Présentation de Qubes OS, l'OS client ultra sécurisé.

Par SIMON | *Publié :* 6 SEPTEMBRE 2011

Qubes OS (à ne pas confondre avec qube OS) est un système d'exploitation orienté sécurité du poste de travail proposé par Invisible Things Labs. Il est basé sur Xen et Fedora. La virtualisation est utilisée massivement pour répondre à la problématique de sécurité : les applications tournent dans des machines virtuelles selon leur domaine d'application, tandis que même le stockage et le réseau fonctionnent dans des domaines distincts.

Par défaut, Qubes OS installe l'hyperviseur Xen, trois VM pour ses services et trois domaines pour les applications utilisateur.

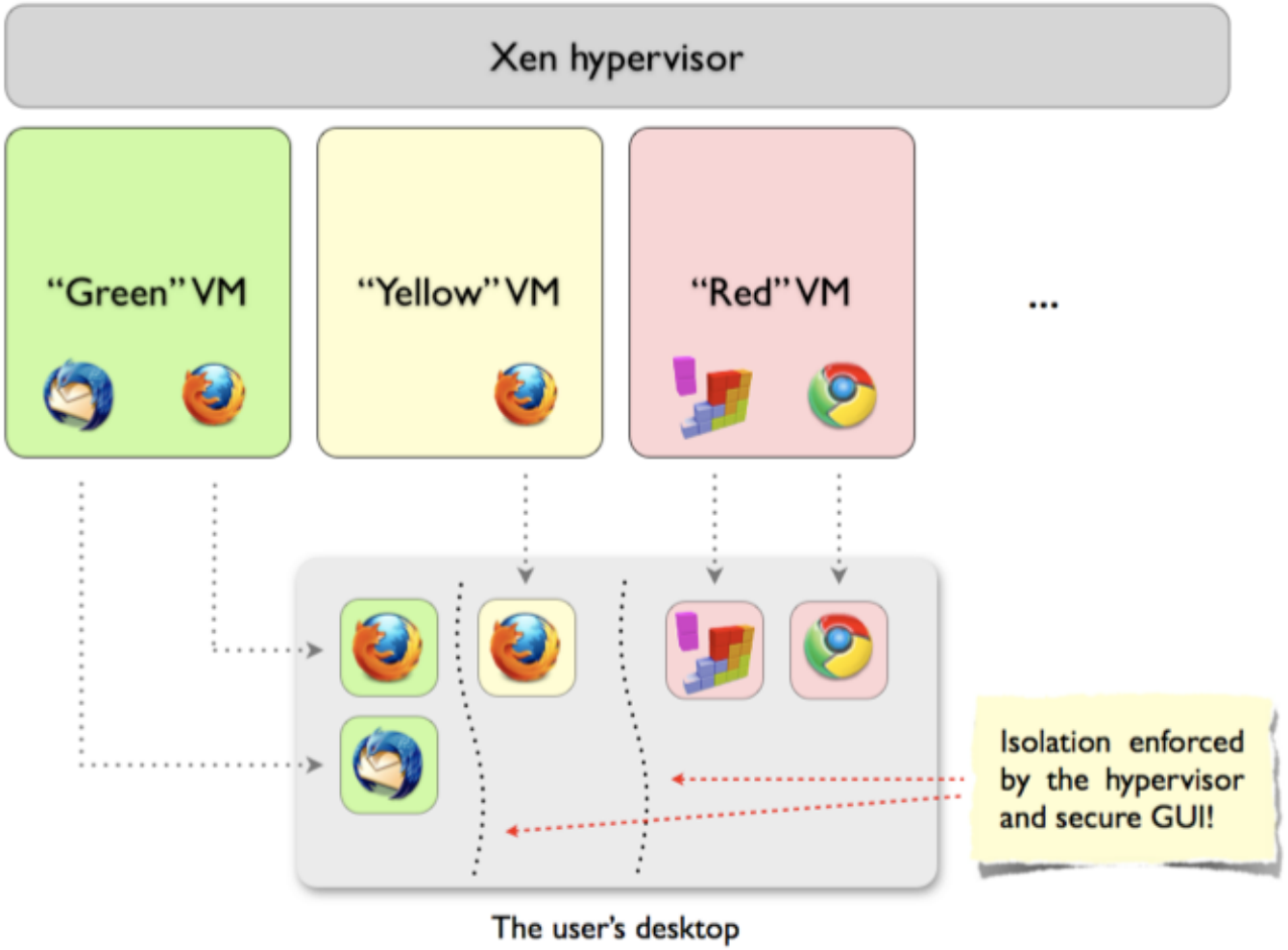
Les VMs pour les services sont les suivantes :

- Le domaine du stockage contient le nécessaire concernant la gestion de fichier : c'est lui qui contient les pilotes et la pile de gestion des media de stockage (disque dur, clef usb, CD/DVD...)

- Le domaine réseau, très exposé sur l'extérieur, contient les pilotes, les piles des protocoles, etc.

- Le domaine administratif héberge l'interface graphique «sécurisée» et le logiciel de gestion. Aucune installation supplémentaire ne doit y être installée.

Concernant les VM applicatives, elles sont toutes basés sur une image disque «maitresse» contenant un navigateur Internet (Firefox), un client mail (Thunderbird), des logiciels de bureautique (Traitement de texte, tableur, création de présentations...) ainsi que diverses applications moins utiles. Invisible Things Labs propose par défaut un domaine rouge, un domaine jaune et un domaine vert.
Le vert est destiné aux applications professionnelles : le navigateur Web pour l'intranet, et le surf professionnel, et le client email/bureautique pour les documents et échanges professionnels uniquement.
Le jaune est plutôt destiné pour les activités personnelles : «social networking», email personnel, documents divers, photos, etc.
Le rouge est destiné aux activités à risque comme le surf aléatoire, le téléchargement de films de vacances / de musique libre de droits, messagerie instantanée, jeux dans le navigateur, l'installation de logiciels de source peu sure, etc.



Bien sur, il est possible de créer plus de domaines, comme un pour le commerce en ligne, un pour la banque, réseau social seulement, etc...

Grace à la virtualisation, aucun débordement n'est possible entre les VM : si l'utilisateur ouvre un lien vérolé depuis sa messagerie instantanée, seule la VM rouge est vérolée : un potentiel pirate n'aura pas accès aux données professionnelles potentiellement critiques puisque celles-ci sont dans une autre VM. De même, si un attaquant découvre une faille dans les drivers de la carte WiFi et s'en sert pour attaquer la machine, celui-ci sera seulement cantonné à la VM réseau : il ne pourra pas se propager sur le reste du système , et n'aura d'ailleurs pas accès à plus d'informations que celles qui transitaient déjà sur le réseau.

Qubes OS implémente aussi le concept de DisposableVM : pour ouvrir un fichier peu sur (au hasard, par exemple un pdf reçu par email), il est possible de le lancer dans une VM jetable grâce à un menu disponible en faisant un clic secondaire. Une fois le document lu, la VM est supprimée. En fait, il s'agit d'une VM template de base mise en veille : son lancement est très rapide puisqu'il s'agit simplement de restaurer la mémoire RAM : il suffit de quelques secondes pour ouvrir le document sans risques !
On peut également modifier le document : les modifications seront remontées jusqu'a l'appVM où se situe vraiment le document. Petit point noir cependant, cela ne protège pas d'une faille potentielle dans le parsing des documents par le navigateur de fichier, celui-ci créant des miniatures.

Architecture de Qubes OS

L'hyperviseur est la clef de voute de la sécurité de Qubes OS puisque la sécurité est basée sur la séparation des différents domaines, il est donc absolument nécessaire de sélectionner soigneusement le logiciel de virtualisation pour assurer la robustesse et la sécurité de la solution. ITL a choisit Xen plutôt que KVM pour diverses raisons : globalement, le fonctionnement de KVM est trop proche d'un Linux «normal» pour s'approprier la confiance des développeurs.
Par exemple, Xen permet de virtualiser dans de domaines séparés les drivers et même l'émulateur d'entrées-sorties, permettant ainsi d'éloigner un maximum l'hyperviseur des brèches potentielles. Ainsi, le code au contact du monde extérieur est réduit à son strict minimum. On peut en plus vider l'hyperviseur de fonctionnalités presque superflues, comme l'ACPI pour s'assurer d'encore plus de sécurité.

Le second point critique après l'hyperviseur est le dom0 puisqu'il héberge le daemon XenStore et l'interface graphique. L'interface n'a pas été placée dans un autre domaine car de toutes façons, si un attaquant arrive à prendre possession de l'interface graphique, il a accès aux VM. Dom0 est donc strictement cantonné au XenStore daemon et à l'interface graphique : il n'a pas accès à l'extérieur (pas de réseau) et ses interfaces avec les autres VM sont réduites au minimum vital. Dom0 stocke également les clefs de chiffrement des disques des autres domaines.
Enfin, dom0 et les autres VM sont protégées de failles hardware grâce à la technologie Intel VTd. (rappelons qu'on peut «aspirer» la mémoire RAM d'un ordinateur simplement via le firewire par DMA)

Gestion du stockage

Afin d'éviter qu'une compromission de la VM contenant les drivers du stockage permette une fuite des données qu'elle gère, les disques virtuels sont chiffrés et les clefs conservées dans dom0. En fait, les disques des VM sont séparés en trois fichiers :

- Chaque VM est installée depuis une image disque commune contenant l'arborescence par défaut. Vu que cette image ne contient pas de données critiques (ça n'est que l'arborescence contenant par exemple les exécutables) elle est simplement signé : une altération de ce disque de base bloquera les processus de boot des VM.

- Si des changements doivent être faits postérieurement, ils sont sauvegardés dans une image disque séparée grâce à la fonctionnalité de Copy on Write, apparu dans la version 2.6 du noyau.

- Les répertoires /home et /var sont dans une dernière image disque, pour chaque AppVM.

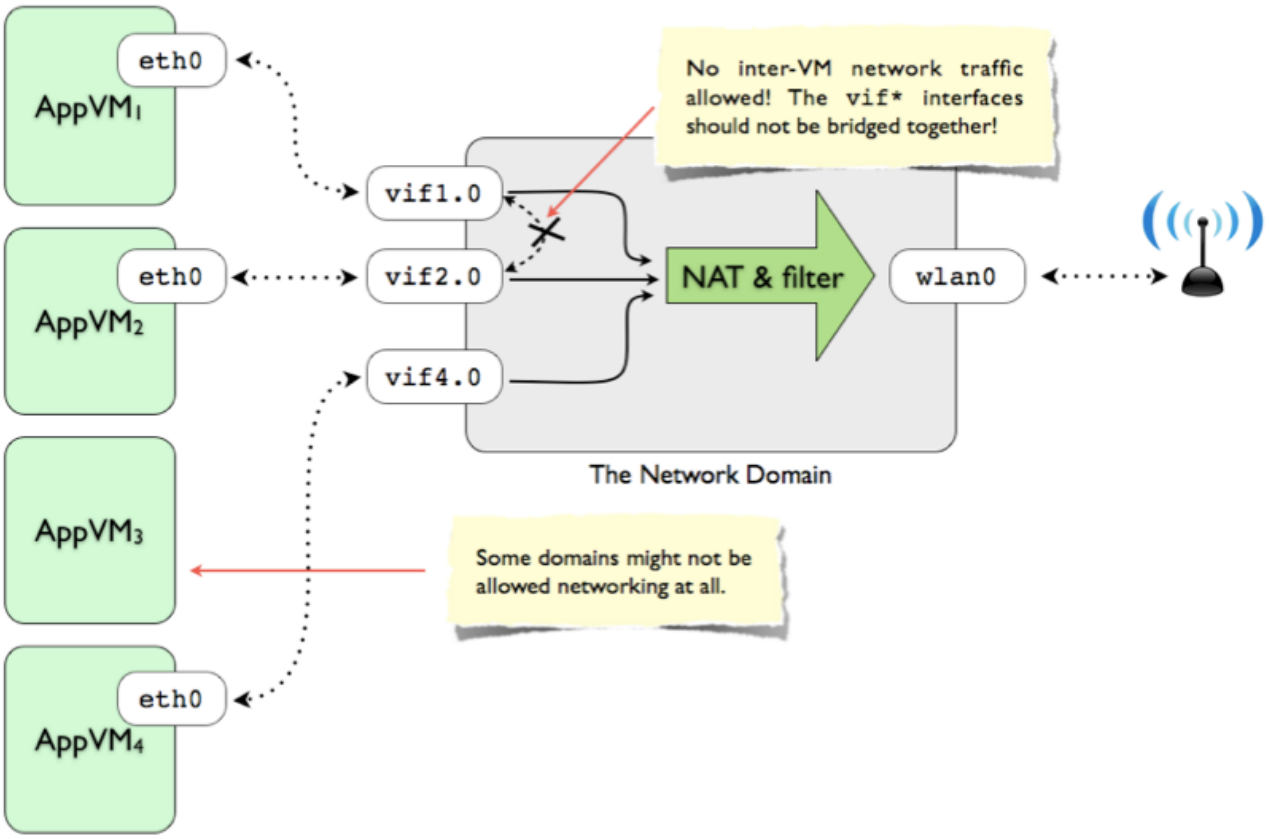
Les deux derniers disques sont bien chiffrés, avec la clef stockée sur dom0.
Le disque différence créé par COW sur le template de base est supprimé lors du reboot de l'appVM afin d'éviter de conserver de possibles altérations. En effet, le fait que les VM soient toutes basées sur la même arborescence de base permet de toutes les mettre à jour en même temps ; si l'utilisateur a besoin d'installer un logiciel en particulier ou d'une version particulière sur une VM seulement, ils doivent être installés dans /usr/local.
Le mot de passe demandé lors de l'installation sert à chiffrer la partition contenant les images disques pour éviter une attaque matérielle sur le domaine de stockage. En revanche, le domaine de stockage ayant accès direct au contrôleur de disque, celui-ci peut manipuler les images de l'hyperviseur et de dom0. C'est à ce moment là que la sécurité de l'OS se base sur la puce TPM, en particulier sur la fonction TXT (Trusted Execution Technology).
Lors du boot, la puce TPM fait une empreinte de dom0 et de l'hyperviseur, avant de les charger et de les exécuter. L'initialisation de dom0 nécessite l'obtention d'un secret auprès de la puce TPM : celle-ci ne le livrera uniquement si l'empreinte est correcte, permettant de déchiffrer la clef de chiffrement du reste du système de fichier. Ainsi, l'hyperviseur et dom0 ne peuvent être altérés sans bloquer le système si l'ordinateur est laissé sans surveillance dans un milieu hostile.
Finalement, la passphrase est demandée à l'utilisateur pour déchiffrer le fichier contenant les clefs nécessaires à la création du domaine de stockage.

À noter qu'à la rédaction de cet article, la version proposée (Beta 1) n'implémente pas encore les fonctionnalités de boot sécurisé via Intel TXT ou de stockage sandboxé, cela viendra dans les prochaines versions et sera sans aucun doute disponible dans la version finale.

Gestion du réseau

Jusqu'à maintenant, le service réseau des systèmes d'exploitation, dont les drivers et la pile, fonctionnent souvent dans l'espace noyau ou avec des privilèges très élevés : une exploitation via une pile buggée ou un driver dysfonctionnel peut donc avoir des conséquences désastreuses. Dans qubes OS, la pile réseau est éloignée hors de l'hyperviseur, ainsi que le code de l'hyperviseur concernant la gestion du réseau, évitant les attaques décrites précédemment. De plus, la technologie VT-d permet à la VM de gestion du réseau d'avoir un accès direct et exclusif au matériel. La surface d'attaque restante est une très hypothétique faille dans la pile tcp/ip de Linux.

Chaque VM est reliée au domaine réseau par des interfaces virtuelles : leurs paquets passent le domaine réseau et sont filtrées par le firewall avant de pouvoir resortir sur le monde réél. Les VM ne sont pas reliées entre elles pour éviter les interactions, voire les covert channels.



Naturellement, certaines VM peuvent ne pas avoir accès au réseau, par exemple celles contenant des fichiers très confidentiels.

Ensuite, des pare-feux individuels peuvent être appliqués à chaque VM : par exemple, la VM professionnelle ne devrait accéder qu'au serveur imaps de l'entreprise, et à l'intranet, de même pour la VM bancaire qui devrait pouvoir accéder qu'au site de la banque. En effet, les navigateurs et client web font parfois des connections à des serveurs sans nous mettre au courant : expliciter précisément les permissions permet une sécurité accrue.

Les développeurs ont préféré mettre le pare-feu dans les VM concernées pour plusieurs raisons. Premièrement, le but de ce pare-feu n'est pas de contenir un malware : si le malware a pu arriver jusque là, le niveau de protection supplémentaire ajouté par le firewall est dérisoire : ces pare-feux permettent surtout que les connections automatiques des applications, potentiellement pas écrites de manière sécurisée, puisse compromettre l'application. De plus, les placer ailleurs offre une surface d'attaque supplémentaire. Bien sur, l'utilisateur peut déplacer lui-même le firewall dans la VM de gestion du réseau, voire ajouter une VM spécialisée.

Dans l'installation par défaut, Qubes promose néanmoins un domaine destiné au firewalling, contrairement à ce qui est annoncée dans le document des spécifications (du coup, on est pas certain que cela reste dans la version finale). C'est une bonne surprise, puisque cela permet de gérer facilement les accès des VM, d'autant plus qu'il y a une petite interface graphique dédiée.

On peut d'ailleurs créer un domaine destinée à la connexion au VPN d'entreprise, ou d'accès au

réseau tor. Un tel domaine prendrait la place du domaine réseau, dans le sens où tout les domaines seraient connectées sur lui, avant de réinjecter le trafic sur le domaine réseau. Ainsi, une compromission du domaine réseau, par exemple via une faille des drivers, ne permet pas d'écouter directement le trafic réseau critique (celui d'entreprise notamment).

Attaques potentielles

Xen est l'élément le plus critique de l'architecture de Qubes OS : si un bug est trouvé et exploité dans l'hyperviseur, toute la sécurité s'écroule. A l'heure qu'il est, un seul exploit a été conçu pour Xen, et il a été créé par les créateurs de Qubes OS, et touchait en plus qu'un module optionnel. La partie critique de Xen fait entre 200'000 et 50'000 lignes de code : les développeurs essayent de s'approcher de 50'000. Le démon XenStore dans dom0 constitue aussi une surface d'attaque puisqu'il s'agit de l'interface entre les VM et l'hyperviseur, ceci est également valable pour le démon de gestion de la GUI. Les codes gérants ces interfaces ont été conçus de manière scrupuleuse et de la manière la plus épurée possible. Des bugs dans le matériels pourraient également être incriminés, par exemple une escalade de privilèges entre les rings du processeur ou des fuites de caches. Ces bugs toucheraient de toute manière tout les autres systèmes.
Ces attaques nécessitent une seule étape, néanmoins d'autres attaques sont possibles si l'attaquant a déjà réussit à mettre la main sur, par exemple, le domaine de stockage.
En effet, celui-ci possède un accès direct au matériel via VT-d : une faille potentielle dans VT-d pourrait permettre au domaine d'utiliser la DMA (pour lire des données dans la mémoire.
L'assaillant pourrait aussi reflasher le firmware des contrôleurs avec un firmware malicieux qui installerait une backdoor sur la MBR qui elle-même installerait une backdoor sur un matériel utilisé exclusivement par dom0 (comme la carte graphique ou la carte son) avant que la protection VT-d soit initialisée. Ces backdoors pourraient ensuite accéder à la mémoire de dom0 via DMA. Ce type d'attaque est très complexe à mettre en œuvre. D'autres composants matériels ou logiciels, tels la puce TPM ou la pile TCP/IP de Linux peuvent aussi ouvrir des brèches, mais ce sont souvent des composants dignes de confiance car critiques et donc bien audités.