Mini projet

Présentation générale

L'objectif de ce mini-projet et de réaliser une plate-forme logiciel de monitoring d'un parc informatique. Cette plate-forme, une fois installée, permettra à l'administrateur système que vous êtes de connaître rapidement à chaque fois que vous le demandez l'état détaillé de l'ensemble des serveurs faisant partis du parc et les dernières alertes du C.E.R.T. (http://www.cert.ssi.gouv.fr/). Le logiciel permettra aussi d'être prévenu par e-mail en cas de situation de crise. Une situation de crise étant par exemple : quand un serveur n'a pas donné signe de vie depuis plus de 30 minutes, quand un disque dur atteint 100% d'occupation ou lorsque la RAM est utilisée à 100% , etc …
Ce projet doit être réalisé en BASH et en Python, la proportion de chacun des langages dans le projet reste à votre appréciation. Vous pourrez utiliser l'ensemble de la librairie standard python et tous les binaires installés par défaut sur une distribution Ubuntu serveur. D'autres librairies python et bash peuvent être utilisées si elles sont mentionnées dans le sujet ou si vous avez l'accord explicite de votre tuteur. Nous maintiendrons une liste officielle des librairies utilisables sur le forum du cours sur e-uapv.

Rendu et Notation

Lorsque vous terminez une partie, vous la présentez à votre enseignant qui pourra vous la valider ou vous demander de la compléter. Lors de la dernière séance de TP, vous devez présenter l'ensemble de votre projet et vous n'aurez plus la possibilité de l'améliorer après cette présentation. Les notes attribuées à chaque partie sont précisées dans le sujet. Il vous sera demandé de nous rendre une archive avec l'intégralité de votre code source. Vous devez joindre également un README expliquant les options de fonctionnement de votre projet et les fonctionnalités que vous avez choisis de développer.

Timeline

Vous aurez 4 séances de TP pour faire le projet (avant la séance d'évaluation finale), et il est attendu de vous que vous travaillez également l'équivalent de 4 séances chez vous, soit au total 24h de travail. Voici certains repères, donnés à titre indicatif, pour vous aider à planifier votre travail :

- Séance 1 : Collecte d'information et stockage
- Séance 2 : Alerte
- Séance 3 : Affichage
- Séance 4 : Interface Web

N'oubliez de faire des backups de vos travaux, ce serait dommage d'avoir 0 parce que les données auraient été curieusement effacées la veille du rendu...

Description spécifique

Le logiciel sera découpé en quatre éléments indépendants mais interconnectés. De cette manière si une brique pose problème le logiciel peut toujours être un ensemble cohérent et fonctionnel. L'absence d'une brique ne sera donc pas bloquant lors de la poursuite du TP.

I. Collecte d'informations

Le premier élément est un collecteur (ou sonde) qui va être installée sur le serveur à surveiller. Le rôle de la sonde est de récupérer et afficher/envoyer à intervalles réguliers (toutes les 1 à 5 minutes) et de façon formatées des informations sur la machine monitorée.

Taches à réaliser :

- Mettre en place au moins trois sondes collectant un ensemble d'informations sur les utilisateurs. (le système : CPU / disque / RAM / nombre de process / nombre d'utilisateurs connectés, …) ***3 points***

Contraintes :

- Au moins une extraction d'informations en Bash
- Au moins une extraction d'informations en Python

Librairies :

- psutil (py)

II. Stockage et archivage

Le second élément est un gestionnaire de stockage et d'archivage qui est capable de stocker des informations, garder un historique de taille défini et de nettoyer les informations obsolètes. Cet outil devra aussi permettre un accès simple et exploitable aux informations qu'il contient.

Taches à réaliser :

- Un moteur de stockage de données avec gestion d'historique ***2 points***
- Un parseur web (http://www.cert.ssi.gouv.fr/) qui va récupérer la dernière alerte CERT et l'envoi au moteur de stockage ***2 points***

Contraintes :

- Les données doivent être stockées sur le disque dur
- Les données trop anciennes doivent être supprimées

Librairies :

- rrdtool (un outils Génial pour gérer les time série très puissant c'était un Standard dans l'industrie pendant les année !) (py et bash)
- sqlite (py et bash)
- (g)dbm (py et bash) (une Bdd NoSQL ancêtre du sql qui est ultra vielle et fonctionne sur tout les système)

2 Options au choix parmi : *2 points*

- L'utilisation d'un format standard de fichier pour le stockage (XML, JSON, …)
- L'utilisation d'une base de données sans serveur (sqlite, ...)
- Prévoir un cas de sauvegarde/restauration
- L'ajout d'une nouvelle sonde ne nécessite pas de modification manuelle du code ou de la base.

III. Affichage & Alerte

Le troisième élément est un outil de visualisation. Il permet de générer des graphiques d'évolution pour les données ayant un historique. Ce module intègre aussi des éléments de contrôles lui permettant de détecter une situation de crise et de prévenir l'administrateur par e-mail.

Taches à réaliser :

- Un module de détection de situation de crise *1 point*
- Un module d'envoi de mail *2 points*
- Un module de génération de graphiques *2 points*

Contraintes :

- Le module doit être capable de présenter les informations pour toutes les sondes réalisées
- Il affichera aussi les historiques sous forme de graphique

Librairies :

- rrdtool (python et bash)
- gnuplot
- pygal

2 Options au choix parmi : *2 points*

- Critères de la situation de crise configurables
- Taille maximum de l'historique configurable
- Contenu de l'e-mail paramétrable (fichier template)
- Envoie de mail via le serveur smtp de l'université

IV. Interface Web

Le quatrième élément est une interface de consultation web qui reprendra les informations des éléments développés dans la partie précédente avec des graphiques plus imagés et potentiellement un complément d'informations.

Taches à réaliser :

- Un module d'affichage web *3 points*
- Un readme d'utilisation *1 points*

Contraintes :

- Il affichera aussi les historiques sous forme de graphiques

Librairies :

- netcat (bash)

- httpie (bash)
- wget (bash)
- curl (bash)
- saxon (bash)
- mush (template enfin bash)
- flask (python)
- request (python)
- beautifulsoup4 (python)
- jinja (python)