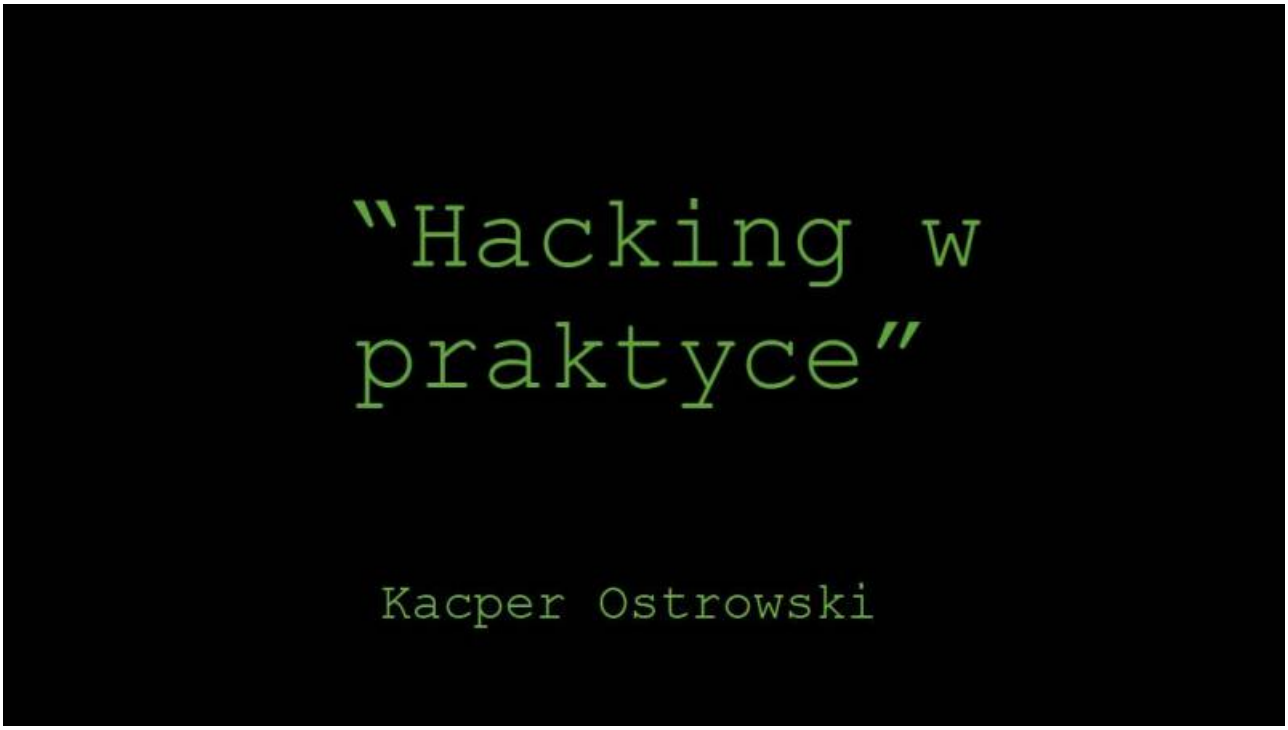
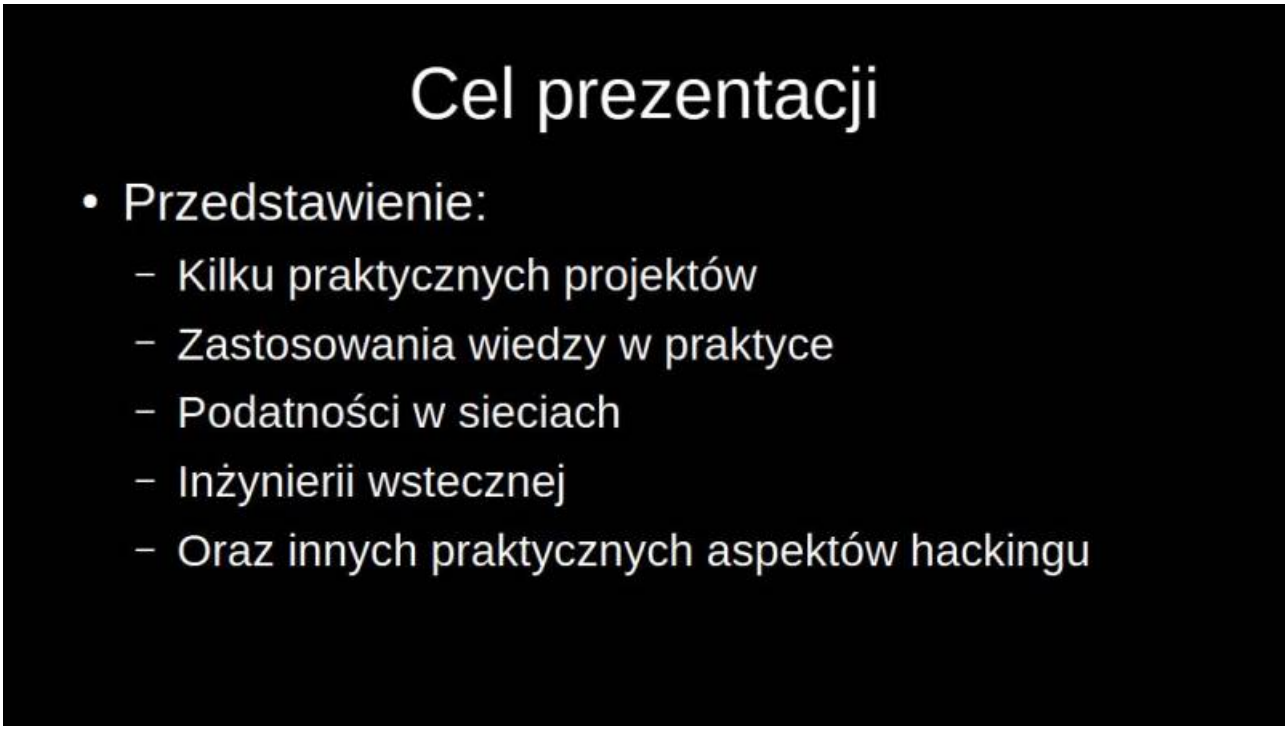
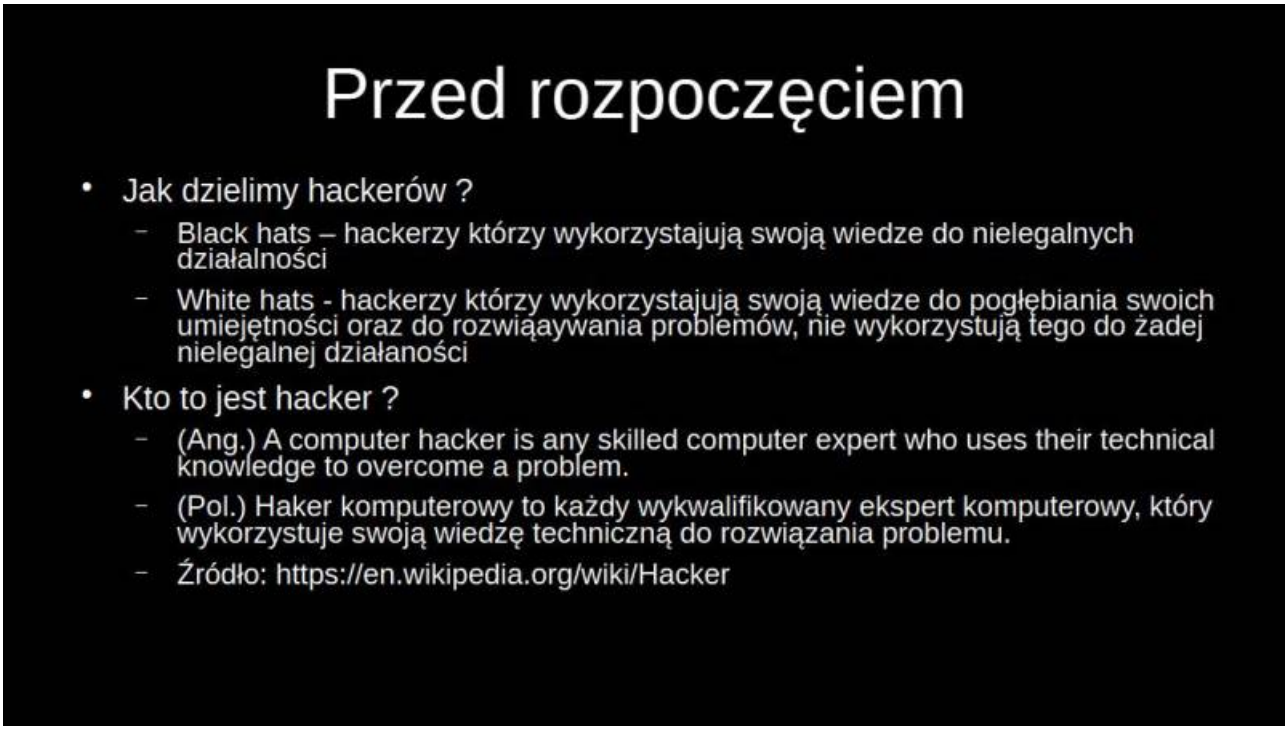
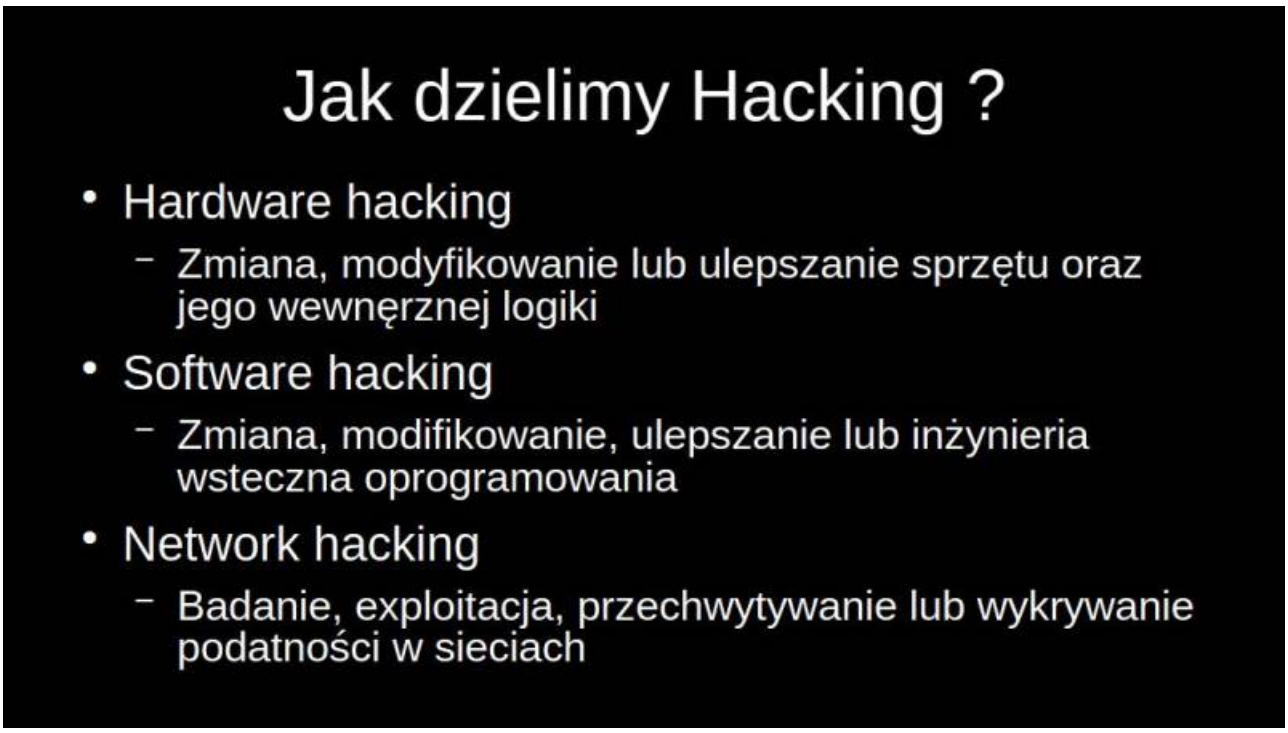
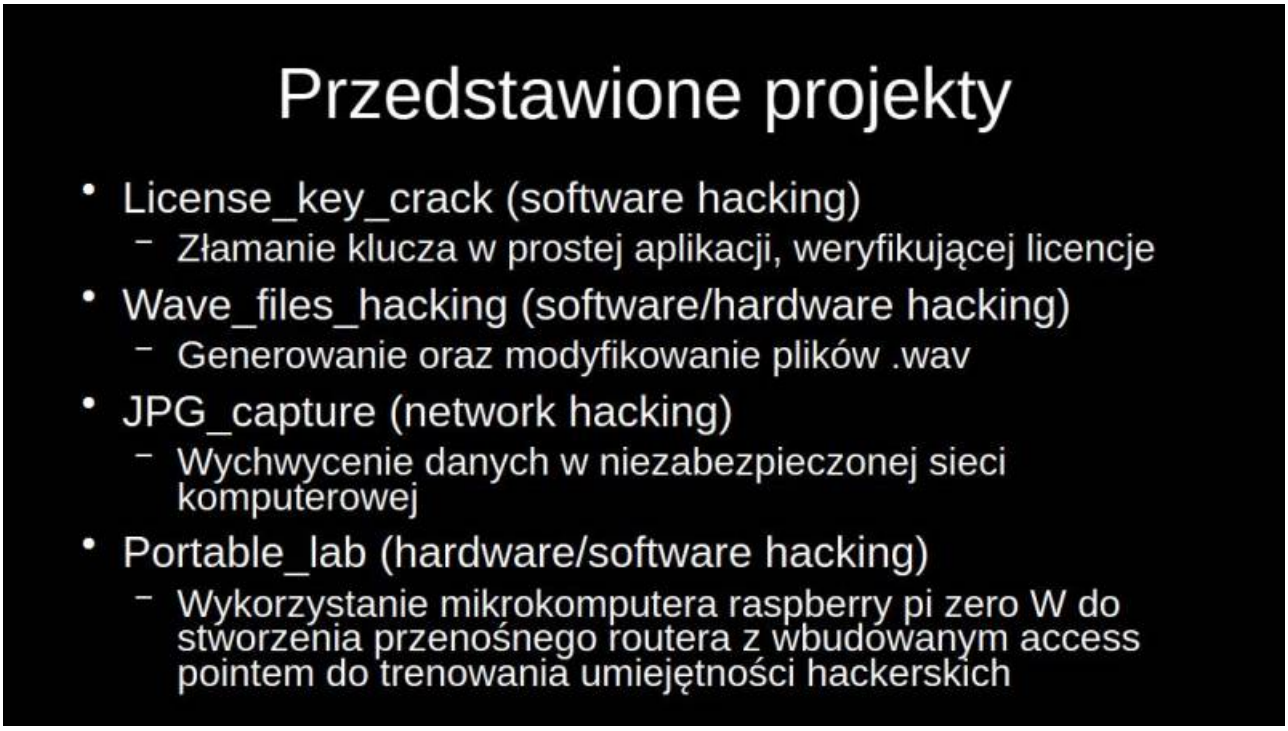
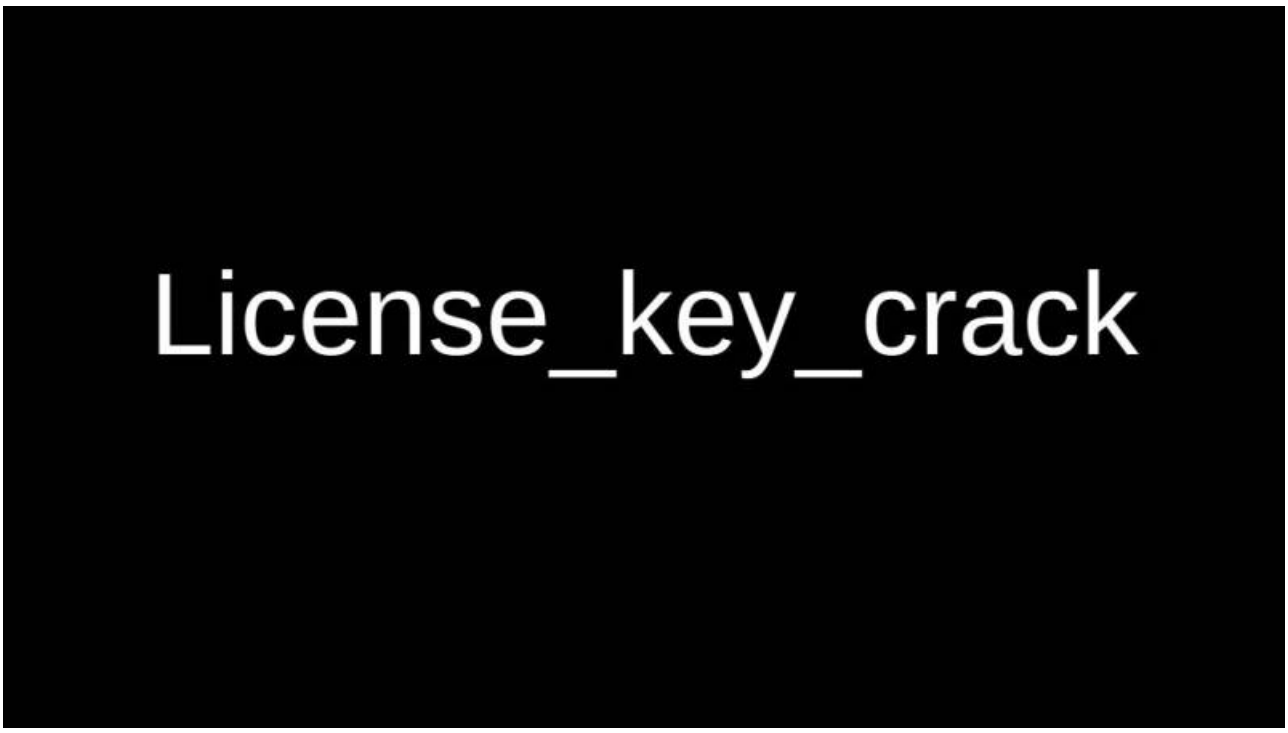
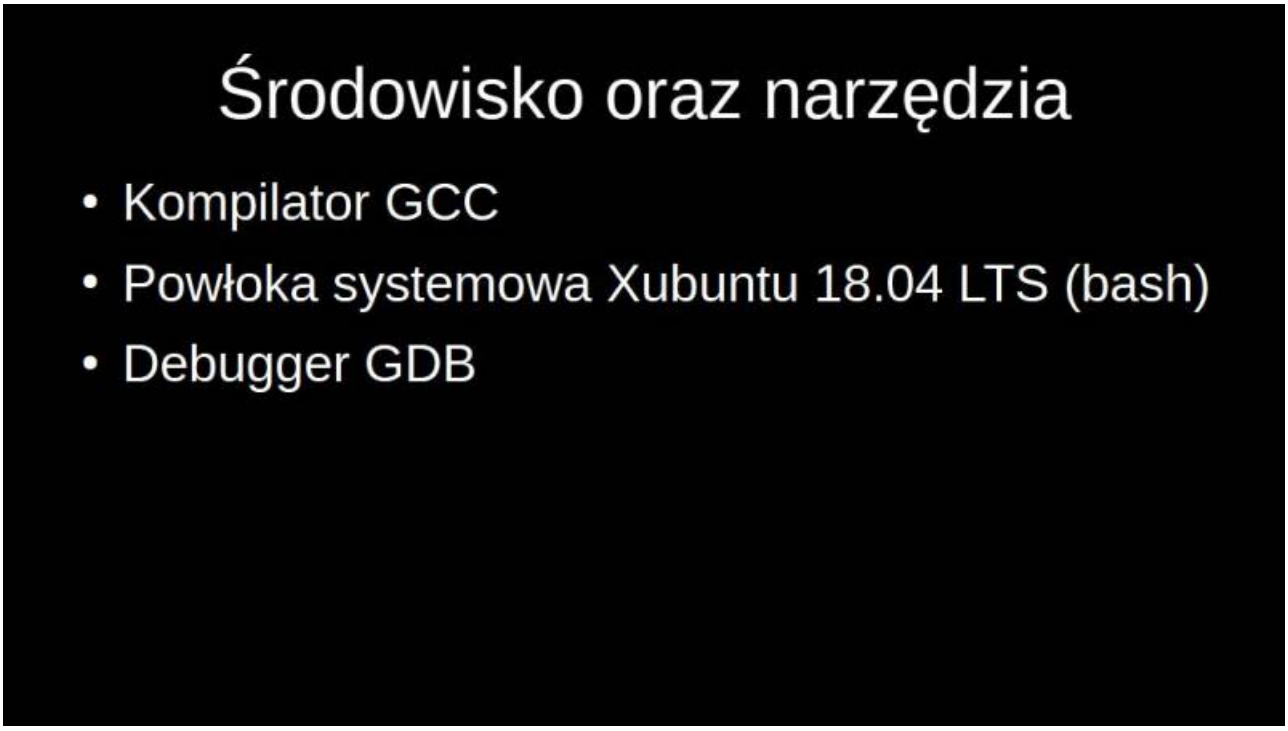
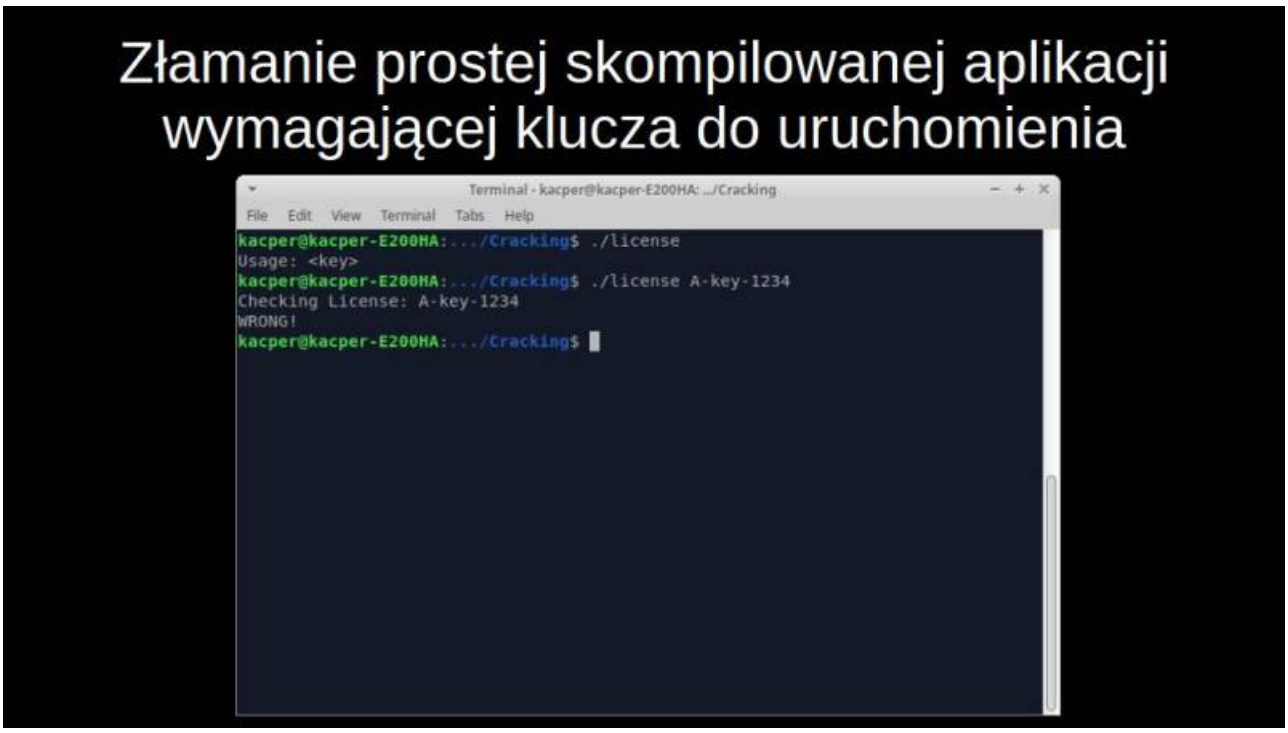
Prezentacja: Hacking w praktyce





- 
Przed rozpoczęciem
Jak dzielimy hackerów ?
Black hats – hackerzy którzy wykorzystajują swoją wiedze do nielegalnych działalności
White hats - hackerzy którzy wykorzystajują swoją wiedze do pogłębiania swoich umiejętności oraz do rozwiąaywania problemów, nie wykorzystują tego do żadej nielegalnej działaności
Kto to jest hacker ?
(Ang.) A computer hacker is any skilled computer expert who uses their technical knowledge to overcome a problem.
(Pol.) Haker komputerowy to każdy wykwalifikowany ekspert komputerowy, który wykorzystuje swoją wiedzę techniczną do rozwiązania problemu.
Źródło: https://en.wikipedia.org/wiki/Hacker
- 
Jak dzielimy Hacking ?
Hardware hacking
Zmiana, modyfikowanie lub ulepszanie sprzętu oraz jego wewnęrznej logiki
Software hacking
Zmiana, modifikowanie, ulepszanie lub inżynieria wsteczna oprogramowania
Network hacking
Badanie, exploitacja, przechwytywanie lub wykrywanie podatności w sieciach

- 
Przedstawione projekty
License_key_crack (software hacking)
Złamanie klucza w prostej aplikacji, weryfikującej licencje
Wave_files_hacking (software/hardware hacking)
Generowanie oraz modyfikowanie plików .wav
JPG_capture (network hacking)
Wychwycenie danych w niezabezpieczonej sieci komputerowej
Portable_lab (hardware/software hacking)
Wykorzystanie mikrokomputera raspberry pi zero W do stworzenia przenośnego routera z wbudowanym access pointem do trenowania umiejętności hackerskich
- 
License_key_crack

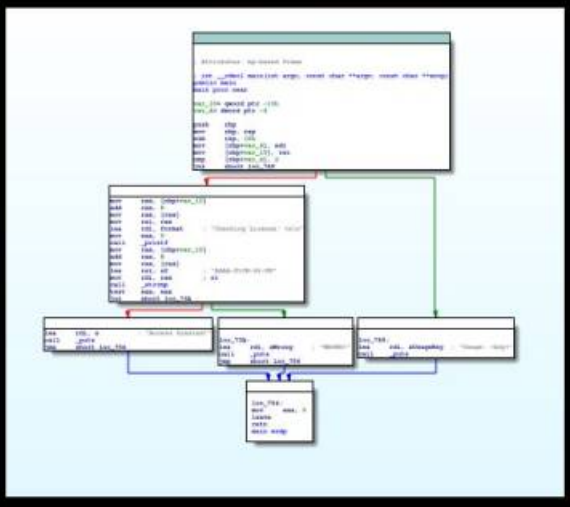
- 
Środowisko oraz narzędzia
• Kompilator GCC
• Powłoka systemowa Xubuntu 18.04 LTS (bash)
• Debugger GDB
- 
Złamanie prostej skompilowanej aplikacji wymagającej klucza do uruchomienia
Terminal - kacper@kacper-E200HA: .../Cracking
File Edit View Terminal Tabs Help
kacper@kacper-E200HA:.../Cracking$ ./license
Usage: <key>
kacper@kacper-E200HA:.../Cracking$ ./license A-key-1234
Checking License: A-key-1234
WRONG!
kacper@kacper-E200HA:.../Cracking$

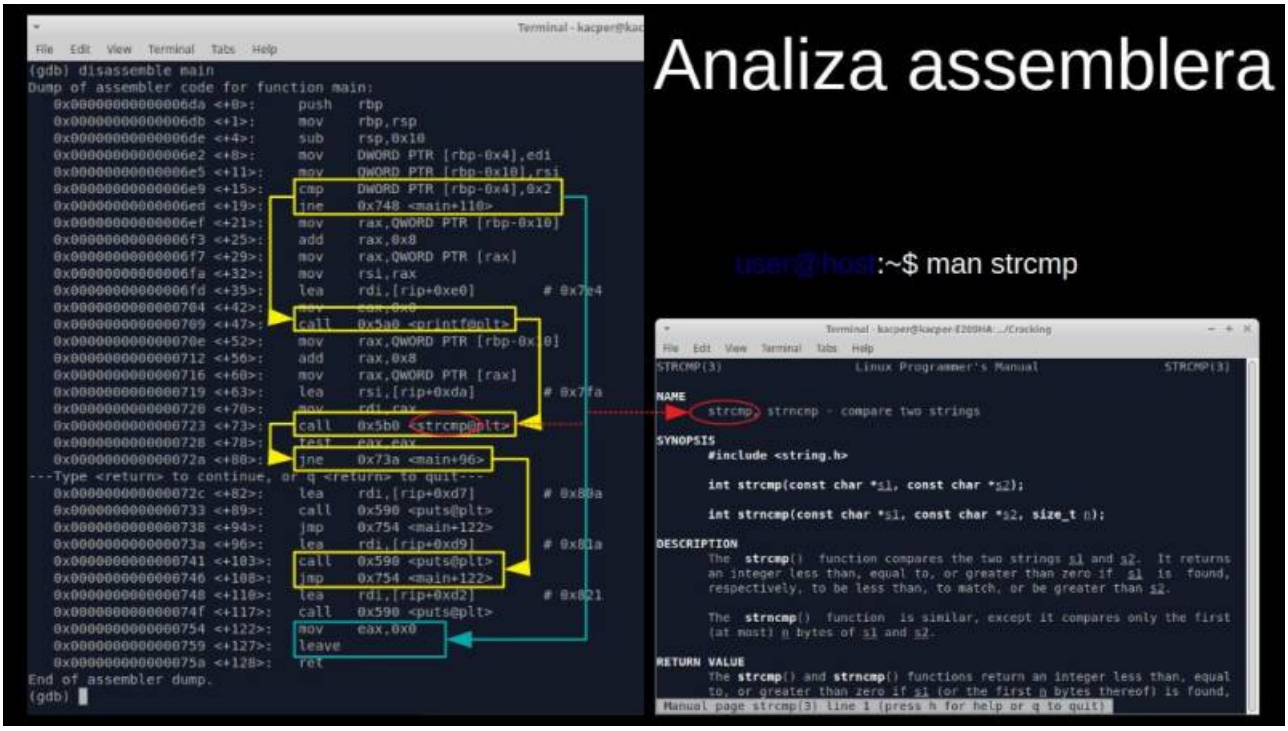
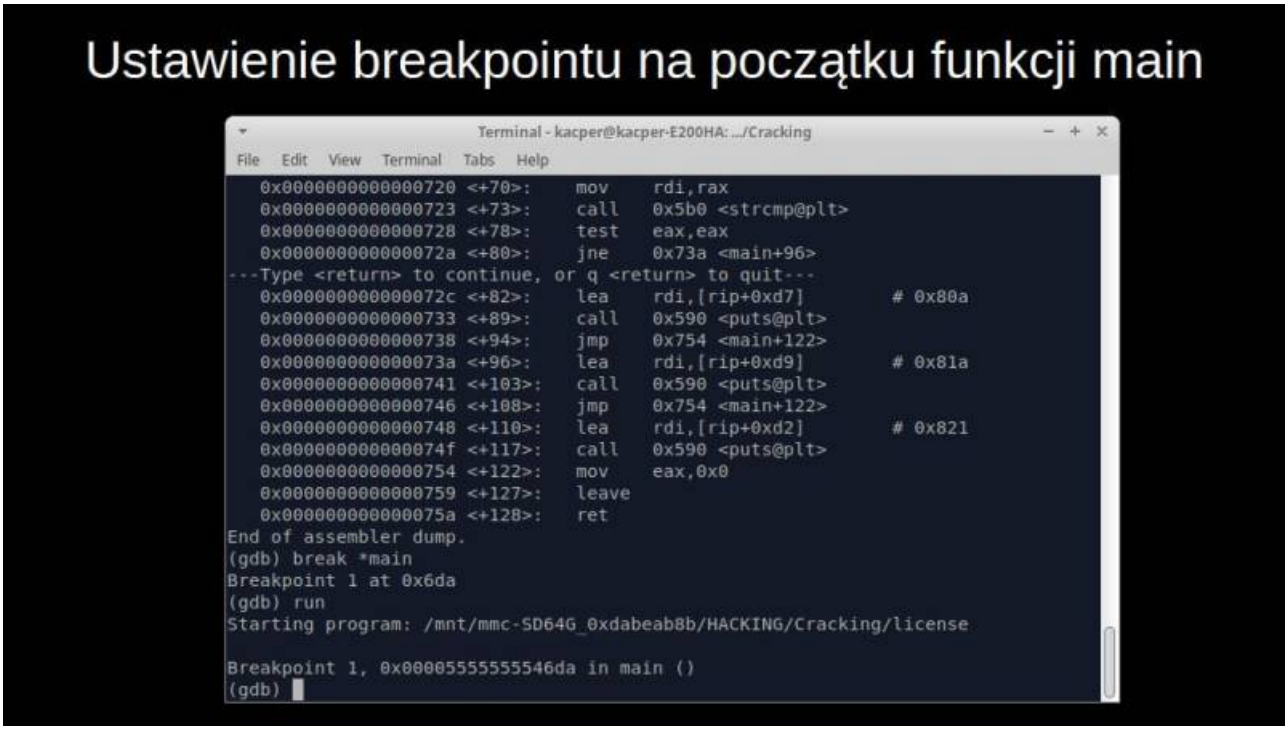
- # Kod aplikacji

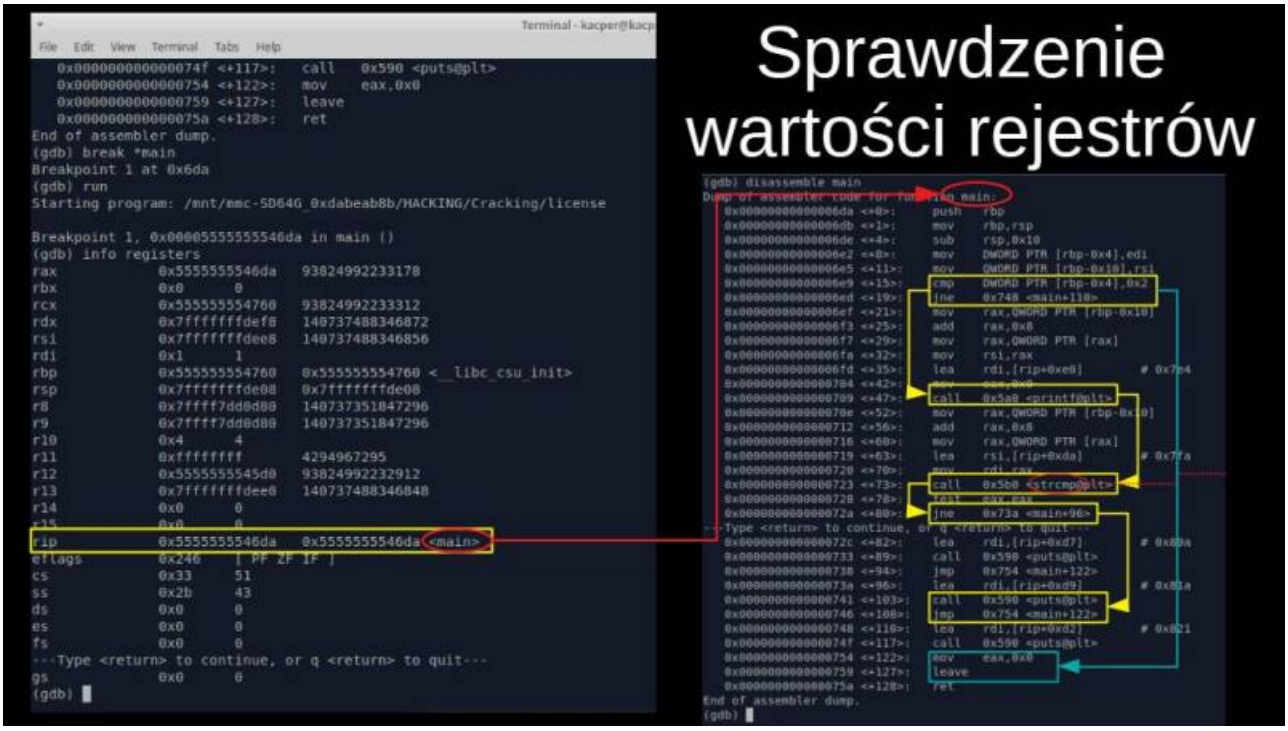
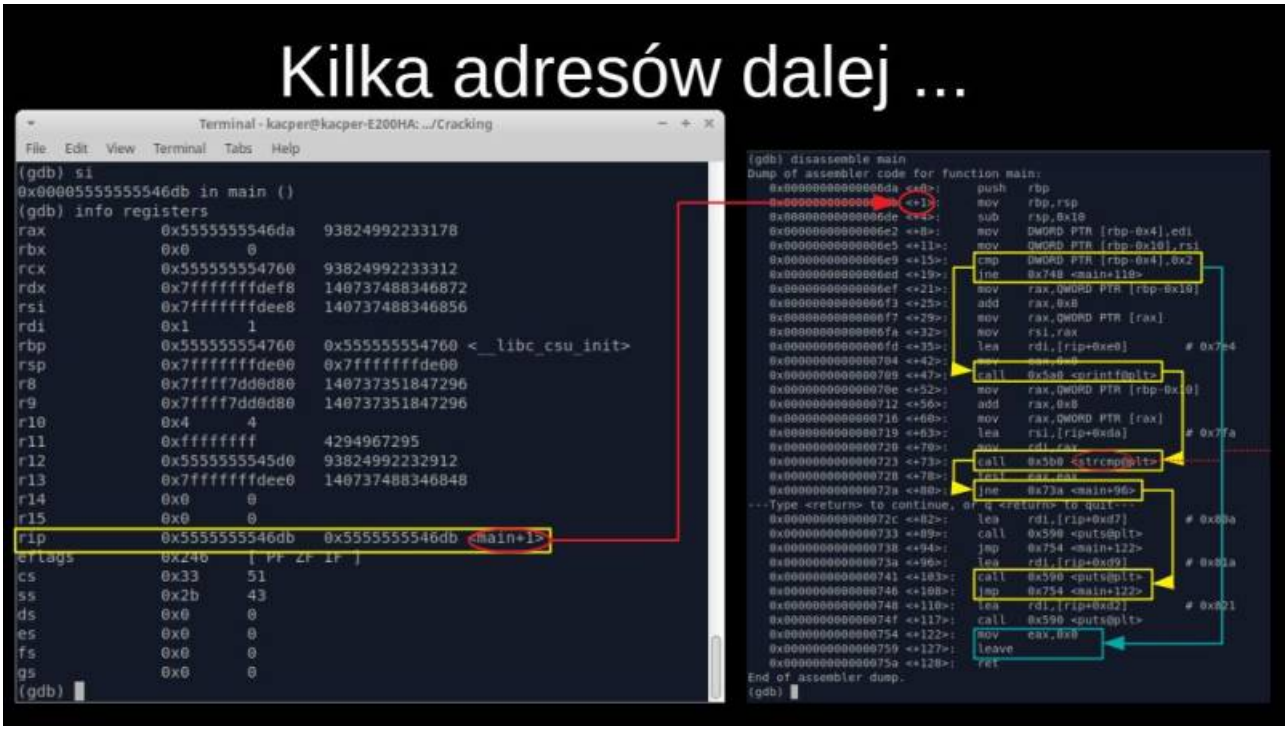
```c
#include <string.h>
#include <stdio.h>

int main(int argc, char *argv[]) {
        if(argc==2) {
        printf("Checking License: %s\n", argv[1]);
        if(strcmp(argv[1], "AAAA-Z10N-42-OK")==0) {
            printf("Access Granted!\n");
        } else {
            printf("WRONG!\n");
        }
    } else {
        printf("Usage: <key>\n");
    }
    return 0;
}
```

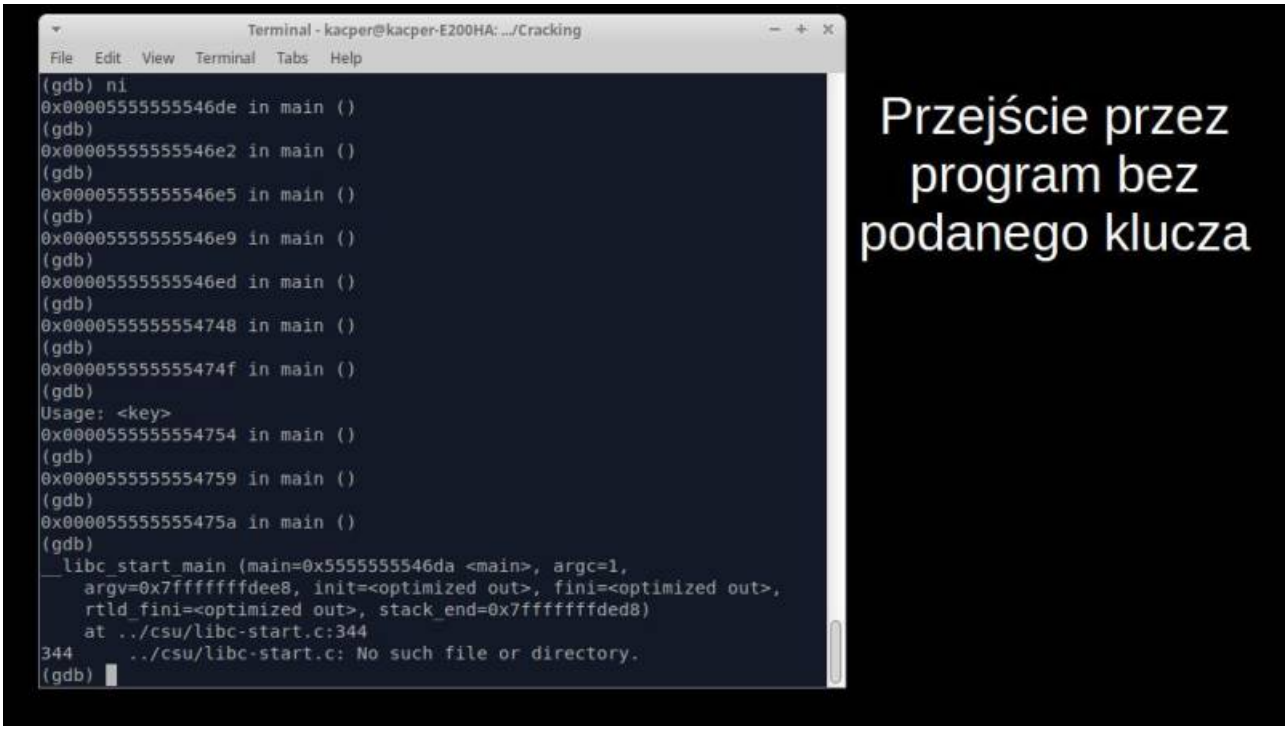
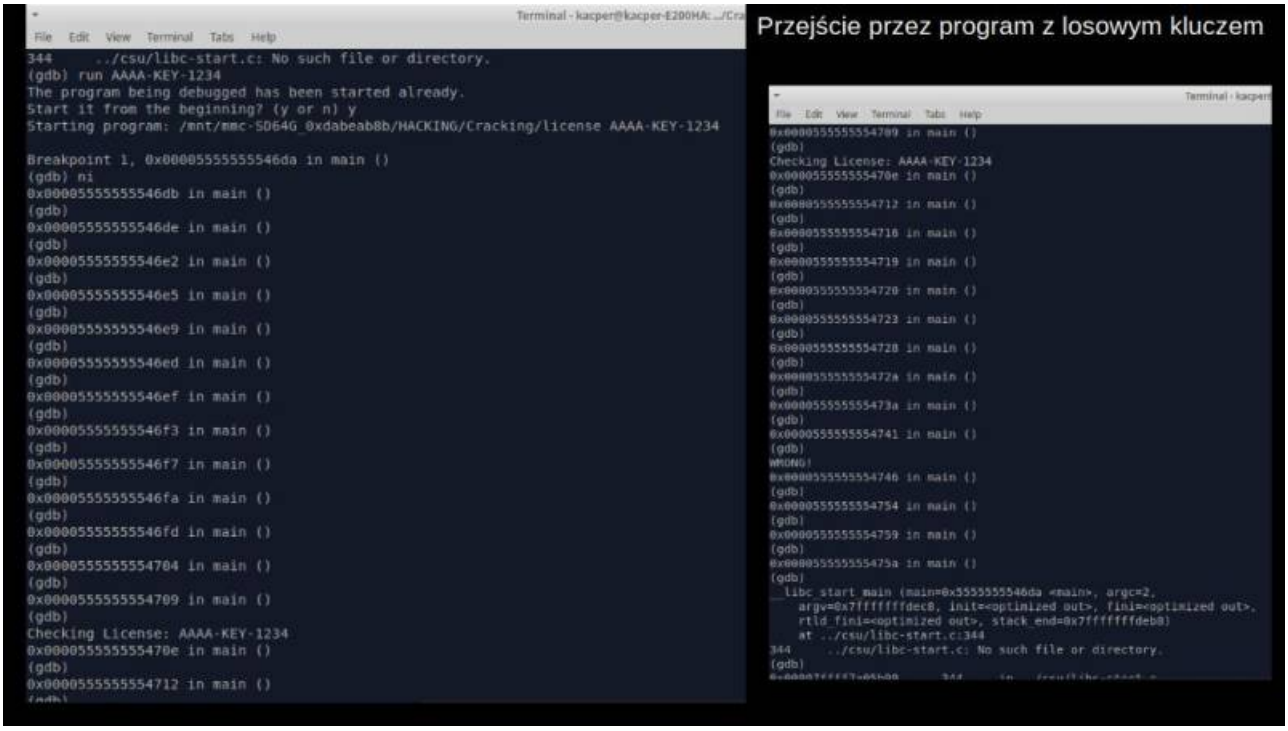
- # Deasemblacja kodu aplikacji

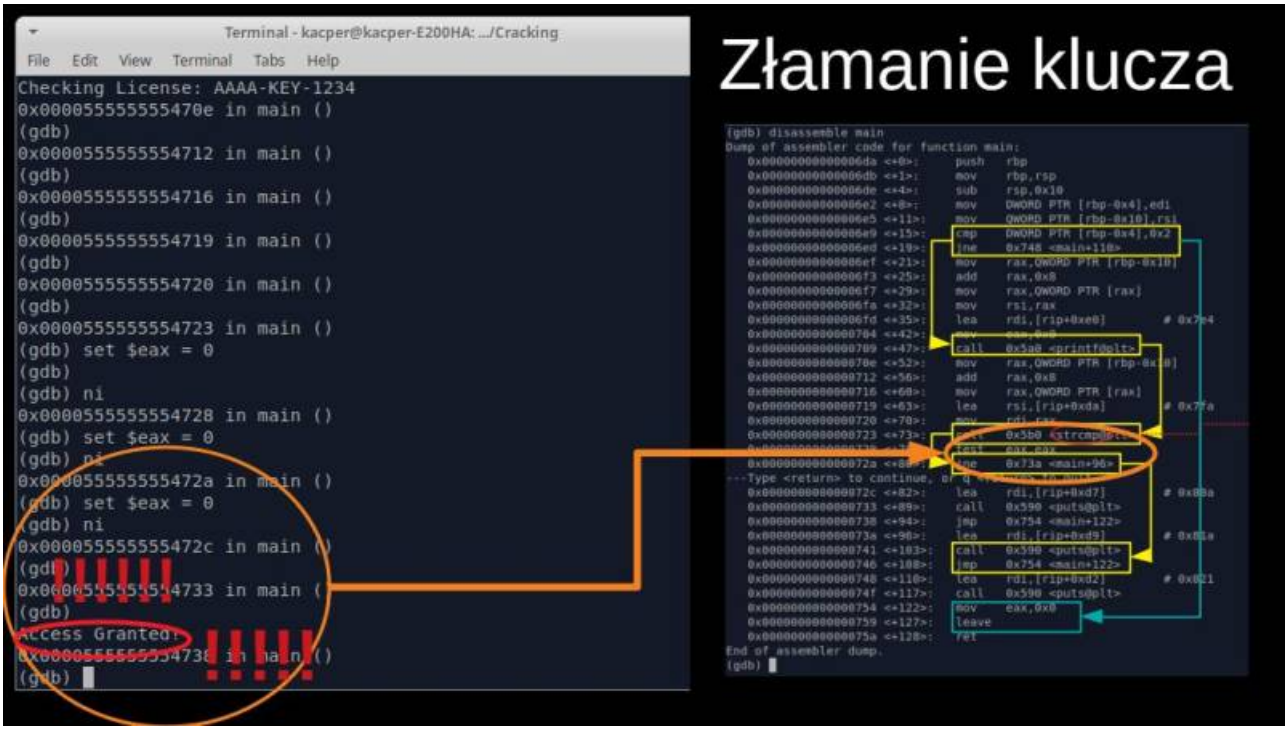
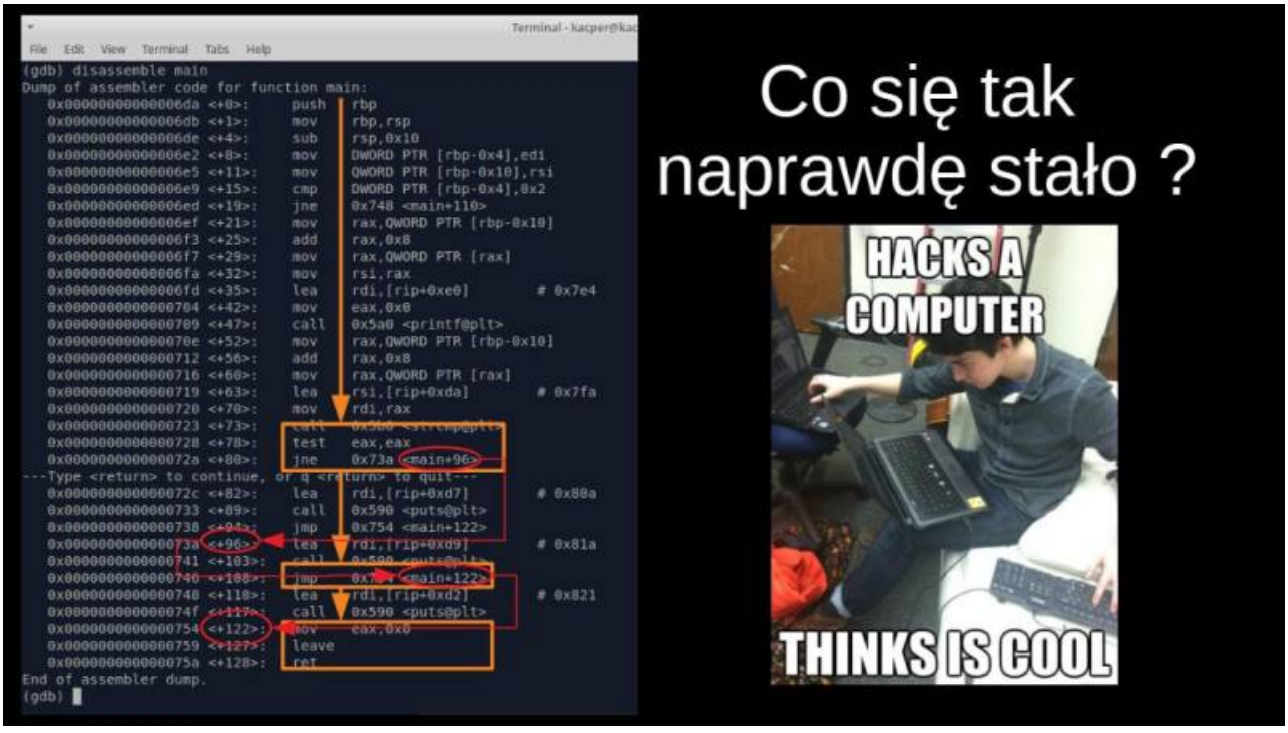
```
(gdb) disassemble main
Dump of assembler code for function main:
   0x00000000000006da <+0>:     push   %rbp
   0x00000000000006db <+1>:     mov    %rsp,%rbp
   0x00000000000006de <+4>:     sub    $0x10,%rsp
   0x00000000000006e2 <+8>:     mov    %edi,-0x4(%rbp)
   0x00000000000006e5 <+11>:    mov    %rsi,-0x10(%rbp)
   0x00000000000006e9 <+15>:    cmpl   $0x2,-0x4(%rbp)
   0x00000000000006ed <+19>:    jne    0x748 <main+110>
   0x00000000000006ef <+21>:    mov    -0x10(%rbp),%rax
   0x00000000000006f3 <+25>:    add    $0x8,%rax
   0x00000000000006f7 <+29>:    mov    (%rax),%rax
   0x00000000000006fa <+32>:    mov    %rax,%rsi
   0x00000000000006fd <+35>:    lea    0xe0(%rip),%rdi        # 0x7e4
   0x0000000000000704 <+42>:    mov    $0x0,%eax
   0x0000000000000709 <+47>:    callq  0x5a0 <printf@plt>
   0x000000000000070e <+52>:    mov    -0x10(%rbp),%rax
   0x0000000000000712 <+56>:    add    $0x8,%rax
   0x0000000000000716 <+60>:    mov    (%rax),%rax
   0x0000000000000719 <+63>:    lea    0xda(%rip),%rsi        # 0x7fa
   0x0000000000000720 <+70>:    mov    %rax,%rdi
   0x0000000000000723 <+73>:    callq  0x5b0 <strcmp@plt>
   0x0000000000000728 <+78>:    test   %eax,%eax
   0x000000000000072a <+80>:    jne    0x73a <main+96>
---Type <return> to continue, or q <return> to quit---
   0x000000000000072c <+82>:    lea    0xd7(%rip),%rdi        # 0x80a
   0x0000000000000733 <+89>:    callq  0x590 <puts@plt>
   0x0000000000000738 <+94>:    jmp    0x754 <main+122>
   0x000000000000073a <+96>:    lea    0xd9(%rip),%rdi        # 0x81a
   0x0000000000000741 <+103>:   callq  0x590 <puts@plt>
   0x0000000000000746 <+108>:   jmp    0x754 <main+122>
   0x0000000000000748 <+110>:   lea    0xd2(%rip),%rdi        # 0x821
   0x000000000000074f <+117>:   callq  0x590 <puts@plt>
   0x0000000000000754 <+122>:   mov    $0x0,%eax
   0x0000000000000759 <+127>:   leaveq
   0x000000000000075a <+128>:   retq
End of assembler dump.
(gdb)
```

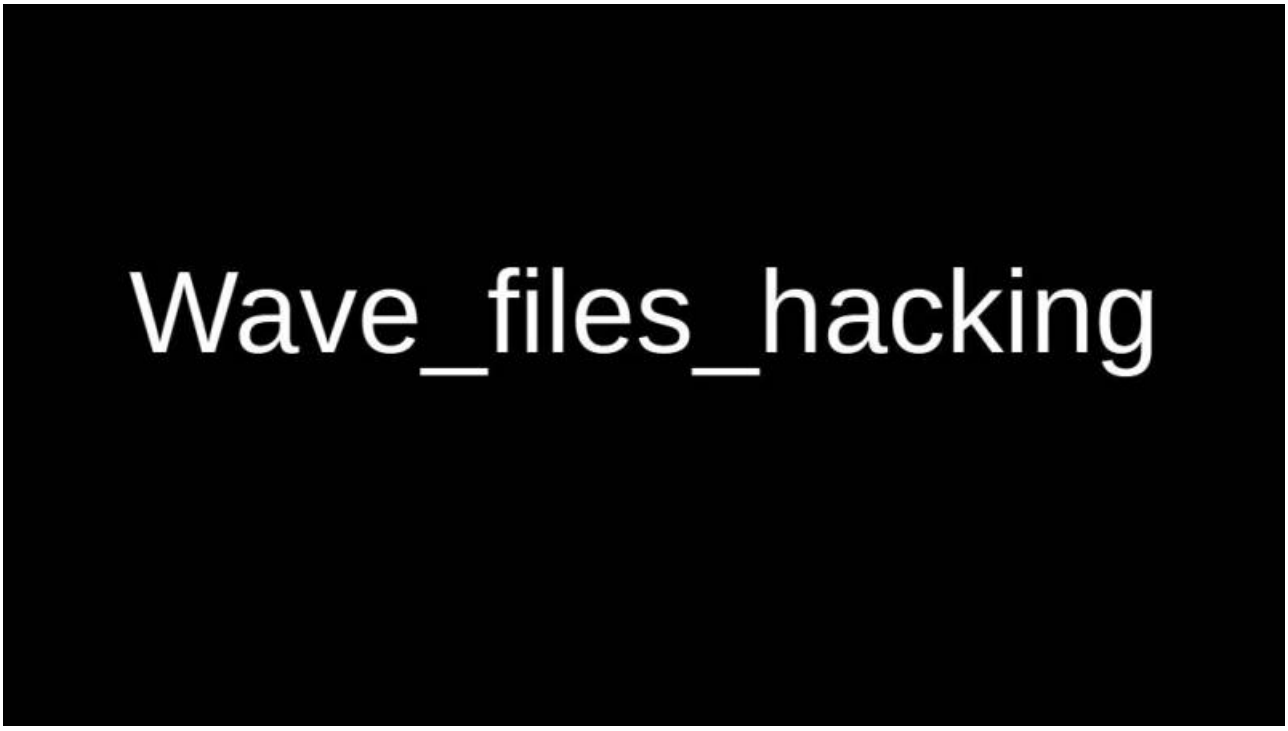
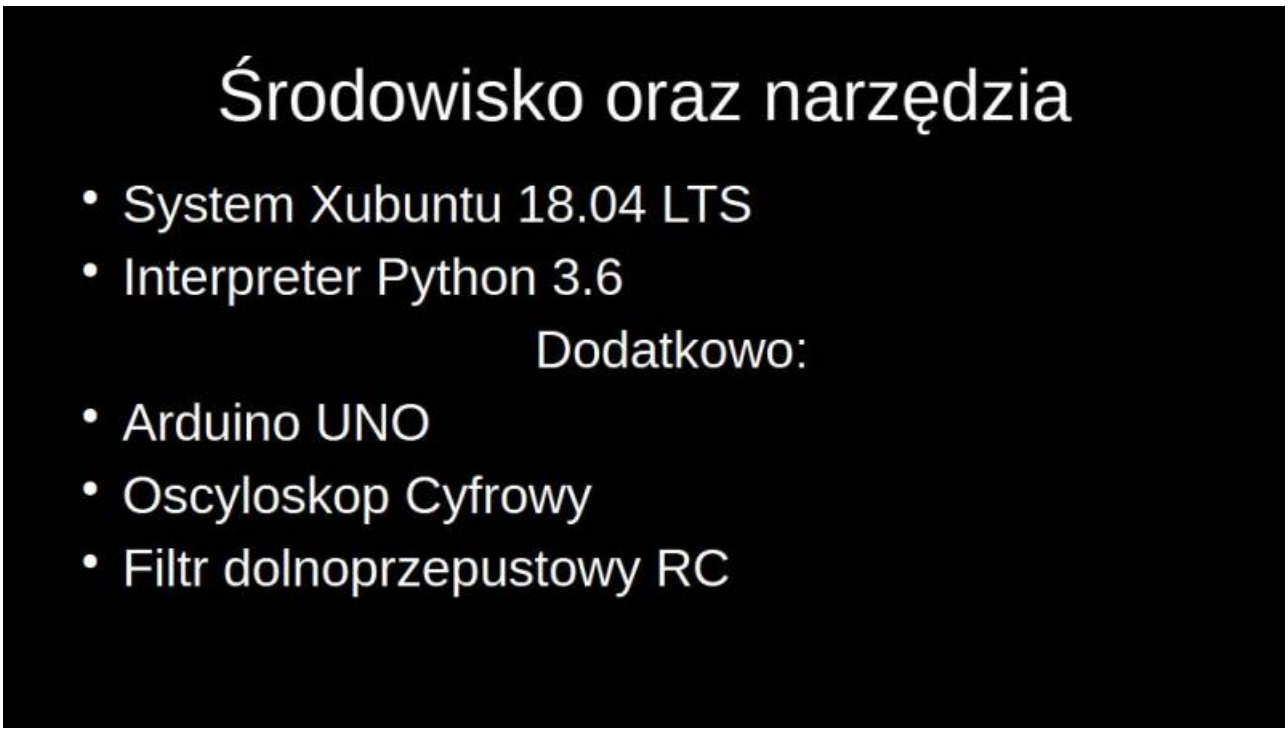


- 
Analiza assemblera
user@host:~$ man strcmp
(gdb) disassemble main
Dump of assembler code for function main:
0x00000000000006da <+0>: push rbp
0x00000000000006db <+1>: mov rbp,rsp
0x00000000000006de <+4>: sub rsp,0x10
0x00000000000006e2 <+8>: mov DWORD PTR [rbp-0x4],edi
0x00000000000006e5 <+11>: mov QWORD PTR [rbp-0x10],rsi
0x00000000000006e9 <+15>: cmp DWORD PTR [rbp-0x4],0x2
0x00000000000006ed <+19>: jne 0x748 <main+110>
0x00000000000006ef <+21>: mov rax,QWORD PTR [rbp-0x10]
0x00000000000006f3 <+25>: add rax,0x8
0x00000000000006f7 <+29>: mov rax,QWORD PTR [rax]
0x00000000000006fa <+32>: mov rsi,rax
0x00000000000006fd <+35>: lea rdi,[rip+0xe0] # 0x7e4
0x0000000000000704 <+42>: mov eax,0x0
0x0000000000000709 <+47>: call 0x5a0 <printf@plt>
0x000000000000070e <+52>: mov rax,QWORD PTR [rbp-0x10]
0x0000000000000712 <+56>: add rax,0x8
0x0000000000000716 <+60>: mov rax,QWORD PTR [rax]
0x0000000000000719 <+63>: lea rsi,[rip+0xda] # 0x7fa
0x0000000000000720 <+70>: mov rdi,rax
0x0000000000000723 <+73>: call 0x5b0 <strcmp@plt>
0x0000000000000728 <+78>: test eax,eax
0x000000000000072a <+80>: jne 0x73a <main+96>
---Type <return> to continue, or q <return> to quit---
0x000000000000072c <+82>: lea rdi,[rip+0xd7] # 0x80a
0x0000000000000733 <+89>: call 0x590 <puts@plt>
0x0000000000000738 <+94>: jmp 0x754 <main+122>
0x000000000000073a <+96>: lea rdi,[rip+0xd9] # 0x81a
0x0000000000000741 <+103>: call 0x590 <puts@plt>
0x0000000000000746 <+108>: jmp 0x754 <main+122>
0x0000000000000748 <+110>: lea rdi,[rip+0xd2] # 0x821
0x000000000000074f <+117>: call 0x590 <puts@plt>
0x0000000000000754 <+122>: mov eax,0x0
0x0000000000000759 <+127>: leave
0x000000000000075a <+128>: ret
End of assembler dump.
(gdb)
STRCMP(3) Linux Programmer's Manual STRCMP(3)
NAME
strcmp, strncmp - compare two strings
SYNOPSIS
#include <string.h>
int strcmp(const char *s1, const char *s2);
int strncmp(const char *s1, const char *s2, size_t n);
DESCRIPTION
The strcmp() function compares the two strings s1 and s2. It returns an integer less than, equal to, or greater than zero if s1 is found, respectively, to be less than, to match, or be greater than s2.
The strncmp() function is similar, except it compares only the first (at most) n bytes of s1 and s2.
RETURN VALUE
The strcmp() and strncmp() functions return an integer less than, equal to, or greater than zero if s1 (or the first n bytes thereof) is found,
Manual page strcmp(3) line 1 (press h for help or q to quit)
- 
Ustawienie breakpointu na początku funkcji main
Terminal - kacper@kacper-E200HA: .../Cracking
File Edit View Terminal Tabs Help
0x0000000000000720 <+70>: mov rdi,rax
0x0000000000000723 <+73>: call 0x5b0 <strcmp@plt>
0x0000000000000728 <+78>: test eax,eax
0x000000000000072a <+80>: jne 0x73a <main+96>
---Type <return> to continue, or q <return> to quit---
0x000000000000072c <+82>: lea rdi,[rip+0xd7] # 0x80a
0x0000000000000733 <+89>: call 0x590 <puts@plt>
0x0000000000000738 <+94>: jmp 0x754 <main+122>
0x000000000000073a <+96>: lea rdi,[rip+0xd9] # 0x81a
0x0000000000000741 <+103>: call 0x590 <puts@plt>
0x0000000000000746 <+108>: jmp 0x754 <main+122>
0x0000000000000748 <+110>: lea rdi,[rip+0xd2] # 0x821
0x000000000000074f <+117>: call 0x590 <puts@plt>
0x0000000000000754 <+122>: mov eax,0x0
0x0000000000000759 <+127>: leave
0x000000000000075a <+128>: ret
End of assembler dump.
(gdb) break *main
Breakpoint 1 at 0x6da
(gdb) run
Starting program: /mnt/mmc-SD64G_0xdabeab8b/HACKING/Cracking/license
Breakpoint 1, 0x00005555555546da in main ()
(gdb)

- 
Sprawdzenie wartości rejestrów
Terminal - kacper@kacp
File Edit View Terminal Tabs Help
0x000000000000074f <+117>: call 0x590 <puts@plt>
0x0000000000000754 <+122>: mov eax,0x0
0x0000000000000759 <+127>: leave
0x000000000000075a <+128>: ret
End of assembler dump.
(gdb) break *main
Breakpoint 1 at 0x6da
(gdb) run
Starting program: /mnt/mmc-SD64G_0xdabeab8b/HACKING/Cracking/license
Breakpoint 1, 0x00005555555546da in main ()
(gdb) info registers
rax 0x5555555546da 93824992233178
rbx 0x0 0
rcx 0x555555554760 93824992233312
rdx 0x7fffffffdef8 140737488346872
rsi 0x7fffffffdee8 140737488346856
rdi 0x1 1
rbp 0x555555554760 0x555555554760 <__libc_csu_init>
rsp 0x7fffffffde08 0x7fffffffde08
r8 0x7ffff7dd0d80 140737351847296
r9 0x7ffff7dd0d80 140737351847296
r10 0x4 4
r11 0xffffffff 4294967295
r12 0x5555555545d0 93824992232912
r13 0x7fffffffdee0 140737488346848
r14 0x0 0
r15 0x0 0
rip 0x5555555546da 0x5555555546da <main>
eflags 0x246 [PF ZF IF]
cs 0x33 51
ss 0x2b 43
ds 0x0 0
es 0x0 0
fs 0x0 0
---Type <return> to continue, or q <return> to quit---
gs 0x0 0
(gdb)
(gdb) disassemble main
Dump of assembler code for function main:
0x00005555555546da <+0>: push rbp
0x00005555555546db <+1>: mov rbp,rsp
0x00005555555546de <+4>: sub rsp,0x10
0x00005555555546e2 <+8>: mov DWORD PTR [rbp-0x4],edi
0x00005555555546e5 <+11>: mov QWORD PTR [rbp-0x10],rsi
0x00005555555546e9 <+15>: cmp DWORD PTR [rbp-0x4],0x2
0x00005555555546ed <+19>: jne 0x5748 <main+110>
0x00005555555546ef <+21>: mov rax,QWORD PTR [rbp-0x10]
0x00005555555546f3 <+25>: add rax,0x8
0x00005555555546f7 <+29>: mov rax,QWORD PTR [rax]
0x00005555555546fa <+32>: mov rsi,rax
0x00005555555546fd <+35>: lea rdi,[rip+0xe0] # 0x7
0x0000555555554704 <+42>: mov eax,0x0
0x0000555555554709 <+47>: call 0x55a0 <printf@plt>
0x000055555555470e <+52>: mov rax,QWORD PTR [rbp-0x
0x0000555555554712 <+56>: add rax,0x8
0x0000555555554716 <+60>: mov rax,QWORD PTR [rax]
0x0000555555554719 <+63>: lea rsi,[rip+0xda] # 0x7
0x0000555555554720 <+70>: mov rdi,rax
0x0000555555554723 <+73>: call 0x55b0 <strcmp@plt>
0x0000555555554728 <+78>: test eax,eax
0x000055555555472a <+80>: jne 0x573a <main+96>
Type <return> to continue, or q <return> to quit---
0x000055555555472c <+82>: lea rdi,[rip+0xd7] # 0x80a
0x0000555555554733 <+89>: call 0x590 <puts@plt>
0x0000555555554738 <+94>: jmp 0x5754 <main+122>
0x000055555555473a <+96>: lea rdi,[rip+0xd9] # 0x81a
0x0000555555554741 <+103>: call 0x590 <puts@plt>
0x0000555555554746 <+108>: jmp 0x5754 <main+122>
0x0000555555554748 <+110>: lea rdi,[rip+0xd2] # 0x821
0x000055555555474f <+117>: call 0x590 <puts@plt>
0x0000555555554754 <+122>: mov eax,0x0
0x0000555555554759 <+127>: leave
0x000055555555475a <+128>: ret
End of assembler dump.
(gdb)
- 
Kilka adresów dalej ...
Terminal - kacper@kacper-E200HA: .../Cracking
File Edit View Terminal Tabs Help
(gdb) si
0x00005555555546db in main ()
(gdb) info registers
rax 0x5555555546da 93824992233178
rbx 0x0 0
rcx 0x555555554760 93824992233312
rdx 0x7fffffffdef8 140737488346872
rsi 0x7fffffffdee8 140737488346856
rdi 0x1 1
rbp 0x555555554760 0x555555554760 <__libc_csu_init>
rsp 0x7fffffffde00 0x7fffffffde00
r8 0x7ffff7dd0d80 140737351847296
r9 0x7ffff7dd0d80 140737351847296
r10 0x4 4
r11 0xffffffff 4294967295
r12 0x5555555545d0 93824992232912
r13 0x7fffffffdee0 140737488346848
r14 0x0 0
r15 0x0 0
rip 0x5555555546db 0x5555555546db <main+1>
eflags 0x246 [PF ZF IF]
cs 0x33 51
ss 0x2b 43
ds 0x0 0
es 0x0 0
fs 0x0 0
gs 0x0 0
(gdb)

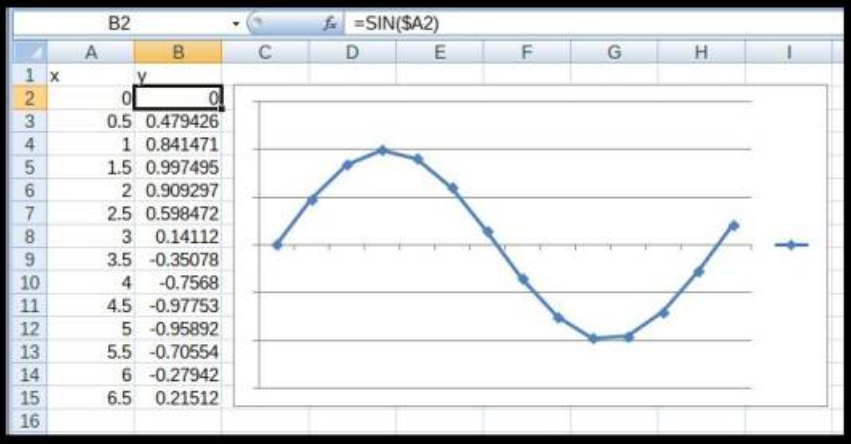
- 
Terminal - kacper@kacper-E200HA: .../Cracking
File Edit View Terminal Tabs Help
(gdb) ni
0x00005555555546de in main ()
(gdb)
0x00005555555546e2 in main ()
(gdb)
0x00005555555546e5 in main ()
(gdb)
0x00005555555546e9 in main ()
(gdb)
0x00005555555546ed in main ()
(gdb)
0x0000555555554748 in main ()
(gdb)
0x000055555555474f in main ()
(gdb)
Usage: <key>
0x0000555555554754 in main ()
(gdb)
0x0000555555554759 in main ()
(gdb)
0x000055555555475a in main ()
(gdb)
__libc_start_main (main=0x5555555546da <main>, argc=1,
argv=0x7fffffffdee8, init=<optimized out>, fini=<optimized out>,
rtld_fini=<optimized out>, stack_end=0x7fffffffded8)
at ../csu/libc-start.c:344
344 ../csu/libc-start.c: No such file or directory.
(gdb)
Przejście przez program bez podanego klucza
- 
Terminal - kacper@kacper-E200HA: .../Cra
File Edit View Terminal Tabs Help
344 ../csu/libc-start.c: No such file or directory.
(gdb) run AAAA-KEY-1234
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /mnt/mmc-SD64G_0xdabeab8b/HACKING/Cracking/license AAAA-KEY-1234
Breakpoint 1, 0x00005555555546da in main ()
(gdb) ni
0x00005555555546db in main ()
(gdb)
0x00005555555546de in main ()
(gdb)
0x00005555555546e2 in main ()
(gdb)
0x00005555555546e5 in main ()
(gdb)
0x00005555555546e9 in main ()
(gdb)
0x00005555555546ed in main ()
(gdb)
0x00005555555546ef in main ()
(gdb)
0x00005555555546f3 in main ()
(gdb)
0x00005555555546f7 in main ()
(gdb)
0x00005555555546fa in main ()
(gdb)
0x00005555555546fd in main ()
(gdb)
0x0000555555554704 in main ()
(gdb)
0x0000555555554709 in main ()
(gdb)
Checking License: AAAA-KEY-1234
0x000055555555470e in main ()
(gdb)
0x0000555555554712 in main ()
Przejście przez program z losowym kluczem

- 
Terminal - kacper@kacper-E200HA: .../Cracking
File Edit View Terminal Tabs Help
Checking License: AAAA-KEY-1234
0x000055555555470e in main ()
(gdb)
0x0000555555554712 in main ()
(gdb)
0x0000555555554716 in main ()
(gdb)
0x0000555555554719 in main ()
(gdb)
0x0000555555554720 in main ()
(gdb)
0x0000555555554723 in main ()
(gdb) set $eax = 0
(gdb)
(gdb) ni
0x0000555555554728 in main ()
(gdb) set $eax = 0
(gdb) ni
0x000055555555472a in main ()
(gdb) set $eax = 0
(gdb) ni
0x000055555555472c in main ()
(gdb)
0x0000555555554733 in main ()
(gdb)
Access Granted!
(gdb)
Złamanie klucza
- 
(gdb) disassemble main
Dump of assembler code for function main:
0x00000000000006da <+0>: push rbp
0x00000000000006db <+1>: mov rbp,rsp
0x00000000000006de <+4>: sub rsp,0x10
0x00000000000006e2 <+8>: mov DWORD PTR [rbp-0x4],edi
0x00000000000006e5 <+11>: mov QWORD PTR [rbp-0x10],rsi
0x00000000000006e9 <+15>: cmp DWORD PTR [rbp-0x4],0x2
0x00000000000006ed <+19>: jne 0x748 <main+110>
0x00000000000006ef <+21>: mov rax,QWORD PTR [rbp-0x10]
0x00000000000006f3 <+25>: add rax,0x8
0x00000000000006f7 <+29>: mov rax,QWORD PTR [rax]
0x00000000000006fa <+32>: mov rsi,rax
0x00000000000006fd <+35>: lea rdi,[rip+0xe0] # 0x7e4
0x0000000000000704 <+42>: mov eax,0x0
0x0000000000000709 <+47>: call 0x5a0 <printf@plt>
0x000000000000070e <+52>: mov rax,QWORD PTR [rbp-0x10]
0x0000000000000712 <+56>: add rax,0x8
0x0000000000000716 <+60>: mov rax,QWORD PTR [rax]
0x0000000000000719 <+63>: lea rsi,[rip+0xda] # 0x7fa
0x0000000000000720 <+70>: mov rdi,rax
0x0000000000000723 <+73>: call 0x5b0 <strcmp@plt>
0x0000000000000728 <+78>: test eax,eax
0x000000000000072a <+80>: jne 0x73a <main+96>
---Type <return> to continue, or q <return> to quit---
0x000000000000072c <+82>: lea rdi,[rip+0xd7] # 0x80a
0x0000000000000733 <+89>: call 0x590 <puts@plt>
0x0000000000000738 <+94>: jmp 0x754 <main+122>
0x000000000000073a <+96>: lea rdi,[rip+0xd9] # 0x81a
0x0000000000000741 <+103>: call 0x590 <puts@plt>
0x0000000000000746 <+108>: jmp 0x754 <main+122>
0x0000000000000748 <+110>: lea rdi,[rip+0xd2] # 0x821
0x000000000000074f <+117>: call 0x590 <puts@plt>
0x0000000000000754 <+122>: mov eax,0x0
0x0000000000000759 <+127>: leave
0x000000000000075a <+128>: ret
End of assembler dump.
(gdb)
-
Terminal - kacper@kac
File Edit View Terminal Tabs Help
(gdb) disassemble main
Dump of assembler code for function main:
0x00000000000006da <+0>: push rbp
0x00000000000006db <+1>: mov rbp,rsp
0x00000000000006de <+4>: sub rsp,0x10
0x00000000000006e2 <+8>: mov DWORD PTR [rbp-0x4],edi
0x00000000000006e5 <+11>: mov QWORD PTR [rbp-0x10],rsi
0x00000000000006e9 <+15>: cmp DWORD PTR [rbp-0x4],0x2
0x00000000000006ed <+19>: jne 0x748 <main+110>
0x00000000000006ef <+21>: mov rax,QWORD PTR [rbp-0x10]
0x00000000000006f3 <+25>: add rax,0x8
0x00000000000006f7 <+29>: mov rax,QWORD PTR [rax]
0x00000000000006fa <+32>: mov rsi,rax
0x00000000000006fd <+35>: lea rdi,[rip+0xe0] # 0x7e4
0x0000000000000704 <+42>: mov eax,0x0
0x0000000000000709 <+47>: call 0x5a0 <printf@plt>
0x000000000000070e <+52>: mov rax,QWORD PTR [rbp-0x10]
0x0000000000000712 <+56>: add rax,0x8
0x0000000000000716 <+60>: mov rax,QWORD PTR [rax]
0x0000000000000719 <+63>: lea rsi,[rip+0xda] # 0x7fa
0x0000000000000720 <+70>: mov rdi,rax
0x0000000000000723 <+73>: call 0x5b0 <strcmp@plt>
0x0000000000000728 <+78>: test eax,eax
0x000000000000072a <+80>: jne 0x73a <main+96>
---Type <return> to continue, or q <return> to quit---
0x000000000000072c <+82>: lea rdi,[rip+0xd7] # 0x80a
0x0000000000000733 <+89>: call 0x590 <puts@plt>
0x0000000000000738 <+94>: jmp 0x754 <main+122>
0x000000000000073a <+96>: lea rdi,[rip+0xd9] # 0x81a
0x0000000000000741 <+103>: call 0x590 <puts@plt>
0x0000000000000746 <+108>: jmp 0x754 <main+122>
0x0000000000000748 <+110>: lea rdi,[rip+0xd2] # 0x821
0x000000000000074f <+117>: call 0x590 <puts@plt>
0x0000000000000754 <+122>: mov eax,0x0
0x0000000000000759 <+127>: leave
0x000000000000075a <+128>: ret
End of assembler dump.
(gdb)
Co się tak naprawdę stało ?
HACKS A COMPUTER
THINKS IS COOL

- 
Wave_files_hacking
- 
Środowisko oraz narzędzia
System Xubuntu 18.04 LTS
Interpreter Python 3.6
Dodatkowo:
Arduino UNO
Oscyloskop Cyfrowy
Filtr dolnoprzepustowy RC

Parametry pliku .wav

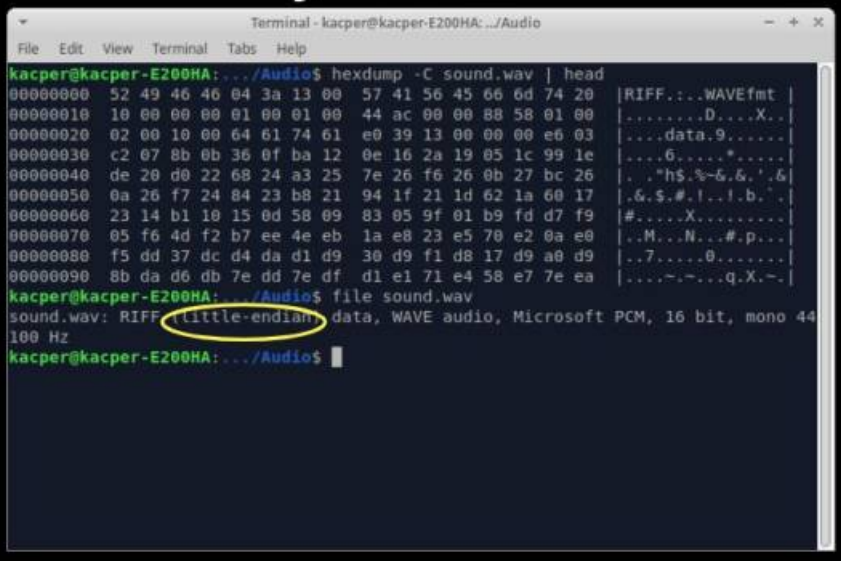
- Bit rate
- Bit depth

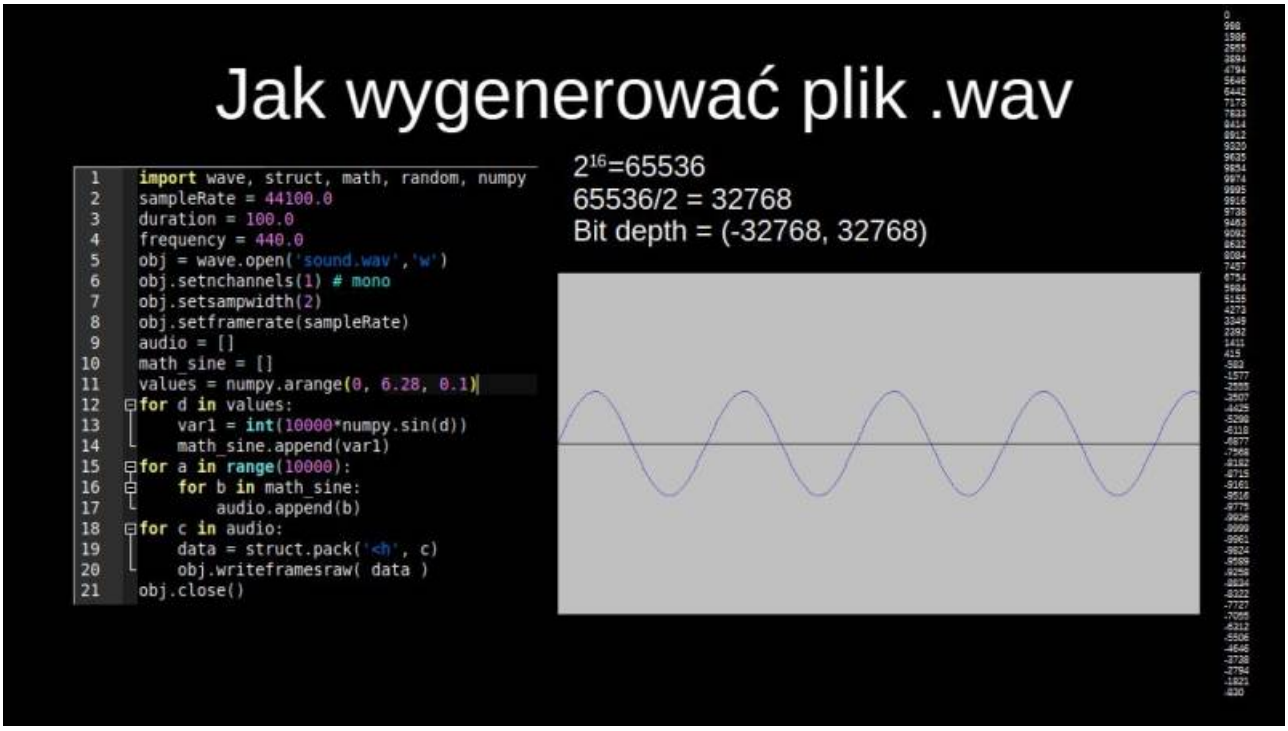
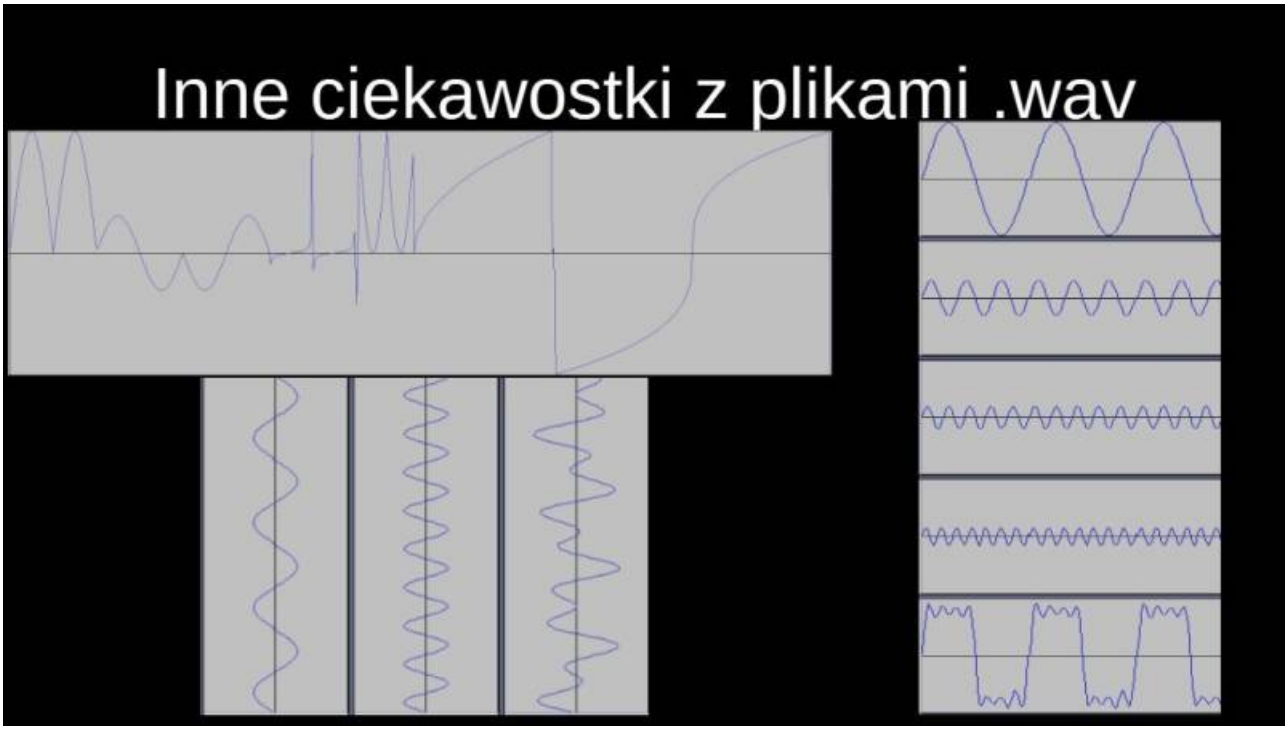


Parametry cd.

$13_D = 1011_{BLE}$

$13_D = 1101_{BBE}$

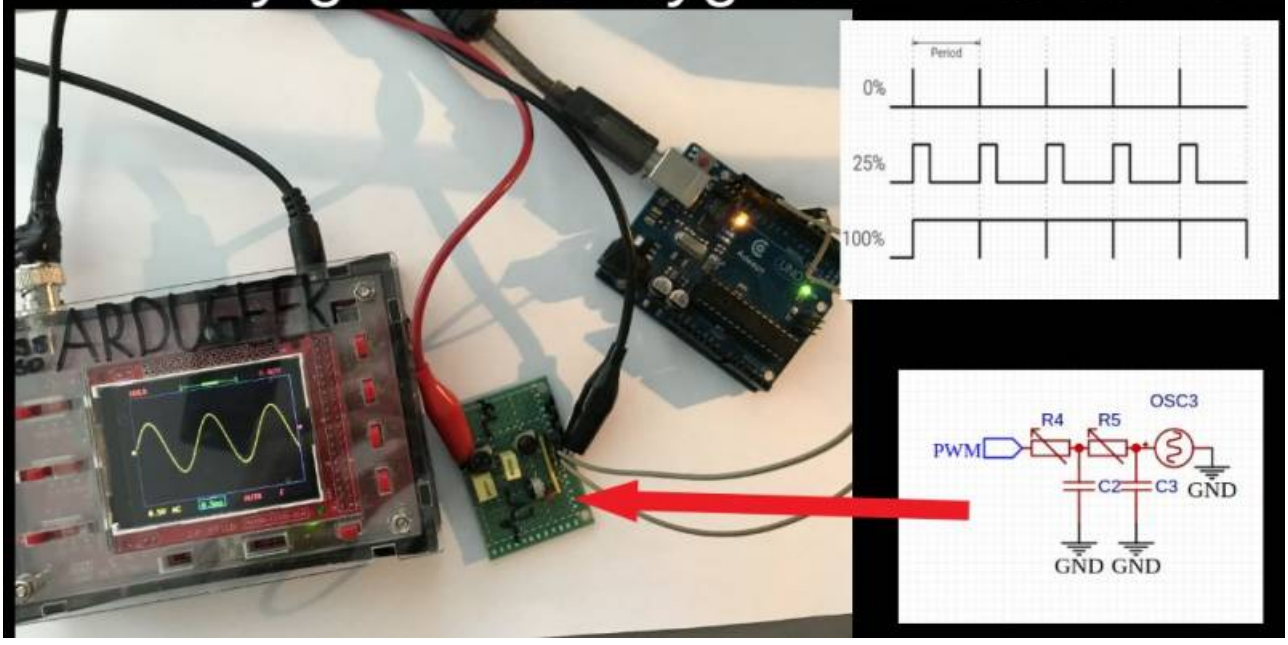


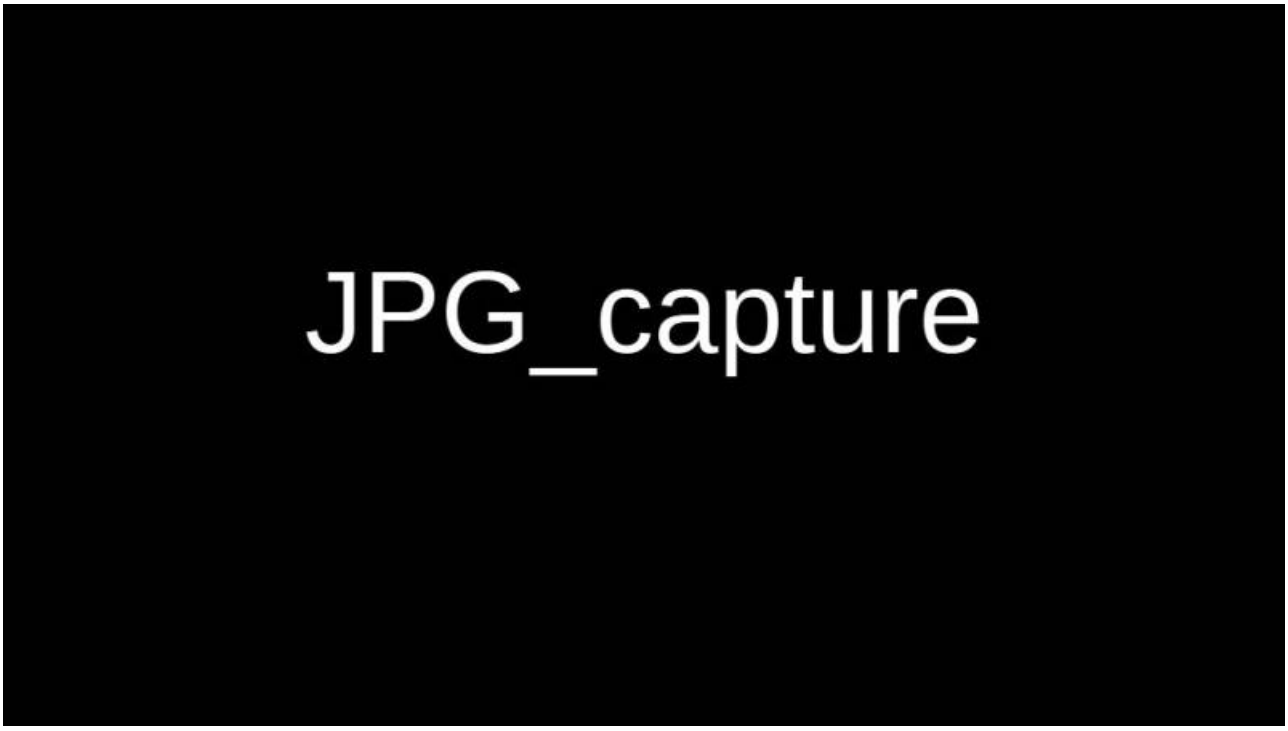
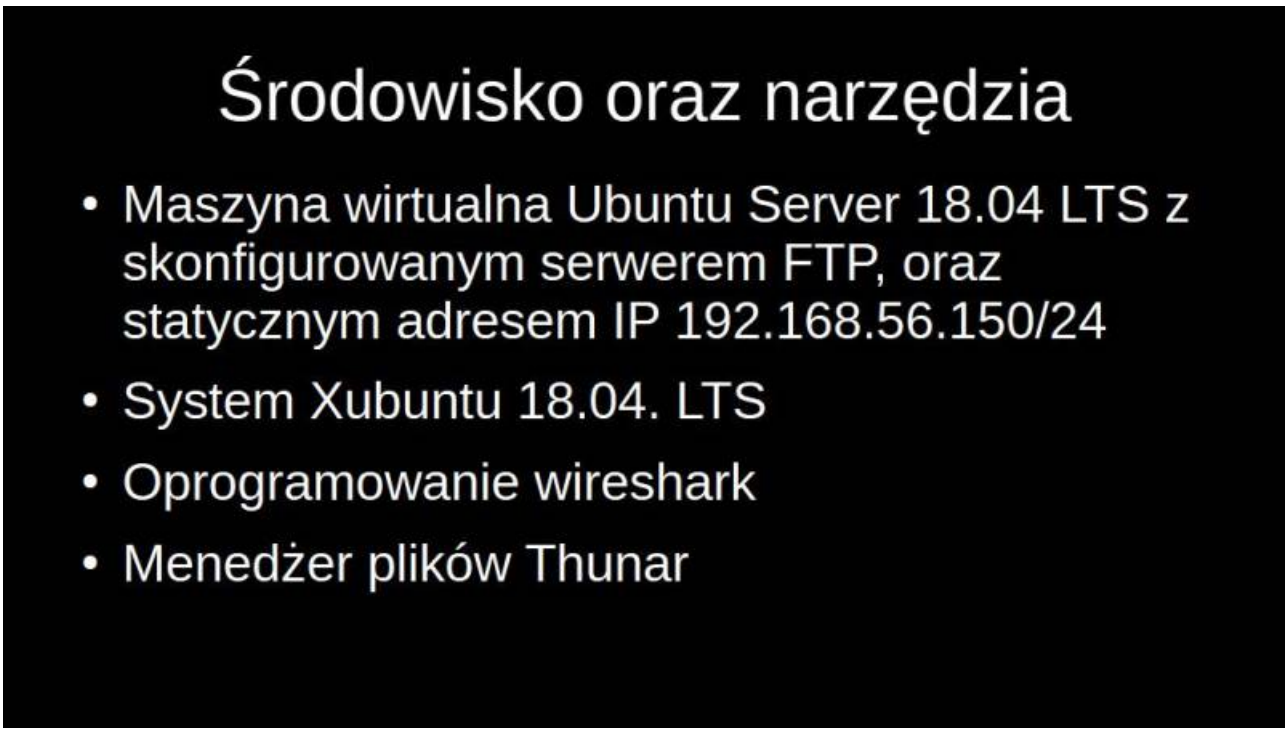
- 
Jak wygenerować plik .wav
import wave, struct, math, random, numpy
sampleRate = 44100.0
duration = 100.0
frequency = 440.0
obj = wave.open('sound.wav','w')
obj.setnchannels(1) # mono
obj.setsampwidth(2)
obj.setframerate(sampleRate)
audio = []
math_sine = []
values = numpy.arange(0, 6.28, 0.1)
for d in values:
var1 = int(10000*numpy.sin(d))
math_sine.append(var1)
for a in range(10000):
for b in math_sine:
audio.append(b)
for c in audio:
data = struct.pack('<h', c)
obj.writeframesraw(data)
obj.close()
2^16=65536
65536/2 = 32768
Bit depth = (-32768, 32768)
- 
Inne ciekawostki z plikami .wav

- # Do czego to może służyć

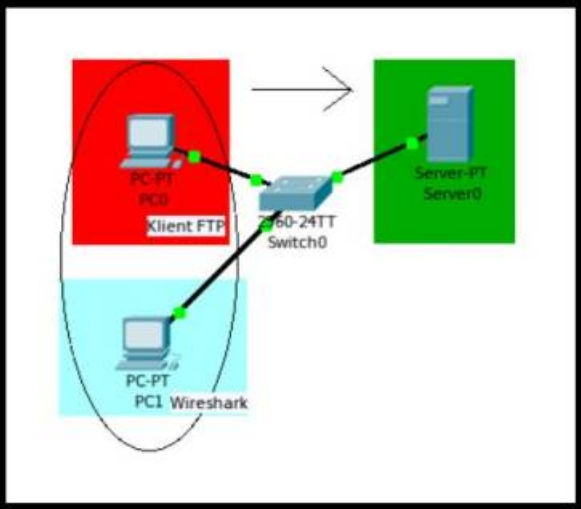
  - Możemy tą metodą spreparować sygnał cyfrowy:
    - Przy pomocy wzmacniacza I anteny wemitować go zakłucając lub uzyskując porządaną łączność
    - Przy pomocy arduino odtwarzać ten sygnał z karty microSD i wprowadzić go do danego układu uzyskując porządany efekt

- # Prosty generator sygnałów z arduino

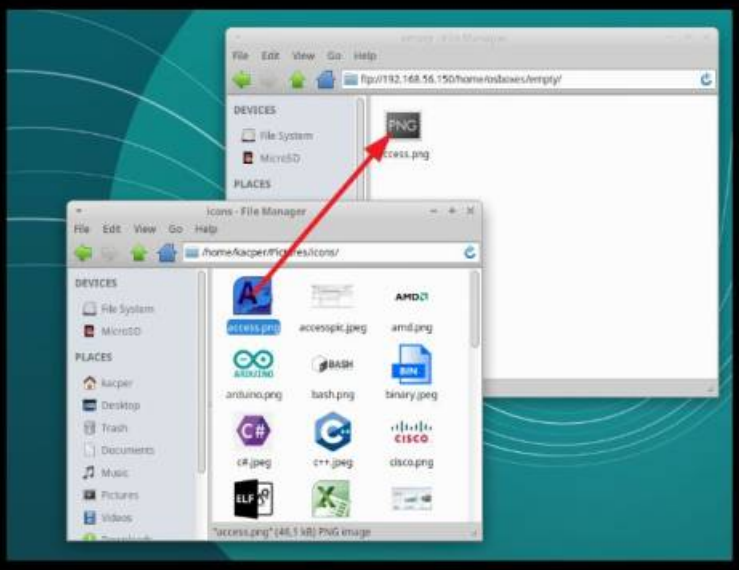


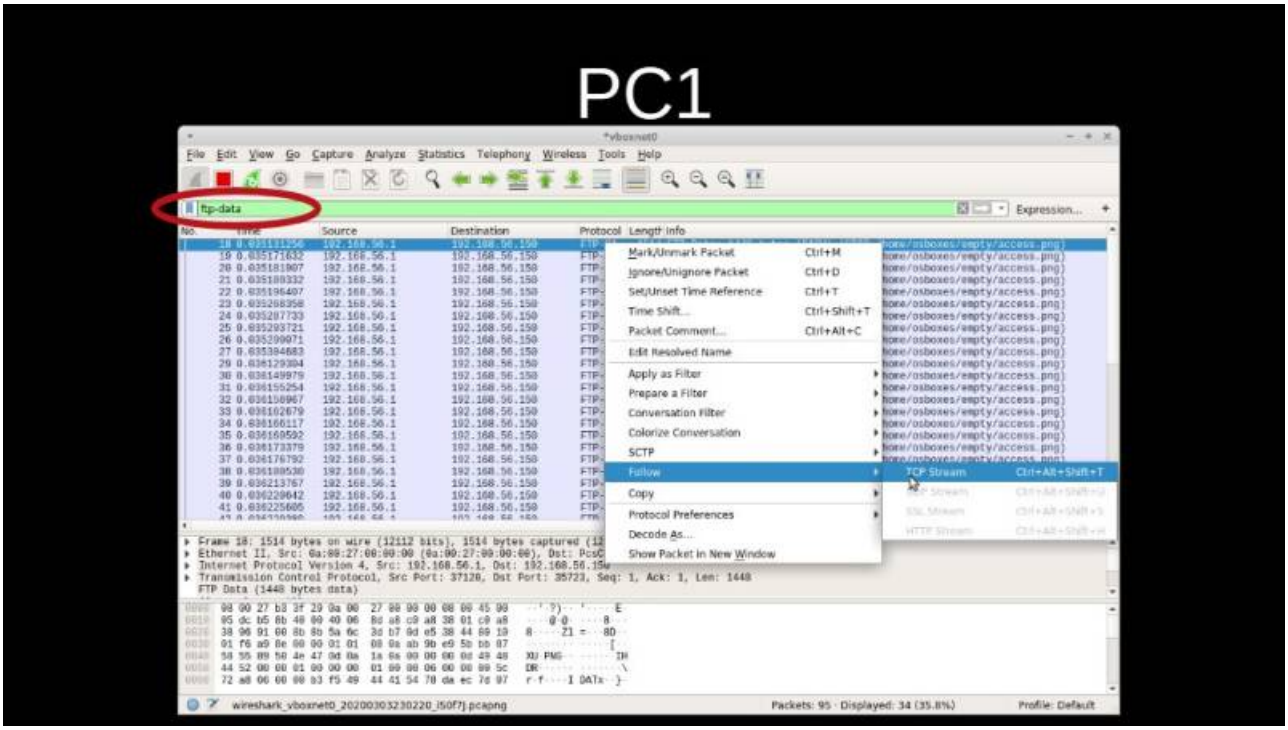
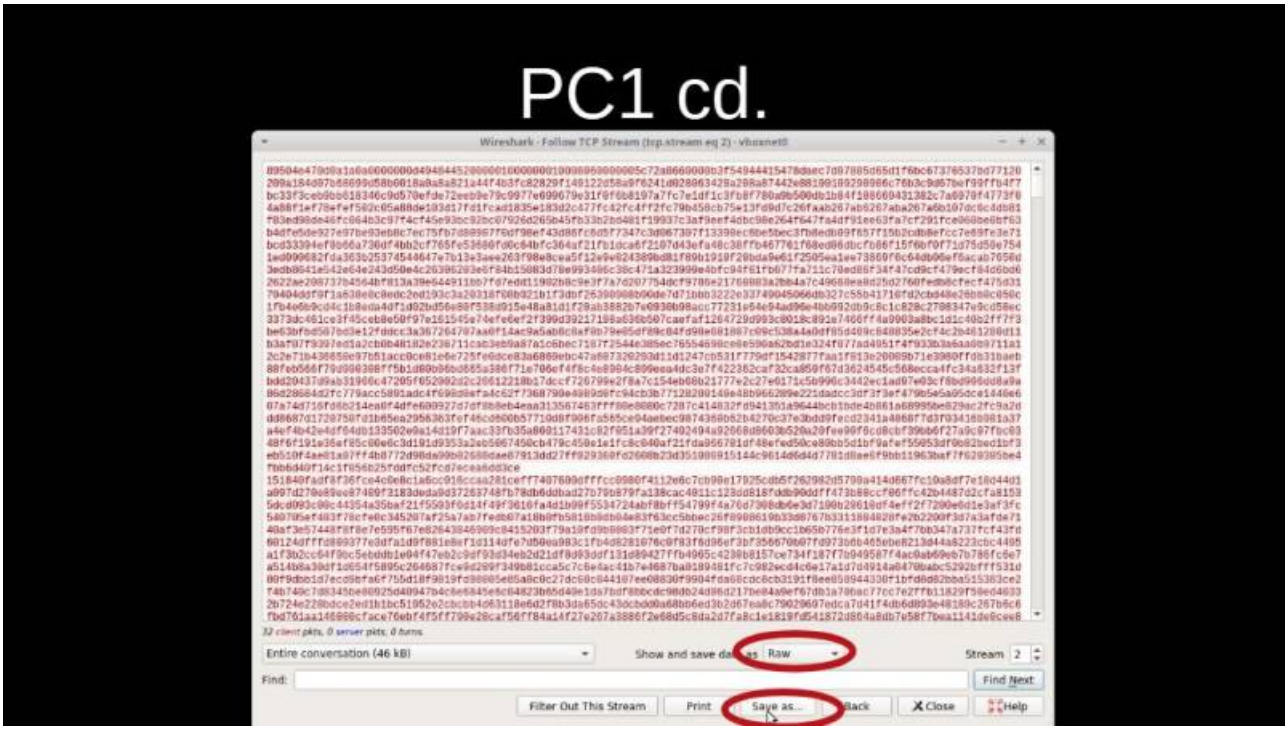
- 
JPG_capture
- 
Środowisko oraz narzędzia
• Maszyna wirtualna Ubuntu Server 18.04 LTS z skonfigurowanym serwerem FTP, oraz statycznym adresem IP 192.168.56.150/24
• System Xubuntu 18.04. LTS
• Oprogramowanie wireshark
• Menedżer plików Thunar

- # Przechwycenie pliku przesyłanego w źle/słabo skonfigurowanej sieci

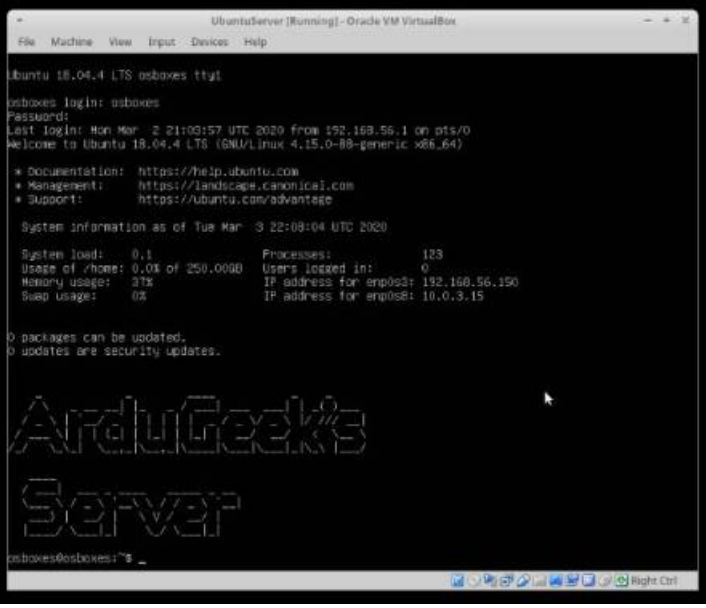


- # PC0



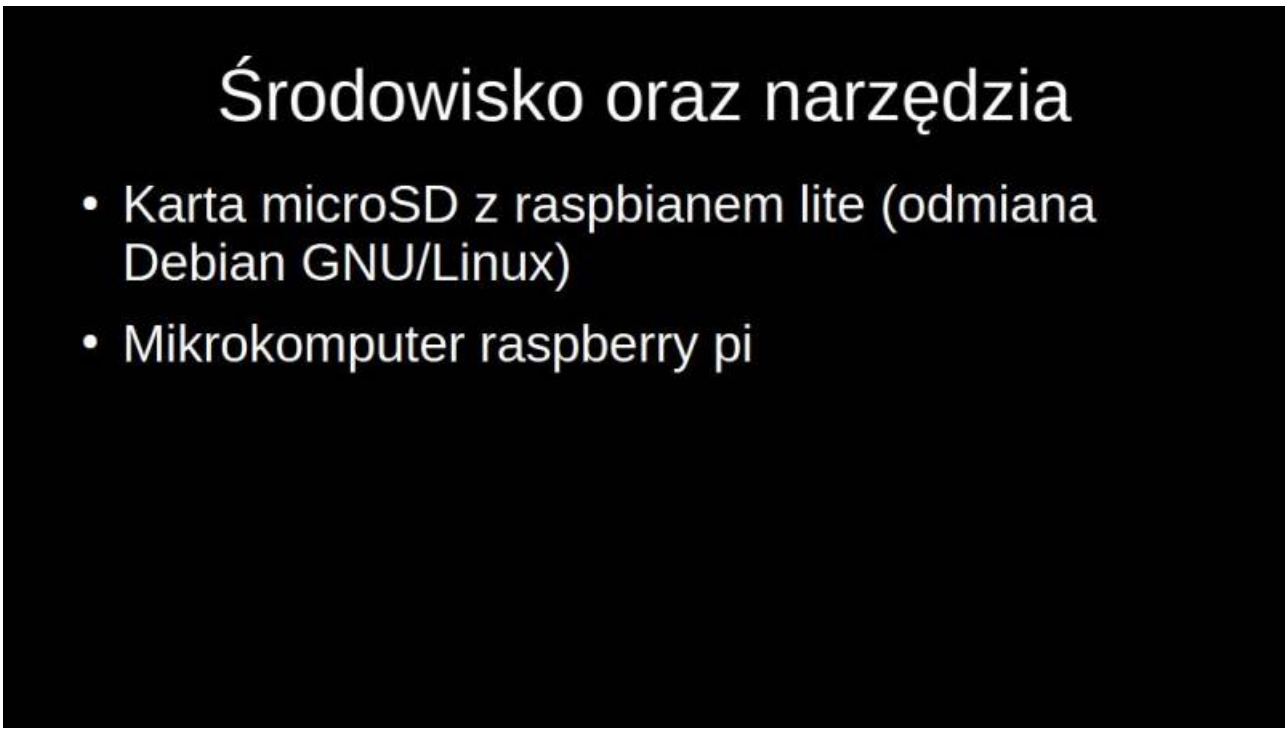
- 
PC1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ftp-data
Mark/Unmark Packet Ctrl+M
Ignore/Unignore Packet Ctrl+D
Set/Unset Time Reference Ctrl+T
Time Shift... Ctrl+Shift+T
Packet Comment... Ctrl+Alt+C
Edit Resolved Name
Apply as Filter
Prepare a Filter
Conversation Filter
Colorize Conversation
SCTP
Follow
Copy
Protocol Preferences
Decode As...
Show Packet in New Window
TCP Stream Ctrl+Alt+Shift+T
wireshark_vboxnet0_20200303230220_I50f7j.pcapng
Packets: 95 · Displayed: 34 (35.8%)
Profile: Default
- 
PC1 cd.
Wireshark · Follow TCP Stream (tcp.stream eq 2) · vboxnet0
Entire conversation (46 kB)
Show and save data as Raw
Stream 2
Find:
Find Next
Filter Out This Stream
Print
Save as...
Back
Close
Help

- # Server0

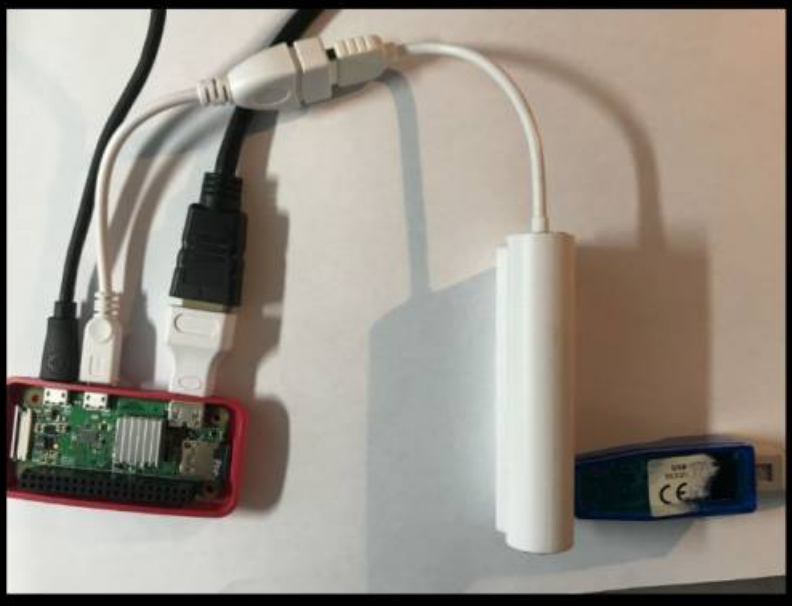


- # Wynik

  - Przechwycone dane zapisane pod rozszerzeniem .jpg, będą widoczne jako

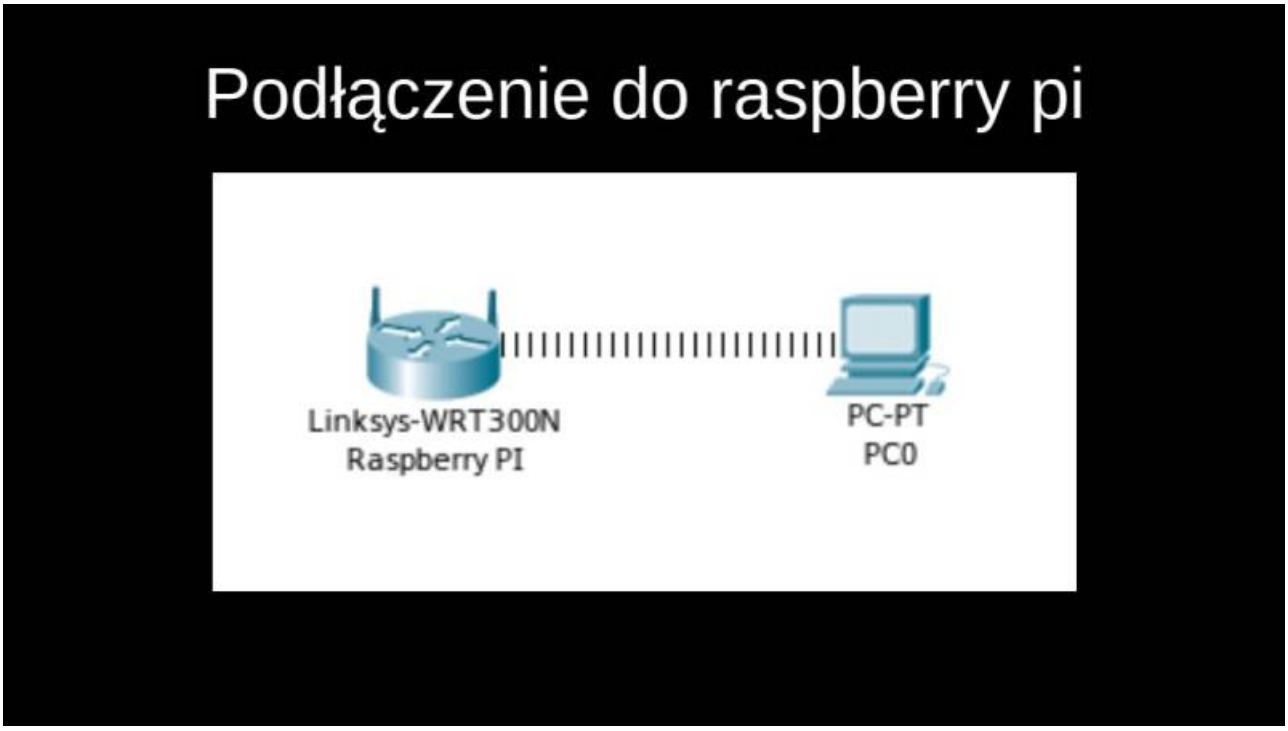
- 
Portable_lab
- 
Środowisko oraz narzędzia
• Karta microSD z raspbianem lite (odmiana Debian GNU/Linux)
• Mikrokomputer raspberry pi

- ## Hardware

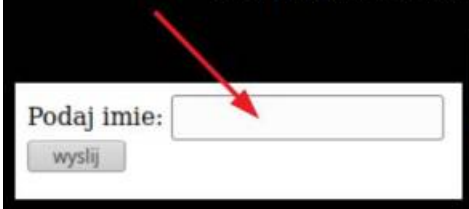


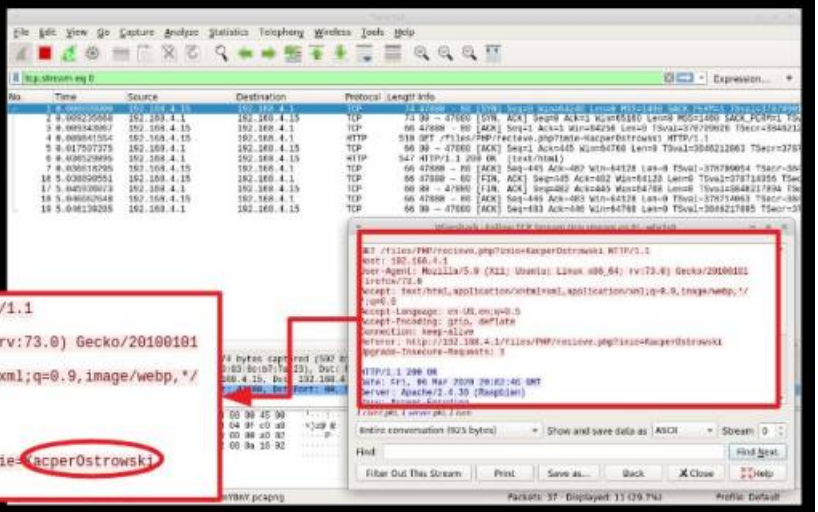
- ## Jakie funkcje mogą być przydatne?

  - Server ssh
  - Server FTP
  - Server http
  - Server mysql
  - Aplikacja phpMyAdmin

- 
Podłączenie do raspberry pi
Linksys-WRT300N
Raspberry PI
PC-PT
PC0
- 
Skonfigurowane usługi
kacper@kacper-E200HA:~$ nmap 192.168.1.170
Starting Nmap 7.60 (https://nmap.org) at 2020-03-06 20:42 CET
Nmap scan report for 192.168.1.170
Host is up (0.0047s latency).
Not shown: 996 closed ports
PORT STATE SERVICE
21/tcp open ftp
22/tcp open ssh
53/tcp open domain
80/tcp open http
Nmap done: 1 IP address (1 host up) scanned in 19.88 seconds
kacper@kacper-E200HA:~$

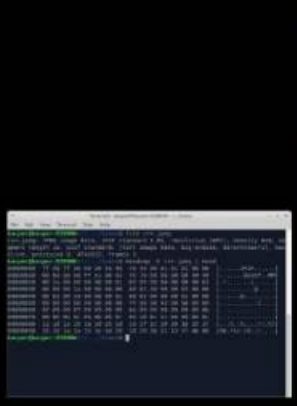
- # Do jakich testów może się przydać Portable_lab
  - Testy aplikacji webowych
  - Testy bazy danych
  - Przechwytywanie danych przesyłanych w sieci stworzonej przez Portable_lab
  - Ćwiczenie umiejętności programowania (aplikacji klienckich dla serwera)
- # Przechwycenie danych z nie zabezpieczonego formularza napisanego w jezyku PHP





```
GET /files/PHP/recieve.php?imie=KacperOstrowski HTTP/1.1
Host: 192.168.4.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:73.0) Gecko/20100101
Firefox/73.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/
*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.4.1/files/PHP/recieve.php?imie=KacperOstrowski
Upgrade-Insecure-Requests: 1
```

- Prezentacja jest dostępna na stronie:
  www.ardugeek.pl
  Pod zakładką Polish Site a cały kod z tej prezentacji jest dostępny pod zakładką Code
- Koło naukowe
  Łączność bez Zakłóceń



- 
Dziękuję za Uwagę
I zapraszam na moją stronę www.ardugeek.pl
ArduGeek
ARDUINO
BASH
AMD
intel
SQL
php
CISCO
fsfe