Low level attacks
Shellcode (part 2)

Mario Alviano

University of Calabria, Italy

A.Y. 2019/2020

Exploiting is a very slow process!

- We have to encode addresses in little-endian
- Those addresses changes very easily
- Plus, we have to compute a few offsets

Exploiting is a very slow process!

- We have to encode addresses in little-endian
- Those addresses changes very easily
- Plus, we have to compute a few offsets

Can we improve a bit the process?

Create a stable environment (1)

- So far, we disabled ASLR
- Still, the addresses retrieved via `gdb` do not match

Create a stable environment (1)

- So far, we disabled ASLR
- Still, the addresses retrieved via `gdb` do not match

Why?

- The stack also contains environment variables
- Check it on `addresses.c`
- Run with and without `gdb`
- The environment is different!

Create a stable environment (2)

- If we are starting the process, we can also choose the environment
- Let’s simplify the environment!

Create a stable environment (2)

- If we are starting the process, we can also choose the environment
- Let's simplify the environment!

Ignore the environment

- Use `env -i` to start the process (use absolute paths)
- Try again `addresses.c`:

```
$ env -i /tmp/a.out
```

Create a stable environment (2)

- If we are starting the process, we can also choose the environment
- Let's simplify the environment!

Ignore the environment

- Use `env -i` to start the process (use absolute paths)
- Try again `addresses.c`:

```
$ env -i /tmp/a.out
$ gdb a.out
(gdb) set exec-wrapper env -i
(gdb) r
```

Create a stable environment (2)

- If we are starting the process, we can also choose the environment
- Let's simplify the environment!

Ignore the environment

- Use `env -i` to start the process (use absolute paths)
- Try again `addresses.c`:

```
$ env -i /tmp/a.out
$ gdb a.out
(gdb) set exec-wrapper env -i
(gdb) r
```

- If you need some environment variable, add it inline
  eg. `$ env -i SHELL="/bin/sh" /tmp/a.out`

Pack memory with Python

- Most of the machines we use are little-endian

Pack memory with Python

- Most of the machines we use are little-endian
- Humans are error-prone on repetitive tasks

Pack memory with Python

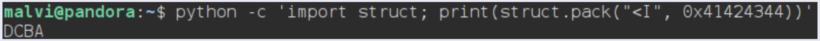
- Most of the machines we use are little-endian
- Humans are error-prone on repetitive tasks
- Better to automate the conversion!

Pack memory with Python

- Most of the machines we use are little-endian
- Humans are error-prone on repetitive tasks
- Better to automate the conversion!

Pythonic solution

```
import struct
struct.pack("<I", address)
```



```
malvi@pandora:~$ python -c 'import struct; print(struct.pack("<I", 0x41424344))'
DCBA
```

GDB enanchement (1)

- There are a few enanchement for `gdb`
- They add pretty printing functionalities

GDB enanchement (1)

- There are a few enanchement for `gdb`
- They add pretty printing functionalities
- One of them is `peda`
- Execute `source <path to peda.py>` in `gdb`
- Try on `victim.c`

Download `peda` from github

`https://github.com/longld/peda`

GDB enanchement (2)

You may want to add the following lines to `.gdb_init`

- Disable `less` for long output

  ```
  set pagination off
  ```

GDB enanchement (2)

You may want to add the following lines to `.gdb_init`

- Disable `less` for long output

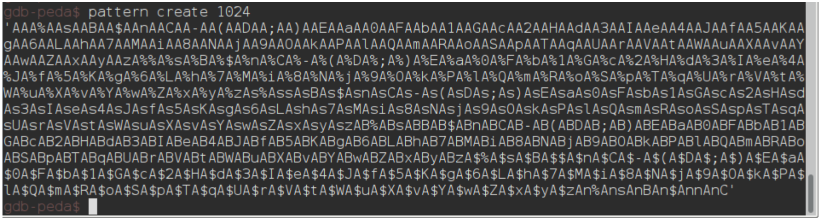
  ```
  set pagination off
  ```

- Keep a history of all the commands typed (search with `ctrl-r`)

  ```
  set history save on
  set history filename ~/.gdb_history
  set history size 32768
  set history expansion on
  ```

Find offsets with `peda` (1)

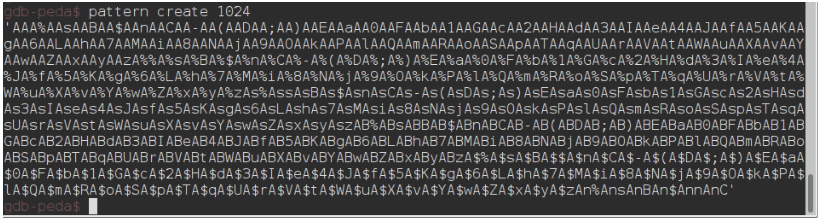
- Use `pattern create` to create a long pattern



```
gdb-peda$ pattern create 1024
'AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAA
gAA6AALAAhAA7AAMAAiAA8AANAAjAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAY
AAwAAZAAxAAyAAzA%%A%sA%BA%$A%nA%CA%-A%(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%2A%HA%dA%3A%IA%eA%4A
%JA%fA%5A%KA%gA%6A%LA%hA%7A%MA%iA%8A%NA%jA%9A%OA%kA%PA%lA%QA%mA%RA%oA%SA%pA%TA%qA%UA%rA%VA%tA%
WA%uA%XA%vA%YA%wA%ZA%xA%yA%zAs%AssAsBAs$AsnAsCAs-As(AsDAs;As)AsEAsaAs0AsFAsbAs1AsGAscAs2AsHAsd
As3AsIAseAs4AsJAsfAs5AsKAsgAs6AsLAshAs7AsMAsiAs8AsNAsjAs9AsOAskAsPAslAsQAsmAsRAsoAsSAspAsTAsqA
sUAsrAsVAstAsWAsuAsXAsvAsYAswAsZAsxAsyAszAB%ABsABBAB$ABnABCAB-AB(ABDAB;AB)ABEABaAB0ABFABbAB1AB
GABcAB2ABHABdAB3ABIABeAB4ABJABfAB5ABKABgAB6ABLABhAB7ABMABiAB8ABNABjAB9ABOABkABPABlABQABmABRABo
ABSABpABTABqABUABrABVABtABWABuABXABvABYABwABZABxAByABzA$%A$sA$BA$$A$nA$CA$-A$(A$DA$;A$)A$EA$aA
$0A$FA$bA$1A$GA$cA$2A$HA$dA$3A$IA$eA$4A$JA$fA$5A$KA$gA$6A$LA$hA$7A$MA$iA$8A$NA$jA$9A$OA$kA$PA$
lA$QA$mA$RA$oA$SA$pA$TA$qA$UA$rA$VA$tA$WA$uA$XA$vA$YA$wA$ZA$xA$yA$zAn%AnsAnBAn$AnnAnC'
gdb-peda$
```

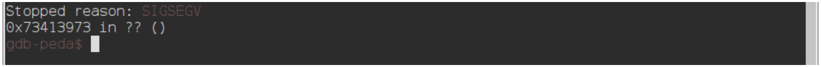
Find offsets with `peda` (1)

- Use `pattern create` to create a long pattern



```
gdb-peda$ pattern create 1024
'AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAA
gAA6AALAAhAA7AAMAAiAA8AANAAjAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAY
AAwAAZAAxAAyAAzA%%A%sA%BA%$A%nA%CA%-A%(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%2A%HA%dA%3A%IA%eA%4A
%JA%fA%5A%KA%gA%6A%LA%hA%7A%MA%iA%8A%NA%jA%9A%OA%kA%PA%lA%QA%mA%RA%oA%SA%pA%TA%qA%UA%rA%VA%tA%
WA%uA%XA%vA%YA%wA%ZA%xA%yA%zAs%AssAsBAs$AsnAsCAs-As(AsDAs;As)AsEAsaAs0AsFAsbAs1AsGAscAs2AsHAsd
As3AsIAseAs4AsJAsfAs5AsKAsgAs6AsLAshAs7AsMAsiAs8AsNAsjAs9AsOAskAsPAslAsQAsmAsRAsoAsSAspAsTAsqA
sUAsrAsVAstAsWAsuAsXAsvAsYAswAsZAsxAsyAszAB%ABsABBAB$ABnABCAB-AB(ABDAB;AB)ABEABaAB0ABFABbAB1AB
GABcAB2ABHABdAB3ABIABeAB4ABJABfAB5ABKABgAB6ABLABhAB7ABMABiAB8ABNABjAB9ABOABkABPABlABQABmABRABo
ABSABpABTABqABUABrABVABtABWABuABXABvABYABwABZABxAByABzA$%A$sA$BA$$A$nA$CA$-A$(A$DA$;A$)A$EA$aA
$0A$FA$bA$1A$GA$cA$2A$HA$dA$3A$IA$eA$4A$JA$fA$5A$KA$gA$6A$LA$hA$7A$MA$iA$8A$NA$jA$9A$OA$kA$PA$
lA$QA$mA$RA$oA$SA$pA$TA$qA$UA$rA$VA$tA$WA$uA$XA$vA$YA$wA$ZA$xA$yA$zAn%AnsAnBAn$AnnAnC'
gdb-peda$
```

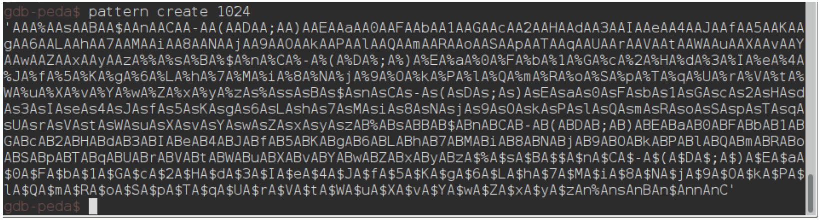
- Crash the process using the pattern



```
Stopped reason: SIGSEGV
0x73413973 in ?? ()
gdb-peda$
```

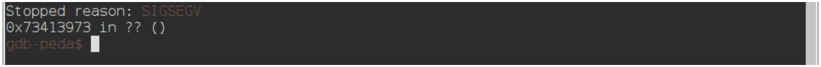
Find offsets with `peda` (1)

- Use `pattern create` to create a long pattern



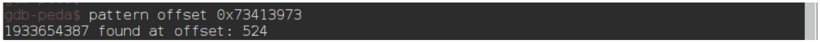
```
gdb-peda$ pattern create 1024
'AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAA
gAA6AALAAhAA7AAMAAiAA8AANAAjAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAY
AAwAAZAAxAAyAAzA%%A%sA%BA%$A%nA%CA%-A%(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%2A%HA%dA%3A%IA%eA%4A
%JA%fA%5A%KA%gA%6A%LA%hA%7A%MA%iA%8A%NA%jA%9A%OA%kA%PA%lA%QA%mA%RA%oA%SA%pA%TA%qA%UA%rA%VA%tA%
WA%uA%XA%vA%YA%wA%ZA%xA%yA%zAs%AssAsBAs$AsnAsCAs-As(AsDAs;As)AsEAsaAs0AsFAsbAs1AsGAscAs2AsHAsd
As3AsIAseAs4AsJAsfAs5AsKAsgAs6AsLAshAs7AsMAsiAs8AsNAsjAs9AsOAskAsPAslAsQAsmAsRAsoAsSAspAsTAsqA
sUAsrAsVAstAsWAsuAsXAsvAsYAswAsZAsxAsyAszAB%ABsABBAB$ABnABCAB-AB(ABDAB;AB)ABEABaAB0ABFABbAB1AB
GABcAB2ABHABdAB3ABIABeAB4ABJABfAB5ABKABgAB6ABLABhAB7ABMABiAB8ABNABjAB9ABOABkABPABlABQABmABRABo
ABSABpABTABqABUABrABVABtABWABuABXABvABYABwABZABxAByABzA$%A$sA$BA$$A$nA$CA$-A$(A$DA$;A$)A$EA$aA
$0A$FA$bA$1A$GA$cA$2A$HA$dA$3A$IA$eA$4A$JA$fA$5A$KA$gA$6A$LA$hA$7A$MA$iA$8A$NA$jA$9A$OA$kA$PA$
lA$QA$mA$RA$oA$SA$pA$TA$qA$UA$rA$VA$tA$WA$uA$XA$vA$YA$wA$ZA$xA$yA$zAn%AnsAnBAn$AnnAnC'
gdb-peda$
```

- Crash the process using the pattern



```
Stopped reason: SIGSEGV
0x73413973 in ?? ()
gdb-peda$
```

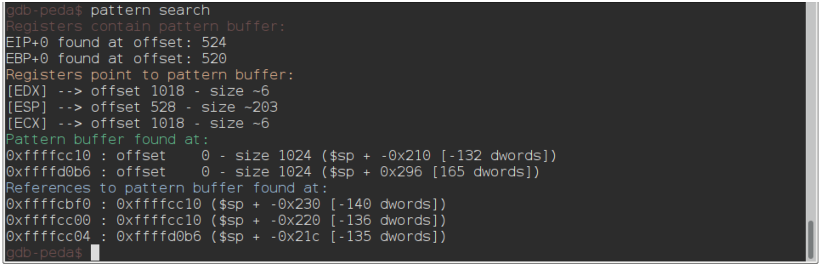
- Use `pattern offset` to compute the offset



```
gdb-peda$ pattern offset 0x73413973
1933654387 found at offset: 524
```

Find offsets with `peda` (2)

- Use `pattern search` to find the address of the pattern



```
gdb-peda$ pattern search
Registers contain pattern buffer:
EIP+0 found at offset: 524
EBP+0 found at offset: 520
Registers point to pattern buffer:
[EDX] --> offset 1018 - size ~6
[ESP] --> offset 528 - size ~203
[ECX] --> offset 1018 - size ~6
Pattern buffer found at:
0xffffcc10 : offset    0 - size 1024 ($sp + -0x210 [-132 dwords])
0xffffd0b6 : offset    0 - size 1024 ($sp + 0x296 [165 dwords])
References to pattern buffer found at:
0xffffcbf0 : 0xffffcc10 ($sp + -0x230 [-140 dwords])
0xffffcc00 : 0xffffcc10 ($sp + -0x220 [-136 dwords])
0xffffcc04 : 0xffffd0b6 ($sp + -0x21c [-135 dwords])
gdb-peda$
```

Build skeletons for your exploits

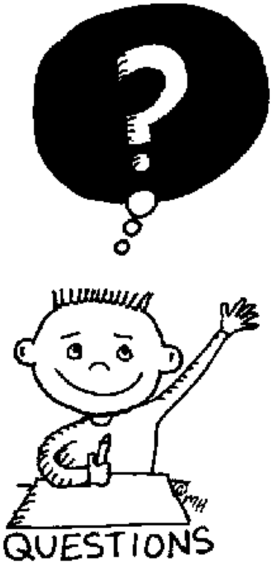
- Use strings of the same length
- Script your exploit as much as possible
- Try `skeleton.shellcode.py` on `victim.c`

Build skeletons for your exploits

- Use strings of the same length
- Script your exploit as much as possible
- Try `skeleton.shellcode.py` on `victim.c`

When you cannot start the process

- You have to try several addresses
- Again, a script may help you!



END OF THE LECTURE