Vampires - the requirements

Maxime Biais
Sebastien Henaff
Pierre Gleyzes
Nicolas Le Coz
Benoit Merouze

15 juin 2003

Table des matières

Chapitre 1

Introduction

Bonjour

Chapitre 2

La présentation de Vampire

2.1 L'utilité de Vampire

Vampire est un outils pour **tester à distance des programmes sur des "compiles farms"** (Usine à compilation) et en général sur n'importe quels architectures réseaux. Il sera écrit en **Python**.

Il prendra un programme sous forme de Tarball Autotools (ou en générale un programme avec différentes caractéristiques que nous définirons plus tard).

Ensuite, Vampire devra **copier** ces données sur les machines où il faudra **tester** ce programme avec **différentes contraintes que l'utilisateur devra imposer** de manière la plus simple et la plus explicite possible à Vampire.

Enfin Vampire doit nous fournir une synthèse pour **décrire le déroulement de ces tests**, et éventuellement les **diffuser** sous différentes formes.

2.2 La portée de Vampire

Vampire doit marcher sur n'importe quels « compiles farms » avec une configuration dans l'idéal identique. Il va de soi que l'utilisation de Vampire doit être indépendant de l'architecture réseau et devra faire face à tous les protocoles que le réseau testeur mettra à disposition de Vampire.

C'est pourquoi Vampire devra être **astucieux et intelligent** pour s'infiltrer sur des machines d'un réseau. Cependant Vampire ne doit **pas violer les contraintes de sécurité d'un réseau**. En fait si il y a de tels contraintes sur les accès à des "compiles farms", cela vient en partie de la politique de sécurisation du réseau.

Voici quelques situations possible pour Vampire face à des "compiles farms" (et autres machines d'un réseau quelconque) et comment il pourra d'une manière générale les contourner en respectant l'intégrité du réseau (qui nous fait la bonté de nous fournir des machines sur son réseau) :

XXXXXXXXXXXXXXXXXXXXXXXXX
AUDITS DE COMPILE FARM
XXXXXXXXXXXXXXXXXXXXXXXXX

2.3 Les capacités de Vampire

Comme nous l'avons indiqué précédemment, Vampire doit être le plus générale possible sans donné plus de travail aux utilisateurs de Vampire. Cependant **ces abstractions ne doit pas amoindrir la qualité et exhaustivité des résultats** que Vampire pourrait fournir.

De plus le résultat de Vampire devront être exportable sous différentes formes. Donc pour résumé il y a quatre grandes classes de difficulté qui peuvent être combiné mais ne doit pas pour autant être une contrainte pour les utilisateurs de Vampire :

- Le type du programme à tester (tarball Autotools, tarball ANT)
- Le type du réseau testeur avec ces protocoles et ses modes de connections (ftp, telnet, ssh, ?)
- Compatibilité des outils (de compilation ou d'exécution) demandés par le programme par rapport aux outils fournis pas les machines hôtes.
- Le format de sortie et le mode de diffusion (sur une page WEB, par messages, par mail, ?.)

Il y a aussi différentes options possible et éventuellement configurable par l'utilisateur :

Déploiement dans le temps : on lui indique une heure précise ou un moment optimale pour exécuter les requêtes. Donc on doit pourvoir déterminer le moment où le réseau est en mesure d' accueillir d'une façon honorable (dans les deux sens : Vampire ne doit pas baisser sensiblement les performances du réseau et aussi on doit pas respecter les délais qui peuvent être exiger pour effectuer nos requêtes).

Statistiques d'un réseau : cet option va être utile pour l'option précédente pour déterminer le « moment optimale ». On devra enregistrer à des actions données les performances associés.

Historique : On devra retracer toutes les utilisations de Vampire et dans l'idéale déterminer des classes de réseau et même déterminer les capacités et les performances d'un réseau vis à vis de Vampire (et pourquoi pas les diffuser à un public utilisateur de Vampire en respectant bien sur l'identité d'un réseau c'est à dire pas de délivrance de mot de passe, ?.).

Automatisation : il faudrait donner aussi la possibilité de grouper des programmes suivant des caractéristiques communes pour ainsi automatiser le testage en faisant des pools de programme soit en les planifiant, ou encore en les regroupant en un programme globale, et en indiquant à Vampire qu'il a différentes parties à tester (une partie est équivalente là à un programme). Bien sur la majorité du travail dans ce cas là est à la charge de Vampire.

Génération et aide à la configuration : On le constate toute suite qu'il y a beaucoup de contraintes à intégrer à Vampire pour l'exécution idéale et souhaitée par l'utilisateur (et aussi implicitement par le réseau hôte). Nous devons tout d'abord faire une documentation claire qui doit mette en évidence toutes les possibilités de Vampires en explicitant comment y arriver. De plus nous devons fournir les moyens pour configurer automatiquement suivant quelques données connues les fichiers de configuration (un réseau déjà connue, une classe de tarball connue, ?). De plus dans un soucis de confort pour l'utilisateur nous pouvons lui fournir une IHM très rapide à prendre en main avec des complétions automatiques ou des options par défaut ?.

Il pourrait aussi générer à partir de quelques informations préliminaires sur un réseau un audit plus complet d'une compile farm et ainsi déduire les capacités de Vampire face à une telle compile farm.

Installation d'outils : Il est fortement souhaitable de fournir un module à Vampire pour lui indiquer ce qu'un programme à besoin pour être compiler et exécuter. On pourra les inclure dans la tarball et Vampire lors des tests devra installer ces outils, tester et enfin désinstaller tous les outils (et programmes bien sur) sur les machines hôtes.