RFC 8576 : Internet of Things (IoT) Security: State of the Art and Challenges

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 16 août 2019

Date de publication du RFC : Avril 2019

https://www.bortzmeyer.org/8576.html

Une blague très courante dit que, dans IoT ("*Internet of Things*", l'Internet des Objets), le S veut dire sécurité…C'est peu dire que la sécurité de l'Iot est mauvaise. Elle est en fait catastrophique, comme l'analyse bien Schneier dans son livre « "*Click here to kill everybody*" <https://www.bortzmeyer.org/click-here.html> ». C'est en grande partie dû à des raisons politico-économiques (les fabriquants se moquent de la sécurité notamment, parce que les failles de sécurité n'ont aucune conséquence négative **pour eux**) et en petite partie aux réelles difficultés techniques qu'il y a à sécuriser des objets qui sont **parfois** contraints en ressources (énergie électrique limitée, par exemple.) Ce nouveau RFC du groupe de recherche T2T <https://datatracker.ietf.org/rg/t2trg> ("*Thing to Thing*") de l'IRTF se penche sur la question et essaie d'identifier les questions à long terme. À lire absolument, si vous vous intéressez à la sécurité des objets connectés. Et à lire également si vous ne vous y intéressez pas car, que vous le vouliez ou non, la sécurité des objets connectés va **vous** toucher.

On ne part pas de zéro pourtant. Contrairement à ce que disent parfois les vendeurs d'objets connectés pour justifier l'insécurité abyssale de leurs produits, des solutions techniques ont été développées. (Voir par exemple cet article qui parle de la sécurité des sondes Atlas <https://labs.ripe.net/Members/kistel/ripe-atlas-probes-as-iot-devices>.) Il existe des protocoles adaptés aux objets, comme CoAP (RFC 7252[1]), une alternative légère à HTTP, qu'on peut sécuriser avec DTLS (RFC 9147). Mais il reste à les déployer.

Notre RFC suit un plan classique : il étudie d'abord le cycle de vie des objets connectés (section 2 du RFC), examine ensuite les risques (section 3), l'état de l'art (section 4), puis les défis pour sécuriser les objets (section 5), et enfin les prochaines étapes du travail nécessaire (section 6).

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7252.txt

Le terme d'Internet des Objets fait partie de ces termes pipeau qui ne veulent pas dire grand'chose. Les « objets » n'ont pas grand'chose à voir, allant d'un ordiphone plus puissant que certains ordinateurs, à des étiquettes RFID, en passant par des voitures connectées qui disposent d'électricité et de puissance de calcul considérables, et des capteurs industriels qui sont au contraire très contraints. Quant à leur connexion, elle se limite parfois au réseau local et, parfois, à envoyer toutes leurs données, aussi privées qu'elles soient, vers leur maitre dans le mythique "*cloud*". C'est le consortium privé Auto-Id <http://www.autoidlabs.org/> qui a popularisé ce terme à la fin des années 1990, pour de simples raisons marketing. À l'époque, c'était limité à des étiquettes RFID n'ayant qu'une connexion très limitée, sans rapport avec l'Internet. Certains ont suggéré de réserver le terme d'« Internet des Objets » aux objets connectés en IP mais ces appels à la rigueur terminologique n'ont en général que peu d'impact. Bref, chercher des solutions pour l'« Internet des Objets » en général n'a que peu de chances d'aboutir, vu la très grande variété de situations que ce terme recouvre.

Mais revenons au début, au cycle de vie de nos objets connectés (section 2 du RFC). Comme la variété des objets connectés est très grande, le RFC choisit de partir d'un exemple spécifique, un système de gestion de bâtiment. Ce système contrôle la climatisation, le chauffage, la ventilation, l'éclairage, la sécurité, etc. Un tel système représente de nombreux objets, dont certains, notamment les capteurs installés un peu partout, peuvent être très contraints en ressource (processeur lent, énergie fournie uniquement par des batteries, etc). Pire, du point de vue des protocoles réseau, certains de ces objets vont passer beaucoup de temps à dormir, pour économiser l'énergie, et ne répondront pas aux autres machines pendant ce temps. Et les objets seront sans doute fabriqués par des entreprises différentes, ce qui soulèvera des questions amusantes d'interopérabilité.

La figure 1 du RFC représente un cycle de vie simplifié. Je le simplifie encore ici. L'objet est successivement :

— fabriqué,
— il peut rester assez longtemps sur l'étagère, attendant un acheteur (ce qui contribue à l'obsolescence de son logiciel),
— installé et configuré (probablement par différents sous-traitants),
— mis en service,
— il va fonctionner un certain temps puis verra des évenements comme une mise à jour logicielle,
— ou des changements de sa configuration,
— à un moment, il cessera d'être utilisé,
— puis sera retiré du bâtiment (à moins qu'il soit oublié, et reste actif pendant des années sans qu'on s'en occupe),
— mais il aura peut-être droit à une reconfiguration et une remise en service à un autre endroit, recommençant le cycle,
— sinon, il sera jeté.

Les différents objets présents dans le bâtiment ne seront pas aux mêmes étapes au même moment.

Le RFC remarque que le cycle de vie ne commence pas forcément à la fabrication de l'objet physique, mais avant. Pour des objets comportant du logiciel, le cycle de vie commence en fait lorsque la première bibliothèque qui sera utilisée est écrite. Les logiciels des objets connectés ont une forte tendance à utiliser des versions anciennes et dépassées des bibliothèques, notamment de celles qui assurent des fonctions de sécurité. Les bogues ont donc une longue durée de vie.

La sécurité est une question cruciale pour les objets connectés, car ils sont en contact avec le monde physique et, si ce sont des actionneurs, ils agissent sur ce monde. Comme le note Schneier <https://www.bortzmeyer.org/click-here.html>, une bogue sur un objet connecté, ce n'est plus seulement un programme qui plante ou un fichier qu'on perd, cela peut être des atteintes physiques aux humains. Et les objets sont nombreux : pirater une machine ne donne pas beaucoup de pouvoir à l'attaquant, mais s'il arrive à trouver une faille lui permettant de pirater via l'Internet tous les objets d'un

vendeur donné, il peut se retrouver à la tête d'un botnet conséquent (c'est exactement ce qui était arrivé avec Mirai).

Quels sont les risques exactement ? La section 3 du RFC décrit les différentes menaces, une liste longue et un peu fourre-tout. Avant tout, le code peut être incorrect, bogué ou mal conçu. C'est le cas de tout logiciel (ne croyez pas un instant les commerciaux qui assurent, sans rien en savoir eux-mêmes, que « le logiciel est conforme à l'état de l'art et aux préconisations de [insérer ici le nom d'un organisme quelconque] »). Mais comme vu plus haut, les conséquences sont plus graves dans le cas des objets connectés. En outre, il y a deux problèmes logiciels qui sont davantage spécifiques aux objets connectés : les mises à jour, pour corriger les bogues, sont plus difficiles, et rarement faites, et le logiciel est globalement négligé, n'étant pas le « cœur de métier » de l'entreprise vendeuse. On voit ainsi des failles de sécurité énormes, qui n'arrivent plus dans l'informatique plus classique.

Autre raison pour laquelle la sécurité des objets connectés est difficile à assurer, le fait que l'attaquant peut avoir un accès physique aux objets (par exemple s'il s'agit d'un type d'objet vendu publiquement). Il peut le démonter, l'étudier, et acquérir ainsi des informations utiles pour le piratage d'autres objets du même type. Si, par exemple, tous les objets d'un même type partagent une clé cryptographique privée, elle pourrait être récupérée ainsi (l'objet connecté typique n'est pas un HSM). Un modèle de sécurité comme celui de la boite noire ne s'applique donc pas. Dans le futur, peut-être que les PUF seront une solution ?

La sécurité impose de mettre à jour le logiciel de l'objet régulièrement. Mais cette mise à jour ouvre elle-même des failles de sécurité. Si le processus de mise à jour n'est pas sécurisé (par exemple par une signature du logiciel), un malveillant pourra peut-être y glisser sa version du logiciel.

Ensuite, l'objet, même s'il fonctionne comme prévu, peut faire fuiter des informations privées. S'il envoie des informations à des tiers (c'est le cas de presque tous les objets conçus pour l'usage domestique) ou s'il transmet en clair, il permet la surveillance de son propriétaire. Le chiffrement est évidemment indispensable, mais il ne protège pas contre les extrémités de la communication (le sextoy connecté <https://video.passageenseine.fr/videos/watch/62631d71-9dad-4466-a424-4b4f1d378913> qui envoie les informations sur son usage au vendeur de l'objet) et, s'il n'est pas accompagné d'une authentification du partenaire avec qui on communique, ile ne protège pas contre l'homme du milieu. Une des difficultés de l'authentification est qu'il faut bien, avant la communication, avitailler l'objet en informations (par exemple les clés publiques de ses correspondants), un défi pour des objets fabriqués en masse. Avitailler une fois l'objet sur le terrain est tout aussi difficile : ces objets n'ont souvent pas d'interface utilisateur. Cela impose des solutions comme le TOFU (faire confiance la première fois, puis continuer avec le même correspondant) ou bien l'appairage (on approche deux objets, on appuie sur un bouton et ils sont désormais appairés, ils ont échangé leurs clés).

Les objets ont souvent une histoire compliquée, étant composée de l'assemblage de divers composants matériels et logiciels, parfois promenés sur de longues distances, entre beaucoup d'entreprises différentes. Une des attaques possibles est de s'insérer quelque part dans cette chaîne d'approvisionnement et d'y glisser du logiciel ou du matériel malveillant. Est-ce que quelqu'un sait vraiment ce que fait cette puce dont on a acheté des dizaines de milliers d'exemplaires à un revendeur ? Dans le cas extrême, c'est l'objet entier qui peut être remplacé par un objet apparemment identique, mais malveillant.

Les objets connectés sont souvent dans des lieux qui ne sont pas physiquement protégés. Par exemple, les capteurs sont placés un peu partout, et parfois accessibles à un attaquant. Une fois qu'on peut mettre la main sur un objet, il est difficile d'assurer sa sécurité. Des informations confidentielles, comme une clé privée, peuvent alors se retrouver entre les mains de l'attaquant. Transformer chaque objet connecté en un coffre-fort inviolable n'est évidemment pas réalistes.

Les objets communiquent entre eux, ou bien avec des passerelles les connectant à l'extérieur. Cela ouvre de nouvelles possibilités d'attaque via le routage. Les objets connectés se servent souvent de protocoles de routage non sécurisés, permettant au malveillant d'injecter de fausses routes, permettant ainsi, par exemple, de détourner le trafic vers une machine contrôlée par ce malveillant.

Enfin, il y a la menace des attaques par déni de service. Les objets sont souvent contraints en ressources et sont donc particulièrement vulnérables aux attaques par déni de service, même légères. Et les objets ne sont pas forcément victimes, ils peuvent être aussi devenir zombies et, recrutés par un logiciel malveillant comme Mirai, être complices d'attaques par déni de service comme cela avait été le cas en octobre 2016. (Un outil comme Shodan permet de trouver facilement des objets vulnérables et/ou piratés.)

Bon, ça, c'étaient les menaces. Mais on n'est pas resté les bras ballants, on a déjà des mécanismes possibles pour faire face à ces attaques. La section 4 de notre RFC décrit l'état de l'art en matière de connexion des objets connectés, et de leur sécurisation.

Déjà, il existe plusieurs protocoles pour les objets connectés, comme ZigBee, BACnet ou DALI. Mais l'IETF se focalise évidemment sur les objets qui utilisent IP, le seul cas où on puisse réellement parler d'≪ Internet des Objets ≫. IP tel qu'il est utilisé sur les ordinateurs classiques n'est pas forcément bien adapté, et des groupes ont développé des adaptations pour les réseaux d'objets (voir par exemple le RFC 4944). De même, il existe des normes pour faire tourner IP sur des tas de couches physiques différentes, comme Bluetooth (RFC 7668), DECT (RFC 8105) ou NFC (RFC pas encore publié). Au-dessus d'IP, le protocole CoAP (RFC 7252) fournit un protocole applicatif plus adapté aux objets que le HTTP classique.

Questions formats, on a également le choix. On a JSON (RFC 8259), mais aussi CBOR (RFC 8949) qui, lui, est un format binaire, sans doute plus adapté aux objets contraints. Tous les deux ont des solutions de sécurité, par exemple la famille JOSE <https://www.bortzmeyer.org/jose.html> pour signer et chiffrer les documents JSON, et son équivalent pour CBOR, CORE (RFC 8152).

Le problème de la sécurité de l'IoT est connu depuis longtemps, et ce ne sont pas les solutions techniques qui manquent, que ce soit pour protéger les objets connectés, ou pour protéger le reste de l'Internet contre ces objets. Certains de ces protocoles de sécurité ne sont pas spécifiques aux objets connectés, mais peuvent être utilisés par eux, c'est le cas de TLS (RFC 8446). Une excuse classique des fabricants d'objets connectés pour ne pas sécuriser les communications avec TLS est le caractère contraint de l'objet (manque de ressources matérielles, processeur, mémoire, énergie, etc). Cet argument peut jouer pour des objets vraiment contraints, des capteurs bon marché disséminés dans l'usine et ne fonctionnant que sur leur batterie mais beaucoup d'objets connectés ne sont pas dans ce cas, et ont largement les moyens de faire tourner TLS. Quand on entend des fabriquants de télévisions connectées ou de voitures connectées expliquer qu'ils ne peuvent pas utiliser TLS car ce protocole est trop coûteux en ressources, on rit ou on s'indigne car c'est vraiment un argument ridicule ; une télévision ou une voiture ont largement assez de ressources pour avoir un processeur qui fait tourner TLS. (Je n'utilise que TLS et SSH pour communiquer avec un Raspberry Pi 1, avec son processeur à 700 MHz et sa consommation électrique de 2 W.)

Outre les protocoles, la sécurité repose sur des règles à suivre. La section 4.3 liste les règles formalisées existantes. Ainsi, GSMA a publié les siennes <http://www.gsma.com/connectedliving/future-iot-networks/iot-security-guidelines>, BITAG <https://www.bitag.org/> également <https://www.bitag.org/report-internet-of-things-security-privacy-recommendations.php>, le DHS étatsunien s'y est mis <https://www.dhs.gov/sites/default/files/publications/Strategic_Principles_for_Securing_the_Internet_of_Things-2016-1115-FINAL....pdf>, l'ENISA aussi <https://www.enisa.europa.eu/publications/ics-scada-dependencies> et notre RFC liste de nombreux autres documents. Si les fabriquants d'objets connectés ne sont pas au courant, ce n'est pas faute d'information, c'est bien de la mauvaise volonté !

C'est d'autant plus grave que, comme l'a illustré le cas de Mirai, les objets connectés non-sécurisés ne sont pas un problème que pour le propriétaire de l'objet, mais peuvent également toucher tout l'Internet. Il est donc logique que beaucoup de voix s'élèvent pour dire qu'il faut arrêter de compter sur la bonne volonté des fabricants d'objets connectés, qui ont largement démontré leur irresponsabilité, et commencer à réguler plus sévèrement. (C'est par exemple une demande du régulateur étatsunien FCC <https://docs.fcc.gov/public/attachments/DOC-342761A1.pdf>.)

Cette disponibilité de très nombreuses solutions techniques ne veut pas dire que tous les problèmes sont résolus. La section 5 du RFC fait ainsi le point sur les défis qui nous restent, et sur lesquels chercheu[Caractère Unicode non montré [2]]r[Caractère Unicode non montré]se[Caractère Unicode non montré]s et ingénieur[Caractère Unicode non montré]e[Caractère Unicode non montré]s devraient se pencher. D'abord, certains objets sont contraints en ressources (pas tous, on l'a vu), comme détaillé dans le RFC 7228. L'Internet est un monde très hétérogène, connectant des machines ayant des ressources très diverses, via des réseaux qui ont des capacités hautement variables. Pour ces objets contraints (qui sont une partie seulement des « objets », une caméra de vidéo-surveillance n'est **pas** un objet contraint), il est raisonnable de chercher à optimiser, par exemple la cryptographie. Ainsi, la cryptographie à courbes elliptiques (RFC 8446) demande en général moins de ressources que RSA.

Les attaques par déni de service sont un autre défi pour les objets connectés, qui disposent de peu de ressources pour y faire face. Des protocoles qui permettent de tester qu'il y a une voie de retour <https://www.bortzmeyer.org/returnability.html> ("*return routability*" ou "*returnability*") peuvent aider à éviter certaines attaques que des protocoles sans ce test (comme le DNS ou comme d'autres protocoles fondés sur UDP) rendent facile. C'est pour cela que DTLS (RFC 9147) ou HIP (RFC 7401) intègrent ce test de réversibilité. Évidemment, cela n'aide pas pour les cas de la diffusion, ou bien lorsque le routage est contrôlé par l'attaquant (ce qui est souvent le cas dans les réseaux « mesh ».) Autre protection, qui existe par exemple dans HIP : forcer l'initiateur d'une connexion à résoudre un problème, un « puzzle », afin d'éviter que les connexions soient « gratuites » pour l'initiateur. La principale limite de cette solution est qu'elle marche mal si les machines impliquées ont des capacités de calcul très différentes (un objet contraint contre un PC). Il y a également le cas, non mentionné par le RFC, où l'attaquant dispose d'un "*botnet*" et ne « paie » donc pas les calculs.

L'architecture actuelle de l'Internet n'aide pas au déploiement de certaines solutions de sécurité. Ainsi, un principe de base en sécurité est d'avoir une sécurité de bout en bout, afin de ne pas dépendre d'intermédiaires malveillants ou piratés, mais c'est rendu de plus en plus difficile par l'abus de "*middleboxes*", qui interfèrent avec beaucoup de comunications. On est donc forcés d'adapter la sécurité à la présence de ces "*middleboxes*", souvent en l'affaiblissant. Par exemple :

— Il faut parfois partager les clés avec les "*middleboxes*" pour qu'elles puissent modifier les paquets, ce qui est évidemment une mauvaise pratique,
— Le chiffrement homomorphe peut aider, en permettant d'effectuer certaines opérations sur des données chiffrées, mais toutes les opérations ne sont pas possibles ainsi, et les bibliothèques existantes, comme SEAL <https://www.microsoft.com/en-us/research/publication/simple-encrypte>, n'ont pas les performances nécessaires,
— Remonter la sécurité depuis le niveau des communications (ce que fait TLS) vers celui des données échangées pourrait aider. C'est ce que font COSE (RFC 8152), JOSE <https://www.bortzmeyer.org/jose.html> (RFC 7520) ou CMS (RFC 5652).

Une fois déployés, les objets connectés vont rester en fonctionnement des années, voire des décennies. Il est donc crucial d'assurer les mises à jour de leur logiciel, ne serait-ce que pour réparer les failles de sécurité qui ne manqueront pas d'être découvertes, qu'elles soient dans le code ou dans les algorithmes utilisés. Par exemple, si les promesses des ordinateurs quantiques se concrétisent un jour, il faudra jeter RSA et les courbes elliptiques (section 5.8 du RFC).

2. Car trop difficile à faire afficher par LaTeX

Mais assurer des mises à jour sûres n'est pas facile, comme le note Bruce Schneier <https://www.schneier.com/essays/archives/2014/01/the_internet_of_thin.html>. C'est que le processus de mise à jour, s'il est insuffisamment sécurisé, peut lui-même servir pour une attaque, par exemple en envoyant du code malveillant à un objet trop naïf. Et puis comment motiver les vendeurs à continuer à fournir des mises à jour logicielles des années après que le dernier exemplaire de ce modèle ait été vendu ? Capitalisme et sécurité ne vont pas bien ensemble. Et il se peut tout simplement que le vendeur ait disparu, que le code source de l'objet ne soit plus disponible, et qu'il soit donc impossible en pratique de développer une mise à jour. (D'où l'importance, même si le RFC ne le dit pas, du logiciel libre.) Enfin, si la mise à jour doit être effectuée manuellement, il est probable qu'elle ne sera pas faite systématiquement. (Un rapport de la FTC états-unienne détaille <https://www.ftc.gov/news-events/press-releases/2015/01/ftc-report-internet-things-urges-companies-adopt-b> également ce problème.)

Mais les mises à jour automatiques posent également des tas de problèmes. Par exemple, pour des ampoules connectées (une idée stupide <https://mashable.com/article/ge-light-bulb-instructional>, mais le monde de l'IoT est plein d'idées stupides), il vaut mieux mettre à jour leur logiciel le jour que la nuit. Et il vaut mieux que toutes les ampoules ne soient pas mises à jour en même temps. Et les mises à jour supposent que le système ait été conçu pour cela. Par exemple, en cryptographie, il est souvent nécessaire de remplacer les algorithmes cryptographiques qui ont été cassés avec le temps, mais beaucoup d'objets connectés utilisent des systèmes cryptographiques mal conçus, qui n'ont pas d'agilité cryptographique. (Au passage, la section 5.8 du RFC traite le cas des possibles futurs ordinateurs quantiques, et des conséquences qu'ils auront pour la cryptographie. Les objets connectés peuvent rester actifs de nombreuses années, et il faut donc penser loin dans le futur.) Ces points, et beaucoup d'autres, avaient été traités dans un atelier de l'IAB, qui avait fait l'objet du RFC 8240. À l'IETF, le groupe de travail SUIT <https://datatracker.ietf.org/wg/suit/> développe des mécanismes pour aider les mises à jour (mais qui ne traiteront qu'une petite partie du problème).

Rapidement dépassés, les objets connectés posent également des problèmes de gestion de la fin de vie. Au bout d'un moment, le vendeur va arrêter les différentes fonctions, comme les mises à jour du logiciel ou, plus radicalement, comme les serveurs dont dépend l'objet <https://www.bortzmeyer.org/objets-esclaves-federez.html>. Cet arrêt peut être volontaire (l'objet n'intéresse plus le vendeur, qui est passé à d'autres gadgets) ou involontaire (vendeur en faillite). Le RFC note qu'une des voies à explorer est la continuation de l'objet avec du logiciel tiers, qui ne dépend plus de l'infrastructure du vendeur. Bien des ordiphones ont ainsi vu leur vie prolongée par CyanogenMod, et bien des routeurs ont bénéficié d'OpenWrt. (D'où l'importance de pouvoir installer ce logiciel tiers, ce qu'interdisent beaucoup de vendeurs.)

Une autre question intéressante de sécurité posée par les objets connectés est la vérification de leurs capacités réelles et de leur comportement effectif. L'acheteur peut avoir l'impression qu'il est le propriétaire de l'objet acheté mais cet objet est suffisamment complexe pour que l'acheteur ne soit pas au courant de tout ce que l'objet fait dans son dos. Le vrai maitre de l'objet est alors le vendeur, qui continue à communiquer avec l'engin connecté. C'est ainsi que des malhonnêtes comme Lidl <https://www.numerama.com/tech/525214-monsieur-cuisine-connect-micro-cache-android-non-securise-le> html> ou Google <https://www.zdnet.fr/actualites/nest-guard-google-a-discretement-instal> htm> avaient installé des micros dans des objets qu'on installe chez soi, et évidemment sans le dire à l'acheteur. Et encore, un micro est un appareil physique, qu'un examen attentif (avez-vous vérifié tous les objets connectés chez vous ?) peut détecter. Mais savoir ce que raconte l'objet connecté à son maitre est plus difficile. Peu d'utilisateurs ont envie de configurer un routeur local, et d'y faire tourner tcpdump pour voir le trafic. Et encore, ce trafic peut être chiffré et l'acheteur (qui, rappelons-le, n'est pas le véritable propriétaire de l'objet, puisqu'il n'a quasiment aucun contrôle, aucune information) n'a pas les clés.

Le problème de fournir des informations à l'utilisateur n'est pas trivial techniquement. Beaucoup d'objets connectés n'ont pas d'interface utilisateur où afficher « je suis en train d'envoyer plein de

données sur vous à mon maitre ≫. Une solution partielle serait une description des capacités de l'engin, et de ses communications, dans un fichier MUD ("*Manufacturer Usage Description*", RFC 8520). Ceci dit, vu le niveau d'éthique dans le monde de l'IoT, gageons que ces fichiers MUD mentiront souvent, notamment par omission.

Puisqu'on a parlé de vie privée, c'est l'occasion de rappeler que l'IoT est une grave menace pour cette vie privée. Le RFC note que, dans le futur, nous serons peut-être entourés de centaines d'objets connectés. (Malheureusement, le RFC parle surtout des risques dus à des tiers qui observeraient le trafic, et très peu des risques dus aux vendeurs qui récoltent les données.) L'IoT permet une intensification considérable du capitalisme de surveillance.

Bref, la situation est mauvaise et, s'il y a en effet quelques progrès (on voit moins souvent des mots de passe identiques pour tous les objets d'un type), ils sont largement annulés par de nouveaux déploiements.