

POINT, CLICK- Got Root!

HOW THE NEW, OPEN SOURCE
FRAMEWORK METHODOLOGY
AFFECTS YOUR NETWORK

LET'S LAY SOME FOUNDATIONS

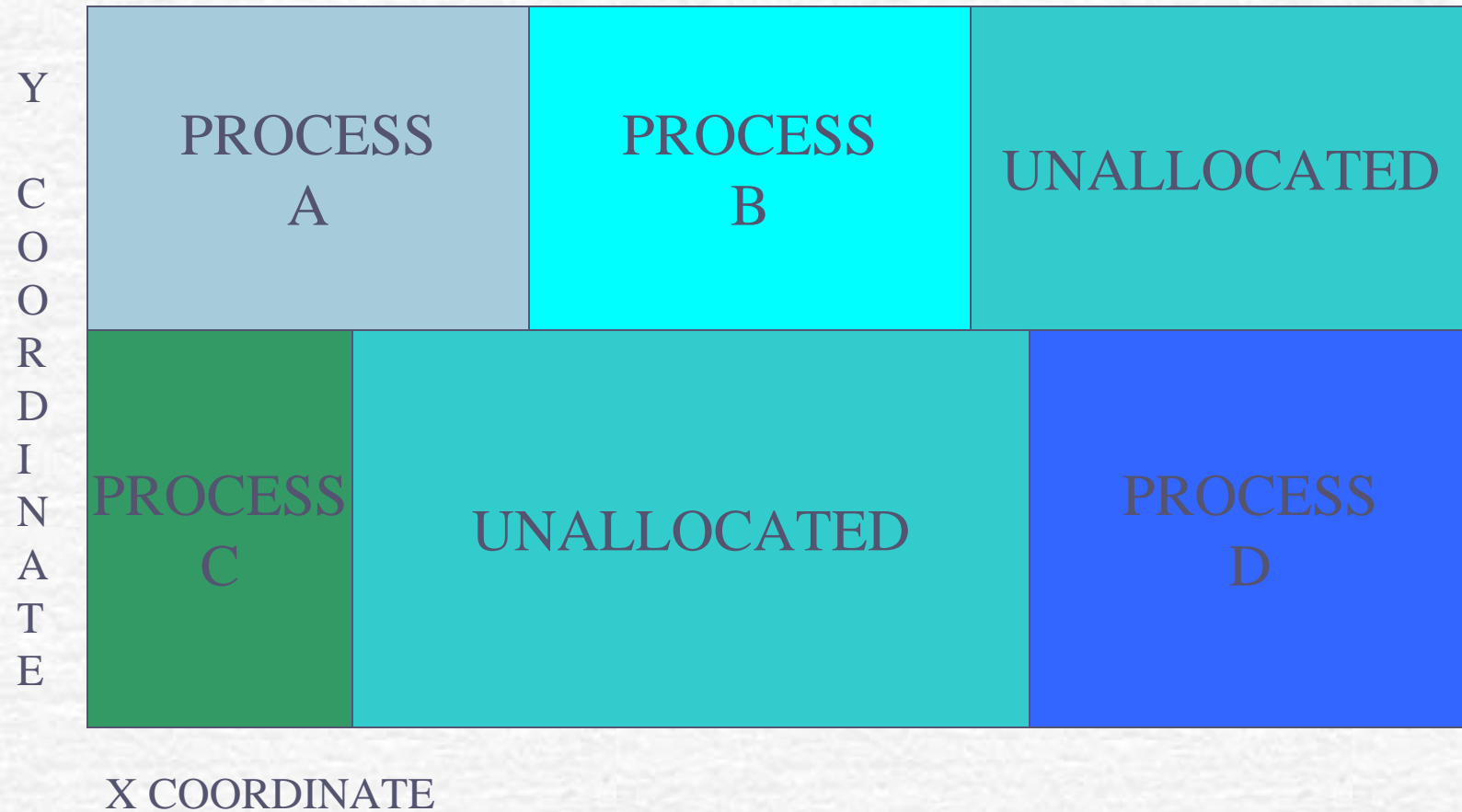
Terms

- Process code
- Initialization
- Stack and stack execution
- Pointers
- Heap
- Overflows
- SHELLCODE- NOP and NOP sleds

OLD METHODOLOGY

- ☞ Research intensive
 - Install test bed
 - OS, Apps, specific gear
 - FUZZING to death
 - Shell code slide or sled
 - Construct reverse shell
 - Or injection vector
 - Or drop "eggs"

VISUALIZE MEMORY

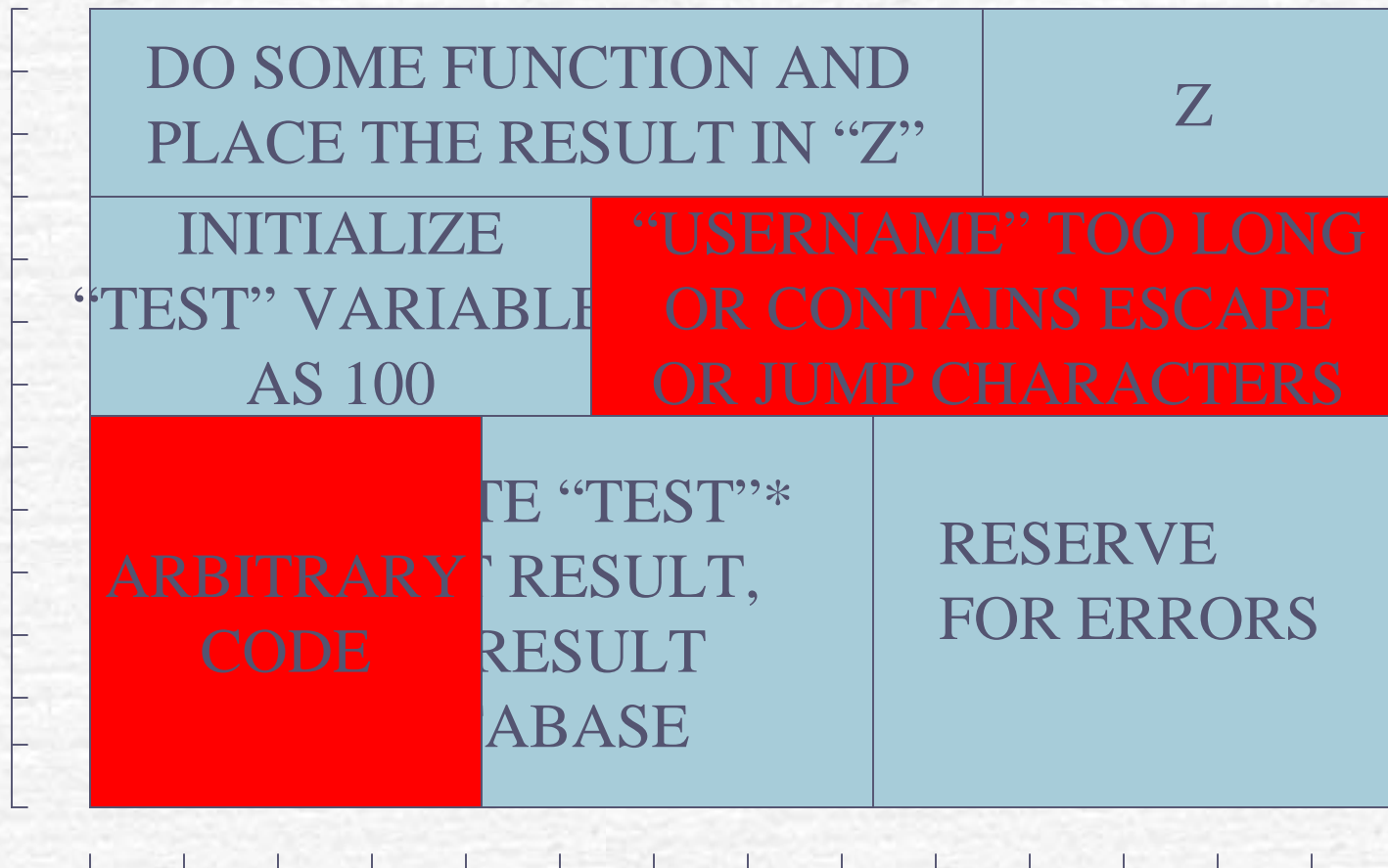


DETAIL OF PROCESS A, AS PROGRAMMED

DO SOME FUNCTION AND PLACE THE RESULT IN "Z"		Z
INITIALIZE "TEST" VARIABLE AS 100	ACCEPT INPUT "USERNAME" FROM EXTERNAL DATABASE	
CALCULATE "TEST"* "Z", PRINT RESULT, STORE RESULT IN DATABASE		RESERVE FOR ERRORS

ADDRESS

"HACKED" PROCESS A



ADDRESS

BREAKING IT DOWN

- ✔ User interface to handle options and settings
- ✔ Network connection handler
- ✔ Handler routine
- ✔ Injection vector
- ✔ SHELLCODE
- ✔ Escape vector

THIS IS A DEFINITE ART

- ☞ Deep knowledge
 - Os, coding, assembly, memory, sniffing, packet assembly
- ☞ Time and patience
- ☞ Major trial and error
- ☞ Lots of intuition

SO WHAT?



- Each exploit path takes time
- Only 1 of 10 bears fruit
- Limited number of masters
- Dissemination time lag

ENTER THE FRAMEWORK

Self contained

- No GUI dev time
- No handler dev time
- No socket dev time
- Ready-made memory interpretation
- Pre-packaged error handling
- Verbose debugging included

ELIMINATES THE TEDIUM

- Common easy language- PERL
- Pre-written support for debugging, logging, NOPS, timeouts, SSL, etc.
- API for module loading and updates
- Clean, optimized handlers
- Open source

METASPLOIT

- ✓ Easy install

- Windows or Linux

- ✓ Install steps?

- As simple as a "double click" on the exe

"TO HECK" WITH SCREENSHOTS

➤ LET'S DO IT!

TWO MORE INTERFACES

➤ MSFCL

- Run major scripts, such as:
 - Scan class B for port 80
 - Check for IIS
 - Run Exploit- get admin
 - Add new account, upload VNC, etc

➤ MSFWEB

- Web user interface

“ROLL YOUR OWN”

MSFPESCAN

- ANALYSE exes and DLLS for pointers, calls, jumps- all the stack info you need
- Remember our “Memory Visual”? All that tedious tracking of pointers, calls, jumps and other addresses is all mapped out FOR YOU!
- Msfpayload allows custom payloads
- Robust debugging and logging
- Huge reduction of Trial and Error method

More Features!

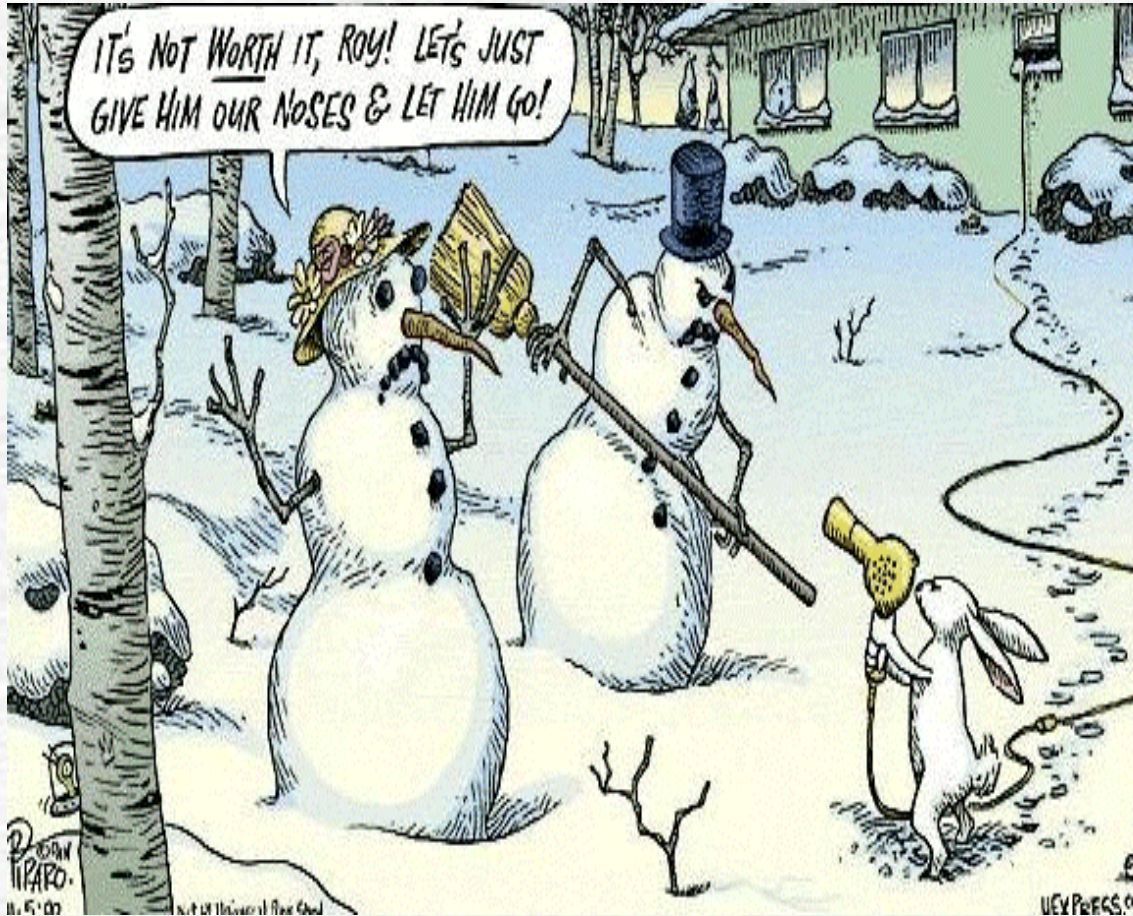
☛ InLineEgg

- Python based tool that translates your payload scripts to assembly- no need to learn assembly language!- Examples included

☛ If you prefer the C language, use ImPurity to generate your shellcode

☛ UpLoadExec- complete payload handler for uploading and starting your "Trojan of choice"

HOW THIS AFFECTS YOU



- SHORTER DEV TIMES FOR SPOIT
- LESS yet MORE PUBLICATION
- SCRIPT KIDDIES ELEVATED

WHAT CAN I DO?

☞ DOWNLOAD IT AND USE IT!

- Vulnerability Assessment
- Firewall Testing
- IDS testing
- Custom App testing

☞ Since it is Free, you have only yourself to blame

REFERENCES

- ☛ <http://www.securityfocus.com/infocus/1800> Article on the metasploit project
- ☛ Stack, pointers and memory, Lally Singh
<http://www.biglal.net/Memory.html>
- ☛ Badc0ded - how to exploit program vulnerabilities,
<http://community.core-sdi.com/~juliano/bufo.html>
- ☛ A memory Allocator, Doug lea
<http://gee.cs.oswego.edu/dl/html/malloc.html>
- ☛ Presentation on advanced exploit development at HITB, HD Moore (PDF)
http://conference.hackinthebox.org/materials/hd_moore/HDMOORE-SLIDES.pdf
- ☛ Designing Shellcode demystified, Murat
<http://www.enderunix.org/docs/en/sc-en.txt>
- ☛ Crash course user guide for Metasploit framework, (PDF)
<http://metasploit.com/projects/Framework/docs/CrashCourse-2.0.pdf>