KPMG

Linux kernel & OS security

Hacking Windows

Pieter Ceelen and Stan Hegt

18 December 2009

AUDIT / TAX / ADVISORY

AUDIT ▪ TAX ▪ ADVISORY

Who am I?

- A hacker in a fancy suit
- Other job: DJ'ing
- Former IST student

Goals of this presentation

Goals

- Teach theory and practice on hacking Microsoft Windows
- Provide insight into the world of information security professionals

Questions and interaction

- Interaction is favoured
- Relevant questions may be asked immediately

Required time

- 2 x 45 minutes

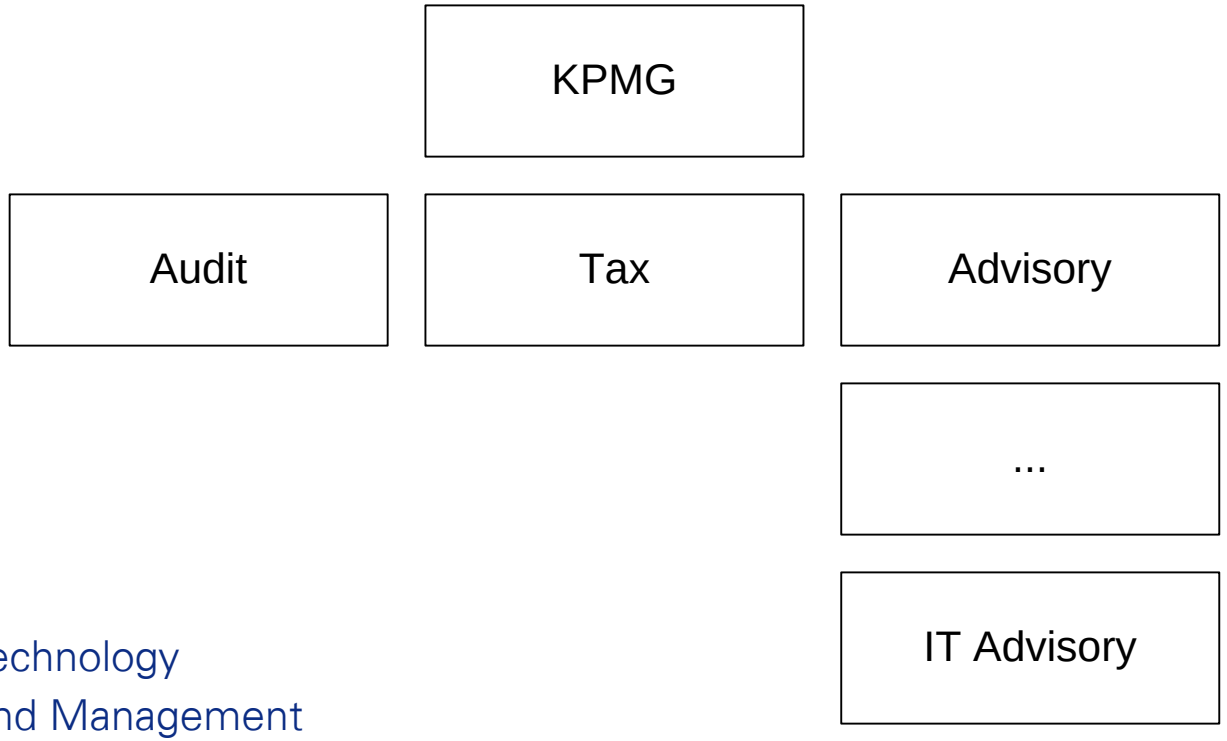
Contents

- Introduction to KPMG
- Buffer overflows in Windows
  - Basic overflows
  - Tricks
- Windows passwords
  - Secure storage
  - Hashing
- Post-exploitation
  - Pass the hash
  - Meterpreter
  - Incognito

Introduction to KPMG

IT Advisory Netherlands

- 250 employees
- Offered Services:
  - IT Assurance
  - IT Projects
  - IT Systems
  - IT Security
  - IT Governance

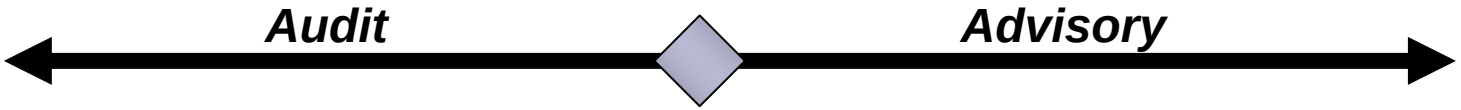


Background employees

- Master degree in e.g.:
  - Computer Science
  - Business Information Technology
  - Industrial Engineering and Management

IT Advisory, Business Unit - ICT Security & Control

- 30 employees with technical background
- Services offered:
  - Security Testing (e.g. ethical hacking)
  - Assessing IT-infrastructures (networks, databases, applications, etc.)
  - Public Key Infrastructures
  - Identity and Access Management



Focus on the past
Assurance

Focus on the future
Implementing improvements

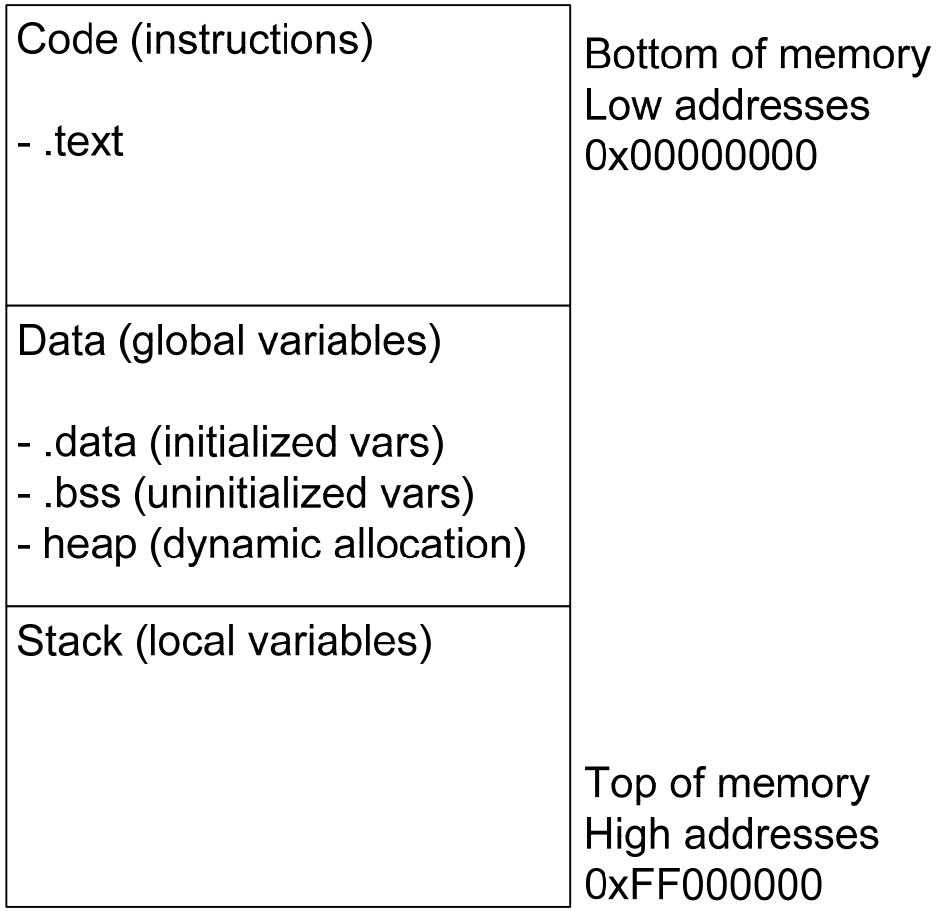
Buffer overflows in Windows

A process' memory

3 Sections in process memory

- Code
- Data
- Stack

Today's focus on stack based buffer overflows!



The stack

Usage

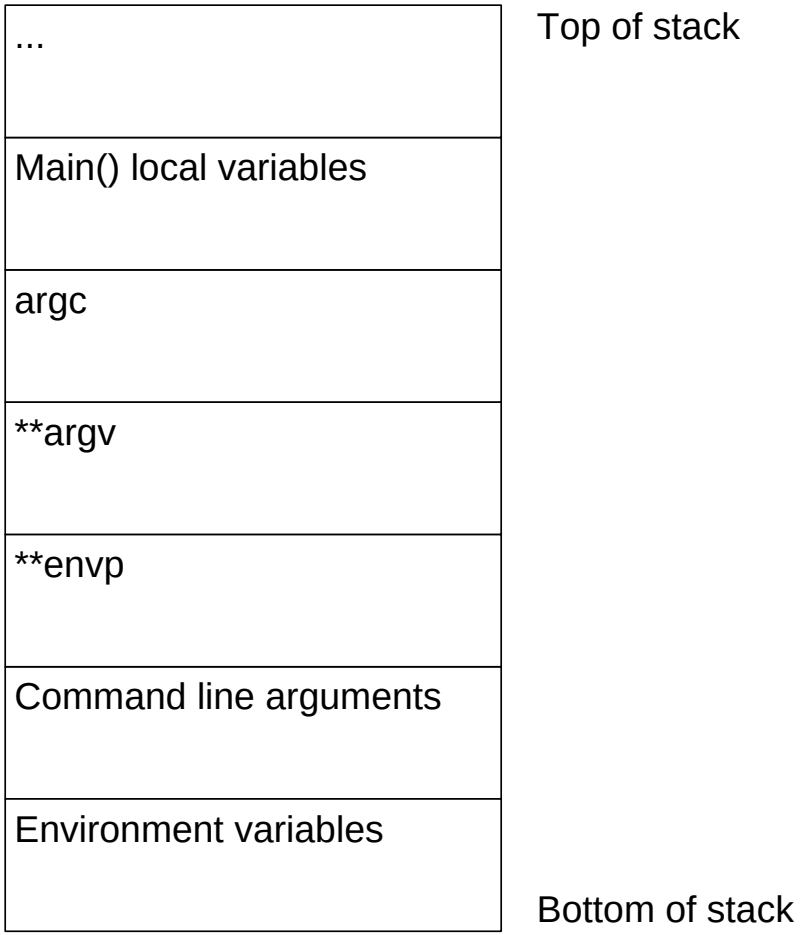
- Pass arguments to functions
- Used as space for local variables

Location

- Top of process' memory
- Grows down
- Top of stack is lowest memory address

ESP register (stack pointer)

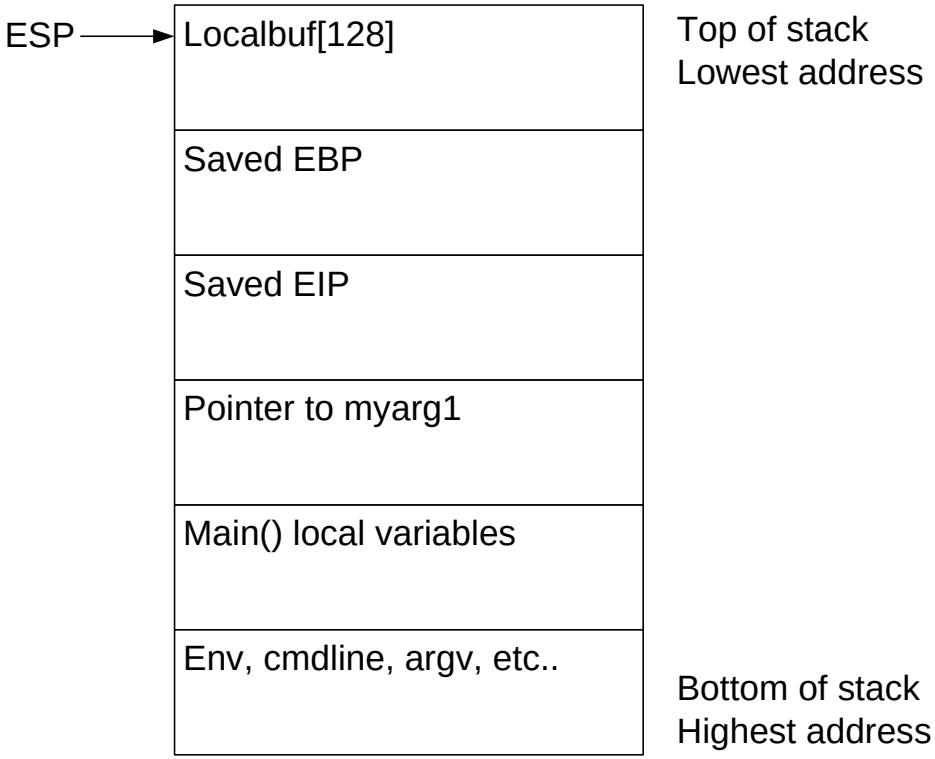
- Point to top of stack
- Push: decrement ESP by 4
- Pop: increment ESP by 4



The stack: a function call

Call of
myfunction(myarg1)
{ char localbuf[128]; … }

- Push *myarg1
- Push EIP (so we know where to return after completion of myfunction)
- Push EBP
- Reserve 128 bytes for localbuf
- ESP points to start of localbuf

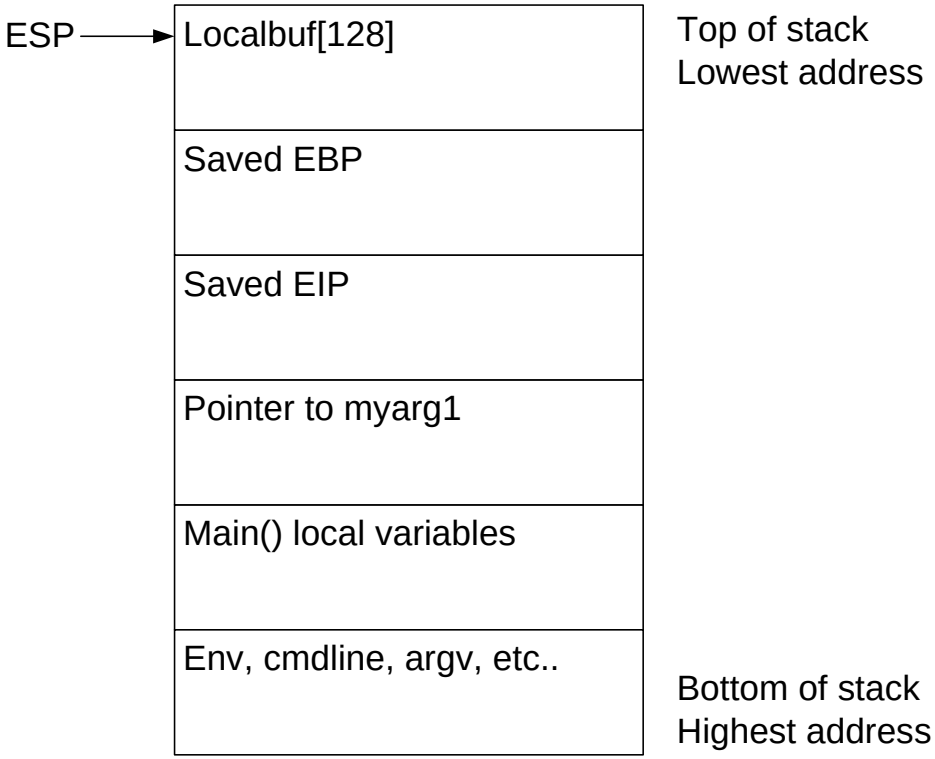


The stack: buffer overflow

Now what if data is copied after mybuf[128]?

- For example by a strcpy(mybuf, argv[1])
- Saved EBP and EIP may be overwritten
- After completion of myfunction() program flow will return to address of saved EIP

If we control saved EIP, we control program flow!



Demonstration

Easy RM to MP3 version 2.7.3.700

- Can be obtained from oldapps.com

WinDBG

- Freeware debugger by Microsoft

Determine buffer size

How large is the buffer we are overflowing?

- Trial and error
- Use Metasploit!

Pattern_create.rb

- Creates magic pattern
- Feed pattern to overflowed buffer

Pattern_offset.rb

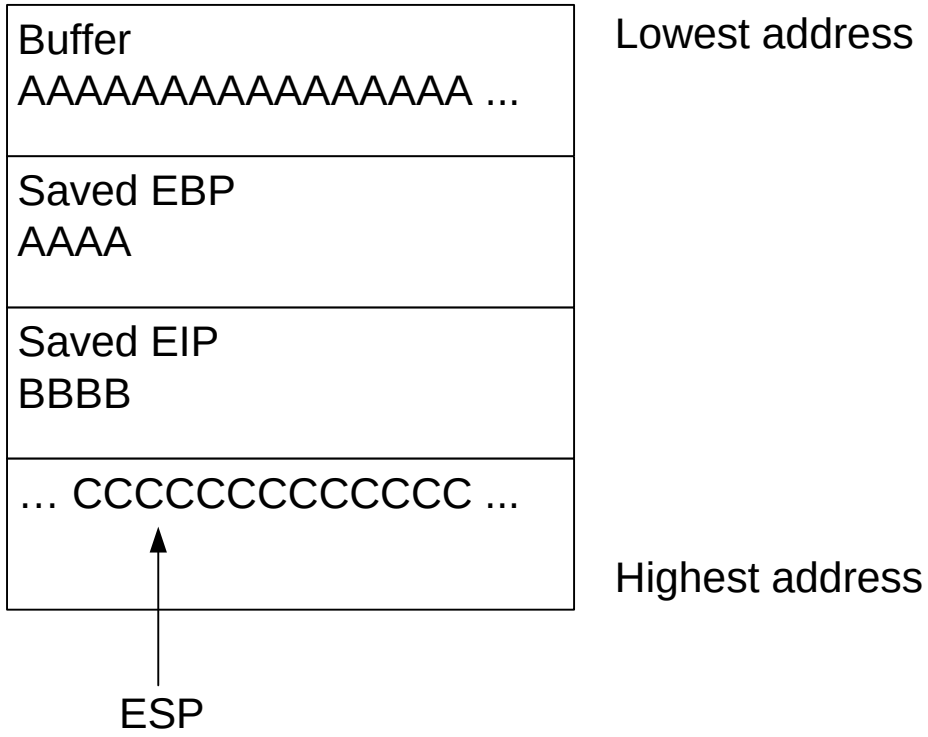
- Recognizes offset of EIP in pattern
- Value of EIP after overflow is size of buffer

The stack: post-mortem

What happened?

- Buffer got filled with A's (26072 – 4)
- Saved EBP got overwritten with AAAA
- Saved EIP got overwritten with BBBB
- Large number of C's after saved EIP
- ESP points somewhere in series of C's

Let's insert some evil code instead of C's and let saved EIP point to this code!



Payload (also called shellcode)

Where to find payload to insert instead of C's?

- Develop it yourself
- Obtain it from an obscure website
- Metasploit to the rescue!

Msfpayload

- Dozens of cool payloads
- Output can be used in Perl directly

Find payload in memory

So we got a cool payload to insert instead of C's.

How do we find the start of our payload in memory?

- Use the pattern_create.rb / pattern_offset.rb from Metasploit again!
- Keep in mind little endian

What went wrong?

Our shellcode contains null bytes!

- Strcpy() will stop copying after null byte
- So our shellcode is not copied..

No worries. Metasploit to the rescue again!

- Use msfencode to obtain an encoded payload with null bytes removed

What went wrong? (2)

Our value for saved EIP contains null bytes!

- Strcpy() will stop copying after null byte
- So our shellcode is not copied..

No worries.

- Exploit fact that ESP register points to start of our shellcode
- FF E4 (= opcode JMP ESP)

Do not try this at home

Basic introduction

- Many things might fail with your specific Windows version
- More reliable: SEH overwrite ()

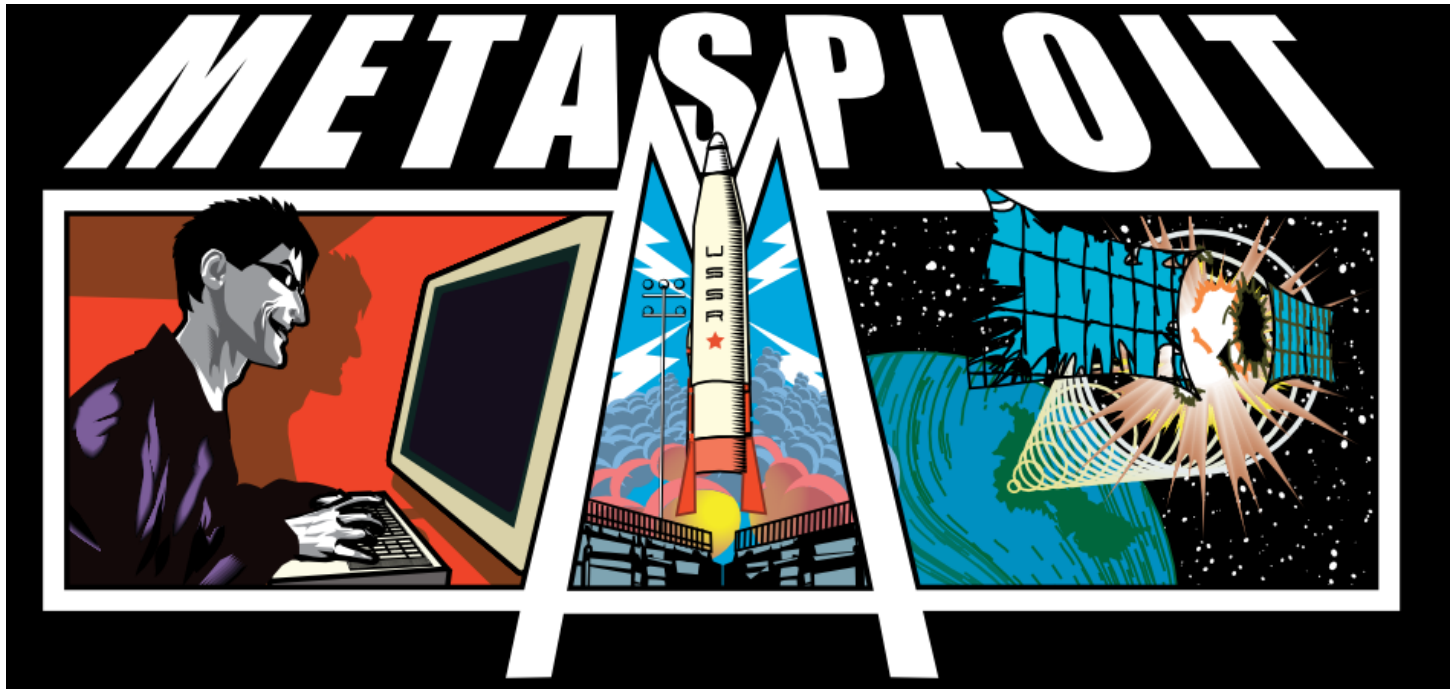
Post exploitation

Post exploitation

So you own a box. Now what?

Metasploit is your friend :-)

- Pass the hash
- Meterpreter
- Incognito



Pass the hash / PSExec

Who needs passwords when one can authenticate using hashes?

- Use the PSExec module of Metasploit
- Enter LM & NTLM hash and go

Demonstration!

Meta Interpreter

Advanced staged payload

- Upload a DLL
- Reflective injection of DLL in heap of compromised process
- Create communication channel
- Run scripts

Presence in memory only (default)

- Installation does not touch disk
- Optional persistent installation on disk **(`run persistance -h`)**

Incognito

What if you own a Windows box and need to impersonate the user that is logged on?

- Capture Windows access tokens using Incognito
- Access any data (network shares!) and applications as being the user

Demonstration!

Speaker's contacts

Thank you for your attention!

Stan Hegt
Junior Advisor, ICT Security & Control

hegt.stan@kpmg.nl
06 - 11 88 50 39