SecurityTube Python Scripting Expert (SPSE)

SecurityTube Python Scripting Expert

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Module 5: Exploitation Techniques

SecurityTube Python Scripting Expert

Part 1:
Exploit Research Basics

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Art of Exploit Research

Platforms
Windows XP – SP1, SP2 …
Windows Vista – SP…
Windows 7 – SP…
Windows Server 2003, 2008 …
Linux
Mac OSX

Exploitation Techniques
Simple Buffer Overflows
SEH, SafeSEH
NX, DEP
ASLR
Stack Cookies
….

Tools of the Trade

- Immunity Debugger (Free)
- Metasploit Framework (Free)
- OllyDbg (Free)
- IDA Pro ($$, through older free version available)
- WinDbg (Free)
- … other tools wherever necessary

Pre-Requisites

- Assembly Language Programming in Linux
- Assembly Language Programming in Windows
- Buffer Overflow Primer on Linux

http://www.securitytube.net/groups?operation=viewall&groupId=0

Objective of this Module

- NOT a comprehensive guide on Exploitation Techniques
  - a course in itself (Multiple OS, Architecture …)
- How to apply Python in conjunction with Immunity Debugger to analyze binaries for exploitation
- Apply Scripting and Automation to Exploitation

Sampling a Buffer Overflow

- Crashing the Program
- Code in Python
- Analyze Crash

We will use Immunity Debugger 1.85 and crash analysis on Windows XP to keep the exploitation part simple

Module 5: Exploitation Techniques

SecurityTube Python Scripting Expert

End of Part 1:
Exploit Research Basics

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Module 5: Exploitation Techniques

SecurityTube Python Scripting Expert

Part 2:
Immunity Debugger Basics

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Immunity Debugger

- We will be using 1.85 which is the latest as of this writing
- Disassembler + Debugger
- Allows for extensions using Python
- Documentation is a bit sparse
  - can learn for shipped extension code

Module 5: Exploitation Techniques

SecurityTube Python Scripting Expert

End of Part 2:
Immunity Debugger Basics

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Module 5: Exploitation Techniques

SecurityTube Python Scripting Expert

Part 3:
Immunity Debugger Scripting Basics

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Scripting Options with ID

- PyCommands
- PyHooks
- PyPlugins

PyCommands

- Simple scripts using the immlib APIs
- Is not cached – can be changed at runtime
- Quick prototyping
  - Python Command Shell
- Needs to define a main() and return a string (shown in status box)
- may have arguments

Communicating Results

- Print to Status box via Return value
- Print to Log Window
- Print to a Table
- Write to File

Exercise

- Write the list of processes in a CSV file
  - the first row should be the table column names

Module 5: Exploitation Techniques

SecurityTube Python Scripting Expert

Part 3:
End of Immunity Debugger Scripting Basics

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Module 5: Exploitation Techniques

SecurityTube Python Scripting Expert

Part 4:
Processes in-Depth

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Emulate the same in Code

- Open / Attach to a Process
- Find List of All Modules in the Process
  - name
  - base address
  - size
  - entry point
  - version

Module 5: Exploitation Techniques

SecurityTube Python Scripting Expert

Part 4:
End of Processes in-Depth

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Module 5: Exploitation Techniques

SecurityTube Python Scripting Expert

Part 5:
Assemble, Disassemble, Search and Locate Instructions

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Assemble / Disassemble Instructions

- Assemble Instructions
  - imm.assemble(….)
  - multiple instructions separated by “\n”
- Disassemble instructions
  - imm.diasm(address).getdisasm()

Search / Locate for Instructions

- imm.search(assembled_instruction)

- Find the corresponding Module
  - imm.findModule(address)
  - returns tuple (module_name, base_address)

Module 5: Exploitation Techniques

SecurityTube Python Scripting Expert

End of Part 5:
Assemble, Disassemble, Search and Locate Instructions

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Module 5: Exploitation Techniques

SecurityTube Python Scripting Expert

Part 6:
PyHooks

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Why Hooks?

- Event Driven – exceptions, function calls, process and thread creations etc.
- Allow for “reactive” approach when events are hit
- Can be part of PyCommand scripts or can be stand alone

PyHooks – Which Events?

	FastLogHook
	STDCALLFastLogHook
	Hook
	BpHook
	LogBpHook
	PreBpHook
	AllExceptHook
	PostAnalysisHook
	AccessViolationHook
	RunUntilAV
	LoadDLLHook
	UnloadDLLHook
	CreateThreadHook
	ExitThreadHook
	CreateProcessHook
	ExitProcessHook

Immunity Debugger Documentation

AllExceptHook

- Triggers when program exceptions happen
- All exceptions included
- Lets write some code! ☺

BpHook/LogBpHook Exercise

- Our vulnerable program uses strcpy()
- Can you create a hook for the strcpy() function which will print all the function arguments
- How can we infer an overflow is about it happen?

Injection Hooking

- FastLogHook, STDCALLFastLogHook
- uses Injection hooking and does not need debugger
- idea is to inject our code and redirect control
  - finish logging and return to normal flow

Injection Hooking Exercise

- Convert the previous exercise to use Injection Hooking

Module 5: Exploitation Techniques

SecurityTube Python Scripting Expert

Part 6:
End of PyHooks

http://www.securitytube.net

Vivek Ramachandran
Course Instructor

Module 5: Exploitation Techniques

SecurityTube Python Scripting Expert

Part 7:
Exploiting a Buffer Overflow

http://www.securitytube.net

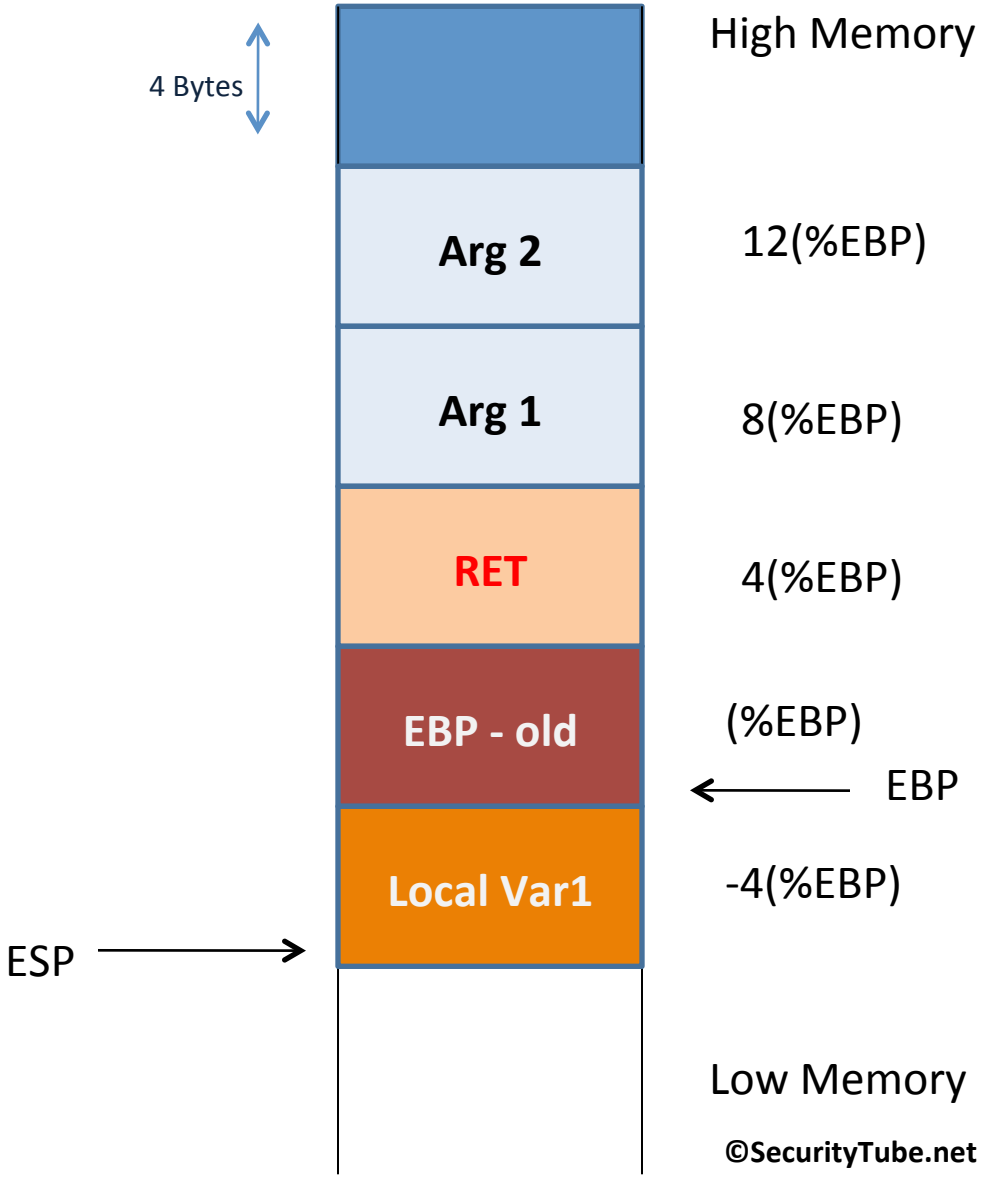
Vivek Ramachandran
Course Instructor

Buffer Overflow Exploitation

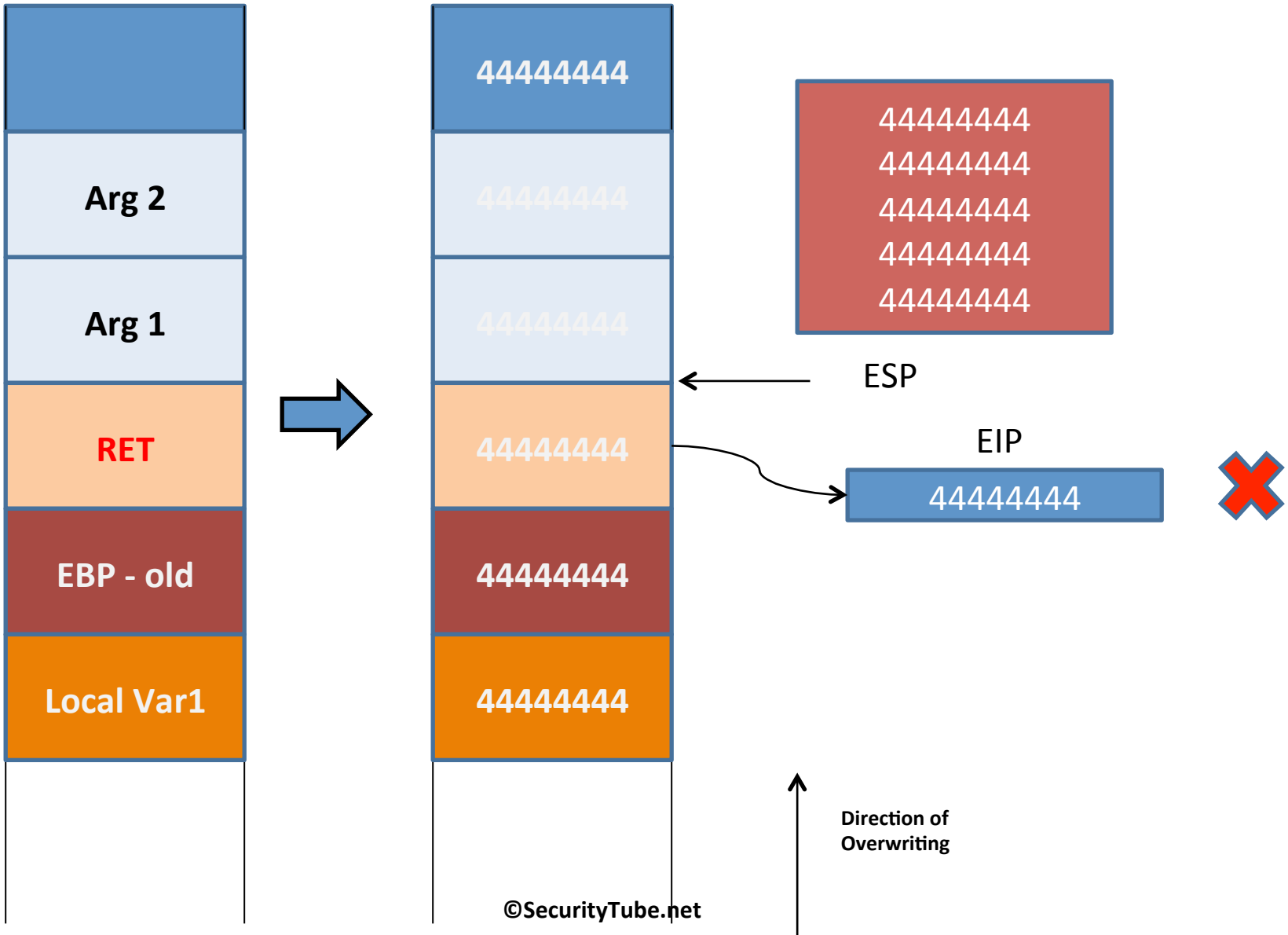
- http://www.securitytube.net/groups?operation=view&groupId=7
  - Buffer Overflow
  - SEH etc.

- Lets analyze simple buffer overflow

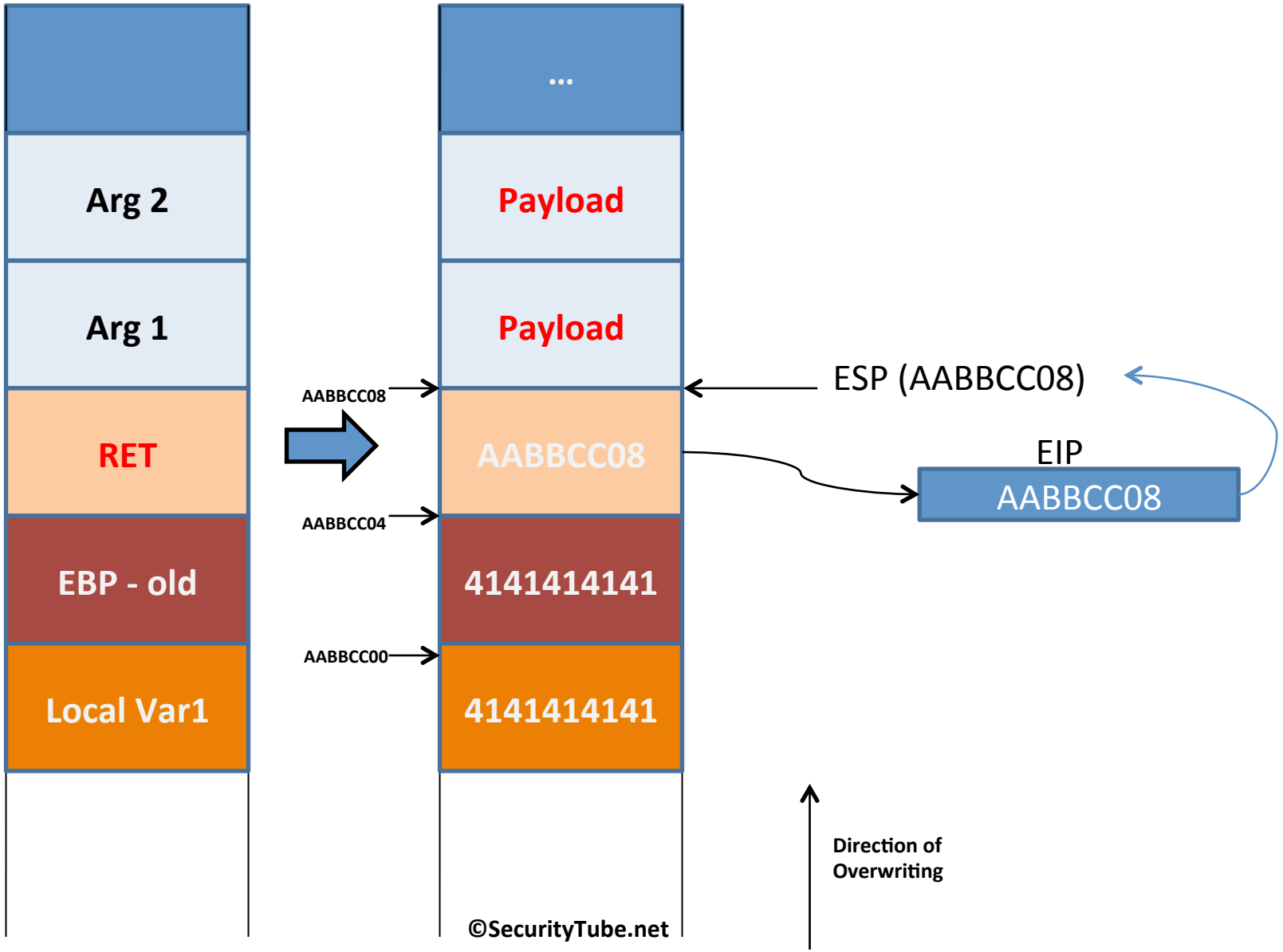
Remember the Stack Layout?



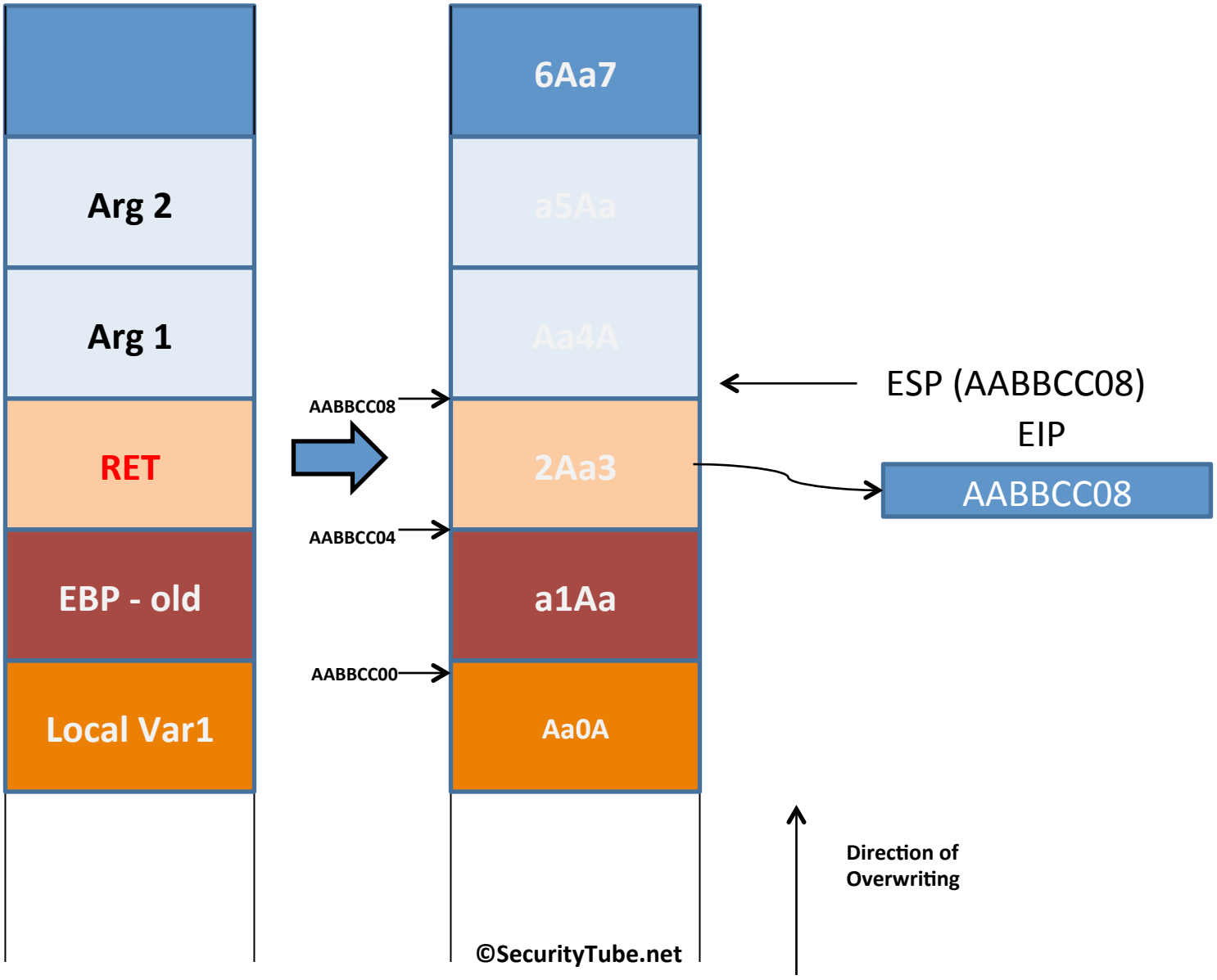
Overwriting the Stack with “A”



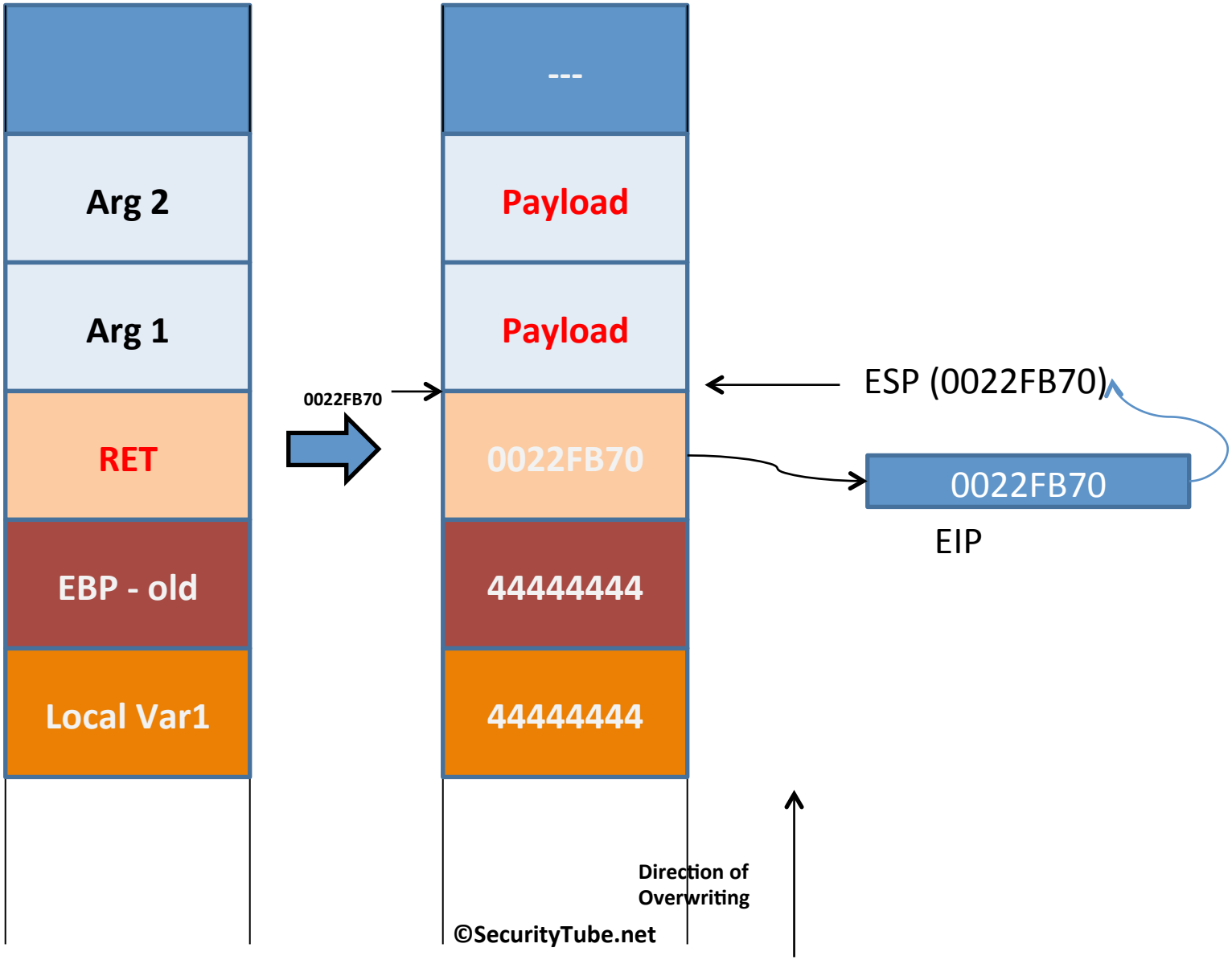
Successful Exploitation



How do we find the offset and memory addresses?



Successful Exploitation



Vulnerable Code – using Strcpy()

```
int VulnerableFunction(char *userInput)
{
        char buffer[256];

        strcpy(buffer, input);

        return 1;
}
```

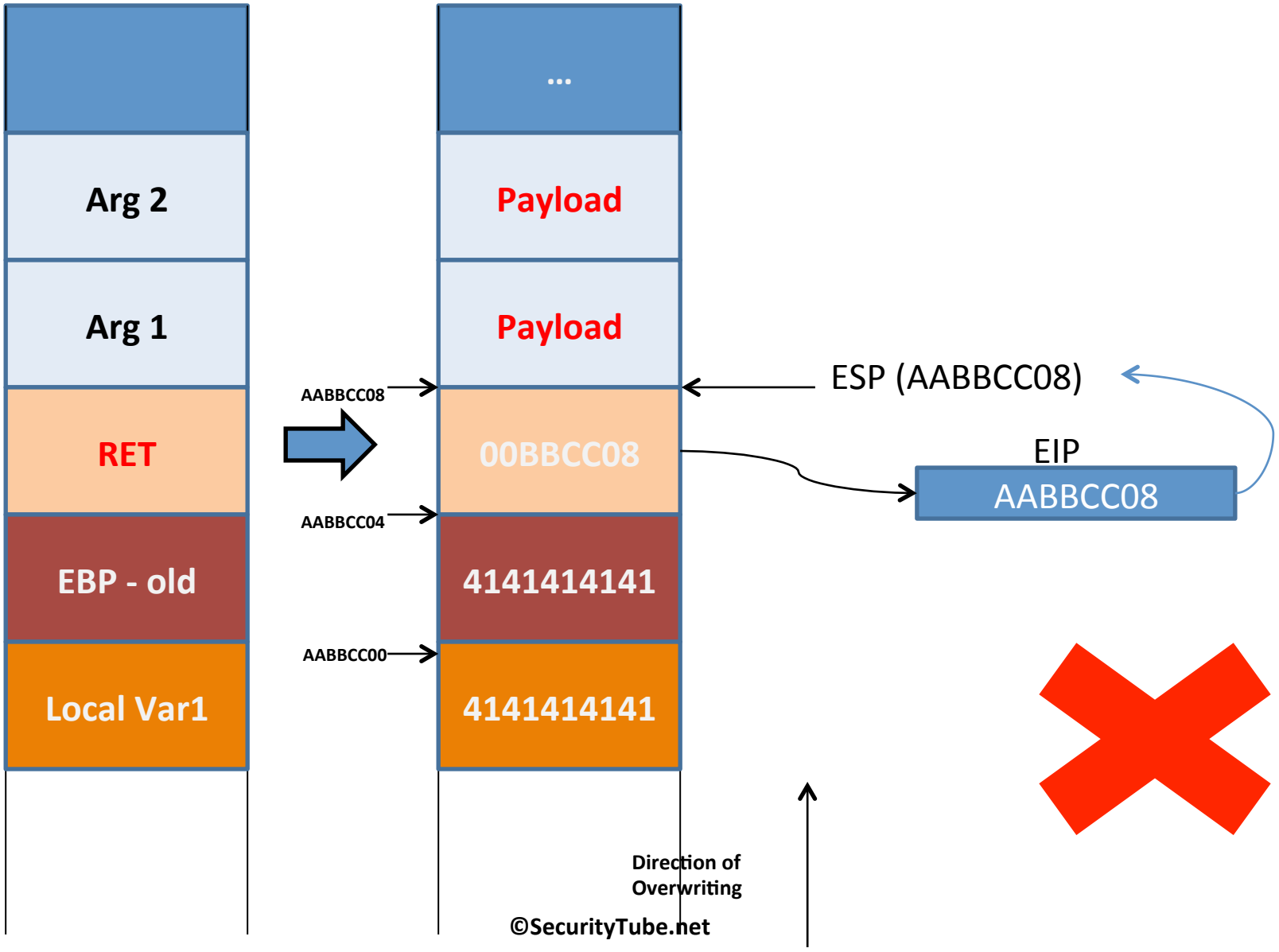
Bad Characters

- Are used to delimit input from the user
- E.g.
  - End of String input is denoted by NULL byte which is 0x00
  - End of HTTP header field is denoted by \r\n
- Bad characters break the input which we can send to the vulnerable program

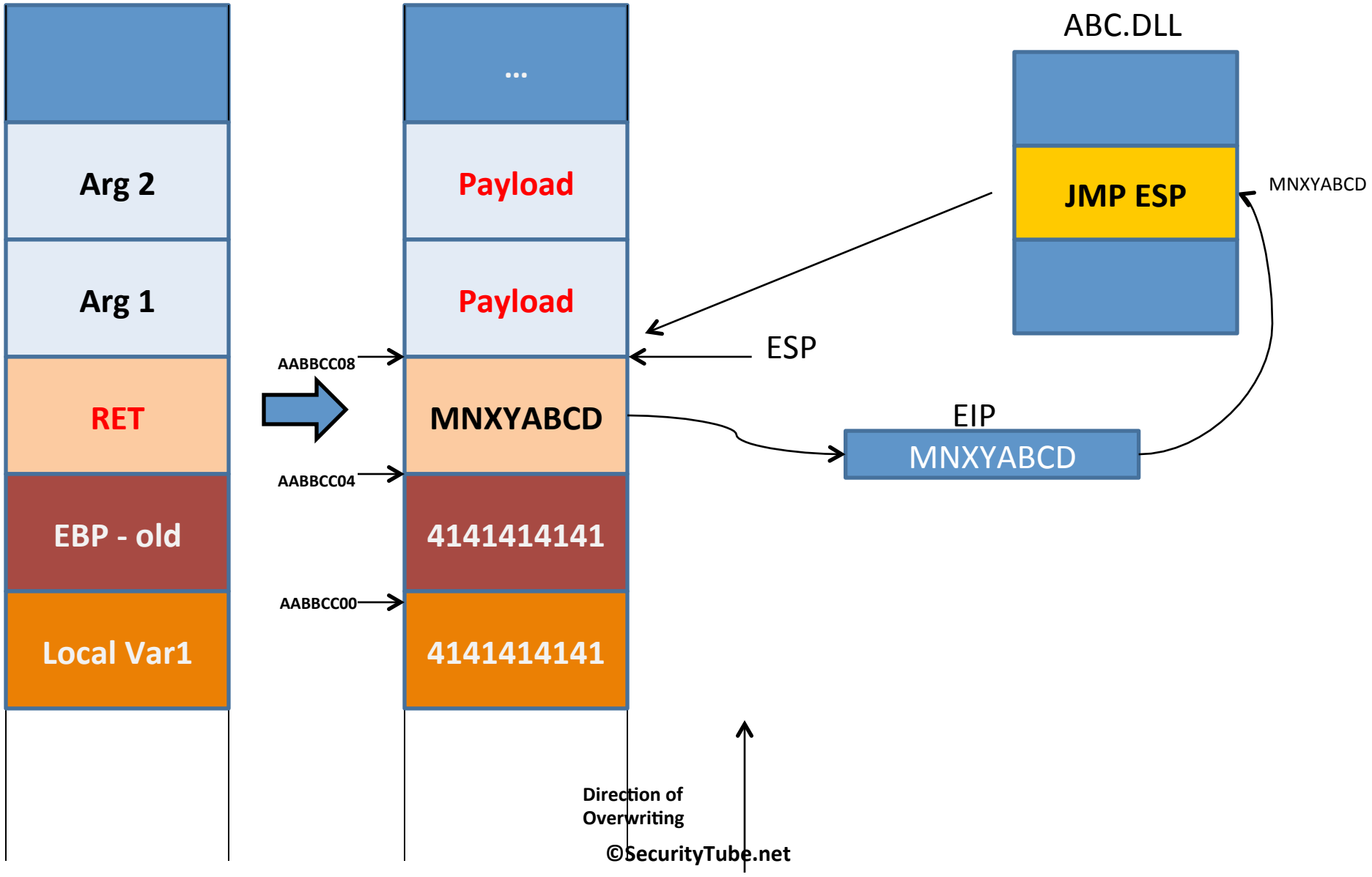
Finding Bad Characters

1. Send program the full list of characters from 0x00 to 0xFF
2. Check using debugger if input breaks
3. If so, find the character which breaks it
4. Remove that character from the list and go back to 1
5. If input no longer breaks, the rest of the characters can be used

Address Contains 0x00 in it



Successful Exploitation



Exercise

- Write a Script to Find Bad Characters in this example

- More examples:
  - http://www.securitytube.net/groups?operation=view&groupId=7
    - Buffer Overflow
    - SEH
    - …

Exercise

- What is DEP, ASLR and SafeSEH?
- Please write PyCommand scripts to find if modules have the above enabled

Module 5: Exploitation Techniques

SecurityTube Python Scripting Expert

Part 6:
Assembling, Disassembling and Searching

http://www.securitytube.net

Vivek Ramachandran
Course Instructor