In informatica, un **call stack** è una zona di memoria di un programma, organizzata in forma di stack, nella quale sono immagazzinate le informazioni sulle subroutine attive in un dato momento (le subroutine attive sono quelle che sono state invocate ma la cui esecuzione non è terminata). Può essere tradotto come *stack delle invocazioni a funzione* (o *chiamate a funzione*) o *stack delle invocazioni a metodo* (o *chiamate a metodo*) a seconda del tipo di subroutine coinvolto.

Questo tipo di pila è spesso chiamato anche **execution stack**, **control stack**, **function stack**, o **run-time stack** , oppure, quando non vi è possibilità di confusione, semplicemente **stack**.

C'è solitamente solo un call stack associato con un programma in esecuzione (o più precisamente con ogni task o thread di un processo), tuttavia stack aggiuntivi possono essere creati per gestire segnali o per il multitasking cooperativo. Poiché ne esiste uno solo in questo importante contesto, ci si riferisce semplicemente allo stack (implicitamente "del task").

Da http://www.phrack.org/archives/issues/49/ il primo "Smashing The Stack For Fun And Profit" (by Aleph1 http://en.wikipedia.org/wiki/Elias_Levy)
http://www.phrack.org/archives/issues/49/14.txt.

Controlling Execution Flow

http://www.offensive-security.com/metasploit-unleashed/Writing_An_Exploit

We now need to determine the correct offset in order get code execution. Fortunately, Metasploit comes to the rescue with two very useful utilities: **pattern_create.rb** and **pattern_offset.rb**. Both of these scripts are located in Metasploit's 'tools' directory. By running pattern_create.rb , the script will generate a string composed of unique patterns that we can use to replace our sequence of 'A's.

```
root@kali:~# /usr/share/metasploit-framework/tools/pattern_create.rb 1
1000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0A
c1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2
Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5.
..
```

After we have successfully overwritten EIP or SEH (or whatever register you are aiming for), we must take note of the value contained in the register and feed this value to pattern_offset.rb to determine at which point in the random string the value appears.

```
(gdb) info frame
```

http://stackoverflow.com/questions/5144727/how-to-interpret-gdb-info-frame-output

Lo **stack pointer** (ESP) è, nelle architetture x86 un registro dedicato della CPU che contiene l'indirizzo della locazione di memoria occupata dal top dello stack per permetterne le operazioni di push

[...] il registro dedicato EBP chiamato anche frame pointer o base pointer che punta, per tutta la durata della procedura, alla prima locazione di memoria del record di attivazione in modo che si possa far riferimento al top dello stack in maniera relativa ad essa.

http://en.wikipedia.org/wiki/Stack_buffer_overflow
http://upload.wikimedia.org/wikipedia/commons/4/4f/Stack_Overflow_2.png
http://upload.wikimedia.org/wikipedia/commons/9/93/Stack_Overflow_3.png
http://upload.wikimedia.org/wikipedia/commons/c/c3/Stack_Overflow_4.png

Altri WARGAMES
http://overthewire.org/wargames/
http://vulnhub.com/resources/
http://vulnhub.com/entry/brainpan_2,56/

Alla ricerca di memory leak!
Valgrind http://valgrind.org/info/developers.html
http://en.wikipedia.org/wiki/Memory_debugger

MISC
http://aerofex.com/

PDF Attachment