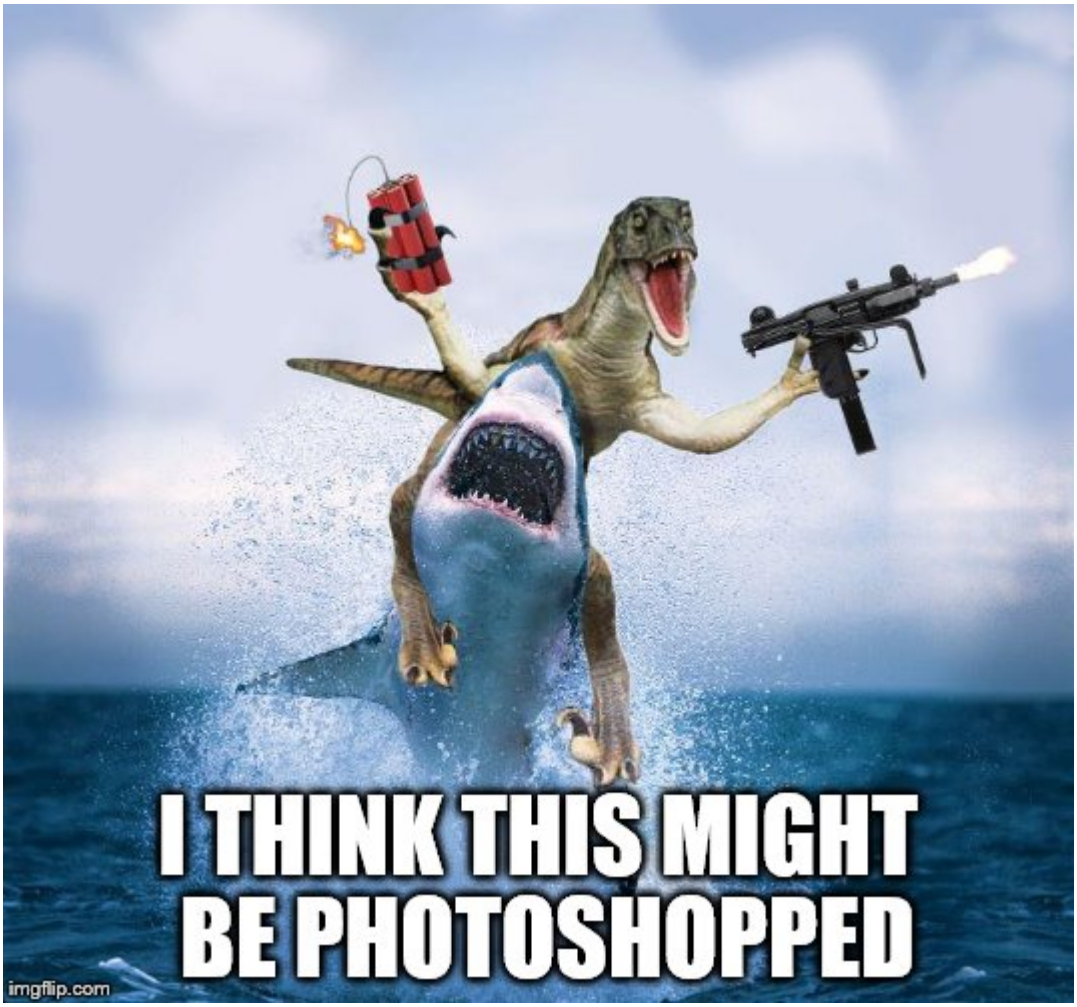
Manuel Complet de Velociraptor

Logo Velociraptor

Guide d'utilisation pour l'investigation numérique et la réponse aux incidents

Version 1.0 - Mai 2025



Ce manuel complet couvre tous les aspects de l'outil forensic Velociraptor, de l'installation à l'utilisation avancée, pour tous les niveaux d'utilisateurs et tous les systèmes d'exploitation.

Table des matières

2. Présentation de Velociraptor
3. Historique et développement
4. Cas d'usage principaux
5. Avantages par rapport aux autres outils
6. **Concepts fondamentaux**
7. Architecture de Velociraptor
8. Modèle client-serveur
9. Flux de données
10. Terminologie essentielle
11. Sécurité et chiffrement
12. **Installation et déploiement**
13. Prérequis système
14. Installation du serveur
15. Déploiement des clients
16. Configuration initiale
17. Topologies de déploiement
18. **Interface utilisateur**
19. Présentation de l'interface web
20. Tableaux de bord
21. Gestion des clients
22. Gestion des utilisateurs et permissions
23. **Velociraptor Query Language (VQL)**
24. Fondamentaux du VQL
25. Syntaxe et structure
26. Types de données
27. Fonctions et plugins
28. Requêtes événementielles
29. Bonnes pratiques et optimisation
30. **Artefacts**
31. Concept d'artefact
32. Création d'artefacts personnalisés

33. Utilisation des artefacts prédéfinis
34. Partage et importation d'artefacts

35. Échange d'artefacts communautaires

36. **Collecte de données**

37. Collecte de fichiers
38. Analyse de la mémoire
39. Analyse du registre Windows
40. Collecte des journaux d'événements

41. Analyse des artefacts système

42. **Chasses (Hunts)**

43. Concept et utilité des chasses
44. Création et configuration de chasses
45. Exécution et surveillance des chasses

46. Analyse des résultats de chasses

47. **Analyse avancée**

48. Notebooks Velociraptor
49. Analyse temporelle et chronologies
50. Analyse de malware

51. Analyse réseau

52. **Cas d'usage pratiques**

    - Investigation d'une compromission par phishing
    - Chasse aux menaces : Détection de Cobalt Strike
    - Audit de conformité : Vérification des correctifs
    - Réponse à un incident Ransomware

53. **Conclusion et ressources**

    - Synthèse des capacités de Velociraptor
    - Évolutions futures et tendances
    - Ressources complémentaires
    - Bibliographie

1. Introduction

Logo Velociraptor

Présentation de Velociraptor

Velociraptor est une plateforme avancée d'investigation numérique et de réponse aux incidents (DFIR - Digital Forensics and Incident Response) conçue pour collecter et analyser des données à grande échelle. Son nom, inspiré du prédateur préhistorique, reflète sa rapidité et son efficacité dans la "chasse" aux menaces informatiques.

Velociraptor se distingue par sa flexibilité et sa puissance, permettant aux analystes de sécurité de mener des investigations approfondies sur des milliers de systèmes simultanément. Grâce à son langage de requête dédié (VQL - Velociraptor Query Language), il offre des capacités d'investigation personnalisables qui s'adaptent à une grande variété de scénarios de sécurité.

Cette plateforme open-source représente une évolution significative dans le domaine des outils DFIR, combinant la puissance des outils commerciaux avec la flexibilité et la transparence du logiciel libre.

Historique et développement

Velociraptor a été créé par Mike Cohen, également connu pour son travail sur GRR (Google Rapid Response), un autre outil d'investigation numérique développé chez Google. Fort de cette expérience, Mike a fondé Velocidex et lancé Velociraptor en 2018 comme une solution plus légère, plus flexible et plus facile à déployer que ses prédécesseurs.

Les étapes clés du développement de Velociraptor incluent :

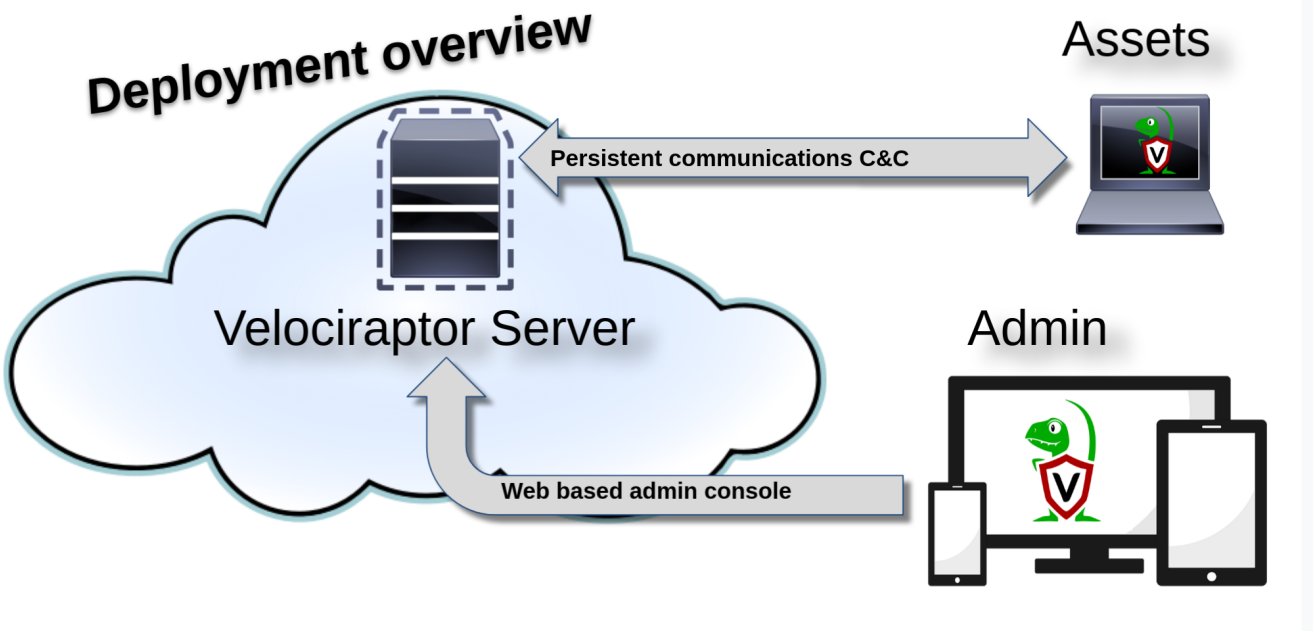
- **2018** : Première version publique de Velociraptor
- **2019** : Introduction du VQL (Velociraptor Query Language), apportant une flexibilité sans précédent
- **2020** : Ajout des notebooks pour l'analyse interactive et la documentation
- **2021** : Amélioration significative des performances et de l'interface utilisateur
- **2022** : Introduction de l'Artifact Exchange pour le partage communautaire
- **2023** : Expansion des capacités cloud et conteneurs
- **2024** : Renforcement des capacités d'analyse avancée et d'automatisation

Depuis sa création, Velociraptor a connu une adoption croissante dans la communauté de la sécurité informatique, bénéficiant de contributions régulières d'experts du monde entier. Son développement actif et sa communauté engagée en font un outil en constante évolution.

Cas d'usage principaux

Velociraptor excelle dans de nombreux scénarios de sécurité, notamment :

Réponse aux incidents



- **Investigation rapide** : Collecte immédiate de preuves volatiles sur des systèmes potentiellement compromis
- **Analyse à grande échelle** : Recherche d'indicateurs de compromission sur des milliers de systèmes
- **Triage et priorisation** : Identification rapide des systèmes les plus critiques nécessitant une attention immédiate
- **Collecte de preuves** : Préservation forensique des artefacts pertinents pour l'analyse approfondie

Chasse aux menaces (Threat Hunting)

- **Recherche proactive** : Détection de menaces avant qu'elles ne déclenchent des alertes traditionnelles
- **Hypothèses de chasse** : Vérification d'hypothèses spécifiques sur des techniques d'attaque potentielles

- **Détection de comportements anormaux** : Identification d'activités suspectes par rapport aux lignes de base
- **Validation de sécurité** : Vérification de l'efficacité des contrôles de sécurité existants

Conformité et audit

- **Vérification de configuration** : Validation de la conformité des systèmes aux standards de sécurité
- **Inventaire logiciel** : Recensement des applications installées et détection des logiciels non autorisés
- **Vérification de correctifs** : Identification des systèmes nécessitant des mises à jour de sécurité
- **Documentation de conformité** : Génération de preuves pour les audits réglementaires

Investigation numérique

- **Analyse forensique** : Collecte et analyse d'artefacts système pour les investigations approfondies
- **Chronologie d'événements** : Reconstruction de la séquence d'actions lors d'un incident
- **Analyse de malware** : Identification et caractérisation des logiciels malveillants
- **Attribution** : Collecte d'indices sur l'origine et les méthodes des attaquants

Avantages par rapport aux autres outils

Velociraptor présente plusieurs avantages distinctifs par rapport aux solutions traditionnelles :

Flexibilité inégalée

- **Langage VQL** : Permet de créer des requêtes personnalisées sans nécessiter de développement logiciel
- **Artefacts personnalisables** : Possibilité de créer et partager des modules d'investigation spécifiques
- **Adaptation rapide** : Capacité à répondre à de nouvelles menaces sans attendre des mises à jour

Efficacité opérationnelle

- **Déploiement léger** : Agent client avec une empreinte minimale (quelques Mo)
- **Performance optimisée** : Impact limité sur les systèmes analysés
- **Scalabilité** : Capacité à gérer des déploiements de quelques dizaines à plusieurs milliers de systèmes

Transparence et contrôle

- **Open source** : Code source accessible et vérifiable
- **Pas de boîte noire** : Fonctionnement transparent et compréhensible
- **Contrôle total** : Maîtrise complète de l'infrastructure et des données collectées

Communauté active

- **Partage d'expertise** : Échange d'artefacts et de techniques d'investigation
- **Développement continu** : Améliorations régulières et réponse rapide aux problèmes
- **Documentation riche** : Ressources abondantes pour l'apprentissage et le perfectionnement

Coût total de possession réduit

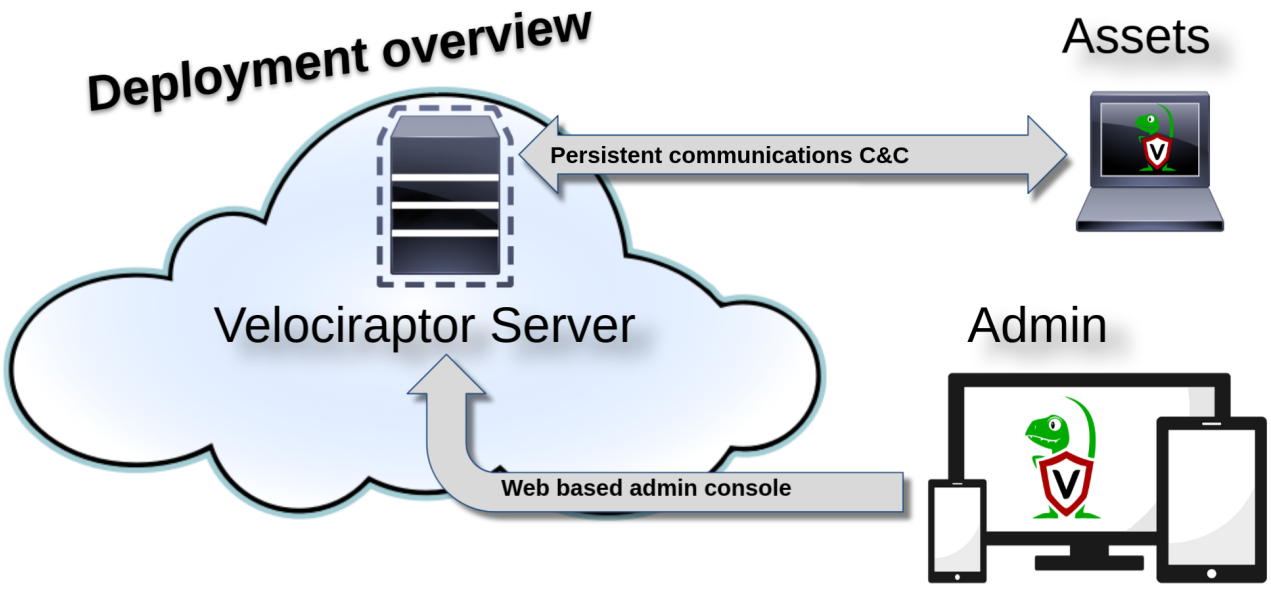
- **Licence libre** : Pas de coûts de licence par endpoint
- **Ressources modestes** : Fonctionne efficacement sur du matériel standard
- **Intégration facile** : S'adapte aux infrastructures et workflows existants

Ces avantages font de Velociraptor un choix privilégié pour les équipes de sécurité cherchant un équilibre entre puissance, flexibilité et maîtrise des coûts dans leurs opérations d'investigation numérique et de réponse aux incidents.

2. Concepts fondamentaux

Architecture de Velociraptor

L'architecture de Velociraptor est conçue pour être à la fois robuste et flexible, permettant des déploiements adaptés à diverses tailles d'organisations et besoins opérationnels.



Composants principaux

L'architecture de Velociraptor comprend plusieurs composants clés qui travaillent ensemble :

1. **Serveur Velociraptor** : Le cœur du système qui coordonne toutes les opérations, stocke les données collectées et fournit l'interface utilisateur.
2. **Clients Velociraptor** : Agents légers installés sur les systèmes à surveiller, responsables de l'exécution des requêtes et de la collecte des données.
3. **Base de données** : Stockage des données collectées, des configurations et des résultats d'analyse. Velociraptor utilise par défaut une base de données intégrée (datastore) mais peut être configuré avec des bases externes.
4. **Interface Web** : Interface graphique permettant aux analystes d'interagir avec le système, de lancer des requêtes et de visualiser les résultats.
5. **API** : Interface programmatique permettant l'intégration avec d'autres outils et systèmes.

Flux de communication

Les communications entre les composants suivent un modèle sécurisé :

1. **Communication client-serveur** : Chiffrée de bout en bout avec TLS mutuel (mTLS), assurant l'authenticité et la confidentialité des échanges.

2. **Polling** : Les clients interrogent périodiquement le serveur pour recevoir de nouvelles tâches, ce qui permet de traverser les pare-feu et NAT sans configuration spéciale.

3. **Compression** : Les données sont compressées avant transmission pour optimiser l'utilisation de la bande passante.

4. **Mise en file d'attente** : Les tâches et résultats sont mis en file d'attente pour assurer la fiabilité même en cas de connectivité intermittente.

Modèle client-serveur

Velociraptor utilise un modèle client-serveur classique mais avec plusieurs caractéristiques spécifiques adaptées aux besoins de l'investigation numérique.

Serveur Velociraptor

Le serveur Velociraptor remplit plusieurs fonctions essentielles :

1. **Coordination** : Gestion des clients, distribution des tâches et collecte des résultats.

2. **Stockage** : Conservation sécurisée des données collectées et des configurations.

3. **Traitement** : Analyse des données, génération de rapports et corrélation d'informations.

4. **Interface utilisateur** : Fourniture d'une interface web pour les analystes.

5. **Authentification et autorisation** : Gestion des utilisateurs et de leurs permissions.

Le serveur peut être déployé de différentes manières :

- **Serveur unique** : Configuration simple pour les petits déploiements.
- **Architecture distribuée** : Séparation des fonctions (frontend, backend, stockage) pour les grands déploiements.
- **Haute disponibilité** : Configuration redondante pour les environnements critiques.

Clients Velociraptor

Les clients Velociraptor sont des agents légers avec des caractéristiques importantes :

1. **Empreinte minimale** : Conçus pour avoir un impact limité sur les performances du système hôte.

2. **Autonomie** : Capables de fonctionner même sans connexion permanente au serveur.
3. **Multiplateforme** : Disponibles pour Windows, Linux, macOS et d'autres systèmes.
4. **Extensibilité** : Peuvent exécuter des requêtes VQL personnalisées envoyées par le serveur.
5. **Isolation** : Opèrent avec des privilèges contrôlés pour limiter les risques de sécurité.

Les clients maintiennent une connexion avec le serveur via un mécanisme de polling, ce qui présente plusieurs avantages :

- Pas besoin d'ouvrir des ports entrants sur les clients
- Fonctionnement à travers les NAT et pare-feu
- Résilience aux interruptions réseau temporaires

Flux de données

Le flux de données dans Velociraptor suit un cycle bien défini qui garantit l'efficacité et la sécurité des opérations.

Collecte de données

1. **Initiation** : Un analyste crée une requête ou sélectionne un artefact via l'interface web.
2. **Distribution** : Le serveur envoie la requête aux clients ciblés lors de leur prochaine connexion.
3. **Exécution** : Les clients exécutent la requête localement, en respectant les limites de ressources définies.
4. **Collecte** : Les données sont collectées selon les paramètres spécifiés (fichiers complets, métadonnées, hachages, etc.).
5. **Transmission** : Les résultats sont compressés, chiffrés et envoyés au serveur.

Traitement et stockage

1. **Réception** : Le serveur reçoit et déchiffre les données envoyées par les clients.
2. **Validation** : Les données sont validées pour assurer leur intégrité.

3. **Indexation** : Les informations sont indexées pour faciliter les recherches ultérieures.
4. **Stockage** : Les données sont enregistrées dans le datastore de Velociraptor.
5. **Post-traitement** : Des analyses supplémentaires peuvent être appliquées aux données collectées.

Analyse et visualisation

1. **Requêtes** : Les analystes peuvent interroger les données collectées via VQL.
2. **Filtrage** : Les résultats peuvent être filtrés et triés selon divers critères.
3. **Corrélation** : Les données de différentes sources peuvent être combinées pour une analyse plus complète.
4. **Visualisation** : Les résultats peuvent être présentés sous forme de tableaux, graphiques ou chronologies.
5. **Export** : Les données peuvent être exportées dans différents formats pour une analyse externe.

Terminologie essentielle

Pour comprendre et utiliser efficacement Velociraptor, il est important de se familiariser avec sa terminologie spécifique :

Termes généraux

- **VQL (Velociraptor Query Language)** : Langage de requête spécifique à Velociraptor, inspiré de SQL mais adapté aux besoins de l'investigation numérique.
- **Artefact** : Ensemble de requêtes VQL packagées avec documentation et paramètres, conçu pour collecter un type spécifique d'information forensique.
- **Chasse (Hunt)** : Opération consistant à exécuter un ou plusieurs artefacts sur un ensemble de clients sélectionnés.
- **Collection** : Ensemble de données collectées suite à l'exécution d'un artefact sur un client spécifique.
- **Notebook** : Document interactif combinant texte, requêtes VQL et résultats, utilisé pour l'analyse et la documentation.

Composants système

- **Frontend** : Composant du serveur qui gère les connexions des clients et l'interface utilisateur.
- **Datastore** : Système de stockage utilisé par Velociraptor pour conserver les données collectées et les configurations.
- **Client Monitoring** : Système de surveillance continue sur les clients, basé sur des requêtes événementielles.
- **Artifact Exchange** : Plateforme de partage d'artefacts entre utilisateurs de Velociraptor.

Concepts VQL

- **Plugin** : Source de données dans VQL qui génère des lignes de résultats (ex: `pslist()`, `glob()`).
- **Fonction** : Traitement qui prend des arguments et retourne une valeur unique (ex: `hash()`, `timestamp()`).
- **LET** : Déclaration qui assigne une valeur ou un résultat de requête à une variable.
- **Requête événementielle** : Requête VQL qui s'exécute en continu et génère des résultats lorsque des événements spécifiques se produisent.

Sécurité et chiffrement

La sécurité est un aspect fondamental de l'architecture de Velociraptor, particulièrement important étant donné la nature sensible des données manipulées.

Modèle de sécurité

Velociraptor implémente un modèle de sécurité robuste basé sur plusieurs principes :

1. **Authentification mutuelle** : Les clients et le serveur s'authentifient mutuellement via des certificats X.509.
2. **Chiffrement de bout en bout** : Toutes les communications sont chiffrées avec TLS moderne.
3. **Contrôle d'accès basé sur les rôles (RBAC)** : Les utilisateurs se voient attribuer des rôles spécifiques qui déterminent leurs permissions.

4. **Principe du moindre privilège** : Les clients opèrent avec les privilèges minimums nécessaires pour accomplir leurs tâches.

5. **Audit complet** : Toutes les actions des utilisateurs sont enregistrées pour traçabilité.

Infrastructure à clé publique (PKI)

Velociraptor utilise sa propre PKI pour gérer les certificats :

1. **Autorité de certification (CA)** : Générée lors de la configuration initiale du serveur.

2. **Certificats serveur** : Émis par la CA pour authentifier le serveur auprès des clients.

3. **Certificats client** : Uniques pour chaque client, émis lors de leur déploiement.

4. **Rotation des certificats** : Possibilité de renouveler les certificats selon les politiques de sécurité.

Protection des données

Les données collectées bénéficient de plusieurs niveaux de protection :

1. **Chiffrement en transit** : Via TLS pour toutes les communications réseau.

2. **Chiffrement au repos** : Option de chiffrement du datastore pour protéger les données stockées.

3. **Contrôle de l'accès** : Restrictions sur qui peut accéder aux données collectées.

4. **Minimisation des données** : Collecte ciblée pour limiter l'exposition d'informations sensibles.

5. **Rétention configurable** : Politiques de conservation des données adaptables aux besoins réglementaires.

Considérations de déploiement sécurisé

Pour maximiser la sécurité d'un déploiement Velociraptor :

1. **Segmentation réseau** : Isolation du serveur Velociraptor dans un segment réseau sécurisé.

2. **Durcissement du système** : Application des meilleures pratiques de sécurisation pour le serveur et les clients.

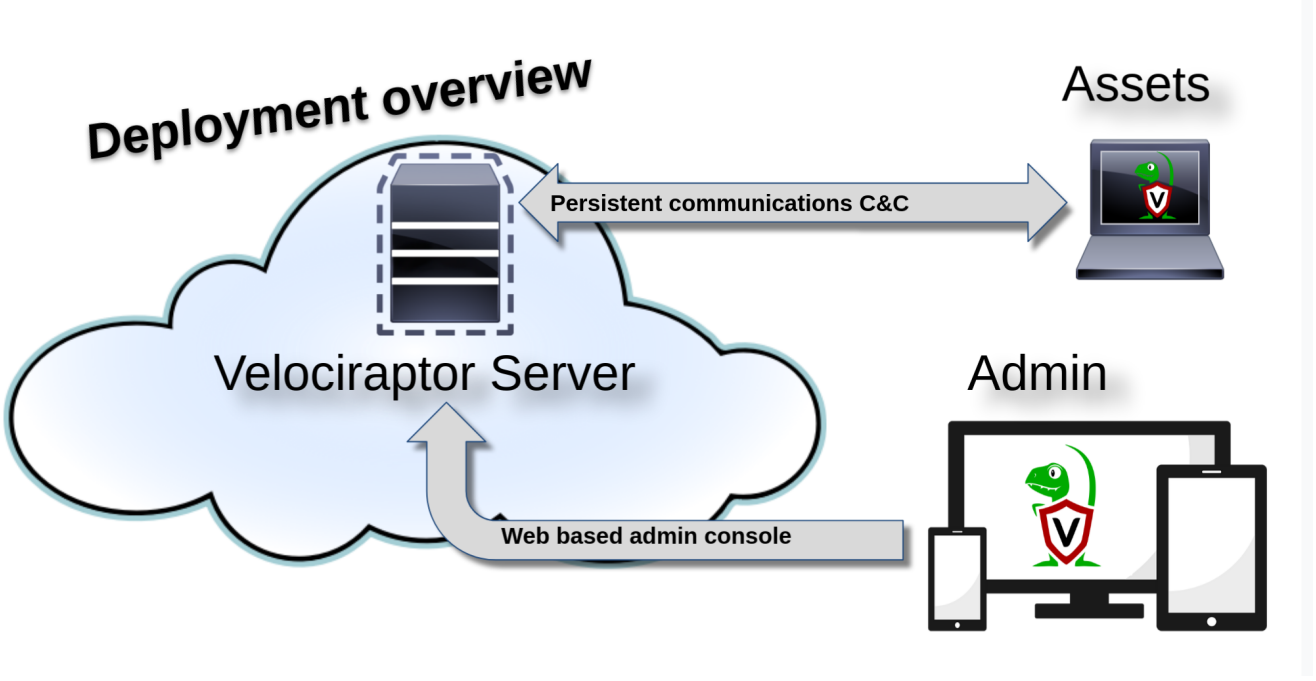
3. **Authentification multifacteur** : Utilisation de l'authentification à deux facteurs pour l'accès à l'interface web.
4. **Surveillance et alertes** : Mise en place de mécanismes de détection d'activités suspectes.
5. **Mises à jour régulières** : Application des correctifs de sécurité dès leur disponibilité.

Ces concepts fondamentaux constituent la base nécessaire pour comprendre le fonctionnement de Velociraptor et exploiter efficacement ses capacités dans divers scénarios d'investigation numérique et de réponse aux incidents.

4. Interface utilisateur

Présentation de l'interface web

L'interface web de Velociraptor est le point central de contrôle pour toutes les opérations de la plateforme. Elle offre une expérience utilisateur intuitive et puissante, permettant aux analystes de sécurité de naviguer efficacement à travers les différentes fonctionnalités du système.



Accès à l'interface web

Pour accéder à l'interface web de Velociraptor :

1. Ouvrez un navigateur web moderne (Chrome, Firefox, Edge, Safari)
2. Accédez à l'URL configurée pour votre serveur Velociraptor (par défaut : https://localhost:8889)
3. Acceptez l'avertissement de certificat si vous utilisez des certificats auto-signés
4. Connectez-vous avec vos identifiants

Structure générale de l'interface

L'interface web de Velociraptor est organisée en plusieurs zones principales :

1. **Barre de navigation latérale** : Située sur le côté gauche de l'écran, elle permet d'accéder aux différentes sections de l'application.

2. **Barre supérieure** : Contient le sélecteur d'organisation, les informations utilisateur, et les options de configuration globales.

3. **Zone de contenu principale** : Occupe la majeure partie de l'écran et affiche le contenu de la section sélectionnée.

4. **Barre d'état** : Située en bas de l'écran, elle affiche des informations sur l'état du serveur et les tâches en cours.

Thème et personnalisation

Velociraptor propose un thème sombre par défaut, optimisé pour réduire la fatigue oculaire lors des longues sessions d'analyse. L'interface peut être personnalisée via les paramètres utilisateur, accessibles depuis le menu déroulant de l'utilisateur dans la barre supérieure.

Compatibilité et optimisation

L'interface web de Velociraptor est conçue pour fonctionner de manière optimale sur les navigateurs modernes et s'adapte à différentes tailles d'écran. Pour une expérience optimale, il est recommandé d'utiliser :

- Un écran avec une résolution minimale de 1920x1080
- Un navigateur web récent avec JavaScript activé
- Une connexion réseau stable vers le serveur Velociraptor

Navigation et fonctionnalités principales

La navigation dans l'interface de Velociraptor est organisée de manière logique pour faciliter l'accès aux différentes fonctionnalités. Voici un aperçu des sections principales et de leur utilisation.

Tableau de bord principal

Le tableau de bord principal est la première page que vous voyez après la connexion. Il offre une vue d'ensemble de votre déploiement Velociraptor, incluant :

- Statistiques sur le nombre de clients connectés
- Graphiques d'activité récente
- Accès rapide aux fonctions fréquemment utilisées
- Notifications et alertes importantes

Section Clients

La section Clients permet de gérer et d'interagir avec les terminaux sur lesquels l'agent Velociraptor est installé :

- **Vue Liste** : Affiche tous les clients avec des informations de base comme le nom d'hôte, l'OS, et le statut de connexion
- **Vue Détaillée** : En cliquant sur un client, vous accédez à une vue détaillée avec :
- Informations système complètes
- Historique des collections
- Possibilité de lancer de nouvelles collections
- Shell VQL pour des requêtes interactives
- Téléchargement de fichiers

Section Chasses (Hunts)

La section Chasses permet de créer et gérer des opérations de collecte à grande échelle :

- **Création de chasse** : Assistant pour configurer une nouvelle chasse
- **Liste des chasses** : Vue d'ensemble de toutes les chasses passées et en cours
- **Détails de chasse** : Statistiques, résultats et gestion d'une chasse spécifique

Section Artefacts

La section Artefacts est dédiée à la gestion des artefacts Velociraptor :

- **Explorateur d'artefacts** : Parcourir les artefacts disponibles par catégorie

- **Création d'artefacts** : Interface pour créer ou modifier des artefacts personnalisés
- **Importation/Exportation** : Outils pour partager des artefacts

Section Notebooks

Les notebooks sont des espaces de travail interactifs pour l'analyse et la documentation :

- **Liste des notebooks** : Vue d'ensemble de tous les notebooks
- **Éditeur de notebook** : Interface pour créer et modifier des notebooks
- **Exécution de VQL** : Cellules interactives pour exécuter du code VQL
- **Visualisation** : Outils pour créer des graphiques et des tableaux

Section Événements

La section Événements permet de surveiller les activités en temps réel :

- **Flux d'événements** : Affichage des événements détectés par les moniteurs
- **Configuration des moniteurs** : Interface pour configurer la surveillance continue
- **Alertes** : Gestion des alertes générées par les événements

Section Serveur

La section Serveur donne accès aux fonctionnalités d'administration :

- **Artefacts serveur** : Exécution d'artefacts sur le serveur lui-même
- **Utilisateurs** : Gestion des comptes utilisateurs et des permissions
- **Configuration** : Paramètres du serveur et options de déploiement
- **Organisations** : Gestion des organisations dans un environnement multi-tenant

Tableaux de bord

Les tableaux de bord dans Velociraptor fournissent des vues synthétiques et des visualisations qui aident à comprendre rapidement l'état du système et les résultats des analyses.

Tableau de bord principal

Le tableau de bord principal offre une vue d'ensemble du déploiement Velociraptor :

1. **Statistiques des clients** :
2. Nombre total de clients
3. Répartition par système d'exploitation

4. Clients en ligne vs hors ligne
5. Dernières connexions
6. **Activité récente** :
7. Chasses récemment lancées
8. Collections récentes
9. Événements importants
10. **État du serveur** :
11. Utilisation des ressources
12. Espace disque disponible
13. Santé du système
14. **Accès rapide** :
15. Liens vers les fonctions fréquemment utilisées
16. Artefacts récemment utilisés
17. Clients récemment consultés

Tableaux de bord des chasses

Chaque chasse dispose de son propre tableau de bord qui présente :

1. **Statistiques d'exécution** :
2. Nombre de clients ciblés
3. Progression de l'exécution
4. Taux de réussite/échec
5. **Visualisations des résultats** :
6. Graphiques récapitulatifs
7. Distributions statistiques
8. Chronologies d'événements
9. **Résultats détaillés** :
10. Tableaux de données filtrables
11. Exportation des résultats
12. Accès aux données brutes

Tableaux de bord des clients

La vue détaillée d'un client inclut plusieurs tableaux de bord spécialisés :

1. **Informations système** :
2. Configuration matérielle
3. Système d'exploitation
4. Utilisateurs et groupes
5. Réseau
6. **Activité** :
7. Processus en cours
8. Connexions réseau
9. Fichiers récemment modifiés
10. **Collections** :
11. Historique des collections
12. Résultats des collections récentes
13. État des collections en cours

Tableaux de bord personnalisés

Velociraptor permet de créer des tableaux de bord personnalisés via les notebooks :

1. **Création** :
2. Utilisation de cellules VQL pour générer des données
3. Ajout de visualisations (graphiques, tableaux)
4. Intégration de texte explicatif
5. **Partage** :
6. Exportation de notebooks
7. Collaboration entre utilisateurs
8. Modèles réutilisables
9. **Automatisation** :
10. Mise à jour programmée
11. Alertes basées sur des conditions
12. Rapports périodiques

Bonnes pratiques pour les tableaux de bord

Pour tirer le meilleur parti des tableaux de bord Velociraptor :

1. **Personnalisation** :
2. Adaptez les tableaux de bord à vos besoins spécifiques
3. Créez des vues dédiées pour différents cas d'usage
4. **Organisation** :
5. Regroupez les informations connexes
6. Utilisez une hiérarchie logique
7. Nommez clairement les tableaux de bord
8. **Optimisation** :
9. Limitez le nombre d'éléments par tableau de bord
10. Privilégiez les visualisations pertinentes
11. Évitez les requêtes trop lourdes pour les mises à jour en temps réel

Gestion des clients

La gestion efficace des clients est essentielle pour tirer pleinement parti de Velociraptor. Cette section détaille les fonctionnalités disponibles pour gérer les terminaux sur lesquels l'agent Velociraptor est installé.

Vue d'ensemble des clients

La section Clients de l'interface web offre une vue complète de tous les clients connectés à votre déploiement Velociraptor :

1. **Liste des clients** :
2. Affichage tabulaire avec filtres et tri
3. Informations de base : nom d'hôte, IP, OS, version
4. Indicateurs d'état : en ligne/hors ligne, dernière connexion
5. Étiquettes personnalisées
6. **Recherche et filtrage** :
7. Recherche par nom d'hôte, IP, étiquette
8. Filtres avancés par OS, version, état

9. Enregistrement des filtres fréquemment utilisés

10. **Actions en masse** :

11. Sélection multiple de clients
12. Application d'étiquettes
13. Lancement de collections sur plusieurs clients
14. Ajout à une chasse existante

Interaction avec un client spécifique

En sélectionnant un client dans la liste, vous accédez à une vue détaillée offrant de nombreuses fonctionnalités :

1. **Informations détaillées** :
2. Configuration matérielle et logicielle
3. Utilisateurs et groupes
4. Configuration réseau

5. Historique des connexions

6. **Collections** :

7. Lancement de collections d'artefacts
8. Visualisation des résultats

9. Historique des collections précédentes

10. **Shell VQL** :

11. Interface interactive pour exécuter des requêtes VQL
12. Historique des commandes

13. Modèles de requêtes prédéfinis

14. **Gestion des fichiers** :

15. Navigation dans le système de fichiers
16. Téléchargement de fichiers
17. Téléversement de fichiers

18. Analyse de fichiers

19. **Surveillance** :

20. Configuration de moniteurs d'événements

21. Visualisation des événements en temps réel
22. Alertes basées sur des conditions

Étiquetage et organisation

L'étiquetage des clients permet de les organiser de manière logique et de faciliter les opérations à grande échelle :

1. **Création d'étiquettes** :
2. Étiquettes personnalisées (ex: "Serveurs", "Postes de travail", "Direction")
3. Étiquettes automatiques basées sur des règles
4. Étiquettes temporaires pour des opérations spécifiques
5. **Utilisation des étiquettes** :
6. Filtrage rapide des clients
7. Ciblage précis pour les chasses
8. Organisation logique du parc
9. **Gestion des étiquettes** :
10. Ajout/suppression d'étiquettes
11. Renommage d'étiquettes
12. Fusion d'étiquettes

Collecte d'artefacts sur les clients

La collecte d'artefacts est l'opération principale effectuée sur les clients :

1. **Sélection d'artefacts** :
2. Parcours du catalogue d'artefacts
3. Recherche par nom ou description
4. Filtrage par catégorie ou plateforme
5. **Configuration des paramètres** :
6. Ajustement des paramètres spécifiques à l'artefact
7. Configuration des limites de ressources
8. Planification de l'exécution
9. **Exécution et suivi** :

10. Lancement de la collecte
11. Suivi de la progression en temps réel

12. Visualisation des résultats intermédiaires

13. **Analyse des résultats** :

14. Visualisation tabulaire ou graphique
15. Filtrage et tri des résultats
16. Exportation pour analyse externe

Gestion du cycle de vie des clients

Velociraptor offre des fonctionnalités pour gérer l'ensemble du cycle de vie des clients :

1. **Déploiement** :
2. Génération de packages d'installation
3. Suivi du déploiement

4. Vérification de la première connexion

5. **Maintenance** :

6. Mise à jour des clients
7. Reconfiguration à distance

8. Diagnostic des problèmes

9. **Suppression** :

10. Désinstallation propre
11. Nettoyage des données
12. Archivage des informations historiques

Bonnes pratiques pour la gestion des clients

Pour une gestion efficace des clients Velociraptor :

1. **Organisation** :
2. Utilisez un système d'étiquetage cohérent
3. Documentez les collections importantes

4. Maintenez une nomenclature claire

5. **Performance** :

6. Évitez les collections trop lourdes sur de nombreux clients simultanément
7. Planifiez les opérations intensives en dehors des heures de pointe

8. Utilisez les limites de ressources appropriées

9. **Sécurité** :

10. Limitez l'accès à la gestion des clients selon le principe du moindre privilège
11. Auditez régulièrement les opérations effectuées sur les clients
12. Documentez les raisons des collections sensibles

Gestion des utilisateurs et permissions

La gestion des utilisateurs et des permissions est un aspect crucial de Velociraptor, particulièrement dans les environnements multi-utilisateurs ou les déploiements à grande échelle. Cette section détaille les fonctionnalités disponibles pour gérer les accès et les privilèges.

Modèle de sécurité

Velociraptor utilise un modèle de sécurité basé sur les rôles (RBAC - Role-Based Access Control) qui permet de définir précisément les actions que chaque utilisateur peut effectuer :

1. **Utilisateurs** : Entités individuelles qui se connectent à Velociraptor
2. **Rôles** : Ensembles de permissions qui définissent les actions autorisées
3. **Organisations** : Conteneurs logiques qui isolent les clients, les artefacts et les données

Types d'authentification

Velociraptor prend en charge plusieurs méthodes d'authentification :

1. **Authentification basique** :
2. Nom d'utilisateur et mot de passe stockés localement
3. Simple à configurer, idéal pour les petits déploiements

4. Moins sécurisé que les méthodes d'authentification externe

5. **Authentification SSO (Single Sign-On)** :

6. Intégration avec des fournisseurs d'identité externes via OIDC ou SAML
7. Support pour Azure AD, Google, Okta, etc.

8. Authentification multifacteur via le fournisseur d'identité

9. **Authentification par certificat** :

10. Utilisation de certificats clients pour l'authentification
11. Haute sécurité pour les environnements sensibles
12. Configuration plus complexe

Rôles prédéfinis

Velociraptor inclut plusieurs rôles prédéfinis avec des niveaux d'accès croissants :

1. **Reader** (Lecteur) :
2. Peut voir les clients et les résultats
3. Ne peut pas lancer de nouvelles collections ou chasses

4. Accès en lecture seule aux notebooks et artefacts

5. **Analyst** (Analyste) :

6. Peut lancer des collections sur des clients individuels
7. Peut créer et modifier des notebooks

8. Ne peut pas lancer de chasses ou modifier des artefacts

9. **Investigator** (Investigateur) :

10. Peut lancer des chasses
11. Peut créer et modifier des artefacts

12. Ne peut pas gérer les utilisateurs ou la configuration du serveur

13. **Administrator** (Administrateur) :

14. Accès complet à toutes les fonctionnalités
15. Peut gérer les utilisateurs et les permissions

16. Peut modifier la configuration du serveur

17. **Org Administrator** (Administrateur d'organisation) :

18. Peut gérer les organisations (uniquement dans l'organisation racine)
19. Peut créer de nouvelles organisations
20. Peut attribuer des administrateurs aux organisations

Gestion des utilisateurs

L'interface d'administration des utilisateurs permet de :

1. **Créer des utilisateurs** :
2. Définition du nom d'utilisateur
3. Attribution de rôles
4. Configuration des paramètres spécifiques
5. **Modifier les utilisateurs** :
6. Changement de rôle
7. Réinitialisation de mot de passe
8. Activation/désactivation de compte
9. **Supprimer des utilisateurs** :
10. Révocation de tous les accès
11. Conservation de l'historique des actions

Permissions granulaires

Au-delà des rôles prédéfinis, Velociraptor permet de définir des permissions granulaires pour des cas d'usage spécifiques :

1. **Permissions par organisation** :
2. Un utilisateur peut avoir différents rôles dans différentes organisations
3. Isolation complète entre organisations
4. **Permissions par artefact** :
5. Restriction de l'accès à certains artefacts sensibles
6. Limitation des paramètres modifiables
7. **Permissions par client** :
8. Restriction de l'accès à certains groupes de clients
9. Utilisation d'étiquettes pour définir les périmètres d'accès

Audit et traçabilité

Velociraptor maintient un journal complet des actions des utilisateurs pour assurer la traçabilité :

1. **Journal d'audit** :
2. Enregistrement de toutes les actions importantes
3. Horodatage et identification de l'utilisateur
4. Conservation à long terme
5. **Visualisation des actions** :
6. Interface dédiée pour consulter le journal d'audit
7. Filtrage par utilisateur, action, date
8. Exportation pour analyse externe
9. **Alertes de sécurité** :
10. Détection des comportements suspects
11. Notification en cas d'actions sensibles
12. Intégration possible avec des SIEM externes

Bonnes pratiques de gestion des utilisateurs

Pour une gestion efficace et sécurisée des utilisateurs :

1. **Principe du moindre privilège** :
2. Attribuez uniquement les permissions nécessaires
3. Utilisez des rôles spécifiques plutôt que des administrateurs généraux
4. Réévaluez régulièrement les permissions
5. **Séparation des tâches** :
6. Distinguez les rôles d'administration et d'analyse
7. Implémentez des validations pour les actions sensibles
8. Documentez les raisons des élévations de privilèges
9. **Intégration avec les processus existants** :
10. Alignez la gestion des utilisateurs avec les processus RH
11. Automatisez la création et la suppression des comptes
12. Synchronisez avec les annuaires d'entreprise existants

Structure du Manuel Complet sur Velociraptor

Page de titre

- Titre : "Manuel Complet sur Velociraptor"
- Sous-titre : "Guide d'installation, d'utilisation et d'analyse forensique"
- Logo Velociraptor
- Date de création

Table des matières

1. Introduction

- Présentation de Velociraptor
- Historique et développement
- Cas d'usage et avantages
- Comparaison avec d'autres outils forensiques

2. Concepts fondamentaux

- Architecture de Velociraptor
- Modèle client-serveur
- Velociraptor Query Language (VQL)
- Artefacts et collecteurs
- Système de chasse (Hunting)

3. Installation et déploiement

- Prérequis système
- Installation sur Windows
- Installation sur Linux
- Installation sur macOS
- Configuration du serveur
- Déploiement des clients
- Multi-tenancy et organisations

4. Interface utilisateur

- Présentation de l'interface web
- Navigation et fonctionnalités principales
- Tableaux de bord
- Gestion des clients
- Gestion des utilisateurs et permissions

5. Velociraptor Query Language (VQL)

- Fondamentaux du VQL
- Syntaxe et structure
- Types de données
- Fonctions et plugins
- Requêtes événementielles
- Bonnes pratiques et optimisation

6. Artefacts

- Concept d'artefact
- Création d'artefacts personnalisés
- Utilisation des artefacts prédéfinis
- Partage et importation d'artefacts
- Échange d'artefacts communautaires

7. Collecte de données

- Collecte de fichiers
- Analyse de la mémoire
- Analyse du registre Windows
- Collecte des journaux d'événements
- Analyse des artefacts système

8. Chasse aux menaces (Hunting)

- Concept de chasse
- Configuration d'une chasse
- Analyse des résultats
- Automatisation des chasses

- Cas pratiques de chasse

9. Analyse forensique

- Méthodologie d'analyse
- Analyse de timeline
- Détection de malware
- Investigation d'incidents
- Corrélation d'événements

10. Notebooks et rapports

- Utilisation des notebooks
- Documentation des investigations
- Génération de rapports
- Exportation des résultats
- Partage des analyses

11. Automatisation et intégration

- Automatisation avec VQL
- Intégration avec d'autres outils
- API Velociraptor
- Scripts et extensions
- Intégration avec SIEM

12. Cas pratiques

- Investigation d'une compromission
- Analyse de ransomware
- Détection de persistance
- Analyse de mouvement latéral
- Réponse à incident

13. Bonnes pratiques et optimisation

- Optimisation des performances
- Gestion des ressources
- Sécurisation de Velociraptor

- Sauvegarde et restauration
- Mise à jour et maintenance

14. Dépannage

- Problèmes courants
- Diagnostic et résolution
- Logs et débogage
- Support communautaire
- Ressources additionnelles

15. Conclusion

- Récapitulatif
- Évolutions futures
- Ressources complémentaires

Annexes

- Référence des commandes VQL
- Liste des artefacts prédéfinis
- Glossaire
- Index
- Bibliographie et références

1. Introduction

Présentation de Velociraptor

Velociraptor est un outil avancé de forensique numérique et de réponse aux incidents (DFIR - Digital Forensic and Incident Response) conçu pour améliorer la visibilité sur les terminaux d'un réseau. Développé comme une solution open-source, Velociraptor offre aux analystes de sécurité et aux équipes de réponse aux incidents une plateforme puissante et flexible pour collecter, analyser et surveiller des données sur des flottes entières de systèmes.

Le nom "Velociraptor" évoque la rapidité et l'agilité, qualités essentielles dans le domaine de la cybersécurité où la vitesse de détection et de réponse est cruciale. Son

logo, représentant un raptor vert tenant un bouclier rouge en forme de "V", symbolise la protection et la vigilance que l'outil apporte aux environnements informatiques.

Velociraptor se distingue par sa capacité à effectuer des collectes ciblées d'artefacts forensiques, à chasser les menaces de manière proactive, et à surveiller en continu les activités suspectes sur les terminaux. Sa flexibilité provient principalement du Velociraptor Query Language (VQL), un langage de requête puissant qui permet aux utilisateurs de créer des requêtes personnalisées pour répondre à des besoins spécifiques d'investigation.

Historique et développement

Velociraptor a été initialement développé par des professionnels de la forensique numérique et de la réponse aux incidents qui avaient besoin d'un outil plus efficace et flexible pour la chasse aux menaces et l'analyse des terminaux. Le projet a été créé par Mike Cohen, un expert reconnu dans le domaine de la forensique numérique, qui avait précédemment travaillé sur des outils comme GRR (Google Rapid Response).

Lancé comme un projet open-source, Velociraptor a rapidement gagné en popularité au sein de la communauté de la sécurité informatique. En 2021, Rapid7, une entreprise leader dans le domaine de la cybersécurité, a acquis Velociraptor tout en maintenant son statut open-source, garantissant ainsi la continuité de son développement et de sa disponibilité pour la communauté.

Aujourd'hui, Velociraptor bénéficie à la fois du soutien d'une entreprise établie et d'une communauté active de contributeurs qui enrichissent constamment ses fonctionnalités et partagent des artefacts personnalisés pour répondre à de nouveaux cas d'usage.

Cas d'usage et avantages

Velociraptor offre une multitude de cas d'usage pour les professionnels de la cybersécurité :

Analyse forensique numérique

- Reconstruction des activités d'un attaquant à travers l'analyse des artefacts système
- Collecte et analyse de preuves numériques pour les investigations
- Extraction et analyse de fichiers spécifiques, journaux d'événements, et autres artefacts forensiques

Chasse aux menaces

- Recherche proactive d'indicateurs de compromission (IOC) sur l'ensemble du parc informatique
- Détection d'adversaires sophistiqués qui auraient échappé aux mécanismes de détection traditionnels
- Validation rapide des hypothèses de sécurité à grande échelle

Réponse aux incidents

- Investigation rapide des incidents de sécurité sur de multiples systèmes simultanément
- Analyse des épidémies de malware et autres activités suspectes
- Collecte de preuves pour déterminer l'étendue d'une compromission

Surveillance continue

- Monitoring des activités suspectes des utilisateurs (comme les fichiers copiés vers des périphériques USB)
- Détection des comportements anormaux pouvant indiquer une menace
- Alertes en temps réel sur des événements critiques de sécurité

Conformité et audit

- Vérification de l'état de sécurité des systèmes
- Collecte de données pour les audits de conformité
- Documentation des contrôles de sécurité en place

Les principaux avantages de Velociraptor incluent :

1. **Flexibilité** : Grâce au VQL, les utilisateurs peuvent créer des requêtes personnalisées pour répondre à pratiquement n'importe quel besoin d'investigation.

2. **Évolutivité** : Velociraptor peut gérer des déploiements allant de quelques dizaines à des dizaines de milliers de terminaux.

3. **Rapidité** : Les requêtes sont exécutées directement sur les terminaux, ce qui permet d'obtenir des résultats en temps quasi réel.

4. **Légèreté** : L'agent Velociraptor a une empreinte système minimale, ce qui limite son impact sur les performances des terminaux.

5. **Open-source** : Étant open-source, Velociraptor peut être audité, modifié et adapté aux besoins spécifiques des organisations.

6. **Communauté active** : Une communauté dynamique partage régulièrement des artefacts, des cas d'usage et des bonnes pratiques.

Comparaison avec d'autres outils forensiques

Velociraptor se positionne dans un écosystème d'outils de forensique numérique et de réponse aux incidents, chacun avec ses forces et ses cas d'usage spécifiques. Voici comment Velociraptor se compare à d'autres solutions populaires :

Velociraptor vs. GRR (Google Rapid Response)

- GRR a été une source d'inspiration pour Velociraptor, mais ce dernier a été conçu pour être plus léger et plus flexible.
- Velociraptor utilise VQL, un langage de requête plus accessible et puissant que le système de flux de GRR.
- Velociraptor a généralement une empreinte système plus légère et des performances supérieures pour les déploiements à grande échelle.

Velociraptor vs. OSQuery

- OSQuery permet d'interroger les systèmes d'exploitation comme des bases de données SQL, tandis que Velociraptor utilise VQL, spécifiquement conçu pour les tâches forensiques.
- Velociraptor intègre des capacités de collecte et d'analyse forensique plus avancées qu'OSQuery.
- Velociraptor offre une interface utilisateur web complète, alors qu'OSQuery est principalement un outil en ligne de commande.

Velociraptor vs. EDR commerciaux (CrowdStrike, Carbon Black, etc.)

- Les solutions EDR commerciales offrent souvent des fonctionnalités de détection et de réponse plus automatisées, tandis que Velociraptor se concentre sur la flexibilité et la personnalisation.
- Velociraptor, étant open-source, peut être déployé sans coûts de licence, contrairement aux solutions commerciales.
- Les solutions EDR commerciales peuvent offrir des intégrations plus étendues avec d'autres produits de sécurité, mais Velociraptor compense par sa flexibilité et sa capacité à s'adapter à des cas d'usage spécifiques.

Velociraptor vs. Outils forensiques traditionnels (EnCase, FTK, etc.)

- Les outils forensiques traditionnels sont souvent conçus pour l'analyse approfondie d'un seul système à la fois, tandis que Velociraptor excelle dans l'analyse à grande échelle.
- Velociraptor permet l'analyse en direct des systèmes en fonctionnement, alors que les outils traditionnels travaillent généralement sur des images de disque.
- Velociraptor est orienté vers la rapidité de collecte et d'analyse, tandis que les outils traditionnels visent l'exhaustivité et l'admissibilité juridique des preuves.

En résumé, Velociraptor se distingue par sa philosophie unique qui consiste à pousser les requêtes vers les terminaux plutôt que de collecter toutes les données dans un emplacement central. Cette approche permet une analyse plus efficace et ciblée, particulièrement adaptée aux environnements comportant un grand nombre de systèmes.

La flexibilité de Velociraptor, combinée à sa légèreté et à sa communauté active, en fait un outil de choix pour les équipes de sécurité cherchant à améliorer leurs capacités de détection, d'investigation et de réponse aux incidents.

2. Concepts fondamentaux

Architecture de Velociraptor

Velociraptor est conçu selon une architecture client-serveur distribuée, optimisée pour la collecte et l'analyse de données forensiques à grande échelle. Cette architecture permet d'équilibrer efficacement la charge de travail entre les terminaux clients et le serveur central, tout en minimisant le trafic réseau et en maximisant la flexibilité opérationnelle.

Composants principaux

L'architecture de Velociraptor comprend plusieurs composants clés qui travaillent ensemble pour fournir ses fonctionnalités avancées :

1. **Client Velociraptor** : Un agent léger installé sur les terminaux à surveiller. Le client exécute les requêtes VQL envoyées par le serveur, collecte les données demandées, et les transmet au serveur. Conçu pour avoir un impact minimal sur les performances du système, il peut fonctionner sur divers systèmes d'exploitation (Windows, Linux, macOS).

2. **Serveur Velociraptor** : Le cœur du système qui coordonne toutes les activités. Le serveur Velociraptor comprend plusieurs sous-composants :

3. **Frontend** : Gère les communications avec les clients
4. **GUI** : Fournit l'interface web d'administration

5. **API** : Permet l'intégration avec d'autres systèmes via gRPC

6. **Stockage de données** : Velociraptor utilise un système de stockage basé sur des fichiers plutôt qu'une base de données traditionnelle. Les données collectées sont organisées dans une structure de répertoires sur le système de fichiers du serveur, ce qui simplifie la sauvegarde et la gestion du cycle de vie des données.

7. **Configuration cryptographique** : Chaque déploiement Velociraptor possède une configuration unique qui inclut des clés cryptographiques. Ces clés garantissent que les communications entre clients et serveur sont sécurisées et que les clients ne peuvent communiquer qu'avec leur serveur désigné.

Flux de communication

La communication entre les composants de Velociraptor suit un modèle bien défini :

1. Les clients Velociraptor maintiennent une connexion persistante avec le serveur, ce qui permet une communication en temps quasi réel.

2. Le serveur envoie des requêtes VQL aux clients, soit individuellement, soit dans le cadre d'une "chasse" (hunt) ciblant plusieurs clients.

3. Les clients exécutent les requêtes VQL localement, traitant et filtrant les données directement sur le terminal.

4. Seuls les résultats pertinents sont transmis au serveur, réduisant ainsi considérablement le volume de données transitant sur le réseau.

5. Le serveur agrège les résultats, les stocke dans son système de fichiers, et les rend disponibles via l'interface web.

Cette approche "requête vers les données" (plutôt que "données vers la requête") constitue un avantage majeur de Velociraptor par rapport aux solutions traditionnelles qui nécessitent de collecter toutes les données avant de pouvoir les analyser.

Modèle client-serveur

Le modèle client-serveur de Velociraptor est conçu pour offrir un équilibre optimal entre centralisation et distribution du traitement, tout en maintenant une sécurité et une efficacité élevées.

Le client Velociraptor

Le client Velociraptor est un agent léger mais puissant qui s'exécute sur les terminaux à surveiller. Ses caractéristiques principales incluent :

1. **Empreinte légère** : Conçu pour minimiser l'utilisation des ressources système (CPU, mémoire, disque, réseau).
2. **Exécution en tant que service** : Sur les systèmes d'exploitation pris en charge, le client s'exécute généralement comme un service système avec des privilèges élevés pour permettre l'accès aux artefacts forensiques.
3. **Moteur VQL intégré** : Le client intègre un interpréteur VQL complet, lui permettant d'exécuter des requêtes complexes localement.
4. **Traitement local** : Les données sont traitées, filtrées et analysées directement sur le terminal, ce qui réduit considérablement le volume de données à transmettre.
5. **Communication sécurisée** : Toutes les communications avec le serveur sont chiffrées et authentifiées à l'aide de certificats TLS et de clés cryptographiques uniques.
6. **Mode hors ligne** : Le client peut fonctionner en mode déconnecté pour des collectes ponctuelles sans nécessiter d'installation permanente.

Le serveur Velociraptor

Le serveur Velociraptor est le centre de contrôle qui coordonne les activités de tous les clients. Ses fonctionnalités principales comprennent :

1. **Frontend** : Gère les connexions entrantes des clients, distribue les tâches et collecte les résultats.
2. **Interface d'administration web** : Fournit une interface graphique intuitive pour gérer les clients, créer et exécuter des requêtes, analyser les résultats et générer des rapports.

3. **Système de stockage basé sur fichiers** : Stocke les données collectées dans une structure de répertoires organisée, sans nécessiter de base de données externe.

4. **Gestion des utilisateurs et des rôles** : Permet de définir des permissions granulaires pour contrôler l'accès aux différentes fonctionnalités et données.

5. **API gRPC** : Permet l'intégration avec d'autres systèmes et outils.

6. **Multi-tenancy** : Supporte la séparation logique des clients en "organisations" distinctes, permettant d'isoler différents groupes de clients tout en utilisant la même infrastructure.

Avantages du modèle client-serveur de Velociraptor

Ce modèle client-serveur offre plusieurs avantages significatifs :

1. **Efficacité réseau** : En traitant les données localement et en ne transmettant que les résultats pertinents, Velociraptor réduit considérablement la charge réseau.

2. **Évolutivité** : Un seul serveur Velociraptor peut gérer des milliers de clients, grâce à l'architecture distribuée qui décharge une grande partie du traitement vers les clients.

3. **Réactivité** : La connexion persistante entre clients et serveur permet d'exécuter des requêtes et d'obtenir des résultats en temps quasi réel.

4. **Flexibilité** : Le langage VQL permet de créer des requêtes personnalisées pour répondre à pratiquement n'importe quel besoin d'investigation.

5. **Sécurité** : Les communications chiffrées et l'authentification mutuelle entre client et serveur garantissent l'intégrité et la confidentialité des données.

Velociraptor Query Language (VQL)

Le Velociraptor Query Language (VQL) est le cœur de la puissance et de la flexibilité de Velociraptor. Il s'agit d'un langage de requête spécialement conçu pour les tâches de forensique numérique et de réponse aux incidents.

Principes fondamentaux du VQL

VQL s'inspire de SQL dans sa syntaxe, mais est spécifiquement adapté aux besoins de la forensique numérique. Ses principes fondamentaux incluent :

1. **Syntaxe inspirée de SQL** : Utilise la structure familière `SELECT ... FROM ... WHERE`, facilitant son apprentissage pour ceux qui connaissent déjà SQL.
2. **Évaluation paresseuse (lazy evaluation)** : VQL n'évalue les expressions que lorsque c'est nécessaire, ce qui améliore considérablement les performances.
3. **Architecture basée sur des plugins** : Les fonctionnalités sont fournies par des plugins extensibles, permettant d'ajouter facilement de nouvelles capacités.
4. **Traitement par flux (streaming)** : Les données sont traitées en flux, ce qui permet de gérer efficacement de grands volumes de données avec une empreinte mémoire limitée.
5. **Composition** : Les requêtes VQL peuvent être composées et imbriquées, permettant de construire des analyses complexes à partir de blocs simples.

Structure d'une requête VQL

Une requête VQL typique suit cette structure :

```
SELECT column1, column2, ...
FROM plugin(arg1=value1, arg2=value2, ...)
WHERE condition
```

- La clause `SELECT` spécifie les colonnes ou expressions à inclure dans les résultats.
- La clause `FROM` spécifie le plugin VQL à utiliser comme source de données.
- La clause `WHERE` (optionnelle) filtre les résultats selon une condition.

VQL propose également des constructions supplémentaires comme :

- `LET` pour définir des variables
- `GROUP BY` pour regrouper les résultats
- `ORDER BY` pour trier les résultats
- `LIMIT` pour limiter le nombre de résultats

Exemple simple de VQL

Voici un exemple simple de requête VQL qui liste les processus en cours d'exécution :

```
SELECT Pid, Name, CommandLine
FROM pslist()
WHERE Name =~ "chrome"
```

Cette requête : 1. Utilise le plugin `pslist()` pour obtenir la liste des processus 2. Sélectionne les colonnes Pid, Name et CommandLine 3. Filtre pour ne montrer que les processus dont le nom contient "chrome"

Puissance et flexibilité du VQL

La véritable puissance du VQL réside dans sa capacité à combiner différents plugins et fonctions pour créer des requêtes complexes adaptées à des besoins spécifiques. Par exemple :

```
LET suspicious_files = SELECT FullPath
FROM glob(globs="C:/Users/*/Downloads/*.exe")
WHERE ModTime > timestamp(string="2023-01-01")

SELECT * FROM hash_file(filename=FullPath)
FROM suspicious_files
```

Cette requête : 1. Identifie tous les fichiers .exe dans les dossiers de téléchargement des utilisateurs modifiés après le 1er janvier 2023 2. Calcule les hachages de ces fichiers pour une analyse ultérieure

VQL permet ainsi aux analystes de créer rapidement des requêtes personnalisées pour répondre à des besoins d'investigation spécifiques, sans nécessiter de développement logiciel.

Artefacts et collecteurs

Les artefacts sont un concept central dans Velociraptor, représentant des ensembles de requêtes VQL packagées pour collecter et analyser des données spécifiques.

Concept d'artefact

Un artefact Velociraptor est essentiellement un ensemble de requêtes VQL documentées et paramétrables, conçu pour collecter et analyser un type spécifique de données forensiques. Les artefacts peuvent être considérés comme des "recettes" réutilisables pour l'investigation numérique.

Les artefacts sont définis dans un format YAML qui inclut : - Métadonnées (nom, description, auteur, etc.) - Paramètres configurables - Une ou plusieurs requêtes VQL - Documentation sur l'interprétation des résultats

Types d'artefacts

Velociraptor prend en charge plusieurs types d'artefacts :

1. **Artefacts clients** : Exécutés sur les terminaux clients pour collecter des données locales.
2. **Artefacts serveur** : Exécutés sur le serveur pour analyser ou traiter des données déjà collectées.
3. **Artefacts d'événements** : Configurés pour s'exécuter en continu et surveiller des événements spécifiques.

Exemple d'artefact

Voici un exemple simplifié d'artefact pour collecter les fichiers récemment modifiés :

```
 name: Custom.RecentlyModifiedFiles
description: Collects files modified in the last N days
parameters:
  - name: DaysAgo
    default: 7
    type: int
  - name: GlobPattern
    default: C:/Users/*/*.exe
    type: string
sources:
  - query: |
      SELECT FullPath, Size, ModTime
      FROM glob(globs=GlobPattern)
      WHERE ModTime > now() - DaysAgo * 86400
```

Cet artefact : 1. Définit deux paramètres configurables : le nombre de jours à considérer et le motif de recherche de fichiers 2. Utilise ces paramètres dans une requête VQL pour trouver les fichiers récemment modifiés

Avantages des artefacts

L'utilisation d'artefacts présente plusieurs avantages :

1. **Réutilisabilité** : Une fois créé, un artefact peut être utilisé dans différentes investigations.

2. **Standardisation** : Les artefacts fournissent une approche standardisée pour collecter des types spécifiques de données.
3. **Partage de connaissances** : Les artefacts encapsulent l'expertise forensique et peuvent être partagés au sein de la communauté.
4. **Paramétrage** : Les artefacts peuvent être adaptés à des besoins spécifiques via leurs paramètres.
5. **Documentation intégrée** : Les artefacts incluent une documentation sur leur fonctionnement et l'interprétation des résultats.

Échange d'artefacts

Velociraptor dispose d'un "Artifact Exchange" communautaire où les utilisateurs peuvent partager et télécharger des artefacts. Cette ressource est précieuse pour les analystes, car elle leur donne accès à des centaines d'artefacts prêts à l'emploi, couvrant un large éventail de cas d'usage forensiques.

Système de chasse (Hunting)

Le système de chasse (Hunting) est l'une des fonctionnalités les plus puissantes de Velociraptor, permettant d'exécuter des requêtes à grande échelle sur l'ensemble du parc informatique.

Concept de chasse

Une "chasse" dans Velociraptor est une opération qui consiste à exécuter un ou plusieurs artefacts sur un ensemble de clients, généralement pour rechercher des indicateurs de compromission ou collecter des données spécifiques à des fins d'analyse.

Les chasses peuvent être : - **Ciblées** : Exécutées sur un sous-ensemble spécifique de clients - **Globales** : Exécutées sur tous les clients du déploiement - **Planifiées** : Configurées pour s'exécuter à des moments précis - **Limitées en ressources** : Configurées pour limiter leur impact sur les performances des systèmes

Processus de chasse

Le processus typique d'une chasse comprend les étapes suivantes :

1. **Définition** : Sélection des artefacts à collecter et configuration de leurs paramètres
2. **Ciblage** : Sélection des clients sur lesquels exécuter la chasse
3. **Configuration des ressources** : Définition des limites de ressources (CPU, mémoire, bande passante)
4. **Lancement** : Démarrage de la chasse

5. **Surveillance** : Suivi de la progression de la chasse
6. **Analyse** : Examen des résultats collectés

Avantages du système de chasse

Le système de chasse de Velociraptor offre plusieurs avantages significatifs :

1. **Évolutivité** : Capacité à exécuter des requêtes sur des milliers de systèmes simultanément
2. **Efficacité** : Traitement des données directement sur les terminaux, réduisant le trafic réseau
3. **Contrôle des ressources** : Possibilité de limiter l'impact sur les performances des systèmes
4. **Flexibilité** : Utilisation de n'importe quel artefact ou requête VQL personnalisée
5. **Rapidité** : Obtention de résultats en temps quasi réel, même sur de grands déploiements

Cas d'usage typiques

Les cas d'usage courants du système de chasse incluent :

1. **Recherche d'IOC** : Recherche d'indicateurs de compromission spécifiques sur l'ensemble du parc
2. **Évaluation de vulnérabilités** : Identification des systèmes vulnérables à une menace spécifique
3. **Inventaire de logiciels** : Collecte d'informations sur les logiciels installés
4. **Conformité** : Vérification de la conformité des systèmes aux politiques de sécurité
5. **Investigation d'incidents** : Collecte de données forensiques sur l'ensemble du parc pour analyser l'étendue d'une compromission

Le système de chasse transforme Velociraptor d'un simple outil de collecte forensique en une plateforme proactive de chasse aux menaces, permettant aux équipes de sécurité de passer d'une posture réactive à une approche proactive de la sécurité.

3. Installation et déploiement

Prérequis système

Avant d'installer Velociraptor, il est important de s'assurer que votre environnement répond aux prérequis nécessaires pour un fonctionnement optimal. Ces prérequis varient selon que vous installez le serveur ou les clients Velociraptor.

Prérequis pour le serveur Velociraptor

Le serveur Velociraptor est principalement supporté sur les systèmes Linux, bien qu'il puisse techniquement fonctionner sur d'autres plateformes pour des déploiements non-productifs (test, évaluation).

Configuration matérielle recommandée : - **CPU** : 4 cœurs minimum, 8 cœurs ou plus recommandés pour les déploiements importants - **Mémoire** : 8 Go minimum, 16 Go ou plus recommandés pour les déploiements importants - **Espace disque** : Dépend du volume de données collectées, mais prévoir au minimum 100 Go pour un déploiement moyen - **Réseau** : Connexion réseau stable avec les clients, idéalement sur un réseau local ou via VPN pour les déploiements distants

Systèmes d'exploitation supportés pour le serveur : - Ubuntu (18.04 LTS, 20.04 LTS, 22.04 LTS) - recommandé - Debian (10, 11) - CentOS/RHEL (7, 8, 9) - Fedora (récent) - Autres distributions Linux utilisant systemd

Logiciels prérequis : - Systemd (pour l'exécution en tant que service) - Accès réseau aux ports nécessaires (par défaut 8000 pour les communications client-serveur et 8889 pour l'interface web) - Certificats SSL (auto-signés ou émis par une autorité de certification)

Prérequis pour les clients Velociraptor

Les clients Velociraptor sont conçus pour fonctionner sur une variété de systèmes d'exploitation avec des exigences matérielles minimales.

Configuration matérielle minimale : - **CPU** : Tout processeur moderne (x86, x86_64, ARM) - **Mémoire** : 256 Mo minimum, 1 Go recommandé - **Espace disque** : 50 Mo pour l'installation, plus espace pour les données collectées temporairement - **Réseau** : Connexion au serveur Velociraptor (directe ou via proxy)

Systèmes d'exploitation supportés pour les clients : - **Windows** : Windows 7 et versions ultérieures (Windows 10/11 recommandés), Windows Server 2012 et versions

ultérieures - **Linux** : La plupart des distributions modernes (Ubuntu, Debian, CentOS, RHEL, Fedora, etc.) - **macOS** : macOS 10.13 (High Sierra) et versions ultérieures

Privilèges requis : - Le client Velociraptor nécessite généralement des privilèges élevés (administrateur/root) pour accéder à certains artefacts forensiques - Il peut fonctionner avec des privilèges limités, mais certaines fonctionnalités seront restreintes

Installation sur Windows

L'installation de Velociraptor sur Windows peut concerner soit le client (le plus courant), soit le serveur pour des environnements de test ou de développement. Nous allons détailler les deux approches.

Installation du client Velociraptor sur Windows

L'installation du client Velociraptor sur Windows peut se faire de plusieurs façons, mais la méthode la plus courante est l'utilisation d'un package MSI généré par le serveur Velociraptor.

Méthode 1 : Installation via MSI (recommandée)

1. **Obtention du package MSI** :
2. Connectez-vous à l'interface web du serveur Velociraptor
3. Accédez à la section "Server Artifacts"
4. Exécutez l'artefact "Server.Utils.CreateMSI"
5. Téléchargez le fichier MSI généré
6. **Installation silencieuse via ligne de commande** : `msiexec /i velociraptor.msi /quiet`
7. **Installation via interface graphique** :
8. Double-cliquez sur le fichier MSI
9. Suivez les instructions de l'assistant d'installation
10. **Vérification de l'installation** :
11. Ouvrez le Gestionnaire des services (services.msc)
12. Vérifiez que le service "Velociraptor" est en cours d'exécution
13. Dans l'interface web du serveur, vérifiez que le client apparaît dans la liste des clients

Méthode 2 : Installation manuelle

1. **Téléchargement du binaire** :
2. Téléchargez le binaire Windows approprié depuis la page de téléchargement de Velociraptor

3. Téléchargez ou générez un fichier de configuration client

4. **Installation en tant que service** : `velociraptor.exe service install --config client.config.yaml velociraptor.exe service start`

5. **Vérification de l'installation** :

6. Vérifiez que le service est en cours d'exécution
7. Vérifiez la connexion au serveur

Méthode 3 : Déploiement via GPO ou outil de gestion de parc

Pour les déploiements à grande échelle, vous pouvez utiliser : - Stratégies de groupe (GPO) - Microsoft SCCM - PDQ Deploy - Autres outils de gestion de parc

Procédure générale : 1. Générez le package MSI depuis le serveur Velociraptor 2. Distribuez le package via votre outil de déploiement 3. Configurez l'installation silencieuse avec les paramètres appropriés

Installation du serveur Velociraptor sur Windows (pour test uniquement)

Bien que non recommandé pour les environnements de production, le serveur Velociraptor peut être installé sur Windows pour des tests ou des évaluations.

1. **Téléchargement du binaire** :

2. Téléchargez le binaire Windows depuis la page de téléchargement de Velociraptor

3. **Génération de la configuration** : `velociraptor.exe config generate --merge_from_file server.config.yaml`

4. **Installation en tant que service** : `velociraptor.exe service install --config server.config.yaml velociraptor.exe service start`

5. **Accès à l'interface web** :

6. Ouvrez un navigateur et accédez à https://localhost:8889
7. Connectez-vous avec les identifiants par défaut ou ceux configurés

Installation sur Linux

Linux est la plateforme recommandée pour l'exécution du serveur Velociraptor en production. Les clients Linux sont également bien supportés pour la collecte de données forensiques.

Installation du serveur Velociraptor sur Linux

Méthode 1 : Installation via le script d'installation (recommandée)

1. **Téléchargement et exécution du script d'installation** : `bash wget https://github.com/Velocidex/velociraptor/releases/download/v0.6.7/velociraptor-v0.6.7-linux-amd64 chmod +x velociraptor-v0.6.7-linux-amd64 sudo ./velociraptor-v0.6.7-linux-amd64 config generate > server.config.yaml`
2. **Configuration du serveur** :
3. Éditez le fichier server.config.yaml selon vos besoins
4. Configurez les paramètres réseau, d'authentification et de stockage
5. **Installation du service** : `bash sudo ./velociraptor-v0.6.7-linux-amd64 --config server.config.yaml debian server` ou pour les systèmes basés sur RPM : `bash sudo ./velociraptor-v0.6.7-linux-amd64 --config server.config.yaml rpm server`
6. **Démarrage du service** : `bash sudo systemctl start velociraptor_server`
7. **Vérification du fonctionnement** : `bash sudo systemctl status velociraptor_server`
8. **Accès à l'interface web** :
9. Ouvrez un navigateur et accédez à l'URL configurée (par défaut https://localhost:8889)
10. Connectez-vous avec les identifiants configurés

Méthode 2 : Installation manuelle

1. **Téléchargement du binaire** : `bash wget https://github.com/Velocidex/velociraptor/releases/download/v0.6.7/velociraptor-v0.6.7-linux-`

`amd64 chmod +x velociraptor-v0.6.7-linux-amd64 sudo mv velociraptor-v0.6.7-linux-amd64 /usr/local/bin/velociraptor`

2. **Génération de la configuration** : `bash sudo velociraptor config generate > /etc/velociraptor/server.config.yaml`

3. **Création du service systemd** : Créez un fichier `/etc/systemd/system/velociraptor.service` avec le contenu suivant : ``` [Unit] Description=Velociraptor server After=network.target

[Service] Type=simple User=velociraptor Group=velociraptor ExecStart=/usr/local/bin/velociraptor --config /etc/velociraptor/server.config.yaml frontend Restart=always RestartSec=5

[Install] WantedBy=multi-user.target ```

1. **Création de l'utilisateur et des répertoires** : `bash sudo useradd -r -s /bin/false velociraptor sudo mkdir -p /var/lib/velociraptor sudo chown velociraptor:velociraptor /var/lib/velociraptor`

2. **Démarrage et activation du service** : `bash sudo systemctl daemon-reload sudo systemctl start velociraptor sudo systemctl enable velociraptor`

Installation du client Velociraptor sur Linux

Méthode 1 : Installation via package DEB/RPM

1. **Génération du package client depuis le serveur** :
2. Connectez-vous à l'interface web du serveur Velociraptor
3. Accédez à la section "Server Artifacts"
4. Exécutez l'artefact "Server.Utils.CreateDEB" ou "Server.Utils.CreateRPM"

5. Téléchargez le package généré

6. **Installation du package** : Pour Debian/Ubuntu : `bash sudo dpkg -i velociraptor_client.deb`

Pour CentOS/RHEL/Fedora : `bash sudo rpm -i velociraptor_client.rpm`

1. **Vérification de l'installation** : `bash sudo systemctl status velociraptor_client`

Méthode 2 : Installation manuelle

1. **Téléchargement du binaire et de la configuration** : `bash wget https://github.com/Velocidex/velociraptor/releases/download/v0.6.7/velociraptor-v0.6.7-linux-amd64 chmod +x velociraptor-v0.6.7-linux-amd64`
2. **Installation en tant que service** : `bash sudo ./velociraptor-v0.6.7-linux-amd64 --config client.config.yaml debian client` ou `bash sudo ./velociraptor-v0.6.7-linux-amd64 --config client.config.yaml rpm client`
3. **Démarrage du service** : `bash sudo systemctl start velociraptor_client`

Installation sur macOS

Velociraptor prend en charge macOS à la fois pour les clients et, dans une moindre mesure, pour le serveur (bien que Linux soit recommandé pour les déploiements de serveur en production).

Installation du client Velociraptor sur macOS

Méthode 1 : Installation via package PKG

1. **Génération du package client depuis le serveur** :
2. Connectez-vous à l'interface web du serveur Velociraptor
3. Accédez à la section "Server Artifacts"
4. Exécutez l'artefact "Server.Utils.CreateMacOSPkg"
5. Téléchargez le package PKG généré
6. **Installation du package** :
7. Double-cliquez sur le fichier PKG téléchargé
8. Suivez les instructions de l'assistant d'installation
9. Vous devrez peut-être autoriser l'installation dans les préférences de sécurité
10. **Vérification de l'installation** : `bash sudo launchctl list | grep velociraptor`

Méthode 2 : Installation manuelle

1. **Téléchargement du binaire** : `bash curl -L -o velociraptor https://github.com/Velocidex/velociraptor/releases/download/v0.6.7/velociraptor-v0.6.7-darwin-amd64 chmod +x velociraptor`
2. **Téléchargement ou création de la configuration client** :
3. Téléchargez le fichier de configuration client depuis le serveur
4. Ou générez-le manuellement
5. **Installation en tant que service LaunchDaemon** : `bash sudo ./velociraptor service install --config client.config.yaml`
6. **Démarrage du service** : `bash sudo ./velociraptor service start`

Installation du serveur Velociraptor sur macOS (pour test uniquement)

Bien que non recommandé pour les environnements de production, le serveur Velociraptor peut être installé sur macOS pour des tests ou des évaluations.

1. **Téléchargement du binaire** : `bash curl -L -o velociraptor https://github.com/Velocidex/velociraptor/releases/download/v0.6.7/velociraptor-v0.6.7-darwin-amd64 chmod +x velociraptor`
2. **Génération de la configuration** : `bash ./velociraptor config generate > server.config.yaml`
3. **Installation en tant que service** : `bash sudo ./velociraptor service install --config server.config.yaml`
4. **Démarrage du service** : `bash sudo ./velociraptor service start`
5. **Accès à l'interface web** :
6. Ouvrez un navigateur et accédez à https://localhost:8889
7. Connectez-vous avec les identifiants par défaut ou ceux configurés

Configuration du serveur

La configuration correcte du serveur Velociraptor est essentielle pour assurer son bon fonctionnement, sa sécurité et ses performances. Cette section détaille les principales options de configuration et les bonnes pratiques.

Fichier de configuration

Le fichier de configuration du serveur Velociraptor est au format YAML et contient toutes les informations nécessaires au fonctionnement du serveur, y compris les clés cryptographiques, les paramètres réseau et les options d'authentification.

Structure typique du fichier de configuration :

```yaml
Client:
  server_urls:
  - https://velociraptor.example.com:8000/
  ca_certificate: |
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
  nonce: <unique_nonce>

Frontend:
  hostname: velociraptor.example.com
  bind_address: 0.0.0.0
  bind_port: 8000
  certificate: |
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
  private_key: |
    -----BEGIN PRIVATE KEY-----
    ...
    -----END PRIVATE KEY-----

GUI:
  bind_address: 127.0.0.1
  bind_port: 8889
  public_url: https://velociraptor.example.com:8889/

Datastore:
  implementation: FileBaseDataStore
  location: /var/lib/velociraptor
  filestore_directory: /var/lib/velociraptor

Logging:
  output_directory: /var/log/velociraptor
```

Options de configuration importantes

Configuration réseau

- **Frontend.bind_address** : Adresse IP sur laquelle le serveur écoute les connexions des clients (0.0.0.0 pour toutes les interfaces)
- **Frontend.bind_port** : Port sur lequel le serveur écoute les connexions des clients (par défaut 8000)
- **GUI.bind_address** : Adresse IP sur laquelle l'interface web est disponible (127.0.0.1 pour localhost uniquement, 0.0.0.0 pour toutes les interfaces)
- **GUI.bind_port** : Port sur lequel l'interface web est disponible (par défaut 8889)
- **GUI.public_url** : URL publique pour accéder à l'interface web

Configuration de stockage

- **Datastore.implementation** : Type de stockage à utiliser (généralement FileBaseDataStore)
- **Datastore.location** : Emplacement du répertoire de stockage principal
- **Datastore.filestore_directory** : Emplacement du répertoire de stockage des fichiers

Configuration de sécurité

- **Client.ca_certificate** : Certificat CA utilisé pour authentifier le serveur auprès des clients
- **Frontend.certificate** et **Frontend.private_key** : Certificat et clé privée du serveur
- **GUI.certificate** et **GUI.private_key** : Certificat et clé privée pour l'interface web

Configuration d'authentification

- **GUI.authenticator.type** : Type d'authentification (Basic, OIDC, SAML, etc.)
- **GUI.authenticator.users** : Liste des utilisateurs pour l'authentification Basic

Bonnes pratiques de configuration

1. **Sécurité** :
2. Utilisez des certificats émis par une autorité de certification reconnue pour les déploiements en production
3. Limitez l'accès à l'interface web en utilisant un proxy inverse avec authentification supplémentaire
4. Configurez des pare-feu pour restreindre l'accès aux ports du serveur
5. **Performance** :

6. Placez le répertoire de stockage sur un volume à haute performance (SSD)
7. Configurez des limites de ressources appropriées pour éviter la surcharge du serveur
8. Utilisez un système de fichiers adapté aux nombreux petits fichiers (ext4, XFS)
9. **Haute disponibilité** :
10. Configurez des sauvegardes régulières du répertoire de stockage
11. Envisagez d'utiliser un système de fichiers distribué pour le stockage dans les grands déploiements
12. Mettez en place une surveillance du serveur pour détecter les problèmes

Déploiement des clients

Le déploiement efficace des clients Velociraptor est essentiel pour tirer pleinement parti des capacités de la plateforme. Cette section couvre les différentes stratégies et méthodes de déploiement pour divers environnements.

Génération des packages clients

Avant de déployer les clients, vous devez générer des packages d'installation spécifiques à votre déploiement Velociraptor. Ces packages contiennent la configuration nécessaire pour que les clients se connectent à votre serveur.

Génération via l'interface web

1. Connectez-vous à l'interface web du serveur Velociraptor
2. Accédez à la section "Server Artifacts"
3. Recherchez et sélectionnez l'artefact correspondant à votre système d'exploitation :
4. "Server.Utils.CreateMSI" pour Windows
5. "Server.Utils.CreateDEB" pour Debian/Ubuntu
6. "Server.Utils.CreateRPM" pour CentOS/RHEL/Fedora
7. "Server.Utils.CreateMacOSPkg" pour macOS
8. Configurez les options selon vos besoins
9. Exécutez l'artefact et téléchargez le package généré

Génération via ligne de commande

Vous pouvez également générer des packages clients à l'aide de la ligne de commande :

```
velociraptor config client > client.config.yaml
velociraptor --config client.config.yaml debian client
```

Stratégies de déploiement

Déploiement à petite échelle

Pour les petits environnements (moins de 50 clients) : - Installation manuelle sur chaque système - Utilisation de scripts d'installation personnalisés - Déploiement via partage réseau

Déploiement à moyenne échelle

Pour les environnements moyens (50-500 clients) : - Utilisation de stratégies de groupe (GPO) pour Windows - Utilisation de scripts de démarrage pour Linux - Utilisation de profils de configuration pour macOS - Déploiement via des outils comme PDQ Deploy ou Ansible

Déploiement à grande échelle

Pour les grands environnements (plus de 500 clients) : - Intégration avec des solutions de gestion de parc (SCCM, Jamf, etc.) - Utilisation de systèmes de déploiement automatisés - Déploiement par phases pour éviter la surcharge du serveur - Utilisation de scripts de post-installation pour la validation

Méthodes de déploiement par système d'exploitation

Windows

1. **Déploiement via GPO** :
2. Créez une stratégie de groupe pour déployer le MSI
3. Configurez les options d'installation silencieuse

4. Appliquez la GPO aux unités d'organisation appropriées

5. **Déploiement via SCCM** :

6. Créez un package SCCM avec le MSI
7. Configurez les options de déploiement

8. Déployez sur les collections cibles

9. **Déploiement via script PowerShell** : `powershell $url = "https://server/velociraptor.msi" $outpath = "$env:TEMP\velociraptor.msi"`

`Invoke-WebRequest -Uri $url -OutFile $outpath Start-Process msiexec.exe -ArgumentList "/i $outpath /quiet" -Wait`

Linux

1. **Déploiement via Ansible** : ```yaml
2. name: Install Velociraptor client apt: deb: https://server/velociraptor_client.deb state: present when: ansible_os_family == "Debian"
3. name: Install Velociraptor client yum: name: https://server/velociraptor_client.rpm state: present when: ansible_os_family == "RedHat" ```
4. **Déploiement via script shell** : `bash #!/bin/bash if [ -f /etc/debian_version ]; then wget -O /tmp/velociraptor.deb https://server/velociraptor_client.deb dpkg -i /tmp/velociraptor.deb elif [ -f /etc/redhat-release ]; then wget -O /tmp/velociraptor.rpm https://server/velociraptor_client.rpm rpm -i /tmp/velociraptor.rpm fi`

macOS

1. **Déploiement via Jamf** :
2. Ajoutez le package PKG à Jamf
3. Créez une politique de déploiement
4. Déployez sur les groupes cibles
5. **Déploiement via script shell** : `bash #!/bin/bash curl -o /tmp/velociraptor.pkg https://server/velociraptor_client.pkg installer -pkg /tmp/velociraptor.pkg -target /`

Validation du déploiement

Après le déploiement, il est important de valider que les clients se connectent correctement au serveur :

1. Connectez-vous à l'interface web du serveur Velociraptor
2. Accédez à la section "Clients"
3. Vérifiez que les nouveaux clients apparaissent dans la liste
4. Exécutez un artefact simple (comme "Generic.Client.Info") pour confirmer la communication bidirectionnelle

Dépannage du déploiement

Si certains clients ne se connectent pas :

1. **Vérifiez la connectivité réseau** :
2. Assurez-vous que les clients peuvent atteindre le serveur sur le port configuré
3. Vérifiez les pare-feu et les proxys qui pourraient bloquer la connexion
4. **Vérifiez l'installation du client** :
5. Sur Windows, vérifiez que le service Velociraptor est en cours d'exécution
6. Sur Linux/macOS, vérifiez le statut du service avec systemctl ou launchctl
7. **Vérifiez les journaux du client** :
8. Windows : `C:\Program Files\Velociraptor\Velociraptor.log`
9. Linux : `/var/log/velociraptor.log`
10. macOS : `/Library/Logs/velociraptor.log`

Multi-tenancy et organisations

Velociraptor prend en charge la multi-tenancy via son système d'organisations, permettant de gérer plusieurs environnements clients isolés au sein d'une même infrastructure.

Concept d'organisations

Dans Velociraptor, une "organisation" (ou "org") est une entité logique séparée qui contient ses propres clients, utilisateurs, artefacts et données. Les organisations sont complètement isolées les unes des autres, ce qui permet de gérer différents groupes de clients avec différentes exigences de sécurité et d'accès.

Caractéristiques clés des organisations : - Isolation complète des données entre organisations - Contrôle d'accès distinct pour chaque organisation - Artefacts personnalisés spécifiques à chaque organisation - Stockage séparé pour chaque organisation

Organisation racine (root)

Lors de l'installation initiale de Velociraptor, une organisation "root" est automatiquement créée. Cette organisation racine a des privilèges spéciaux :

1. Les administrateurs de l'organisation racine peuvent créer et gérer d'autres organisations
2. Les artefacts personnalisés de l'organisation racine sont automatiquement disponibles dans toutes les organisations enfants

Création de nouvelles organisations

Pour créer une nouvelle organisation :

1. Connectez-vous à l'interface web de Velociraptor en tant qu'administrateur de l'organisation racine
2. Accédez à la section "Server Artifacts"
3. Exécutez l'artefact "Server.Orgs.NewOrg"
4. Spécifiez un nom pour la nouvelle organisation
5. Configurez les options supplémentaires selon vos besoins
6. Exécutez l'artefact pour créer l'organisation

Gestion des utilisateurs dans les organisations

Chaque organisation a son propre ensemble d'utilisateurs et de rôles. Pour ajouter un utilisateur à une organisation :

1. Passez à l'organisation cible à l'aide du sélecteur d'organisation dans l'interface web
2. Accédez à la section "Users"
3. Cliquez sur "Add User"
4. Spécifiez les détails de l'utilisateur et son rôle dans l'organisation

Déploiement de clients pour différentes organisations

Les clients Velociraptor sont configurés pour se connecter à une organisation spécifique via un "nonce" unique inclus dans leur configuration. Pour déployer des clients pour une organisation spécifique :

1. Passez à l'organisation cible à l'aide du sélecteur d'organisation
2. Générez des packages d'installation clients spécifiques à cette organisation en utilisant les artefacts serveur appropriés
3. Déployez ces packages sur les systèmes qui doivent appartenir à cette organisation

Cas d'usage de la multi-tenancy

La multi-tenancy de Velociraptor est particulièrement utile dans les scénarios suivants :

1. **Fournisseurs de services gérés (MSP)** :
2. Gestion de plusieurs clients sur une infrastructure partagée
3. Isolation complète des données entre clients
4. Personnalisation des artefacts et des analyses pour chaque client
5. **Grandes entreprises avec divisions distinctes** :
6. Séparation des environnements de production, de test et de développement
7. Isolation entre différentes unités commerciales ou filiales
8. Conformité avec les exigences réglementaires de séparation des données
9. **Environnements à haute sécurité** :
10. Séparation des systèmes selon leur niveau de classification
11. Contrôle d'accès granulaire basé sur le besoin de savoir
12. Isolation des données sensibles

4. Interface utilisateur

Présentation de l'interface web

L'interface web de Velociraptor est le point central de contrôle pour toutes les opérations de la plateforme. Elle offre une expérience utilisateur intuitive et puissante, permettant aux analystes de sécurité de naviguer efficacement à travers les différentes fonctionnalités du système.

Accès à l'interface web

Pour accéder à l'interface web de Velociraptor :

1. Ouvrez un navigateur web moderne (Chrome, Firefox, Edge, Safari)
2. Accédez à l'URL configurée pour votre serveur Velociraptor (par défaut : https://localhost:8889)
3. Acceptez l'avertissement de certificat si vous utilisez des certificats auto-signés
4. Connectez-vous avec vos identifiants

Structure générale de l'interface

L'interface web de Velociraptor est organisée en plusieurs zones principales :

1. **Barre de navigation latérale** : Située sur le côté gauche de l'écran, elle permet d'accéder aux différentes sections de l'application.
2. **Barre supérieure** : Contient le sélecteur d'organisation, les informations utilisateur, et les options de configuration globales.
3. **Zone de contenu principale** : Occupe la majeure partie de l'écran et affiche le contenu de la section sélectionnée.
4. **Barre d'état** : Située en bas de l'écran, elle affiche des informations sur l'état du serveur et les tâches en cours.

Thème et personnalisation

Velociraptor propose un thème sombre par défaut, optimisé pour réduire la fatigue oculaire lors des longues sessions d'analyse. L'interface peut être personnalisée via les paramètres utilisateur, accessibles depuis le menu déroulant de l'utilisateur dans la barre supérieure.

Compatibilité et optimisation

L'interface web de Velociraptor est conçue pour fonctionner de manière optimale sur les navigateurs modernes et s'adapte à différentes tailles d'écran. Pour une expérience optimale, il est recommandé d'utiliser :

- Un écran avec une résolution minimale de 1920x1080
- Un navigateur web récent avec JavaScript activé
- Une connexion réseau stable vers le serveur Velociraptor

Navigation et fonctionnalités principales

La navigation dans l'interface de Velociraptor est organisée de manière logique pour faciliter l'accès aux différentes fonctionnalités. Voici un aperçu des sections principales et de leur utilisation.

Tableau de bord principal

Le tableau de bord principal est la première page que vous voyez après la connexion. Il offre une vue d'ensemble de votre déploiement Velociraptor, incluant :

- Statistiques sur le nombre de clients connectés
- Graphiques d'activité récente
- Accès rapide aux fonctions fréquemment utilisées
- Notifications et alertes importantes

Section Clients

La section Clients permet de gérer et d'interagir avec les terminaux sur lesquels l'agent Velociraptor est installé :

- **Vue Liste** : Affiche tous les clients avec des informations de base comme le nom d'hôte, l'OS, et le statut de connexion
- **Vue Détaillée** : En cliquant sur un client, vous accédez à une vue détaillée avec :
- Informations système complètes
- Historique des collections
- Possibilité de lancer de nouvelles collections
- Shell VQL pour des requêtes interactives
- Téléchargement de fichiers

Section Chasses (Hunts)

La section Chasses permet de créer et gérer des opérations de collecte à grande échelle :

- **Création de chasse** : Assistant pour configurer une nouvelle chasse
- **Liste des chasses** : Vue d'ensemble de toutes les chasses passées et en cours
- **Détails de chasse** : Statistiques, résultats et gestion d'une chasse spécifique

Section Artefacts

La section Artefacts est dédiée à la gestion des artefacts Velociraptor :

- **Explorateur d'artefacts** : Parcourir les artefacts disponibles par catégorie
- **Création d'artefacts** : Interface pour créer ou modifier des artefacts personnalisés
- **Importation/Exportation** : Outils pour partager des artefacts

Section Notebooks

Les notebooks sont des espaces de travail interactifs pour l'analyse et la documentation :

- **Liste des notebooks** : Vue d'ensemble de tous les notebooks
- **Éditeur de notebook** : Interface pour créer et modifier des notebooks
- **Exécution de VQL** : Cellules interactives pour exécuter du code VQL
- **Visualisation** : Outils pour créer des graphiques et des tableaux

Section Événements

La section Événements permet de surveiller les activités en temps réel :

- **Flux d'événements** : Affichage des événements détectés par les moniteurs
- **Configuration des moniteurs** : Interface pour configurer la surveillance continue
- **Alertes** : Gestion des alertes générées par les événements

Section Serveur

La section Serveur donne accès aux fonctionnalités d'administration :

- **Artefacts serveur** : Exécution d'artefacts sur le serveur lui-même
- **Utilisateurs** : Gestion des comptes utilisateurs et des permissions
- **Configuration** : Paramètres du serveur et options de déploiement
- **Organisations** : Gestion des organisations dans un environnement multi-tenant

Tableaux de bord

Les tableaux de bord dans Velociraptor fournissent des vues synthétiques et des visualisations qui aident à comprendre rapidement l'état du système et les résultats des analyses.

Tableau de bord principal

Le tableau de bord principal offre une vue d'ensemble du déploiement Velociraptor :

1. **Statistiques des clients** :
2. Nombre total de clients
3. Répartition par système d'exploitation
4. Clients en ligne vs hors ligne

5. Dernières connexions

6. **Activité récente** :

7. Chasses récemment lancées
8. Collections récentes

9. Événements importants

10. **État du serveur** :

11. Utilisation des ressources
12. Espace disque disponible

13. Santé du système

14. **Accès rapide** :

15. Liens vers les fonctions fréquemment utilisées
16. Artefacts récemment utilisés
17. Clients récemment consultés

Tableaux de bord des chasses

Chaque chasse dispose de son propre tableau de bord qui présente :

1. **Statistiques d'exécution** :
2. Nombre de clients ciblés
3. Progression de l'exécution

4. Taux de réussite/échec

5. **Visualisations des résultats** :

6. Graphiques récapitulatifs
7. Distributions statistiques

8. Chronologies d'événements

9. **Résultats détaillés** :

10. Tableaux de données filtrables
11. Exportation des résultats
12. Accès aux données brutes

Tableaux de bord des clients

La vue détaillée d'un client inclut plusieurs tableaux de bord spécialisés :

1. **Informations système** :
2. Configuration matérielle
3. Système d'exploitation
4. Utilisateurs et groupes
5. Réseau
6. **Activité** :
7. Processus en cours
8. Connexions réseau
9. Fichiers récemment modifiés
10. **Collections** :
11. Historique des collections
12. Résultats des collections récentes
13. État des collections en cours

Tableaux de bord personnalisés

Velociraptor permet de créer des tableaux de bord personnalisés via les notebooks :

1. **Création** :
2. Utilisation de cellules VQL pour générer des données
3. Ajout de visualisations (graphiques, tableaux)
4. Intégration de texte explicatif
5. **Partage** :
6. Exportation de notebooks
7. Collaboration entre utilisateurs
8. Modèles réutilisables
9. **Automatisation** :
10. Mise à jour programmée
11. Alertes basées sur des conditions
12. Rapports périodiques

Bonnes pratiques pour les tableaux de bord

Pour tirer le meilleur parti des tableaux de bord Velociraptor :

1. **Personnalisation** :
2. Adaptez les tableaux de bord à vos besoins spécifiques
3. Créez des vues dédiées pour différents cas d'usage
4. **Organisation** :
5. Regroupez les informations connexes
6. Utilisez une hiérarchie logique
7. Nommez clairement les tableaux de bord
8. **Optimisation** :
9. Limitez le nombre d'éléments par tableau de bord
10. Privilégiez les visualisations pertinentes
11. Évitez les requêtes trop lourdes pour les mises à jour en temps réel

Gestion des clients

La gestion efficace des clients est essentielle pour tirer pleinement parti de Velociraptor. Cette section détaille les fonctionnalités disponibles pour gérer les terminaux sur lesquels l'agent Velociraptor est installé.

Vue d'ensemble des clients

La section Clients de l'interface web offre une vue complète de tous les clients connectés à votre déploiement Velociraptor :

1. **Liste des clients** :
2. Affichage tabulaire avec filtres et tri
3. Informations de base : nom d'hôte, IP, OS, version
4. Indicateurs d'état : en ligne/hors ligne, dernière connexion
5. Étiquettes personnalisées
6. **Recherche et filtrage** :
7. Recherche par nom d'hôte, IP, étiquette
8. Filtres avancés par OS, version, état

9. Enregistrement des filtres fréquemment utilisés

10. **Actions en masse** :

11. Sélection multiple de clients
12. Application d'étiquettes
13. Lancement de collections sur plusieurs clients
14. Ajout à une chasse existante

Interaction avec un client spécifique

En sélectionnant un client dans la liste, vous accédez à une vue détaillée offrant de nombreuses fonctionnalités :

1. **Informations détaillées** :
2. Configuration matérielle et logicielle
3. Utilisateurs et groupes
4. Configuration réseau

5. Historique des connexions

6. **Collections** :

7. Lancement de collections d'artefacts
8. Visualisation des résultats

9. Historique des collections précédentes

10. **Shell VQL** :

11. Interface interactive pour exécuter des requêtes VQL
12. Historique des commandes

13. Modèles de requêtes prédéfinis

14. **Gestion des fichiers** :

15. Navigation dans le système de fichiers
16. Téléchargement de fichiers
17. Téléversement de fichiers

18. Analyse de fichiers

19. **Surveillance** :

20. Configuration de moniteurs d'événements

21. Visualisation des événements en temps réel
22. Alertes basées sur des conditions

Étiquetage et organisation

L'étiquetage des clients permet de les organiser de manière logique et de faciliter les opérations à grande échelle :

1. **Création d'étiquettes** :
2. Étiquettes personnalisées (ex: "Serveurs", "Postes de travail", "Direction")
3. Étiquettes automatiques basées sur des règles
4. Étiquettes temporaires pour des opérations spécifiques
5. **Utilisation des étiquettes** :
6. Filtrage rapide des clients
7. Ciblage précis pour les chasses
8. Organisation logique du parc
9. **Gestion des étiquettes** :
10. Ajout/suppression d'étiquettes
11. Renommage d'étiquettes
12. Fusion d'étiquettes

Collecte d'artefacts sur les clients

La collecte d'artefacts est l'opération principale effectuée sur les clients :

1. **Sélection d'artefacts** :
2. Parcours du catalogue d'artefacts
3. Recherche par nom ou description
4. Filtrage par catégorie ou plateforme
5. **Configuration des paramètres** :
6. Ajustement des paramètres spécifiques à l'artefact
7. Configuration des limites de ressources
8. Planification de l'exécution
9. **Exécution et suivi** :

10. Lancement de la collecte
11. Suivi de la progression en temps réel

12. Visualisation des résultats intermédiaires

13. **Analyse des résultats** :

14. Visualisation tabulaire ou graphique
15. Filtrage et tri des résultats
16. Exportation pour analyse externe

Gestion du cycle de vie des clients

Velociraptor offre des fonctionnalités pour gérer l'ensemble du cycle de vie des clients :

1. **Déploiement** :
2. Génération de packages d'installation
3. Suivi du déploiement

4. Vérification de la première connexion

5. **Maintenance** :

6. Mise à jour des clients
7. Reconfiguration à distance

8. Diagnostic des problèmes

9. **Suppression** :

10. Désinstallation propre
11. Nettoyage des données
12. Archivage des informations historiques

Bonnes pratiques pour la gestion des clients

Pour une gestion efficace des clients Velociraptor :

1. **Organisation** :
2. Utilisez un système d'étiquetage cohérent
3. Documentez les collections importantes

4. Maintenez une nomenclature claire

5. **Performance** :

6. Évitez les collections trop lourdes sur de nombreux clients simultanément
7. Planifiez les opérations intensives en dehors des heures de pointe

8. Utilisez les limites de ressources appropriées

9. **Sécurité** :

10. Limitez l'accès à la gestion des clients selon le principe du moindre privilège
11. Auditez régulièrement les opérations effectuées sur les clients
12. Documentez les raisons des collections sensibles

Gestion des utilisateurs et permissions

La gestion des utilisateurs et des permissions est un aspect crucial de Velociraptor, particulièrement dans les environnements multi-utilisateurs ou les déploiements à grande échelle. Cette section détaille les fonctionnalités disponibles pour gérer les accès et les privilèges.

Modèle de sécurité

Velociraptor utilise un modèle de sécurité basé sur les rôles (RBAC - Role-Based Access Control) qui permet de définir précisément les actions que chaque utilisateur peut effectuer :

1. **Utilisateurs** : Entités individuelles qui se connectent à Velociraptor
2. **Rôles** : Ensembles de permissions qui définissent les actions autorisées
3. **Organisations** : Conteneurs logiques qui isolent les clients, les artefacts et les données

Types d'authentification

Velociraptor prend en charge plusieurs méthodes d'authentification :

1. **Authentification basique** :
2. Nom d'utilisateur et mot de passe stockés localement
3. Simple à configurer, idéal pour les petits déploiements

4. Moins sécurisé que les méthodes d'authentification externe

5. **Authentification SSO (Single Sign-On)** :

6. Intégration avec des fournisseurs d'identité externes via OIDC ou SAML
7. Support pour Azure AD, Google, Okta, etc.

8. Authentification multifacteur via le fournisseur d'identité

9. **Authentification par certificat** :

10. Utilisation de certificats clients pour l'authentification
11. Haute sécurité pour les environnements sensibles
12. Configuration plus complexe

Rôles prédéfinis

Velociraptor inclut plusieurs rôles prédéfinis avec des niveaux d'accès croissants :

1. **Reader** (Lecteur) :
2. Peut voir les clients et les résultats
3. Ne peut pas lancer de nouvelles collections ou chasses

4. Accès en lecture seule aux notebooks et artefacts

5. **Analyst** (Analyste) :

6. Peut lancer des collections sur des clients individuels
7. Peut créer et modifier des notebooks

8. Ne peut pas lancer de chasses ou modifier des artefacts

9. **Investigator** (Investigateur) :

10. Peut lancer des chasses
11. Peut créer et modifier des artefacts

12. Ne peut pas gérer les utilisateurs ou la configuration du serveur

13. **Administrator** (Administrateur) :

14. Accès complet à toutes les fonctionnalités
15. Peut gérer les utilisateurs et les permissions

16. Peut modifier la configuration du serveur

17. **Org Administrator** (Administrateur d'organisation) :

18. Peut gérer les organisations (uniquement dans l'organisation racine)
19. Peut créer de nouvelles organisations
20. Peut attribuer des administrateurs aux organisations

Gestion des utilisateurs

L'interface d'administration des utilisateurs permet de :

1. **Créer des utilisateurs** :
2. Définition du nom d'utilisateur
3. Attribution de rôles
4. Configuration des paramètres spécifiques
5. **Modifier les utilisateurs** :
6. Changement de rôle
7. Réinitialisation de mot de passe
8. Activation/désactivation de compte
9. **Supprimer des utilisateurs** :
10. Révocation de tous les accès
11. Conservation de l'historique des actions

Permissions granulaires

Au-delà des rôles prédéfinis, Velociraptor permet de définir des permissions granulaires pour des cas d'usage spécifiques :

1. **Permissions par organisation** :
2. Un utilisateur peut avoir différents rôles dans différentes organisations
3. Isolation complète entre organisations
4. **Permissions par artefact** :
5. Restriction de l'accès à certains artefacts sensibles
6. Limitation des paramètres modifiables
7. **Permissions par client** :
8. Restriction de l'accès à certains groupes de clients
9. Utilisation d'étiquettes pour définir les périmètres d'accès

Audit et traçabilité

Velociraptor maintient un journal complet des actions des utilisateurs pour assurer la traçabilité :

1. **Journal d'audit** :
2. Enregistrement de toutes les actions importantes
3. Horodatage et identification de l'utilisateur
4. Conservation à long terme
5. **Visualisation des actions** :
6. Interface dédiée pour consulter le journal d'audit
7. Filtrage par utilisateur, action, date
8. Exportation pour analyse externe
9. **Alertes de sécurité** :
10. Détection des comportements suspects
11. Notification en cas d'actions sensibles
12. Intégration possible avec des SIEM externes

Bonnes pratiques de gestion des utilisateurs

Pour une gestion efficace et sécurisée des utilisateurs :

1. **Principe du moindre privilège** :
2. Attribuez uniquement les permissions nécessaires
3. Utilisez des rôles spécifiques plutôt que des administrateurs généraux
4. Réévaluez régulièrement les permissions
5. **Séparation des tâches** :
6. Distinguez les rôles d'administration et d'analyse
7. Implémentez des validations pour les actions sensibles
8. Documentez les raisons des élévations de privilèges
9. **Intégration avec les processus existants** :
10. Alignez la gestion des utilisateurs avec les processus RH
11. Automatisez la création et la suppression des comptes
12. Synchronisez avec les annuaires d'entreprise existants

5. Velociraptor Query Language (VQL)

Fondamentaux du VQL

Le Velociraptor Query Language (VQL) est le cœur de la puissance et de la flexibilité de Velociraptor. C'est un langage de requête spécialement conçu pour les tâches de forensique numérique et de réponse aux incidents, permettant aux analystes de créer des requêtes personnalisées pour collecter et analyser des données sur les terminaux.

Origine et philosophie

VQL a été développé pour répondre aux limitations des outils forensiques traditionnels, qui nécessitent souvent le développement de nouveaux modules ou plugins pour chaque nouveau type d'analyse. La philosophie de VQL est de fournir un langage flexible et expressif qui permet aux analystes de créer rapidement des requêtes adaptées à leurs besoins spécifiques, sans nécessiter de développement logiciel.

Les principes fondamentaux de VQL incluent :

1. **Simplicité** : Une syntaxe inspirée de SQL, familière pour de nombreux professionnels
2. **Extensibilité** : Une architecture basée sur des plugins qui peut être facilement étendue
3. **Efficacité** : Un traitement optimisé pour minimiser l'impact sur les systèmes analysés
4. **Expressivité** : La capacité de décrire des analyses complexes de manière concise

Syntaxe de base

La syntaxe de VQL est délibérément similaire à SQL pour faciliter son apprentissage. Une requête VQL typique suit cette structure :

```
SELECT column1, column2, ...
FROM plugin(arg1=value1, arg2=value2, ...)
WHERE condition
```

- La clause `SELECT` spécifie les colonnes ou expressions à inclure dans les résultats
- La clause `FROM` spécifie le plugin VQL à utiliser comme source de données
- La clause `WHERE` (optionnelle) filtre les résultats selon une condition

Contrairement à SQL, VQL n'impose pas de restrictions sur l'utilisation des espaces blancs, ce qui permet de formater les requêtes pour une meilleure lisibilité :

```
SELECT
  Name, Size, ModTime
FROM
  glob(globs="C:/Users/*/*.exe")
WHERE
  ModTime > timestamp(string="2023-01-01")
```

Variables et expressions

VQL permet de définir des variables à l'aide de la clause `LET` :

```
LET recent_files = SELECT FullPath, Size, ModTime
FROM glob(globs="C:/Users/*/*.exe")
WHERE ModTime > timestamp(string="2023-01-01")

SELECT * FROM recent_files
```

Les variables peuvent contenir : - Des valeurs simples (chaînes, nombres, booléens) - Des tableaux de valeurs - Des objets (dictionnaires clé-valeur) - Des résultats de requêtes (comme dans l'exemple ci-dessus)

Les expressions en VQL peuvent utiliser divers opérateurs : - Arithmétiques : `+`, `-`, `*`, `/` - Comparaison : `=`, `!=`, `<`, `>`, `<=`, `>=` - Logiques : `AND`, `OR`, `NOT` - Correspondance de motifs : `=~` (expression régulière)

Plugins et fonctions

VQL s'appuie sur deux types de composants extensibles :

1. **Plugins** : Sources de données qui génèrent des lignes de résultats
2. Exemple : `pslist()`, `glob()`, `registry()`
3. **Fonctions** : Traitent des données et retournent des valeurs
4. Exemple : `timestamp()`, `hash()`, `parse_pe()`

Les plugins et fonctions acceptent des arguments nommés :

```
SELECT FullPath, hash(path=FullPath, algorithm="MD5") AS MD5
FROM glob(globs="C:/Windows/*.exe")
```

Évaluation paresseuse

Une caractéristique importante de VQL est l'évaluation paresseuse (lazy evaluation), qui retarde l'évaluation des expressions jusqu'à ce qu'elles soient réellement nécessaires. Cela permet d'optimiser les performances en évitant les calculs inutiles.

Par exemple, dans la requête suivante :

```
 SELECT FullPath, Size, hash(path=FullPath) AS Hash
FROM glob(globs="C:/Windows/*.exe")
WHERE Size > 1000000
```

La fonction `hash()` ne sera évaluée que pour les fichiers dont la taille est supérieure à 1 Mo, ce qui évite de calculer des hachages pour des fichiers qui seront filtrés.

Flux de données (Streaming)

VQL traite les données en flux (streaming), ce qui signifie que les lignes sont traitées une par une, sans nécessiter de charger l'ensemble des résultats en mémoire. Cela permet de traiter efficacement de grands volumes de données avec une empreinte mémoire limitée.

Cette approche est particulièrement importante pour les opérations forensiques, où les ensembles de données peuvent être très volumineux.

Syntaxe et structure

La syntaxe de VQL, bien qu'inspirée de SQL, présente des particularités importantes à comprendre pour exploiter pleinement sa puissance. Cette section détaille les éléments syntaxiques et structurels du langage.

Structure d'une requête complète

Une requête VQL complète peut inclure plusieurs clauses et constructions :

```
 LET variable = expression

SELECT
  column1 AS alias1,
  column2 AS alias2,
  { SELECT nested FROM plugin() } AS nested_column
FROM
  plugin(arg1=value1, arg2=value2)
WHERE
```

```
  condition
GROUP BY
  column1, column2
ORDER BY
  column1 DESC
LIMIT 10
```

Clause SELECT

La clause `SELECT` spécifie les colonnes ou expressions à inclure dans les résultats :

```
SELECT Name, Size, ModTime
```

Vous pouvez utiliser `*` pour sélectionner toutes les colonnes :

```
SELECT *
```

Les colonnes peuvent être renommées à l'aide de `AS` :

```
SELECT Name AS FileName, Size AS FileSize
```

Vous pouvez également inclure des expressions calculées :

```
SELECT Name, Size / 1024 / 1024 AS SizeMB
```

Clause FROM

La clause `FROM` spécifie la source de données, généralement un plugin VQL :

```
FROM glob(globs="C:/Users/*/*.exe")
```

Les plugins peuvent prendre plusieurs arguments :

```
FROM hash_file(
  filename=FullPath,
  algorithms=["MD5", "SHA1", "SHA256"]
)
```

Clause WHERE

La clause `WHERE` filtre les résultats selon une condition :

```
WHERE Size > 1000000 AND Name =~ "chrome"
```

Les conditions peuvent utiliser divers opérateurs : - Égalité : `=`, `!=` - Comparaison : `<`, `>`, `<=`, `>=` - Logiques : `AND`, `OR`, `NOT` - Correspondance de motifs : `=~` (expression régulière) - Inclusion : `IN` (pour les tableaux)

Clause GROUP BY

La clause `GROUP BY` regroupe les résultats selon une ou plusieurs colonnes :

```
SELECT User, COUNT() AS FileCount
FROM glob(globs="C:/Users/*/*.exe")
GROUP BY User
```

Lorsque vous utilisez `GROUP BY`, vous pouvez utiliser des fonctions d'agrégation comme : - `COUNT()` : Nombre de lignes - `SUM(column)` : Somme des valeurs - `MIN(column)` : Valeur minimale - `MAX(column)` : Valeur maximale - `AVG(column)` : Moyenne des valeurs

Clause ORDER BY

La clause `ORDER BY` trie les résultats selon une ou plusieurs colonnes :

```
ORDER BY Size DESC, Name ASC
```

Vous pouvez spécifier la direction du tri : - `ASC` : Ordre croissant (par défaut) - `DESC` : Ordre décroissant

Clause LIMIT

La clause `LIMIT` limite le nombre de résultats retournés :

```
LIMIT 10
```

Requêtes imbriquées

VQL permet d'imbriquer des requêtes à l'intérieur d'autres requêtes :

```
SELECT Name, {
  SELECT COUNT(*) FROM hash_file(filename=FullPath)
} AS HashCount
FROM glob(globs="C:/Windows/*.exe")
```

Les requêtes imbriquées peuvent être utilisées dans : - La clause `SELECT` (comme ci-dessus) - La clause `FROM` (comme source de données) - La clause `WHERE` (comme condition)

Commentaires

VQL prend en charge les commentaires de style SQL :

```
-- Ceci est un commentaire sur une ligne

/*
  Ceci est un commentaire
  sur plusieurs lignes
*/
```

Chaînes de caractères

VQL prend en charge plusieurs formats de chaînes de caractères :

1. **Chaînes simples** avec guillemets doubles ou simples : `"Ceci est une chaîne"` `'Ceci est aussi une chaîne'`
2. **Chaînes brutes** avec trois guillemets simples, qui préservent les sauts de ligne et n'interprètent pas les caractères d'échappement : `''' Ceci est une chaîne brute sur plusieurs lignes Les caractères comme \n ne sont pas interprétés '''`

Identifiants avec espaces ou caractères spéciaux

Si un identifiant (nom de colonne, etc.) contient des espaces ou des caractères spéciaux, vous pouvez l'entourer de backticks (`) :

```
SELECT `File Name`, `Last Modified`
FROM plugin()
```

Types de données

VQL manipule différents types de données, chacun avec ses propres caractéristiques et opérations associées. Comprendre ces types est essentiel pour écrire des requêtes efficaces et éviter les erreurs.

Types de base

Chaînes de caractères (String)

Les chaînes représentent du texte et sont délimitées par des guillemets simples ou doubles :

```
"Ceci est une chaîne"
'Ceci est aussi une chaîne'
```

Opérations courantes sur les chaînes : - Concaténation : `"Hello" + " " + "World"` - Correspondance de motifs : `Name =~ "^chrome.*\\.exe$"` - Fonctions de manipulation : `split()`, `join()`, `encode()`, `decode()`

Nombres (Number)

VQL prend en charge les nombres entiers et à virgule flottante :

```
42        // Entier
3.14159   // Flottant
```

Opérations courantes sur les nombres : - Arithmétiques : `+`, `-`, `*`, `/`, `%` (modulo) - Comparaison : `<`, `>`, `<=`, `>=`, `=`, `!=` - Fonctions mathématiques : `round()`, `floor()`, `ceil()`

Booléens (Boolean)

Les booléens représentent des valeurs de vérité :

```
TRUE
FALSE
```

Opérations courantes sur les booléens : - Logiques : `AND`, `OR`, `NOT`

Null

La valeur `NULL` représente l'absence de valeur :

```
NULL
```

Pour tester si une valeur est NULL :

```
 WHERE column = NULL    // Incorrect
WHERE column IS NULL   // Correct
```

Types composites

Tableaux (Array)

Les tableaux sont des collections ordonnées de valeurs, délimitées par des crochets :

```
 [1, 2, 3, 4, 5]
["a", "b", "c"]
[TRUE, FALSE, TRUE]
```

Opérations courantes sur les tableaux : - Accès par index : `array[0]` (les indices commencent à 0) - Test d'appartenance : `value IN array` - Fonctions : `len()`, `append()`, `filter()`

Dictionnaires (Dict/Object)

Les dictionnaires sont des collections de paires clé-valeur, délimitées par des accolades :

```
{
  "name": "John",
  "age": 30,
  "active": TRUE
}
```

Opérations courantes sur les dictionnaires : - Accès par clé : `dict.key` ou `dict["key"]` - Test d'existence de clé : `key IN dict`

Ensembles (Set)

Les ensembles sont des collections non ordonnées de valeurs uniques :

```
set(items=[1, 2, 3, 3, 4])  // Résultat : {1, 2, 3, 4}
```

Opérations courantes sur les ensembles : - Test d'appartenance : `value IN set` - Opérations ensemblistes : union, intersection, différence

Types spéciaux

Horodatages (Timestamp)

Les horodatages représentent des dates et heures :

```
timestamp(string="2023-01-15T14:30:00Z")
```

Opérations courantes sur les horodatages : - Comparaison : `time1 < time2` - Fonctions : `now()`, `timestamp()`, `format()`

Flux de données (Stored Query)

Les requêtes stockées représentent des flux de données qui peuvent être traités ultérieurement :

```
 LET query = SELECT * FROM pslist()
SELECT Name FROM query WHERE Pid < 1000
```

Conversion de types

VQL effectue certaines conversions de types automatiquement, mais vous pouvez également les forcer explicitement :

```
 int(string="42")              // Conversion en entier
float(string="3.14")          // Conversion en flottant
string(value=42)              // Conversion en chaîne
bool(value="true")            // Conversion en booléen
timestamp(string="2023-01-15") // Conversion en horodatage
```

Vérification de types

Pour vérifier le type d'une valeur, vous pouvez utiliser la fonction `type()` :

```
SELECT type(value=column) AS ColumnType
FROM plugin()
```

Fonctions et plugins

Les fonctions et plugins constituent le cœur de la puissance de VQL, fournissant les capacités de collecte et d'analyse de données. Cette section présente les fonctions et plugins les plus couramment utilisés et explique comment les utiliser efficacement.

Différence entre fonctions et plugins

Dans VQL, il existe une distinction importante entre les fonctions et les plugins :

- **Fonctions** : Traitent des données et retournent une valeur unique
- Utilisées dans les expressions (SELECT, WHERE, etc.)
- Exemple : `hash(path="file.exe")` retourne une valeur de hachage
- **Plugins** : Sources de données qui génèrent des lignes de résultats
- Utilisés après la clause FROM
- Exemple : `FROM pslist()` génère une liste de processus

Plugins fondamentaux

Système de fichiers

- **glob()** : Recherche de fichiers par motif `SELECT * FROM glob(globs="C:/Users/*/*.exe")`
- **stat()** : Informations sur un fichier `SELECT * FROM stat(filename="C:/Windows/notepad.exe")`
- **read_file()** : Lecture du contenu d'un fichier `SELECT * FROM read_file(filename="C:/Windows/system.ini")`
- **parse_csv()**, **parse_json()**, **parse_xml()** : Analyse de fichiers structurés `SELECT * FROM parse_csv(filename="data.csv")`

Système et processus

- **pslist()** : Liste des processus en cours d'exécution `SELECT Pid, Name, CommandLine FROM pslist()`

- **netstat()** : Connexions réseau actives `SELECT Pid, Process, LocalAddr, RemoteAddr FROM netstat()`
- **info()** : Informations sur le système `SELECT OS, Hostname, Architecture FROM info()`
- **users()** : Comptes utilisateurs `SELECT Name, Description, UUID FROM users()`

Windows spécifique

- **registry()** : Accès au registre Windows `SELECT * FROM registry(root="HKEY_LOCAL_MACHINE", key="SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run")`
- **wmi()** : Requêtes WMI (Windows Management Instrumentation) `SELECT * FROM wmi(query="SELECT * FROM Win32_Service WHERE State='Running'")`
- **prefetch()** : Analyse des fichiers prefetch `SELECT * FROM prefetch()`
- **usn()** : Journal USN (Update Sequence Number) `SELECT * FROM usn(drive="C:")`

Linux spécifique

- **proc()** : Accès au système de fichiers /proc `SELECT * FROM proc()`
- **dpkg()** : Packages installés (Debian/Ubuntu) `SELECT * FROM dpkg()`
- **rpm()** : Packages installés (RHEL/CentOS/Fedora) `SELECT * FROM rpm()`

macOS spécifique

- **plist()** : Analyse des fichiers plist `SELECT * FROM plist(filename="/Library/Preferences/com.apple.SoftwareUpdate.plist")`
- **spotlight()** : Recherche via Spotlight `SELECT * FROM spotlight(query="kind:application")`

Fonctions essentielles

Manipulation de chaînes

- **split()** : Divise une chaîne en tableau `SELECT split(string="a,b,c", sep=",") AS Parts`

- **join()** : Joint les éléments d'un tableau en chaîne `SELECT join(array=["a", "b", "c"], sep="-") AS Joined`
- **regex_replace()** : Remplace par expression régulière `SELECT regex_replace(source="Hello World", re="World", replace="VQL") AS Result`

Hachage et cryptographie

- **hash()** : Calcule le hachage d'une chaîne ou d'un fichier `SELECT hash(path="file.exe", algorithm="SHA256") AS SHA256`
- **encrypt()**, **decrypt()** : Chiffrement et déchiffrement `SELECT encrypt(string="secret", key="password") AS Encrypted`

Date et heure

- **now()** : Horodatage actuel `SELECT now() AS CurrentTime`
- **timestamp()** : Conversion en horodatage `SELECT timestamp(string="2023-01-15") AS Date`
- **format()** : Formatage d'horodatage `SELECT format(format="%Y-%m-%d", epoch=now()) AS FormattedDate`

Conversion et vérification

- **int()**, **float()**, **string()**, **bool()** : Conversion de types `SELECT int(string="42") AS Number`
- **type()** : Retourne le type d'une valeur `SELECT type(value=Column) AS ColumnType`

Création de plugins et fonctions personnalisés

VQL permet de créer des fonctions personnalisées à l'aide de la clause `LET` :

```
LET my_function(x, y) = x * y + 10

SELECT my_function(x=5, y=3) AS Result
```

Pour des cas plus complexes, vous pouvez utiliser des fonctions anonymes :

```
LET my_complex_function = func(x, y) {
  LET result = x * y
  IF result > 10 THEN
    return result + 10
  ELSE
    return result
  END
}

SELECT my_complex_function(x=5, y=3) AS Result
```

Bonnes pratiques pour l'utilisation des fonctions et plugins

1. **Documentation** : Consultez la documentation officielle pour connaître tous les arguments disponibles
2. **Performance** : Utilisez des plugins spécifiques plutôt que des solutions génériques
3. **Filtrage** : Filtrez les données le plus tôt possible dans la chaîne de traitement
4. **Réutilisation** : Créez des fonctions pour le code réutilisé fréquemment
5. **Lisibilité** : Nommez clairement vos fonctions et variables personnalisées

Requêtes événementielles

Les requêtes événementielles sont un aspect puissant de VQL qui permet de surveiller en continu les systèmes pour détecter des événements spécifiques. Contrairement aux requêtes standard qui s'exécutent une seule fois et retournent un résultat, les requêtes événementielles s'exécutent en continu et génèrent des résultats chaque fois qu'un événement d'intérêt se produit.

Concept de requête événementielle

Une requête événementielle en VQL est une requête qui utilise un plugin événementiel comme source de données. Ces plugins sont conçus pour générer des événements au fil du temps, plutôt que de retourner un ensemble de résultats statique.

La structure d'une requête événementielle est similaire à celle d'une requête standard, mais elle utilise un plugin événementiel dans la clause FROM :

```
 SELECT * FROM evtx(path="C:/Windows/System32/winevt/Logs/
Security.evtx")
WHERE EventID = 4624
```

Cette requête surveille le journal de sécurité Windows et génère un résultat chaque fois qu'un événement de connexion réussie (ID 4624) est enregistré.

Plugins événementiels courants

Surveillance de fichiers

- **watch_csv()** : Surveille un fichier CSV pour les nouvelles lignes `SELECT * FROM watch_csv(filename="/var/log/apache2/access.log", accessor="auto")`
- **watch()** : Surveille un répertoire pour les modifications de fichiers `SELECT * FROM watch(path="C:/Users/*/Downloads")`

Journaux d'événements Windows

- **evtx()** : Surveille les journaux d'événements Windows `SELECT * FROM evtx(path="C:/Windows/System32/winevt/Logs/Security.evtx")`
- **wmi_events()** : Surveille les événements WMI `SELECT * FROM wmi_events(query="SELECT * FROM __InstanceCreationEvent WITHIN 1 WHERE TargetInstance ISA 'Win32_Process'")`

Surveillance système

- **process_tracker()** : Surveille la création et la terminaison de processus `SELECT * FROM process_tracker()`
- **dns_resolver()** : Surveille les résolutions DNS `SELECT * FROM dns_resolver()`
- **usn_watcher()** : Surveille le journal USN pour les modifications de fichiers `SELECT * FROM usn_watcher(drive="C:")`

Surveillance réseau

- **netstat_listener()** : Surveille les nouvelles connexions réseau `SELECT * FROM netstat_listener()`
- **http_client()** : Effectue des requêtes HTTP périodiques `SELECT * FROM http_client(url="https://example.com/api", method="GET", period=60)`

Création de moniteurs d'événements

Les moniteurs d'événements sont des artefacts spéciaux qui encapsulent des requêtes événementielles et peuvent être déployés sur les clients Velociraptor pour une surveillance continue.

Structure typique d'un moniteur d'événements :

```
 name: Custom.ProcessMonitor
description: Monitors process creation events
sources:
  - name: ProcessEvents
    query: |
      SELECT Timestamp, Name, CommandLine, Pid, Ppid
      FROM process_tracker()
      WHERE CommandLine =~ "powershell -enc"
```

Lorsqu'un moniteur d'événements est déployé, il s'exécute en arrière-plan sur le client et envoie des résultats au serveur chaque fois qu'un événement correspondant est détecté.

Traitement des événements

Les événements générés par les requêtes événementielles peuvent être traités de différentes manières :

1. **Filtrage** : Utilisez la clause WHERE pour filtrer les événements pertinents `WHERE EventID = 4624 AND TargetUserName != "SYSTEM"`
2. **Enrichissement** : Ajoutez des informations supplémentaires aux événements `SELECT *, { SELECT Description FROM known_processes WHERE Name = ProcessName } AS ProcessInfo`
3. **Agrégation** : Regroupez les événements pour détecter des modèles `SELECT SourceIP, COUNT(*) AS FailureCount FROM evtx(path="Security.evtx") WHERE EventID = 4625 GROUP BY SourceIP`
4. **Corrélation** : Combinez des événements de différentes sources ``` LET login_events = SELECT * FROM evtx(path="Security.evtx") WHERE EventID = 4624 LET process_events = SELECT * FROM process_tracker()

SELECT login.TargetUserName, process.CommandLine FROM login_events AS login JOIN process_events AS process ON login.TargetLogonId = process.LogonId ```

Bonnes pratiques pour les requêtes événementielles

1. **Performance** :
2. Filtrez les événements le plus tôt possible
3. Limitez le nombre de moniteurs actifs simultanément
4. Évitez les opérations coûteuses dans les requêtes événementielles
5. **Fiabilité** :
6. Gérez les cas d'erreur et les exceptions
7. Utilisez des périodes de récupération pour les sources instables
8. Testez les moniteurs dans des environnements contrôlés avant déploiement
9. **Gestion des données** :
10. Définissez des stratégies de rétention pour les événements
11. Utilisez l'agrégation pour réduire le volume de données
12. Priorisez les événements critiques

Bonnes pratiques et optimisation

L'écriture de requêtes VQL efficaces et maintenables est essentielle pour tirer le meilleur parti de Velociraptor. Cette section présente les bonnes pratiques et techniques d'optimisation pour améliorer la performance, la lisibilité et la maintenabilité de vos requêtes.

Optimisation des performances

Filtrage précoce

Filtrez les données le plus tôt possible dans la chaîne de traitement pour réduire la quantité de données à traiter :

```
 -- Moins efficace
SELECT * FROM glob(globs="C:/**/*.exe")
WHERE ModTime > timestamp(string="2023-01-01")

-- Plus efficace
SELECT * FROM glob(globs="C:/**/*.exe",
                   accessor="ntfs",
                   modify_time=timestamp(string="2023-01-01"))
```

De nombreux plugins VQL acceptent des arguments de filtrage qui sont appliqués au niveau du plugin, ce qui est généralement plus efficace qu'un filtrage ultérieur avec WHERE.

Utilisation de plugins spécialisés

Utilisez des plugins spécialisés plutôt que des solutions génériques :

```
 -- Moins efficace
SELECT FullPath FROM glob(globs="C:/Windows/System32/*.dll")
WHERE FullPath =~ "ntdll.dll$"

-- Plus efficace
SELECT FullPath FROM glob(globs="C:/Windows/System32/ntdll.dll")
```

Limitation des résultats

Utilisez LIMIT pour restreindre le nombre de résultats lorsque vous n'avez pas besoin de l'ensemble complet :

```
SELECT * FROM pslist() LIMIT 10
```

Éviter les calculs redondants

Utilisez LET pour stocker les résultats intermédiaires et éviter les calculs redondants :

```
 -- Moins efficace
SELECT Name, hash(path=FullPath) FROM glob(globs="*.exe")
WHERE hash(path=FullPath) =~ "^deadbeef"

-- Plus efficace
SELECT Name, Hash FROM glob(globs="*.exe")
LET Hash = hash(path=FullPath)
WHERE Hash =~ "^deadbeef"
```

Utilisation d'index et d'accesseurs optimisés

Velociraptor propose différents "accesseurs" pour accéder aux données. Utilisez l'accesseur le plus approprié pour votre cas d'usage :

```
 -- Utilise l'accesseur NTFS pour un accès direct au système de
fichiers
SELECT * FROM glob(globs="C:/**/*.exe", accessor="ntfs")
```

```
-- Utilise l'accesseur de registre pour un accès direct au
registre
SELECT * FROM glob(globs="HKEY_LOCAL_MACHINE/SOFTWARE/**",
accessor="registry")
```

Lisibilité et maintenabilité

Formatage et indentation

Utilisez une indentation cohérente et des sauts de ligne pour améliorer la lisibilité :

```
SELECT
  Name,
  Size,
  ModTime,
  hash(path=FullPath) AS Hash
FROM
  glob(globs="C:/Windows/*.exe")
WHERE
  Size > 1000000
  AND ModTime > timestamp(string="2023-01-01")
ORDER BY
  Size DESC
```

Commentaires

Ajoutez des commentaires pour expliquer le but et la logique de vos requêtes :

```
-- Cette requête identifie les exécutables Windows récemment
modifiés
-- qui pourraient indiquer une compromission
SELECT
  FullPath,
  ModTime,
  hash(path=FullPath) AS Hash
FROM
  glob(globs="C:/Windows/System32/*.exe")
WHERE
  -- Fichiers modifiés après l'installation typique
  ModTime > timestamp(string="2023-01-01")
```

Variables nommées

Utilisez des variables nommées pour décomposer les requêtes complexes en parties plus compréhensibles :

```
LET system_exes = SELECT * FROM glob(globs="C:/Windows/System32/
*.exe")

LET recent_exes = SELECT * FROM system_exes
WHERE ModTime > timestamp(string="2023-01-01")

LET suspicious_exes = SELECT *, hash(path=FullPath) AS Hash
FROM recent_exes
WHERE Hash NOT IN known_hashes

SELECT * FROM suspicious_exes
```

Fonctions réutilisables

Créez des fonctions pour le code réutilisé fréquemment :

```
LET is_suspicious_file(path) = (
  SELECT * FROM stat(filename=path)
  WHERE ModTime > timestamp(string="2023-01-01")
      AND Size < 1000
)

SELECT * FROM foreach(
  row={
    SELECT FullPath FROM glob(globs="C:/Windows/System32/*.exe")
  },
  query={
    SELECT * FROM is_suspicious_file(path=FullPath)
  }
)
```

Gestion des erreurs

Vérification d'existence

Vérifiez l'existence des fichiers ou des clés de registre avant d'y accéder :

```
LET file_path = "C:/Windows/System32/config.sys"

SELECT * FROM if(
  condition={
    SELECT * FROM stat(filename=file_path)
  },
  then={
    SELECT * FROM read_file(filename=file_path)
  },
  else={
```

```
    SELECT "File does not exist" AS Error
  }
)
```

Gestion des valeurs NULL

Gérez explicitement les valeurs NULL pour éviter les erreurs :

```
SELECT
  Name,
  IF(condition=Size, then=Size, else=0) AS Size
FROM
  glob(globs="C:/**/*.exe")
```

Validation des entrées

Validez les entrées utilisateur avant de les utiliser dans des requêtes :

```
LET user_input = "potentially malicious input"

SELECT * FROM if(
  condition=user_input =~ "^[a-zA-Z0-9_]+$",
  then={
    -- Utilisation sécurisée de l'entrée
    SELECT * FROM glob(globs=user_input)
  },
  else={
    SELECT "Invalid input" AS Error
  }
)
```

Techniques avancées d'optimisation

Parallélisation

Utilisez `thread()` pour exécuter des requêtes en parallèle :

```
SELECT * FROM thread(
  thread_count=10,
  query={
    SELECT * FROM glob(globs="C:/Users/*/*.exe")
  }
)
```

Mise en cache

Utilisez `cache()` pour mettre en cache les résultats de requêtes coûteuses :

```
LET expensive_query = SELECT * FROM glob(globs="C:/**/*.exe")

SELECT * FROM cache(
  query=expensive_query,
  key="exe_files",
  ttl=600  -- Cache valide pendant 10 minutes
)
```

Traitement par lots

Traitez les données par lots pour réduire l'utilisation de la mémoire :

```
SELECT * FROM foreach(
  row={
    SELECT FullPath FROM glob(globs="C:/**/*.exe")
  },
  query={
    SELECT *, hash(path=FullPath) AS Hash FROM scope()
  },
  batch=100  -- Traitement par lots de 100 fichiers
)
```

Mesure et profilage

Pour optimiser vos requêtes, il est important de mesurer leurs performances :

```
LET start_time = now()

-- Votre requête ici
SELECT * FROM glob(globs="C:/**/*.exe")

SELECT
  now() - start_time AS ExecutionTime,
  format(format="%v seconds", args=now() - start_time) AS
FormattedTime
```

Vous pouvez également utiliser le plugin `profile()` pour des informations de profilage plus détaillées :

```
SELECT * FROM profile(
  query={
```

```
    SELECT * FROM glob(globs="C:/**/*.exe")
  }
)
```

6. Artefacts

Concept d'artefact

Les artefacts sont l'un des concepts les plus importants et puissants de Velociraptor. Ils représentent des ensembles de requêtes VQL packagées, documentées et paramétrables, conçus pour collecter et analyser des types spécifiques de données forensiques.

Définition et objectif

Un artefact Velociraptor peut être considéré comme une "recette" réutilisable pour l'investigation numérique. Il encapsule l'expertise forensique sous forme de requêtes VQL, permettant même aux utilisateurs moins expérimentés d'effectuer des analyses complexes sans avoir à maîtriser tous les détails techniques.

Les artefacts servent plusieurs objectifs :

1. **Standardisation** : Ils fournissent une approche cohérente pour collecter des types spécifiques de données.
2. **Réutilisation** : Une fois créés, ils peuvent être utilisés dans différentes investigations.
3. **Partage de connaissances** : Ils permettent de capturer et partager l'expertise forensique.
4. **Automatisation** : Ils peuvent être exécutés automatiquement dans le cadre de chasses ou de surveillances.
5. **Documentation** : Ils incluent des explications sur leur fonctionnement et l'interprétation des résultats.

Structure d'un artefact

Les artefacts sont définis dans un format YAML qui comprend plusieurs sections :

```
 name: Windows.System.Processes
description: |
  Collecte la liste des processus en cours d'exécution sur un
système Windows.
```

```
  Cette liste inclut des informations détaillées comme le PID,
le nom du processus,
  la ligne de commande, et l'utilisateur exécutant le processus.

parameters:
  - name: ProcessRegex
    description: Expression régulière pour filtrer les noms de
processus
    type: regex
    default: .

sources:
  - name: Processes
    description: Liste des processus en cours d'exécution
    query: |
      SELECT
        Pid,
        Name,
        CommandLine,
        Username
      FROM
        pslist()
      WHERE
        Name =~ ProcessRegex
```

Les principales sections d'un artefact sont :

1. **name** : Identifiant unique de l'artefact, généralement sous forme de chemin hiérarchique (ex: Windows.System.Processes)
2. **description** : Documentation détaillée sur l'artefact, son objectif et son interprétation
3. **parameters** : Définition des paramètres configurables par l'utilisateur
4. **sources** : Une ou plusieurs requêtes VQL qui constituent le cœur de l'artefact
5. **precondition** (optionnel) : Condition préalable pour déterminer si l'artefact est applicable
6. **reports** (optionnel) : Définition de visualisations pour les résultats

Types d'artefacts

Velociraptor prend en charge plusieurs types d'artefacts, chacun avec un objectif spécifique :

1. **Artefacts clients** : Exécutés sur les terminaux clients pour collecter des données locales.

2. Exemple : Windows.System.Processes, Linux.Sys.Users

3. **Artefacts serveur** : Exécutés sur le serveur pour analyser ou traiter des données déjà collectées.

4. Exemple : Server.Utils.CreateMSI, Server.Forensic.Timeline

5. **Artefacts d'événements** : Configurés pour s'exécuter en continu et surveiller des événements spécifiques.

6. Exemple : Windows.Events.ProcessCreation, Linux.Events.FileModification

7. **Artefacts de notebook** : Conçus pour être exécutés dans des notebooks pour l'analyse interactive.

8. Exemple : Notebook.VirusTotal.LookupHash, Notebook.System.ListArtifacts

Cycle de vie d'un artefact

Le cycle de vie typique d'un artefact comprend les étapes suivantes :

1. **Création** : Développement de l'artefact en définissant ses paramètres et requêtes VQL
2. **Test** : Validation du fonctionnement de l'artefact dans différents environnements
3. **Déploiement** : Ajout de l'artefact au catalogue de Velociraptor
4. **Utilisation** : Exécution de l'artefact dans le cadre d'investigations ou de chasses
5. **Maintenance** : Mise à jour de l'artefact pour corriger des bugs ou ajouter des fonctionnalités
6. **Partage** : Publication de l'artefact dans l'Artifact Exchange pour la communauté

Création d'artefacts personnalisés

La création d'artefacts personnalisés est l'une des compétences les plus précieuses pour un utilisateur de Velociraptor. Elle permet d'adapter l'outil à des besoins spécifiques et d'automatiser des tâches d'investigation récurrentes.

Préparation et planification

Avant de créer un artefact, il est important de bien définir son objectif et sa portée :

1. **Définir l'objectif** : Quel type de données souhaitez-vous collecter ou analyser ?
2. **Identifier les sources de données** : Quels fichiers, registres ou autres sources contiennent ces données ?
3. **Déterminer les paramètres** : Quels aspects de l'artefact devraient être configurables par l'utilisateur ?

4. **Planifier la structure** : Comment organiser les requêtes VQL pour atteindre l'objectif ?

Création via l'interface web

Velociraptor offre une interface graphique pour créer et tester des artefacts :

1. Connectez-vous à l'interface web de Velociraptor
2. Accédez à la section "Artifacts" dans le menu principal
3. Cliquez sur "New Artifact" pour ouvrir l'éditeur d'artefacts
4. Remplissez les différentes sections (nom, description, paramètres, sources)
5. Utilisez le bouton "Test" pour valider le fonctionnement de l'artefact
6. Cliquez sur "Save" pour ajouter l'artefact au catalogue

L'éditeur d'artefacts offre des fonctionnalités utiles comme la coloration syntaxique, l'auto-complétion et la validation du format YAML.

Structure détaillée d'un artefact

Examinons plus en détail les différentes sections d'un artefact :

En-tête et métadonnées

```
name: Custom.Windows.BrowserHistory
description: |

Collecte l'historique de navigation des principaux navigateurs
web sur Windows.

  Navigateurs supportés :
  - Google Chrome
  - Mozilla Firefox
  - Microsoft Edge

author: Votre Nom
reference:
  - https://example.com/browser-forensics
  - https://example.com/chrome-artifacts
```

Les métadonnées incluent : - **name** : Identifiant unique, généralement avec un préfixe indiquant la catégorie - **description** : Documentation détaillée, formatée en Markdown - **author** (optionnel) : Créateur de l'artefact - **reference** (optionnel) : Liens vers des ressources externes pertinentes

Paramètres

```yaml
parameters:
  - name: DateAfter
    description: Ne collecter que l'historique après cette date
    type: timestamp
    default: "2023-01-01"

  - name: UserFilter
    description: Expression régulière pour filtrer les profils
utilisateurs
    type: regex
    default: .

  - name: BrowserSelection
    description: Navigateurs à inclure dans la collecte
    type: choices
    default: ALL
    choices:
      - ALL
      - CHROME
      - FIREFOX
      - EDGE
```

Les paramètres peuvent avoir différents types : - **string** : Chaîne de caractères - **int** : Nombre entier - **bool** : Valeur booléenne (vrai/faux) - **timestamp** : Date et heure - **regex** : Expression régulière - **choices** : Sélection parmi une liste prédéfinie - **csv** : Valeurs séparées par des virgules

Précondition

```yaml
precondition: |
  SELECT OS FROM info() WHERE OS = 'windows'
```

La précondition est une requête VQL qui détermine si l'artefact est applicable. Si elle ne retourne aucun résultat, l'artefact est ignoré.

Sources

```yaml
sources:
  - name: ChromeHistory
    description: Historique de navigation Chrome
    precondition: |
      SELECT * FROM scope() WHERE BrowserSelection = 'ALL' OR
BrowserSelection = 'CHROME'
    query: |
```

```
      LET chrome_history = SELECT * FROM glob(
        globs="C:/Users/*/AppData/Local/Google/Chrome/User Data/
*/History")

      SELECT
        Timestamp,
        Url,
        Title,
        VisitCount,
        'Chrome' AS Browser,
        Username
      FROM sqlite(
        file=chrome_history.FullPath,
        query="SELECT url, title, visit_count, last_visit_time
FROM urls")
      WHERE Timestamp > timestamp(string=DateAfter)

  - name: FirefoxHistory
    description: Historique de navigation Firefox
    precondition: |
      SELECT * FROM scope() WHERE BrowserSelection = 'ALL' OR
BrowserSelection = 'FIREFOX'
    query: |
      -- Requête VQL pour Firefox
```

Chaque source représente une requête VQL distincte qui peut : - Avoir sa propre précondition - Accéder aux paramètres définis dans l'artefact - Retourner un ensemble de résultats indépendant

Rapports et visualisations

```
reports:
  - type: TIMELINE
    template: |
      {{ .Timestamp }} {{ .Browser }} visited {{ .Url }}
({{ .Title }})

  - type: HTML
    template: |
      <h3>Browser History Summary</h3>
      <table class="table">
        <tr>
          <th>Browser</th>
          <th>Count</th>
        </tr>
        {{ range . | group "Browser" }}
          <tr>
            <td>{{ .Browser }}</td>
            <td>{{ len . }}</td>
          </tr>
```

```
        {{ end }}
      </table>
```

Les rapports définissent comment les résultats sont présentés : - **TIMELINE** : Format pour l'affichage dans une chronologie - **HTML** : Visualisation HTML personnalisée - **SERVER_EVENT** : Format pour les événements serveur - **MONITORING_DAILY** : Rapport quotidien pour la surveillance

Bonnes pratiques pour la création d'artefacts

1. **Nommage cohérent** :
2. Utilisez un préfixe de catégorie (Windows, Linux, MacOS, Generic)
3. Suivez une hiérarchie logique (System.Network.Connections)
4. Utilisez des noms descriptifs et clairs
5. **Documentation complète** :
6. Expliquez clairement l'objectif de l'artefact
7. Documentez les sources de données utilisées
8. Fournissez des conseils pour l'interprétation des résultats
9. Incluez des références externes pertinentes
10. **Paramétrage flexible** :
11. Rendez les artefacts adaptables à différents scénarios
12. Fournissez des valeurs par défaut sensées
13. Documentez chaque paramètre
14. **Performance et efficacité** :
15. Optimisez les requêtes VQL pour minimiser l'impact
16. Utilisez des préconditions pour éviter les exécutions inutiles
17. Filtrez les données le plus tôt possible
18. **Modularité** :
19. Divisez les artefacts complexes en sources multiples
20. Réutilisez les requêtes VQL communes via des variables LET
21. Envisagez de créer des artefacts composites qui en appellent d'autres
22. **Test approfondi** :

23. Testez sur différentes versions de systèmes d'exploitation
24. Vérifiez le comportement avec différentes valeurs de paramètres
25. Validez les résultats par rapport à d'autres outils

Utilisation des artefacts prédéfinis

Velociraptor est livré avec une vaste bibliothèque d'artefacts prédéfinis qui couvrent de nombreux cas d'usage courants en forensique numérique et réponse aux incidents. Cette section explore comment utiliser efficacement ces artefacts.

Exploration du catalogue d'artefacts

L'interface web de Velociraptor offre plusieurs façons d'explorer les artefacts disponibles :

1. **Vue par catégorie** :
2. Accédez à la section "Artifacts" dans le menu principal
3. Parcourez les artefacts organisés par catégorie (Windows, Linux, MacOS, etc.)
4. Utilisez les flèches d'expansion pour explorer les sous-catégories
5. **Recherche** :
6. Utilisez la barre de recherche pour trouver des artefacts par nom ou description
7. Filtrez par type d'artefact (client, serveur, événement)
8. Filtrez par système d'exploitation
9. **Vue détaillée** :
10. Cliquez sur un artefact pour voir sa description complète
11. Consultez les paramètres disponibles et leurs valeurs par défaut
12. Examinez le code VQL sous-jacent pour comprendre son fonctionnement

Artefacts essentiels par catégorie

Voici une sélection d'artefacts prédéfinis particulièrement utiles, organisés par catégorie :

Système et configuration

- **Generic.Client.Info** : Informations de base sur le système client
- **Windows.System.Pslist** : Liste des processus en cours d'exécution
- **Windows.System.Services** : Services Windows configurés

- **Linux.Sys.Users** : Comptes utilisateurs sur les systèmes Linux
- **MacOS.System.LaunchAgents** : Agents de lancement macOS (persistance)

Réseau

- **Windows.Network.NetstatEnriched** : Connexions réseau avec enrichissement des processus
- **Windows.Network.DNSCache** : Cache DNS du système
- **Linux.Network.Netstat** : Connexions réseau sur Linux
- **Generic.Network.InterfaceAddresses** : Adresses des interfaces réseau

Fichiers et système de fichiers

- **Windows.Search.FileFinder** : Recherche avancée de fichiers
- **Windows.Forensics.Prefetch** : Analyse des fichiers prefetch
- **Windows.Forensics.NTFS** : Analyse de la table MFT
- **Linux.Forensics.FileTimeline** : Chronologie des fichiers sur Linux

Registre Windows

- **Windows.Registry.NTUser** : Analyse des ruches de registre utilisateur
- **Windows.Registry.AppCompatCache** : Cache de compatibilité des applications
- **Windows.Registry.UserAssist** : Historique d'exécution des programmes

Journaux et événements

- **Windows.EventLogs.EvtxHunter** : Recherche dans les journaux d'événements Windows
- **Windows.EventLogs.PowershellHistory** : Historique des commandes PowerShell
- **Linux.Syslog.SearchLog** : Recherche dans les journaux système Linux

Navigateurs web

- **Windows.Forensics.ChromeHistory** : Historique de navigation Chrome
- **Windows.Forensics.FirefoxHistory** : Historique de navigation Firefox
- **Windows.Forensics.BrowserDownloads** : Téléchargements des navigateurs

Malware et menaces

- **Windows.Detection.PsexecService** : Détection des services PsExec
- **Windows.Forensics.Amcache** : Analyse du cache AMCache
- **Windows.Detection.ProcessInjection** : Détection d'injection de processus
- **Generic.Detection.YaraScanner** : Analyse de fichiers avec Yara

Exécution d'artefacts sur des clients

Pour exécuter un artefact sur un client spécifique :

1. Accédez à la section "Clients" dans le menu principal
2. Sélectionnez le client cible dans la liste
3. Cliquez sur "Collect" pour ouvrir l'assistant de collecte
4. Recherchez et sélectionnez l'artefact souhaité
5. Configurez les paramètres selon vos besoins
6. Cliquez sur "Launch" pour démarrer la collecte

L'interface affichera la progression de la collecte et les résultats au fur et à mesure qu'ils sont reçus.

Exécution d'artefacts via des chasses

Pour exécuter un artefact sur plusieurs clients simultanément :

1. Accédez à la section "Hunt Manager" dans le menu principal
2. Cliquez sur "New Hunt" pour ouvrir l'assistant de chasse
3. Sélectionnez les artefacts à collecter
4. Configurez les paramètres pour chaque artefact
5. Définissez les critères de ciblage des clients (tous, par étiquette, par expression régulière)
6. Configurez les limites de ressources pour minimiser l'impact
7. Lancez la chasse

Vous pourrez suivre la progression de la chasse et consulter les résultats agrégés.

Configuration des paramètres d'artefacts

La plupart des artefacts prédéfinis offrent des paramètres configurables pour adapter leur comportement à vos besoins spécifiques :

1. **Paramètres de filtrage** :
2. Expressions régulières pour cibler des fichiers ou processus spécifiques
3. Plages de dates pour limiter la collecte à une période

4. Listes d'inclusions ou d'exclusions

5. **Paramètres de performance** :

6. Limites de taille de fichier
7. Nombre maximal de résultats

8. Profondeur de recherche récursive

9. **Paramètres de comportement** :

10. Options d'upload de fichiers
11. Niveau de détail des résultats
12. Modes d'analyse spécifiques

Interprétation des résultats

Les résultats des artefacts peuvent être consultés et analysés de plusieurs façons :

1. **Vue tabulaire** :
2. Affichage par défaut des résultats
3. Tri et filtrage des colonnes

4. Exportation au format CSV ou JSON

5. **Visualisations** :

6. Graphiques et diagrammes pour certains artefacts
7. Chronologies pour les artefacts temporels

8. Vues personnalisées définies dans l'artefact

9. **Notebooks** :

10. Création de notebooks pour une analyse approfondie
11. Combinaison de résultats de plusieurs artefacts

12. Ajout de contexte et de documentation

13. **Post-traitement** :

14. Utilisation d'artefacts serveur pour analyser les résultats
15. Corrélation entre différentes collectes
16. Exportation pour analyse dans d'autres outils

Bonnes pratiques pour l'utilisation des artefacts

1. **Commencez par des artefacts génériques** :
2. Generic.Client.Info pour les informations de base
3. Generic.System.VirtualFileSystem pour explorer le système de fichiers

4. Generic.Client.Stats pour évaluer les performances du client

5. **Utilisez des artefacts ciblés** :

6. Préférez des artefacts spécifiques plutôt que des collectes générales
7. Limitez la portée avec des paramètres appropriés

8. Collectez uniquement ce dont vous avez besoin

9. **Planifiez vos collectes** :

10. Évaluez l'impact potentiel avant de lancer des collectes à grande échelle
11. Planifiez les collectes intensives en dehors des heures de pointe

12. Utilisez des limites de ressources appropriées

13. **Documentez vos investigations** :

14. Notez les artefacts utilisés et leur configuration
15. Documentez les raisons de chaque collecte
16. Conservez les résultats importants dans des notebooks

Partage et importation d'artefacts

Le partage d'artefacts est un aspect important de l'écosystème Velociraptor, permettant à la communauté de bénéficier collectivement de l'expertise forensique. Cette section explore les mécanismes de partage et d'importation d'artefacts.

Exportation d'artefacts

Pour partager un artefact que vous avez créé ou modifié :

1. **Exportation via l'interface web** :
2. Accédez à la section "Artifacts" dans le menu principal
3. Sélectionnez l'artefact à exporter

4. Cliquez sur le bouton "Export" pour télécharger le fichier YAML

5. **Exportation manuelle** :

6. Les artefacts sont stockés dans le répertoire de données de Velociraptor
7. Vous pouvez copier directement les fichiers YAML

8. Format standard : `nom_artefact.yaml`

9. **Exportation programmatique** :

10. Utilisez l'API Velociraptor pour exporter des artefacts

11. Utilisez des artefacts serveur comme `Server.Utils.ExportArtifact`

Importation d'artefacts

Pour ajouter un artefact externe à votre déploiement Velociraptor :

1. **Importation via l'interface web** :
2. Accédez à la section "Artifacts" dans le menu principal
3. Cliquez sur le bouton "Import"
4. Sélectionnez le fichier YAML à importer
5. Vérifiez et confirmez les détails de l'artefact
6. **Importation en masse** :
7. Vous pouvez importer plusieurs artefacts à la fois
8. Sélectionnez un dossier contenant des fichiers YAML
9. Velociraptor validera chaque artefact avant l'importation
10. **Importation programmatique** :
11. Utilisez l'API Velociraptor pour importer des artefacts
12. Utilisez des artefacts serveur comme `Server.Utils.ImportArtifact`

Velociraptor Artifact Exchange

L'Artifact Exchange est un référentiel communautaire d'artefacts Velociraptor partagés par des utilisateurs du monde entier :

1. **Accès à l'Artifact Exchange** :
2. Via le site web de Velociraptor
3. Directement depuis l'interface web (section "Artifact Exchange")
4. Sur GitHub : https://github.com/Velocidex/velociraptor-docs/tree/master/exchange
5. **Navigation et recherche** :
6. Parcourez les artefacts par catégorie
7. Recherchez par nom, description ou auteur
8. Filtrez par système d'exploitation ou type d'artefact
9. **Téléchargement et importation** :

10. Téléchargez les artefacts individuellement
11. Importez-les dans votre déploiement Velociraptor
12. Testez-les dans un environnement contrôlé avant utilisation en production

Contribution à l'Artifact Exchange

Pour contribuer vos propres artefacts à la communauté :

1. **Préparation de l'artefact** :
2. Assurez-vous que l'artefact est bien documenté
3. Testez-le sur différentes versions de systèmes d'exploitation
4. Vérifiez qu'il ne contient pas d'informations sensibles
5. **Soumission via GitHub** :
6. Créez un fork du dépôt Velociraptor Docs
7. Ajoutez votre artefact au répertoire exchange
8. Soumettez une pull request avec une description détaillée
9. **Révision et publication** :
10. Les mainteneurs examineront votre soumission
11. Des modifications peuvent être demandées
12. Une fois approuvé, votre artefact sera disponible pour tous

Gestion des versions d'artefacts

Lorsque vous travaillez avec des artefacts partagés ou personnalisés, la gestion des versions devient importante :

1. **Versionnement explicite** :
2. Incluez un numéro de version dans la description
3. Documentez les changements entre versions
4. Utilisez un préfixe de version dans le nom pour les changements majeurs
5. **Contrôle des modifications** :
6. Gardez une trace des modifications apportées aux artefacts
7. Documentez les raisons des changements
8. Testez les nouvelles versions avant déploiement
9. **Compatibilité** :

10. Vérifiez la compatibilité avec différentes versions de Velociraptor
11. Notez les dépendances sur des plugins ou fonctions spécifiques
12. Testez sur différentes plateformes si l'artefact se veut multiplateforme

Sécurité et validation

Lorsque vous importez des artefacts externes, il est important de prendre des précautions :

1. **Révision du code** :
2. Examinez le code VQL avant d'exécuter l'artefact
3. Vérifiez qu'il n'y a pas de comportement malveillant
4. Comprenez ce que fait l'artefact avant de l'utiliser
5. **Test dans un environnement contrôlé** :
6. Testez d'abord sur un système isolé
7. Vérifiez l'impact sur les performances
8. Validez les résultats avant utilisation à grande échelle
9. **Validation des sources** :
10. Préférez les artefacts de sources réputées
11. Vérifiez l'auteur et les contributions précédentes
12. Soyez particulièrement vigilant avec les artefacts qui exécutent des commandes système

Échange d'artefacts communautaires

L'échange d'artefacts communautaires est un aspect fondamental de l'écosystème Velociraptor, permettant aux utilisateurs de bénéficier de l'expertise collective et d'accélérer leurs investigations. Cette section explore plus en détail l'Artifact Exchange et les meilleures pratiques pour l'utilisation d'artefacts communautaires.

L'écosystème de l'Artifact Exchange

L'Artifact Exchange de Velociraptor est une plateforme dynamique qui continue d'évoluer :

1. **Origine et évolution** :
2. Créé pour faciliter le partage d'expertise entre analystes
3. Croissance constante avec des contributions régulières

4. Évolution vers un standard de facto pour les artefacts forensiques

5. **Organisation et structure** :

6. Artefacts organisés par catégories (Windows, Linux, macOS, Generic)
7. Sous-catégories fonctionnelles (Forensics, Detection, Triage)

8. Métadonnées riches pour faciliter la découverte

9. **Gouvernance et qualité** :

10. Processus de révision par les mainteneurs
11. Standards de documentation et de code
12. Retours de la communauté pour amélioration continue

Artefacts communautaires populaires

Voici une sélection d'artefacts communautaires particulièrement utiles et populaires :

1. **Windows.Forensics.USNJournal** :
2. Analyse du journal USN pour la chronologie des fichiers
3. Particulièrement utile pour la détection de suppression de fichiers

4. Développé par des experts en forensique NTFS

5. **Windows.Detection.Yara.ProcessMemory** :

6. Analyse la mémoire des processus avec des règles Yara
7. Permet de détecter des malwares qui n'existent qu'en mémoire

8. Hautement configurable avec des règles personnalisées

9. **Linux.Forensics.BashHistory** :

10. Collecte et analyse l'historique des commandes bash
11. Inclut les historiques de tous les utilisateurs

12. Utile pour tracer les activités d'un attaquant

13. **MacOS.Forensics.UnifiedLogs** :

14. Extraction et analyse des Unified Logs de macOS
15. Filtrage avancé par processus, subsystème ou niveau

16. Essentiel pour les investigations sur macOS

17. **Generic.Forensic.Timeline** :

18. Crée une chronologie unifiée à partir de multiples sources
19. Combine les journaux d'événements, MFT, préfetch, etc.
20. Visualisation interactive des événements

Cas d'usage des artefacts communautaires

Les artefacts communautaires peuvent être utilisés dans divers scénarios :

1. **Réponse aux incidents** :
2. Collections initiales pour triage rapide
3. Détection d'indicateurs de compromission spécifiques
4. Collecte de preuves volatiles
5. **Investigations proactives** :
6. Chasses régulières pour détecter des menaces
7. Vérification de conformité
8. Inventaire de logiciels et configurations
9. **Forensique numérique** :
10. Analyse approfondie post-incident
11. Reconstruction de chronologies d'attaques
12. Documentation pour rapports légaux
13. **Automatisation de la sécurité** :
14. Surveillance continue avec des artefacts d'événements
15. Réponse automatisée à certaines détections
16. Collecte périodique de données de sécurité

Adaptation et personnalisation

Les artefacts communautaires servent souvent de point de départ pour des adaptations spécifiques :

1. **Modification pour besoins spécifiques** :
2. Ajout de filtres supplémentaires
3. Extension pour couvrir des applications internes
4. Optimisation pour votre environnement
5. **Combinaison d'artefacts** :

6. Création d'artefacts composites
7. Séquençage de plusieurs collectes
8. Corrélation de résultats de différents artefacts
9. **Intégration avec des flux de travail** :
10. Adaptation pour s'intégrer à vos processus existants
11. Modification du format de sortie pour d'autres outils
12. Ajout de rapports personnalisés

Contribution à la communauté

La contribution d'artefacts à la communauté est un processus enrichissant :

1. **Identification des besoins** :
2. Repérez les lacunes dans les artefacts existants
3. Identifiez des cas d'usage non couverts
4. Pensez aux problèmes récurrents que vous résolvez
5. **Développement collaboratif** :
6. Discutez de vos idées sur les forums ou Discord
7. Sollicitez des retours sur les versions préliminaires
8. Collaborez avec d'autres contributeurs
9. **Documentation et partage** :
10. Créez une documentation claire et complète
11. Incluez des exemples d'utilisation et d'interprétation
12. Soumettez à l'Artifact Exchange avec des explications détaillées
13. **Maintenance continue** :
14. Répondez aux questions des utilisateurs
15. Mettez à jour en fonction des retours
16. Adaptez aux nouvelles versions de Velociraptor

Tendances et évolution

L'écosystème des artefacts communautaires continue d'évoluer :

1. **Spécialisation croissante** :

2. Artefacts pour des menaces spécifiques
3. Couverture de technologies émergentes

4. Analyses de plus en plus sophistiquées

5. **Intégration avec le renseignement sur les menaces** :

6. Artefacts basés sur des IOCs actuels
7. Détection de tactiques d'attaquants spécifiques

8. Mise à jour dynamique basée sur les flux de renseignement

9. **Automatisation et orchestration** :

10. Artefacts conçus pour des workflows automatisés
11. Intégration avec des plateformes SOAR

12. Réponse automatisée basée sur les résultats

13. **Visualisation et reporting avancés** :

14. Rapports interactifs intégrés
15. Visualisations personnalisées
16. Formats adaptés aux différentes parties prenantes

7. Collecte de données

Collecte de fichiers

La collecte de fichiers est l'une des opérations les plus fondamentales et fréquentes dans Velociraptor. Cette section détaille les différentes méthodes et bonnes pratiques pour collecter efficacement des fichiers à partir des terminaux clients.

Méthodes de collecte de fichiers

Velociraptor offre plusieurs approches pour collecter des fichiers, chacune adaptée à des besoins spécifiques :

Collecte directe via l'interface utilisateur

La méthode la plus simple pour collecter des fichiers individuels :

1. Accédez à la vue détaillée d'un client

2. Naviguez dans l'onglet "VFS" (Virtual File System)
3. Parcourez l'arborescence des fichiers
4. Sélectionnez un fichier et cliquez sur "Download"

Cette méthode est idéale pour : - Collecte ad hoc de fichiers spécifiques - Exploration interactive du système de fichiers - Vérification rapide de fichiers suspects

Collecte via artefacts dédiés

Pour des collectes plus structurées et répétables :

1. Utilisez des artefacts comme `Windows.Search.FileFinder` ou `Linux.Search.FileFinder`
2. Spécifiez des critères de recherche (chemins, motifs, taille, date)
3. Configurez les options de collecte (hachage, upload)
4. Exécutez l'artefact sur un ou plusieurs clients

Cette approche est recommandée pour : - Collecte basée sur des critères complexes - Recherche et collecte à grande échelle - Collectes standardisées et documentées

Collecte via requêtes VQL personnalisées

Pour les cas nécessitant une flexibilité maximale :

```
SELECT * FROM foreach(
  row={
    SELECT FullPath FROM glob(globs="C:/Users/*/AppData/Local/
Temp/*.exe")
    WHERE Size < 1000000 AND ModTime >
timestamp(string="2023-01-01")
  },
  query={
    SELECT FullPath, Size, ModTime, upload(file=FullPath) AS
Upload
    FROM scope()
  }
)
```

Cette méthode est adaptée pour : - Logique de collecte complexe - Intégration avec d'autres opérations - Collectes hautement personnalisées

Options et paramètres de collecte

Lors de la collecte de fichiers, plusieurs options permettent d'adapter le comportement selon vos besoins :

Filtrage des fichiers

Velociraptor offre de nombreux critères pour cibler précisément les fichiers à collecter :

1. **Filtrage par chemin et nom** :
2. Motifs glob (ex: `C:/Users/*/*.exe`)
3. Expressions régulières (ex: `.*\.pdf$`)
4. Chemins exacts ou listes de chemins
5. **Filtrage par attributs** :
6. Taille (minimale/maximale)
7. Date de création/modification/accès
8. Attributs de fichier (caché, système, etc.)
9. **Filtrage par contenu** :
10. Correspondance de chaînes ou d'expressions régulières
11. Signatures YARA
12. En-têtes de fichiers (magic bytes)

Méthodes d'acquisition

Différentes méthodes d'acquisition sont disponibles selon les besoins :

1. **Upload complet** :
2. Transfert intégral du fichier vers le serveur
3. Option la plus complète mais potentiellement volumineuse
4. Permet une analyse ultérieure approfondie
5. **Hachage uniquement** :
6. Calcul et collecte des hachages (MD5, SHA1, SHA256)
7. Impact minimal sur le réseau
8. Utile pour la vérification d'intégrité ou la comparaison avec des IOCs
9. **Métadonnées uniquement** :
10. Collecte des attributs sans le contenu (taille, dates, permissions)
11. Très léger en termes de ressources
12. Idéal pour les inventaires ou les chronologies

13. **Échantillonnage** :

14. Collecte partielle (premiers/derniers octets)
15. Compromis entre taille et utilité
16. Utile pour l'identification de types de fichiers

Gestion des ressources

Pour minimiser l'impact sur les systèmes et le réseau :

1. **Limites de taille** :
2. Définition d'une taille maximale pour les fichiers collectés
3. Évite les transferts excessivement volumineux

4. Configurable par artefact ou globalement

5. **Throttling** :

6. Limitation de la bande passante utilisée
7. Réduction de l'impact sur les performances réseau

8. Particulièrement important pour les collectes à grande échelle

9. **Planification** :

10. Exécution des collectes pendant les heures creuses
11. Répartition des collectes volumineuses
12. Priorisation des collectes urgentes

Collecte de fichiers spéciaux

Certains types de fichiers nécessitent des approches spécifiques :

Fichiers système et verrouillés

Pour collecter des fichiers en cours d'utilisation ou verrouillés :

1. **Accesseur NTFS** (Windows) : `vql SELECT * FROM glob(globs="C:/Windows/System32/ntoskrnl.exe", accessor="ntfs")`
2. Contourne les verrous du système d'exploitation
3. Accès direct aux structures NTFS

4. Fonctionne même pour les fichiers système actifs

5. **Shadow Copy** (Windows) : `vql SELECT * FROM glob(globs="\\\\?\\GLOBALROOT\\Device\\HarddiskVolumeShadowCopy1\\Windows\\System32\\config\\SYSTEM", accessor="file")`
6. Utilise les copies instantanées de volume
7. Accès aux fichiers verrouillés via leurs copies
8. Nécessite des privilèges administratifs
9. **Raw Disk Access** (multiplateforme) : `vql SELECT * FROM raw_file(file="/dev/sda1", offset=1234567, length=1024)`
10. Accès direct au périphérique de stockage
11. Permet de récupérer des données même sur des systèmes de fichiers endommagés
12. Nécessite des privilèges élevés

Fichiers supprimés

Pour récupérer des fichiers supprimés mais non écrasés :

1. **Analyse MFT** (Windows) : `vql SELECT * FROM parse_mft(filename="C:/$MFT") WHERE IsDeleted`
2. Identifie les entrées MFT marquées comme supprimées
3. Peut récupérer les métadonnées et parfois le contenu
4. Particulièrement efficace pour les suppressions récentes
5. **Carving de fichiers** : `vql SELECT * FROM file_carve( device="/dev/sda1", signatures=["PDF", "JPEG", "ZIP"] )`
6. Recherche des signatures de fichiers dans l'espace non alloué
7. Peut récupérer des fichiers supprimés depuis longtemps
8. Processus intensif en ressources

Fichiers dans le cloud et synchronisés

Pour les fichiers stockés dans des services cloud :

1. **OneDrive** (Windows) : `vql SELECT * FROM glob(globs="C:/Users/*/OneDrive/**/*.docx") WHERE Attributes.is_placeholder`
2. Détecte les fichiers placeholder vs. téléchargés
3. Peut forcer le téléchargement si nécessaire
4. Utile pour les investigations impliquant des fichiers cloud

5. **Dropbox, Google Drive, etc.** :

6. Identification des répertoires de synchronisation
7. Analyse des bases de données de métadonnées
8. Collecte des fichiers disponibles localement

Bonnes pratiques pour la collecte de fichiers

Pour une collecte efficace et forensiquement saine :

1. **Documentation et chaîne de possession** :
2. Documentez clairement les critères de collecte
3. Enregistrez les hachages pour vérifier l'intégrité

4. Maintenez des métadonnées complètes (dates, sources, etc.)

5. **Minimisation des données** :

6. Collectez uniquement ce qui est nécessaire
7. Utilisez des filtres précis pour limiter le volume

8. Préférez les métadonnées ou hachages quand le contenu n'est pas essentiel

9. **Priorisation** :

10. Commencez par les données volatiles
11. Identifiez et collectez d'abord les fichiers critiques

12. Échelonnez les collectes volumineuses

13. **Validation** :

14. Vérifiez que les fichiers ont été collectés correctement
15. Confirmez l'intégrité via les hachages
16. Assurez-vous que les métadonnées importantes sont préservées

Analyse de la mémoire

L'analyse de la mémoire vive (RAM) est une composante essentielle de la forensique numérique moderne, permettant d'accéder à des informations qui ne sont pas disponibles sur le disque. Velociraptor offre plusieurs méthodes pour capturer et analyser la mémoire des systèmes.

Capture de mémoire

Velociraptor propose différentes approches pour capturer la mémoire, adaptées à divers scénarios :

Capture de mémoire complète

Pour obtenir une image complète de la mémoire physique :

1. **Utilisation de Winpmem** (Windows) : `vql SELECT * FROM winpmem(destination="/tmp/memory.raw")`
2. Capture l'intégralité de la mémoire physique
3. Utilise le pilote winpmem intégré à Velociraptor
4. Nécessite des privilèges administratifs
5. **Utilisation de LiME** (Linux) : `vql SELECT * FROM linux_memory( destination="/tmp/memory.lime", driver="/tmp/lime.ko" )`
6. Module noyau pour la capture mémoire sur Linux
7. Peut être compilé dynamiquement pour le noyau cible
8. Haute fidélité de capture
9. **Utilisation d'outils natifs** (macOS) : `vql SELECT * FROM process_execution( argv=["/usr/bin/sudo", "/usr/sbin/dtrace", "-w", "-n", "BEGIN { printf(\"Dumping memory...\"); } profile-1hz { exit(0); }"] )`
10. Utilise les outils intégrés comme dtrace
11. Approche plus limitée que sur Windows/Linux
12. Peut nécessiter des configurations supplémentaires

Capture de mémoire de processus

Pour cibler la mémoire d'un processus spécifique :

1. **Dump de processus** (Windows) : `vql SELECT * FROM process_dump(pid=1234, destination="/tmp/process.dmp")`
2. Capture l'espace mémoire d'un processus spécifique
3. Beaucoup plus léger qu'une capture complète
4. Idéal pour l'analyse ciblée de malware

5. **Procdump** (Linux) : `vql SELECT * FROM process_memory(pid=1234)`

6. Extrait les segments mémoire d'un processus Linux
7. Peut filtrer par type de segment (heap, stack, etc.)
8. Utile pour les investigations ciblées

Analyse de mémoire en direct

Velociraptor permet d'analyser la mémoire directement sur le système, sans nécessiter une capture complète :

Analyse de processus en mémoire

Pour examiner les processus actifs :

1. **Énumération des processus** : `vql SELECT * FROM pslist()`
2. Liste tous les processus en cours d'exécution
3. Inclut PID, PPID, ligne de commande, etc.

4. Point de départ pour l'identification de processus suspects

5. **Analyse des DLLs chargées** (Windows) : `vql SELECT * FROM modules(pid=1234)`

6. Énumère les modules chargés dans un processus
7. Identifie les DLLs potentiellement malveillantes

8. Détecte les injections de code

9. **Analyse des handles** (Windows) : `vql SELECT * FROM handles(pid=1234)`

10. Liste les handles ouverts par un processus
11. Révèle les accès à des fichiers, registres, etc.

12. Utile pour comprendre le comportement d'un processus

13. **Analyse des mappings mémoire** (Linux) : `vql SELECT * FROM proc_maps(pid=1234)`

14. Affiche les mappings mémoire d'un processus Linux
15. Identifie les bibliothèques chargées et régions mémoire
16. Aide à détecter les comportements anormaux

Scan de mémoire

Pour rechercher des motifs spécifiques en mémoire :

1. **Scan YARA** : `vql SELECT * FROM process_yara(pid=1234, rules='''
   rule suspicious_pattern { strings: $s1 = "password" nocase $s2 =
   "credential" nocase condition: any of them } ''')`
2. Recherche des motifs définis par règles YARA
3. Peut cibler un processus spécifique ou tous les processus
4. Hautement personnalisable pour différentes menaces
5. **Recherche de chaînes** : `vql SELECT * FROM
   grep(source=process_memory(pid=1234), pattern="password=")`
6. Recherche des chaînes spécifiques en mémoire
7. Plus simple mais moins flexible que YARA
8. Utile pour les recherches rapides

Analyse post-capture

Une fois la mémoire capturée, Velociraptor peut faciliter son analyse :

Intégration avec des outils d'analyse

Velociraptor peut s'intégrer avec des outils spécialisés :

1. **Volatility** : `vql SELECT * FROM
   process_execution(argv=["volatility3", "-f", "/tmp/memory.raw",
   "windows.pslist.PsList"])`
2. Exécute Volatility sur une image mémoire
3. Permet d'utiliser l'ensemble des plugins Volatility
4. Peut être automatisé pour des analyses complexes
5. **Rekall** : `vql SELECT * FROM process_execution(argv=["rekall", "-
   f", "/tmp/memory.raw", "pslist"])`
6. Alternative à Volatility pour certaines analyses
7. Offre des fonctionnalités complémentaires
8. Utile pour des cas spécifiques

Extraction d'artefacts mémoire

Pour extraire des informations spécifiques des captures mémoire :

1. **Extraction de processus** : `vql SELECT * FROM process_execution( argv=["volatility3", "-f", "/tmp/memory.raw", "windows.pslist.PsList", "--output-file=/tmp/processes.json", "--output=json"] )`
2. Extrait la liste des processus d'une image mémoire
3. Formate les résultats pour analyse ultérieure
4. Base pour des investigations plus approfondies
5. **Extraction de connexions réseau** : `vql SELECT * FROM process_execution( argv=["volatility3", "-f", "/tmp/memory.raw", "windows.netscan.NetScan", "--output-file=/tmp/network.json", "--output=json"] )`
6. Identifie les connexions réseau actives au moment de la capture
7. Révèle des communications potentiellement malveillantes
8. Aide à établir la portée d'une compromission
9. **Extraction de code injecté** : `vql SELECT * FROM process_execution( argv=["volatility3", "-f", "/tmp/memory.raw", "windows.malfind.Malfind", "--output-file=/tmp/malfind.json", "--output=json"] )`
10. Détecte les injections de code en mémoire
11. Identifie les techniques d'évasion avancées
12. Crucial pour l'analyse de malware sophistiqués

Cas d'usage et bonnes pratiques

L'analyse de mémoire est particulièrement utile dans certains scénarios :

Détection de malware avancé

Pour les menaces qui opèrent principalement en mémoire :

1. **Malware sans fichier** :
2. Détection de code exécuté directement en mémoire
3. Identification de scripts PowerShell/JavaScript obfusqués

4. Révélation de charges utiles déchiffrées en mémoire

5. **Rootkits et bootkits** :

6. Identification de modifications du noyau
7. Détection de hooks et patches en mémoire

8. Comparaison des structures en mémoire vs. sur disque

9. **Ransomware** :

10. Capture des clés de chiffrement en mémoire
11. Identification des processus de chiffrement
12. Potentiel de récupération pour certaines variantes

Investigation d'incidents

Pour comprendre l'étendue et la nature d'une compromission :

1. **Établissement de chronologie** :
2. Processus en cours d'exécution au moment de l'incident
3. Connexions réseau actives

4. Utilisateurs connectés et contextes de sécurité

5. **Exfiltration de données** :

6. Identification de données sensibles en mémoire
7. Détection de mécanismes d'exfiltration

8. Évaluation de l'impact potentiel

9. **Persistance et mouvement latéral** :

10. Découverte de mécanismes de persistance chargés
11. Identification d'outils de mouvement latéral
12. Détection de techniques de privilège escalation

Bonnes pratiques pour l'analyse mémoire

Pour maximiser la valeur de l'analyse mémoire :

1. **Priorisation** :
2. Capturez la mémoire le plus tôt possible dans une investigation
3. Commencez par les systèmes les plus critiques ou suspects

4. Préservez la mémoire avant toute action potentiellement disruptive

5. **Contextualisation** :

6. Combinez l'analyse mémoire avec d'autres sources (logs, disque)
7. Documentez l'état du système au moment de la capture

8. Corrélation avec d'autres indicateurs de compromission

9. **Minimisation de l'impact** :

10. Utilisez des captures de processus ciblées quand approprié
11. Planifiez les captures complètes pour minimiser les perturbations

12. Considérez l'impact sur les systèmes critiques

13. **Validation** :

14. Vérifiez l'intégrité des captures mémoire
15. Confirmez les résultats avec plusieurs méthodes d'analyse
16. Documentez la méthodologie pour la reproductibilité

Analyse du registre Windows

Le registre Windows est une base de données hiérarchique qui stocke les paramètres de configuration pour le système d'exploitation Windows et les applications. Il constitue une source extrêmement riche d'artefacts forensiques, rendant son analyse essentielle dans les investigations sur les systèmes Windows.

Structure et accès au registre

Pour comprendre l'analyse du registre, il est important de connaître sa structure fondamentale :

Organisation du registre

Le registre Windows est organisé en ruches (hives) et clés :

1. **Ruches principales** :
2. HKEY_LOCAL_MACHINE (HKLM) : Configuration système
3. HKEY_CURRENT_USER (HKCU) : Configuration de l'utilisateur actuel
4. HKEY_USERS (HKU) : Configuration de tous les utilisateurs
5. HKEY_CLASSES_ROOT (HKCR) : Associations de fichiers et COM

6. HKEY_CURRENT_CONFIG (HKCC) : Configuration matérielle actuelle

7. **Fichiers de ruche** :

8. %SystemRoot%\System32\Config\SAM
9. %SystemRoot%\System32\Config\SECURITY
10. %SystemRoot%\System32\Config\SOFTWARE
11. %SystemRoot%\System32\Config\SYSTEM
12. %UserProfile%\NTUSER.DAT

Accès au registre avec Velociraptor

Velociraptor offre plusieurs méthodes pour accéder au registre :

1. **Plugin registry()** : `vql SELECT * FROM registry( root="HKEY_LOCAL_MACHINE", key="SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run" )`
2. Accès en direct au registre du système
3. Permet de parcourir les clés et valeurs
4. Nécessite des privilèges appropriés
5. **Accesseur registry** : `vql SELECT * FROM glob( globs="HKEY_LOCAL_MACHINE/SOFTWARE/Microsoft/Windows/*/Run/*", accessor="registry" )`
6. Utilise la syntaxe glob pour des recherches plus flexibles
7. Permet des requêtes récursives et avec wildcards
8. Particulièrement utile pour des recherches larges
9. **Analyse de fichiers de ruche** : `vql SELECT * FROM raw_registry( file="C:/Windows/System32/config/SOFTWARE" )`
10. Analyse directe des fichiers de ruche
11. Fonctionne même si le système n'est pas démarré
12. Permet d'analyser des ruches de registre extraites

Artefacts forensiques clés dans le registre

Le registre Windows contient de nombreux artefacts précieux pour les investigations :

Persistance et démarrage automatique

Pour identifier les mécanismes de persistance :

1. **Clés Run et RunOnce** : `vql SELECT * FROM registry( root="HKEY_LOCAL_MACHINE", key="SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run" )`
2. Points de persistance les plus courants
3. Souvent ciblés par les malwares
4. Présents dans HKLM et HKCU
5. **Services Windows** : `vql SELECT * FROM registry( root="HKEY_LOCAL_MACHINE", key="SYSTEM\\CurrentControlSet\\Services" )`
6. Configuration des services système
7. Inclut le chemin de l'exécutable et les paramètres de démarrage
8. Mécanisme de persistance privilégié pour les menaces avancées
9. **Tâches planifiées** : `vql SELECT * FROM registry( root="HKEY_LOCAL_MACHINE", key="SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Schedule\\TaskCache" )`
10. Métadonnées sur les tâches planifiées
11. Complémentaire à l'analyse des fichiers XML de tâches
12. Peut révéler des tâches supprimées

Activité utilisateur et historique

Pour tracer l'activité des utilisateurs :

1. **UserAssist** : `vql SELECT * FROM parse_userassist(key="HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\UserAssist")`
2. Historique des applications exécutées via l'interface graphique
3. Inclut le nombre d'exécutions et horodatages
4. Données encodées en ROT-13 (décodées par Velociraptor)
5. **MRU (Most Recently Used)** : `vql SELECT * FROM registry( root="HKEY_CURRENT_USER", key="Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\RecentDocs" )`

6. Fichiers récemment ouverts
7. Organisés par type de fichier

8. Utile pour établir une chronologie d'activité

9. **TypedPaths** :
   `vql SELECT * FROM registry( root="HKEY_CURRENT_USER", key="Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\TypedPaths" )`

10. Chemins tapés dans l'explorateur Windows
11. Indique l'accès manuel à des répertoires
12. Peut révéler des activités suspectes

Configuration système et réseau

Pour comprendre la configuration du système :

1. **Interfaces réseau** : `vql SELECT * FROM registry( root="HKEY_LOCAL_MACHINE", key="SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\Interfaces" )`
2. Configuration des interfaces réseau
3. Adresses IP, DNS, DHCP

4. Historique des réseaux connectés

5. **Zones de sécurité Internet** : `vql SELECT * FROM registry( root="HKEY_CURRENT_USER", key="Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\ZoneMap\\Domains" )`

6. Sites ajoutés à des zones de sécurité spécifiques
7. Peut indiquer des tentatives de contournement de sécurité

8. Important pour les investigations de phishing

9. **AppCompatCache (ShimCache)** : `vql SELECT * FROM parse_shimcache()`

10. Cache de compatibilité des applications
11. Contient des métadonnées sur les exécutables lancés
12. Conservé même après la suppression des fichiers

Artefacts de sécurité et d'authentification

Pour les investigations liées à la sécurité :

1. **Amcache** : `vql SELECT * FROM parse_amcache()`
2. Historique des applications installées et exécutées
3. Inclut les hachages de fichiers et métadonnées
4. Particulièrement utile pour la détection de malware
5. **SAM (Security Accounts Manager)** : `vql SELECT * FROM raw_registry( file="C:/Windows/System32/config/SAM" )`
6. Base de données des comptes locaux
7. Contient les hachages de mots de passe (nécessite des privilèges élevés)
8. Crucial pour les investigations d'accès non autorisés
9. **SRUM (System Resource Usage Monitor)** : `vql SELECT * FROM parse_srum()`
10. Statistiques d'utilisation des ressources
11. Historique d'exécution des applications
12. Données de consommation réseau par application

Techniques d'analyse avancées

Pour des investigations plus approfondies du registre :

Analyse historique et temporelle

Pour examiner les changements au fil du temps :

1. **Points de restauration système** : `vql SELECT * FROM glob( globs="/System Volume Information/_restore*/RP*/snapshot/_REGISTRY_*", accessor="ntfs" )`
2. Copies historiques des ruches de registre
3. Permet de voir les changements au fil du temps
4. Crucial pour établir des chronologies
5. **Transaction Logs** : `vql SELECT * FROM parse_registry_transaction( registry_file="C:/Windows/System32/config/SOFTWARE", transaction_log="C:/Windows/System32/config/SOFTWARE.LOG1" )`

6. Journaux de transactions du registre
7. Peut révéler des modifications récentes

8. Utile pour la récupération de données

9. **Analyse différentielle** : ```vql LET baseline = SELECT FullPath, Data FROM registry(root="HKEY_LOCAL_MACHINE", key="SOFTWARE\Microsoft\Windows\CurrentVersion\Run")

LET current = SELECT FullPath, Data FROM raw_registry(file="C:/evidence/SYSTEM")

SELECT * FROM diff(a=baseline, b=current) ``` - Compare différentes versions du registre - Identifie les ajouts, modifications et suppressions - Particulièrement utile pour l'analyse d'impact

Détection de techniques d'évasion

Pour identifier les tentatives de dissimulation :

1. **Clés masquées** : `vql SELECT * FROM registry( root="HKEY_LOCAL_MACHINE", key="SOFTWARE\\Microsoft\\Windows\\CurrentVersion" ) WHERE FullPath =~ "\\\\\\s+$"`
2. Recherche de clés avec espaces ou caractères spéciaux
3. Souvent utilisées pour masquer des configurations malveillantes

4. Difficiles à repérer dans les outils standard

5. **WOW64 Redirection** : `vql SELECT * FROM registry( root="HKEY_LOCAL_MACHINE", key="SOFTWARE\\Wow6432Node" )`

6. Registre spécifique aux applications 32 bits sur systèmes 64 bits
7. Souvent négligé dans les analyses

8. Utilisé par certains malwares pour la persistance

9. **Valeurs de grande taille** : `vql SELECT FullPath, Size(item=Data) AS DataSize FROM registry( root="HKEY_LOCAL_MACHINE", key="SOFTWARE" ) WHERE DataSize > 10000`

10. Détection de valeurs anormalement grandes
11. Potentiellement utilisées pour stocker des charges utiles
12. Indicateur de techniques de stockage alternatives

Bonnes pratiques pour l'analyse du registre

Pour maximiser l'efficacité des investigations du registre :

1. **Approche systématique** :
2. Commencez par les artefacts les plus courants (Run, Services)
3. Élargissez progressivement vers des zones plus spécifiques
4. Documentez toutes les découvertes avec leur contexte
5. **Contextualisation** :
6. Corrélation avec d'autres artefacts (fichiers, logs)
7. Vérification des valeurs suspectes (hachage des exécutables référencés)
8. Comparaison avec des systèmes sains similaires
9. **Extraction et préservation** :
10. Exportez les ruches complètes pour analyse hors ligne
11. Préservez les journaux de transaction
12. Documentez la méthode d'acquisition
13. **Analyse à grande échelle** :
14. Utilisez des chasses pour identifier des anomalies sur plusieurs systèmes
15. Créez des artefacts personnalisés pour des menaces spécifiques
16. Automatisez l'analyse des indicateurs courants

Collecte des journaux d'événements

Les journaux d'événements constituent une source cruciale d'informations pour les investigations de sécurité et l'analyse forensique. Velociraptor offre des capacités puissantes pour collecter et analyser ces journaux sur différentes plateformes.

Journaux d'événements Windows (EVT/EVTX)

Windows stocke ses journaux d'événements dans des fichiers au format EVTX (ou EVT pour les versions plus anciennes), qui contiennent des informations détaillées sur l'activité du système.

Collecte des journaux Windows

Velociraptor propose plusieurs méthodes pour collecter ces journaux :

1. **Collecte directe des fichiers EVTX** : `vql SELECT * FROM glob( globs="C:/Windows/System32/winevt/Logs/*.evtx", accessor="ntfs" )`
2. Accès direct aux fichiers de journaux
3. Contourne les verrous du système
4. Permet de collecter tous les journaux d'un coup
5. **Utilisation de l'artefact Windows.EventLogs.EvtxHunter** : `vql SELECT * FROM Artifact.Windows.EventLogs.EvtxHunter( EventLogNames=["Security", "System", "Application"], DateAfter="2023-01-01" )`
6. Interface simplifiée pour la collecte ciblée
7. Filtrage par date et type de journal
8. Optimisé pour les investigations courantes
9. **Collecte via WMI** : `vql SELECT * FROM wmi( query="SELECT * FROM Win32_NTLogEvent WHERE LogFile='Security' AND TimeGenerated > '20230101000000.000000-000'" )`
10. Utilise l'API Windows native
11. Peut être plus lent mais offre un filtrage côté serveur
12. Utile pour des requêtes très spécifiques

Analyse des journaux Windows

Une fois collectés, Velociraptor offre des outils puissants pour analyser ces journaux :

1. **Analyse basique avec parse_evtx** : `vql SELECT * FROM parse_evtx( filename="C:/Windows/System32/winevt/Logs/Security.evtx" ) WHERE EventID = 4624`
2. Analyse le contenu des fichiers EVTX
3. Permet de filtrer par ID d'événement, source, etc.
4. Fonctionne sur les fichiers collectés ou en direct
5. **Recherche avancée avec EvtxHunter** : `vql SELECT * FROM Artifact.Windows.EventLogs.EvtxHunter( SearchRegex="mimikatz|lsass", EventIDRegex="10|4688", DateAfter="2023-01-01" )`

6. Recherche par expression régulière dans le contenu
7. Combinaison de multiples critères de filtrage
8. Optimisé pour les performances
9. **Utilisation de règles Sigma** : `vql SELECT * FROM sigma( file="C:/Windows/System32/winevt/Logs/Security.evtx", rule="rules/windows/process_creation/proc_creation_win_mimikatz.yml" )`
10. Applique des règles de détection standardisées
11. Bénéficie de la bibliothèque communautaire Sigma
12. Idéal pour la détection de menaces connues

Journaux d'événements critiques Windows

Certains journaux Windows sont particulièrement importants pour les investigations de sécurité :

1. **Journal de sécurité** :
2. EventID 4624/4625 : Connexions réussies/échouées
3. EventID 4688 : Création de processus
4. EventID 4720/4724 : Création/modification de compte
5. EventID 4648 : Authentification explicite (RunAs)
6. EventID 4672 : Attribution de privilèges spéciaux
7. **Journal système** :
8. EventID 7045/7040 : Installation/modification de service
9. EventID 104 : Effacement du journal d'événements
10. EventID 1102 : Effacement du journal de sécurité
11. EventID 6005/6006 : Démarrage/arrêt du service d'événements
12. **Journaux d'applications et services** :
13. PowerShell (ID 4103, 4104) : Exécution de scripts
14. Sysmon (si installé) : Surveillance avancée du système
15. AppLocker : Application des politiques d'exécution
16. Windows Defender : Détections de menaces

Journaux système Linux (Syslog)

Sur les systèmes Linux, les journaux système (syslog) et journald (pour les systèmes utilisant systemd) sont les principales sources d'informations d'événements.

Collecte des journaux Linux

Velociraptor offre plusieurs méthodes pour collecter ces journaux :

1. **Collecte directe des fichiers de journaux** : `vql SELECT * FROM glob( globs=["/var/log/syslog*", "/var/log/auth.log*", "/var/log/secure*"] )`
2. Accès direct aux fichiers de journaux
3. Inclut les fichiers compressés (.gz)
4. Approche simple et efficace
5. **Utilisation de l'artefact Linux.Syslog.SearchLog** : `vql SELECT * FROM Artifact.Linux.Syslog.SearchLog( LogFiles=["/var/log/auth.log", "/var/log/secure"], SearchRegex="Failed password|Invalid user" )`
6. Interface simplifiée pour la recherche dans les journaux
7. Filtrage par expression régulière
8. Optimisé pour les investigations courantes
9. **Collecte via journalctl** : `vql SELECT * FROM process_execution( argv=["journalctl", "-o", "json", "--since", "2023-01-01"], output="STDOUT" )`
10. Utilise l'outil natif journalctl
11. Accès aux journaux systemd
12. Permet un filtrage avancé côté système

Analyse des journaux Linux

Une fois collectés, Velociraptor offre des outils pour analyser ces journaux :

1. **Analyse basique avec parse_lines** : `vql SELECT * FROM parse_lines( filename="/var/log/auth.log" ) WHERE Line =~ "Failed password"`
2. Analyse ligne par ligne des fichiers de journaux
3. Permet de filtrer par contenu
4. Simple mais efficace pour de nombreux cas

5. **Analyse structurée avec parse_syslog** : `vql SELECT Timestamp, Program, Message FROM parse_syslog( filename="/var/log/syslog" ) WHERE Program = "sshd"`

6. Parse le format syslog standard
7. Extrait les champs structurés (horodatage, programme, etc.)

8. Facilite l'analyse et le filtrage

9. **Analyse JSON avec parse_json_array** : `vql SELECT * FROM foreach( row={ SELECT Stdout FROM process_execution( argv=["journalctl", "-o", "json", "--since", "2023-01-01"] ) }, query={ SELECT * FROM parse_json_array(data=Stdout) WHERE _SYSTEMD_UNIT = "sshd.service" } )`

10. Traite la sortie JSON de journalctl
11. Permet un filtrage avancé sur les champs structurés
12. Idéal pour les systèmes utilisant systemd

Journaux critiques Linux

Certains journaux Linux sont particulièrement importants pour les investigations de sécurité :

1. **Journaux d'authentification** :
2. /var/log/auth.log (Debian/Ubuntu)
3. /var/log/secure (RHEL/CentOS)
4. Contiennent les tentatives de connexion, utilisations de sudo, etc.

5. Essentiels pour détecter les tentatives d'accès non autorisés

6. **Journaux système** :

7. /var/log/syslog (Debian/Ubuntu)
8. /var/log/messages (RHEL/CentOS)
9. Contiennent les messages système généraux

10. Utiles pour identifier les changements de configuration

11. **Journaux d'audit** :

12. /var/log/audit/audit.log
13. Contiennent des enregistrements détaillés des événements système
14. Particulièrement utiles lorsque auditd est configuré

Journaux macOS

macOS utilise un système de journalisation unique appelé Unified Logging System, accessible via l'outil `log`.

Collecte des journaux macOS

Velociraptor offre plusieurs méthodes pour collecter ces journaux :

1. **Collecte via l'outil log** : `vql SELECT * FROM process_execution( argv=["log", "show", "--predicate", "eventMessage contains 'sudo'", "--last", "24h", "--style", "json"], output="STDOUT" )`
2. Utilise l'outil natif log
3. Permet un filtrage avancé

4. Accès à l'Unified Logging System

5. **Collecte des fichiers de journaux traditionnels** : `vql SELECT * FROM glob( globs=["/var/log/system.log*", "/var/log/install.log*"] )`

6. Accès aux journaux traditionnels (toujours présents)
7. Complémentaire à l'Unified Logging System
8. Utile pour la compatibilité avec les analyses standard

Analyse des journaux macOS

Une fois collectés, Velociraptor offre des outils pour analyser ces journaux :

1. **Analyse JSON des journaux unifiés** : `vql SELECT * FROM foreach( row={ SELECT Stdout FROM process_execution( argv=["log", "show", "--last", "24h", "--style", "json"] ) }, query={ SELECT * FROM parse_json_array(data=Stdout) WHERE eventMessage =~ "DENIED" } )`
2. Traite la sortie JSON de l'outil log
3. Permet un filtrage avancé sur les champs structurés

4. Accès complet aux données de journalisation

5. **Analyse des journaux traditionnels** : `vql SELECT * FROM parse_lines( filename="/var/log/system.log" ) WHERE Line =~ "error|failure|denied"`

6. Analyse ligne par ligne des fichiers de journaux traditionnels
7. Approche simple mais efficace
8. Complémentaire à l'analyse des journaux unifiés

Journaux d'applications et services

Au-delà des journaux système, de nombreuses applications et services maintiennent leurs propres journaux qui peuvent être précieux pour les investigations.

Serveurs web

Pour les serveurs web comme Apache, Nginx, IIS :

1. **Apache/Nginx** : `vql SELECT * FROM parse_lines( filename="/var/log/apache2/access.log" ) WHERE Line =~ "\\.php\\?id=['\\\"]|/etc/passwd|eval\\("`
2. Recherche de tentatives d'exploitation web
3. Détection d'injections SQL, LFI, etc.
4. Crucial pour les investigations de compromission web
5. **IIS** : `vql SELECT * FROM parse_csv_with_regex(filename="C:/inetpub/logs/LogFiles/W3SVC1/u_ex*.log", regex="^(?P<date>\\S+) (?P<time>\\S+) (?P<s_ip>\\S+) (?P<cs_method>\\S+) (?P<cs_uri_stem>\\S+) (?P<cs_uri_query>\\S+) (?P<s_port>\\d+) (?P<cs_username>\\S+) (?P<c_ip>\\S+) (?P<cs_user_agent>.+) (?P<sc_status>\\d+) (?P<sc_substatus>\\d+) (?P<sc_win32_status>\\d+) (?P<time_taken>\\d+)$") WHERE cs_uri_query =~ "exec|..\\/|xp_cmdshell"`
6. Analyse des logs IIS avec format personnalisé
7. Détection de tentatives d'exploitation
8. Reconstruction des requêtes suspectes

Bases de données

Pour les bases de données comme MySQL, PostgreSQL, MSSQL :

1. **MySQL** : `vql SELECT * FROM parse_lines( filename="/var/log/mysql/mysql.log" ) WHERE Line =~ "Access denied|failed|error"`
2. Détection de tentatives d'accès non autorisés
3. Identification d'erreurs potentiellement exploitables

4. Suivi des modifications de schéma

5. **PostgreSQL** : `vql SELECT * FROM parse_lines( filename="/var/log/postgresql/postgresql-*.log" ) WHERE Line =~ "authentication failed|FATAL|ERROR"`

6. Analyse des erreurs d'authentification
7. Détection d'activités suspectes
8. Suivi des modifications de configuration

Bonnes pratiques pour la collecte de journaux

Pour maximiser l'efficacité de la collecte et de l'analyse des journaux :

1. **Priorisation** :
2. Commencez par les journaux les plus pertinents pour l'investigation
3. Ciblez des périodes temporelles spécifiques

4. Utilisez des filtres pour réduire le volume de données

5. **Préservation de contexte** :

6. Collectez les métadonnées des fichiers de journaux (dates, permissions)
7. Préservez l'intégrité des journaux originaux

8. Documentez la méthode de collecte

9. **Corrélation** :

10. Combinez les journaux de différentes sources
11. Établissez des chronologies unifiées

12. Recherchez des modèles cohérents entre différents journaux

13. **Automatisation** :

14. Créez des artefacts personnalisés pour des scénarios récurrents
15. Utilisez des règles de détection standardisées (Sigma)
16. Implémentez des analyses périodiques pour la détection proactive

Analyse des artefacts système

L'analyse des artefacts système implique l'examen de divers composants et fichiers du système d'exploitation qui peuvent révéler des informations cruciales lors d'une

investigation. Velociraptor offre de nombreuses capacités pour collecter et analyser ces artefacts.

Artefacts de démarrage et persistance

Les mécanismes de démarrage et de persistance sont souvent ciblés par les attaquants pour maintenir leur accès aux systèmes compromis.

Windows

Sur les systèmes Windows, plusieurs emplacements permettent l'exécution automatique de programmes :

1. **Dossiers de démarrage** : `vql SELECT * FROM glob( globs=[ "C:/Users/*/AppData/Roaming/Microsoft/Windows/Start Menu/Programs/Startup/*", "C:/ProgramData/Microsoft/Windows/Start Menu/Programs/Startup/*" ] )`
2. Fichiers et raccourcis dans les dossiers de démarrage
3. Accessibles aux utilisateurs sans privilèges élevés
4. Souvent utilisés pour la persistance de base
5. **Services Windows** : `vql SELECT * FROM Artifact.Windows.System.Services() WHERE StartMode = "Auto" AND Status = "Running"`
6. Services configurés pour démarrer automatiquement
7. Nécessitent généralement des privilèges élevés
8. Mécanisme de persistance privilégié pour les menaces avancées
9. **Tâches planifiées** : `vql SELECT * FROM Artifact.Windows.System.TaskScheduler()`
10. Tâches configurées pour s'exécuter à des moments spécifiques
11. Peuvent être configurées avec différents niveaux de privilèges
12. Offrent des options de déclenchement flexibles (démarrage, connexion, etc.)
13. **WMI Event Subscriptions** : `vql SELECT * FROM wmi( query="SELECT * FROM __FilterToConsumerBinding" )`
14. Mécanisme de persistance avancé

15. Difficile à détecter avec les outils standard
16. Utilisé par les menaces sophistiquées

Linux

Sur les systèmes Linux, plusieurs mécanismes permettent l'exécution automatique :

1. **Systemd Units** : `vql SELECT * FROM glob( globs=[ "/etc/systemd/system/*.service", "/usr/lib/systemd/system/*.service", "/home/*/.config/systemd/user/*.service" ] )`
2. Services systemd configurés pour démarrer automatiquement
3. Mécanisme principal sur les distributions modernes
4. Présents au niveau système et utilisateur
5. **Crontabs** : `vql SELECT * FROM glob( globs=["/etc/crontab", "/var/spool/cron/*", "/etc/cron.d/*"] )`
6. Tâches planifiées à intervalles réguliers
7. Peuvent être configurées au niveau système ou utilisateur
8. Format standardisé facilitant l'analyse
9. **Scripts de démarrage** : `vql SELECT * FROM glob( globs=[ "/etc/init.d/*", "/etc/rc*.d/*", "/etc/profile.d/*", "/home/*/.bashrc", "/home/*/.profile" ] )`
10. Scripts exécutés au démarrage du système ou à la connexion
11. Varient selon la distribution et la configuration
12. Incluent les profils de shell utilisateur

macOS

Sur macOS, plusieurs mécanismes spécifiques existent :

1. **Launch Agents/Daemons** : `vql SELECT * FROM glob( globs=[ "/Library/LaunchAgents/*.plist", "/Library/LaunchDaemons/*.plist", "/System/Library/LaunchAgents/*.plist", "/System/Library/LaunchDaemons/*.plist", "/Users/*/Library/LaunchAgents/*.plist" ] )`
2. Mécanisme principal de persistance sur macOS
3. Agents (contexte utilisateur) vs Daemons (contexte système)
4. Fichiers plist contenant la configuration

5. **Login Items** : `vql SELECT * FROM darwin_login_items()`
6. Applications lancées à la connexion utilisateur
7. Configurables via les préférences système
8. Visibles pour l'utilisateur
9. **Crontabs et scripts de démarrage** : `vql SELECT * FROM glob( globs=[ "/etc/crontab", "/usr/lib/cron/tabs/*", "/etc/periodic/*/*", "/Users/*/.bash_profile" ] )`
10. Similaires à Linux
11. Incluent des mécanismes spécifiques à macOS comme periodic

Artefacts de configuration système

Les fichiers de configuration système peuvent révéler des modifications malveillantes ou des informations sur l'état du système.

Fichiers hosts et DNS

Les modifications des configurations DNS peuvent indiquer des tentatives de redirection malveillante :

1. **Fichier hosts** : `vql SELECT * FROM Artifact.Generic.Network.Hosts()`
2. Mappages statiques entre noms d'hôtes et adresses IP
3. Souvent modifiés pour rediriger le trafic
4. Présent sur tous les systèmes d'exploitation
5. **Configuration DNS** : `vql SELECT * FROM Artifact.Windows.Network.DNSCache()`
6. Cache DNS actuel
7. Révèle les domaines récemment consultés
8. Peut indiquer des communications avec des domaines malveillants

Configuration de pare-feu

Les modifications de pare-feu peuvent indiquer des tentatives d'établir des communications persistantes :

1. **Windows Firewall** : `vql SELECT * FROM wmi( query="SELECT * FROM FirewallRule" )`
2. Règles de pare-feu Windows

3. Peut révéler des règles autorisant des communications suspectes
4. Important pour identifier les canaux de commande et contrôle
5. **IPTables (Linux)** : `vql SELECT * FROM process_execution( argv=["iptables", "-L", "-n", "-v"], output="STDOUT" )`
6. Règles de pare-feu Linux
7. Configuration actuelle du filtrage réseau
8. Peut révéler des règles suspectes

Utilisateurs et groupes

Les modifications des comptes peuvent indiquer des tentatives d'établir des accès persistants :

1. **Comptes Windows** : `vql SELECT * FROM Artifact.Windows.System.LocalUsers()`
2. Comptes utilisateurs locaux
3. Peut révéler des comptes créés par des attaquants
4. Inclut des informations sur les privilèges et l'état des comptes
5. **Comptes Linux/macOS** : `vql SELECT * FROM parse_lines( filename="/etc/passwd" )`
6. Comptes utilisateurs sur systèmes Unix
7. Peut révéler des comptes suspects ou des modifications
8. À combiner avec l'analyse de /etc/shadow pour les informations d'authentification

Artefacts de navigation web

Les artefacts de navigation web peuvent révéler des activités suspectes ou des compromissions :

Historique et téléchargements

Pour tracer l'activité de navigation :

1. **Chrome** : `vql SELECT * FROM Artifact.Windows.Forensics.ChromeHistory()`
2. Historique de navigation Chrome
3. Téléchargements, cookies, favoris

4. Stocké dans des bases SQLite

5. **Firefox** : `vql SELECT * FROM Artifact.Windows.Forensics.FirefoxHistory()`

6. Historique de navigation Firefox
7. Structure similaire à Chrome mais schéma différent

8. Également stocké dans des bases SQLite

9. **Safari (macOS)** : `vql SELECT * FROM glob( globs="/Users/*/Library/Safari/History.db" )`

10. Historique de navigation Safari
11. Spécifique à macOS
12. Format de base de données propriétaire

Extensions et modules complémentaires

Les extensions de navigateur peuvent être utilisées à des fins malveillantes :

1. **Extensions Chrome** : `vql SELECT * FROM glob( globs="C:/Users/*/AppData/Local/Google/Chrome/User Data/*/Extensions/*" )`
2. Extensions installées dans Chrome
3. Peuvent avoir des privilèges étendus

4. Parfois utilisées comme persistance ou pour le vol d'informations

5. **Extensions Firefox** : `vql SELECT * FROM glob( globs="C:/Users/*/AppData/Roaming/Mozilla/Firefox/Profiles/*/extensions/*" )`

6. Extensions installées dans Firefox
7. Structure différente de Chrome
8. Également potentiellement malveillantes

Artefacts de connexion et d'authentification

Les artefacts liés aux connexions peuvent révéler des accès non autorisés :

Historique de connexion

Pour tracer les connexions au système :

1. **Windows Logon Events** : `vql SELECT * FROM Artifact.Windows.EventLogs.EvtxHunter( EventIDRegex="4624|4625", DateAfter="2023-01-01" )`
2. Événements de connexion réussie (4624) ou échouée (4625)
3. Inclut des informations détaillées sur la source et le type
4. Essentiel pour détecter les accès non autorisés
5. **Linux Auth Logs** : `vql SELECT * FROM Artifact.Linux.Syslog.SearchLog( LogFiles=["/var/log/auth.log"], SearchRegex="session opened|session closed|Failed password" )`
6. Journaux d'authentification Linux
7. Trace les connexions réussies et échouées
8. Inclut les utilisations de sudo

Accès à distance

Pour identifier les connexions à distance :

1. **RDP (Windows)** : `vql SELECT * FROM Artifact.Windows.EventLogs.RDPAuth()`
2. Connexions via Remote Desktop Protocol
3. Inclut les tentatives réussies et échouées
4. Crucial pour détecter les accès à distance non autorisés
5. **SSH (Linux/macOS)** : `vql SELECT * FROM Artifact.Linux.Syslog.SSHLogin()`
6. Connexions via Secure Shell
7. Trace les tentatives d'authentification
8. Peut révéler des tentatives de force brute

Bonnes pratiques pour l'analyse des artefacts système

Pour maximiser l'efficacité de l'analyse des artefacts système :

1. **Approche systématique** :
2. Commencez par les artefacts les plus pertinents pour l'investigation
3. Établissez une chronologie des événements

4. Corrélation entre différents types d'artefacts

5. **Contextualisation** :

6. Comparez avec des systèmes sains similaires
7. Établissez une ligne de base pour identifier les anomalies

8. Tenez compte de l'environnement et des politiques de l'organisation

9. **Automatisation** :

10. Créez des artefacts personnalisés pour des scénarios récurrents
11. Utilisez des chasses pour identifier des anomalies à grande échelle

12. Implémentez des analyses périodiques pour la détection proactive

13. **Documentation** :

14. Documentez toutes les découvertes avec leur contexte
15. Préservez les artefacts originaux
16. Maintenez une chaîne de possession claire

8. Chasses (Hunts)

Concept et utilité des chasses

Les chasses (hunts) constituent l'une des fonctionnalités les plus puissantes de Velociraptor, permettant d'exécuter des collectes et des analyses à grande échelle sur de nombreux systèmes simultanément. Cette section explore le concept des chasses et leur utilité dans différents contextes.

Définition et principes fondamentaux

Une chasse dans Velociraptor est une opération qui consiste à exécuter un ou plusieurs artefacts sur un ensemble de clients sélectionnés selon des critères spécifiques. Contrairement aux collectes individuelles qui ciblent un seul système, les chasses permettent d'appliquer la même investigation à travers tout un parc informatique.

Les principes fondamentaux des chasses incluent :

1. **Scalabilité** : Les chasses sont conçues pour s'exécuter efficacement sur des centaines ou milliers de systèmes.

2. **Ciblage précis** : Les clients peuvent être sélectionnés selon divers critères (étiquettes, noms, expressions régulières).
3. **Contrôle des ressources** : Des limites peuvent être définies pour minimiser l'impact sur les systèmes et le réseau.
4. **Agrégation des résultats** : Les données collectées sont centralisées et peuvent être analysées de manière unifiée.
5. **Parallélisation** : Les chasses s'exécutent en parallèle sur les clients, optimisant le temps total d'exécution.

Cas d'usage des chasses

Les chasses sont particulièrement utiles dans plusieurs scénarios :

Réponse aux incidents

Lors d'un incident de sécurité, les chasses permettent de :

1. **Rechercher des indicateurs de compromission** :
2. Déployer rapidement des recherches d'IOCs sur tous les systèmes
3. Identifier l'étendue d'une compromission
4. Détecter des systèmes potentiellement affectés mais non encore identifiés
5. **Collecter des preuves volatiles** :
6. Capturer simultanément des données volatiles sur de nombreux systèmes
7. Préserver des preuves avant qu'elles ne soient altérées
8. Établir une chronologie cohérente à travers l'infrastructure
9. **Appliquer des mesures correctives** :
10. Exécuter des scripts de remédiation sur les systèmes affectés
11. Isoler des systèmes compromis
12. Supprimer des fichiers malveillants identifiés

Chasse aux menaces (Threat Hunting)

Pour la recherche proactive de menaces :

1. **Détection de comportements suspects** :
2. Rechercher des modèles d'activité anormaux
3. Identifier des configurations suspectes
4. Détecter des techniques d'attaque connues (basées sur MITRE ATT&CK, etc.)

5. **Établissement de lignes de base** :

6. Documenter l'état normal des systèmes
7. Identifier les variations par rapport à ces lignes de base

8. Détecter les anomalies statistiques

9. **Validation d'hypothèses** :

10. Tester des théories sur des techniques d'attaque potentielles
11. Vérifier si certaines vulnérabilités sont exploitées
12. Confirmer ou infirmer des suspicions de sécurité

Conformité et audit

Pour les besoins de conformité et d'audit :

1. **Vérification de politiques** :
2. Contrôler l'application des politiques de sécurité
3. Identifier les systèmes non conformes

4. Documenter l'état de conformité pour les audits

5. **Inventaire des actifs** :

6. Recenser les logiciels installés
7. Identifier les versions obsolètes ou non supportées

8. Détecter les logiciels non autorisés

9. **Validation de correctifs** :

10. Vérifier l'application des correctifs de sécurité
11. Identifier les systèmes vulnérables
12. Prioriser les efforts de mise à jour

Avantages par rapport aux approches traditionnelles

Les chasses Velociraptor offrent plusieurs avantages par rapport aux méthodes traditionnelles d'investigation à grande échelle :

1. **Efficacité et rapidité** :
2. Exécution parallèle sur de nombreux systèmes
3. Résultats disponibles en temps réel

4. Réduction significative du temps d'investigation

5. **Flexibilité** :

6. Personnalisation complète des requêtes via VQL
7. Adaptation aux besoins spécifiques de chaque investigation

8. Possibilité de combiner plusieurs artefacts dans une même chasse

9. **Impact contrôlé** :

10. Limitation précise des ressources utilisées
11. Planification des exécutions pendant les périodes creuses

12. Réduction des perturbations sur les systèmes de production

13. **Centralisation** :

14. Collecte et analyse centralisées des données
15. Vue unifiée des résultats

16. Corrélation facilitée entre différents systèmes

17. **Reproductibilité** :

18. Documentation complète des requêtes exécutées
19. Possibilité de répéter les mêmes analyses dans le temps
20. Comparaison historique des résultats

Considérations éthiques et légales

L'utilisation des chasses soulève des considérations importantes :

1. **Respect de la vie privée** :
2. Les chasses peuvent potentiellement accéder à des données sensibles
3. Nécessité de limiter la collecte aux informations strictement nécessaires

4. Importance d'informer les utilisateurs des politiques de surveillance

5. **Cadre légal** :

6. Conformité avec les réglementations locales (RGPD en Europe, etc.)
7. Documentation des justifications pour chaque chasse

8. Conservation appropriée des preuves pour les besoins légaux

9. **Proportionnalité** :

10. Équilibre entre les besoins de sécurité et l'impact sur les systèmes

11. Limitation des chasses aux objectifs définis
12. Éviter les collectes excessives ou non justifiées

Création et configuration de chasses

La création et la configuration efficaces des chasses sont essentielles pour maximiser leur valeur tout en minimisant leur impact. Cette section détaille les étapes et les bonnes pratiques pour configurer des chasses dans Velociraptor.

Interface de création de chasses

Velociraptor offre une interface dédiée pour la création et la gestion des chasses :

1. **Accès à l'interface** :
2. Naviguez vers la section "Hunt Manager" dans le menu principal
3. Cliquez sur "New Hunt" pour démarrer la création d'une nouvelle chasse

4. L'assistant de création s'ouvre avec plusieurs étapes à compléter

5. **Structure de l'assistant** :

6. Sélection des artefacts à collecter
7. Configuration des paramètres pour chaque artefact
8. Définition des critères de ciblage des clients
9. Configuration des limites de ressources

10. Révision et lancement de la chasse

11. **Options avancées** :

12. Planification temporelle
13. Configuration des notifications
14. Options de traitement des résultats
15. Paramètres de durée et d'expiration

Sélection et configuration des artefacts

La première étape consiste à choisir les artefacts à exécuter :

1. **Recherche d'artefacts** :
2. Utilisez la barre de recherche pour trouver des artefacts pertinents
3. Filtrez par catégorie (Windows, Linux, MacOS, Generic)

4. Consultez les descriptions pour comprendre leur fonction

5. **Sélection multiple** :

6. Ajoutez plusieurs artefacts à une même chasse
7. Organisez-les dans un ordre logique

8. Combinez des artefacts complémentaires pour une investigation complète

9. **Configuration des paramètres** :

10. Ajustez les paramètres spécifiques à chaque artefact
11. Définissez des filtres pour cibler précisément les données recherchées

12. Utilisez des expressions régulières pour des recherches avancées

13. **Exemple de configuration** : ``` Artefact: Windows.Search.FileFinder Paramètres:

    - SearchFilesGlob: C:\Users*\Downloads*.exe
    - DateAfter: 2023-01-01
    - UploadFiles: Y
    - SizeMax: 10485760 # 10 MB ```

Ciblage des clients

Le ciblage permet de définir quels clients participeront à la chasse :

1. **Méthodes de ciblage** :
2. **Tous les clients** : La chasse s'exécute sur tous les clients connectés
3. **Par étiquette** : Cible les clients ayant des étiquettes spécifiques
4. **Par expression régulière** : Filtre les clients selon leur nom d'hôte ou autres attributs

5. **Par système d'exploitation** : Limite la chasse à des OS spécifiques

6. **Stratégies de ciblage efficaces** :

7. Commencez par un petit groupe de systèmes pour valider la chasse
8. Utilisez des étiquettes organisationnelles (département, fonction, etc.)

9. Combinez plusieurs critères pour un ciblage précis

10. **Exemples de ciblage** : ``` # Ciblage par expression régulière sur le nom d'hôte LAPTOP-.* # Tous les ordinateurs portables

Ciblage par étiquette finance,hr # Systèmes des départements finance et RH

Ciblage par système d'exploitation SELECT * FROM info() WHERE OS = 'windows' AND OSMajor >= 10 ```

Limites de ressources et planification

Pour minimiser l'impact sur les systèmes et le réseau :

1. **Limites de ressources client** :
2. **CPU Limit** : Pourcentage maximal d'utilisation CPU (ex: 25%)
3. **Max Execution Time** : Durée maximale d'exécution (ex: 30 minutes)
4. **Max Idle Time** : Temps d'inactivité avant abandon (ex: 10 minutes)
5. **Max Rows** : Nombre maximal de résultats à retourner
6. **Limites de ressources serveur** :
7. **Max Clients** : Nombre maximal de clients exécutant la chasse simultanément
8. **Ops Per Second** : Nombre d'opérations par seconde autorisées
9. **Progress Timeout** : Délai avant de considérer un client comme inactif
10. **Planification temporelle** :
11. **Exécution immédiate** : La chasse démarre dès sa création
12. **Exécution différée** : Planification à une date et heure spécifiques
13. **Exécution récurrente** : Configuration de répétitions périodiques
14. **Exemple de configuration** : ``` Limites client:
    - CPU Limit: 15%
    - Max Execution Time: 1800 # 30 minutes
    - Max Rows: 10000

Limites serveur: - Max Clients: 100 - Ops Per Second: 10 ```

Options avancées

Des options supplémentaires permettent d'affiner le comportement des chasses :

1. **Notifications** :
2. Configuration d'alertes par email à la fin de la chasse
3. Notifications en cas de détections spécifiques
4. Rapports automatiques des résultats

5. **Traitement des résultats** :

6. **Collecte complète** : Tous les résultats sont stockés
7. **Filtrage côté client** : Seuls les résultats pertinents sont envoyés

8. **Post-traitement** : Application d'analyses supplémentaires aux résultats

9. **Durée et expiration** :

10. Définition d'une date d'expiration pour la chasse
11. Configuration du comportement après expiration

12. Politique de conservation des résultats

13. **Priorité** :

14. Définition du niveau de priorité par rapport aux autres tâches
15. Impact sur l'ordonnancement des exécutions
16. Gestion des conflits entre chasses concurrentes

Bonnes pratiques pour la configuration

Pour optimiser l'efficacité et minimiser les risques :

1. **Validation préalable** :
2. Testez les artefacts sur un système individuel avant de lancer une chasse
3. Vérifiez les résultats et l'impact sur les performances

4. Ajustez les paramètres en fonction des observations

5. **Approche progressive** :

6. Commencez par un petit groupe de systèmes
7. Élargissez progressivement le périmètre

8. Surveillez les résultats et ajustez si nécessaire

9. **Documentation** :

10. Documentez clairement l'objectif de chaque chasse
11. Notez les paramètres utilisés et leur justification

12. Conservez un historique des chasses pour référence future

13. **Minimisation de l'impact** :

14. Planifiez les chasses intensives pendant les heures creuses

15. Utilisez des limites de ressources appropriées

16. Privilégiez les collectes ciblées plutôt que générales

17. **Sécurité et confidentialité** :

18. Limitez l'accès à la création de chasses aux utilisateurs autorisés
19. Évitez de collecter des données sensibles non nécessaires
20. Respectez les politiques de confidentialité de l'organisation

Exécution et surveillance des chasses

Une fois configurée, l'exécution et la surveillance efficaces d'une chasse sont essentielles pour garantir son succès et gérer tout problème potentiel. Cette section détaille les aspects pratiques de l'exécution des chasses et les méthodes pour les surveiller.

Lancement d'une chasse

Le processus de lancement d'une chasse comprend plusieurs étapes importantes :

1. **Vérification finale** :
2. Revue des artefacts sélectionnés et de leurs paramètres
3. Confirmation des critères de ciblage

4. Validation des limites de ressources

5. **Activation de la chasse** :

6. Cliquez sur "Launch Hunt" pour démarrer la chasse
7. La chasse est ajoutée à la file d'attente du serveur

8. Un identifiant unique est attribué à la chasse

9. **Phases d'exécution** :

10. **Initialisation** : Le serveur prépare la chasse
11. **Déploiement** : Les clients éligibles reçoivent les instructions
12. **Exécution** : Les clients exécutent les artefacts configurés
13. **Collecte** : Les résultats sont transmis au serveur

14. **Finalisation** : La chasse est marquée comme terminée

15. **États possibles d'une chasse** :

16. **Running** : La chasse est en cours d'exécution
17. **Paused** : La chasse a été temporairement suspendue
18. **Stopped** : La chasse a été arrêtée manuellement
19. **Completed** : La chasse s'est terminée normalement
20. **Error** : La chasse a rencontré des erreurs critiques

Interface de surveillance des chasses

Velociraptor offre une interface dédiée pour surveiller les chasses en cours et passées :

1. **Vue d'ensemble des chasses** :
2. Liste de toutes les chasses avec leur état
3. Statistiques de base (clients ciblés, complétés, avec erreurs)
4. Filtres pour afficher les chasses actives, terminées, etc.
5. **Vue détaillée d'une chasse** :
6. Accès en cliquant sur une chasse spécifique
7. Informations complètes sur la configuration
8. Statistiques détaillées d'exécution
9. Accès aux résultats collectés
10. **Tableau de bord de progression** :
11. Graphiques montrant l'avancement de la chasse
12. Répartition des états des clients
13. Chronologie d'exécution
14. Métriques de performance
15. **Journal d'activité** :
16. Événements importants liés à la chasse
17. Erreurs et avertissements
18. Actions administratives (pause, reprise, arrêt)

Métriques et statistiques clés

Plusieurs métriques permettent d'évaluer la progression et l'efficacité d'une chasse :

1. **Métriques de couverture** :
2. **Clients ciblés** : Nombre total de clients correspondant aux critères
3. **Clients contactés** : Clients qui ont reçu les instructions

4. **Clients complétés** : Clients ayant terminé l'exécution

5. **Taux de complétion** : Pourcentage de clients ayant terminé

6. **Métriques de performance** :

7. **Temps moyen d'exécution** : Durée moyenne par client
8. **Utilisation des ressources** : CPU, mémoire, réseau
9. **Taux de transfert** : Volume de données collectées par unité de temps

10. **Latence** : Délai entre l'exécution et la réception des résultats

11. **Métriques de résultats** :

12. **Nombre total de résultats** : Lignes de données collectées
13. **Taille des données** : Volume total des données collectées
14. **Taux de détection** : Pourcentage de clients avec résultats positifs

15. **Distribution des résultats** : Répartition par type, gravité, etc.

16. **Métriques d'erreurs** :

17. **Clients en erreur** : Nombre de clients ayant rencontré des erreurs
18. **Types d'erreurs** : Classification des problèmes rencontrés
19. **Taux d'échec** : Pourcentage de clients en erreur
20. **Impact des erreurs** : Évaluation de la fiabilité des résultats

Gestion des chasses en cours

Plusieurs actions peuvent être effectuées sur les chasses en cours d'exécution :

1. **Pause et reprise** :
2. Suspension temporaire de l'exécution
3. Utile pour réduire la charge pendant les heures de pointe

4. Reprise ultérieure sans perte de progression

5. **Modification des paramètres** :

6. Ajustement des limites de ressources
7. Modification du nombre maximal de clients simultanés

8. Adaptation en fonction des observations

9. **Arrêt anticipé** :

10. Arrêt complet de la chasse avant sa fin naturelle

11. Utile en cas d'impact imprévu ou de résultats suffisants

12. Les résultats déjà collectés restent disponibles

13. **Extension de la portée** :

14. Ajout de nouveaux clients à une chasse existante
15. Modification des critères de ciblage
16. Prolongation de la durée d'exécution

Résolution des problèmes courants

Plusieurs problèmes peuvent survenir lors de l'exécution des chasses :

1. **Clients non répondants** :
2. **Symptôme** : Clients ciblés mais sans progression
3. **Causes possibles** : Clients hors ligne, problèmes de communication

4. **Solutions** : Vérifier l'état des clients, leur connectivité, redéployer l'agent

5. **Erreurs d'exécution** :

6. **Symptôme** : Clients signalant des erreurs
7. **Causes possibles** : Permissions insuffisantes, ressources limitées, bugs

8. **Solutions** : Examiner les logs détaillés, ajuster les paramètres, corriger les artefacts

9. **Performance dégradée** :

10. **Symptôme** : Exécution anormalement lente
11. **Causes possibles** : Surcharge des systèmes, limites trop restrictives

12. **Solutions** : Ajuster les limites de ressources, réduire la portée, optimiser les artefacts

13. **Volume de données excessif** :

14. **Symptôme** : Collecte de données bien supérieure aux attentes
15. **Causes possibles** : Filtres trop larges, paramètres mal configurés
16. **Solutions** : Affiner les filtres, limiter les uploads, arrêter et reconfigurer la chasse

Bonnes pratiques pour la surveillance

Pour une surveillance efficace des chasses :

1. **Surveillance proactive** :

2. Vérifiez régulièrement l'état des chasses importantes
3. Configurez des alertes pour les anomalies

4. Intervenez rapidement en cas de problèmes

5. **Analyse progressive** :

6. Commencez à analyser les résultats dès qu'ils arrivent
7. Identifiez les tendances et anomalies précocement

8. Ajustez la chasse si nécessaire en fonction des premiers résultats

9. **Documentation des observations** :

10. Notez les comportements inhabituels
11. Documentez les problèmes rencontrés et leurs solutions

12. Créez une base de connaissances pour les futures chasses

13. **Évaluation post-exécution** :

14. Analysez les performances globales de la chasse
15. Identifiez les opportunités d'optimisation
16. Documentez les leçons apprises pour améliorer les futures chasses

Analyse des résultats de chasses

L'analyse efficace des résultats de chasses est cruciale pour extraire des informations pertinentes et prendre des décisions éclairées. Cette section détaille les méthodes et outils disponibles dans Velociraptor pour analyser les résultats des chasses.

Interface d'analyse des résultats

Velociraptor offre plusieurs vues pour explorer les résultats des chasses :

1. **Vue tabulaire** :
2. Affichage par défaut des résultats
3. Colonnes correspondant aux champs retournés par les artefacts
4. Options de tri, filtrage et pagination

5. Possibilité d'exporter les données (CSV, JSON)

6. **Vue détaillée** :

7. Affichage complet d'une ligne de résultat spécifique

8. Visualisation structurée des données complexes
9. Accès aux métadonnées associées (client, horodatage, etc.)

10. Liens vers les fichiers collectés

11. **Visualisations** :

12. Graphiques générés automatiquement selon le type de données
13. Chronologies pour les événements temporels
14. Diagrammes de distribution pour les analyses statistiques

15. Cartes pour les données géographiques

16. **Vue agrégée** :

17. Statistiques sur l'ensemble des résultats
18. Regroupements par client, type de résultat, etc.
19. Identification des tendances et anomalies
20. Vue d'ensemble rapide des découvertes importantes

Filtrage et recherche avancée

Pour naviguer efficacement dans de grands volumes de résultats :

1. **Filtres de base** :
2. Filtrage par colonne avec opérateurs de comparaison
3. Recherche textuelle dans l'ensemble des résultats
4. Filtres prédéfinis pour les cas d'usage courants

5. Combinaison de plusieurs filtres

6. **Requêtes VQL personnalisées** :

7. Utilisation de VQL pour des filtres complexes
8. Transformation et enrichissement des données
9. Corrélation entre différentes sources

10. Analyses statistiques avancées

11. **Exemple de requête de filtrage** : `vql SELECT * FROM source(artifact="Windows.Search.FileFinder") WHERE Size > 1000000 AND FullPath =~ "\\Users\\.*\\.exe$" AND ModificationTime > timestamp(string="2023-01-01")`

12. **Recherche par IOCs** :

13. Filtrage par hachages de fichiers connus
14. Recherche de noms de domaines ou IP suspects
15. Identification de chemins de fichiers malveillants
16. Détection de clés de registre compromises

Post-traitement des résultats

Les résultats bruts peuvent nécessiter un traitement supplémentaire pour maximiser leur valeur :

1. **Enrichissement des données** :
2. Ajout d'informations contextuelles (réputation, classification)
3. Corrélation avec des sources externes (threat intelligence)
4. Résolution de noms d'utilisateurs, chemins, etc.

5. Calcul de métriques dérivées

6. **Déduplication et normalisation** :

7. Élimination des résultats redondants
8. Standardisation des formats (dates, chemins, etc.)
9. Regroupement des variantes similaires

10. Simplification des structures complexes

11. **Exemple de post-traitement** : ```vql LET results = SELECT * FROM source(artifact="Windows.Search.FileFinder")

LET enriched = SELECT FullPath, Size, ModificationTime, hash(path=FullPath) AS Hash, lookup_hash(hash=Hash) AS Reputation FROM results

SELECT * FROM enriched WHERE Reputation.Score > 70 ```

1. **Extraction d'informations** :
2. Identification de patterns dans les résultats
3. Extraction de données structurées à partir de texte
4. Génération de statistiques et métriques
5. Création de résumés et synthèses

Corrélation entre chasses et sources

La corrélation des résultats de différentes sources permet une analyse plus complète :

1. **Corrélation temporelle** :
2. Alignement d'événements sur une chronologie

3. Identification de séquences d'actions
4. Détection de relations cause-effet

5. Reconstruction de la chronologie d'un incident

6. **Corrélation entre systèmes** :

7. Identification de comportements similaires sur différents systèmes
8. Traçage de la propagation d'une menace
9. Détection de mouvements latéraux

10. Comparaison entre systèmes compromis et sains

11. **Corrélation avec des sources externes** :

12. Intégration avec des plateformes SIEM
13. Enrichissement avec des données de threat intelligence
14. Comparaison avec des bases de connaissance de malware

15. Validation par des sources indépendantes

16. **Exemple de requête de corrélation** : ```vql LET process_creation = SELECT * FROM source(artifact="Windows.EventLogs.EvtxHunter", hunt_id="H.123") WHERE EventID = 4688

LET file_creation = SELECT * FROM source(artifact="Windows.Forensics.NTFS", hunt_id="H.124") WHERE Action = "FILE_CREATED"

SELECT p.Computer AS Computer, p.NewProcessName AS ProcessName, f.FullPath AS CreatedFile, p.EventTime AS ProcessTime, f.Timestamp AS FileTime FROM process_creation AS p JOIN file_creation AS f ON p.Computer = f.Computer WHERE f.Timestamp - p.EventTime < 60 ```

Exportation et partage des résultats

Pour faciliter la collaboration et l'intégration avec d'autres outils :

1. **Formats d'exportation** :
2. **CSV** : Format tabulaire simple pour les tableurs
3. **JSON** : Format structuré pour l'intégration avec d'autres outils
4. **HTML** : Rapports formatés pour la présentation

5. **ZIP** : Archive contenant les résultats et fichiers collectés

6. **Méthodes d'exportation** :

7. Exportation directe depuis l'interface web
8. Utilisation d'artefacts serveur dédiés
9. Exportation programmatique via l'API

10. Génération automatique de rapports

11. **Intégration avec d'autres plateformes** :

12. Envoi vers des systèmes SIEM (Splunk, ELK, etc.)
13. Intégration avec des plateformes de ticketing
14. Alimentation de bases de connaissances

15. Partage avec des équipes d'analyse externes

16. **Exemple d'exportation via VQL** : `vql SELECT * FROM source(artifact="Windows.Search.FileFinder", hunt_id="H.123") WHERE Size > 1000000 | format csv`

Création de rapports et visualisations

Pour communiquer efficacement les résultats :

1. **Notebooks Velociraptor** :
2. Environnements interactifs combinant texte, code et résultats
3. Possibilité d'exécuter des requêtes VQL directement
4. Création de visualisations personnalisées

5. Documentation du processus d'analyse

6. **Rapports automatisés** :

7. Génération de rapports basés sur des modèles
8. Inclusion de statistiques et métriques clés
9. Mise en évidence des résultats importants

10. Distribution automatique aux parties prenantes

11. **Visualisations avancées** :

12. Chronologies interactives
13. Graphes de relations
14. Cartes thermiques

15. Tableaux de bord personnalisés

16. **Exemple de création de notebook** : ```vql // Cellule de notebook pour analyser les résultats d'une chasse LET hunt_results = SELECT * FROM hunt_results(hunt_id="H.123")

// Statistiques générales SELECT count() AS TotalResults, count(item=ClientId, unique=TRUE) AS AffectedClients FROM hunt_results

// Visualisation des types de fichiers SELECT parse_pe(file=FullPath).CompanyName AS Company, count() AS FileCount FROM hunt_results GROUP BY Company | chart bar_chart(x=Company, y=FileCount) ```

Bonnes pratiques pour l'analyse des résultats

Pour maximiser la valeur des analyses :

1. **Approche méthodique** :
2. Commencez par une vue d'ensemble des résultats
3. Identifiez les tendances et anomalies principales
4. Approfondissez progressivement les points d'intérêt
5. Documentez le processus d'analyse et les découvertes
6. **Priorisation** :
7. Concentrez-vous d'abord sur les résultats à fort impact
8. Identifiez les systèmes critiques affectés
9. Évaluez la gravité et l'urgence des découvertes
10. Établissez un plan d'action basé sur les priorités
11. **Contextualisation** :
12. Considérez l'environnement et les spécificités de l'organisation
13. Tenez compte des faux positifs courants
14. Comparez avec des lignes de base connues
15. Évaluez l'impact business des découvertes
16. **Collaboration** :
17. Partagez les résultats avec les équipes concernées
18. Sollicitez l'expertise de spécialistes pour l'interprétation
19. Établissez un processus de révision des analyses
20. Documentez les conclusions pour référence future

9. Analyse avancée

Notebooks Velociraptor

Les notebooks Velociraptor constituent un outil puissant pour l'analyse interactive et la documentation des investigations. Ils permettent de combiner du texte explicatif, des requêtes VQL et leurs résultats dans un document cohérent et partageable.

Concept et utilité des notebooks

Les notebooks s'inspirent des environnements interactifs comme Jupyter, adaptés spécifiquement aux besoins de l'investigation numérique :

1. **Définition et principes** :
2. Documents interactifs combinant documentation et code exécutable
3. Organisation en cellules de différents types (texte, requêtes, résultats)
4. Exécution séquentielle ou sélective des cellules

5. Persistance des résultats pour référence future

6. **Avantages pour les investigations** :

7. **Documentation en temps réel** : Capture du raisonnement et des étapes d'analyse
8. **Reproductibilité** : Les requêtes peuvent être ré-exécutées pour vérification
9. **Collaboration** : Partage facile entre analystes et équipes

10. **Pédagogie** : Excellent outil pour la formation et le transfert de connaissances

11. **Cas d'usage typiques** :

12. Analyse approfondie d'incidents de sécurité
13. Documentation de procédures d'investigation
14. Création de rapports détaillés pour les parties prenantes
15. Développement et test de nouvelles requêtes VQL
16. Exploration interactive de données collectées

Interface et fonctionnalités des notebooks

L'interface des notebooks Velociraptor offre un environnement complet pour l'analyse :

1. **Accès et création** :
2. Section dédiée "Notebooks" dans l'interface web
3. Création d'un nouveau notebook via le bouton "New Notebook"

4. Options pour créer à partir de modèles ou notebooks existants

5. Association possible avec un client ou une chasse spécifique

6. **Structure de l'interface** :

7. Barre d'outils supérieure avec les actions principales
8. Zone d'édition centrale pour les cellules
9. Panneau latéral pour la navigation et les métadonnées

10. Barre d'état inférieure avec informations contextuelles

11. **Types de cellules** :

12. **Cellules Markdown** : Pour le texte formaté, les explications et la documentation
13. **Cellules VQL** : Pour les requêtes exécutables
14. **Cellules de résultats** : Affichage automatique des résultats des requêtes

15. **Cellules de visualisation** : Graphiques et représentations visuelles des données

16. **Fonctionnalités d'édition** :

17. Ajout, suppression et réorganisation des cellules
18. Formatage du texte avec Markdown (titres, listes, tableaux, etc.)
19. Coloration syntaxique pour le VQL
20. Auto-complétion et suggestions pour les requêtes

Création et utilisation de notebooks

La création et l'utilisation efficaces des notebooks impliquent plusieurs étapes :

1. **Planification du notebook** :
2. Définition claire de l'objectif d'analyse
3. Organisation logique des sections et étapes
4. Identification des données nécessaires

5. Structuration du flux d'investigation

6. **Rédaction de la documentation** :

7. Utilisation de cellules Markdown pour le contexte
8. Description claire de l'hypothèse d'investigation
9. Explication des requêtes et de leur objectif

10. Documentation des observations et conclusions

11. **Exemple de cellule Markdown** : ```markdown ## Analyse de persistance malveillante

Cette section examine les mécanismes de persistance potentiellement utilisés par l'attaquant. Nous allons analyser :

1. Les clés de registre Run et RunOnce
2. Les tâches planifiées suspectes
3. Les services nouvellement créés

Hypothèse : L'attaquant a utilisé plusieurs mécanismes de persistance pour maintenir l'accès. ```

1. **Création de requêtes VQL** :
2. Développement progressif des requêtes
3. Décomposition des analyses complexes en étapes
4. Utilisation de variables pour stocker les résultats intermédiaires

5. Commentaires explicatifs dans le code

6. **Exemple de cellule VQL** : ```vql // Recherche des clés Run et RunOnce suspectes LET registry_keys = SELECT * FROM registry(root="HKEY_LOCAL_MACHINE", key="SOFTWARE\Microsoft\Windows\CurrentVersion\Run")

// Filtrage des entrées suspectes SELECT FullPath, Data FROM registry_keys WHERE Data =~ "powershell|cmd.exe|regsvr32|rundll32" ```

1. **Exécution et itération** :
2. Exécution des cellules individuellement ou en séquence
3. Analyse des résultats intermédiaires
4. Ajustement des requêtes en fonction des observations
5. Ajout de nouvelles cellules pour approfondir l'analyse

Visualisations dans les notebooks

Les notebooks permettent de créer diverses visualisations pour faciliter l'analyse :

1. **Types de visualisations disponibles** :
2. **Tableaux** : Affichage structuré des données tabulaires
3. **Graphiques à barres** : Comparaison de valeurs entre catégories
4. **Graphiques linéaires** : Évolution de valeurs dans le temps
5. **Diagrammes circulaires** : Répartition proportionnelle
6. **Chronologies** : Séquence temporelle d'événements

7. **Graphes de relations** : Connexions entre entités
8. **Création de visualisations** : `vql // Analyse de la distribution des processus par utilisateur SELECT Username, count() AS ProcessCount FROM pslist() GROUP BY Username | chart bar_chart(x=Username, y=ProcessCount)`
9. **Personnalisation des visualisations** :
10. Ajustement des titres et légendes
11. Configuration des axes et échelles
12. Sélection des palettes de couleurs
13. Filtrage des données à visualiser
14. **Interactivité** :
15. Survol pour afficher des détails
16. Zoom et déplacement dans les graphiques
17. Filtrage dynamique des données
18. Basculement entre différentes vues

Partage et collaboration

Les notebooks facilitent la collaboration entre analystes :

1. **Options de partage** :
2. Partage au sein de l'instance Velociraptor
3. Exportation au format HTML ou PDF
4. Contrôle d'accès basé sur les rôles
5. Notifications de modifications
6. **Collaboration en temps réel** :
7. Visibilité des modifications par d'autres utilisateurs
8. Commentaires et annotations
9. Historique des versions
10. Attribution claire des contributions
11. **Intégration dans les flux de travail** :
12. Lien avec des tickets d'incident
13. Référencement dans la documentation d'investigation

14. Utilisation comme base pour les rapports formels

15. Archivage pour référence future

16. **Modèles et bonnes pratiques** :

17. Création de modèles standardisés pour différents types d'analyses
18. Bibliothèque de notebooks réutilisables
19. Documentation des méthodologies d'investigation
20. Formation des nouveaux analystes

Bonnes pratiques pour les notebooks

Pour maximiser la valeur des notebooks :

1. **Structure et organisation** :
2. Commencez par un résumé clair de l'objectif
3. Organisez le notebook en sections logiques
4. Utilisez des titres et sous-titres cohérents

5. Incluez une conclusion résumant les découvertes

6. **Documentation** :

7. Expliquez le contexte de l'investigation
8. Documentez les hypothèses et leur évolution
9. Commentez les requêtes complexes

10. Notez les limitations et incertitudes

11. **Requêtes efficaces** :

12. Optimisez les requêtes pour la performance
13. Utilisez des variables pour les résultats intermédiaires
14. Décomposez les analyses complexes

15. Incluez des exemples de résultats attendus

16. **Visualisations pertinentes** :

17. Choisissez le type de visualisation adapté aux données
18. Limitez les graphiques aux informations essentielles
19. Utilisez des titres et légendes explicites

20. Assurez-vous que les visualisations sont interprétables sans contexte supplémentaire

21. **Maintenance et révision** :

22. Mettez à jour les notebooks avec les nouvelles découvertes
23. Révisez périodiquement pour la pertinence
24. Versionnez les modifications importantes
25. Archivez de manière organisée pour référence future

Analyse temporelle et chronologies

L'analyse temporelle est fondamentale dans les investigations numériques, permettant de reconstruire la séquence des événements et de comprendre la chronologie d'un incident. Velociraptor offre des outils puissants pour créer et analyser des chronologies.

Importance des chronologies en investigation

Les chronologies jouent un rôle crucial dans la compréhension des incidents :

1. **Valeur investigative** :
2. Établissement de la séquence exacte des événements
3. Identification des relations cause-effet
4. Détection des anomalies temporelles

5. Reconstruction du déroulement d'une attaque

6. **Défis spécifiques** :

7. Synchronisation des horloges entre systèmes
8. Différences de fuseaux horaires
9. Formats de date/heure hétérogènes

10. Volume important d'événements à analyser

11. **Types de chronologies** :

12. **Macro-chronologies** : Vue d'ensemble sur une longue période
13. **Micro-chronologies** : Focus sur des périodes critiques
14. **Chronologies thématiques** : Centrées sur un type d'activité
15. **Chronologies multi-sources** : Combinant différentes origines de données

Sources de données temporelles

Velociraptor peut exploiter diverses sources pour construire des chronologies :

1. **Journaux d'événements** : `vql SELECT Timestamp, EventID, Channel, Computer, EventData FROM parse_evtx(filename="C:/Windows/System32/winevt/Logs/Security.evtx")`
2. Journaux Windows (Security, System, Application)
3. Journaux d'applications (IIS, Exchange, etc.)
4. Syslog sur systèmes Unix
5. **Système de fichiers** : `vql SELECT FullPath, ModTime AS Timestamp, Size, "File Modified" AS EventType FROM glob(globs="C:/Users/*/Documents/**")`
6. Dates de création/modification/accès aux fichiers
7. Journal USN (Windows)
8. Métadonnées MFT
9. **Artefacts système** : `vql SELECT LastRunTime AS Timestamp, Name, Command, "Task Execution" AS EventType FROM windows_scheduled_tasks()`
10. Préfetch (Windows)
11. Historique de navigation
12. Historique de commandes
13. Tâches planifiées
14. **Mémoire volatile** : `vql SELECT CreateTime AS Timestamp, Name, Pid, CommandLine, "Process Creation" AS EventType FROM pslist()`
15. Processus en cours d'exécution
16. Connexions réseau actives
17. Sessions utilisateur

Construction de chronologies unifiées

La création de chronologies cohérentes implique plusieurs étapes :

1. **Normalisation des horodatages** : `vql LET events = SELECT timestamp(epoch=Timestamp) AS StandardTime, Source, Description FROM raw_events`
2. Conversion vers un format standard
3. Ajustement des fuseaux horaires
4. Gestion des formats propriétaires
5. **Fusion de sources multiples** : ```vql LET file_events = SELECT ModTime AS Timestamp, FullPath AS Description, "File System" AS Source FROM glob(globs="C:/Users/**")

LET log_events = SELECT EventTime AS Timestamp, Message AS Description, Channel AS Source FROM parse_evtx(filename="Security.evtx")

SELECT * FROM chain(a=file_events, b=log_events) ORDER BY Timestamp ``` - Combinaison de différentes sources - Alignement temporel - Résolution des conflits

1. **Enrichissement contextuel** : `vql SELECT Timestamp, Description, Source, lookup_user(username=User) AS UserInfo, lookup_file(path=FilePath) AS FileInfo FROM timeline`
2. Ajout d'informations contextuelles
3. Résolution des identifiants
4. Corrélation avec des données externes
5. **Filtrage et priorisation** : `vql SELECT * FROM timeline WHERE Timestamp BETWEEN timestamp(string="2023-01-15T00:00:00Z") AND timestamp(string="2023-01-15T23:59:59Z") AND Importance > 3`
6. Concentration sur les périodes pertinentes
7. Filtrage des événements de routine
8. Mise en évidence des événements critiques

Visualisation des chronologies

Velociraptor offre plusieurs options pour visualiser les chronologies :

1. **Vue tabulaire** :
2. Affichage chronologique simple

3. Tri et filtrage flexibles
4. Export vers des formats externes

5. Idéal pour l'analyse détaillée

6. **Chronologies graphiques** : `vql SELECT * FROM timeline | timeline(timestamp=Timestamp, description=Description, category=Source)`

7. Représentation visuelle des événements
8. Regroupement par catégories
9. Zoom et navigation temporelle

10. Identification visuelle des clusters d'activité

11. **Graphiques de densité** : `vql SELECT timestamp(epoch=Timestamp, format="2006-01-02") AS Day, count() AS EventCount FROM timeline GROUP BY Day | chart bar_chart(x=Day, y=EventCount)`

12. Visualisation de la distribution temporelle
13. Identification des pics d'activité
14. Détection des anomalies statistiques

15. Vue d'ensemble des tendances

16. **Intégration avec des outils externes** :

17. Export vers des outils spécialisés (Timeline Explorer, etc.)
18. Génération de formats standardisés (CSV, JSON)
19. Visualisations personnalisées via notebooks
20. Intégration avec des plateformes SIEM

Analyse des anomalies temporelles

L'identification des anomalies temporelles est cruciale pour détecter les activités suspectes :

1. **Patterns temporels suspects** :
2. Activités en dehors des heures de travail
3. Séquences d'événements inhabituellement rapides
4. Gaps ou interruptions inexpliquées

5. Modifications de l'ordre chronologique normal

6. **Détection statistique** : ```vql LET baseline = SELECT hour(timestamp=Timestamp) AS Hour, count() AS NormalCount FROM historical_events GROUP BY Hour

LET current = SELECT hour(timestamp=Timestamp) AS Hour, count() AS CurrentCount FROM timeline GROUP BY Hour

SELECT b.Hour, b.NormalCount, c.CurrentCount, c.CurrentCount / b.NormalCount AS Ratio FROM baseline AS b JOIN current AS c ON b.Hour = c.Hour WHERE Ratio > 3 ``` - Comparaison avec des lignes de base historiques - Calcul d'écarts statistiques - Détection de fréquences anormales - Identification de corrélations inhabituelles

1. **Anti-forensics et manipulation temporelle** : `vql SELECT FullPath, ModTime AS ReportedTime, parse_mft(filename="C:/$MFT").StandardInfo.ModifiedTime AS MFTTime, ReportedTime - MFTTime AS TimeDifference FROM glob(globs="C:/Users/**/*.exe") WHERE abs(TimeDifference) > 3600`
2. Détection de modification d'horodatages
3. Incohérences entre différentes sources temporelles
4. Identification de techniques anti-forensiques
5. Validation croisée des métadonnées temporelles

Bonnes pratiques pour l'analyse temporelle

Pour maximiser l'efficacité de l'analyse temporelle :

1. **Préparation et planification** :
2. Identifiez les sources temporelles pertinentes
3. Documentez les fuseaux horaires et configurations
4. Établissez des lignes de base de comportement normal

5. Définissez clairement la portée temporelle de l'analyse

6. **Construction méthodique** :

7. Commencez par les événements clés confirmés
8. Élargissez progressivement la chronologie
9. Documentez les hypothèses et incertitudes

10. Validez les événements critiques via plusieurs sources

11. **Analyse contextuelle** :

12. Considérez le contexte organisationnel et technique

13. Tenez compte des activités légitimes planifiées
14. Corrélation avec des événements externes connus

15. Évaluation de la signification business des événements

16. **Documentation et présentation** :

17. Créez des représentations visuelles claires
18. Documentez la méthodologie et les sources
19. Mettez en évidence les événements critiques
20. Adaptez le niveau de détail à l'audience

Analyse de malware

L'analyse de malware est un aspect crucial des investigations de sécurité, permettant d'identifier et de comprendre les logiciels malveillants. Velociraptor offre plusieurs fonctionnalités pour faciliter cette analyse.

Détection initiale de malware

La première étape consiste à identifier la présence potentielle de malware :

1. **Recherche par hachage** : `vql SELECT FullPath, hash(path=FullPath) AS Hash, Size, ModTime FROM glob(globs="C:/Users/**/*.exe") WHERE Hash.MD5 IN hash_list`
2. Comparaison avec des bases de hachages connus
3. Détection rapide de malware identifié

4. Efficace pour les variantes connues

5. **Analyse YARA** : `vql SELECT * FROM yara( files="C:/Users/**/*.exe", rules=''' rule suspicious_characteristics { strings: $a = "CreateRemoteThread" ascii wide $b = "VirtualAllocEx" ascii wide $c = {68 ?? ?? ?? ?? 68 ?? ?? ?? ?? FF 55} condition: uint16(0) == 0x5A4D and 2 of them } ''' )`

6. Utilisation de règles YARA personnalisées
7. Détection basée sur des caractéristiques spécifiques

8. Identification de familles de malware

9. **Détection comportementale** : `vql SELECT * FROM Artifact.Windows.Detection.ProcessInjection()`

10. Utilisation d'artefacts spécialisés
11. Détection de comportements suspects
12. Identification de techniques d'attaque connues
13. **Analyse des anomalies** : `vql SELECT FullPath, Company, SignatureStatus FROM authenticode(filename=glob(globs="C:/Windows/System32/*.dll")) WHERE Company != "Microsoft Corporation" AND FullPath =~ "^C:/Windows/System32/"`
14. Identification de fichiers inhabituels
15. Détection de signatures invalides
16. Repérage d'emplacements suspects

Analyse statique de fichiers suspects

Une fois les fichiers suspects identifiés, l'analyse statique permet d'en comprendre les caractéristiques :

1. **Analyse des en-têtes PE** : `vql SELECT FullPath, parse_pe(file=FullPath).Sections AS Sections, parse_pe(file=FullPath).Imports AS Imports, parse_pe(file=FullPath).ExportedFunctions AS Exports FROM suspicious_files`
2. Examen de la structure du fichier exécutable
3. Identification des sections suspectes
4. Analyse des importations et exportations
5. **Extraction de chaînes** : `vql SELECT FullPath, strings(file=FullPath, min=6, context=10) AS ExtractedStrings FROM suspicious_files`
6. Extraction des chaînes de caractères
7. Identification d'URLs, adresses IP, commandes
8. Détection de techniques d'obfuscation
9. **Analyse de signature** : `vql SELECT FullPath, authenticode(filename=FullPath).SignatureStatus AS Status, authenticode(filename=FullPath).Signer AS Signer, authenticode(filename=FullPath).Timestamp AS SignTime FROM suspicious_files`
10. Vérification de la signature numérique

11. Identification de signatures falsifiées
12. Analyse des certificats utilisés
13. **Détection de packing et obfuscation** : `vql SELECT FullPath, parse_pe(file=FullPath).Sections AS Sections, entropy(string=read_file(filename=FullPath, length=1024*1024)) AS Entropy FROM suspicious_files WHERE Entropy > 7.5`
14. Calcul d'entropie pour détecter la compression/chiffrement
15. Identification de sections avec permissions inhabituelles
16. Détection de techniques anti-analyse

Analyse dynamique et comportementale

L'analyse dynamique observe le comportement du malware en exécution :

1. **Surveillance des processus** : `vql SELECT * FROM process_tracker( pid=4567, // PID du processus suspect duration=60 )`
2. Suivi des processus créés
3. Détection des injections de code
4. Observation des terminaisons de processus
5. **Surveillance du système de fichiers** : `vql SELECT * FROM file_tracker( root="C:/", duration=60 )`
6. Suivi des créations/modifications de fichiers
7. Détection des fichiers temporaires
8. Identification des modifications persistantes
9. **Surveillance du registre** : `vql SELECT * FROM registry_tracker( root="HKEY_LOCAL_MACHINE\\SOFTWARE", duration=60 )`
10. Suivi des modifications du registre
11. Détection des mécanismes de persistance
12. Identification des changements de configuration
13. **Surveillance réseau** : `vql SELECT * FROM netstat_tracker( duration=60 )`

14. Suivi des connexions réseau
15. Détection des communications C2
16. Identification des téléchargements secondaires

Analyse de mémoire pour la détection de malware

L'analyse de la mémoire est particulièrement efficace pour les malwares sophistiqués :

1. **Scan de mémoire avec YARA** : `vql SELECT * FROM process_yara( pid=4567, rules=''' rule injected_code { strings: $shellcode = {31 C0 50 68 2F 2F 73 68 68 2F 62 69 6E} condition: $shellcode } ''' )`
2. Recherche de signatures en mémoire
3. Détection de code injecté
4. Identification de shellcode
5. **Analyse des régions mémoire** : `vql SELECT Process, pid, BaseAddress, Size, Protection, Type FROM process_memory_map(pid=4567) WHERE Protection =~ "RWX"`
6. Identification des régions mémoire suspectes
7. Détection de permissions inhabituelles
8. Repérage de zones d'exécution dynamiques
9. **Extraction de code injecté** : `vql SELECT Process, pid, BaseAddress, upload( accessor="process", filename=format(format="/%d/%#x", args=[pid, BaseAddress]), length=Size ) AS Dump FROM suspicious_regions`
10. Extraction des régions mémoire pour analyse
11. Capture de code injecté
12. Préservation de charges utiles volatiles
13. **Détection de techniques d'évasion** : `vql SELECT Process, pid, CommandLine, Modules FROM pslist() WHERE len(Modules) < 5`
14. Identification de processus avec peu de modules
15. Détection de techniques de masquage
16. Repérage d'injections directes en mémoire

Intégration avec des services externes

Velociraptor peut s'intégrer avec des services d'analyse externes :

1. **VirusTotal** : `vql SELECT FullPath, hash(path=FullPath) AS Hash, virustotal(hash=Hash.SHA256) AS VTResults FROM suspicious_files`
2. Vérification des hachages contre VirusTotal
3. Obtention des détections multimoteurs
4. Accès aux analyses précédentes
5. **Sandbox automatisés** : `vql SELECT FullPath, upload_sandbox(file=FullPath, sandbox="cuckoo") AS SandboxResults FROM suspicious_files`
6. Soumission automatique à des environnements d'analyse
7. Récupération des rapports comportementaux
8. Analyse dans un environnement contrôlé
9. **OSINT et threat intelligence** : `vql SELECT Domain, lookup_domain(domain=Domain) AS IntelResults FROM extracted_iocs`
10. Enrichissement avec des données de renseignement
11. Vérification contre des listes de menaces connues
12. Corrélation avec des campagnes identifiées
13. **Plateformes d'analyse spécialisées** : `vql SELECT FullPath, upload_to_analysis(file=FullPath, service="hybrid-analysis") AS Results FROM suspicious_files`
14. Intégration avec des services d'analyse avancée
15. Obtention d'analyses comportementales détaillées
16. Accès à des capacités spécialisées

Extraction et analyse d'indicateurs de compromission (IOCs)

L'identification et l'extraction d'IOCs sont essentielles pour comprendre la portée d'une infection :

1. **Extraction d'URLs et domaines** : `vql SELECT FullPath, regex_extract( source=read_file(filename=FullPath),`

`regex="https?://([\\w\\.-]+)[:\\d]*/?\\S*" ) AS ExtractedURLs FROM suspicious_files`

2. Identification des points de communication
3. Détection des serveurs de commande et contrôle
4. Découverte de mécanismes de téléchargement
5. **Extraction d'adresses IP** : `vql SELECT FullPath, regex_extract( source=read_file(filename=FullPath), regex="\\b(?:[0-9]{1,3}\\.){3}[0-9]{1,3}\\b" ) AS ExtractedIPs FROM suspicious_files`
6. Identification des communications réseau codées en dur
7. Détection des points de contact directs
8. Découverte de mécanismes de contournement DNS
9. **Extraction de hachages et mutexes** : `vql SELECT FullPath, regex_extract( source=read_file(filename=FullPath), regex="CreateMutex\\w*\\([^,]*,\\s*[^,]*,\\s*\"([^\"]+)\"" ) AS ExtractedMutexes FROM suspicious_files`
10. Identification des mécanismes de persistance
11. Détection des techniques anti-analyse
12. Découverte d'identifiants uniques de malware
13. **Génération de règles de détection** : `vql SELECT format(format="rule detected_%v { strings: $s1 = \"%v\" ascii wide condition: any of them }", args=[regex_replace(source=FileName, re="[^\\w]", replace="_"), ExtractedString]) AS YaraRule FROM extracted_strings WHERE Significance > 0.8`
14. Création automatique de signatures
15. Génération de règles YARA
16. Facilitation de la détection à grande échelle

Bonnes pratiques pour l'analyse de malware

Pour une analyse de malware efficace et sécurisée :

1. **Sécurité et isolation** :
2. Utilisez des environnements isolés pour l'analyse dynamique
3. Évitez d'exécuter du code suspect sur des systèmes de production

4. Manipulez les échantillons avec précaution pour éviter la contamination

5. **Méthodologie structurée** :

6. Commencez par l'analyse statique avant l'exécution
7. Documentez chaque étape et observation
8. Établissez des hypothèses et testez-les systématiquement

9. Combinez plusieurs techniques d'analyse

10. **Contextualisation** :

11. Considérez le contexte de découverte du malware
12. Corrélation avec d'autres indicateurs d'incident
13. Évaluation de l'impact potentiel sur l'organisation

14. Identification des vecteurs d'infection initiaux

15. **Partage et collaboration** :

16. Documentez les IOCs pour partage
17. Contribuez aux plateformes de threat intelligence
18. Collaborez avec d'autres analystes
19. Maintenez une base de connaissances interne

Analyse réseau

L'analyse réseau est un aspect fondamental des investigations de sécurité, permettant d'identifier les communications suspectes et de comprendre les mouvements des attaquants. Velociraptor offre plusieurs capacités pour collecter et analyser les données réseau.

Collecte de données réseau

Velociraptor peut collecter diverses informations réseau à partir des terminaux :

1. **Connexions actives** : `vql SELECT * FROM netstat()`
2. Liste des connexions TCP/UDP en cours
3. Processus associés à chaque connexion
4. États des connexions TCP

5. Ports d'écoute

6. **Configuration réseau** : `vql SELECT * FROM interfaces()`

7. Interfaces réseau configurées
8. Adresses IP et MAC
9. Configuration DNS
10. Routes et passerelles
11. **Cache DNS** : `vql SELECT * FROM dns_cache()`
12. Entrées du cache DNS local
13. Résolutions récentes
14. TTL des entrées
15. Détection de résolutions suspectes
16. **Historique de connexions** : `vql SELECT * FROM windows_events( eventlog="Microsoft-Windows-Sysmon/Operational", eventid=3 )`
17. Événements de connexion réseau (via Sysmon)
18. Historique des communications
19. Métadonnées temporelles
20. Processus initiateurs

Analyse des connexions suspectes

L'identification des communications suspectes est essentielle pour détecter les compromissions :

1. **Détection de connexions inhabituelles** : `vql SELECT Process, Pid, RemoteAddr, RemotePort, State FROM netstat() WHERE RemotePort NOT IN (80, 443, 53) AND RemoteAddr !~ "^(10\\.|192\\.168\\.|172\\.(1[6-9]|2[0-9]|3[0-1]))"`
2. Identification des connexions vers des ports non standard
3. Filtrage des communications internes légitimes
4. Détection de canaux de commande et contrôle
5. **Analyse des processus avec connexions** : `vql SELECT n.Process, n.Pid, n.RemoteAddr, n.RemotePort, p.CommandLine, p.Username FROM netstat() AS n JOIN pslist() AS p ON n.Pid = p.Pid WHERE p.Name =~ "powershell|cmd|rundll32|regsvr32"`
6. Corrélation entre connexions et processus
7. Identification de processus légitimes avec comportement suspect

8. Détection de techniques Living-off-the-Land
9. **Détection de tunnels et proxys** : `vql SELECT Process, Pid, LocalAddr, LocalPort, RemoteAddr, RemotePort FROM netstat() WHERE LocalPort > 10000 AND State = "LISTEN" AND Process !~ "known_service"`
10. Identification de serveurs proxy non autorisés
11. Détection de tunnels de données
12. Repérage de redirections de port suspectes
13. **Analyse temporelle des connexions** : `vql SELECT timestamp(epoch=EventTime) AS Time, Process, SourceIp, DestinationIp, DestinationPort FROM sysmon_events WHERE EventID = 3 ORDER BY Time`
14. Établissement de chronologies de communication
15. Identification de patterns temporels suspects
16. Détection de communications périodiques (beaconing)

Analyse des artefacts réseau

Divers artefacts peuvent révéler des informations sur l'activité réseau passée :

1. **Historique de navigation** : `vql SELECT * FROM Artifact.Windows.Forensics.ChromeHistory()`
2. Sites web visités
3. Téléchargements effectués
4. Chronologie d'activité
5. Potentiels sites malveillants
6. **Fichiers hosts et configuration DNS** : `vql SELECT * FROM parse_hosts_file()`
7. Modifications du fichier hosts
8. Redirections DNS malveillantes
9. Blocage de sites de sécurité
10. Détournement de domaines légitimes
11. **Cookies et données de session** : `vql SELECT * FROM Artifact.Windows.Forensics.ChromeCookies()`

12. Traces de connexions à des services
13. Jetons d'authentification
14. Préférences et paramètres

15. Indicateurs de compromission persistants

16. **Journaux de pare-feu** : `vql SELECT * FROM parse_lines( filename="C:/Windows/System32/LogFiles/Firewall/pfirewall.log" )`

17. Connexions bloquées et autorisées
18. Tentatives de connexion échouées
19. Activités de scan
20. Violations de politique

Détection de communications malveillantes

L'identification des communications avec des infrastructures malveillantes est cruciale :

1. **Vérification contre des listes de réputation** : `vql SELECT RemoteAddr, Process, lookup_ip(ip=RemoteAddr) AS Reputation FROM netstat() WHERE Reputation.Score > 70`
2. Comparaison avec des bases de données de réputation
3. Identification d'IPs connues comme malveillantes

4. Évaluation du risque des communications

5. **Détection de domaines générés algorithmiquement (DGA)** : `vql SELECT Domain, entropy(string=Domain) AS Entropy, len(string=Domain) AS Length FROM dns_queries WHERE Entropy > 4 AND Length > 15 AND Domain !~ "\\.cdn\\.|cloudfront|akamai"`

6. Identification de domaines à haute entropie
7. Filtrage des faux positifs (CDNs légitimes)

8. Détection de techniques d'évasion réseau

9. **Analyse des certificats SSL/TLS** : `vql SELECT RemoteAddr, Process, ssl_connections(ip=RemoteAddr, port=RemotePort) AS Certificate FROM netstat() WHERE RemotePort = 443 AND (Certificate.Issuer !~ "DigiCert|Comodo|Let's Encrypt|GlobalSign" OR Certificate.ValidFrom < now() - 86400*365*5)`

10. Vérification des certificats utilisés
11. Détection de certificats auto-signés ou expirés

12. Identification de tentatives d'interception SSL
13. **Analyse des patterns de communication** : `vql SELECT DestinationIp, count() AS ConnectionCount, min(item=EventTime) AS FirstSeen, max(item=EventTime) AS LastSeen, LastSeen - FirstSeen AS Duration FROM sysmon_events WHERE EventID = 3 GROUP BY DestinationIp ORDER BY ConnectionCount DESC`
14. Identification des destinations les plus contactées
15. Détection de communications anormalement fréquentes
16. Analyse des patterns temporels de communication

Analyse de mouvement latéral

La détection des mouvements latéraux est essentielle pour comprendre la propagation d'une attaque :

1. **Détection de connexions SMB/RPC** : `vql SELECT Process, Pid, RemoteAddr, RemotePort FROM netstat() WHERE RemotePort IN (445, 135, 139) AND Process !~ "System|lsass.exe"`
2. Identification de connexions aux services Windows
3. Détection d'accès aux partages réseau
4. Repérage d'authentifications distantes
5. **Analyse des authentifications réseau** : `vql SELECT * FROM windows_events( eventlog="Security", eventid=4624 ) WHERE EventData.LogonType IN (3, 10)`
6. Identification des connexions réseau
7. Détection d'authentifications inhabituelles
8. Traçage des accès entre systèmes
9. **Détection d'outils d'administration distante** : `vql SELECT Process, CommandLine, Pid FROM pslist() WHERE Name =~ "psexec|wmic|winrm|sc.exe|at.exe|schtasks.exe" AND CommandLine =~ "\\\\\\\\[\\w\\.]+"`
10. Identification d'outils d'administration utilisés pour le mouvement latéral
11. Détection de commandes exécutées à distance
12. Repérage de techniques Living-off-the-Land

13. **Analyse des sessions distantes** : `vql SELECT * FROM windows_events( eventlog="Microsoft-Windows-TerminalServices-LocalSessionManager/Operational", eventid=21 )`

14. Identification des sessions RDP
15. Détection de connexions à distance
16. Traçage des accès interactifs

Bonnes pratiques pour l'analyse réseau

Pour maximiser l'efficacité de l'analyse réseau :

1. **Contextualisation** :
2. Comprenez l'architecture réseau normale
3. Établissez des lignes de base de communication
4. Tenez compte des applications et services légitimes

5. Considérez les modèles de communication métier

6. **Corrélation multi-sources** :

7. Combinez les données réseau avec d'autres artefacts
8. Corrélation avec les journaux d'authentification
9. Validation par l'analyse de processus

10. Confirmation via plusieurs sources de données

11. **Analyse temporelle** :

12. Établissez des chronologies de communication
13. Identifiez les séquences d'événements
14. Détectez les changements de comportement

15. Corrélation avec d'autres activités suspectes

16. **Documentation et partage** :

17. Documentez les IOCs réseau découverts
18. Partagez les indicateurs avec les équipes de sécurité
19. Créez des règles de détection basées sur les découvertes
20. Maintenez une base de connaissances des communications malveillantes

10. Cas d'usage pratiques

Cette section présente des cas d'usage concrets de Velociraptor pour illustrer comment l'outil peut être appliqué dans des scénarios d'investigation réels. Chaque cas d'usage détaille les étapes, les artefacts utilisés et les analyses effectuées.

Investigation d'une compromission par phishing

Ce scénario décrit l'investigation d'un poste de travail potentiellement compromis suite à un email de phishing.

Contexte

Un utilisateur signale avoir cliqué sur un lien dans un email suspect et avoir entré ses identifiants sur une page web ressemblant à la page de connexion de l'entreprise.

Objectifs

1. Déterminer si le poste de travail a été compromis.
2. Identifier les actions effectuées par l'attaquant.
3. Collecter les indicateurs de compromission (IOCs).
4. Évaluer l'étendue de la compromission.

Étapes d'investigation

1. Collecte initiale d'informations

- **Artefact utilisé** : `Generic.Client.Info`
- **Objectif** : Obtenir des informations de base sur le système (OS, nom d'hôte, utilisateur connecté).
- **Analyse** : Vérifier la configuration de base et l'état du client.

2. Analyse de l'activité de navigation

- **Artefacts utilisés** : `Windows.Forensics.ChromeHistory`, `Windows.Forensics.FirefoxHistory`, `Windows.Forensics.EdgeHistory`
- **Objectif** : Identifier les sites web visités au moment de l'incident.
- **Analyse** : Rechercher l'URL de phishing signalée, identifier les téléchargements suspects, examiner les cookies et l'historique récent.

3. Recherche de fichiers suspects

- **Artefact utilisé** : `Windows.Search.FileFinder`
- **Paramètres** :
- `SearchFilesGlob` : `C:\Users\<username>\Downloads\*`, `C:\Users\<username>\AppData\Local\Temp\*`
- `DateAfter` : Heure approximative de l'incident
- `UploadFiles` : `Y`
- **Objectif** : Identifier les fichiers potentiellement téléchargés via le phishing.
- **Analyse** : Examiner les fichiers téléchargés, vérifier leurs hachages contre VirusTotal, analyser statiquement les exécutables suspects.

4. Analyse des processus en cours et récents

- **Artefacts utilisés** : `Windows.System.Pslist`, `Windows.Forensics.Prefetch`
- **Objectif** : Identifier les processus suspects exécutés après l'incident.
- **Analyse** : Rechercher des processus inhabituels, des lignes de commande suspectes (PowerShell, cmd), corréler avec les fichiers téléchargés.

5. Analyse de la persistance

- **Artefacts utilisés** : `Windows.System.Autoruns`, `Windows.System.Services`, `Windows.System.TaskScheduler`
- **Objectif** : Détecter les mécanismes de persistance potentiellement installés.
- **Analyse** : Rechercher des entrées de démarrage automatique suspectes, des services ou tâches planifiées inhabituels créés après l'incident.

6. Analyse des journaux d'événements

- **Artefact utilisé** : `Windows.EventLogs.EvtxHunter`
- **Paramètres** :
- `EventLogNames` : `Security`, `System`, `Application`, `Microsoft-Windows-PowerShell/Operational`
- `DateAfter` : Heure approximative de l'incident
- **Objectif** : Identifier les événements suspects (connexions, création de processus, exécution de scripts).
- **Analyse** : Rechercher les événements de connexion inhabituels, les créations de processus liées aux fichiers suspects, les exécutions de scripts PowerShell obfusqués.

7. Analyse réseau

- **Artefacts utilisés** : `Windows.Network.NetstatEnriched`, `Windows.Network.DNSCache`
- **Objectif** : Identifier les communications réseau suspectes.
- **Analyse** : Rechercher les connexions vers des adresses IP ou domaines connus comme malveillants, identifier les communications C2 potentielles, examiner les résolutions DNS récentes.

Conclusion de l'investigation

- Synthétiser les découvertes dans un notebook Velociraptor.
- Lister les IOCs identifiés (hachages, IPs, domaines, fichiers).
- Évaluer le niveau de compromission et les actions de l'attaquant.
- Recommander des actions de remédiation (réinitialisation du mot de passe, réinstallation du poste, blocage des IOCs).

Chasse aux menaces : Détection de Cobalt Strike

Ce scénario décrit une chasse proactive visant à détecter la présence de Cobalt Strike, un outil de post-exploitation fréquemment utilisé par les attaquants.

Contexte

L'équipe de sécurité souhaite vérifier proactivement si des systèmes de l'organisation sont compromis par Cobalt Strike.

Objectifs

1. Identifier les systèmes potentiellement infectés par Cobalt Strike.
2. Collecter des preuves de l'activité de Cobalt Strike.
3. Comprendre les techniques utilisées par l'attaquant.

Étapes de la chasse

1. Configuration de la chasse

- **Nom** : Cobalt Strike Detection Hunt
- **Description** : Recherche proactive d'indicateurs de Cobalt Strike
- **Ciblage** : Tous les systèmes Windows
- **Limites** : Configurer des limites raisonnables pour minimiser l'impact.

2. Sélection des artefacts

- **Artefact 1** : `Windows.Detection.ProcessInjection`
- **Objectif** : Détecter les techniques d'injection de processus couramment utilisées par Cobalt Strike.
- **Artefact 2** : `Windows.Network.NetstatEnriched`
- **Objectif** : Identifier les connexions réseau suspectes pouvant correspondre à des communications C2 de Cobalt Strike (ports inhabituels, processus suspects).
- **Artefact 3** : `Windows.Detection.NamedPipes`
- **Objectif** : Rechercher les pipes nommés connus pour être utilisés par Cobalt Strike pour la communication inter-processus.
- **Artefact 4** : `Windows.Memory.YaraProcessMemory`
- **Paramètres** : Utiliser des règles YARA spécifiques à Cobalt Strike.
- **Objectif** : Scanner la mémoire des processus pour détecter les signatures de Cobalt Strike.
- **Artefact 5** : `Windows.System.Services`
- **Objectif** : Rechercher des services suspects potentiellement créés par Cobalt Strike pour la persistance.

3. Exécution et surveillance

- Lancer la chasse.
- Surveiller la progression et les erreurs éventuelles.
- Analyser les premiers résultats au fur et à mesure qu'ils arrivent.

4. Analyse des résultats

- **Process Injection** : Examiner les détections pour identifier les processus sources et cibles, vérifier si les patterns correspondent à Cobalt Strike.
- **Network Connections** : Filtrer les connexions suspectes, vérifier les IPs/domaines distants contre des listes de réputation, rechercher les ports par défaut de Cobalt Strike.
- **Named Pipes** : Identifier les pipes nommés correspondant aux signatures connues de Cobalt Strike.
- **Yara Memory Scan** : Analyser les détections YARA, identifier les processus infectés, extraire potentiellement les configurations de beacon.
- **Services** : Examiner les services suspects, vérifier les chemins d'exécutables et les descriptions.

5. Investigation approfondie des systèmes suspects

- Pour les systèmes présentant des indicateurs forts, lancer des investigations individuelles plus approfondies :
- Collecte de mémoire complète.
- Analyse temporelle détaillée.
- Collecte de fichiers suspects identifiés.
- Analyse des journaux d'événements spécifiques.

Conclusion de la chasse

- Documenter les systèmes identifiés comme potentiellement compromis.
- Synthétiser les preuves collectées pour chaque système.
- Initier des procédures de réponse aux incidents pour les systèmes confirmés.
- Mettre à jour les règles de détection et les futures chasses en fonction des découvertes.

Audit de conformité : Vérification des correctifs

Ce scénario décrit l'utilisation de Velociraptor pour vérifier si les correctifs de sécurité critiques ont été appliqués sur l'ensemble du parc Windows.

Contexte

Une vulnérabilité critique a été annoncée et un correctif publié. L'équipe de sécurité doit vérifier rapidement si tous les systèmes Windows sont protégés.

Objectifs

1. Identifier les systèmes Windows sur lesquels le correctif spécifique (KBxxxxxxx) est manquant.
2. Évaluer le niveau de risque lié à cette vulnérabilité dans l'organisation.
3. Fournir une liste des systèmes nécessitant une remédiation urgente.

Étapes de l'audit

1. Configuration de la chasse

- **Nom** : Audit Correctif KBxxxxxxx
- **Description** : Vérification de l'installation du correctif KBxxxxxxx
- **Ciblage** : Tous les systèmes Windows
- **Limites** : Faible impact attendu, limites standard.

2. Sélection de l'artefact

- **Artefact utilisé** : `Windows.System.PatchLevel`
- **Paramètres** :
- `KBID` : `KBxxxxxxx` (remplacer par le numéro de KB réel)
- **Objectif** : Vérifier la présence ou l'absence du correctif spécifié sur chaque système.

3. Exécution et surveillance

- Lancer la chasse.
- Surveiller la progression.
- La chasse devrait être rapide car l'artefact est léger.

4. Analyse des résultats

- **Filtrage des résultats** : `vql SELECT Computer, OSVersion, PatchStatus FROM source(artifact="Windows.System.PatchLevel", hunt_id="H.audit") WHERE PatchStatus = "Missing"`
- **Identification des systèmes vulnérables** : La requête ci-dessus liste tous les systèmes où le correctif est manquant.
- **Évaluation du risque** : Analyser la distribution des systèmes vulnérables (serveurs critiques, postes de travail, etc.).
- **Génération de rapport** : Exporter la liste des systèmes vulnérables pour l'équipe de gestion des correctifs.

Conclusion de l'audit

- Fournir la liste des systèmes non conformes à l'équipe responsable.
- Planifier une nouvelle chasse après la campagne de déploiement pour vérifier l'efficacité de la remédiation.
- Utiliser les résultats pour améliorer les processus de gestion des correctifs.

Réponse à un incident Ransomware

Ce scénario décrit les étapes initiales de réponse à un incident impliquant une suspicion d'attaque par ransomware.

Contexte

Plusieurs utilisateurs signalent des fichiers chiffrés et des demandes de rançon affichées sur leurs écrans.

Objectifs

1. Confirmer l'infection par ransomware.
2. Identifier la variante du ransomware.
3. Déterminer l'étendue de l'infection.
4. Identifier le point d'entrée initial (si possible rapidement).
5. Collecter des échantillons et des IOCs.

Étapes de réponse initiale

1. Isolation des systèmes suspects

- Utiliser les capacités d'isolation réseau (si disponibles via EDR ou pare-feu) ou déconnecter physiquement les machines suspectes pour contenir la propagation.

2. Configuration de la chasse de triage

- **Nom** : Ransomware Triage Hunt
- **Description** : Collecte initiale d'informations sur les systèmes suspects de ransomware.
- **Ciblage** : Systèmes signalés et potentiellement voisins.
- **Limites** : Priorité élevée, limites de ressources ajustées pour une collecte rapide.

3. Sélection des artefacts de triage

- **Artefact 1** : `Windows.Search.FileFinder`
- **Paramètres** : Rechercher les extensions de fichiers chiffrés connues, les notes de rançon (`*.txt`, `*.html` avec des mots-clés comme "ransom", "decrypt"). Upload des notes de rançon.
- **Objectif** : Confirmer le chiffrement et identifier la variante via la note de rançon.
- **Artefact 2** : `Windows.System.Pslist`
- **Objectif** : Identifier les processus suspects potentiellement liés au ransomware.
- **Artefact 3** : `Windows.Forensics.Prefetch`
- **Objectif** : Identifier les exécutables récemment lancés, y compris potentiellement le binaire du ransomware.
- **Artefact 4** : `Windows.EventLogs.EvtxHunter`
- **Paramètres** : Rechercher les événements de création de processus suspects, les événements liés aux services Shadow Copy (VSS) souvent supprimés par les ransomwares.
- **Objectif** : Tracer l'exécution initiale et les actions du ransomware.
- **Artefact 5** : `Generic.Client.Info`
- **Objectif** : Collecter les informations de base pour chaque système affecté.

4. Exécution et analyse rapide

- Lancer la chasse sur les systèmes suspects.
- Analyser les résultats au fur et à mesure :
- Confirmer le chiffrement et identifier la variante (note de rançon, extensions).
- Identifier le processus suspect (Pslist, Prefetch).
- Rechercher les événements de suppression de VSS (journaux d'événements).
- Collecter les hachages des exécutables suspects.

5. Collecte d'échantillons et investigation approfondie

- Sur les systèmes confirmés, collecter les échantillons du ransomware identifiés via Prefetch ou Pslist.
- **Artefact** : `Windows.Search.FileFinder` (avec chemin exact ou hachage) ou collecte manuelle via VFS.
- Lancer des investigations plus approfondies sur des systèmes clés pour identifier le point d'entrée :
- Analyse temporelle détaillée.
- Analyse des journaux d'authentification.
- Recherche de mouvements latéraux.

Conclusion de la réponse initiale

- Confirmer l'étendue de l'infection.
- Identifier la variante du ransomware et rechercher des décrypteurs potentiels.
- Collecter les IOCs (hachages, noms de fichiers, notes de rançon) pour blocage.
- Fournir les informations aux équipes de remédiation et de communication.
- Planifier les étapes suivantes de l'investigation (point d'entrée, exfiltration de données éventuelle).

Ces cas d'usage illustrent la flexibilité et la puissance de Velociraptor pour différents scénarios de sécurité. En adaptant les artefacts, les paramètres et les flux de travail, les équipes de sécurité peuvent répondre efficacement à une large gamme de menaces et d'incidents.

11. Conclusion et ressources

Synthèse des capacités de Velociraptor

Velociraptor s'est imposé comme un outil incontournable dans le domaine de la réponse aux incidents et de l'investigation numérique. Au terme de ce manuel complet, il convient de synthétiser les principales capacités qui font la force de cette plateforme.

Puissance et flexibilité

Velociraptor se distingue par sa flexibilité exceptionnelle, principalement grâce au langage VQL (Velociraptor Query Language) qui permet aux analystes de créer des requêtes personnalisées adaptées à leurs besoins spécifiques. Cette approche offre plusieurs avantages majeurs :

1. **Adaptabilité** : Contrairement aux outils traditionnels qui nécessitent des mises à jour pour chaque nouveau type d'analyse, Velociraptor permet aux utilisateurs de développer rapidement de nouvelles capacités via VQL.

2. **Expressivité** : La syntaxe inspirée de SQL, combinée à une architecture extensible basée sur des plugins, permet d'exprimer des analyses complexes de manière concise et lisible.

3. **Évolutivité** : L'architecture de Velociraptor est conçue pour évoluer avec les besoins des organisations, depuis les petites équipes jusqu'aux déploiements à grande échelle.

Efficacité opérationnelle

Velociraptor optimise l'efficacité des équipes de sécurité de plusieurs façons :

1. **Rapidité d'exécution** : Les chasses permettent de déployer des investigations sur des milliers de systèmes simultanément, réduisant considérablement le temps nécessaire pour identifier l'étendue d'une compromission.

2. **Automatisation** : Les artefacts encapsulent l'expertise forensique et permettent d'automatiser des tâches d'investigation complexes, rendant les analyses avancées accessibles même aux analystes moins expérimentés.

3. **Réduction des délais** : En éliminant le besoin de déployer des outils spécialisés pour chaque investigation, Velociraptor permet une réponse immédiate aux incidents.

4. **Centralisation** : La plateforme offre un point central pour la collecte, l'analyse et la documentation des investigations, facilitant la collaboration et le partage de connaissances.

Visibilité et contrôle

Velociraptor offre une visibilité sans précédent sur les environnements informatiques :

1. **Visibilité en temps réel** : Capacité à interroger les systèmes en direct et à surveiller les événements en continu.
2. **Analyse approfondie** : Accès à des données forensiques détaillées, depuis le système de fichiers jusqu'à la mémoire volatile.
3. **Détection avancée** : Capacité à identifier des menaces sophistiquées grâce à des analyses comportementales et des corrélations multi-sources.
4. **Documentation complète** : Les notebooks permettent de documenter les investigations de manière structurée et reproductible.

Communauté et écosystème

L'un des atouts majeurs de Velociraptor réside dans sa communauté active et son écosystème ouvert :

1. **Open Source** : Le code source ouvert garantit la transparence et permet aux organisations de personnaliser l'outil selon leurs besoins.
2. **Partage d'artefacts** : L'Artifact Exchange facilite le partage d'expertise entre professionnels de la sécurité.
3. **Innovation continue** : La communauté contribue régulièrement à l'amélioration de l'outil et au développement de nouvelles capacités.
4. **Documentation et formation** : Ressources abondantes pour l'apprentissage et le perfectionnement.

Évolutions futures et tendances

Velociraptor continue d'évoluer pour répondre aux défis émergents de la cybersécurité. Plusieurs tendances se dessinent pour l'avenir de la plateforme :

Intégration et interopérabilité

L'intégration avec d'autres outils et plateformes de sécurité devient de plus en plus importante :

1. **Intégration SIEM/SOAR** : Développement de connecteurs pour les principales plateformes de gestion des informations et événements de sécurité et d'orchestration des réponses.
2. **APIs et automatisation** : Expansion des capacités d'API pour faciliter l'intégration dans des workflows automatisés.
3. **Formats standardisés** : Support accru pour les formats d'échange standardisés comme STIX/TAXII pour le partage d'indicateurs de menaces.

Intelligence artificielle et apprentissage automatique

L'incorporation de techniques d'IA et d'apprentissage automatique représente une évolution naturelle :

1. **Détection d'anomalies** : Utilisation de l'apprentissage automatique pour identifier des comportements anormaux sans règles prédéfinies.
2. **Priorisation intelligente** : Algorithmes pour hiérarchiser les alertes et les résultats d'investigation selon leur criticité probable.
3. **Analyse prédictive** : Anticipation des mouvements potentiels des attaquants basée sur les actions observées.

Expansion des capacités cloud et conteneurs

Avec l'adoption croissante des environnements cloud et des architectures conteneurisées :

1. **Support natif des plateformes cloud** : Développement d'artefacts spécifiques pour AWS, Azure, GCP et autres fournisseurs cloud.
2. **Analyse de conteneurs** : Capacités améliorées pour l'investigation de conteneurs et d'orchestrateurs comme Kubernetes.
3. **Architectures sans agent** : Exploration de méthodes d'investigation ne nécessitant pas d'agent pour certains environnements cloud.

Réponse automatisée

L'évolution vers des capacités de réponse automatisée :

1. **Playbooks intégrés** : Développement de workflows de réponse prédéfinis pour les scénarios courants.
2. **Remédiation automatique** : Capacités pour corriger automatiquement certaines compromissions selon des critères prédéfinis.
3. **Isolation dynamique** : Mécanismes pour isoler automatiquement les systèmes compromis tout en maintenant la visibilité.

Ressources complémentaires

Pour approfondir vos connaissances et rester à jour avec les évolutions de Velociraptor, voici une sélection de ressources essentielles :

Documentation officielle

- **Site web officiel** : https://www.velocidex.com/
- **Documentation** : https://docs.velociraptor.app/
- **GitHub** : https://github.com/Velocidex/velociraptor

Communauté et support

- **Forum de discussion** : https://forum.velocidex.com/
- **Canal Discord** : https://discord.gg/velocidex
- **Artifact Exchange** : https://docs.velociraptor.app/exchange/

Formation et apprentissage

- **Blog Velocidex** : https://www.velocidex.com/blog/
- **Tutoriels vidéo** : Chaîne YouTube Velocidex
- **Environnement de test** : Velociraptor Playground

Conférences et présentations

- **VeloCon** : Conférence annuelle dédiée à Velociraptor
- **SANS DFIR Summit** : Présentations régulières sur Velociraptor
- **BlackHat/DefCon** : Ateliers et présentations sur les cas d'usage avancés

Livres et publications

- "Digital Forensics with Velociraptor" (à paraître)
- "Incident Response with Open Source Tools" - Chapitre sur Velociraptor
- Papers académiques sur l'utilisation de VQL pour l'investigation numérique

Mot de la fin

Velociraptor représente une évolution significative dans le domaine de la réponse aux incidents et de l'investigation numérique. Sa flexibilité, sa puissance et son approche ouverte en font un outil précieux pour les équipes de sécurité de toutes tailles.

Ce manuel a couvert les aspects fondamentaux et avancés de Velociraptor, depuis l'installation et la configuration jusqu'aux analyses sophistiquées et aux cas d'usage pratiques. Cependant, la véritable maîtrise de cet outil viendra avec la pratique, l'expérimentation et la participation à la communauté.

Nous vous encourageons à explorer les capacités de Velociraptor dans votre propre environnement, à contribuer à l'écosystème en partageant vos artefacts et vos découvertes, et à rester curieux face aux évolutions constantes du domaine de la cybersécurité.

La sécurité est un voyage, non une destination, et Velociraptor est un compagnon précieux sur ce chemin. Bonne chasse !

Bibliographie

Voici une sélection de références qui ont contribué à l'élaboration de ce manuel et qui peuvent servir de ressources complémentaires pour approfondir certains aspects de Velociraptor et de l'investigation numérique.

Publications officielles Velociraptor

1. Cohen, M. (2023). "Velociraptor Documentation". Velocidex Enterprises. https://docs.velociraptor.app/

2. Cohen, M. (2022). "VQL Reference Guide". Velocidex Enterprises. https://docs.velociraptor.app/vql_reference/

3. Velocidex Team (2023). "Velociraptor Artifact Exchange". https://docs.velociraptor.app/exchange/

Articles et présentations

1. Cohen, M. (2021). "Velociraptor - Digging Deeper". Présentation à SANS DFIR Summit.
2. Bromiley, M. (2022). "Threat Hunting at Scale with Velociraptor". SANS Institute Whitepaper.
3. Hafemann, T. (2023). "Advanced VQL for Incident Response". Présentation à VeloCon 2023.
4. Stokes, J. (2022). "Comparing DFIR Tools: Velociraptor vs. Traditional Approaches". Digital Forensics Magazine, Vol. 43.

Livres et chapitres

1. Cohen, M. & Bromiley, M. (2023). "Digital Forensics with Velociraptor". CRC Press.
2. Carrier, B. (2022). "Open Source Digital Forensics". Chapitre 7: "Velociraptor for Enterprise Investigations". Wiley.
3. Carvey, H. (2023). "Windows Forensic Analysis". Chapitre 12: "Advanced Collection with Velociraptor". Syngress.

Articles académiques

1. Johnson, A. et al. (2022). "Comparative Analysis of Digital Forensic Platforms for Enterprise Environments". Journal of Digital Investigation, 41, 301-315.
2. Patel, S. & Rodriguez, M. (2023). "VQL: A Novel Approach to Digital Forensics Query Languages". IEEE Symposium on Security and Privacy, 1023-1037.
3. Williams, T. (2022). "Performance Evaluation of Remote Forensic Collection Tools". Digital Forensics Research Workshop (DFRWS), Proceedings.

Standards et frameworks

1. NIST (2023). "Guide to Integrating Forensic Techniques into Incident Response" (SP 800-86 Rev. 1). National Institute of Standards and Technology.
2. SANS Institute (2023). "Incident Handler's Handbook". SANS Reading Room.

3. MITRE (2023). "ATT&CK Framework v13". MITRE Corporation. https://attack.mitre.org/

Blogs et ressources en ligne

1. Velocidex Blog (2023). "VQL Tips and Tricks Series". https://www.velocidex.com/blog/

2. Digital Forensics Discord (2023). "Velociraptor Channel Archives". https://discord.gg/digitalforensics

3. GitHub Discussions (2023). "Velociraptor Community Artifacts". https://github.com/Velocidex/velociraptor/discussions

Ressources de formation

1. SANS FOR508 (2023). "Advanced Digital Forensics, Incident Response, and Threat Hunting". Module: "Enterprise-Scale Hunting with Velociraptor".

2. Cybrary (2023). "Velociraptor Mastery Course". Formation en ligne.

3. Velocidex (2023). "Velociraptor Training Materials". https://docs.velociraptor.app/training/

Cette bibliographie n'est pas exhaustive mais offre un point de départ pour explorer davantage les concepts et techniques présentés dans ce manuel. Les dates indiquées correspondent aux versions les plus récentes au moment de la rédaction.