Département de formation doctorale en informatique
UFR STMIA

École doctorale IAEM
Université de Lorraine, France

Gestion des Vulnérabilités dans les Réseaux et Systèmes Autonomes

THÈSE

(Résumé étendu)

présentée et soutenue publiquement le 12 juin 2014

pour l'obtention du grade de

Docteur de l'Université de Lorraine

(Spécialité Informatique)

par

Martín BARRÈRE CAMBRÚN

Composition du jury

Président:	Le président
Rapporteurs :	Dr. Michelle SIBILLA. Professeur à l'Université Toulouse III Paul Sabatier, France. Dr. Raouf BOUTABA. Professeur à l'Université de Waterloo, Canada.
Examinateurs :	Dr. Emil LUPU. Reader at Imperial College London, U.K. Dr. Nacer BOUDJLIDA. Professeur à l'Université de Lorraine, France. Dr. Rémi BADONNEL. Maître de Conférences à l'Université de Lorraine, France. Dr. Olivier FESTOR. Professeur à l'Université de Lorraine, France.

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

Table des matières

Résumé / Abstract

Le déploiement d'équipements informatiques à large échelle, sur les multiples infrastructures interconnectées de l'Internet, a eu un impact considérable sur la complexité de la tâche de gestion. L'informatique autonome permet de faire face à cet enjeu en spécifiant des objectifs de haut niveau et en déléguant autant que possible les activités de gestion aux réseaux et systèmes eux-mêmes. Cependant, lorsque des changements sont opérés par les administrateurs ou directement par les équipements autonomes, des configurations vulnérables peuvent être involontairement introduites, même si celles-ci sont correctes d'un point de vue opérationnel. Ces vulnérabilités offrent un point d'entrée pour des attaques de sécurité. Les environnements autonomes doivent être capables de se protéger pour éviter leur compromission et la perte de leur autonomie. À cet égard, les mécanismes de gestion des vulnérabilités sont essentiels pour assurer une configuration sûre de ces environnements.

Cette thèse porte sur la conception et le développement de nouvelles méthodes et techniques pour la gestion des vulnérabilités dans les réseaux et systèmes autonomes, afin de leur permettre de détecter, d'évaluer et de corriger leurs propres expositions aux failles de sécurité. Nous présenterons tout d'abord un état de l'art sur l'informatique autonome et la gestion de vulnérabilités, en mettant en relief les défis importants qui doivent être relevés dans ce cadre. Nous décrirons ensuite notre approche d'intégration du processus de gestion des vulnérabilités dans ces environnements, et en détaillerons les différentes facettes, notamment : extension de l'approche dans le cas de vulnérabilités distribuées, prise en compte du facteur temps en considérant une historisation des paramètres de configuration, et application en environnements contraints en utilisant des techniques probabilistes. Nous présenterons également les prototypes et les résultats expérimentaux qui ont permis d'évaluer ces différentes contributions.

Mots clés: sécurité, gestion de réseaux, informatique autonome, gestion de vulnérabilités.

Over the last years, the massive deployment of computing devices over disparate interconnected infrastructures has dramatically increased the complexity of network management. Autonomic computing has emerged as a novel paradigm to cope with this challenging reality. By specifying high-level objectives, autonomic computing aims at delegating management activities to the networks themselves. However, when changes are performed by administrators and self-governed entities, vulnerable configurations may be unknowingly introduced. Vulnerabilities constitute the main entry point for security attacks. Hence, self-governed entities unable to protect themselves will eventually get compromised and consequently, they will lose their own autonomic nature. In that context, vulnerability management mechanisms are vital to ensure safe configurations, and with them, the survivability of any autonomic environment.

This thesis targets the design and development of novel autonomous mechanisms for dealing with vulnerabilities, in order to increase the security of autonomic networks and systems. We first present a comprehensive state of the art in autonomic computing and vulnerability management, and point out important challenges that should be faced in order to fully integrate the vulnerability management process into the autonomic management plane. Afterwards, we present our contributions which include autonomic assessment strategies for device-based vulnerabilities and extensions in several dimensions, namely, distributed vulnerabilities (spatial), past hidden vulnerable states (temporal), and mobile security assessment (technological). In addition, we present vulnerability remediation approaches able to autonomously bring networks and systems into secure states. The scientific approaches presented in this thesis have been largely validated by an extensive set of experiments which are also discussed in this manuscript.

Keywords: security, network management, autonomic computing, vulnerability management.

Chapitre 1

Introduction générale

Sommaire

1.1 Contexte scientifique et technique

Le large déploiement de systèmes informatiques sur des infrastructures hétérogènes interconnectées, a radicalement changé les approches liées à la gestion des réseaux. En particulier, l'Internet a joué un rôle fondamental en fournissant une plateforme pour les milliers de technologies hétérogènes qui constituent actuellement notre monde numérique. Aujourd'hui, presque tous les réseaux, y compris l'Internet lui-même, sont en pleine expansion en réponse à de nombreux facteurs. À mesure que les utilisateurs finaux deviennent de plus en plus familiarisés avec les technologies informatiques, une demande croissante en nouveaux besoins accompagne cette expansion. Ces nouvelles exigences apparaissent sous différentes formes, qui ont un impact direct sur les mécanismes et les ressources utilisés pour les atteindre. Inversement, la technologie peut également être perçue comme un flux dynamique qui façonne, dans une certaine mesure, le comportement des utilisateurs finaux. Cette symbiose entre la technologie et ses utilisateurs agit en tant que moteur pour leur évolution respective.

Toutefois, cette évolution n'est pas sans danger. Les approches constructives ont tendance à pousser à bout les limites des technologies actuelles. Ainsi, alors que les réseaux sont devenus la principale plateforme pour l'échange d'informations et que toutes sortes de services ont été bâtis au-dessus d'eux, la complexité de leur gestion s'est considérablement accrue. De plus, ces scénarios sont dynamiques, ce qui met encore plus à l'épreuve les solutions de gestion actuelles. Dans ce contexte, l'approche de l'informatique autonome a été conçue comme une réponse à ce problème. Et si nous n'avions plus besoin des tâches d'administration spécifiques pour adapter nos réseaux à chaque service donné ? Et si nous pouvions simplement exprimer nos attentes envers les réseaux, de telle sorte que ce soit l'infrastructure sous-jacente elle-même qui s'occupe de leur gestion ? D'un point de vue haut-niveau, ces questions peuvent nous aider intuitivement à comprendre l'esprit de l'informatique autonome. En effet, l'informatique autonome vise à libérer les administrateurs de tâches de gestion lourdes et sujettes aux erreurs. L'idée principale est de définir le comportement attendu des réseaux en spécifiant des objectifs de haut niveau, et déléguer la responsabilité de la réalisation de ces objectifs aux réseaux eux-mêmes.

Dans cette optique, les réseaux et les systèmes autonomes peuvent répondre efficacement aux problématiques liées à la gestion du développement technologique actuelle, qui se veut de plus en plus rapide. Cependant, afin d'atteindre cet objectif, la sécurité est essentielle. La capacité de tout système à être autonome implique que chacun de ces systèmes doit s'auto-protéger. Si cette condition n'est pas satisfaite, il est alors possible de compromettre ces entités autonomes, ce qui peut affecter non seulement leur propre comportement, mais aussi leur environnement. Pour cette raison, **la capacité des réseaux et des systèmes autonomes à gérer les vulnérabilités ainsi que leur propre exposition à des attaques est un facteur critique pour leur survie**. Cette question constitue le cœur de notre travail. **Cette thèse vise à fournir de nouveaux mécanismes autonomes pour faire face aux vulnérabilités, afin d'accroître la sécurité des réseaux et des systèmes autogérés**.

1.2 Problématique

Dans la sécurité informatique, les vulnérabilités sont des *bugs* ou des faiblesses dans la conception, la mise en œuvre, ou la configuration d'un système. Ces faiblesses permettent à un attaquant de passer outre les politiques de sécurité et de porter atteinte à l'intégrité du système. Les vulnérabilités constituent le principal point d'entrée pour pénétrer dans les systèmes informatiques et obtenir un accès non autorisé à des actifs au sein de ces systèmes. Par conséquent, la capacité à gérer les vulnérabilités est essentielle pour n'importe quel système informatique. Les réseaux et systèmes autonomes ne font pas exception, d'autant plus que leur nature autonome augmente la complexité du processus de gestion des vulnérabilités. En effet, les tâches habituellement effectuées par des administrateurs humains sur des systèmes ordinaires, doivent maintenant être effectuées par des entités autonomes régies par leurs propres moyens. Le processus de gestion des vulnérabilités consiste essentiellement en la détection et la correction de vulnérabilités. Néanmoins, la conception des réseaux et des systèmes autonomes incluant ce processus pose des problèmes importants. **Comment pouvons-nous fournir des environnements autonomes intégrant des mécanismes conçus pour accroître leur prise de conscience vis-à-vis des vulnérabilités ? Quelles méthodes faut-il employer pour identifier les failles de sécurité de manière autonome ? Comment les systèmes doivent-ils procéder pour atténuer et éliminer les vulnérabilités détectées, tout en maintenant leur état interne opérationnel et fiable ?** Ces questions, pouvant être rejointes par d'autres, constituent les principaux problèmes que cette thèse vise à traiter.

L'informatique autonome a ouvert de nouveaux horizons pour traiter les problèmes où les méthodes traditionnelles semblent échouer. En particulier, des mécanismes de gestion capables d'accompagner correctement l'évolution des réseaux dynamiques à large échelle, et capables de faire face à leur complexité croissante, sont tout simplement indispensables. L'informatique autonome s'intègre parfaitement à ces exigences. Cependant, si les infrastructures autonomes ne développent pas les mécanismes et les techniques nécessaires pour se protéger contre les menaces de sécurité, leur pouvoir réel et leur utilité risquent d'être fortement réduits. L'objectif de cette thèse est de contribuer dans ce sens en fournissant un état de l'art sur l'informatique autonome et la gestion de vulnérabilités, mais aussi soulever et résoudre les questions scientifiques manquantes nécessaires pour consolider les fondations et la sécurité de l'informatique autonome.

1.3 Organisation du document

Ce document vise à fournir un aperçu du contenu présenté dans cette thèse. Une documentation complète peut être trouvée dans le manuscrit principal. Le présent document est organisé comme suit. Le chapitre 2 présente les concepts clés de l'informatique autonome et de la gestion de vulnérabilités, ainsi que la connexion entre ces deux mondes. Le chapitre 3 décrit les contributions de cette thèse et les prototypes développés pour valider nos approches. Le chapitre 4 présente les conclusions générales et décrit les perspectives à long terme de ce travail.

Chapitre 2

Informatique autonome et gestion des vulnérabilités

Sommaire

2.1 Introduction

Le développement croissant des réseaux, et la multiplication des services fournis par ceux-ci, ont considérablement augmenté la complexité de la gestion des réseaux. L'informatique autonome a été introduite pour résoudre cette complexité en concevant des réseaux et services capables de faire leur travail en autogestion, en suivant des politiques qui indiquent les objectifs attendus. Dans ce chapitre, nous présenterons tout d'abord les concepts essentiels de l'informatique autonome et de la gestion de vulnérabilités. Ensuite, nous expliquerons pourquoi la gestion de vulnérabilités constitue une activité essentielle pour assurer l'autonomie réelle, et présenterons un aperçu de l'approche adoptée dans cette thèse pour atteindre cet objectif.

2.2 Vue d'ensemble de l'informatique autonome

Le paradigme de l'informatique autonome a été proposée en 2001 par IBM [9], comme une réponse à la complexité croissante de la gestion de systèmes informatiques [14, 6]. La vision de l'informatique autonome est fortement inspiré du système nerveux : à chaque fois que nous respirons, nous ne le faisons pas consciemment, et pourtant, c'est une fonction vitale qui suit un comportement humaine de haut niveau, survivre. Également, l'informatique autonome vise à fournir une infrastructure où les réseaux sont gérés à travers des objectifs de haut niveau. Les composants du réseau sous-jacents effectueront des tâches conformément à ces règles ainsi qu'à un environnement changeant et dynamique, sans aucune intervention humaine explicite. En effet, les solutions autonomes visent à imiter le comportement humain au cours du processus, à travers du raisonnement et de la prise de décisions, pour le bon fonctionnement du système. Cette perspective vise à fournir des bases solides pour développer des infrastructures évolutives

et flexibles en mesure de répondre à une réalité technologique exigeante et changeante [21], [8]. Dans cette thèse, un système autonome est défini comme suit :

Définition 1 (Système autonome)**.** *Un système autonome est une entité autogérée, capable de se gérer sans aucun type de contrôle externe et d'effectuer des tâches sans intervention humaine, afin d'atteindre les objectifs pour lesquels elle a été créée. Bien que le but du système autonome est défini par les objectifs de haut niveau, la réalisation de cet objectif est déléguée au système lui-même. Les activités internes effectuées par un système autonome peuvent comprendre la perception de l'environnement, l'analyse, le raisonnement, la prise de décision, la planification et l'exécution des actions qui doivent assurer le bon fonctionnement du système.*

Il est important de souligner que les systèmes autonomes sont régis par des politiques ou des objectifs de haut niveau. C'est ce qui fait la différence clé entre les solutions autonomes et automatisés. Habituellement, les solutions autonomes sont constituées de composants appelés entités autonomes. Les entités autonomes sont conçues pour servir à un but précis comme la surveillance d'un équipement réseau ou la fourniture d'un service de routage. Ces entités peuvent aussi être combinées pour construire des solutions autonomes plus complexes. Afin d'organiser leur nature autonome, les systèmes autogérés impliquent un ensemble de domaines fonctionnels appelé propriétés *auto-**, définies comme suit [16] :

- **auto-configuration**, qui fournit des mécanismes et techniques pour la configuration automatique des composants et des services,
- **auto-optimisation**, qui couvre des méthodes de monitorage et d'adaptation des paramètres afin d'obtenir un fonctionnement optimal selon les politiques qui régissent le système,
- **auto-guérison**, pour détecter, diagnostiquer et réparer automatiquement les problèmes de logiciels et équipements localisés, et
- **auto-protection**, pour effectuer des activités destinées à la défense du système contre les menaces de sécurité potentielles.

Les propriétés *auto-** visent à résoudre de manière autonome les exigences de haut niveau. Cependant, leur mise en œuvre est complexe et pose des problèmes difficiles. Les tâches d'administration effectuées par des humains, et les changements opérés par les entités autonomes pour réussir les objectifs de haut niveau, peuvent générer des états vulnérables par inadvertance. Même si ces changements peuvent améliorer l'environnement d'un point de vue opérationnel, les configurations résultantes peuvent être non sécurisées, ce qui augmente l'exposition aux menaces de sécurité.

Les éléments autonomes coexistent au sein d'environnements dynamiques, et interagissent avec d'autres éléments autonomes et non autonomes. Si un élément autonome est compromis, ses fonctions et ses capacités deviendront peu fiables, et en conséquence, les éléments autonomes qui utilisent ses services deviendront aussi potentiellement compromis. Cela entraîne inévitablement de la méfiance et l'échec de l'environnement autonome. Par conséquent, les systèmes autonomes doivent être capables de gérer leur propre état et d'effectuer les activités nécessaires pour atteindre des configurations sécurisées. Les éléments autonomes incapables d'assurer cette capacité seront de plus en plus vulnérables, précaires et inutiles. L'automatisation réelle n'est possible que si les réseaux et les systèmes autonomes intègrent pleinement des mécanismes de gestion de vulnérabilités pour assurer des configurations sécurisées. Ceci constitue l'objectif principal de cette thèse, qui est analysé dans la section suivante.

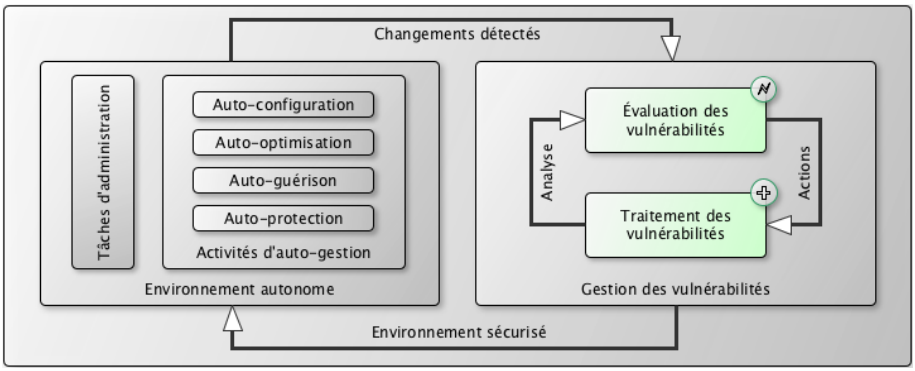

FIGURE 2.1 – Positionnement de la gestion des vulnérabilités par rapport aux activités d'auto-gestion

2.3 Gestion des vulnérabilités dans les environnements autonomes

Les systèmes autonomes doivent s'autogérer en prenant les décisions nécessaires pour répondre aux objectifs de leurs politiques. En tant que tels, ces systèmes doivent traiter des aspects de sécurité, assurer un bon fonctionnement et garantir leurs résultats. Dans cette optique, la gestion de vulnérabilités, en particulier l'évaluation et le traitement des vulnérabilités, est essentiel. Le concept de vulnérabilité considéré dans cette thèse est décrit par la définition suivante :

Définition 2 (Vulnérabilité)**.** *Une vulnérabilité peut être considérée comme une faille ou une faiblesse dans les procédures de sécurité d'un système, la conception, la mise en œuvre, ou les contrôles internes de ce système, qui peut être exploitée et conduire à une violation de la politique de sécurité du système [17], [11].*

Cette thèse vise à fournir des mécanismes autogérés et cohérents pour l'évaluation et le traitement des vulnérabilités, afin d'assurer la sécurité des configurations dans les environnements autonomes. Dans cette perspective, la gestion de vulnérabilités est une question transversalement liée à des activités d'auto-configuration et d'auto-protection des réseaux autonomes. Ce processus, sous la forme d'une boucle de contrôle comme décrit à la figure 2.1, permet l'évaluation et le traitement des états vulnérables potentiellement générés par les tâches d'administrateurs ainsi que les activités d'auto-gestion. L'idée principale est que les changements et les actions effectués dans le système sont surveillés en permanence, et analysés à la recherche de vulnérabilités. Lorsque des états vulnérables sont détectées, des mesures correctives sont appliquées afin de sécuriser l'environnement. La boucle de contrôle de gestion des vulnérabilités reste active pendant toute la durée de vie de l'environnement autonome sous surveillance. Dans ce contexte, la mise en place d'un processus de sécurité permettant de faire face aux vulnérabilités nécessite la spécification d'une politique qui définisse l'état désiré du système, ainsi qu'un état initial connu et fiable pour garantir l'identification de vulnérabilités et la conformité de la politique de sécurité du système [23].

Les principales activités réalisées pendant le cycle de vie de la gestion de vulnérabilités peuvent être mises en correspondance avec les différentes activités présentes dans les composants autonomes. La figure 2.2 décrit cette correspondance, en illustrant le cycle de vie global d'un composant autonome et les principales activités requises par la gestion de vulnérabilités [10]. Les activités d'identification de vulnérabilités ont lieu dans la phase de monitorage où les tâches de surveillance pour l'évaluation et l'analyse d'états vulnérables sont effectuées (I), en profitant des connaissances en matière de sécurité. Lorsqu'un problème de sécurité est trouvé, il est classé (II) et les changements sont planifiés. Les contre-mesures d'une vulnérabilité sont prévues en

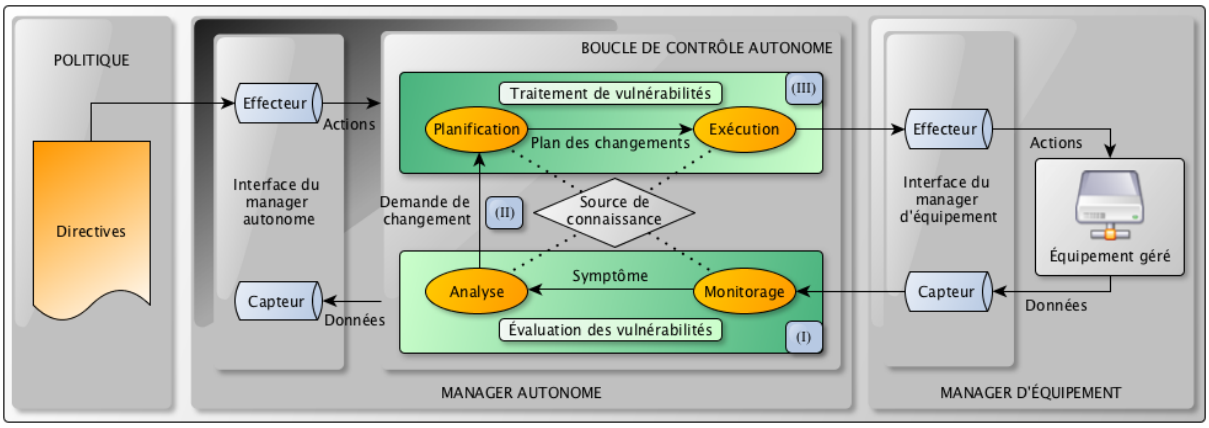

FIGURE 2.2 – Confrontation du processus de gestion de vulnérabilités aux systèmes autonomes

fonction de plusieurs facteurs tels que l'importance, le risque et l'impact de cette vulnérabilité. Finalement, un plan de changements est généré et les tâches de correction sont exécutées (III) afin de maintenir la sécurité et être conforme à la politique actuelle du système. La figure 2.2 illustre l'approche globale adoptée dans cette thèse pour l'intégration des activités de gestion de vulnérabilités dans le plan autonome. Une analyse en profondeur sur les méthodes et techniques existantes pour traiter les vulnérabilités dans les systèmes autonomes et non autonomes est disponible dans le manuscrit complet de cette thèse.

Au regard de nos contributions, nous avons exploité les avantages du protocole SCAP (*Content Security Automation Protocol*) [2]. SCAP est un ensemble de spécifications qui normalise le format et la nomenclature des informations sur les failles connues des logiciels et de systèmes ainsi que les configurations de sécurité recommandées. Ces avis sont annotés avec des identifiants communs et spécifiés en XML. En particulier, nous nous sommes focalisés sur le langage OVAL (*Open Vulnerability and Assessment Language*) [18] et XCCDF (*eXtensible Liste Configuration Description Format*) [25]. OVAL, soutenu par la MITRE Corporation [15], standardise la façon d'évaluer et de rapporter l'état des systèmes informatiques. XCCDF, soutenu par le NIST (National Institute of Standards and Technology) [17], fournit un support pour la création des critères de sécurité de référence, ainsi que la présentation des résultats de l'évaluation. Ensemble, OVAL et XCCDF permettent non seulement d'identifier des vulnérabilités mais aussi de remettre les systèmes dans un état de conformité grâce au traitement des vulnérabilités et des erreurs de configuration identifiés.

Dans le langage OVAL, une vulnérabilité donnée est décrite à partir d'une *définition OVAL*. Une *définition OVAL* spécifie un critère qui combine logiquement une série de *tests OVAL*. Chaque *test OVAL* représente le processus par lequel une condition ou propriété spécifique est évaluée sur le système cible. Chaque *test OVAL* examine un *objet OVAL* à la recherche d'un *état OVAL* spécifique. Donc, un *test OVAL* sera vrai si l'état de l'*objet OVAL* référencé correspond à l'*état OVAL* spécifié. Le résultat global pour les critères spécifiés dans la *définition OVAL* sera construit en utilisant les résultats de chaque *test OVAL* référencé.

A titre d'exemple, nous considérons la situation illustrée à la figure 2.3 où une vulnérabilité de la plateforme Cisco IOS [12] vient d'être divulguée. Pour que cette vulnérabilité soit présente, deux conditions doivent se produire simultanément : (I) la version de la plateforme doit être *12.4* et (II) le service *ip finger* doit être activé. Cette vulnérabilité peut être exprimée dans un document OVAL en spécifiant une définition OVAL qui organise deux tests OVAL comme une combinaison logique. Un test est en charge d'évaluer la version du système et l'autre doit vérifier l'état du service. Les objets OVAL utilisés par ces tests seront : un objet qui représente

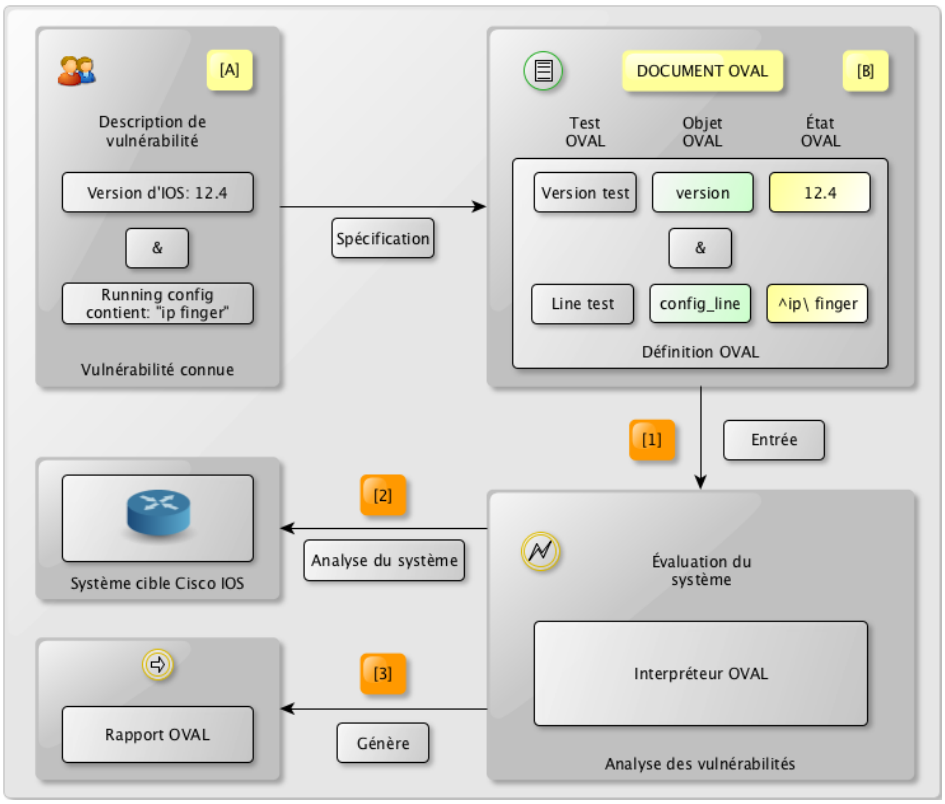

FIGURE 2.3 – Exemple d'utilisation du langage OVAL avec le système Cisco IOS

la version du système et un autre objet qui représente la configuration en cours, respectivement. Finalement, les états OVAL, un état pour la version et un autre pour le service, expriment les états qui devraient être observés sur chaque objet pour considérer le système comme étant vulnérable. Une fois que le document OVAL a été spécifié, l'évaluation peut se résumer en trois étapes principales. À l'étape 1, le document est consommé par un interpréteur OVAL. À l'étape 2, le système cible est analysé à la recherche de vulnérabilités. L'analyse OVAL se décompose en deux parties, à savoir, la collecte d'objets OVAL nécessaires qui doivent être analysés, et la comparaison des objets OVAL recueillis face aux états OVAL spécifiés. Enfin, un rapport est produit à l'étape 3 en indiquant les résultats du processus d'évaluation. Dans cet exemple particulier, il est prévu d'observer la valeur 12.4 en tant que version du système, et une ligne qui commence avec la directive *ip finger* dans le fichier de configuration courant. Si ces deux propriétés sont observées, la vulnérabilité est présente sur le système cible.

L'utilisation de langages standards comme OVAL est essentielle pour augmenter la prise de conscience des environnements autonomes vis-à-vis des vulnérabilités. En outre, les langages tels que XCCDF contribuent fortement au développement des activités de traitement de manière autonome. Les réseaux et systèmes autonomes doivent être capables de capitaliser les connaissances fournies par ces avis de sécurité afin d'améliorer leur capacité à faire face aux vulnérabilités. Dans cette thèse, nous avons poursuivi cet objectif en proposant des nouvelles approches qui seront présentées dans le chapitre suivant.

2.4 Synthèse

Le développement accéléré des réseaux et de leurs services requière de nouvelles méthodes de gestion pour répondre à leurs propriétés. Le paysage technologique change rapidement et les utilisateurs collaborent pour façonner son devenir. Par conséquent, il est essentiel de promouvoir des approches dynamiques et adaptatives pour faire face à cette réalité changeante. L'informatique autonome fournit des bases solides qui peuvent accompagner cette évolution et aider à résoudre plusieurs défis dans la gestion des réseaux actuels. Cependant, si nous voulons faire confiance aux solutions autonomes, les systèmes autogérés doivent garantir la sécurité de leurs configurations. Dans ce contexte, l'objectif de cette thèse est de proposer de nouveaux mécanismes autonomes de gestion des vulnérabilités, afin d'augmenter la sécurité des réseaux et des systèmes autonomes. Dans le prochain chapitre, nous présenterons un résumé de nos contributions scientifiques et des prototypes développés au cours de cette thèse.

Chapitre 3

Contributions

Sommaire

3.1 Introduction

De nos jours, les technologies numériques se développent rapidement, le nombre d'utilisateurs finaux croît également fortement, avec une demande constante en nouveaux services. Les réseaux informatiques constituent la plateforme de ce monde numérique hétérogène. Cette évolution accélérée a poussé les limites des approches de gestion de réseaux traditionnelles, qui s'adaptent de plus en plus difficilement aux exigences des nouveaux réseaux. L'informatique autonome a émergé comme une réponse pour aborder ce défi en déléguant des tâches de gestion aux réseaux eux-mêmes. Cependant, lorsque des modifications sont effectuées par les administrateurs et les entités autogérées, des configurations vulnérables peuvent être accidentellement introduites. Les vulnérabilités constituent le principal point d'entrée pour les attaques informatiques. Aussi, les entités autogérés qui seront incapables de se protéger, finiront par être compromises et perdront leur nature autonome en conséquence. Dans ce contexte, les mécanismes de gestion de vulnérabilités sont essentiels pour assurer des configurations sécurisées, et pour permettre le maintien de n'importe quel environnement autonome. Cette thèse vise la conception et le développement de nouvelles approches pour la gestion des vulnérabilités dans les réseaux et les systèmes autonomes. Dans ce chapitre, nous présenterons brièvement nos contributions scientifiques sur la gestion autonome des vulnérabilités dans la section 3.2. Ensuite, nous décrirons les prototypes mise en œuvre dans la section 3.3. La section 3.4 conclut le chapitre par une synthèse.

3.2 Une plateforme autonome de gestion des vulnérabilités

Cette section présente les contributions de cette thèse. En considérant le processus de gestion des vulnérabilités, nous classons nos contributions en deux catégories principales : détection des vulnérabilités et correction des vulnérabilités. La figure 3.1 présente nos travaux de recherche organisés en différentes sections où les flèches pointillés illustrent le flux de lecture principale à travers eux.

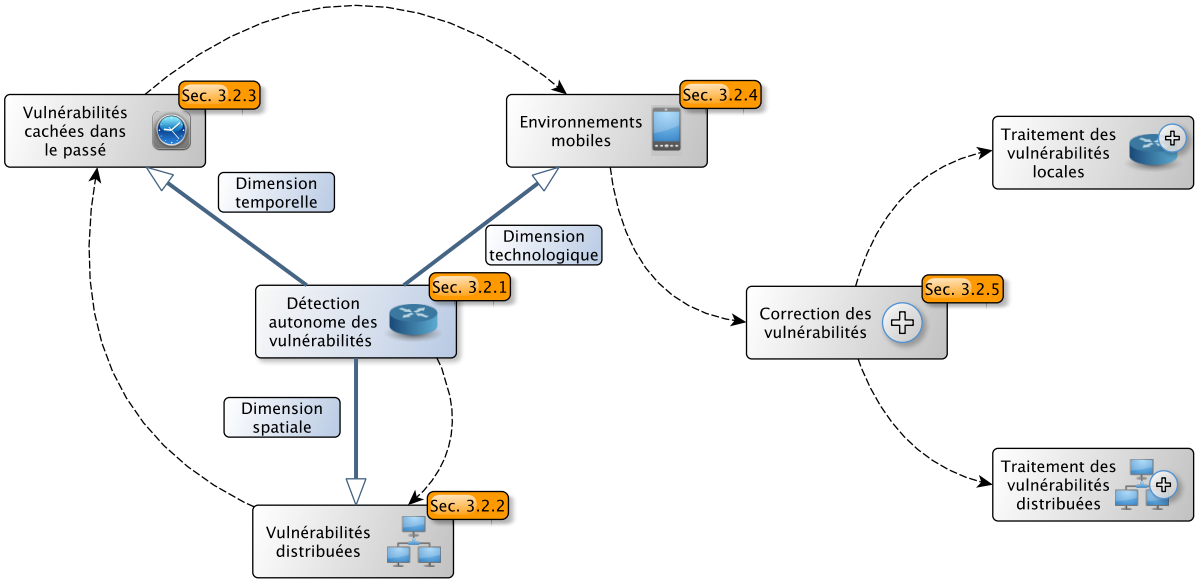

FIGURE 3.1 – Organisation des contributions

La première contribution présentée dans la section 3.2.1, consiste en une approche d'évaluation autonome des vulnérabilités pour des machines individuelles. A partir de celle-ci, trois dimensions représentées par des flèches pleines étendent l'activité d'évaluation des vulnérabilités à de nouveaux scénarios tenant compte de contraintes spatiales, temporelles et technologiques. La section 3.2.2 étend le concept de vulnérabilités individuelles à des vulnérabilités composées qui sont reparties à travers le réseau. Nous appelons cette extension spatiale, vulnérabilités distribuées. La section 3.2.3 constitue la deuxième dimension prenant en compte le facteur temps. Cette approche permet d'augmenter la sécurité actuelle de systèmes informatiques en analysant les états vulnérables cachés dans le passé. La section 3.2.4 traduit la dimension technologique où nous avons étudié de nouvelles approches pour l'évaluation des vulnérabilités en environnements aux ressources limitées, comme les réseaux mobiles. Tous ces chapitres, de 3.2.1 à 3.2.4, entrent dans la catégorie correspondant à l'évaluation des vulnérabilités. La section 3.2.5 ferme la boucle de gestion en considérant nos contributions sur les activités de correction des vulnérabilités. Ces contributions sont présentées en deux parties, à savoir, le traitement des vulnérabilités individuelles, et le traitement des vulnérabilités distribuées, tous les deux décrits dans la section 3.2.5.

3.2.1 Détection autonome des vulnérabilités

En général, les changements qui sont effectués par les réseaux et les systèmes autonomes peuvent générer des vulnérabilités et accroître leur exposition aux attaques de sécurité. Notre objectif est de permettre aux réseaux autonomes d'exploiter la connaissance fournie par des descriptions de vulnérabilités afin de maintenir leurs configurations fiables. Dans ce contexte, notre

première contribution consiste en une approche autonome pour l'évaluation de vulnérabilités sur des hôtes individuels. À cette fin, nous avons intégré des descriptions de vulnérabilités dans le plan de gestion de ces systèmes autonomes. Nous avons notamment choisi la plateforme Cisco IOS comme une preuve de concept [12]. En traduisant automatiquement les descriptions de vulnérabilités en des politiques qui sont interprétables par un système de configuration autonome, les agents autonomes distribués à travers le réseau deviennent en mesure d'évaluer leur propre exposition en terme de sécurité. Nous avons utilisé le langage OVAL [18] comme un moyen pour spécifier les descriptions de vulnérabilités, et Cfengine [3] en tant que plataforme de gestion par politique pour notre solution. Cette approche, illustrée à la figure 3.2, fournit ainsi un mécanisme pour accroître la prise de conscience des vulnérabilités pour des environnements autogérés.

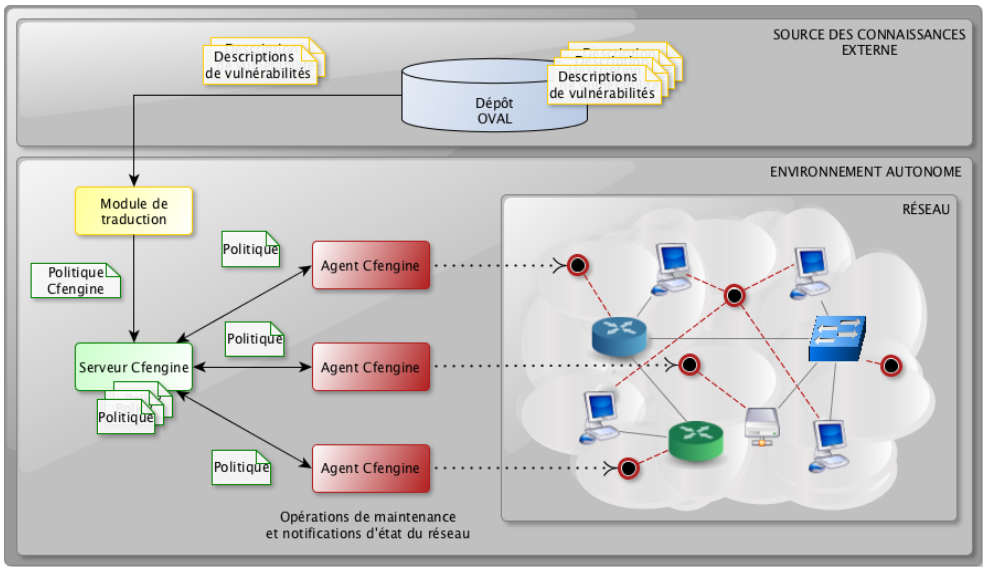

FIGURE 3.2 – High-level architecture

L'architecture proposée implique un dépôt OVAL où les descriptions de vulnérabilités (connues) sont stockées. Ensuite, ces descriptions sont traduites et prises en compte par Cfengine. Pour ce faire, un module de traduction est placé entre le dépôt OVAL et le serveur de politique Cfengine. Ce module, expliqué à la section 3.3, consomme les descriptions disponibles dans le dépôt OVAL et produit des règles Cfengine qui permettent aux agents Cfengine d'être conscient de leurs faiblesses de sécurité. L'architecture Cfengine est basée sur un modèle client-serveur. Le serveur conserve les politiques générées par le module de traduction pendant que les agents autonomes prennent ces nouvelles politiques au moment opportun. De cette manière, les politiques générées sont déployées par le serveur Cfengine dans ses agents Cfengine (points dans le nuage). Ces agents autonomes sont en charge de la gestion des équipements présents dans le réseau cible, afin de détecter et de prévenir des configurations vulnérables lorsque les activités d'auto-gestion sont effectuées. Quand une vulnérabilité est découverte sur un équipement surveillé, les agents Cfengine sont capables de générer des alertes spécifiques, voire d'effectuer des opérations de correction. Notre approche a été validée par un large ensemble d'expériences dont les résultats indiquent sa faisabilité en termes de fonctionnalité et d'intégration dans l'outil de gestion autonome Cfengine. Même si nous nous concentrons sur l'outil Cfengine, actuellement utilisé sur de nombreuses équipements, notre approche peut également être généralisée et être appliquée à d'autres outils de configuration basés sur des politiques tels que Puppet [20] ou Chef [4] notamment.

La prise de conscience des vulnérabilités constitue la première étape vers des infrastructures

autonomes sécurisées, capables de détecter et de corriger les failles de sécurité potentielles. Cependant, l'évaluation de vulnérabilités effectuée individuellement sur les éléments réseau peuvent ne pas fournir une vue complète de l'exposition du réseau.

3.2.2 Extension à des vulnérabilités distribuées

Les activités d'évaluation de vulnérabilités sont traditionnellement effectuées sur chaque équipement réseau de manière individuelle, indépendamment les uns des autres. Parfois cependant, deux ou plusieurs équipements combinés ensemble peuvent produire un état de réseau vulnérable que les approches basées sur les hôtes individuels, ne sont pas capables de détecter. Nous nous référons à ces failles de sécurité comme des vulnérabilités distribuées, qui constituent notre extension dans la dimension spatiale. Les vulnérabilités distribuées doivent être évaluées avec une vue consolidée du réseau afin de détecter des états vulnérables qui peuvent impliquer simultanément deux ou plusieurs équipements réseau. Dans cette thèse, nous avons étendu notre approche pour décrire et évaluer des vulnérabilités distribuées dans les environnements autonomes. Aucuns travaux n'ont précédemment formalisé la notion de vulnérabilités distribuées. Par conséquent, nous avons introduit une construction mathématique qui permet de spécifier formellement les vulnérabilités distribuées ainsi qu'un langage interpretable par des machines pour les décrire. Nous avons également décrit une infrastructure autonome pour l'évaluation de vulnérabilités distribuées qui utilise les connaissances fournies par ces descriptions. Ainsi, notre stratégie permet d'augmenter la prise de conscience de vulnérabilités des équipements individuels ainsi que celle du réseau dans son ensemble.

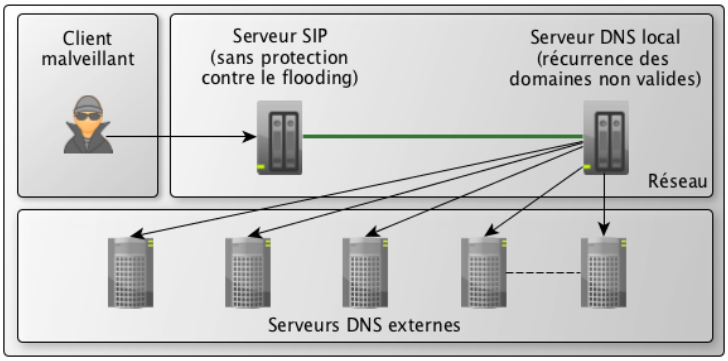

FIGURE 3.3 – Scénario d'une vulnérabilité distribuée

Afin d'expliquer le concept de vulnérabilité distribuée, nous considérons l'exemple décrit dans [24] et illustré à la figure 3.3. Ce scénario implique deux hôtes liés, un serveur SIP (*Session Initiation Protocol*) et un serveur DNS (*Domain Name System*). Chacun a des propriétés spécifiques, cependant, ils constituent ensemble une vulnérabilité niveau réseau potentiellement exploitable. En effet, une attaque par déni de service (DoS) sur le serveur SIP peut être effectuée en l'inondant de noms de domaine irrésolvables qui requièrent l'intervention d'un serveur DNS local. Le serveur DNS local est configuré pour demander la résolution des domaines inconnus vers des serveurs externes, ce qui augmente le nombre de demandes en attente et par conséquent, le temps de réponse pour chaque requête SIP. Dans cette configuration, inonder un serveur SIP avec ce type de messages empêchera celui-ci de répondre aux demandes légitimes. Il est important de souligner que les deux serveurs et la relation entre eux sont des conditions nécessaires pour que la vulnérabilité distribuée soit présente. Si le serveur DNS n'est pas présent ou s'il n'est pas conforme aux conditions spécifiques requises, le serveur SIP répondra immédiatement au client

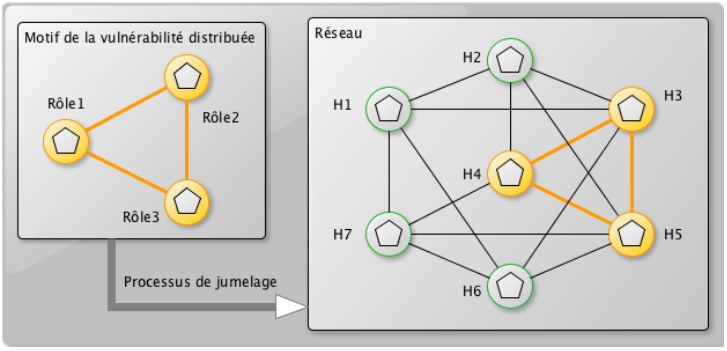

FIGURE 3.4 – Processus de jumelage d'une vulnérabilité distribuée

SIP que sa demande SIP a échoué. Même dans une telle situation, des milliers de requêtes SIP peuvent effondrer le serveur SIP, mais ceci correspond à un scénario légèrement différent qui pourrait être spécifié en utilisant des définitions OVAL standard. D'un autre côté, s'il n'y a pas de serveur SIP, il est clair que la vulnérabilité distribuée n'a pas sa place dans cet environnement.

Formellement, nous définissons une vulnérabilité distribuée comme la projection d'un motif impliquant des états spécifiques d'hôtes (rôles) et des relations entre eux sur un réseau cible, comme illustré à la figure 3.4. Nous avons conçu le langage DOVAL (*Distributed OVAL*), construit au dessus du langage OVAL, comme un moyen pour décrire les vulnérabilités distribuées d'une

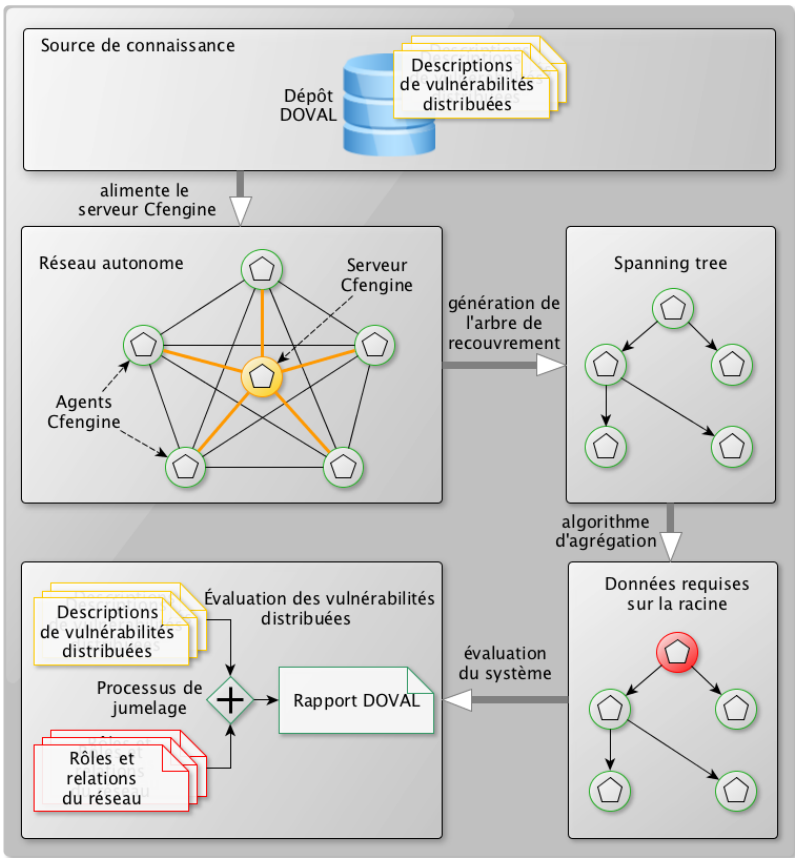

FIGURE 3.5 – Processus d'évaluation des vulnérabilités distribuées

manière interpretable par des machines. En raison de la taille et de la dynamique des réseaux actuels, l'évaluation et la détection d'états distribués vulnérables n'est pas une tâche triviale. Par conséquent, le problème est partitionné en plusieurs étapes, à savoir, (1) la génération d'une topologie réduite sans boucles (*spanning tree*) du réseau sous-jacent, (2) la collecte des informations sur les hôtes et le réseau, et (3) l'évaluation des spécifications DOVAL sur les données recueillies. La figure 3.5 illustre les étapes et l'architecture de l'approche proposée.

Dans cette architecture, les vulnérabilités distribuées sont spécifiées en utilisant le langage DOVAL et sont stockées dans une base de données. Un serveur Cfengine est alimenté par cette connaissance et traduit en des politiques Cfengine, de la même manière que nous l'avons fait avant pour l'évaluation des hôtes individuels. Notre approche considère le déploiement d'agents Cfengine à travers le réseau, où chaque agent est en charge de contrôler un équipement réseau. Afin d'évaluer l'existence d'une vulnérabilité distribuée, un arbre de recouvrement (*spanning tree*) est construit au dessus du réseau cible pour minimiser les chemins et éviter les boucles du réseau. La spécification DOVAL est ensuite transmis à travers l'arbre et les informations nécessaires sont recueillies en utilisant un algorithme d'agrégation sur les nœuds. Chaque agent Cfengine évalue l'équipement qu'il contrôle afin de découvrir quels sont les rôles qu'il peut jouer dans la spécification de la vulnérabilité distribuée. Cette information est transmise et stockée par le nœud racine de l'arbre de recouvrement. Enfin, la spécification de la vulnérabilité distribuée est comparée avec les informations recueillies, et un rapport DOVAL est généré pour informer sur les états vulnérables distribués. En partageant et parallélisant l'évaluation des vulnérabilités distribuées, nos expériences ont montré un comportement linéaire par rapport à la taille du réseau, assurant ainsi le passage à l'échelle dans notre contexte.

Les vulnérabilités distribuées constituent une extension dans la dimension spatiale de notre approche, qui est indispensable pour accroître la sécurité complète des réseaux autonomes. La capacité à comprendre l'état d'éléments autonomes dans le passé et, par conséquent, les vulnérabilités potentielles cachées, constitue également un élément important pour renforcer les mécanismes de sécurité dans le présent.

3.2.3 Analyse dans le passé des états vulnérables cachés

En général, les activités d'évaluation des vulnérabilités n'analysent que les états actuels des systèmes. Cependant, un système compromis dans le passé par une vulnérabilité inconnue à cet instant, peut constituer encore une menace potentielle pour la sécurité dans le présent. En effet, une porte dérobée (*backdoor*) installée par un attaquant par exemple, peut être maintenue dans le système, même si la vulnérabilité originale qui a permis l'attaque a été éradiquée. En conséquence, l'exposition à des attaques dans le passé doit être prise en compte. Notre extension dans la dimension temporelle vise à accroître la sécurité des systèmes informatiques en identifiant dans le passé des états vulnérables cachés. Dans ce contexte, nous avons proposé une modélisation pour détecter cette exposition, et une infrastructure distribuée basée sur OVAL qui permet aux équipements réseau de collecter de façon autonome les informations nécessaires, et analyser automatiquement leur exposition de sécurité dans le passé.

L'infrastructure proposée considère deux processus cycliques indépendants. Le premier processus génère des images des états des systèmes (*snapshots*) de manière autonome. Le deuxième permet de détecter de manière effective les expositions de sécurité dans le passé. La figure 3.6 illustre l'architecture proposée et identifie les principaux composants ainsi que les processus de communication entre eux.

La séquence désignée par les étapes I, II et III constitue le processus de génération d'images ou *snapshots*. À l'étape I, l'analyseur d'exposition fournit des directives pour la collecte des

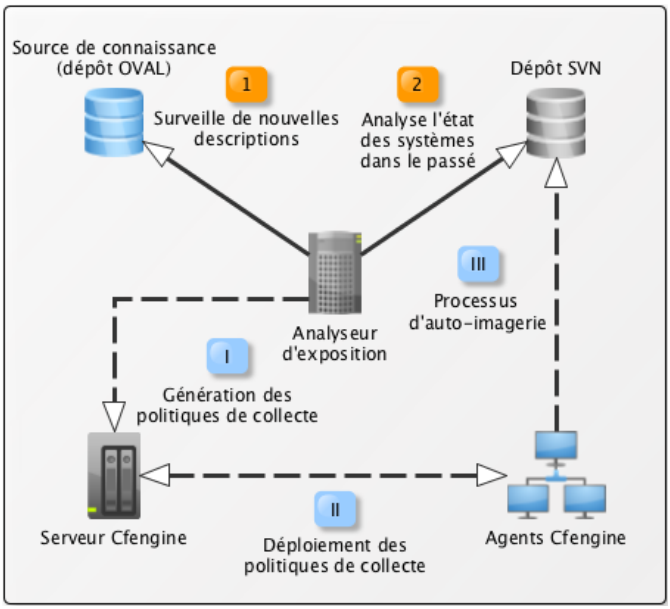

FIGURE 3.6 – Détection des vulnérabilités passés à partir d'une historisation des configurations

données qui seront utilisées pour la construction des images représentées en XML. Ces directives sont spécifiées dans le langage OVAL et traduites automatiquement en des politiques Cfengine. La capacité à exprimer des objets OVAL sans spécifier un état OVAL spécifique nous permet d'utiliser les documents OVAL comme un inventaire des objets à collecter. À l'étape II, les règles Cfengine générées sont transmises aux agents autonomes répartis dans le réseau. Ces agents sont en charge du contrôle des équipements réseau et de la collecte des données pour la construction des images. Enfin, ces images de configurations sont stockées automatiquement dans un dépôt SVN [22] à l'étape III. Le processus de génération d'images constitue une activité autonome et il est effectué indépendamment du processus de détection. Ce dernier, en charge d'externaliser (*outsourcing*) les activités d'évaluation des vulnérabilités, est composé de deux étapes. À l'étape 1, l'analyseur surveille le dépôt de descriptions régulièrement pour vérifier les nouvelles vulnérabilités. Lorsqu'une nouvelle description est disponible (étape 2), il analyse les images stockées dans le dépôt qui ont été créés après la divulgation publique de celle-ci. De cette manière, notre stratégie d'évaluation est capable d'identifier de façon autonome, les périodes de temps durant lesquels les équipements réseau ont été exposés à des menaces de sécurité. Ceci permet d'améliorer encore l'évaluation de vulnérabilités.

La capacité à externaliser les activités d'évaluation de vulnérabilités peut fournir un appui solide lorsque les équipements ont des ressources contraintes, comme les équipements mobiles. Étant axé sur la gestion de vulnérabilités et les environnements autonomes, nous avons ensuite analysé la mise en œuvre de notre approche dans les réseaux mobiles.

3.2.4 Prise en compte des vulnérabilités dans les réseaux mobiles

Le développement des technologies et des services mobiles a contribué au déploiement à large échelle des *smartphones* et tablettes. Ces environnements sont exposés à un large éventail d'attaques de sécurité et peuvent contenir des informations critiques sur les utilisateurs tels que les répertoires de contacts et l'historique des appels téléphoniques. L'évaluation des vulnérabilités de configuration est un défi majeur pour ces environnements, mais cette activité doit être effectuée d'une manière légère afin de réduire l'impact sur leurs ressources. Dans cette thèse, nous avons

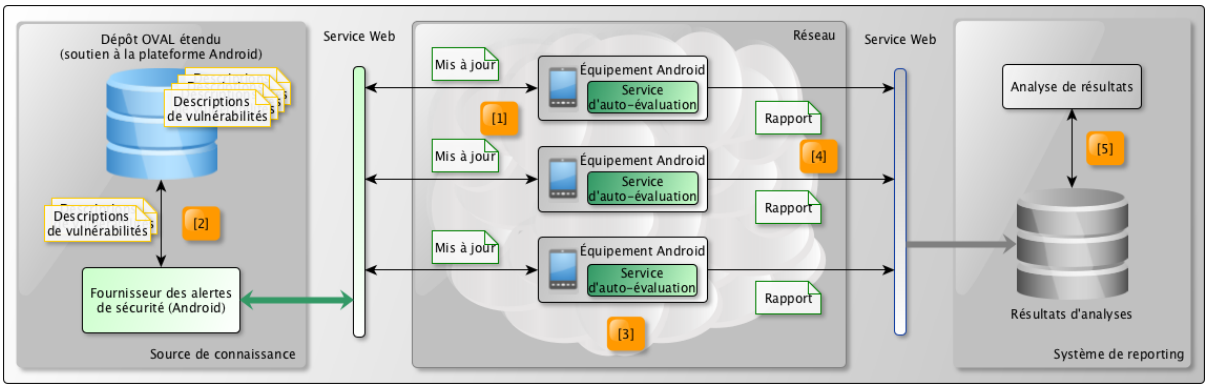

FIGURE 3.7 – Architecture d'auto-évaluation de vulnérabilités basée sur OVAL pour Android

conçu deux méthodes complémentaires pour l'évaluation des vulnérabilités dans des entités mobiles, ce qui constitue notre extension dans la dimension technologique. La première méthode considère une stratégie d'auto-évaluation qui permet aux équipements mobiles d'évaluer leur propre exposition de sécurité. Afin de réduire encore plus la charge de travail côté mobile, nous avons également proposé une stratégie probabiliste intégrée à une architecture client-serveur. Les deux méthodes ciblent la plateforme Android [1] comme preuve de concept, bien que ces approches peuvent être facilement adaptées à d'autres plateformes mobiles.

Auto-évaluation des vulnérabilités

Notre première méthode considère que les entités mobiles sont complètement en charge de l'évaluation de leur exposition de sécurité. À cette fin, nous avons développé un modèle mathématique pour soutenir les activités d'évaluation de vulnérabilités avec des équipements aux ressources limitées. L'architecture globale de notre solution est illustrée à la figure 3.7, qui considère une infrastructure distribuée composée de trois piliers principaux : (1) une source de connaissance (dépôt OVAL) qui fournit les descriptions, (2) des équipements Android qui exécutent le service d'auto-évaluation et (3) un système de *reporting* pour stocker les résultats d'analyses, qui peut être utilisé pour effectuer des analyses plus approfondies. Le processus global est défini comme suit. Tout d'abord à l'étape 1, l'équipement Android surveille périodiquement la source de connaissance à la recherche de nouvelles mises à jour des descriptions de vulnérabilités. Ceci est réalisé en utilisant un service Web offert par le fournisseur d'alertes de sécurité. À l'étape 2, le fournisseur examine sa base de données et renvoie les nouvelles entrées trouvées. À l'intérieur de l'équipement Android, l'outil de mise à jour en cours d'exécution synchronise ses descriptions. Lorsque de nouvelles informations sont disponibles ou des modifications de configuration sont produites dans le système, un service d'auto-évaluation est déclenché afin d'analyser l'équipement (étape 3). À l'étape 4, le rapport contenant les données collectées et les résultats d'analyses est envoyé au système de reporting en effectuant une requête au service web. À l'étape 5, les résultats obtenus sont stockés. Ces informations peuvent également être utilisés avec des objectifs différents tels que des investigations numériques (*forensics*) ou des analyses statistiques.

Évaluation probabiliste des vulnérabilités

La délégation des activités d'évaluation de vulnérabilités directement aux équipements mobiles offre un niveau d'autonomie plus élevé dans le processus. Cependant, lorsque ces activités sont effectuées, le processus qui contrôle le comportement global de l'analyse dans l'entité mobile consomme beaucoup de ressources. Nous avons remarqué que l'externalisation de ce contrôle

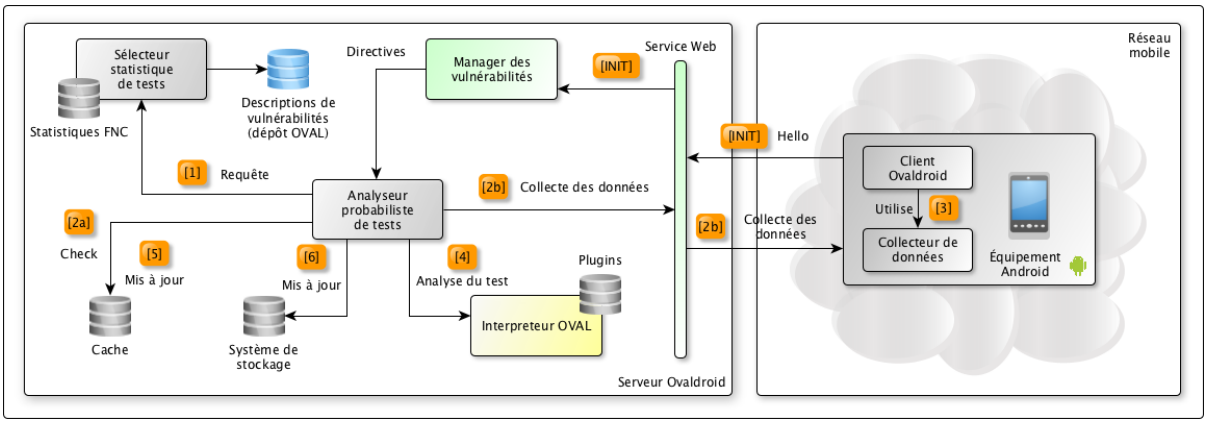

FIGURE 3.8 – Évaluation probabiliste de vulnérabilités pour la plateforme Android

peut encore potentiellement fournir des économies sur les ressources limitées dans équipements mobiles. Dans ce contexte, notre deuxième méthode consiste à centraliser les principaux aspects logistiques de l'évaluation de vulnérabilités en tant que service web. Les clients mobiles ne doivent que fournir au serveur les données nécessaires pour analyser les vulnérabilités connues, décrites avec le langage OVAL. En configurant la fréquence d'analyse et le pourcentage des vulnérabilités à évaluer dans chaque analyse de sécurité, notre infrastructure permet de borner l'allocation des ressources du client ainsi que d'externaliser le processus d'évaluation. Notre stratégie consiste à répartir les activités d'évaluation dans le temps allégeant ainsi la charge de travail sur les entités mobiles, et en même temps, assurer une couverture complète et précise de l'ensemble des vulnérabilités. Cette technique entraîne un processus d'évaluation plus rapide, généralement fait dans le *cloud*, et peut réduire considérablement l'utilisation de ressources du côté client.

L'infrastructure proposée, nommée Ovaldroid, est illustrée à la figure 3.8. Ovaldroid est composé de deux éléments principaux, à savoir : un serveur qui gère l'ensemble du processus d'évaluation, et les clients situés sur le réseau mobile qui utilisent le service d'évaluation des vulnérabilités. Les clients mobiles communiquent périodiquement avec le serveur Ovaldroid afin d'informer sur leur disponibilité pour être analysés. En considérant l'historique des évaluations, le système côté serveur décide s'il est nécessaire de procéder à une nouvelle analyse. Si c'est le cas, l'analyseur probabiliste de tests lance un processus dans lequel une séquence de tests OVAL est exécutée jusqu'à ce que le pourcentage spécifié de vulnérabilités à évaluer soit atteint. Les tests OVAL sont sélectionnés statistiquement en fonction de leur contribution par rapport aux vulnérabilités qui restent à évaluer ainsi que de l'impact de ces vulnérabilités (étape 1). La composante probabiliste prend en compte également la dernière fois que ces tests ont été évalués afin d'éviter le phénomène de famine pour les tests. Notre stratégie intègre un cache pour les objets collectés et les résultats de tests obtenus, ce qui diminue encore davantage la charge de travail côté mobile (étape 2a). Si les données requises ne sont pas disponibles dans le cache, elles sont recueillies à partir de l'équipement mobile (étapes 2b -3) et analysés ensuite (étape 4). Les objets recueillis et les résultats obtenus sont stockés sur le serveur (étapes 5-6). Le processus continue au cours des étapes 1 à 6 jusqu'à ce que le pourcentage de couverture de vulnérabilités spécifié par l'administrateur soit atteint. Nos expérimentations ont montré des résultats prometteurs en indiquant une réduction significative de la charge de travail côté mobile.

Les activités d'évaluation de vulnérabilités jouent un rôle essentiel dans le processus de gestion des vulnérabilités. Cependant, une autonomie réelle ne peut être atteint que si ce processus est complété par une phase de corrections des vulnérabilités.

3.2.5 Correction des vulnérabilités

La gestion des vulnérabilités joue un rôle crucial pour maintenir des configurations fiables et prévenir de nouvelles attaques. Lorsqu'une vulnérabilité a été détectée, les activités de correction sont essentielles pour éliminer cette faille de sécurité. Cependant, ces corrections ne doivent pas générer de nouveaux états vulnérables. Dans cette section, nous présenterons deux méthodes de traitement pour respectivement les vulnérabilités locales et distribuées. Notre première méthode formalise le processus de décision pour la correction d'états vulnérables sur des équipements individuels comme un problème de satisfiabilité (SAT) [5, 19]. Dans ce contexte, nous présenterons une infrastructure autonome qui est en mesure d'évaluer les descriptions de vulnérabilités OVAL, et d'effectuer des actions correctives en utilisant des descriptions d'états de machine futurs basées sur XCCDF [25] et le protocole NETCONF [7]. La deuxième méthode cible les vulnérabilités distribuées. Là, nous proposons une stratégie autonome où les nœuds du réseau collaborent pour remédier les vulnérabilités dans lesquelles ils sont impliqués.

Correction des vulnérabilités locales

L'activité de correction des vulnérabilités constitue une tâche difficile et exigeante. Un seul changement peut avoir un impact ou activer d'autres états vulnérables qui n'étaient pas présents avant le changement. Le même effet peut se produire sur d'autres politiques du système ; dans ce travail cependant, nous traitons uniquement des vulnérabilités de sécurité. Dans ce contexte, la recherche de changements appropries qui, ensemble, peuvent fournir une configuration fiable du système peut générer une explosion combinatoire. En effet, ceci est un problème de décision classé comme un problème NP-complet [5]. Afin de faire face à ce problème, nous proposons de formaliser le problème de décision des changements comme un problème de satisfiabilité booléen ou problème SAT [19]. Étant donné une expression booléenne, le problème SAT consiste à trouver une affectation des variables telles que la formule est évaluée à *vrai*. En spécifiant l'ensemble de descriptions de vulnérabilités comme une formule logique propositionnelle, nous fixons les propriétés du système que nous ne pouvons pas changer, et nous libérons les variables dont les changements sont possibles. Nous utilisons un moteur de résolution SAT pour déterminer les modifications que doivent être faites pour sécuriser le système. Afin de fournir des solutions proactives et réactives, nous proposons le concept d'état futur. Un état futur vise a décrire l'état d'un système après l'application d'un changement spécifique. Ces descriptions peuvent être utilisées pour analyser les effets des modifications de sécurité sans changer réellement le système. Lorsque cette information n'est pas disponible, nous utilisons le protocole NETCONF [7] et sa notion d'état candidat où les changements peuvent être appliqués, analysés et restaurés si nécessaire.

Notre approche SAT, illustré à la figure 3.9, comprend deux processus indépendants, à savoir, un processus pour maintenir des représentations logiques de descriptions OVAL mises à jour, et un deuxième processus pour l'exécution des activités de gestion des vulnérabilités. Le premier processus est en charge de surveiller la base de données qui contient les descriptions des vulnérabilités OVAL (étape I) et la conversion des nouvelles descriptions de vulnérabilités dans des expressions booléennes équivalentes (étape II). Indépendamment, un second processus est en charge de traiter les vulnérabilités, qui est orchestré par le composant de gestion des vulnérabilités (ou manager). À l'étape 1, il communique avec l'analyseur OVAL afin de lancer le processus d'évaluation. L'analyseur récupère ensuite les descriptions de vulnérabilités OVAL a partir d'un dépôt (étape 2), et collecte les données nécessaires à partir des équipements contrôlés (étape 3). Une fois que l'évaluation a été effectuée, l'analyseur renvoie les résultats au manager.

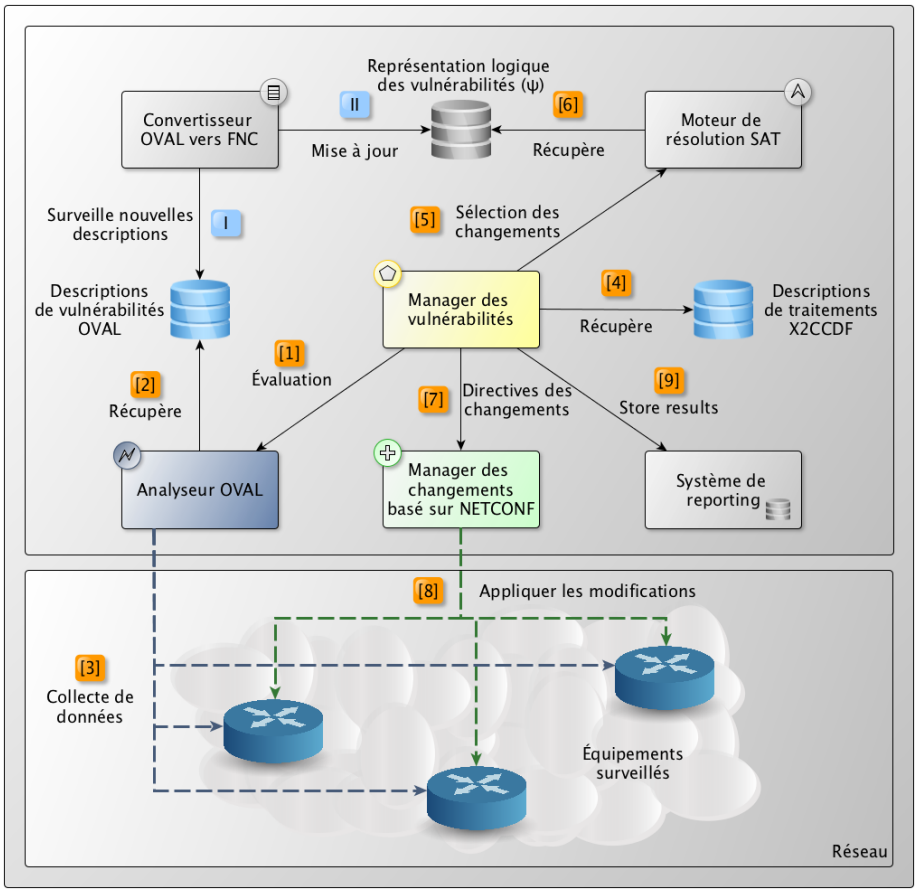

FIGURE 3.9 – Correction de configurations vulnérables avec SAT

Si le système s'avère vulnérable, le manager analyse les descriptions de traitement disponibles (étape 4) et les corrèle avec les propriétés qui peuvent être modifiées dans le système cible. En tenant compte de l'état actuel du système ainsi que des propriétés qui peuvent être changées, le solveur SAT est utilisé (étape 5) pour décider quelles modifications doivent être appliquées afin de sécuriser le système. Le solveur SAT utilise ensuite une représentation logique précisant qu'aucune des vulnérabilités peut se produire (étape 6). Une solution proposée par le solveur SAT indique quelles propriétés doivent être modifiées dans le système pour qu'il présente un état sécurisé. À l'étape 7, le manager interprète ces informations et envoie des directives spécifiques au sous-système de gestion des changements basé sur NETCONF. Le protocole NETCONF est utilisé à l'étape 8 pour communiquer et effectuer les modifications spécifiées sur le système. Enfin, les résultats obtenus sont stockés à l'étape 9. Nos avons effectué plusieurs expériences sur la plateforme Cisco IOS pour évaluer la faisabilité de notre approche. Les résultats obtenus indiquent un comportement linéaire et stable en termes de temps et de charge de travail.

Vers la correction des vulnérabilités distribuées

La correction des vulnérabilités distribuées peut comporter plusieurs tâches à accomplir sur différents équipements. Afin de préciser les tâches de correction de manière interpretable par des machines, nous avons conçu DXCCDF, un langage basé sur XML et construit au-dessus de XCCDF. De cette manière, nous utilisons DOVAL pour spécifier des vulnérabilités distribuées et DXCCDF pour décrire des traitements. Lors de l'analyse des vulnérabilités distribuées, certains

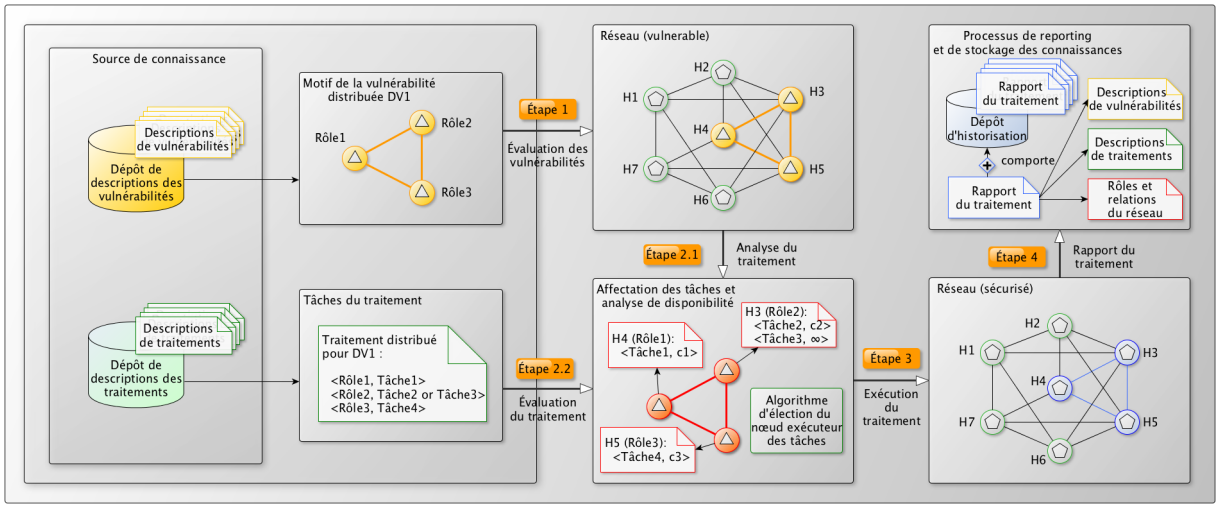

FIGURE 3.10 – Traitement collaboratif des vulnérabilités distribuées

équipements peuvent présenter des états particuliers qui ne leur permettent pas d'effectuer des actions correctives, ou les rendent plus coûteux que pour les autres équipements concernés. Des facteurs tels que la disponibilité, la capacité, ou même la cohérence des politiques doivent être pris en considération lors du processus de traitement. Nous associons à cet ensemble de facteurs un coût pour le nœud afin d'exécuter une tâche de correction. Les modèles pour calculer le coût des tâches sont au-delà de la portée de ce travail et peuvent impliquer plusieurs activités telles que l'évaluation des risques et des techniques de gestion des changements.

Notre méthode de correction considère ces coûts, comme illustré à la figure 3.10. La connaissance est constituée par des dépôts de descriptions de vulnérabilités ainsi que par des spécifications de traitements. À l'étape 1, la description d'une vulnérabilité est récupérée et évaluée dans le réseau, comme indiqué à la section 3.2.2. Si une ou plusieurs instances d'états vulnérables sont trouvées dans le réseau, une analyse des traitements est déclenchée à l'étape 2.1, et la spécification du traitement distribué correspondant est consommé depuis le dépôt à l'étape 2.2. En considérant les tâches disponibles pour corriger le problème de sécurité, les équipements sont analysés à travers le réseau pour trouver un nœud candidat pour exécuter la tâche de correction. Une fois que l'exécution du traitement a été fait et que le réseau a été sécurisé à l'étape 3, un rapport de traitement est généré à l'étape 4. Ce rapport comprend la description de la vulnérabilité, la description du traitement utilisé pour corriger la vulnérabilité, et les informations du réseau recueillies pour effectuer l'activité de correction. Les rapports générés sont stockés dans une base de données d'historisation offrant la possibilité d'exploiter les expériences passées dans la sélection des futurs traitements.

3.3 Mise en œuvre

Afin d'évaluer la faisabilité et l'évolutivité des approches proposées, nous avons développé différents prototypes de mise en œuvre qui servent comme supports à nos expériences. Nous présentons maintenant une brève description de chacun d'eux, bien qu'une documentation complète peut être trouvée dans le manuscrit principal de cette thèse. Notre premier prototype, appelé Ovalyzer, est un système de traduction OVAL vers Cfengine, écrit en Java [13], qui permet l'intégration des descriptions de vulnérabilité OVAL dans le plan de gestion. Ovalyzer joue le rôle d'un module de traduction décrit à la section 3.2.1, qui génère des politiques Cfengine asociées aux

alertes de sécurité OVAL. Ensuite, les politiques Cfengine générés sont consommés par des agents autonomes déployés dans le réseau, devenant ainsi en mesure d'évaluer leur propre exposition en terme de sécurité.

Notre deuxième prototype de mise en œuvre vise à faire face aux expositions de sécurité passées inconnues. En réutilisant l'idée derrière Ovalyzer, ce prototype est capable de générer de manière autonome des images basées sur XML de l'état des systèmes sous surveillance, en suivant des règles Cfengine. Ces images sont ensuite stockées dans un dépôt SVN. Lorsque de nouvelles descriptions de vulnérabilités deviennent disponibles, un analyseur d'exposition évalue automatiquement les images stockées dans le passé afin d'identifier des vulnérabilités cachées.

Notre troisième prototype, appelé Ovaldroid, vise les activités d'évaluation de vulnérabilités sur la plateforme Android. En effet, nous avons mis en place deux méthodes présentées dans la section 3.2.4. Tout d'abord, nous avons développé un service d'auto-évaluation léger capable de surveiller un fournisseur externe à la recherche de nouvelles définitions de vulnérabilités, et d'évaluer sa propre exposition de sécurité. Ensuite, nous avons développé une extension probabiliste où les activités d'évaluation sont contrôlées et effectuées par un serveur externe, réduisant ainsi davantage la charge de travail côté mobile.

Un ensemble d'expériences a été réalisé en utilisant ces prototypes dans leurs scénarios respectifs. Même si ces prototypes sont dans un stade de développement précoce (preuve de concept), ils peuvent être améliorés et étendus, et ont fourni un appui important pour évaluer les approches présentées dans les sections précédentes, en fournissant des résultats prometteurs.

3.4 Synthèse

L'objectif de cette thèse est de fournir de nouveaux mécanismes autonomes pour faire face aux vulnérabilités, afin d'accroître la sécurité des réseaux et des systèmes autogérés. Dans ce chapitre, nous avons fourni une synthèse de nos contributions. En particulier, nous avons présenté une approche autonome pour augmenter la prise de conscience de vulnérabilités dans des environnements autogérées. Nous avons étendu cette approche en considérant également les vulnérabilités distribuées (dimension spatiale), les états vulnérables cachés dans le passé (dimension temporelle) et les équipements mobiles (dimension technologique). De plus, nous avons analysé le traitement des vulnérabilités locales et distribuées. Les prototypes utilisés pour valider les approches présentées dans cette thèse ont également été discutés à la fin de ce chapitre.

Chapitre 4

Conclusion générale

Sommaire

4.1 Résumé des contributions

Le déploiement à grande échelle d'équipements informatiques divers sur des réseaux dynamiques et évolutifs a fortement augmenté la complexité de la gestion des réseaux. L'informatique autonome est devenu un domaine de recherche très important au sein de la communauté scientifique, en fournissant des bases solides et des perspectives prometteuses pour faire face à ce problème. Cependant, deux points principaux nécessitent une attention particulière. Tout d'abord, les questions de sécurité dans des environnements autonomes n'ont pas été suffisamment examinés, en particulier, les mécanismes de gestion des vulnérabilités. Deuxièmement, l'expertise obtenue à partir des approches autonomes a été à peine expérimenté dans les environnements non autonomes. Dans cette thèse, nous avons poursuivi les deux objectifs : enquêter et développer de nouvelles approches de gestion des vulnérabilités pour les environnements autonomes, et transférer les principes autonomes vers des scénarios non autonomes. Dans ce chapitre, nous présentons les conclusions générales de nos travaux de recherche ainsi que nos implantations techniques. Enfin, nous exposons un ensemble de perspectives de recherche.

4.1.1 Gestion autonome des vulnérabilités

Nos contributions peuvent être classés en deux grandes catégories selon le processus de gestion de vulnérabilités, à savoir : la détection des vulnérabilités et leur correction.

La détection de vulnérabilités est une activité essentielle qui permet aux systèmes informatiques d'être conscientes des menaces de sécurité. Dans cette thèse, nous avons présenté plusieurs méthodes pour évaluer les vulnérabilités de manière autonome dans des différents scénarios. Tout

d'abord, nous avons proposé une approche qui permet d'intégrer les descriptions de vulnérabilité OVAL (pour Cisco IOS) dans le plan de gestion autonome. En traduisant les descriptions en des politiques Cfengine, les agents autonomes déployés à travers le réseau deviennent capables d'analyser leur propre exposition en terme de sécurité. Nous avons étendu cette approche pour capturer les vulnérabilités distribuées (**dimension spatiale**). Ce concept considère les situations où deux ou plusieurs équipements réseau peuvent présenter des états locaux sûrs, mais lorsqu'ils sont combinés ensemble, un état vulnérable se pose au niveau réseau. Nous avons également proposé une approche pour augmenter de manière autonome la sécurité actuelle des systèmes informatiques en analysant les états vulnérables cachés dans le passé (**dimension temporelle**). Notre approche est en mesure d'identifier les périodes de vulnérabilité dans le passé où des activités malveillantes peuvent avoir eu lieu. Cette fonction permet d'effectuer des investigations numériques afin d'identifier des failles de sécurité actuelles. En outre, nous avons présenté une méthode pour étendre notre approche dans les environnements mobiles (**dimension technologique**). En effet, nous avons proposé deux méthodes complémentaires qui visent à faire face à l'activité d'évaluation dans le cas d'équipements avec des ressources limitées. Tout d'abord, nous avons mis en place un service d'évaluation de vulnérabilités autonome et léger qui permet aux équipements Android d'évaluer leur propre exposition de sécurité. Ensuite, nous avons étendu cette approche en considérant une méthode probabiliste où les activités d'évaluation sont confiées à un serveur externe qui contrôle l'ensemble du processus d'évaluation, diminuant encore la charge de travail côte mobile.

Afin de clore la boucle de contrôle liée à la gestion des vulnérabilités, nous avons proposé deux méthodes autonomes de correction des vulnérabilités axées sur les vulnérabilités locales et distribuées respectivement. La première méthode considère l'ensemble de toutes les descriptions de vulnérabilités connues comme une conjonction de formules logiques propositionnelles. Ensuite, le problème est codé comme un problème SAT, et un solveur SAT est utilisé pour trouver des configurations sécurisées. Nos expériences en utilisant le protocole NETCONF sur la plateforme Cisco IOS confirment la faisabilité de notre approche. Notre seconde méthode repose sur un mécanisme collaboratif pour décrire et remédier les vulnérabilités distribuées. Cette approche considère des descriptions de correction qui sont prises en compte par les agents Cfengine déployés dans le réseau. Le processus de traitement implique un algorithme distribué qui recueille et analyse les informations des équipements réseau et sélectionne les nœuds candidats pour appliquer les mesures correctives en considérant les coûts pour chaque nœud. Même si nous n'avons pas un prototype complet pour notre deuxième méthode, nous avons effectué une évaluation analytique de sa performance, en obtenant des coûts linéaires lorsque cette approche est intégré dans le processus de gestion des vulnérabilités.

4.1.2 Mise en œuvre

Dans l'objectif de prouver la faisabilité technique de nos contributions, nous avons développé trois prototypes de mise en œuvre qui correspondent à trois scénarios différents d'évaluation des vulnérabilités. Tout d'abord, nous avons mis en place Ovalyzer, un système de traduction de OVAL vers la plateforme Cfengine. Ovalyzer génère des politiques Cfengine qui représentent les descriptions de vulnérabilités OVAL, ce qui permet aux agents autonomes d'effectuer des activités d'auto-évaluation des vulnérabilités au cours des opérations de maintenance. Deuxièmement, nous avons développé un prototype pour identifier les vulnérabilités cachées dans le passé. Nous avons réutilisé les fondements d'Ovalyzer, mais cette fois pour générer de manière automatique, des images de la configuration des systèmes surveillés. Lorsque de nouvelles descriptions de vulnérabilité sont publiées, notre prototype est capable d'analyser l'historique des images du

système à la recherche de périodes vulnérables en exploitant ces nouvelles données. Enfin, nous avons développé Ovaldroid, un système de détection des vulnérabilités OVAL pour Android, qui fonctionne comme un service d'auto-évaluation léger déployé sur l'équipement mobile. Afin de réduire davantage la charge côté mobile, nous avons mis en œuvre une méthode probabiliste qui libère les entités mobiles de l'exécution des tâches d'analyse. Ces activités sont externalisées dans le serveur, puis les résultats sont transmis aux clients mobiles. Tous ces prototypes ont servi en tant qu'infrastructure pour analyser et confirmer la faisabilité et l'évolutivité de notre approche de gestion.

4.2 Perspectives

4.2.1 Défense autonome proactive par anticipation des états vulnérables

Dans cette thèse, nous avons proposé des approches d'analyse de vulnérabilités dans le présent et le passé, ainsi que des approches de correction des vulnérabilités. Ce schéma peut être complété par l'anticipation de la trajectoire de la configuration prise par un système afin d'éviter des modifications qui le conduisent à des états vulnérables connus. En effet, nous avons montré comment les vulnérabilités et l'état des systèmes peuvent être caractérisés par les propriétés qu'ils présentent. En considérant que nous avons n propriétés que nous pouvons modéliser, un système cible pourrait être représenté comme un point unique d'un espace à n dimensions. Les vulnérabilités connues auraient leurs points correspondants dans cet espace, et les vulnérabilités similaires formeraient des groupes (*clusters*) ou sous-espaces vulnérables. L'idée est qu'en observant le mouvement d'un système cible, sa trajectoire pourrait être contrôlée ou maintenue dans cet espace. Si une telle trajectoire indique des niveaux élevés de proximité avec des états vulnérables, elle pourrait être déviée en évitant les changements qui pourraient conduire le système dans ces sous-espaces vulnérables. Cette approche pourrait fournir une mesure continue de l'état des systèmes autonomes au regard des vulnérabilités afin d'anticiper et d'éviter les états de configuration critiques.

4.2.2 Plateforme unifiée de gestion autonome

Les approches proposées dans ce travail renforcent la sécurité d'un réseau de différents points de vue (temporel, spatial, technologique). Cependant, ces approches doivent être davantage unifiée en une plateforme commune et cohérente, capable de fournir toutes ces fonctionnalités de manière transparente. Les changements nécessaires pour remettre un système dans un état sécurisé peuvent aussi contredire les exigences opérationnelles existantes. Par conséquent, le principal défi est de fournir des mécanismes capables de coexister avec d'autres systèmes basés sur des politiques, en maintenant une cohérence à tous les niveaux, a la fois des points de vue opérationnels et de sécurité. De plus, notre approche pour la gestion de vulnérabilités distribuées nécessite un travail technique plus développé, ainsi qu'une analyse complémentaire des mesures nécessaires pour l'exécution des tâches collaboratives dans le contexte d'investigations numériques. La construction d'un modèle standard capable de répondre à ces différents aspects, de façon unifié, serait extrêmement utile pour la communauté de l'informatique autonome, en fournissant une base commune pour la gestion autonome de vulnérabilités.

4.2.3 Sécurité autonome pour les technologies actuelles et émergentes

L'amélioration de la sécurité pour les paradigmes actuels, tels que l'informatique en nuage (*cloud computing*), et les nouveaux modèles comme les réseaux logiciels (SDN, *software-defined networks*) et l'Internet des objets (IoT, *Internet of Things*), est également un challenge majeur. Ainsi, le *cloud computing* permette d'augmenter les capacités de stockage et de traitement tout en découplant les services de l'équipement sous-jacent. Plus récemment, les réseaux logiciels séparent également la gestion (plan de contrôle) du matériel qui met en œuvre effectivement les fonctionnalités du réseau (plan de données). Les deux approches visent à fournir des services fiables et évolutifs, tout en réduisant la complexité de leur gestion et de leur mise en œuvre. L'informatique autonome peut être extrêmement utile dans ce contexte. En fournissant des mécanismes d'auto-configuration et d'auto-protection, ces nouveaux modèles peuvent offrir un support pour la sécurité. Par ailleurs, l'Internet des objets donne lieu à une formidable croissance d'équipements divers interconnectés. Cette tendance indique clairement un besoin de mécanismes de gestion évolutifs et adaptatifs, où la sécurité doit être aussi autonome que le sont les objets eux-mêmes. Dans ce contexte, les solutions de sécurité autonomes pourraient être un élément clé dans l'évolution de l'Internet du futur.

4.3 Publications relatives

Articles dans des revues internationales avec comité de lecture

- **Martín Barrère**, Rémi Badonnel, and Olivier Festor. *Vulnerability Assessment in Autonomic Networks and Services: a Survey.* **IEEE Communications Surveys & Tutorials**, 16(2):988-1004, May 2014. (Impact factor at acceptance date: 6.311).

Chapitres d'ouvrages scientifiques

- **Martín Barrère**, Gaëtan Hurel, Rémi Badonnel, and Olivier Festor. *Increasing Android Security using a Lightweight OVAL-based Vulnerability Assessment Framework.* In Automated Security Management, E. Al-Shaer et al, Eds. Springer International Publishing, 2013, ch. 3, pp. 41-58, ISBN: 978-3-319-01432-6. Book chapter based on our paper selected from the 5th International Symposium on Configuration Analytics and Automation (**SafeConfig'12**), October 3-4, 2012, Baltimore, USA.

Articles dans des conférences internationales avec comité de lecture

- **Martín Barrère**, Rémi Badonnel, and Olivier Festor. *A SAT-based Autonomous Strategy for Security Vulnerability Management.* In Proceedings of the IEEE/IFIP Network Operations and Management Symposium (**NOMS'14**), Mini-Conference, May 5-9, 2014, Krakow, Poland.

- **Martín Barrère**, Gaëtan Hurel, Rémi Badonnel, and Olivier Festor. *A Probabilistic Cost-efficient Approach for Mobile Security Assessment.* In Proceedings of the 9th IEEE International Conference on Network and Service Management (**CNSM'13**), October 14-18, 2013, Zürich, Switzerland. (Acceptance rate 18.1%, 21 out of 116 papers).

- ◇ **Martín Barrère**, Rémi Badonnel, and Olivier Festor. *Improving Present Security through the Detection of Past Hidden Vulnerable States*. In Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (**IM'13**), Mini-Conference, May 27-31, 2013, Ghent, Belgium.

- ◇ **Martín Barrère**, Rémi Badonnel, and Olivier Festor. *Collaborative Remediation of Configuration Vulnerabilities in Autonomic Networks and Systems*. In Proceedings of the 8th IEEE International Conference on Network and Service Management (**CNSM'12**), Mini-Conference, October 22-26, 2012, Las Vegas, USA.

- ◇ **Martín Barrère**, Rémi Badonnel, and Olivier Festor. *Towards the Assessment of Distributed Vulnerabilities in Autonomic Networks and Systems*. In Proceedings of the IEEE/IFIP International Network Operations and Management Symposium (**NOMS'12**), April 16-20, 2012, Maui, Hawaii, USA. (Acceptance rate 26.2%, 55 out of 210 papers).

- ◇ **Martín Barrère**, Rémi Badonnel, and Olivier Festor. *Supporting Vulnerability Awareness in Autonomic Networks and Systems with OVAL*. In Proceedings of the 7th IEEE International Conference on Network and Service Management (**CNSM'11**), October 24-28, 2011, Paris, France. (Acceptance rate 14.6%, 24 out of 164 papers).

- ◇ **Martín Barrère**, Gustavo Betarte, and Marcelo Rodríguez. *Towards Machine-assisted Formal Procedures for the Collection of Digital Evidence*. In Proceedings of the 9th IEEE Annual International Conference on Privacy, Security and Trust (**PST'11**), July 19-21, 2011, Montreal, Canada.

- ◇ **Martín Barrère**, Rémi Badonnel, and Olivier Festor. *Towards Vulnerability Prevention in Autonomic Networks and Systems*. In Proceedings of the 5th International Conference on Autonomous Infrastructure, Management and Security (**AIMS'11**), Ph.D. Symposium, Springer, June 13-17, 2011, Nancy, France.

Démonstrations et séminaires

- ◇ **Martín Barrère**, Gaëtan Hurel, Rémi Badonnel, and Olivier Festor. *Ovaldroid: an OVAL-based Vulnerability Assessment Framework for Android*. Demonstration Sessions of the IFIP/IEEE International Symposium on Integrated Network Management (**IM'13**), May 27-31, 2013, Ghent, Belgium.

- ◇ **Martín Barrère**. *Vulnerability Management for Safe Configurations in Autonomic Networks and Systems*. Ph.D. Seminar, **NSS Department of Loria**, March 28, 2013, Nancy, France.

- ◇ **Martín Barrère**, Rémi Badonnel, and Olivier Festor. *Ovalyzer: an OVAL to Cfengine Translator*. Ph.D. Student Demo Contest of the IEEE/IFIP International Network Operations and Management Symposium (**NOMS'12**), April 16-20, 2012, Maui, Hawaii, USA.

Bibliographie

[1] Android. http://www.android.com/. Last visited on November, 2013.

[2] J. Banghart and C. Johnson. The Technical Specification for the Security Content Automation Protocol (SCAP). Nist Special Publication. http://scap.nist.gov/revision/, 2011. Last visited on January, 2013.

[3] Cfengine. http://www.cfengine.com/. Last visited on November, 2013.

[4] Chef. http://www.getchef.com/chef/. Last visited on November, 2013.

[5] S. A. Cook. The Complexity of Theorem-Proving Procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[6] S. Dobson, F. Zambonelli, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, and N. Schmidt. A Survey of Autonomic Communications. *ACM Transactions on Autonomous and Adaptive Systems*, 1(2):223–259, December 2006.

[7] R. Enns, M. Bjorklund, J. Schönwälder, and A. Bierman. RFC 6241, Network Configuration Protocol (NETCONF). http://tools.ietf.org/html/rfc6241, June 2011.

[8] M. C. Huebscher and J. A. McCann. A Survey of Autonomic Computing–Degrees, Models, and Applications. *ACM Comput. Surv.*, 40:7:1–7:28, August 2008.

[9] IBM. http://www.ibm.com/. Last visited on November, 2013.

[10] IBM. An Architectural Blueprint for Autonomic Computing. *IBM White Paper*, 2006.

[11] V. Igure and R. Williams. Taxonomies of Attacks and Vulnerabilities in Computer Systems. *IEEE Communications Surveys & Tutorials*, 10(1):6–19, January 2008.

[12] Cisco IOS. http://www.cisco.com/. Last visited on November, 2013.

[13] Java technology. http://www.oracle.com/technetwork/java/. Last visited on November, 2013.

[14] J. O. Kephart and D. M. Chess. The Vision of Autonomic Computing. *Computer*, 36(1):41–50, January 2003.

[15] MITRE Corporation. http://www.mitre.org/. Last visited on November, 2013.

[16] Z. Movahedi, M. Ayari, R. Langar, and G. Pujolle. A Survey of Autonomic Network Architectures and Evaluation Criteria. *IEEE Communications Surveys & Tutorials*, PP:1–27, May 2011.

[17] NIST, National Institute of Standards and Technology. http://www.nist.gov/. Last visited on November, 2013.

[18] The OVAL Language. http://oval.mitre.org/. Last visited on November, 2013.

[19] M.R. Prasad, A. Biere, and A. Gupta. A Survey of Recent Advances in SAT-Based Formal Verification. *STTT*, 7(2):156–173, 2005.

[20] Puppet. `http://www.puppetlabs.com/`. Last visited on November, 2013.

[21] N. Samaan and A. Karmouch. Towards Autonomic Network Management: an Analysis of Current and Future Research Directions. *IEEE Communications Surveys & Tutorials*, 11(3):22–36, July 2009.

[22] Apache Subversion. `http://subversion.apache.org/`. Last visited on November, 2013.

[23] A. Williams and M. Nicolett. Improve IT Security with Vulnerability Management. `http://www.gartner.com/id=480703`, 2005. Last visited on November, 2013.

[24] G. Zhang, S. Ehlert, T. Magedanz, and D. Sisalem. Denial of Service Attack and Prevention on SIP VoIP Infrastructures using DNS Flooding. In *Proceedings of the 1st International Conference on Principles, Systems and Applications of IP Telecommunications (IPTComm'07)*, pages 57–66, New York, NY, USA, 2007. ACM.

[25] N. Ziring and S. D. Quinn. Specification for the Extensible Configuration Checklist Description Format (XCCDF). NIST (National Institute of Standards and Technology). `http://scap.nist.gov/specifications/xccdf/`. Last visited on January, 2013.