale de l'application. Le constructeur initialise plusieurs variables en appelant des fonctions de lecture de fichier, de récupération de nom d'hôte, d'adresse IP et de nom de domaine associé à l'adresse IP de l'hôte. Le constructeur configure également la fenêtre en définissant sa taille et son alignement au centre de l'écran, et en empêchant le redimensionnement par l'utilisateur. Enfin, le constructeur crée plusieurs widgets Tkinter pour afficher du texte et des couleurs de fond.

Button_clear_nmap_command()

Fonction : Button_clear_nmap_command

Description : Effacer les résultats du Scan Nmap

Button_scan_nmap_command()

Fonction : Button_scan_nmap_command

Description : Permets de lancer le scan nmap et d'afficher les résultats dans le label

Button_scan_nmap_machine()

Fonction : Button_scan_nmap_machine
Description : Permets de lancer le scan de machine et d'afficher les résultats dans le label

Button_scan_via_ip()

Fonction : Button_scan_via_ip
Description : Permets de lancer le scan de machine et d'afficher les résultats dans le label Return : None Paramètres : None

Button_speedtest_clear_command()

Fonction : Button_speedtest_clear_command
Description : Effacer les résultats du Speedtest

Button_speedtest_command()

Fonction : Button_speedtest_command
Description : Permets de lancer le speedtest et d'afficher les résultats dans les labels

about()

Fonction : about Description : Fonction qui permet d'afficher la version de l'application

def about(self) : crée une nouvelle méthode appelée about et lorsqu'elle est appelée, elle affiche une boîte de message d'information avec le message « A propos » et la version de Ma Semabox en utilisant la fonction get_version_semabox()

getInputBoxValue()

update_code()

Fonction : update_code Description : Fonction qui permet de vérifier si il y a une mise à jour de l'application

def update_code(self) : crée une nouvelle méthode appelée update_code et lorsqu'elle est appelée, elle affiche une boîte de message d'information avec le message « Update » et le status du code à l'aide de check_code_gitlab_application(get_latest_commit_date(os.getcwd()))

update_ping_latency()

Fonction : update_ping_latency Description : Fonction qui permet de mettre à jour le ping

- def update_ping_latency(self) : crée une nouvelle méthode appelée update_ping_latency et lorsqu'elle est appelée, elle met à jour le ping dans le label « Text_Label_Ping » et relance la fonction après 1 seconde
- self.ping = get_ping() : récupère le ping
- self.Text_Label_Ping[« text »] = f »PING : {self.ping} ms » : met à jour le texte du label « Text_Label_Ping » avec le ping
- root.after(1000, self.update_ping_latency) : relance la fonction après 1 seconde
- return : None

Return : None Paramètres : None

Semabox.verification_permission()

Fonction : verification_permission

Description
[Permets de vérifier si l'utilisateur est root]

- def verification_permission() : crée une nouvelle méthode appelée verification_permission
- with contextlib.suppress(Exception) : permet de supprimer les erreurs
- if platform.system() == « Linux » and os.getuid() != 0 : vérifie si l'utilisateur est root
- print(« Vous devez être root pour lancer cette application ») : affiche un message d'erreur

Return : None Paramètres : None

2.2 Semabox_CLI module

Description :

Ce script utilise la bibliothèque pyfiglet pour créer un titre décoré, la bibliothèque prettytable pour afficher les informations de manière structurée et la bibliothèque termcolor pour mettre en couleur les informations affichées. Il importe également des modules personnalisés pour récupérer les informations sur le serveur, comme cli_get_info_server, cli_get_info_system, cli_get_scan_nmap, cli_get_speedtest, cli_latence, cli_ping, scan_all_machine. Le script contient également plusieurs fonctions pour afficher les informations récupérées dans un menu structuré et coloré. Il utilise la fonction subprocess.run pour lancer des commandes système comme ping, nmap, speedtest. Il utilise également des concepts de threading pour lancer certaines fonctions en arrière-plan comme cli_latence. Il permet de scanner des machines dans un réseau local.

Semabox_CLI.**afficher_menu**()

Affiche un menu avec les différentes options disponibles

Semabox_CLI.**choix_menu**() → str

Demande à l'utilisateur s'il souhaite continuer et redirige vers le menu principal ou quitte le programme

Semabox_CLI.**choix_menu_scan**() → str

Affiche un menu avec les différentes options de scan disponibles

Semabox_CLI.**clear_screen**()

Efface l'écran

Semabox_CLI.**cli_info_server**() → None

Affiche les informations du serveur

Semabox_CLI.**cli_materiel**() → None

Affiche les informations sur le matériel

Semabox_CLI.**cli_scan**() → None

Affiche les informations sur le scan de ports de la machine

Semabox_CLI.**cli_scan_all_machine**() → None

Affiche les informations sur des scans de ports de machines dans un réseau local

Semabox_CLI.**cli_scan_port_machine**()

Scans les ports d'une machine donnée en entrée, affiche les résultats dans un tableau et demande à l'utilisateur s'il souhaite scanner une nouvelle machine

Semabox_CLI.**cli_speedtest**() → None

Affiche les informations sur le speedtest de la machine

Semabox_CLI.**menu**() → None

Affiche un menu avec les différentes options disponibles et permet à l'utilisateur de choisir une option pour effectuer une action spécifique.

Semabox_CLI.**verification_permission**()

CHAPITRE 3

SemaAPI

3.1 API module

Description de l'application Flask :

Ceci est une application Flask qui expose une API REST permettant d'exécuter des scripts Python et de retourner leur sortie sous forme de chaîne de caractères JSON.

L'API peut être accédée en utilisant trois méthodes HTTP : POST, GET et PUT. La fonction create_script exécute le script spécifié lorsqu'elle est accédée via une requête POST et retourne une représentation en chaîne de caractères JSON de la sortie du script.

La fonction get_script retourne la même représentation en chaîne de caractères JSON de la sortie du script lorsqu'elle est accédée via une requête GET.

La fonction update_script exécute le script spécifié lorsqu'elle est accédée via une requête PUT et retourne une représentation en chaîne de caractères JSON de la sortie du script. La sortie du script doit être un dictionnaire Python, sinon un message d'erreur sera retourné.

La fonction get_ip_address du module info_server est importée et utilisée dans les trois fonctions pour récupérer l'adresse IP du serveur.

API.**about**()

Description :
Cette fonction est un gestionnaire de route pour la page "A propos". Elle est appelée chaque fois qu'une demande est effectuée sur la page "A propos".

API.**add_script**(*ip*, *arg*)

Description de la fonction create_script :
Cette fonction exécute le script spécifié en utilisant subprocess.run et retourne une représentation en chaîne de caractères JSON de la sortie du script.

API.**after_request**(*response*)

C'est une fonction Flask en Python avec le décorateur @app.after_request qui s'exécute après chaque requête

HTTP. Il enregistre une chaîne contenant la méthode HTTP, l'URL et le code de statut de la réponse en utilisant la méthode info du journal associé à l'application Flask. Finalement, il retourne la réponse HTTP.

API.**catch_all**(*path*)

Cette fonction est un gestionnaire d'erreur pour l'erreur HTTP 404 (page non trouvée). Elle est appelée chaque fois qu'une demande entraîne une erreur 404.

API.**create_script**(*script*) → dict

Description de la fonction create_script :

Cette fonction exécute le script spécifié en utilisant subprocess.run et retourne une représentation en chaîne de caractères JSON de la sortie du script.

API.**index**()

Description de la fonction index :

Cette fonction affiche les informations relatives au serveur en exécutant le script "info_server.py" et en récupérant le dictionnaire de résultat. Si le dictionnaire est vide, une erreur HTTP 404 est retournée avec un message d'erreur personnalisé « Aucune information sur le serveur disponible » La page html "index.html" est ensuite rendue en utilisant les informations récupérées.

API.**page_not_found**(*error*)

Cette fonction est un gestionnaire d'erreur pour l'erreur HTTP 404 (page non trouvée). Elle est appelée chaque fois qu'une demande entraîne une erreur 404.

API.**page_not_found2**(*error*)

Cette fonction est un gestionnaire d'erreur pour l'erreur HTTP 500 (erreur interne du serveur). Elle est appelée chaque fois qu'une demande entraîne une erreur 500. Elle renvoie la réponse générée par le modèle de la page d'erreur HTTP 500, ainsi que le code d'erreur 500.

API.**page_not_found3**(*error*)

Cette fonction est un gestionnaire d'erreur pour l'erreur HTTP 400 (requête incorrecte). Elle est appelée chaque fois qu'une demande entraîne une erreur 400. Elle renvoie la réponse générée par le modèle de la page d'erreur HTTP 400, ainsi que le code d'erreur 400.

API.**page_not_found4**(*error*)

Cette fonction est un gestionnaire d'erreur pour l'erreur HTTP 501 (fonctionnalité non implémentée). Elle est appelée chaque fois qu'une demande entraîne une erreur 501.

API.**tools**()

La fonction tools implémente le point d'entrée "/tools" d'une application Flask et gère les requêtes GET et POST.

Pour les requêtes POST, elle met à jour les statuts speedtest_status et scan_status en fonction de la présence de "go" ou "reset" dans la requête.

Pour les requêtes GET, elle met à jour les statuts speedtest_status et scan_status en "reset".

Elle utilise les fonctions api_get_info_system, api_server_is_up, api_get_public_ip, api_web_speedtest et api_web_scan_nmap pour récupérer diverses informations sur le serveur. Si les informations récupérées ne sont pas valides (vides ou pas des objets dict), elle renvoie une erreur HTTP 404 avec un message d'erreur personnalisé.

Enfin, elle retourne une vue en utilisant le template "Pages/SemaWeb/tools.html" avec les informations récupérées sur le serveur.

CHAPITRE 4

SemaOS

4.1 etat_server module

etat_server.**api_server_is_up**(*host='192.168.1.14'*)

Description :
Cette fonction vérifie si le serveur est en ligne en envoyant une requête ping à l'adresse IP spécifiée.

Args :
host (str) : L'adresse IP du serveur à vérifier.

Returns :
bool : True si le serveur est en ligne, False sinon.

etat_server.**server_is_up**(*host='192.168.1.14'*)

Description :
Cette fonction vérifie si le serveur est en ligne en envoyant une requête ping à l'adresse IP spécifiée.

Args :
host (str) : L'adresse IP du serveur à vérifier.

Returns :
bool : True si le serveur est en ligne, False sinon.

4.2 generation_UID module

generation_UID.**check_file**()

Cette fonction vérifie si le fichier « UID.txt » existe dans le dossier « Semabox_UID ». Si le fichier n'existe pas, on appelle la fonction creation_dossier() avec un identifiant généré par generate_id(). Sinon, on appelle la fonction lire_fichier().

generation_UID.**creation_dossier**(*id_semabox*)

Cette fonction crée un dossier nommé « Semabox_UID » et y crée un fichier texte nommé « UID.txt » contenant l'identifiant passé en paramètre.

Paramètres :
— id_semabox (str) : identifiant à écrire dans le fichier « UID.txt ».

generation_UID.**generate_id**()

Cette fonction génère un identifiant unique (UUID) et le retourne sous forme de chaîne de caractères.

generation_UID.**lire_fichier**()

Cette fonction lit le fichier « UID.txt » dans le dossier « Semabox_UID » et retourne son contenu.

4.3 info_server module

Description :
Ce module contient des fonctions permettant de récupérer des informations sur le serveur sur lequel le code est exécuté. Les informations incluent le nom d'hôte, l'adresse IP, l'adresse IP publique, le nom de domaine, l'uid généré par generation_UID.py et la version de Semabox. Il contient également des fonctions pour redémarrer ou éteindre le serveur, et pour vérifier si le serveur est en ligne.

info_server.**api_get_public_ip**(*ip='90.48.187.251'*) → dict

info_server.**api_info_server**(*version='2.0.0'*, *lire_uid='328b2df8-1a59-4f2d-88f2-5375e037083c'*, *hostname='Alpha_Win'*, *ip='192.168.1.14'*, *dns='Alpha_Win.home'*, *ip_public='90.48.187.251'*) → dict

Description :
Cette fonction retourne un dictionnaire contenant des informations sur le serveur sur lequel le code est exécuté, telles que le nom d'hôte, l'adresse IP, l'adresse IP publique, le nom de domaine, l'uid généré par generation_UID.py et la version de Semabox.

info_server.**cli_get_info_server**(*version='2.0.0'*, *lire_uid='328b2df8-1a59-4f2d-88f2-5375e037083c'*, *hostname='Alpha_Win'*, *ip='192.168.1.14'*, *dns='Alpha_Win.home'*, *ip_public='90.48.187.251'*)

Description :
Cette fonction retourne un dictionnaire contenant les informations du serveur

Returns :
dict : informations du serveur

info_server.**get_address_network**(*ip='192.168.1.14'*) → str

Description :
Cette fonction récupère l'adresse de réseau à partir d'une adresse IP et d'un masque de sous-réseau

Args :
ip (str) : adresse IP

Returns :
str : adresse de réseau

info_server.**get_dns**(*ip*) → str

Cette fonction retourne le nom de domaine associé à l'adresse IP spécifiée.

Paramètres :
— ip (str) : adresse IP pour laquelle on souhaite récupérer le nom de domaine.

info_server.**get_hostname**() → str

Cette fonction retourne le nom d'hôte de la machine sur laquelle le code est exécuté.

info_server.**get_ip_address**() → str

Cette fonction retourne l'adresse IP de la machine sur laquelle le code est exécuté.

info_server.**get_public_ip**() → str

info_server.**get_version_semabox**() → str

Cette fonction retourne la version de SemaBox en lisant le fichier « version.txt » dans le répertoire « Application/modules ».

4.4 materiel_server module

Description : Ce module contient des fonctions qui permettent de récupérer des informations sur le serveur exécutant le script.

Importation des modules Python nécessaires :

psutil - Ce module permet de récupérer des informations sur le système, notamment sur les ressources utilisées (CPU, RAM, disques, etc.). platform - Ce module permet de récupérer des informations sur le système d'exploitation du serveur (nom, version, etc.).

Fonctions :

get_info_system() - Cette fonction retourne un dictionnaire contenant des informations sur le serveur exécutant le script. Les informations incluent le nombre de coeurs CPU, l'utilisation CPU, la quantité de mémoire RAM, le nombre de disques, le temps écoulé depuis l'allumage et le nom de l'OS. cli_get_info_system

materiel_server.**api_get_info_system**() → dict

Description :

Cette fonction retourne un dictionnaire contenant des informations sur le serveur exécutant le script. Les informations incluent le nombre de coeurs CPU, l'utilisation CPU, la quantité de mémoire RAM, le nombre de disques, le temps écoulé depuis l'allumage et le nom de l'OS.

Returns :

dict : Un dictionnaire contenant les informations sur le serveur.

materiel_server.**cli_get_info_system**() → dict

Description :

Cette fonction retourne un dictionnaire contenant des informations sur le serveur exécutant le script. Les informations incluent le nombre de coeurs CPU, l'utilisation CPU, la quantité de mémoire RAM, le nombre de disques, le temps écoulé depuis l'allumage et le nom de l'OS.

Returns :

dict : Un dictionnaire contenant les informations sur le serveur.

materiel_server.**get_info_system**() → dict

Description :

Cette fonction retourne un dictionnaire contenant des informations sur le serveur exécutant le script. Les informations incluent le nombre de coeurs CPU, l'utilisation CPU, la quantité de mémoire RAM, le nombre de disques, le temps écoulé depuis l'allumage et le nom de l'OS.

Returns :

dict : Un dictionnaire contenant les informations sur le serveur.

4.5 ping module

ping.**api_ping**()

ping.**cli_ping**()

ping.**get_ping**() → int

Envoie une requête ICMP et renvoie la durée du ping en millisecondes.

Returns :
int : Durée du ping en millisecondes.

Raises :
icmplib.exceptions.PingError : Si une erreur se produit lors de l'envoi de la requête ICMP.

4.6 restart_server module

restart_server.**redemarrer**() → dict

Description :
Cette fonction redémarre le serveur.

Returns :
str : Un message indiquant que le redémarrage du serveur est en cours.

4.7 scan_other_servers module

scan_other_servers.**api_scan_machine**(*network=IPv4Network('192.168.1.0/24')*) → dict

scan_other_servers.**cli_scan_machine**() → str

scan_other_servers.**scan_all_machine**(*network=IPv4Network('192.168.1.0/24')*) → dict

4.8 scan_port_other_servers module

scan_port_other_servers.**api_scan_port_other_machine**(*ip_args*) → dict

Description :
Cette fonction scanne les ports ouverts sur l'hôte local en utilisant l'outil nmap et retourne un dictionnaire contenant les informations sur les ports ouverts.

scan_port_other_servers.**scan_port_other_machine**(*ip_args*) → str

Description :
Cette fonction scanne les ports ouverts sur l'hôte local en utilisant l'outil nmap et retourne une chaîne de caractères contenant les informations sur les ports ouverts.

4.9 scan_servers module

Information :

Importation des modules Python nécessaires :
nmap - Ce module permet d'utiliser l'outil nmap en Python.

Importation des modules Python personnalisés :
info_server - Ce module contient la fonction get_ip_address() qui retourne l'adresse IP de l'hôte local.

Fonctions :
scan_nmap() - Cette fonction scanne les ports ouverts sur l'hôte local en utilisant l'outil nmap et retourne une chaîne de caractères contenant les informations sur les ports ouverts. api_scan_nmap() - Cette fonction scanne les ports ouverts sur l'hôte local en utilisant l'outil nmap et retourne un dictionnaire contenant les informations sur les ports ouverts.

scan_servers.**api_scan_nmap**() → dict

Description :
Cette fonction scanne les ports ouverts sur l'hôte local en utilisant l'outil nmap et retourne un dictionnaire contenant les informations sur les ports ouverts.

scan_servers.**api_web_scan_nmap**() → dict

Description :
Cette fonction scanne les ports ouverts sur l'hôte local en utilisant l'outil nmap et retourne un dictionnaire contenant les informations sur les ports ouverts.

scan_servers.**cli_get_scan_nmap**() → dict

Description :
Cette fonction scanne les ports ouverts sur l'hôte local en utilisant l'outil nmap et retourne un dictionnaire contenant les informations sur les ports ouverts.

scan_servers.**scan_nmap**() → str

4.10 server_speedtest module

server_speedtest.**api_speedtest**() → dict

Calcule la vitesse de téléchargement, d'envoi et de ping.

Returns :
dict : Dictionnaire contenant les vitesses de téléchargement, d'envoi et de ping.

server_speedtest.**api_web_speedtest**() → dict

Calcule la vitesse de téléchargement, d'envoi et de ping.

Returns :
dict : Dictionnaire contenant les vitesses de téléchargement, d'envoi et de ping.

server_speedtest.**cli_get_speedtest**() → dict

Calcule la vitesse de téléchargement, d'envoi et de ping.

Returns :
dict : Dictionnaire contenant les vitesses de téléchargement, d'envoi et de ping.

server_speedtest.**get_download_speed**(*test_file_url='http ://ipv4.download.thinkbroadband.com/50MB.zip'*) → str

Calcule la vitesse de téléchargement en mégabits par seconde.

Args :
test_file_url (str) : URL du fichier de test à télécharger.

Returns :
str : Vitesse de téléchargement en mégabits par seconde, formatée avec 2 chiffres après la virgule.

Raises :
requests.exceptions.RequestException : Si une erreur se produit lors du téléchargement du fichier de test.

server_speedtest.**get_ping**(*host='google.com'*) → int

Envoie une requête ICMP et renvoie la durée du ping en millisecondes.

Args :
host (str) : Hôte à ping.

Returns :
int : Durée du ping en millisecondes.

Raises :
icmplib.exceptions.PingError : Si une erreur se produit lors de l'envoi de la requête ICMP.

server_speedtest.**get_upload_speed**(*file_path='SemaOS\\download\\TESTMB.zip'*, *upload_url='https ://www.googleapis.com/upload/drive/v3/files ?uploadType=media'*) → str

Calcule la vitesse d'envoi en mégabits par seconde.

Args :
file_path (str) : Chemin d'accès au fichier à envoyer. upload_url (str) : URL de téléchargement.

Returns :
str : Vitesse d'envoi en mégabits par seconde, formatée avec 2 chiffres après la virgule.

Raises :
requests.exceptions.RequestException : Si une erreur se produit lors de l'envoi du fichier. FileNotFoundError : Si le fichier spécifié par *file_path* n'existe pas.

4.11 update_code module

update_code.**check_code_gitlab**() → dict

Description :
Cette fonction vérifie si le code du serveur est à jour.

Returns :
bool : True si le code est à jour, False sinon.

update_code.**check_code_gitlab_application**(*date*) → str

Description :
Cette fonction vérifie si le code du serveur est à jour.

Args :
date (datetime) : date du dernier commit

Returns :
str : message de statut de mise à jour

update_code.**get_latest_commit_date**(*repo_path*) → datetime

Description :
Cette fonction récupère la date du dernier commit dans un dépôt git.

Args :
repo_path (str) : Chemin du dépôt git.

Returns :
datetime : Date du dernier commit.

CHAPITRE 5

Indices and tables

— genindex
— modindex
— search

Index des modules Python

a

API, 9

e

etat_server, 11

g

generation_UID, 11

i

info_server, 12
install_enterprise, 1
install_single_user, 2

m

materiel_server, 13

p

ping, 14

r

restart_server, 14

s

scan_other_servers, 14
scan_port_other_servers, 14
scan_servers, 15
Semabox, 5
Semabox_CLI, 7
server_speedtest, 15

u

update_code, 16

Index

A

about() (*dans le module API*), 9
about() (*méthode Semabox.App*), 6
add_bdd_record() (*dans le module install_enterprise*), 1
add_dns_record() (*dans le module install_enterprise*), 2
add_script() (*dans le module API*), 9
afficher_menu() (*dans le module Semabox_CLI*), 7
after_request() (*dans le module API*), 9
API
    module, 9
api_get_info_system() (*dans le module materiel_server*), 13
api_get_public_ip() (*dans le module info_server*), 12
api_info_server() (*dans le module info_server*), 12
api_ping() (*dans le module ping*), 14
api_scan_machine() (*dans le module scan_other_servers*), 14
api_scan_nmap() (*dans le module scan_servers*), 15
api_scan_port_other_machine() (*dans le module scan_port_other_servers*), 14
api_server_is_up() (*dans le module etat_server*), 11
api_speedtest() (*dans le module server_speedtest*), 15
api_web_scan_nmap() (*dans le module scan_servers*), 15
api_web_speedtest() (*dans le module server_speedtest*), 15
App (*classe dans Semabox*), 5

B

Button_clear_nmap_command() (*méthode Semabox.App*), 5
Button_scan_nmap_command() (*méthode Semabox.App*), 5
Button_scan_nmap_machine() (*méthode Semabox.App*), 5
Button_scan_via_ip() (*méthode Semabox.App*), 6
Button_speedtest_clear_command() (*méthode Semabox.App*), 6
Button_speedtest_command() (*méthode Semabox.App*), 6

C

catch_all() (*dans le module API*), 10
check_code_gitlab() (*dans le module update_code*), 16
check_code_gitlab_application() (*dans le module update_code*), 16
check_file() (*dans le module generation_UID*), 11
choix_menu() (*dans le module Semabox_CLI*), 7
choix_menu_scan() (*dans le module Semabox_CLI*), 7
clear_screen() (*dans le module Semabox_CLI*), 7
cli_get_info_server() (*dans le module info_server*), 12
cli_get_info_system() (*dans le module materiel_server*), 13
cli_get_scan_nmap() (*dans le module scan_servers*), 15
cli_get_speedtest() (*dans le module server_speedtest*), 15
cli_info_server() (*dans le module Semabox_CLI*), 7
cli_materiel() (*dans le module Semabox_CLI*), 7
cli_ping() (*dans le module ping*), 14
cli_scan() (*dans le module Semabox_CLI*), 7
cli_scan_all_machine() (*dans le module Semabox_CLI*), 7
cli_scan_machine() (*dans le module scan_other_servers*), 14
cli_scan_port_machine() (*dans le module Semabox_CLI*), 7
cli_speedtest() (*dans le module Semabox_CLI*), 7
create_script() (*dans le module API*), 10
creation_dossier() (*dans le module generation_UID*), 11

D

define_os() (*dans le module install_single_user*), 2

define_permissions_linux() (*dans le module install_single_user*), 2

E

etat_server
    module, 11

G

generate_id() (*dans le module generation_UID*), 12
generation_UID
    module, 11
get_address_network() (*dans le module info_server*), 12
get_dns() (*dans le module info_server*), 12
get_download_speed() (*dans le module server_speedtest*), 15
get_hostname() (*dans le module info_server*), 12
get_info_system() (*dans le module materiel_server*), 13
get_ip_address() (*dans le module info_server*), 12
get_latest_commit_date() (*dans le module update_code*), 16
get_ping() (*dans le module ping*), 14
get_ping() (*dans le module server_speedtest*), 16
get_public_ip() (*dans le module info_server*), 13
get_upload_speed() (*dans le module server_speedtest*), 16
get_version_semabox() (*dans le module info_server*), 13
getInputBoxValue() (*méthode Semabox.App*), 6

I

index() (*dans le module API*), 10
info_server
    module, 12
install_enterprise
    module, 1
install_nmap() (*dans le module install_single_user*), 3
install_single_user
    module, 2

L

lire_fichier() (*dans le module generation_UID*), 12

M

main() (*dans le module install_enterprise*), 2
materiel_server
    module, 13
menu() (*dans le module Semabox_CLI*), 7
module
    API, 9
    etat_server, 11
    generation_UID, 11
    info_server, 12
    install_enterprise, 1
    install_single_user, 2
    materiel_server, 13
    ping, 14
    restart_server, 14
    scan_other_servers, 14
    scan_port_other_servers, 14
    scan_servers, 15
    Semabox, 5
    Semabox_CLI, 7
    server_speedtest, 15
    update_code, 16

P

page_not_found() (*dans le module API*), 10
page_not_found2() (*dans le module API*), 10
page_not_found3() (*dans le module API*), 10
page_not_found4() (*dans le module API*), 10
ping
    module, 14
pre_installation() (*dans le module install_enterprise*), 2

R

redemarrer() (*dans le module restart_server*), 14
restart_server
    module, 14

S

scan_all_machine() (*dans le module scan_other_servers*), 14
scan_nmap() (*dans le module scan_servers*), 15
scan_other_servers
    module, 14
scan_port_other_machine() (*dans le module scan_port_other_servers*), 14
scan_port_other_servers
    module, 14
scan_servers
    module, 15
Semabox
    module, 5
Semabox_CLI
    module, 7
server_is_up() (*dans le module etat_server*), 11
server_speedtest
    module, 15

T

tools() (*dans le module API*), 10

U

update_code

module, 16
update_code() (*méthode Semabox.App*), 6
update_ping_latency() (*méthode Semabox.App*), 6

V

verification_permission() (*dans le module Semabox*), 6
verification_permission() (*dans le module Semabox_CLI*), 7
version_python() (*dans le module install_single_user*), 3