

# CVE 2020-25637

Problème de double *free* dans libvirt

BRAHMI Boudjema  
AYYACH Oussama

Mr. F-X Dupé



7 Décembre 2020

Version 1.2

# Sommaire

- I. Présentation générale de la faille.
- II. Double free ().
- III. Présentation des outils.
- IV. Installation et prérequis de l'environnement de travail.
- V. Présentation technique.
- VI. Démonstration.
- VII. Conclusion.

# Présentation générale de la faille.

- ◆ Présentation de la CVE
- ◆ Pourquoi elle est importante



- Dans ce chapitre, on parlera de la faille d'une vue globale, sa description et pourquoi est-elle importante ?

# Présentation générale de la faille

## Présentation de la CVE

- Le problème de double free a été constaté dans l'API libvirt, dans les versions antérieures à 6.8.0, chargée de demander des informations sur les interfaces réseau d'un domaine QEMU en cours d'exécution.
- Cette faille affecte le pilote de contrôle d'accès polkit. Plus précisément, les clients se connectant à la socket en lecture-écriture avec des permissions ACL limitées pourraient utiliser cette faille pour faire planter le démon libvirt.

# Présentation générale de la faille

## Présentation de la CVE

- Date d'allocation de ID: 16/09/2020
- Trouvée sur: QEMU jusqu'à la version 6.7.x
- Répartition des scores selon CVSS

	Vecteur	Complexité	Authentification	Score (CVSS v3)
<b>Red Hat</b>	Local	Faible	Aucune	6.7
<b>NVD</b>	Local	Élevée	Aucune	6.4

# Présentation générale de la faille

Pourquoi la faille est importante ?

- La menace la plus importante de cette vulnérabilité concerne la confidentialité et l'intégrité des données ainsi que la disponibilité du système.

Confidentialité	Intégrité	Disponibilité
Oui	Oui	Oui



# Double free ()

- 💧 Définition
- 💧 Comment ça fonctionne ?
- 💧 Memory leak & double free



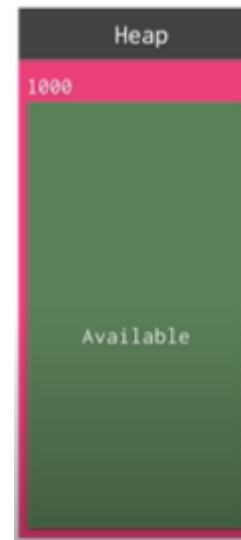
- La deuxième partie est consacrée pour parler du problème de double free(), sa description et comment ça fonctionne.

# Double free ()

## Définition

- Un double free () est tout simplement là où un chunk de mémoire qui a été précédemment alloué par l'une des routines d'allocation qui est plus tard free () plus d'une fois, généralement deux fois donc le titre.

free (ptr)



free (ptr)

double  
free() is  
undefined



# Double free ()

Comment ça fonctionne ?

- Appeler deux fois `free()` sur la même valeur peut entraîner une fuite de mémoire. Lorsqu'un programme appelle `free()` deux fois avec le même argument, les structures de données de gestion de la mémoire du programme sont corrompues et peuvent permettre à un utilisateur malveillant d'écrire des valeurs dans des espaces mémoire arbitraires.

```
char* ptr = malloc(sizeof(char));  
  
*ptr = 'a';  
free(ptr);  
free(ptr);
```

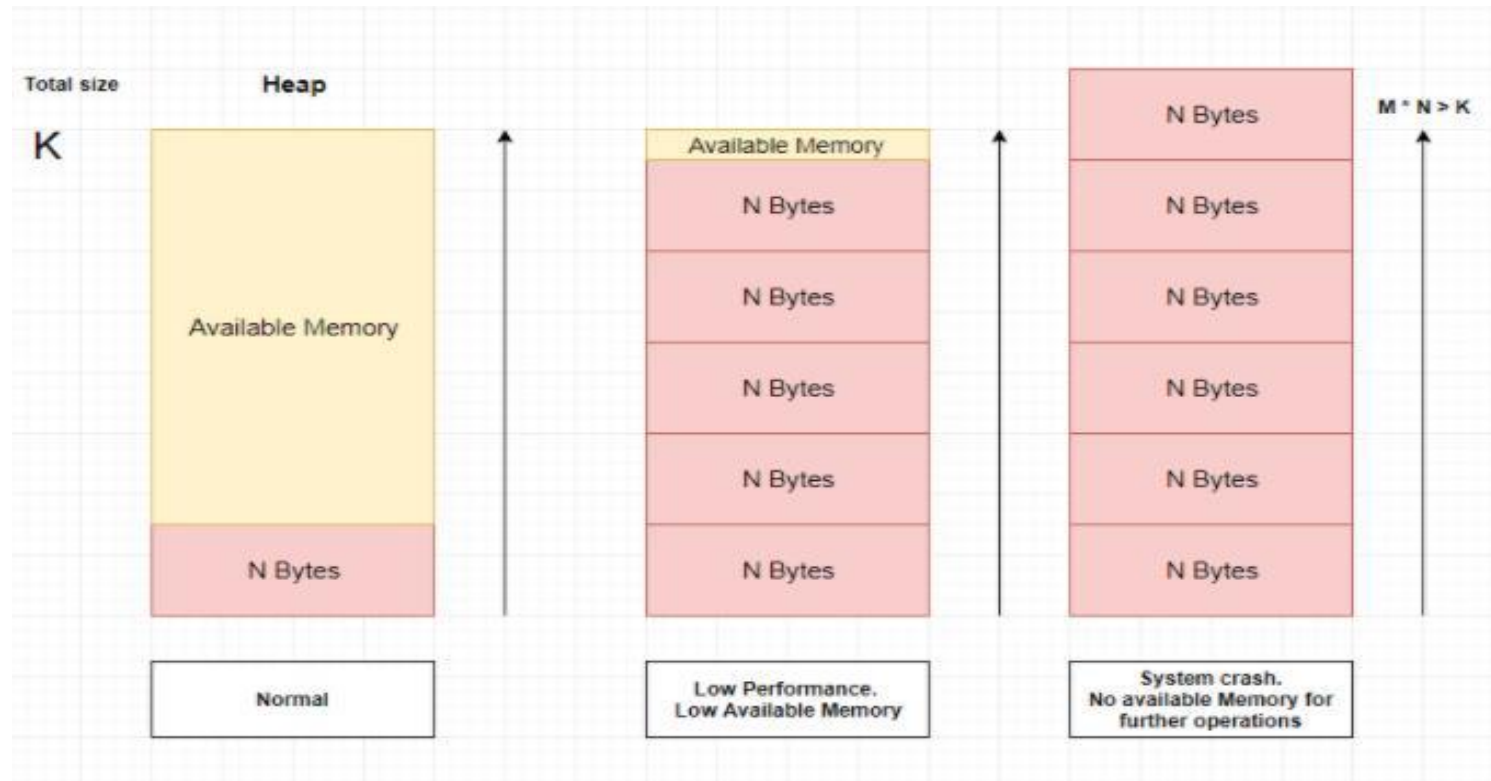
# Double free ()

## Memory leak & double free ()

- En informatique, une fuite de mémoire est un type particulier de consommation non intentionnelle de mémoire par un programme qui ne libère pas comme il le devrait la mémoire dont il n'a plus besoin.
  - Si nous ne désallouons pas la mémoire dynamique, elle résidera dans la section du heap.
- ↓ La quantité de mémoire = ↓ les performances du système

# Double free ()

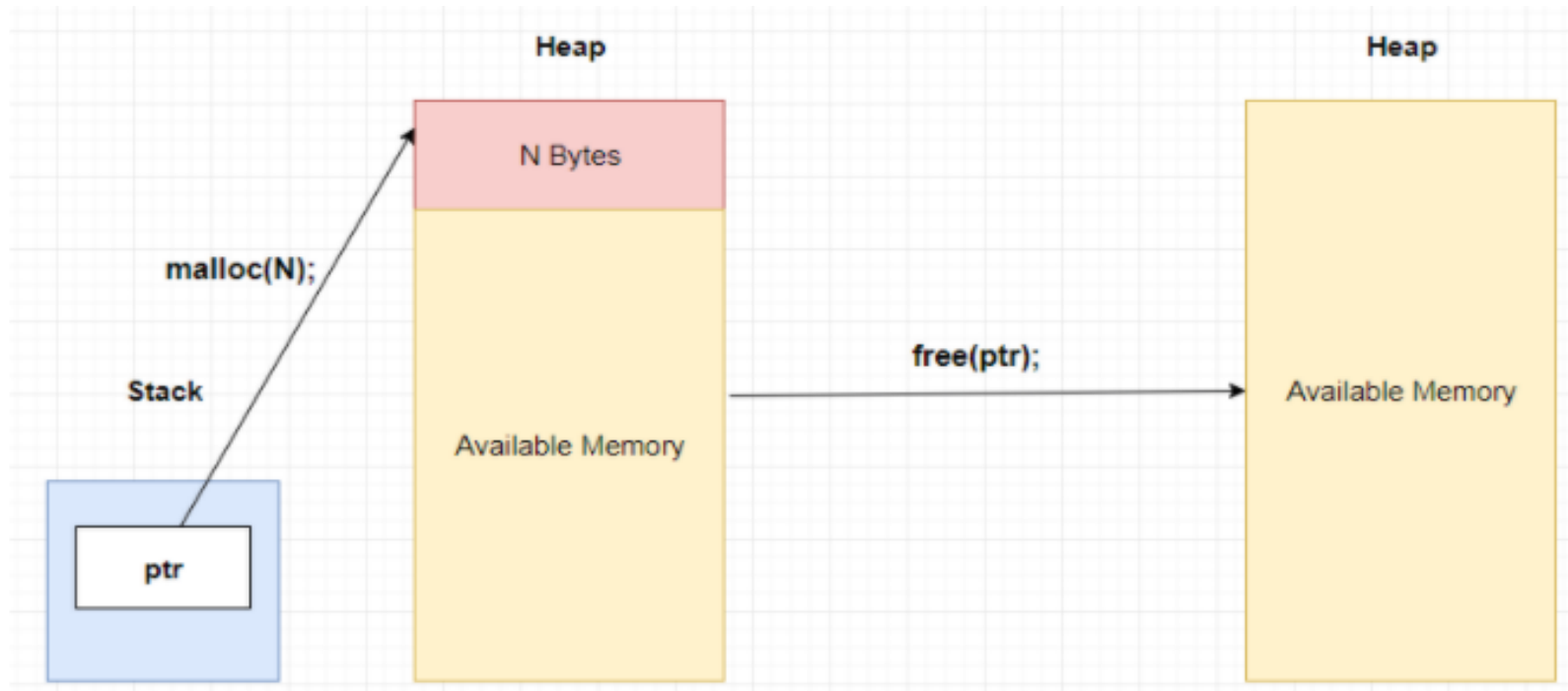
## Memory leak & double free ()



# Double free ()

## Memory leak & double free ()

```
char *ptr; ptr = malloc(N); //do something free(ptr);
```



# Présentation des outils.

- ◆ Présentation d'un Schéma
- ◆ Présentation KVM
- ◆ Présentation Qemu
- ◆ Présentation Libvirt
- ◆ Présentation de pilote Polkit
- ◆ Présentation des outils de débogage

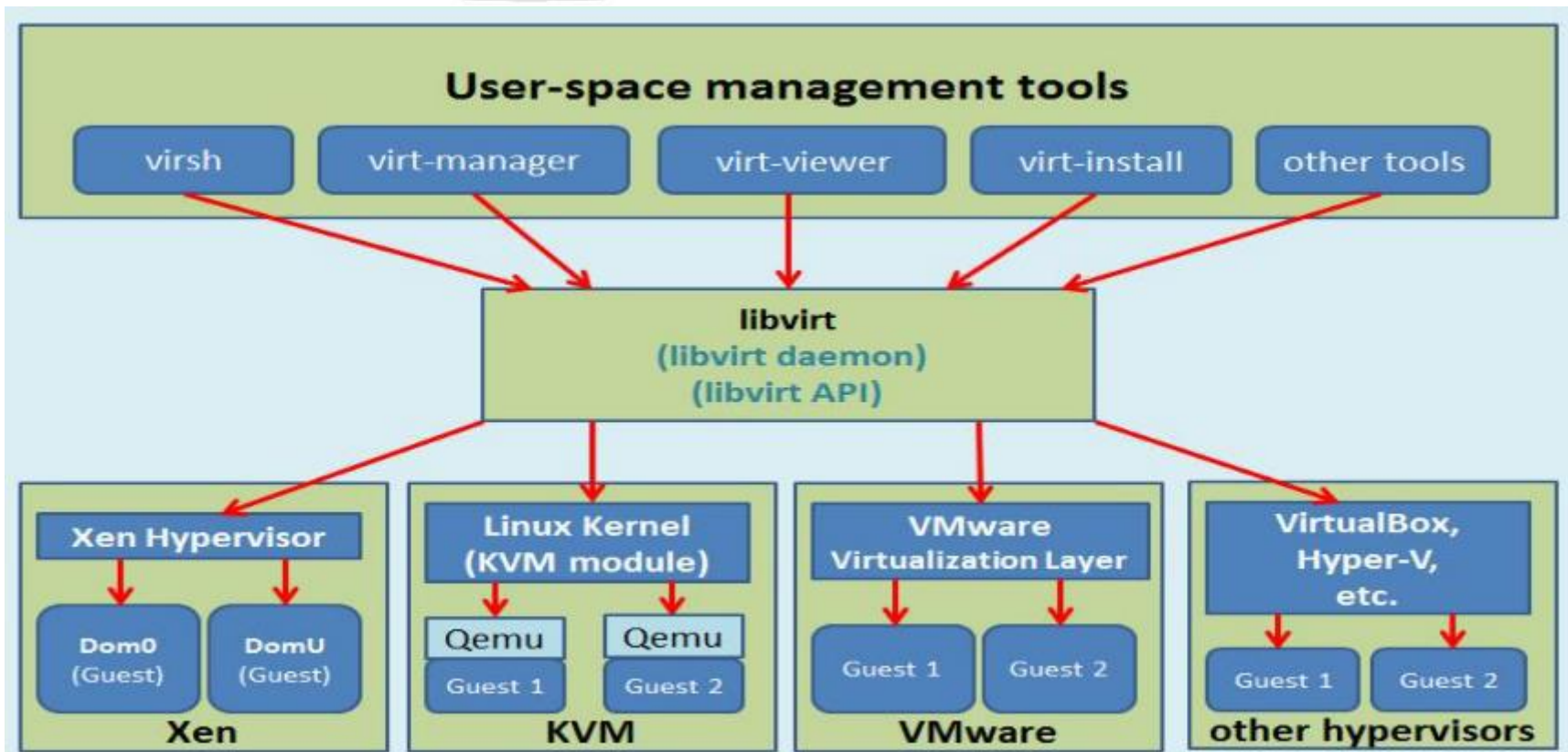


- Dans ce chapitre, on parlera de différents outils importants pour exploiter cette faille comme par exemple la bibliothèque libvirt, émulateur d'architecture Qemu, l'hyperviseur KVM etc.



# Présentation d'un Schéma

## Présentation des outils



# Présentation KVM

## Présentation des outils

### ■ KVM (Kernel-based Virtual Machine) :

La machine virtuelle basée sur le noyau (KVM) est une technologie de virtualisation open source. KVM vous permet de transformer Linux en un hyperviseur de type 1 qui permet à une machine hôte d'exécuter plusieurs environnements virtuels isolés appelés invités ou machines virtuelles (VM).



# Présentation Qemu

## Présentation des outils

### ■ Qemu :

Qemu est un émulateur de diverses architectures dont x86 (Hyperviseur de type 2).

Combiné au pilote KVM, il permet de réaliser de l'accélération Hardware (HVM).

# Présentation Libvirt

## Présentation des outils

### ■ Libvirt :

libvirt est un ensemble d'outils fournissant une API à de nombreuses technologies de virtualisation différentes. Lorsque vous utilisez libvirt, vous n'avez pas à vous soucier de ce "bakend" compliqué: Xen, KVM, VirtualBox ou autre. De plus, vous pouvez utiliser libvirt à l'intérieur des programmes Ruby (ainsi que Python, C ++ et bien d'autres).

Vous pouvez également vous connecter à distance aux machines virtuelles via des canaux sécurisés.

# Présentation de pilote Polkit

## Présentation des outils

### ■ Le pilote Polkit :

Une installation par défaut de libvirt utilisera généralement polkit pour authentifier la connexion utilisateur initiale à libvirtd.

Il s'agit d'une vérification très grossière, permettant soit un accès complet en lecture-écriture à toutes les API, soit simplement un accès en lecture seule. Le pilote de contrôle d'accès polkit de libvirt s'appuie sur cette capacité pour permettre un contrôle fin sur les opérations qu'un utilisateur peut effectuer sur un objet.

# Présentation des outils de débogage

## Présentation des outils

### ■ Les outils de débogage :

#### 1. GDB

un débogueur portable qui fonctionne sur de nombreux systèmes de type Unix et fonctionne pour de nombreux langages de programmation.

#### 2. Valgrind

Valgrind est un outil de programmation pour le débogage de la mémoire, la détection des fuites de mémoire et le profilage.

# Installation et prérequis de l'environnement de travail.

- ✦ Installation qemu-kvm et libvirt
- ✦ Installation vagrant
- ✦ Boite vagrant
- ✦ Lancer une VM depuis un fichier vagrant



- Dans ce chapitre, on parlera de la mise en place de l'environnement de travail.

# Installation qemu-kvm et libvirt

Installation et prérequis de l'environnement de travail

## ■ Installation qemu-kvm et libvirt :

Avant tout il est nécessaire de vérifier la compatibilité de notre système d'exploitation il suffit de passer cette commande:

```
grep -E -c "vmx|svm" /proc/cpuinfo
```

Si la commande répond ok vous pouvez procéder à l'installation

```
sudo apt-get -y install qemu-kvm libvirt-bin virt-top outils-  
libguestfs virtinst bridge-utils
```

# Installation vagrant

Installation et péréquer de l'environnement de travail

## ■ Installation vagrant :

```
sudo apt -y install vagrant
```

Une fois que vous avez installé Vagrant et KVM, vous devriez être prêt à installer un plugin libvirt pour pouvoir commencer à gérer les machines virtuelles KVM à l'aide de Vagrant.

```
vagrant plugin install vagrant-libvirt
```

vous pouvez confirmer que le plugin a été installé

```
liste de plugins
```



# Boite vagrant

Installation et péréquer de l'environnement de travail

## ■ Boite vagrant :

Il est possible d'utiliser un modèle prêt directement par vagrant dans notre cas on utilisera une image ubuntu 20.04.

Pour ajouter notre image a la boite vagrant il suffit de passer cette commande

```
vagrant box add generic / ubuntu1804 --provider = libvirt
```

vous pouvez Consultez la liste des boîtes présente localement

```
vagrant box list
```

# Lancer une VM depuis un fichier vagrant

Installation et p  r  quer de l'environnement de travail

## ■ Exemple de fichier vagrant :

```
# -*- mode: ruby -*-  
# vi: set ft=ruby :  
ENV['VAGRANT_DEFAULT_PROVIDER'] = 'libvirt'  
Vagrant.configure("2") do |config|  
  ##### DEFINE VM #####  
  config.vm.define « ubuntu-01" do |  
    config| config.vm.hostname = « ubuntu-01"  
    config.vm.box = «generic/ubuntu2004"  
    config.vm.box_check_update = false  
    config.vm.network "private_network", ip: "192.168.18.9"  
    config.vm.provider :libvirt do |v|  
      v.memory = 1024  
    end  
  end  
End
```

Vous pouvez lancer cette vm :

**vagrant up**

# Présentation technique.

- 💧 La faille dans Libvirt
- 💧 Explication du Code
- 💧 Exécution du code
- 💧 Comment activer la faille
- 💧 Correction de la faille



- Dans ce chapitre, on parlera de la pratique de la faille dans Libvirt ensuite on a un code qui rentre en jeu pour activer la faille finalement la proposition de la solution.

# La faille dans Libvirt

## Présentation technique

### ■ La faille dans Libvirt :

Un attaquant peut provoquer l'utilisation d'une zone mémoire libérée via `qemuAgentGetInterfaces ()` de libvirt, précisément dans la zone de mémoire de la variable `ifname` afin de mener un déni de service, et éventuellement d'exécuter du code.

Cette fonction traite une liste des interfaces d'un domaine en cours d'exécution avec leur adresses IP et MAC .

Ensuite renvoie le nombre d'interfaces en cas de succès, sinon -1 en cas d'erreur.

# Explication du code

## Présentation technique

```
21 static void
22 getDomainInfo(const char *uri, const char *name)
23 {
24     virConnectPtr conn = NULL; /* the hypervisor connection */
25     virDomainPtr dom = NULL; /* the domain being checked */
26     virDomainInfo info; /* the information being fetched */
27     int ret;
28
29     conn = virConnectOpen(uri);
30     if (conn == NULL) {
31         fprintf(stderr, "Failed to connect to hypervisor\n");
32         goto error;
33     }
34
35     /* Find the domain of the given name */
36     dom = virDomainLookupByName(conn, name);
37     if (dom == NULL) {
38         fprintf(stderr, "Failed to find Domain %s\n", name);
39         goto error;
40     }
41
42     /* Get the information */
43     ret = virDomainGetInfo(dom, &info);
44     if (ret < 0) {
45         fprintf(stderr, "Failed to get information for Domain %s\n", name);
46         goto error;
47     }
```

Se connecter au démon à l'aide d'un URI pour gérer les instances de qemu.

Essayez de rechercher un domaine sur l'hyperviseur donné en fonction de son nom.

Extraire des informations sur un domaine. Notez que si la connexion utilisée pour obtenir le domaine est limitée, seul un ensemble partiel d'informations peut être extrait.

# Explication du code

## Présentation technique

```
42  /* Get the information */
43  ret = virDomainGetInfo(dom, &info);
44  if (ret < 0) {
45      fprintf(stderr, "Failed to get information for Domain %s\n", name);
46      goto error;
47  }
48
49  /* Get iface information */
50  virDomainInterfacePtr *ifaces = NULL;
51  int ifaces_count = 0;
52  size_t i, j;
53
54  if ((ifaces_count = virDomainInterfaceAddresses(dom, &ifaces,
55      VIR_DOMAIN_INTERFACE_ADDRESSES_SRC_AGENT, 0)) < 0)
56      goto cleanup;
57
58  for (i = 0; i < ifaces_count; i++) {
59      printf("name: %s", ifaces[i]->name);
60      if (ifaces[i]->hwaddr)
61          printf(" hwaddr: %s", ifaces[i]->hwaddr);
62
63      for (j = 0; j < ifaces[i]->naddrs; j++) {
64          virDomainIPAddressPtr ip_addr = ifaces[i]->addrs + j;
65          printf("[addr: %s prefix: %d type: %d]",
66              ip_addr->addr, ip_addr->prefix, ip_addr->type);
67      }
68      printf("\n");
69  }
```

Renvoie un pointeur vers le tableau alloué de pointeurs vers les interfaces présentes dans un domaine donné avec leurs adresses IP et MAC. Notez qu'une seule interface peut avoir plusieurs adresses IP, voire 0.



# Explication du code

## Présentation technique

```
71 cleanup:
72     if (ifaces && ifaces_count > 0)
73         for (i = 0; i < ifaces_count; i++)
74             virDomainInterfaceFree(ifaces[i]);
75     free(ifaces);
76
77     printf("Domain %s: %d CPUs\n", name, info.nrVirtCpu);
78
79 error:
80     if (dom != NULL)
81         virDomainFree(dom);
82     if (conn != NULL)
83         virConnectClose(conn);
84 }
85
86 int main(int argc, char **argv)
87 {
88     if (argc != 3) {
89         fprintf(stderr, "syntax: %s: URI NAME\n", argv[0]);
90         return 1;
91     }
92     getDomainInfo(argv[1], argv[2]);
93
94     return 0;
95 }
```

Libérez l'objet d'interface. La structure de données est libérée et ne doit plus être utilisée par la suite.

Libérez l'objet de domaine. L'instance en cours d'exécution est maintenue en vie. La structure de données est libérée et ne doit plus être utilisée par la suite.



# Exécution du code

## Présentation technique

- Avant tout installer la bibliothèque :

```
sudo apt-get install -y libvirt-dev
```

- Compilation :

```
gcc -g -Wall info1.c -o info1 -lvirt
```

- Exécution du code :

```
./info1 qemu:///system vagrant-vms_ubuntu-01
```

- Exécution avec débogage :

```
valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes  
--verbose --log-file=valgrind-out.txt ./info1 qemu:///system  
vagrant-vms_ubuntu-01
```

# Comment activer cette faille

## Présentation technique

- Une fois le fichier vagrant est lancer, il faut lancer le code précédant dans une machine virtuelle elle-même lancée par Qemu/KVM. Donc faire des VMs dans une VM c'est à dire exécuter le code dans la VM crée par vagrant.
- Il est possible d'exécuter ce code plusieurs fois dans autre VMs (bien sur dans les VMs lancer par Qemu/KVM),

# Correction de cette faille

## Présentation technique

- Il suffit de passer la variable ifname à null c'est à dire la variable ifname doit être libérée pour chaque interface.

# Démonstration.

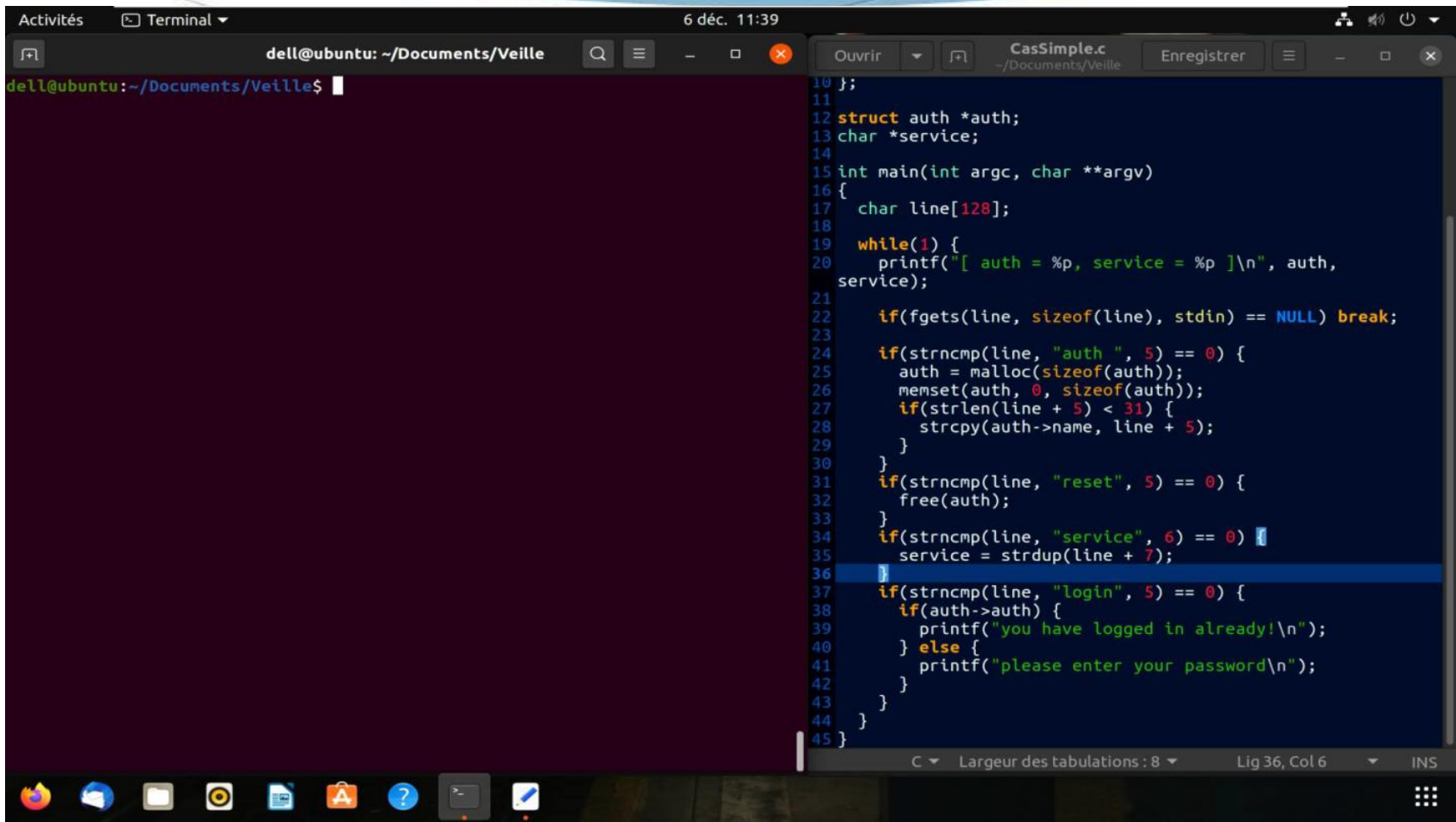
- Un cas simple
- Un cas difficile



- Dans ce chapitre, on fera deux cas de démonstration un cas facile et un cas difficile (Libvirt)

# Un cas simple

## démonstration



The screenshot shows a Linux desktop environment. On the left, a terminal window titled 'Terminal' displays the prompt 'dell@ubuntu: ~/Documents/Veille\$'. On the right, a code editor window titled 'CasSimple.c' shows the following C code:

```
10 ;;  
11  
12 struct auth *auth;  
13 char *service;  
14  
15 int main(int argc, char **argv)  
16 {  
17     char line[128];  
18  
19     while(1) {  
20         printf("[ auth = %p, service = %p ]\n", auth,  
21             service);  
22  
23         if(fgets(line, sizeof(line), stdin) == NULL) break;  
24  
25         if(strncmp(line, "auth ", 5) == 0) {  
26             auth = malloc(sizeof(auth));  
27             memset(auth, 0, sizeof(auth));  
28             if(strlen(line + 5) < 31) {  
29                 strcpy(auth->name, line + 5);  
30             }  
31  
32             if(strncmp(line, "reset", 5) == 0) {  
33                 free(auth);  
34             }  
35  
36             if(strncmp(line, "service", 6) == 0) {  
37                 service = strdup(line + 7);  
38             }  
39  
40             if(strncmp(line, "login", 5) == 0) {  
41                 if(auth->auth) {  
42                     printf("you have logged in already!\n");  
43                 } else {  
44                     printf("please enter your password\n");  
45                 }  
46             }  
47         }  
48     }  
49 }
```

The code editor window also shows a status bar at the bottom indicating 'C', 'Largeur des tabulations : 8', 'Lig 36, Col 6', and 'INS'.

# Un cas difficile

## démonstration

```
./info1 qemu:///system vagrant-vms_ubuntu-01
```

```
nom: vnet0 hwaddr:52:54:00:0e5:62[addr: 192.168.121.186 préfixe: 24 type: 0]
```

```
Domaine vagabond-vms_ubuntu-01: 2 Processeurs 3:26 PM
```

```
valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes --verbose --  
log-file=valgrind-out.txt ./info1 qemu:///system vagrant-vms_ubuntu-01
```

```
nom: vnet0 hwaddr:52:5 4:00:0e:e5:62[addr: 192.168.121.186 préfixe: 24 type:  
0]
```

```
Domaine vagabond-vms_ubuntu-01: 2 Processeurs 3:26 PM
```



# Conclusion

- Ce projet nous a appris l'importance de la faille double free une simple variable libérée deux fois peut entraîner la perturbation des trois principes fondamentaux de la sécurité la confidentialité et l'intégrité des données ainsi que la disponibilité du système.
- Le projet n'est pas abouti jusqu'à la fin pour des raisons de complexité de la faille et de matériel ainsi le manque du temps.



# Acronymes

ACRONYME	SIGNIFICATION
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
NVD	National Vulnerability Database
KVM	Kernel-based Virtual Machine
VM	Virtual Machine

# Bibliographie

- <https://access.redhat.com/security/cve/cve-2020-25637>
- <https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=double+free>
- [https://heap-exploitation.dhaval kapil.com/attacks/double\\_free](https://heap-exploitation.dhaval kapil.com/attacks/double_free)
- <https://sensepost.com/blog/2017/linux-heap-exploitation-intro-series-riding-free-on-the-heap-double-free-attacks/>
- [https://owasp.org/www-community/vulnerabilities/Doubly\\_freeing\\_memory](https://owasp.org/www-community/vulnerabilities/Doubly_freeing_memory)
- <https://www.log2base2.com/C/pointer/free-in-c.html>
- [https://owasp.org/www-community/vulnerabilities/Doubly\\_freeing\\_memory](https://owasp.org/www-community/vulnerabilities/Doubly_freeing_memory)
- <https://libvirt.org/>
- <https://www.redhat.com/en/topics/virtualization/what-is-KVM>
- [virsh commands cheatsheet to manage KVM guest virtual machines | ComputingForGeeks](#)
- [1881037 – \(CVE-2020-25637\) CVE-2020-25637 libvirt: double free in qemuAgentGetInterfaces\(\) in qemu\\_agent.c \(redhat.com\)](#)
- [lec13-HeapAttacks.pdf \(indiana.edu\)](#)
- [SensePost | Linux heap exploitation intro series: riding free on the heap – double free attacks!](#)

# Merci de votre attention

**Avez-vous des  
questions ?**