Etat de l'art

Sur le fonctionnement et l'architecture générale d'un botnet

BELLEC – SORAGNA - FERNANDEZ DEL BUSTO JIMENEZ – ZANELLA

Sommaire

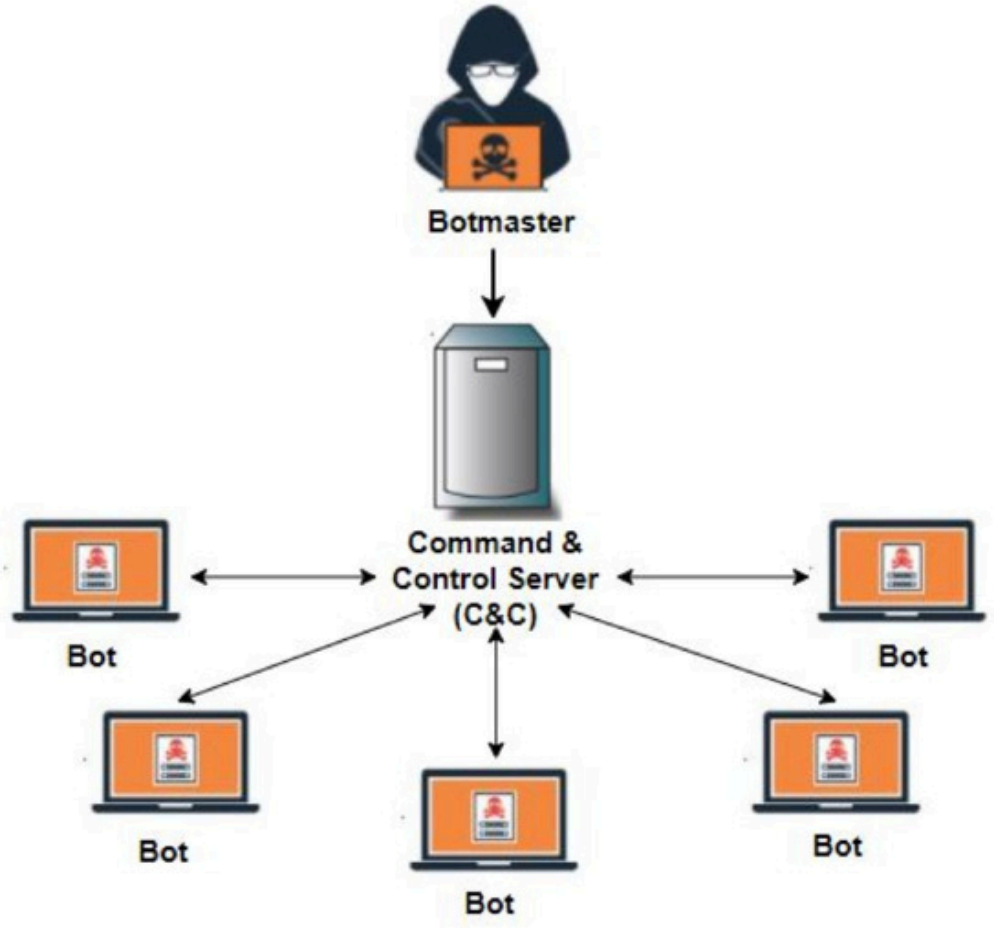
La formulation de la problématique

Dans le cadre de notre formation d'ingénieure spécialisée en cybersécurité, nous sommes amenés à réaliser un botnet.

À travers ce projet, nous mettrons en pratique notre savoir-faire en développement C, en implémentation de structures de données, en programmation système et réseaux, et en gestion de projet.
L'objectif de ce projet est de concevoir et de développer un botnet avec une architecture composée :

- d'un serveur de commande et de contrôle (C&C) : ce serveur central est responsable de la coordination des bots et de l'envoi des instructions
- d'une multitude de bots : des clients légers qui s'enregistrent auprès du serveur C&C, exécutent des commandes reçues de ce serveur et lui renvoient un rapport sur ces exécutions.

Point définition:
Un botnet est un groupe d'ordinateurs ou de dispositifs sous le contrôle d'un attaquant, utilisé pour mener des activités malveillantes contre une victime ciblée.
Le master (aussi C2 ou C&C pour Command and Control), est un serveur qui envoie les commandes aux ordinateurs infectés appelés bot ou slave. Voici un schéma explicatif :



Les botnets ont plusieures fonctions :

Ecrire et lire des données sur l'ordinateur infecté
Exfiltrer des informations sensibles comme des mots de passe de navigateurs
Exécuter des commandes sur l'ordinateur infecté
Lancer une attaque DDos

Recherche et sélection des publications

Nous avons préalablement recherché les codes sources d'anciens botnets afin de comprendre leur fonctionnement. Nous avons trouvé ce répertoire qui liste une centaine de botnet avec leur code source :
https://github.com/maestron/botnets/tree/master

Notamment ce botnet : https://github.com/maestron/botnets/tree/master/VirusPack/spybot1.4
Nous avons ainsi une vision beaucoup plus claire sur les commandes à implémenter.

Ces botnets sont complets, ce qui est hors du cadre de notre projet.

Pour le C2, il devra implémenter une structure de données à choisir pour stocker les bots. D'après les différents documents que nous avons pu analyser, deux choix s'offrent à nous :

- Une liste chaînée
- Une hash table

Pour la liste chaînée, l'avantage principal est qu'elle est facile à implémenter et que l'ajout et suppression est rapide en début/fin de liste. Malheureusement, la recherche est lente et la gestion mémoire devient complexe si le nombre de bots à gérer est grand.
Cette solution est donc adaptée si le nombre de bots est faible (< 1000).

Pour la hash table, l'accès, la recherche, l'insertion et la suppression sont très efficaces. Il est d'ailleurs recommandé d'utiliser une hash table pour la gestion d'un grand nombre de bots (10 000 +). Nous allons donc nous tourner vers cette solution. Nous devrons faire attention à la bonne gestion des collisions.

Communications et méthodes de dissimulation

Après avoir été infecté, si l'architecture n'est pas basée sur le principe du P2P, un nouveau bot doit se connecter au serveur de contrôle et de commande (C&C) afin de rejoindre le botnet. D'après nos recherches, la majorité des botnets se basent principalement sur les protocoles HTTPS et DNS. Les techniques de Domain Fronting et de DNS Tunneling permettent de garder discrètes ces communications Bot – Serveur C&C.

Les méthodes de dissimulation utilisées

HTTPS et Domain Fronting - L'utilisation de HTTPS sur le port 443 permet de cacher le trafic malveillant parmi le trafic web légitime. De plus, le protocole TLS peut être utilisé pour chiffrer les communications. Le bot envoie alors des requêtes GET ou POST périodiquement au serveur C&C dans le but de récupérer des instructions.

La technique du « Domain fronting » consiste à utiliser le header HTTP Host et le « Server Name Indication » (SNI, un champ d'extension du protocole TLS) pour rendre le trafic légitime aux yeux du pare-feu de la machine infectée. Ceci en choisissant un nom de domaine connu comme étant fiable et hébergé sur le même CDN que le domaine malveillant dans le champ SNI de la requête HTTPS. Le serveur légitime reçoit puis réoriente alors la requête vers le domaine malveillant indiqué dans le header Host contenu dans l'entête HTTP de la requête. Le serveur intermédiaire sert de "couverture" au serveur attaquant, permettant à ce dernier de dissimuler le fait que la requête lui est destinée ou qu'il en est le véritable expéditeur.

DNS et DNS Tunneling - Le protocole DNS est parfois utilisé dans le cadre de la mise en place de botnets car les données contenues dans les paquets DNS sont la plupart du temps jugées fiables par les pare-feu. La technique qui consiste à utiliser le protocole DNS pour transmettre des données est appelée « DNS Tunneling ». Dans l'idée, un attaquant possède un domaine qu'il contrôle. Il configure ensuite un serveur DNS personnalisé pour ce domaine. Ainsi, toutes les requêtes vers un sous-domaine de ce nom de domaine seront envoyées directement au serveur de l'attaquant. Si une requête DNS vers le domaine de l'attaquant est préfixée d'informations encodées en base64, le serveur DNS de l'attaquant reçoit le paquet DNS et est en capacité d'extraire et de décoder les données. Une transmission d'informations avec un serveur malveillant peut ainsi avoir lieu sans mettre en alerte ni l'antivirus, ni le pare-feu de la machine infectée.

Idées d'implémentations

Lors de l'envoi d'une requête HTTP ou DNS, les données sont encapsulées dans un segment TCP ou UDP, puis dans un segment IP, le tout enfin encapsulé dans une trame Ethernet. En C, les sockets utilisent les protocoles TCP et UDP. Il nous faudra alors encapsuler les données relatives aux requêtes HTTP et DNS de notre botnet à l'intérieur de paquets TCP et UDP.

Autrement, des librairies pour le langage C permettant de faire tourner des serveurs HTTP semblent exister, il est notamment possible de citer la librairie GNU Libmicrohttpd.

En ce qui concerne les requêtes HTTPS et la technique du domain Fronting, nous pourrions par exemple utiliser des sockets et la librairie OpenSSL pour utiliser TLS afin de chiffrer nos communications bot-serveur à l'aide d'un certificat. De plus, modifier l'extension SNI du protocole TLS semble possible grâce à la fonction *SSL_set_tlsext_host_name()* de la librairie C openssl. Nous pourrions ainsi essayer d'implémenter la technique du domain fronting exposée précédemment. Il semblerait néanmoins que pour que la technique fonctionne, les domaines contenus dans le header Host de HTTP et le champ d'extension SNI de TLS soient hébergés sur le même CDN (Content Delivery Network). Il nous reste donc à effectuer des recherches sur ce sujet.

La technique du DNS Tunneling semble quant à elle complexe à mettre en œuvre dans le cadre de notre projet : nous ne possédons pas de nom de domaine ni les connaissances nécessaires pour mettre en place et administrer un serveur DNS.

Ainsi, il nous semble préférable de nous concentrer sur le fait de garantir la discrétion des communications bot-serveur en utilisant le protocole HTTP, potentiellement couplé à un chiffrement à l'aide de TLS (HTTPS). Il nous sera alors possible d'essayer d'implémenter la technique du domain fronting en modifiant le champ SNI du protocole TLS avant d'envoyer nos requêtes, comme exposé précédemment, même s'il nous reste encore à effectuer des recherches sur les CDN et leur réelle importance dans le fonctionnement de la technique.

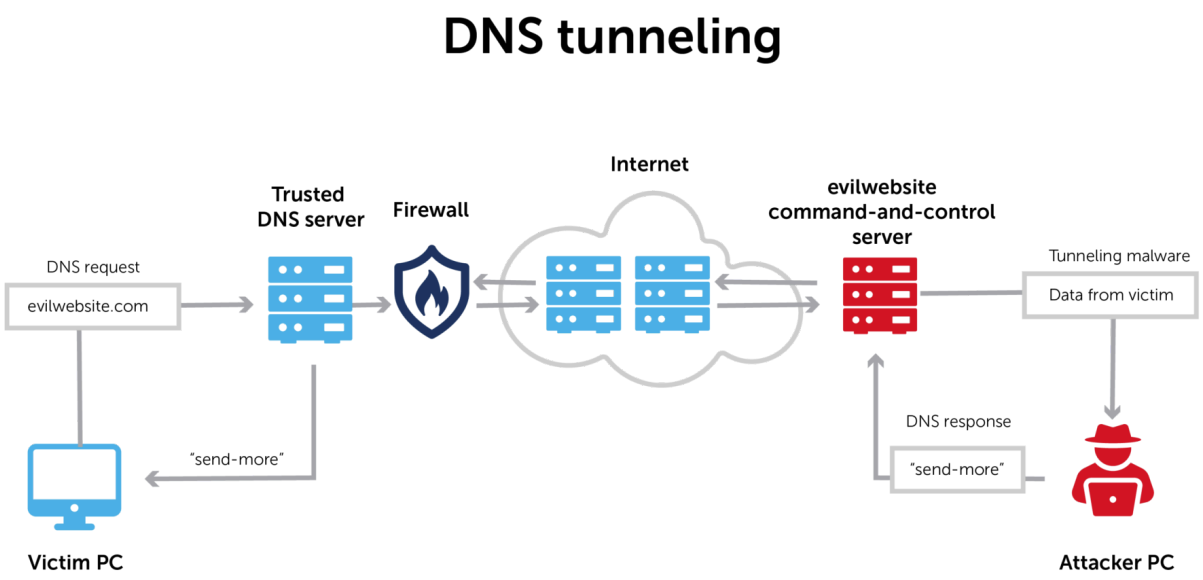


Schéma - https://bluecatnetworks.com/blog/why-you-should-pay-attention-to-dns-tunneling/

Méthodes de persistance

Comme nous l'avons vu, les botnets sont souvent déployés sur au moins plusieurs centaines de machines afin qu'ils soient performants et puissent accomplir leur tâches. Mais sans persistance, un botnet devient très limité dans le temps - il faudrait alors constamment infecter de nouvelles machines (ou les mêmes) à chaque attaque. Ce processus est évidemment long et compliqué, et n'est pas praticable si le botnet nécessite un grand nombre de machines ou une attaque qui doit pouvoir être faite à n'importe quel moment.

La persistance d'un programme sur une machine revient à permettre à un acteur d'activer le programme une seule fois, et ensuite de garantir (à des niveaux différents) que le programme reste actif malgré des changements sur la machine.

Il convient de s'assurer que le programme soit exécuté sur la machine même après un redémarrage, une mise à jour, ou même dans certains cas après une tentative de retirer le programme de la machine; et ce le plus rapidement et discrètement possible.

En règle générale, il existe de nombreuses manières d'assurer la persistance - en se basant sur des techniques simples ou complexes selon l'effet attendu.

Il existe un guide intéressant qui reprend les éléments de la persistance de malware sur des machines, disponible ici: https://lsecqt.github.io/Red-Teaming-Army/malware-development/malware-development-with-c---basic-persistence/

Analysons quelques-unes des techniques principales dévoilées par le MITRE ATT&CK:

Utiliser des comptes utilisateurs

Dans des contextes d'entreprise, qui ont souvent à gérer plusieurs dizaines voire centaines de comptes utilisateurs, une technique répandue revient à s'introduire dans un compte utilisateur en particulier, puis de manipuler les documents de configuration pour créer de nouveaux comptes, accéder à d'autres comptes existants, ou d'augmenter le niveau de permissions accordé à l'utilisateur.

Ce contexte multi-utilisateur est propice aux attaques, puisque souvent associé à un grand nombre de machines potentiellement infectables.

Utiliser le scheduler

Les systèmes d'exploitations courants (Windows, Linux, Mac, …) proposent en général un "scheduler" - un système qui permet de définir des événements à l'avance qui seront exécutés à une date et heure donnée, ou selon des conditions prédéfinies.

Par exemple, le scheduler peut automatiser le lancement d'un programme lorsque l'utilisateur se connecte à sa machine, ou se lancer chaque midi. Par la même occasion, on peut spécifier des préconditions - comme s'assurer que la machine est connectée à internet, ou que l'utilisateur soit absent de sa machine depuis un temps donné.

Cette technique requiert cependant souvent des droits administrateur selon le type de tâche programmée, mais ceci peut être contourné à l'aide de protocoles possédant des failles, tels que RPC ou via le file printer sharing.

Scripts d'initialisation, de boot / login, et processus systèmes

Windows notamment permet de mettre en place des processus qui seront lancés au démarrage de la machine ou lorsque l'utilisateur se connecte à son compte.

Ce système se base sur le registre Windows (Windows Registry), c'est un des systèmes les plus répandus car très puissant, facile à mettre en place, et souvent utilisé par des programmes non-malveillants. Cependant, son plus grand défaut est le manque de secret - l'application peut être facilement retrouvée, et donc à fortiori détectée et retirée du système pour peu que l'utilisateur prête attention à la sécurité de son système.

Compromettre un programme existant

Une autre méthode très répandue est l'utilisation d'un programme jugé connu et sécuritaire pour installer une application malveillante et la maintenir sur le système. L'avantage de ce système est que l'utilisateur ne pensera pas forcément à une faille de sécurité dans l'application, et le système est bien plus discret.

Cependant, ce modèle est aussi plus compliqué à mettre en place, puisqu'il faut alors trouver une faille dans un programme et savoir comment se l'approprier dans le cadre du malware. Par ailleurs, l'attaque est alors limitée aux machines possédant ce logiciel (hors utilisation de plusieurs techniques d'infiltration).

Exécution par événement (Hijack)

Cette méthode repose sur l'utilisation des services de publish / subscribe (notamment sur Windows) pour lancer un programme selon certains événements systèmes, qui sont écoutés par une autre application et déclenche l'exécution du malware.

Virtualisation

Les systèmes modernes possèdent presque tous un moyen de virtualisation capable de créer des machines virtuelles ou conteneurs. Ces environnements virtuels sont détachés du système hôte - l'avantage est alors clair : l'application malveillante devient alors difficilement détectable par des antivirus.

Cette méthode cependant doit être accompagnée d'une autre méthode pour instancier la machine virtuelle ou le conteneur - rendant la virtualisation une technique dite de dissimulation.

Extensions de navigateur

Les utilisateurs jouissent en général de la possibilité d'installer des extensions dans leurs navigateurs, qui sont exécutées par celui-ci.

Un malware contenu dans le navigateur est intéressant, puisque le navigateur est généralement une application fréquemment utilisée, et sur laquelle on porte rarement une attention particulière quant à la sécurité.

Cependant, ce modèle repose sur la création d'une application d'extension de navigateur, qui sort du cadre de notre projet.

Utilisation des BITS job

Le service BITS (Background Intelligent Transfer Service) est un service Windows qui est utilisé par Windows pour télécharger, exécuter, ou nettoyer des applications.
C'est un système souvent utilisé par des applications connues (ou "régulières") pour faire des téléchargements en asynchrone HTTP.

Les méthodes d'utilisation dans le cadre d'un malware en C sont de passer par PowerShell pour créer le BITS job.
https://learn.microsoft.com/en-us/windows/win32/bits/using-windows-powershell-to-create-bits-transfer-jobs

Cependant, cette méthode nécessite déjà une interaction avec la machine pour qu'il soit possible de créer le BITS job. Ainsi, cette technique est utile lorsqu'un premier malware est déjà installé et que l'on souhaite installer d'autres applications ou mettre à jour le système actuel de manière dissimulée.

DLL Search Order Hijacking

Sur Windows, il existe des fichiers nommés DLL (Dynamic Link Library), qui sont des bibliothèques partagées par des applications. Il en existe qui sont installés par défaut, mais certaines applications peuvent nécessiter des bibliothèques additionnelles.

La lecture de ces DLL est faite de manière séquentielle - il est alors possible de créer son propre DLL qui est appelé par une application connue, et de le placer avant le "vrai" DLL. Le malware est alors exécuté sur le système.

On remarque qu'ici aussi, cette méthode repose sur une intrusion faite au préalable sur le système, afin de remplacer le DLL au moment opportun.

La synthèse

Dans le cadre de ce projet, nous sommes amenés à établir une communication entre un serveur C&C et des clients infectés. L'objectif d'un botnet est avant tout de passer inaperçu et d'être persistant. Pour cela, nous avons identifié deux méthodes de dissimulation: le domain fronting et le DNS Tunneling, et plusieurs moyens d'assurer sa persistance.

Le Domain Fronting repose sur l'utilisation du protocole HTTPS sur le port 443 afin que le trafic malveillant soit confondu avec le trafic web légitime. Dans cette technique, le header HTTP Host et le Server Name Indication sont utilisés pour rendre le trafic invisible aux pare-feu en passant par un serveur intermédiaire dont le nom de domaine est hébergé sur le même CDN que le domaine malveillant. Cette technique peut être implémentée en C au moyen de sockets et des librairies OpenSSL et GNU Libmicrohttpd.

Quant à elle, la technique de DNS Tunneling repose dans le contrôle d'un domaine. Les requêtes préfixées d'informations encodées en base64 vers des sous-domaines de ce dernier sont directement récupérées par l'attaquant.Cette méthode permet de transmettre des données sans alerter les systèmes de sécurité.

Quant aux méthodes de persistance, elles supposent les machines client déjà infectées. Elles garantissent que le malware reste actif même après des redémarrages ou des mises à jour.

Des fonctionnalités du système de la machine peuvent être exploitées à cet effet. Quelques exemples sont le scheduler qui permet de définir des préconditions à l'exécution d'une tâche, et les scripts d'initialisation qui permettent de lancer des processus au démarrage du système. Pour utiliser le scheduler, il peut être nécessaire d'avoir des droits administrateur alors que les scripts d'initialisation sont plus simples à mettre en place mais moins discrets.

Autres méthodes supposent de compromettre un programme existant, reposant ainsi sur la connaissance d'une faille et son exploitation, ou sur la mise en place d'une application comme façade (souvent des extensions de navigateur). Ceci représente un travail considérable.

Nous avons également identifié l'utilisation des BITS Jobs qui réalise des tâches en arrière-plan et le DDL Search Order Hijacking. Ces méthodes nécessitent une interaction préalable avec la machine pour soit créer le job en utilisant PowerShell par exemple, soit un fichier DLL placé avant le fichier légitime.

Pour assurer une persistance efficace, il est souvent nécessaire de combiner plusieurs méthodes. Par exemple, utiliser le scheduler pour des tâches périodiques et les scripts d'initialisation pour garantir le lancement au démarrage.

La conclusion

À travers cet état de l'art, nous avons exploré les principaux composants techniques nécessaires à la conception d'un botnet, en mettant en lumière les choix d'implémentation, les méthodes de communication furtive, ainsi que les mécanismes de persistance.

La conception d'un serveur C&C efficace passe par un choix judicieux de structures de données. Après analyse comparative entre les listes chaînées et les tables de hachage, notre choix s'est porté sur cette dernière pour sa performance en insertion, suppression et recherche, particulièrement adaptée à la gestion d'un grand nombre de bots.

Concernant les communications, deux techniques principales ont retenu notre attention : le domain fronting et le DNS tunneling. Bien que toutes deux permettent de dissimuler le trafic entre les bots et le serveur, la mise en œuvre du DNS tunneling s'est révélée hors de notre portée dans le cadre de ce projet, du fait des contraintes techniques (nom de domaine, serveur DNS dédié). À l'inverse, l'utilisation du protocole HTTPS, combinée à la technique de domain fronting via la bibliothèque OpenSSL, apparaît comme une voie plus accessible et suffisamment discrète pour nos besoins.

En synthèse, ce travail d'analyse nous a permis d'identifier les solutions techniques les plus pertinentes au regard des objectifs du projet et des contraintes associées. Il constitue ainsi une base solide pour l'étape suivante, à savoir la conception et le développement du botnet, en orientant nos choix vers des techniques à la fois fonctionnelles et réalistes dans un contexte pédagogique.

Ressources

Information gathering : https://github.com/hackirby/skuld/ is a Go-written Malware targeting Windows systems, extracting User Data from Discord, Browsers, Crypto Wallets and more, from every user on every disk.

https://github.com/Slowerzs/ThievingFox

Article sur le Header HTTP Host et l'extension SNI du protocole TLS : https://techcommunity.microsoft.com/blog/azurenetworkingblog/understanding-http-host-header-and-sni/3880591

Article sur les protocoles TCP, UDP, le modèle TCP / IP et l'encapsulation des données : https://www.cyberview.fr/les-protocoles

Librairie C GNU libmicrohttpd :
https://www.gnu.org/software/libmicrohttpd/