PuttyRider

# Pivoting from Windows to Linux in a penetration test

“With great power comes great responsibility”

Adrian Furtunã, PhD

adif2k8@gmail.com

root@bt:~# Agenda

- Idea origin and usage scenario
- PuttyRider features
- Technical details
- Demo

root@bt:~# Whoami

- Technical Manager – Security Services at KPMG Romania
- Leading the penetration testers team
- PhD, OSCP, CEH
- Speaker at security events and conferences:
  - ZeroNights
  - Hack.lu
  - Hacktivity
  - Defcamp, OWASP Romania, etc

root@bt:~# Whoami

- Technical Manager – Security Services at KPMG Romania
- Leading the penetration testers team
- PhD, OSCP, CEH
- Speaker at security events and conferences:
  - ZeroNights
  - Hack.lu
  - Hacktivity
  - Defcamp, OWASP Romania, etc

We're Hiring!

2 x Penetration Tester

root@bt:~# How come this idea?

# Internal penetration test

# Gained Domain Admin

# Target data was located on Linux/Unix servers

# No obvious vulnerabilities on target servers

root@bt:~# How come this idea?

- Internal penetration test
- Gained Domain Admin
- Target data was located on Linux/Unix servers
- No obvious vulnerabilities on target servers
- How to gain access to data?

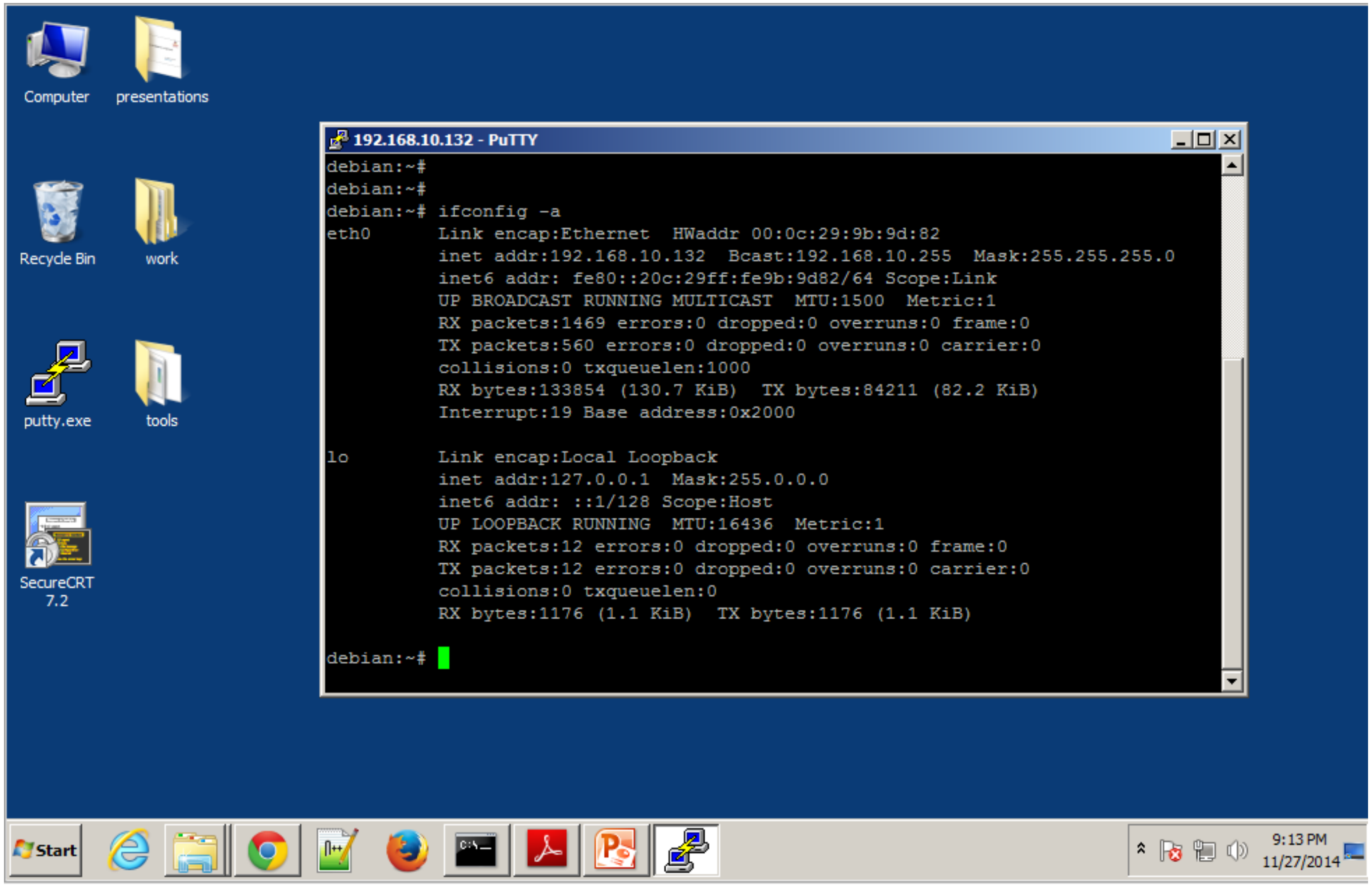
root@bt:~# How come this idea?

- Internal penetration test
- Gained Domain Admin
- Target data was located on Linux/Unix servers
- No obvious vulnerabilities on target servers
- How to gain access to data?

Target the sysadmins ;)

root@bt:~# Get access to sysadmin’s machine

- Metasploit + Domain Admin credentials
- meterpreter> screenshot



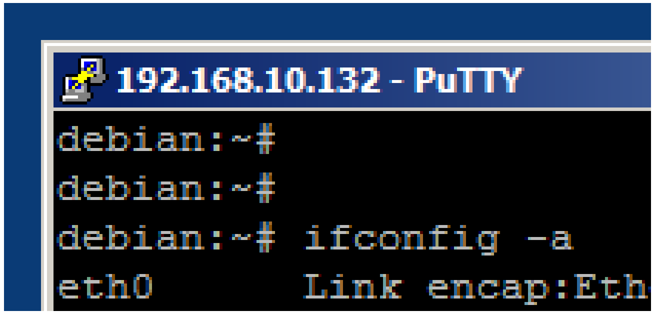
```
debian:~#
debian:~#
debian:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:0c:29:9b:9d:82
          inet addr:192.168.10.132  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe9b:9d82/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1469 errors:0 dropped:0 overruns:0 frame:0
          TX packets:560 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:133854 (130.7 KiB)  TX bytes:84211 (82.2 KiB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1176 (1.1 KiB)  TX bytes:1176 (1.1 KiB)

debian:~#
```

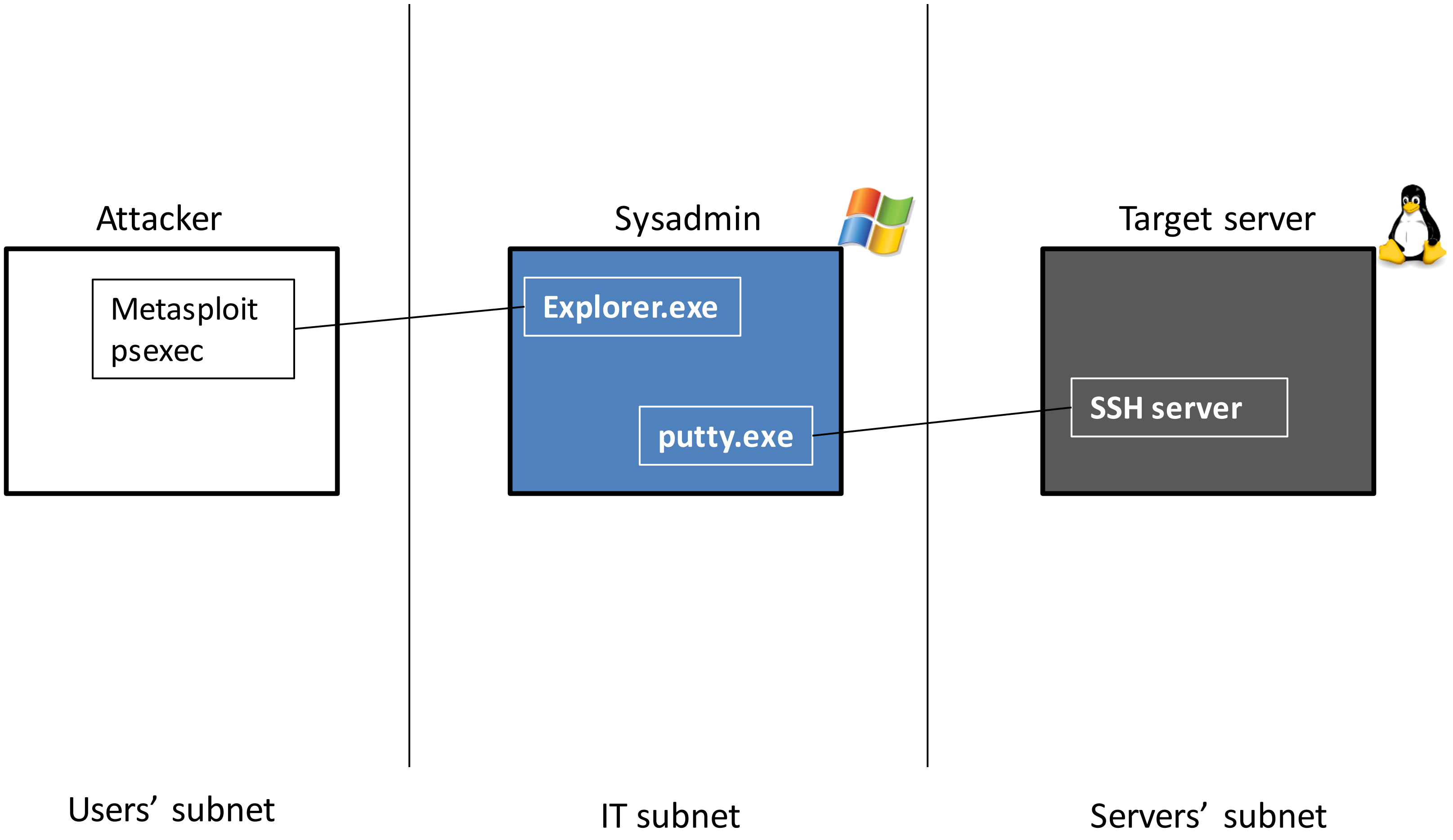
root@bt:~# We’re closer to the target

- The admin is already connected to our target server with Putty
- How to take advantage of this?



```
debian:~#
debian:~#
debian:~# ifconfig -a
eth0      Link encap:Eth
```

root@bt:~# Something is missing…

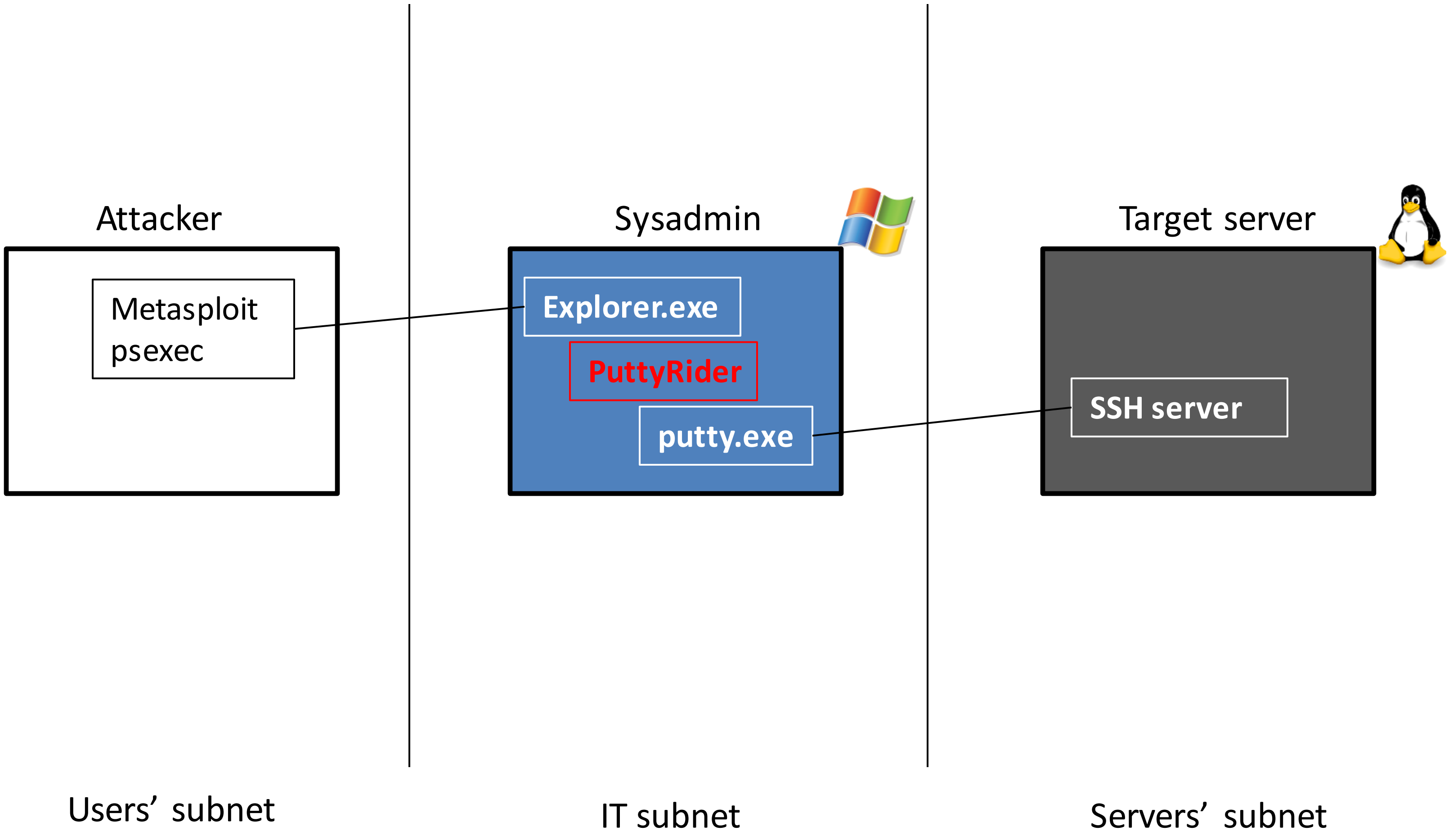


root@bt:~# Introducing PuttyRider

# 'Ride' the existing Putty connection:

- > Sniff all conversation between admin and server
  - ▪ Passwords included
- > Inject shell commands in the existing session
  - ▪ Run as the current user
- > Transparent to the victim user
- > Interactive with the attacker (ex. via remote shell)
- > Independent of connection protocol (SSH, Telnet, Rlogin)

root@bt:~# PuttyRider - the missing piece



root@bt:~# PuttyRider - feature list

# Operation modes:

> List existing Putty processes

> Inject PuttyRider DLL in a certain Putty process

> Wait in backgroud for new Puttys and inject in those also

> Automatically execute a shell command after injection

> Eject DLL from all Putty processes

# Output modes:

> Write all conversation to a file on local machine

> Initiate a reverse connection to attacker’s machine and start an interactive session

Technical Details

root@bt:~# Previous work

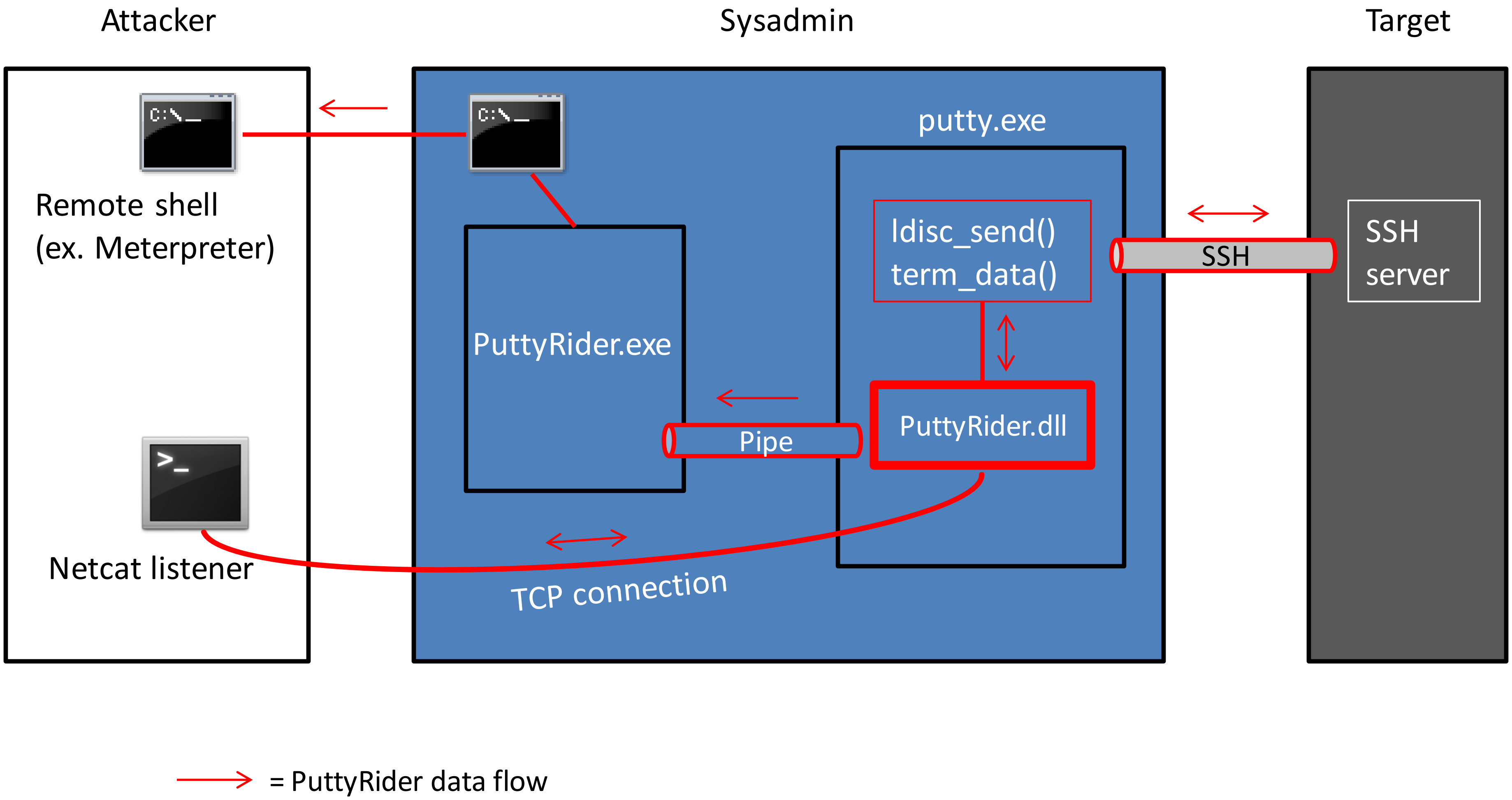
- PuttyHijack[1]
  - Written by Brett Moore, Insomnia Security – 2008
  - Uses DLL injection to hijack the Putty process
  - Only works for Putty 0.60 (2007)
    - Other Putty versions crash
    - Latest Putty is 0.63
  - Just a proof of concept with multiple limitations
    - Cannot specify in which Putty process to inject
    - No information about the connection endpoints
    - Hard to maintain/extend code (C + ASM mixture)

[1] https://www.insomniasec.com/downloads/tools/PuttyHijackV1.0.rar

root@bt:~# PuttyRider – technical overview

- Started from the PuttyHijack proof of concept
  - DLL injection
  - Function hooking
- Written in C on Windows with Win32 API
- Works on all recent versions of Putty
- Works on recent operating systems (Windows 8, Windows 7, etc)
- Does not require administrative privileges

root@bt:~# PuttyRider architecture



root@bt:~# Interesting functions in Putty

- # `ldisc_send()`
  - > Defined in ldisc.c
  - > "ldisc.c: PuTTY line discipline. Sits between the input coming from keypresses in the window, and the output channel leading to the back end."
  - > `void ldisc_send(void *handle, char *buf, int len, int interactive)`

- # `term_data()`
  - > Defined in terminal.c
  - > This source file implements a terminal emulator
  - > The function handles data to be sent to the terminal window
  - > `int term_data(Terminal *term, int is_stderr, const char *data, int len)`

root@bt:~# Finding address of internal Putty function

# Putty.exe does not export any function

# Cannot use LoadLibrary & GetProcAddress

root@bt:~# Finding address of internal Putty function

# Putty.exe does not export any function

# Cannot use LoadLibrary & GetProcAddress

# Downloaded source code & binaries of all Putty versions

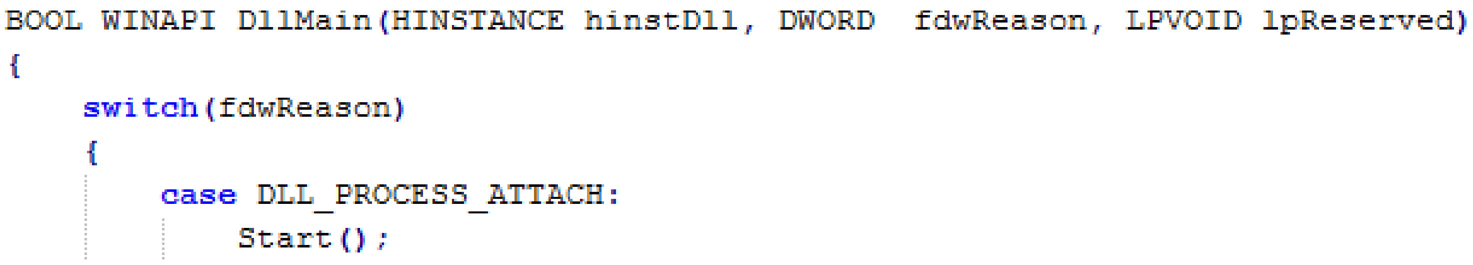
# Extracted binary signatures that match the beginning of target functions

> `ldisc_send()` - 1 signature – 21 bytes - covers Putty 0.54 .. 0.63

> `term_data()` - 3 signatures – 18 bytes - cover Putty 0.54 .. 0.63

# Memory search the .text section of putty.exe for the signatures

root@bt:~# Function hooking

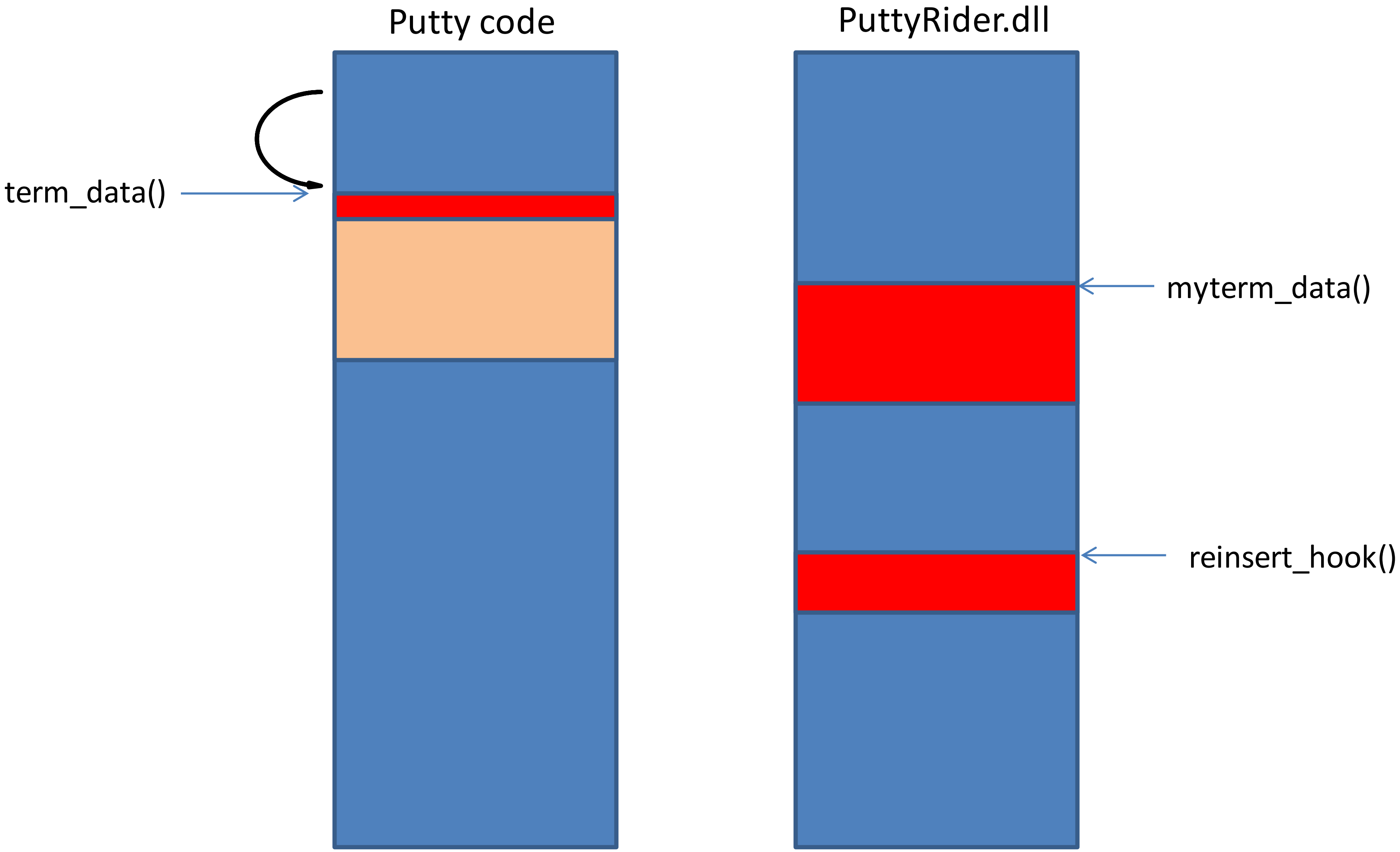


```
BOOL WINAPI DllMain(HINSTANCE hinstDll, DWORD  fdwReason, LPVOID lpReserved)
{
    switch(fdwReason)
    {
        case DLL_PROCESS_ATTACH:
            Start();
```

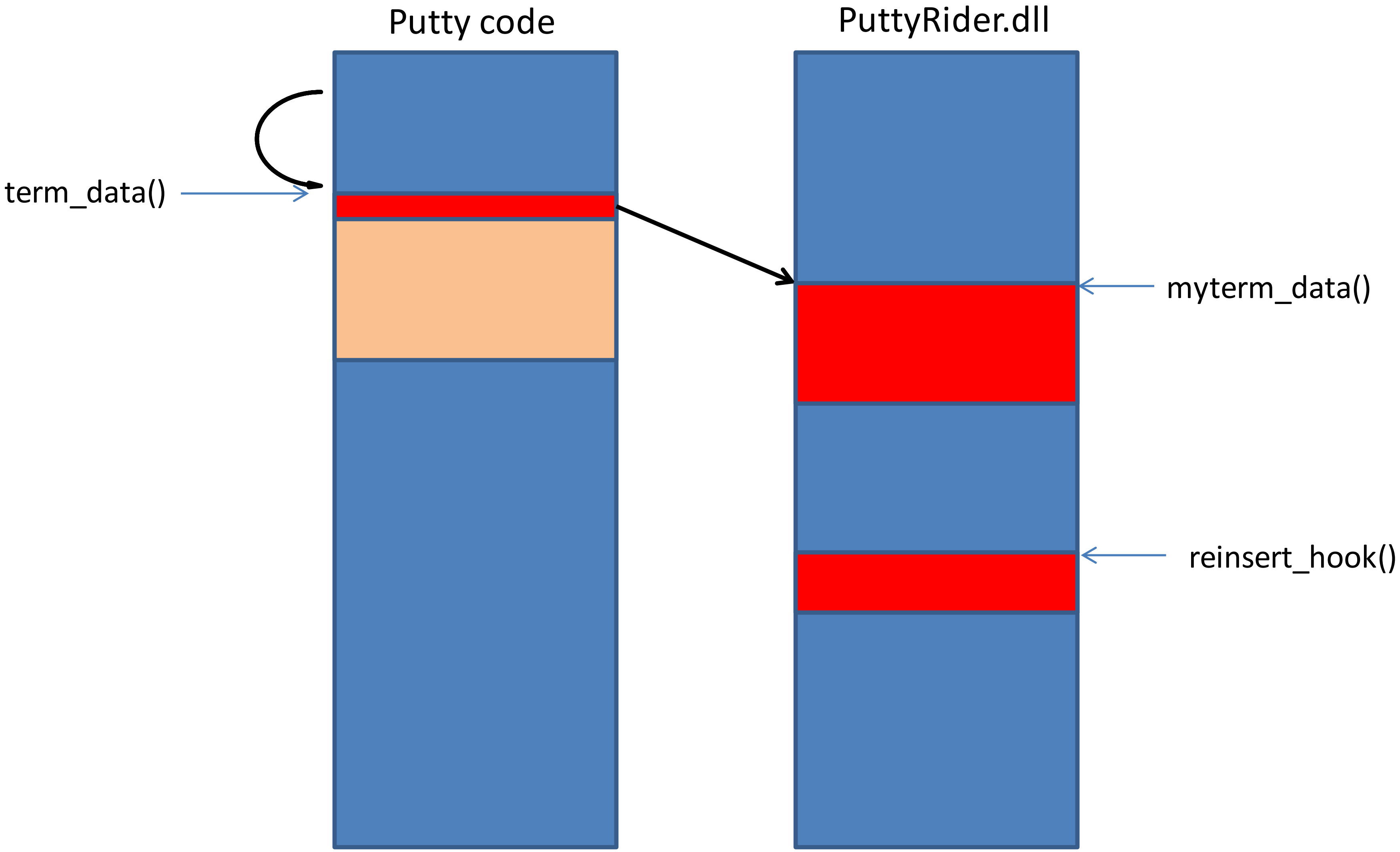
- Locate starting address of `ldisc_send()` and `term_data()` in .text section of putty.exe
- Define `myldisc_send()`
- Define `myterm_data()`
- Overwrite first bytes of `ldisc_send()` with “`JMP addr_of_myldisc_send`”
- Overwrite first bytes of `term_data()` with “`JMP addr_of_myterm_data`”

(translated to opcode)

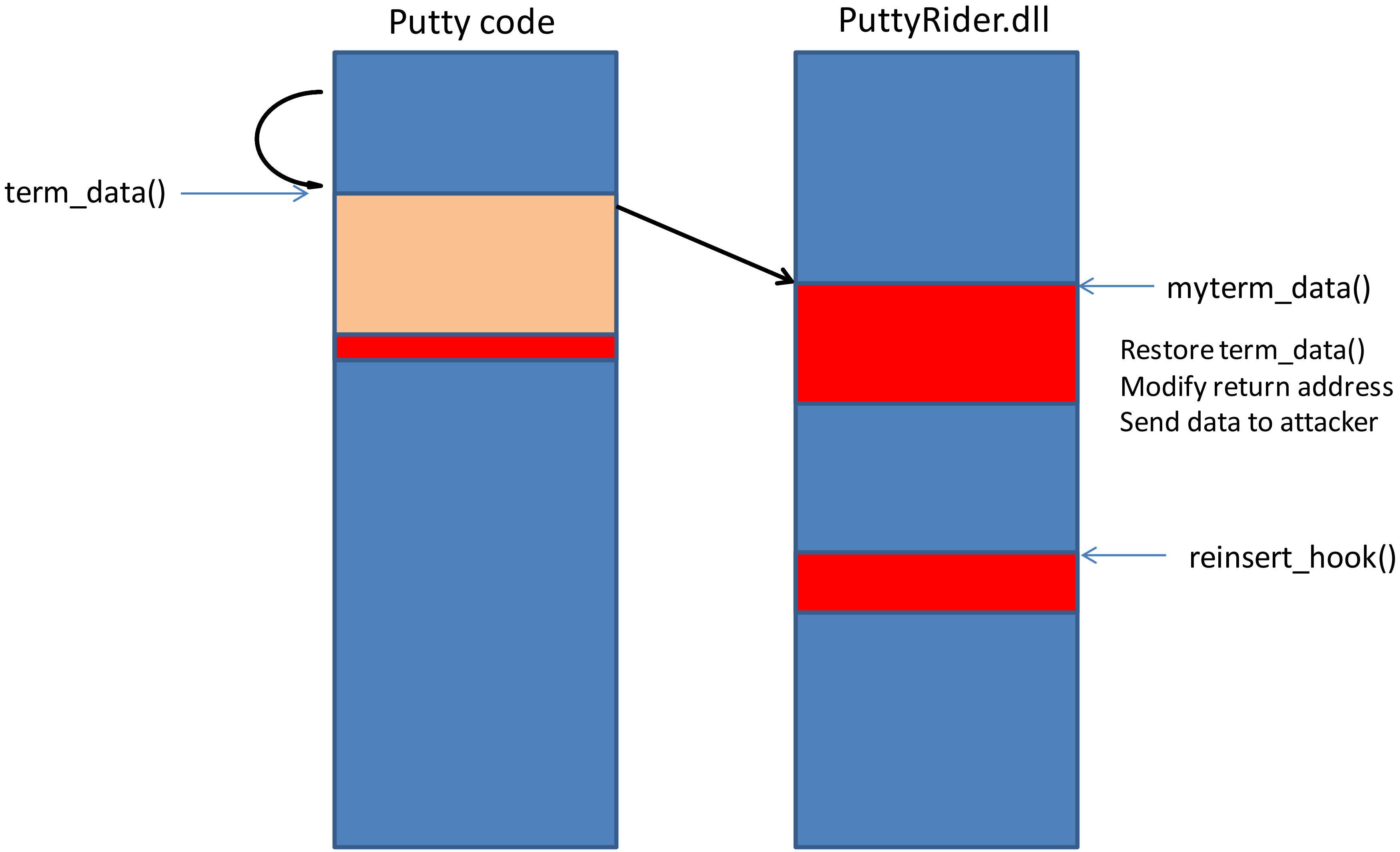
root@bt:~# Hooking in action



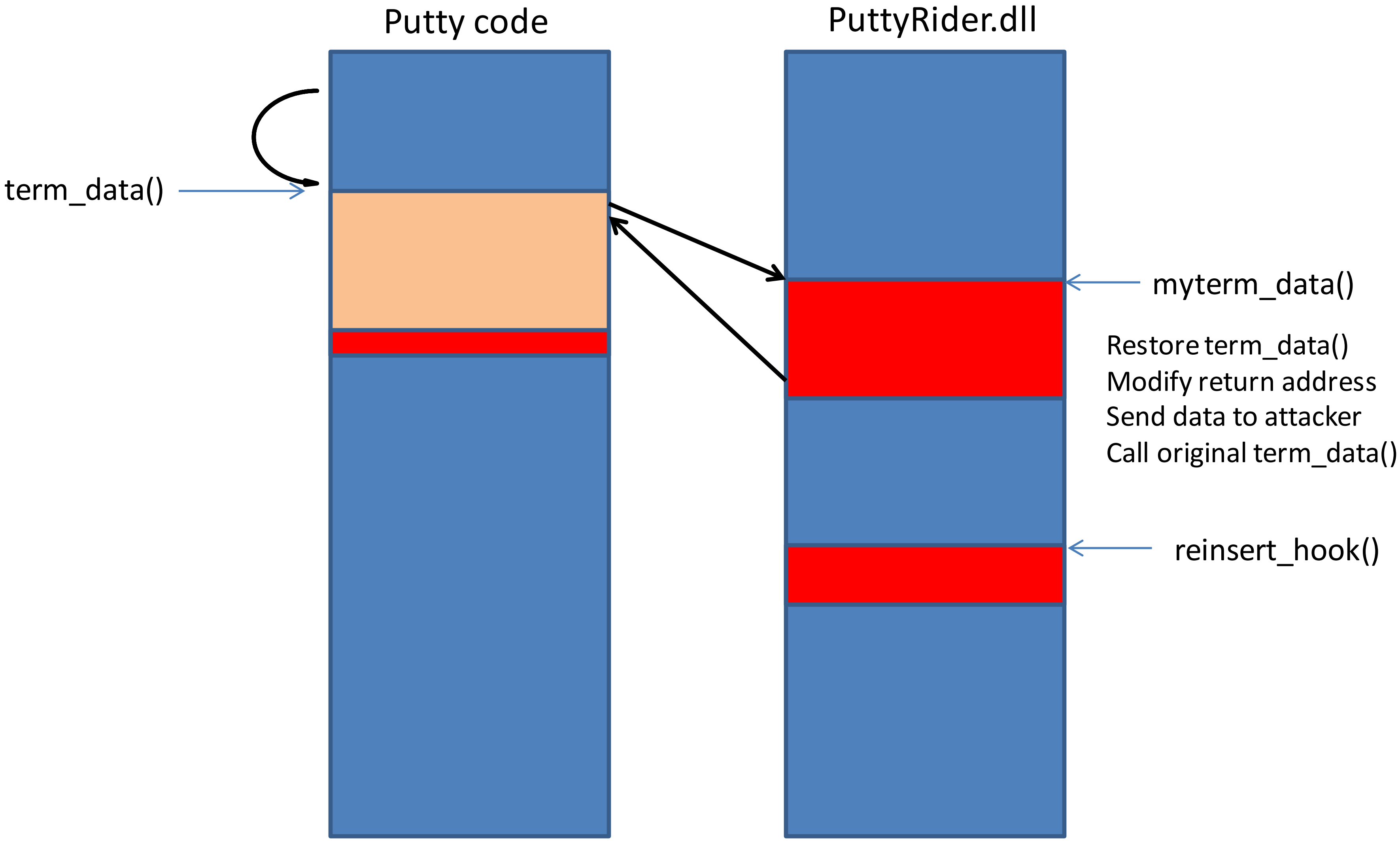
root@bt:~# Hooking in action



root@bt:~# Hooking in action

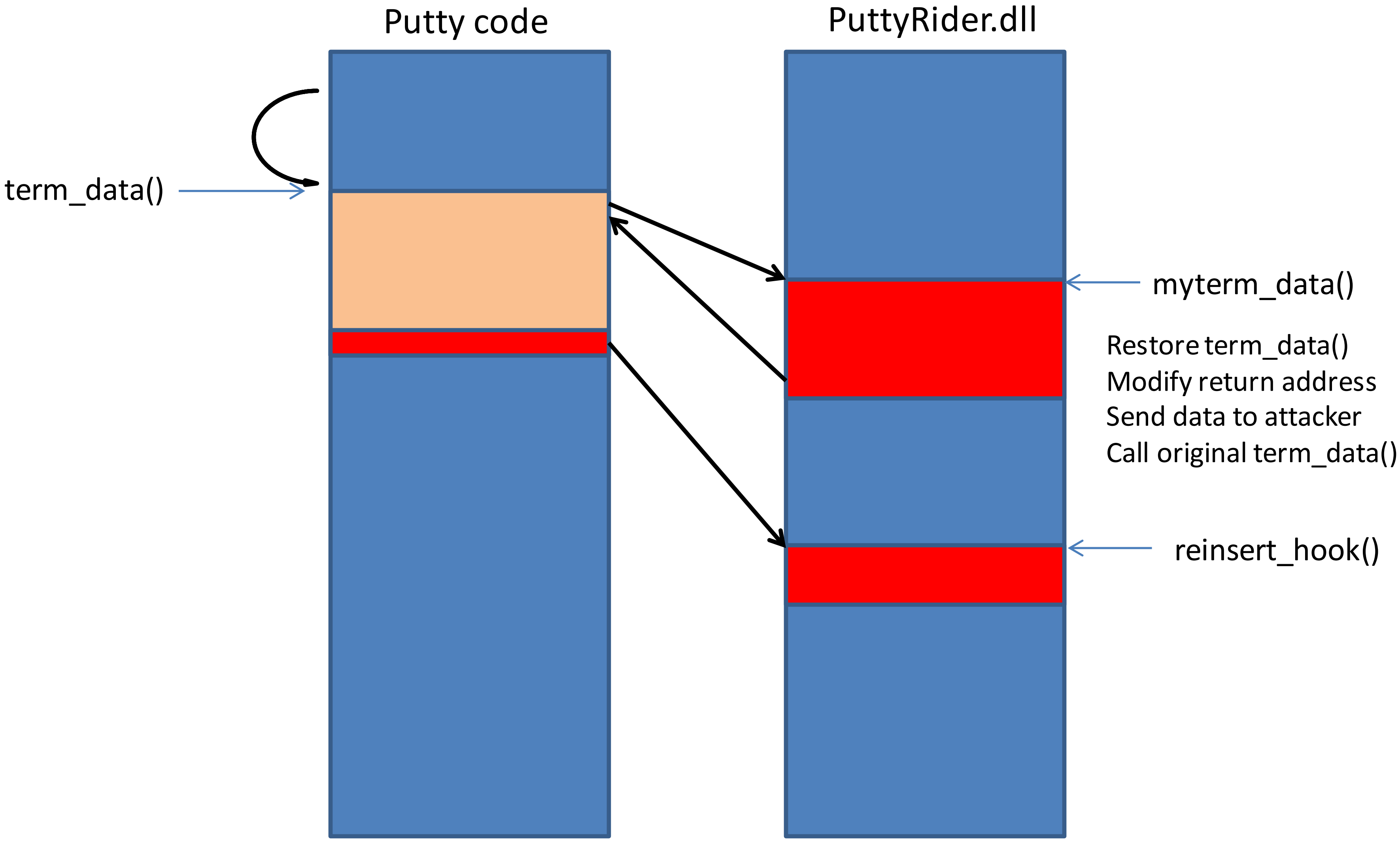


root@bt:~# Hooking in action



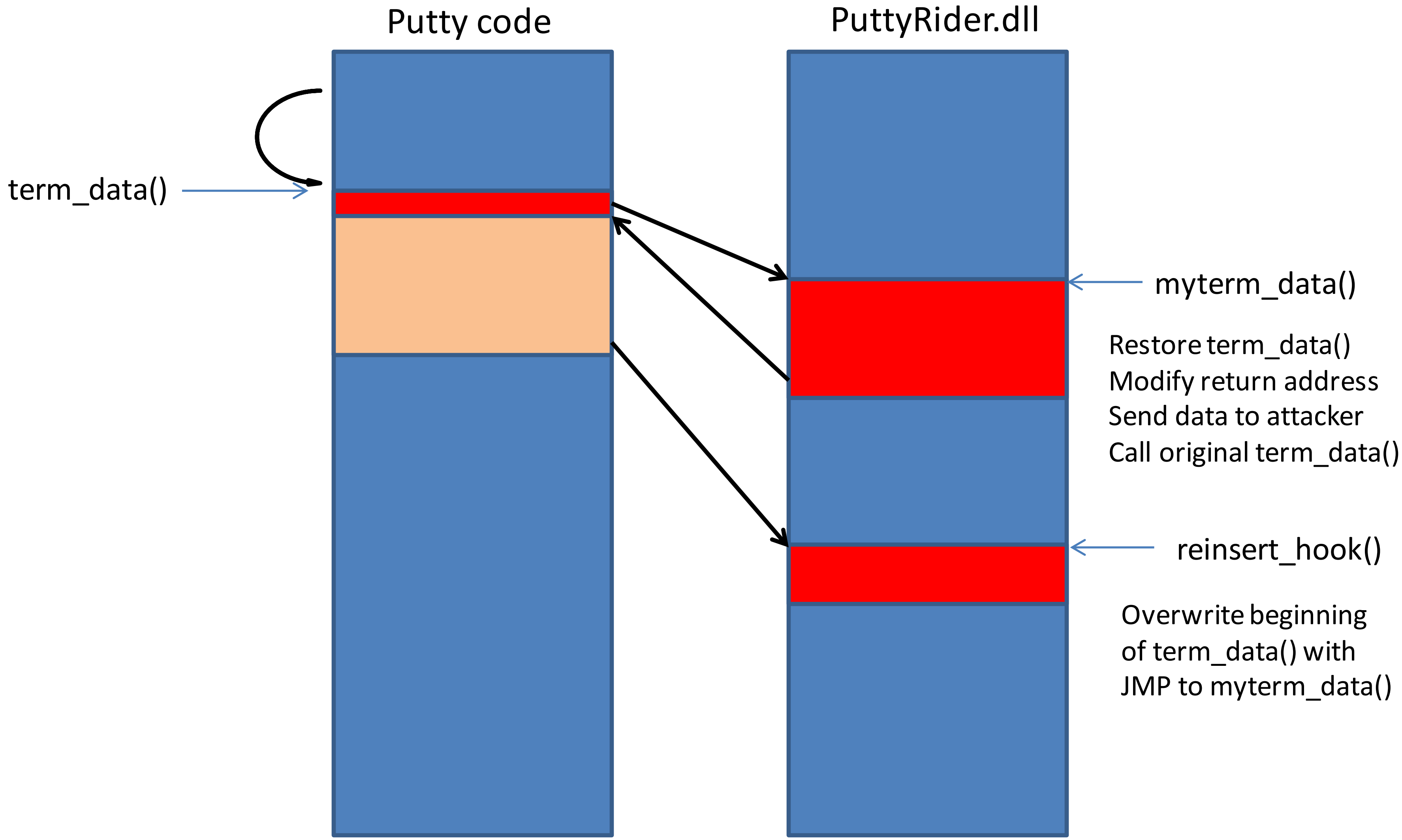
Restore term_data()
Modify return address
Send data to attacker
Call original term_data()

root@bt:~# Hooking in action



Restore term_data()
Modify return address
Send data to attacker
Call original term_data()

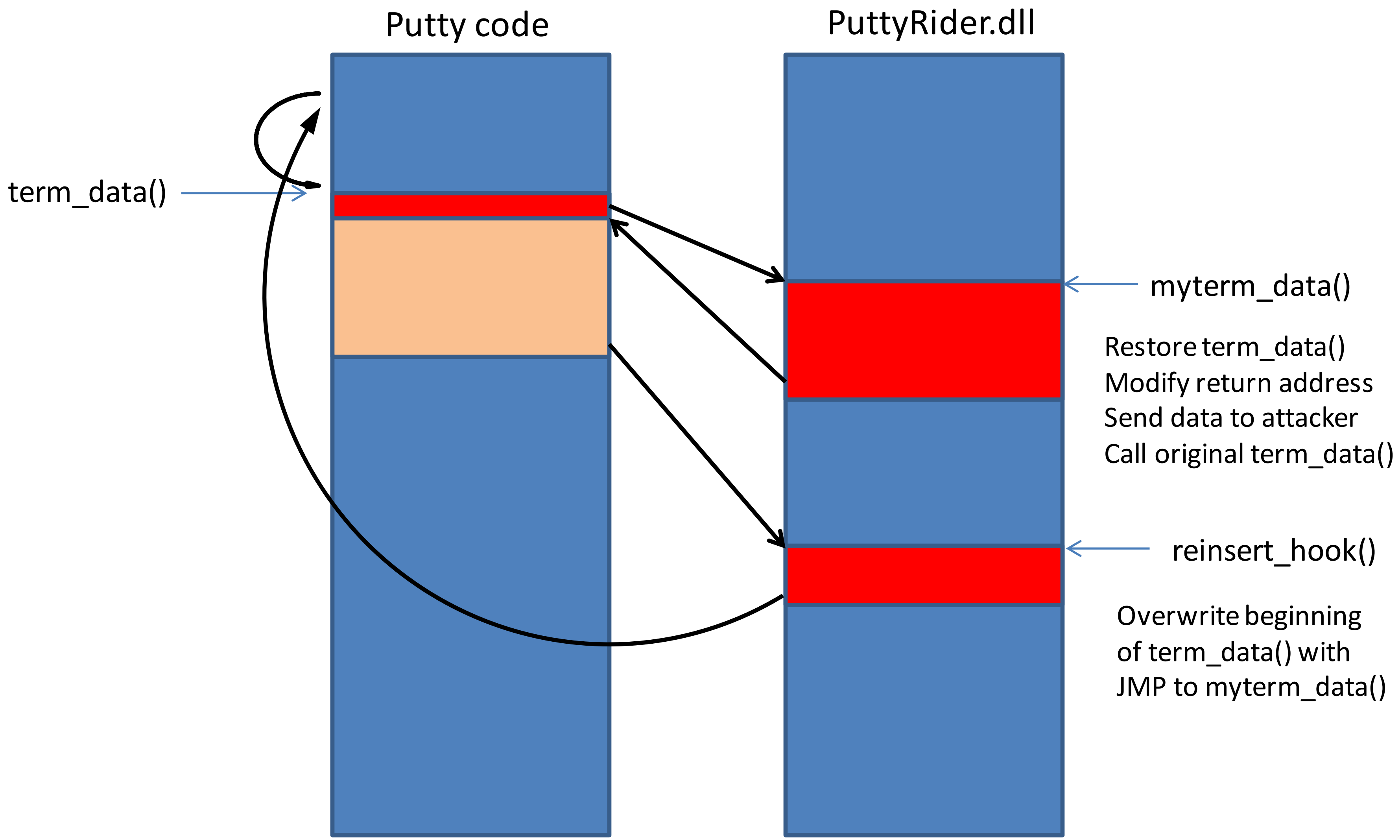
root@bt:~# Hooking in action



Restore term_data()
Modify return address
Send data to attacker
Call original term_data()

Overwrite beginning of term_data() with JMP to myterm_data()

root@bt:~# Hooking in action



Restore term_data()
Modify return address
Send data to attacker
Call original term_data()

Overwrite beginning of term_data() with JMP to myterm_data()

root@bt:~#

Demo

root@bt:~# Download PuttyRider

# Will be available on https://github.com/ in a few days

Q & A

Thank you!

Adrian Furtunã, PhD

adif2k8@gmail.com