PER ALTRI APPUNTI CONSULTARE IL SITO:

https://luigi-v.github.io/Appunti_Universita/

Per altri appunti consultare il sito: *https://luigi-v.github.io/Appunti_Universita/*

By *Vollono Luigi*

1. TERMINOLOGIA

Verranno introdotti una serie di termini che vengono usati successivamente:

- ***Asset***: è una entità generica che può interagire col mondo circostante e la sua natura dipende dal contesto. Potrebbe essere un dispositivo hardware, software, un dato, un algoritmo o anche una persona. Nel nostro caso l'***asset*** sarà il ***software*** che può interagire con l'utente e bisogna capire in che modo avviene l'interazione. Un utente può interagire con un asset in tre modi: correttamente, non correttamente in modo involontario o non correttamente in modo malizioso. Un *uso non corretto di un asset* può comportare rischi gravi, tra cui: furto di dati sensibili (password o conti correnti), beni preziosi o denaro, modifica o distribuzione di informazioni sensibili, compromissione di servizi;
- ***Minaccia (threat)***: è una potenziale causa di incidenti, il risultato è un danno all'asset. Le minacce possono tramutarsi in due mod: accidentale o doloso. Microsoft propose una classificazione delle minacce sotto il nome di ***STRIDE***:
  - ***S****poofing*, quando un utente si spaccia per un'altra entità;
  - ***T****ampering*, modifica non autorizzata delle informazioni;
  - ***R****epudiation*, ripudiare o negare l'esecuzione di un azione;
  - ***I****nformation Disclosure*, divulgazione di informazioni sensibili;
  - ***D****enial of Service*, negazione di un servizio;
  - ***E****levation of Privilege*, elevazione dei privilegi.
- ***Attaccante***: colui che interagisce con l'asset in modo malizioso e doloso, cercando di ottenere un fine. L'attaccante è alla ricerca di un malfunzionamento sfruttabile che si possa tramutare in un exploit. Ha lo scopo di tramutare una minaccia in realtà, motivato dal conseguire un vantaggio. Esistono vari tipi di attaccanti tra cui:
  - *White hat* (*ethical hacker*): viola asset per fini non maliziosi, come stimare il livello di sicurezza;
  - *Black hat*: viola asset per fini maliziosi o per tornaconto personale;
  - *Gray hat*: viola asset e, in cambio di denaro, si offre di irrobustirli;
  - *Hacktivist*: viola asset per fini ideologici, politici o religiosi. Svolge attività di cyber terrorismo e rende accessibili al pubblico documenti confidenziali;
  - *Nation state*: team di attaccanti sponsorizzati da una nazione;
  - *Organized criminal gang*: team di attaccanti che viola asset per profitti illegali.
- ***Bug***: è un errore di implementazione dell'asset;
- ***Difetto***: è una deviazione dell'asset da requisiti e specifiche di progetto;
- ***Debolezza (weakness)***: è un difetto che potrebbe rendere reale una minaccia, la presenza di una debolezza non indica necessariamente che l'asset possa essere compromesso;
- ***Vulnerabilità***: è una debolezza che un attaccante è in grado di usare per tramutare una minaccia in realtà. È la somma di tre fattori:
  - *Debolezza esistente* (weakness);
  - *Accessibilità* dell'attaccante alla debolezza;
  - *Capacità di sfruttare la debolezza* per conseguire un vantaggio.
- ***Exploit***: è una procedura che è in grado di sfruttare una vulnerabilità. Una volta che l'attaccante ha individuato una debolezza, la trasforma in una vulnerabilità sfruttabile, dopodiché è in grado di causare un comportamento inatteso dell'asset, ottenendo un vantaggio.
- ***Vettore di attacco***: è uno strumento qualsiasi attraverso il quale si può veicolare una vulnerabilità (modalità di attacco). Può essere una connessione TCP verso un server, una shell locale, una linea telefonica incustodita;
- ***Superficie di attacco***: è l'insieme di tutti i suoi vettori di attacco. Esso misura l'esposizione dell'asset agli attacchi.
- ***Politica di sicurezza (security policy)***: definisce in modo non ambiguo il livello di sicurezza di un asset. Risponde ad una serie di domande del tipo: Che significa che "l'asset è sicuro"? Da quali interazioni ci si vuole difendere? Da quali utenti ci si vuole difendere?. La politica di sicurezza nasce spesso da un'*analisi dei rischi* (*risk analysis*) che cerca di identificare cosa potrà andare storto con l'asset, quanto sarà probabile un incidente o quanto costerà un incidente.
- ***Meccanismi di sicurezza***: è uno strumento che consente di attuare una politica di sicurezza. Ci sono tre diverse categorie:
  - *Meccanismi di prevenzione*, tendono ad impedire le interazioni tra un asset e un utente, come un firewall;
  - *Meccanismi di rilevazione*, controllano le interazioni tra un asset e un utente;
  - *Meccanismi di reazione*, per ripristinare il sistema in seguito ad un incidente.

  Operazioni tipiche dei meccanismi di sicurezza sono:
  - *Autenticazione*, verificare che l'utente che si presenta all'asset è effettivamente chi dice di essere;
  - *Controllo degli accessi*, verificare che l'utente ha i diritti per accedere all'asset;
  - *Auditing*, monitorare e registrare le interazioni di un utente con l'asset;
  - *Azione*, svolgere azioni correttive per far rispettare la politica di sicurezza.
- ***Prevenzione***: un asset soggetto a prevenzione non è in grado di interagire con nessuno. L'aspetto positivo è che non è attaccabile dai malintenzionati, ma l'aspetto negativo è che non è utilizzabile neanche dagli utenti normali. È necessario un aumento dell'esposizione dell'asset affinché gli utenti possano fruirne, come l'apertura di porte TCP in un servizio di rete, piuttosto che disconnessione totale, e una lettura di input in una applicazione locale, piuttosto che nessun input.
- ***Rilevazione***: con l'aumento dell'esposizione aumentano anche i rischi. I rischi vanno controllati con meccanismi di rilevazione, come il controllo del traffico sulle porte TCP aperte o il controllo degli input passati a una funzione.

ASSET E SICUREZZA:

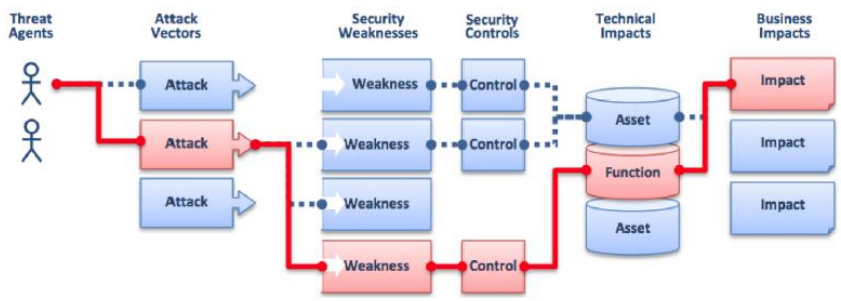
Per quanto riguarda la ***sicurezza di un asset***, dipende da che funzionalità offre un asset. Un asset non esposto al pubblico e che non offre funzionalità è più sicuro, ma molto inutile. Pertanto, si vuole un asset che offre funzionalità ma che sia ben protetto, questi tipi di asset però possono incorrere in abusi non autorizzato. Un esempio di abuso includono la violazione della triade ***CIA***:

- ***Confidentiality***: è stato causato un accesso in lettura non autorizzato, ovvero l'utente sfruttando l'asset riesce ad accedere a dati sensibili. Pertanto, bisogna impedire l'interazione in lettura tra un asset e un utente non autorizzato;
- ***Integrity***: è stato causato un accesso, non solo in lettura, ma anche in scrittura, non autorizzato. Pertanto, bisogna impedire l'interazione in scrittura tra asset e un utente non autorizzato;
- ***Availability***: bisogna rendere disponibili le funzioni di un asset a utenti esplicitamente autorizzati.

La proprietà più importante da considerare è l'***integrità*** rispetto alla confidenzialità, ad esempio un software che gestisce un conto corrente non si vuole rendere noto all'attaccante quanti soldi si ha sul conto, ma soprattutto a modificare la quantità corrente. In alcuni scenari la *disponibilità* può essere di intralcio, cioè la privacy di un utente può implicare la mancata disponibilità di informazioni e servizi a terzi, in quanto per proteggere la privacy si impedisce a terzi di utilizzare determinati servizi.

COME OPERA UN ATTACCATE:

Un attaccante potrebbe avere, individuato in una superficie di attacco, un determinato vettore di attacco che può sfruttare una debolezza nella sicurezza e gli faccia ottenere il controllo di alcune funzionalità dell'asset e gli porta una conseguenza che può essere furto di dati, impedire servizi, ecc…

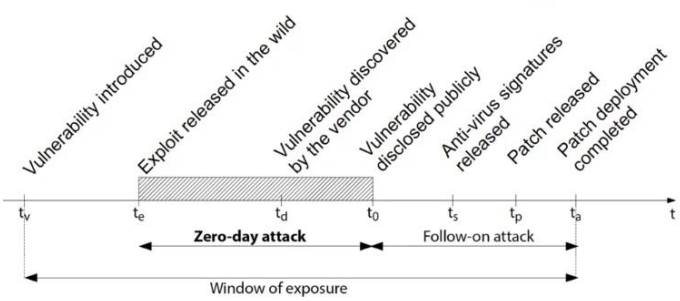


2. CATALOGAZIONE DELLE VULNERABILITÀ

Per ***vulnerabilità del software*** si intende una debolezza presente, comprensibile e sfruttabile da un attaccante. Il *ciclo di vita* di una vulnerabilità ha diverse fasi:

1. (t_v) Rilascio di un software, o di una nuova versione, che presenta una vulnerabilità;
2. (t_e) Scoperta della vulnerabilità da parte di un attaccante, rilascio di un exploit per sfruttare la vulnerabilità non notificando il fornitore;
3. (t_d) Scoperta della vulnerabilità da parte del fornitore, o tramite segnalazione, viene notificato il problema;
4. (t_0) Divulgazione della vulnerabilità al pubblico;
5. (t_s) Rilevazione dell'exploit da parte degli antivirus, aggiornando i sistemi di protezione in modo da poter rilevare l'exploit;
6. (t_p) Rilascio di una patch da parte del venditore;
7. (t_a) Mitigazione dell'exploit su tutti i sistemi.

NOTA:
Tra la 2 e 4 fase, nell'intervallo [t_e, t_0], la vulnerabilità è stata sfruttata all'insaputa di tutti, questo è chiamato ***zero-day attack***.
Tra la 4 e 7 fase, nell'intervallo [t_0, t_a], l'attacco avviene in presenza della conoscenza pubblica della vulnerabilità ed è chiamato ***follow-on attack***.
Più vicini si è allo zero-day, più è probabile che un attacco al software abbia successo.



È importante tenere traccia delle vulnerabilità, perché venire a conoscenza di un exploit che funzioni prima che la vulnerabilità venga resa nota, da una garanzia di successo dell'attacco siccome non sono ancora note delle contromisure.

È importante avere un catalogo aggiornato di possibili vulnerabilità e ciò viene fatto tramite:

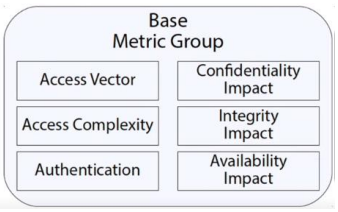
- ***Enumerazione***: costruzione di una tupla univoca per ciascuna vulnerabilità (id, tipo, vettore di attacco, minaccia, exploit);
- ***Catalogazione***: inserimento della tupla in un archivio;

In passato, sono stati proposti diversi archivi indipendenti da diverse aziende. Una stessa vulnerabilità venne catalogata per 12 volte da team differenti.

Nel 1999 venne creato il ***MITRE***, un ente no-profit, che ha introdotto un catalogo uniforme delle vulnerabilità chiamato ***CVE*** (***Common Vulnerability Exposures***). Le vulnerabilità presenti nell'archivio violano almeno una proprietà della triade CIA, sono identificate da una stringa univoca CVE-ANNO-NUMERO e sono descritte da una scheda esplicativa contenente (descrizione + link + data creazione).

Il CVE enumera le vulnerabilità, ma non stabilisce quale vulnerabilità debba essere gestita urgentemente e come possano impattare su sistemi diversi. Non è solo importante archiviare le vulnerabilità all'interno di un catalogo, ma anche assegnare una sorta di punteggio che indichi il livello di gravità di questi oggetti all'interno dell'archivio. Così viene introdotto il ***CVSS*** (***Common Vulnerability Scoring System***), ovvero un sistema di punteggio associato alle vulnerabilità dell'archivio, stima la gravità di ogni vulnerabilità e assegna ad ogni CVE id un punteggio da 0 (impatto nullo) a 10 (impatto critico). Ci sono due versioni del CVSS (v2 e v3), entrambe assegnano il punteggio in base a tre gruppi di metriche. Ad ogni metrica è associata una domanda a risposta multipla, ciascuna risposta fornisce un peso numerico e i singoli pesi sono poi aggregati in un risultato finale tramite una serie di formule.

BASE (BASE METRIC):
Stimano la gravità delle vulnerabilità, a prescindere da fattori temporali e ambientali.



La metrica ***Access Vector*** risponde alla domanda:

- *Tramite quale vettore di accesso può essere sfruttata la vulnerabilità?*

Valore	Descrizione	Punt.
Local (L)	L'attaccante deve avere accesso fisico/un account sul sistema.	0.395
Adjacent Network (A)	L'attaccante deve avere accesso al dominio di broadcast o di collisione del sistema.	0.646
Network (N)	L'interfaccia vulnerabile è al livello 3 o superiore della pila ISO/OSI.	1.0

La metrica ***Access Complexity*** risponde alla domanda:

- *Quanto è difficile sfruttare la vulnerabilità?*

Valore	Descrizione	Punt.
High (H)	Lo sfruttamento richiede condizioni particolari (corsa critica, tecniche di social engineering).	0.35
Medium (M)	Lo sfruttamento richiede alcune condizioni (ad es., configurazione non di default).	0.646
Low (L)	Lo sfruttamento non richiede nulla di particolare (funziona su sistemi standard).	1.0

La metrica ***Authentication*** risponde alla domanda:

- *Quante volte un attaccante si deve autenticare per sfruttare la vulnerabilità?*

Valore	Descrizione	Punt.
Multiple (M)	Lo sfruttamento richiede due o più autenticazioni (anche con le stesse credenziali).	0.45
Single (S)	Lo sfruttamento richiede una sola autenticazione.	0.56
None (N)	Lo sfruttamento non richiede alcuna forma di autenticazione.	0.704

La metrica ***Confidentiality Impact*** risponde alla domanda:

- *Qual è l'impatto della vulnerabilità sulla confidenzialità del sistema?*

Valore	Descrizione	Punt.
None (N)	Non vi è impatto alcuno.	0.0
Partial (P)	É possibile divulgare solo un sotto-insieme dei dati offerti dal sistema.	0.275
Complete (C)	É possibile divulgare l'intero insieme dei dati offerti dal sistema.	0.660

La metrica ***Integrity Impact*** risponde alla domanda:

- *Qual è l'impatto della vulnerabilità sull'integrità del sistema?*

Valore	Descrizione	Punt.
None (N)	Non vi è impatto alcuno.	0.0
Partial (P)	É possibile modificare solo un sotto-insieme dei dati offerti dal sistema.	0.275
Complete (C)	É possibile modificare l'intero insieme dei dati offerti dal sistema.	0.660

La metrica ***Availability Impact*** risponde alla domanda:

- *Qual è l'impatto della vulnerabilità sulla disponibilità del sistema?*

Valore	Descrizione	Punt.
None (N)	Non vi è impatto alcuno.	0.0
Partial (P)	É possibile ridurre parzialmente le prestazioni e/o le funzioni offerte dal sistema.	0.275
Complete (C)	É possibile ridurre completamente le prestazioni e/o le funzioni offerte dal sistema.	0.660

Le risposte relative alle metriche base sono presentate sotto forma di stringa di testo, che viene chiamata ***vector string***, è formata da coppie di abbreviazioni ***mestrica:risposta*** separate dal carattere ***/***. Esempio

AV:N/AC:L/Au:N/C:P/I:P/A:C

Viene dato un punteggio a questa stringa in base alle metriche delle tabelle, che verrà restituito il ***Punteggio Base***, ovvero la stima della gravità della vulnerabilità senza considerare fattori temporali ed ambientali.

$$Exploitability = 20 * AccessVector * AccessComplexity * Authentication$$

$$Impact = 10.41 * (1 - (1 - ConfImpact) * (1 - IntegImpact) * (1 - AvailImpact))$$

$$f(Impact) = \begin{cases} 0 & if\ Impact = 0 \\ 1.176 & otherwise \end{cases}$$

$$BaseScore = roundTo1Decimal(((0.6 * Impact) + (0.4 * Exploitability) - 1.5) * f(Impact))$$

I punteggi associati alle altre due metriche sono opzionali e si calcolano nello stesso modo (rispondendo a questionari). Esistono ***relazioni tra i punteggi***, infatti il punteggio Temporale ingloba il Base e quello Ambientale ingloba il Temporale. Pertanto, il punteggio Ambientale è quello più generale possibile che ingloba i due precedenti.

TEMPORALI (TEMPORAL METRIC):

Stimano la gravità della vulnerabilità dal punto di vista temporale.

La metrica ***Exploitability*** risponde alla domanda:

- *Qual è lo stato attuale delle tecniche di sfruttamento della vulnerabilità?*

Valore	Descrizione	Punt.
Unproven (U)	L'exploit non è pubblico, oppure esiste in linea solo teorica.	0.85
Proof of Concept (P)	È disponibile una bozza dimostrativa (**Proof of Concept**, **PoC**). Richiede adattamenti non banali per funzionare.	0.9
Functional (F)	É disponibile un exploit funzionante nella maggioranza dei casi in cui la vulnerabilità è presente.	0.95
High (H)	La vulnerabilità può essere sfruttata in modo automatico (anche da worm e virus).	1.0
Not Defined (ND)	Si ignori tale punteggio.	1.0

La metrica ***Remediation Level*** risponde alla domanda:

- È presente un rimedio per mitigare la vulnerabilità?

Valore	Descrizione	Punt.
Official fix (O)	Il vendor mette a disposizione un rimedio ufficiale (patch, aggiornamento software).	0.87
Temporary fix (T)	Il vendor mette a disposizione un rimedio ufficiale, ma temporaneo.	0.90
Workaround (W)	Una terza parte (NON il vendor) mette a disposizione un rimedio non ufficiale.	0.95
Unavailable (U)	Non è disponibile un rimedio, o è impossibile applicare una soluzione suggerita.	1.0
Not Defined (ND)	Si ignori tale punteggio.	1.0

La metrica ***Report Confidence*** risponde alla domanda:

- *La vulnerabilità esiste veramente? È descritta in maniera credibile?*

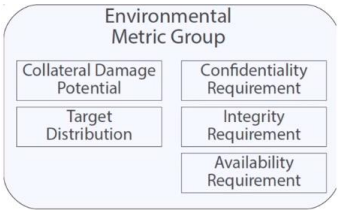
Valore	Descrizione	Punt.
Unconfirmed (UC)	La vulnerabilità è divulgata da una singola fonte non confermata, o da più fonti in mutuo conflitto.	0.9
Uncorroborated (UR)	La vulnerabilità è divulgata da più fonti concordi. Può esistere un livello residuo di incertezza.	0.95
Confirmed (C)	La vulnerabilità è confermata dal vendor.	1.0
Not Defined (ND)	Si ignori tale punteggio.	1.0

Il ***Punteggio Temporale*** stima la gravità della vulnerabilità includendo il fattore temporale

$$TemporalScore = roundTo1Decimal(BaseScore * Exploitab * RemedLvl * ReportConf)$$

AMBIENTALI (ENVIRONEMENT METRIC):

Stimano la gravità della vulnerabilità dal punto di vista ambientale.



La metrica ***Collateral Damage Potential*** risponde alla domanda:

- *Qual è l'impatto della vulnerabilità sui sistemi fisici, sulle persone e sulle risorse finanziarie?*

Valore	Descrizione	Punt.
None (N)	Nessun impatto.	0
Low (L)	Danno fisico basso, perdita marginale di guadagno.	0.1
Low-Medium (LM)	Danno fisico ed economico moderato.	0.3
Medium-High (MH)	Danno fisico ed economico significativo.	0.4
High (H)	Danno fisico ed economico catastrofico.	0.5
Not Defined (ND)	Si ignori tale punteggio.	1.0

La metrica ***Target Distribution*** risponde alla domanda:

- *Quale percentuale di asset è soggetta alla vulnerabilità?*

Valore	Descrizione	Punt.
None (N)	Percentuale nulla.	0
Low (L)	1%-25% degli asset.	0.25
Medium (M)	26%-75% degli asset.	0.75
High (H)	76%-100% degli asset.	1.0
Not Defined (ND)	Si ignori tale punteggio.	1.0

La metrica ***Confidentiality Requirement*** risponde alla domanda:

- *Qual è l'impatto di una perdita di confidenzialità?*

Valore	Descrizione	Punt.
Low (L)	L'impatto è lieve.	0.5
Medium (M)	L'impatto è serio.	1.0
High (H)	L'impatto è catastrofico.	1.51
Not Defined (ND)	Si ignori tale punteggio.	1.0

La metrica ***Integrity Requirement*** risponde alla domanda:

- *Qual è l'impatto di una perdita di integrità?*

Valore	Descrizione	Punt.
Low (L)	L'impatto è lieve.	0.5
Medium (M)	L'impatto è serio.	1.0
High (H)	L'impatto è catastrofico.	1.51
Not Defined (ND)	Si ignori tale punteggio.	1.0

La metrica ***Availability Requirement*** risponde alla domanda:

- *Qual è l'impatto di una perdita di disponibilità?*

Valore	Descrizione	Punt.
Low (L)	L'impatto è lieve.	0.5
Medium (M)	L'impatto è serio.	1.0
High (H)	L'impatto è catastrofico.	1.51
Not Defined (ND)	Si ignori tale punteggio.	1.0

Il ***Punteggio Ambientale*** stima la gravità della vulnerabilità includendo il fattore ambientale

$$AdjImp = min(10, 10.41 * (1 - (1 - ConfImp * ConfReq) * (1 - IntImp * IntReq) * (1 - AvImp * AvReq)))$$

$$AdjTemp = punteggio\ Temporal\ ricalcolato\ con\ AdjImp\ al\ posto\ di\ Impact$$

$$EnvironmentalScore = roundTo1\ Decimal((AdjTemp + (10 - AdjTemp) * CollatDamPot) * TargetDist)$$

PUNTEGGI CVSS:

I punteggi CVSS ***Base*** e ***Temporali*** sono calcolati dai venditori di software, mentre quello ***Ambientale*** è calcolato dagli amministratori delle infrastrutture. Questi punteggi vengono utilizzati da chiunque abbia a che fare con il processo di gestione della sicurezza. Esiste un foglio di calcolo che, rispondendo alle domande, è possibile ottenere il punteggio.

3. CATALOGAZIONE DELLE DEBOLEZZE

Oltre al *catalogo delle vulnerabilità* col sistema CVE e come attribuire loro dei punteggi basati su metriche col sistema CVSS, esiste un *catalogo delle debolezze* col sistema CWE e un sistema CWSS per l'attribuzione di punteggi agli oggetti del catalogo.

Il ***Common Weaknesses Enumeration*** (***CWE***) è un sistema per catalogare in modo uniforme le debolezze software. Un esempio è:

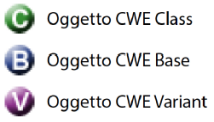
CWE-276: Incorrect Default Permissions, CWE-272: Least Privilege Violation, CWE-426: Untrusted Search Path

Oltre a essere menzionato CWE, per capire a quale archivio appartiene, e un numero (diverso all'archivio precedente in cui compariva anche l'anno), è presente anche una breve descrizione della debolezza.

Il catalogo CWE è un insieme di oggetti, ciascuno dotato di un ***identificatore*** e di un numero di ***attributi***, che possono essere *Abstraction*, *Relationships*, ecc… Un oggetto del catalogo, a differenza del catalogo delle vulnerabilità, può essere la descrizione di una singola debolezza o un elenco di identificatori a singole debolezze in relazione tra loro.

L'attributo ***Abstraction*** specifica il tipo di debolezza, e può essere di tre tipi diversi:

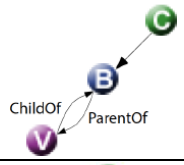
- **C - *Class***: debolezza descritta in termini generali, senza riferimenti a linguaggi o tecnologie specifiche;
- **B - *Base***: debolezza descritta in modo più dettagliato, in modo da poter intuire tecniche di rilevazione e prevenzione;
- **V - *Variant***: debolezza descritta nei minimi dettagli, nell'ambito di uno specifico linguaggio e tecnologia.



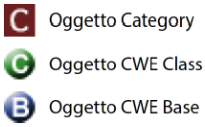
L'attributo ***Relationships*** specifica il tipo di relazioni che l'oggetto ha con altri oggetti del catalogo.

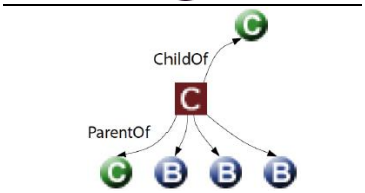
Esempio di relazioni:

- ***ChildOf***: l'oggetto è figlio di un altro oggetto;
- ***ParentOf***: l'oggetto è padre di un altro oggetto.



Un oggetto ***Category*** punta ad un insieme di oggetti che condividono uno specifico attributo. Essendo un oggetto raggruppatore, di solito ha più relazioni *ParentOf* che *ChildOf*.

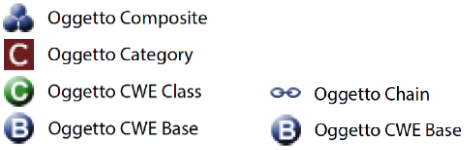


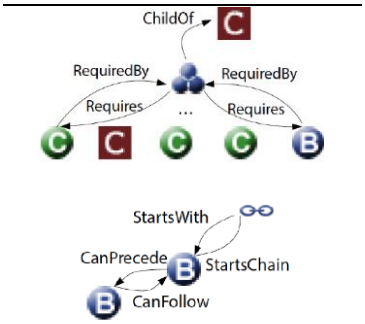


Un oggetto ***Compound*** mette in relazione tra loro diverse debolezze implicate in una vulnerabilità.

Due tipologie:

- ***Composite***: aggrega tutte le debolezze che sfruttate insieme, provocano una vulnerabilità;
- ***Chain***: aggrega tutte le debolezze che, sfruttate in cascata, provocano una vulnerabilità.





Così come per il catalogo CVE esiste il sistema di punteggi CVSS, anche per il catalogo CWE esiste un sistema analogo, ovvero il ***Common Weaknesses Scoring System*** (***CWSS***), anche qui ad ogni CWE id è assegnato un punteggio ma va da 0 (nullo) a 100 (catastrofico).

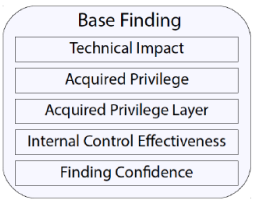
Il punteggio CWSS è dato dal prodotto di tre sottopunteggi: BaseFinding (tra 0 e 100), Attack Surface (tra 0 e 1) ed Environmental (tra 0 e 1). Ad ogni metrica è associata una domanda a risposta multipla, ciascuna risposta fornisce un peso numerico e i singoli pesi sono poi aggregati in un risultato finale tramite una serie di formule.

Le risposte relative alle domande del questionario sono presentate sotto forma di stringa di testo. Tale stringa, detta ***vector string***, è formata da terne di abbreviazioni ***domanda:risposta,peso*** separate dal carattere ***/***. Esempio:

TI:H,0.9/AP:A,1.0/AL:A,1.0/IC:N,1.0/FC:T,1.0

BASE FINDING:

Stimano il rischio della debolezza in sé, l'accuratezza della scoperta, la robustezza dei meccanismi di protezione.



La metrica ***Technical Impact*** risponde alla domanda:

- *Nell'ipotesi che la debolezza possa essere sfruttata con successo, qual è la principale conseguenza tecnica?*

Valore	Descrizione	Punt.
Critical (C)	Controllo completo; interruzione delle operazioni.	1.0
High (H)	Controllo di molte operazioni; accesso ad informazioni critiche.	0.9
Medium (M)	Controllo di alcune operazioni; accesso ad informazioni importanti.	0.6
Low (L)	Controllo minimo; accesso ad informazioni irrilevanti.	0.3
None (N)	La debolezza non porta ad una vulnerabilità.	0.0

La metrica ***Acquired Privilege*** risponde alla domanda:

- *Nell'ipotesi che la debolezza possa essere sfruttata con successo, che tipi di privilegi si ottengono?*

Valore	Descrizione	Punt.
Administrator (A)	L'attaccante diventa amministratore (`root` in UNIX, `SYSTEM` in Windows, `admin` su un router).	1.0
Partially Privileged User (P)	L'attaccante diventa un utente con alcuni privilegi, ma non tutti quelli di un amministratore.	0.9
Regular User (RU)	L'attaccante divente un utente normale, senza privilegi particolari.	0.7
Limited or Guest (L)	L'attaccante diventa un utente con privilegi ristretti (ad esempio, `nobody` su UNIX).	0.6
None (N)	L'attaccante non riesce a diventare un utente.	0.1

La metrica ***Acquired Privilege Layer*** risponde alla domanda:

- *Nell'ipotesi che la debolezza possa essere sfruttata con successo, a che livello operazionale si ottengono i privilegi?*

Valore	Descrizione	Punt.
Application (A)	L'attaccante acquisisce privilegi a livello di utente di una applicazione software.	1.0
System (S)	L'attaccante acquisisce privilegi a livello di utente di un sistema operativo.	0.9
Network (N)	L'attaccante acquisisce il privilegio di accesso alla rete.	0.7
Enterprise Infrastructure (E)	L'attaccante acquisisce l'accesso ad una porzione dell'infrastruttura (router, switch, DNS, controller di dominio, firewall, …).	1.0

La metrica ***Internal Control Effectiveness*** risponde alla domanda:

- *Qual è l'efficacia delle contromisure, a livello di codice?*

Valore	Descrizione	Punt.
None (N)	Non esistono contromisure.	1.0
Limited (L)	Esiste un meccanismo semplice o fortuito, in grado di rintuzzare un attaccante occasionale.	0.9
Moderate (M)	Esiste un meccanismo standard con dei limiti, aggirabile con un po' di impegno da un esperto.	0.7
Indirect (I)	Un meccanismo non specifico per la debolezza ne riduce l'impatto in maniera indiretta.	0.5
Best-Available (B)	È implementato il meccanismo migliore noto. Un attaccante esperto e determinato potrebbe aggirarlo con l'aiuto di altre debolezze.	0.3
Complete (C)	Il meccanismo impedisce lo sfruttamento.	0.0

La metrica ***Finding Confidence*** risponde alla domanda:

- *Quanto si è sicuri che il difetto individuato sia una debolezza e possa essere usato da un attaccante?*

Valore	Descrizione	Punt.
Proven True (T)	La debolezza esiste ed è raggiungibile da un attaccante.	1.0
Proven Locally True (LT)	La debolezza esiste, ma non è chiaro se sia o meno sfruttabile da un attaccante.	0.8
Proven False (F)	Il difetto/bug non costituisce una debolezza e/o non è sfruttabile da un attaccante.	0.0

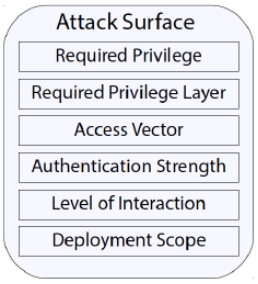
Il Punteggio ***Base Findings*** è un valore tra 0 e 100, calcolato nel modo seguente:

$$f(TI)=\begin{cases}0 & if\ TI=0\\ 1 & otherwise\end{cases}$$

$$BaseFindingScore=[(10*TI+5*(AP+AL)+5*FC)*f(TI)*IC]*4.0$$

ATTACK SURFACE:

Stimano le barriere che un attaccante deve superare per sfruttare la debolezza.



La metrica ***Required Privilege*** risponde alla domanda:

- *Quali privilegi deve già possedere l'utente per sfruttare la debolezza?*

Valore	Descrizione	Punt.
None (N)	Non sono richiesti privilegi particolari.	1.0
Limited / Guest (L)	L'attaccante deve già avere i privilegi di un utente ristretto.	0.9
Regular User (RU)	L'attaccante deve già avere i privilegi di un utente normale.	0.7
Partially Privileged User (P)	L'attaccante deve già avere i privilegi di un utente speciale (con alcuni privilegi in più rispetto ad uno normale, ma non tutti quelli di un amministratore).	0.6
Administrator (A)	L'attaccante deve già avere i privilegi di un utente amministratore.	0.1

La metrica ***Required Privilege Layer*** risponde alla domanda:

- *A quale livello operazionale l'attaccante deve avere privilegi per poter sfruttare la debolezza?*

Valore	Descrizione	Punt.
Application (A)	L'attaccante deve già avere privilegi applicativi.	1.0
System (S)	L'attaccante deve già avere privilegi a livello di sistema operativo.	0.9
Network (N)	L'attaccante deve già avere i privilegi di accesso alla rete.	0.7
Enterprise Infrastructure (E)	L'attaccante deve già avere i privilegi a livello di infrastruttura (router, switch, DNS, controller di dominio, firewall, …).	1.0

La metrica ***Access Vector*** risponde alla domanda:

- *Attraverso quale canale deve comunicare l'attaccante per poter sfruttare la debolezza?*

Valore	Descrizione	Punt.
Internet (I)	L'attaccante deve avere accesso ad Internet.	1.0
Intranet (R)	L'attaccante deve accesso ad una Intranet schermata da un proxy Web.	0.8
Private Network (V)	L'attaccante deve avere accesso ad una rete privata disponibile solo ad alcuni utenti fidati.	0.8
Adjacent Network (A)	L'attaccante deve avere accesso fisico al dominio di broadcast o di collisione della rete.	0.7
Local (L)	L'attaccante deve avere accesso locale ad una shell.	0.5
Physical (P)	L'attaccante deve avere accesso fisico all'asset.	0.2

La metrica ***Authentication Strength*** risponde alla domanda:

- *Quanto la procedura di autenticazione protegge la debolezza?*

Valore	Descrizione	Punt.
None (N)	Non è prevista alcuna forma di autenticazione.	1.0
Weak (W)	È prevista una autenticazione debole (username e password).	0.9
Moderate (M)	È prevista una autenticazione moderatamente forte (uso di certificati, autenticazione basata su conoscenza, one-time password).	0.8
Strong (S)	È prevista una autenticazione forte (token hardware, multi-fattore).	0.7

La metrica ***Level of Interaction*** risponde alla domanda:

- *Quali azioni deve compiere la vittima per consentire all'attaccante di svolgere l'attacco con successo?*

Valore	Descrizione	Punt.
Automated (A)	Non è richiesta interazione umana.	1.0
Typical / Limited (T)	L'attaccante deve convincere l'utente a svolgere una azione normale nel contesto del software.	0.9
Moderate (M)	L'attaccante deve convincere l'utente a svolgere una azione sospetta per un conoscente della sicurezza.	0.8
Opportunistic (N)	L'attaccante non può controllare direttamente la vittima; può solo capitalizzare errori altrui.	0.3
High (H)	L'attaccante deve usare il social engineering.	0.1
No (NI) Interaction	Non è possibile alcuna interazione.	0.0

La metrica ***Deployment Scope*** risponde alla domanda:

- *In quali piattaforma e/o configurazioni si presenta la debolezza?*

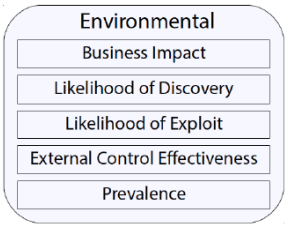
Valore	Descrizione	Punt.
All (A)	La debolezza si manifesta in tutte le piattaforme ed in tutte le configurazioni.	1.0
Moderate (M)	La debolezza si manifesta nelle piattaforme e/o nelle configurazioni più comuni.	0.9
Rare (R)	La debolezza si manifesta solo in piattaforme e/o nelle configurazioni più rare.	0.5
Potentially Reachable (P)	La debolezza è potenzialmente sfruttabile. In questo specifico istante tutti i percorsi di codice sembrano sicuri e/o la debolezza è codice "morto" (non raggiungibile in pratica).	0.1

Il Punteggio ***Attack Surface*** è un valore tra 0 e 1, calcolato nel modo seguente:

$$AttackSurfaceScore = [20*(RP+RL+AV)+20*SC+15*IN*5*AS]/100.0$$

ENVIRONMENTAL:

Stimano le specificità legate ad uno specifico contesto operativo.



La metrica ***Business Impact*** risponde alla domanda:

- *Qual è l'impatto ambientale di uno sfruttamento della sicurezza?*

Valore	Descrizione	Punt.
Critical (C)	L'azienda può fallire.	1.0
High (H)	Le operazioni aziendali sono colpite gravemente.	0.9
Medium (M)	Alcune operazioni aziendali sono colpite, ma non quelle più comuni.	0.6
Low (L)	L'impatto aziendale è minimo.	0.3
None (N)	Non vi è impatto aziendale alcuno.	0.0

La metrica ***Likelihood of Discovery*** risponde alla domanda:

- *Qual è la probabilità che un attaccante scopra la debolezza?*

Valore	Descrizione	Punt.
High (H)	È molto probabile che un attaccante riesca a scoprire la debolezza usando tecniche semplici e senza accesso al codice sorgente del software.	1.0
Medium (M)	Un attaccante potrebbe riuscire a scoprire la debolezza, ma solo con accesso al codice sorgente del software e tanto tempo a disposizione.	0.6
Low (L)	È improbabile che un attaccante riesca a scoprire la debolezza senza avere capacità particolari, accesso al codice sorgente e tanto tempo a disposizione.	0.2

La metrica ***Likelyhood of Exploit*** risponde alla domanda:

- *Qual è la probabilità che, una volta scoperta la debolezza, un attaccante con il giusto privilegio sia in grado di sfruttarla?*

Valore	Descrizione	Punt.
High (H)	È molto probabile che un attaccante riesca a sfruttare la debolezza tramite un exploit di facile implementazione.	1.0
Medium (M)	Un attaccante potrebbe riuscire a sfruttare la debolezza. Le probabilità di successo variano; potrebbero essere necessari più tentativi.	0.6
Low (L)	È improbabile che un attaccante riesca a sfruttare la debolezza.	0.2
None (N)	L'attaccante non ha alcuna chance di successo.	0.0

La metrica ***External Control Effectiveness*** risponde a:

- *Qual è l'efficacia delle contromisure esterne (NON a livello di codice)?*

Valore	Descrizione	Punt.
None (N)	Non esistono contromisure.	1.0
Limited (L)	Esiste un meccanismo semplice o fortuito, in grado di rintuzzare un attaccante occasionale.	0.9
Moderate (M)	Esiste un meccanismo standard con dei limiti, aggirabile con un po' di impegno da un esperto.	0.7
Indirect (I)	Un meccanismo non specifico per la debolezza ne riduce l'impatto in maniera indiretta.	0.5
Best-Available (B)	È implementato il meccanismo migliore noto. Un attaccante esperto e determinato potrebbe aggirarlo con l'aiuto di altre debolezze.	0.3
Complete (C)	Il meccanismo impedisce lo sfruttamento.	0.1

La metrica ***Prevalence*** risponde alla domanda:

- *Qual è la frequenza di occorrenza della debolezza nel software in generale?*

Valore	Descrizione	Punt.
Widespread (W)	La debolezza è presente nella maggioranza (se non la totalità) dei software in esecuzione nella infrastruttura considerata.	1.0
High (H)	La debolezza si incontra spesso, ma non è diffusa su ampio spettro.	0.9
Common (C)	La debolezza si incontra di tanto in tanto.	0.8
Limited (L)	La debolezza si incontra raramente (oppure, mai).	0.7

Il Punteggio ***Environmental*** è un valore tra 0 e 1, calcolato nel modo seguente:

$$f(BI) = \begin{cases} 0 & if\ BI = 0 \\ 1 & otherwise \end{cases}$$

$$EnvironmentalScore = [(10*BI+3*DI+4*EX+3*P)*f(BI)*EC]/20.0$$

CVSS e CWSS sono molto simili, ma ci sono alcune differenze:

- CVSS assume che una vulnerabilità sia già stata scoperta e verificata, mentre CWSS può essere utilizzata prima che ciò accada. CVSS cataloga gli errori fatti, mente CWSS cataloga gli errori fattibili;
- In CVSS alcuni aspetti combinano caratteristiche multiple, che sono invece separate in CWSS. Ad esempio, Access Complexity (AC) si suddivide in Required Privilege Level e Level of Interaction.

4. ESECUZIONE CON PRIVILEGI ELEVATI

Nei sistemi UNIX può capitare che un processo necessiti di ***privilegi elevati*** (*privilegi di root*). Comandi che richiedono permessi di *root*:

- ***passwd***, per modificare il file */etc/passwd*;
- *ping*, per aprire *socket* e porte;
- *mount*, per modificare *file system* residenti su memorie di massa removibili.

L'elevazione dei privilegi può essere effettuata in due modi:

- ***Manuale***:
  1. L'utente digita i comandi opportuni per diventare un amministratore (*deve conoscere la password di admin*);
  2. L'utente esegue il comando che gli interessa.

  Il comando *su* consente a un utente di eseguire comandi con i privilegi di un altro utente, in questo caso l'utente root.
  La procedura richiede la conoscenza della password di root.
  Una volta acquisiti i privilegi di root, l'utente può modificare la sua password (che sarà scritta nel file */etc/passwd*).

```
$ su –
Password:
$ passwd masucci
Changing password for user masucci.
New password:
Retype new password:
passwd: all authentication tokens updated succesfully
```

  Con questo sistema è *necessario conoscere la password di root*. Dopo aver acquisito i privilegi, l'utente può ispezionare il sistema, modificare file e compromettere servizi. Inoltre, è necessario immettere la password di root ogni volta che serve acquisire privilegi elevati. C'è il rischio che per agevolare l'operazione, si decida di *non impostare alcuna password per la root*, in tal caso, chiunque accede al terminale può diventare amministratore.

  La modalità manuale ha il vantaggio che bisogna conoscere la password per consentire l'accesso e svolgere azioni.

- ***Automatica*** (*tipicamente usata in UNIX*):
  1. L'utente esegue il comando che gli interessa (*direttamente senza richiedere esplicitamente l'elevazione dei privilegi*);
  2. Il SO esegue l'elevazione dei privilegi (anche a nostra insaputa).

  In questo caso, può portare dei problemi se associata a delle debolezze.

  L'utente richiede di modificare la password mediante il comando *passwd*. Il SO effettua l'elevazione automatica ai privilegi di *root*, consentendo all'utente di modificare la password (che sarà scritta nel file */etc/passwd*).

```
$ passwd masucci
Changing password for user masucci.
Current password:
New password:
Retype new password:
passwd: all authentication tokens updated succesfully
```

ORGANIZZAZIONE DEL FILE SYSTEM:

Per capire perché il comando *passwd* faccia ottenere i privilegi di root all'utente, bisogna rivedere come è ***organizzato il file system*** nei sistemi UNIX.

In UNIX, un ***utente*** è caratterizzato da:

- *Username* (stringa);
- *UID*, User IDentification number (intero), la *root* è l'utente con *UID=0*.

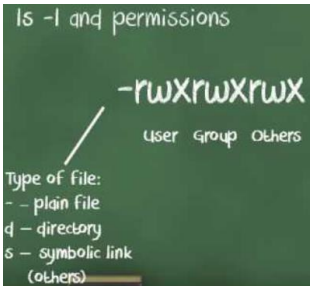
Un ***gruppo*** è caratterizzato da:

- *Groupname* (stringa)
- *GID*, Group IDentification number (intero)

Queste informazioni possono essere visualizzate con il ***comando id***.

L'***accesso ai file*** è regolato da permessi, definiti come tre terne di azioni (una terna per ciascuna tipologia di utenti):

- *Azioni*: ***R***ead, ***W***rite, e***X***ecute;
- *Tipologie di utenti*: ***U***ser (proprietario), ***G***roup (gruppo di lavoro), ***O***ther (altri utenti).



NOTA: eXecute su un file indica che il file può essere eseguito, mentre eXecute su una directory indica che si può entrare in essa.

I permessi vengono ***rappresentati*** in:

- ***Forma ottale***:
  Read -> 4, Write -> 2, eXecute -> 1.
  La terna di azioni viene rappresentata con la somma di permessi.
  Esempio: *rwxrwxr-x -> 4+2+1 4+2+1 4+1 -> 775*.
- ***Forma simbolica***:
  Read -> r, Write -> w, eXecute -> x, Creatore (del file) -> u, Gruppo -> g, Altri -> o.
  Per una modifica su un file, bisogna specificare la tipologia di utente ± le azioni -> u+rw.

Permesso finale le modifiche sono separate da , (virgole). Esempio: *rwxrwxr-x -> ug+rwx, o+rx*.
I permessi possono essere impostati con il comando:

chmod nuovi-permessi nome-file

Esempio, se si esegue *chmod 777 nomefile*, vengono dati tutti i permessi possibili su quel file.

#	Permission	rwx
7	read, write and execute	rwx
6	read and write	rw-
5	read and execute	r-x
4	read only	r--
3	write and execute	-wx
2	write only	-w-
1	execute only	--x
0	none	---

Si evince quindi, che a ciascun file sia associata una ***stringa di permessi*** costituita da 9 bit. In realtà ci sono altri tre bit aggiuntivi associati ai permessi: *SETUID*, *SETGID*, *STICKY*, che portano da 9 a 12 bit. Quando sono posti uguali a 1, consentono una elevazione dei privilegi. La rappresentazioni ottale / simbolica è: per ***SETUID: 4 / s***, mentre per ***SETGID: 2 / s***.

- ***SETUID***:
  Per impostare a 1 il bit SETUID di un file, si utilizza il comando: *chmod u+s nomefile*.
  Per verificare che questo bit è "acceso" per un determinato file, invochiamo *ls –l nomefile*, se è presente il permesso "s" al posto della "x" e se il file è evidenziato in rosso, è possibile a chi lancia il comando diventare root, anche se il file non è di sua proprietà.

```
barbara@barbara-VirtualBox:/usr/bin$ ls -la passwd
-rwsr-xr-x 1 root root 54256 mar 26  2019 passwd
```

Per individuare ***tutti*** i file con il bit SETUID acceso, possiamo usare il comando *find* con l'opzione *–perm* (filtro in base ai permessi).
Esempio: *find / -perm /u+s*.
Per evitare di visualizzare i messaggi di errore (permission denied) si usa *find / -perm /u+s 2>/dev/null*.

```
barbara@barbara-VirtualBox:~$ find /usr/bin -perm /u+s
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/sudo
/usr/bin/newgrp
/usr/bin/pkexec
```

- ***SETGID***:
  Per impostare a 1 il bit SETGID di un file, si utilizza il comando: *chmod g+s file*. Per verificare che il bit è "acceso" per un determinato file, invochiamo *ls –l nomefile*, se è presente il permesso "s" e se il file è evidenziato in giallo.

```
barbara@barbara-VirtualBox:/usr/bin$ ls -la wall
-rwxr-sr-x 1 root tty 27368 gen 27  2020 wall
```

  Per individuare ***tutti*** i file con il bit SETGID acceso possiamo usare utilizzare il comando *find* con l'opzione *–perm* (filtro in base ai permessi). Esempio: *find / -perm /g+s*. Per evitare i messaggi di errore (permission denied) si usa *find / -perm /g+s 2>/dev/null*.

UTENTI E GRUPPI REALI:

Nei SO UNIX, il descrittore di un processo memorizza ***una prima coppia di credenziali***:

- *Real User ID* (***RUID***) è l'UID di chi ha lanciato il comando;
- *Real Group ID* (***RGID***) è il GID di chi ha lanciato il comando.

E ***una seconda coppia di credenziali***:

- *Effective User ID* (***EUID***) è l'UID di chi possiede il file;
- *Effective Group ID* (***EGID***) è il GID di chi possiede il file.

In condizioni normali, quando ***SETUID*** e ***SETGID*** *disattivati*, queste coppie, real User/Group ID e effective User/Group ID, coincidono:

- EUID è l'UID *di chi ha lanciato il comando*;
- EGID è il GID *di chi ha lanciato il comando*.

Se invece i bit ***SETUID*** e ***SETGID*** sono *attivati* (nella stringa dei metadati troveremo la *s*) ha importanza l'effettività:

- EUID è l'UID *di chi possiede il file*;
- EGID è il GID *di chi possiede il file*.

Quando i bit sono attivi, chi lancia il comando può assumere i privilegi di chi possiede il file e *ottiene i pieni poteri dell'utente root*, in tal caso, il processo può fare di tutto. Inoltre, il privilegio ottenuto *è mantenuto per l'intera esecuzione*.

PROBLEMA DEI PRIVILEGI ELEVATI:

L'elevazione automatica dei privilegi è una funzionalità interessante offerta da UNIX. Tuttavia, il conferimento dei pieni poteri di root, quando non sono strettamente necessari, è una ***debolezza***. Anche il mantenimento dei privilegi per tutta la durata dell'esecuzione lo è. Non si tratta di ***vulnerabilità***, perché le debolezza non sono sfruttabili da sole. Se un programma è scritto in modo corretto, la sola elevazione automatica non può dare alcun vantaggio a un attaccante. Tuttavia, se unita ad altre debolezze, l'elevazione automatica dei privilegi può avere conseguenze devastanti per la sicurezza di un programma.
Per risolvere l'abbassamento e ripristino dei privilegi, se l'applicazione non svolge operazioni critiche, può decidere di abbassare i propri privilegi a quelli dell'utente che ha eseguito il comando (***privilege drop***). Quando l'applicazione svolge operazioni critiche, ripristina nuovamente i privilegi ottenuti con l'elevazione automatica (***privilege restore***).
Per implementare l'abbassamento ed il ripristino dei privilegi di un programma, vengono usate delle chiamate di sistema che assolvono al compito, relativamente a diversi sistemi come: i primissimi sistemi UNIX, UNIX System V, UNIX BSD, POSIX e UNIX moderni.
Nei ***primissimi sistemi UNIX*** sono previste solo due tipologie di user (e group) ID: RUID (RGID) e EUID (EGID).
I valori RUID ed EUID possono essere determinati mediante le chiamate di sistema:

- *getuid()*, che restituisce il ***R***UID del processo invocante;
- *geteuid()*, che restituisce l'***E***UID del processo invocante.

Si compili *getuid_unix.c*:

gcc –o getuid_unix getuid_unix.c

Si esegua una prima volta *getuid_unix* come *utente normale*:

./getuid_unix

Viene stampato:

RUID del processo = 1000

Si esegua una seconda volta *getuid_unix* come *root*:

sudo ./getuid_unix

Viene stampato:

RUID del processo = 0

getuid_unix.c

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {

      int uid = getuid();

      printf("RUID del processo = %d\n", uid);
      exit(0);
}
```

NOTA: È 0 perché è stato messo *sudo* prima di eseguire il programma, e quindi come se fossimo root, mentre era 1000 perché era stato mandato come utente normale.

Si compili *geteuid_unix.c*:

gcc –o geteuid_unix geteuid_unix.c

Si esegua una prima volta *geteuid_unix* come *utente normale*:

./geteuid_unix

Viene stampato:

EUID del processo = 1000

Si esegua una seconda volta *geteuid_unix* come *root*:

sudo ./geteuid_unix

Viene stampato:

EUID del processo = 0

geteuid_unix.c

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {

      int euid = geteuid();

      printf("EUID del processo = %d\n", euid);
      exit(0);
}
```

NOTA: è ancora 1000 e 0 nel primo e secondo caso perché il bit ***SETUID*** è disattivato, pertanto coincidono UID ed EUID.

Se si imposta il proprietario di *geteuid_unix* a root, tramite:

sudo chown root geteuid_unix

Si imposti il bit SETUID per il file *geteuid_unix*:

sudo chmod u+s geteuid_unix

Si esegua *geteuid_unix*:

./geteuid_unix

Nonostante il processo venga lanciato dall'utente, viene stampato:

EUID del processo = 0

La presenza del bit SETUID=1 ha consentito l'esecuzione del programma con i privilegi del proprietario del file (root).

È anche possibile ***cambiare il valore dell'EUID di un processo***. La chiamata di sistema *setuid(uid)* imposta l'EUID del processo al valore uid in input. Ad esempio, *setuid(0)* imposta ***E***UID a 0 (root). In caso di errore, l'output della chiamata è -1.

Si compili setuid_unix.c:

gcc –o setuid_unix setuid_unix.c

Si esegua setuid_unix una prima volta da *utente normale*:

./setuid_unix

Viene stampato:

Prima di elevazione a root: RUID=1000 e EUID=1000

Viene stampato il messaggio di errore:

Non sono riuscito a impostare EUID=0

Si imposti il proprietario di *setuid_unix* a *root*:

sudo chown root setuid_unix

Si imposti il bit *SETUID* per *setuid_unix*:

sudo chmod u+s setuid_unix

Si esegua setuid_unix una seconda volta

./setuid_unix

Viene stampato:

Prima di elevazione a root: RUID=1000 e EUID=0

Dopo elevazione a root: RUID=0 e EIUD=0

setuid_unix.c

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {

        int uid = getuid();
        int euid = geteuid();

        printf("Prima di elevazione a root:
                RUID del processo = %d\n", uid);
        printf("Prima di elevazione a root:
                EUID del processo = %d\n", euid);
if (setuid(0) == -1) {
       printf("Non sono riuscito ad impostare
              EUID = 0 (root).\n");
       exit(1);
}

uid = getuid();
euid = geteuid();
printf("Dopo elevazione a root:
              RUID del processo = %d\n", uid);
printf("Dopo elevazione a root:
              EUID del processo = %d\n", euid);
exit(0);
}
```

Per effettuare un *abbassamento dei privilegi* basta eseguire il comando:

setuid(getuid());

In tal modo, infatti, l'EUID del processo viene posto uguale al RUID.

Si compili drop_priv_unix.c:

gcc –o drop_priv_unix drop_priv_unix.c

Si esegua drop_priv_unix una prima volta *utente normale*:

./drop_priv_unix

Viene stampato:

Prima dell'abbassamento dei privilegi: RUID=1000 e EUID=1000

Dopo l'abbassamento dei privilegi: RUID=1000 e EUID=1000

Si imposti il proprietario a root:

sudo chown root drop_priv_unix

Si imposti il bit SETUID:

sudo chmod u+s drop_priv_unix

Si esegua drop_priv_unix una seconda volta:

./drop_priv_unix

Viene stampato:

Prima dell'abbassamento dei privilegi: RUID=1000 e EUID=0

Dopo l'abbassamento dei privilegi: RUID=1000 e EUID=1000

drop_priv_unix.c

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {

 int uid, euid;

 uid = getuid();
 euid = geteuid();
 printf("Prima dell'abbassamento dei privilegi:
        RUID del processo = %d\n", uid);
 printf("Prima dell'abbassamento dei privilegi:
        EUID del processo = %d\n", euid);
        if (setuid(uid) == -1) {
              printf("Non sono riuscito ad abbassare i
                     privilegi.\n");
              exit(1);
        }

        uid = getuid();
        euid = geteuid();
        printf("Dopo l'abbassamento dei privilegi:
                     RUID del processo = %d\n", uid);
        printf("Dopo l'abbassamento dei privilegi:
                     EUID del processo = %d\n", euid);
        exit(0);
}
```

Non è possibile ripristinare i privilegi elevati dopo averli abbassati, perché nei primissimi sistemi UNIX, l'abbassamento dei privilegi tramite lo statement *setuid(getuid());* non permette più il ripristino del privilegio elevato che si aveva in precedenza. Si tratta quindi di un *abbassamento permanente dei privilegi*.

Supponiamo che un processo parta SETUID root, abbassi i suoi privilegi con *setuid(getuid())*, invochi *setuid(0)* per ripristinare i privilegi di root, dopodiché viene generato un errore.

Si compili *drop_rest_unix.c*:

gcc -o drop_rest_unix drop_rest_unix.c

Si esegua *drop_rest_unix* una prima volta:

./drop_rest_unix

Viene stampato:

Prima dell'abbassamento dei privilegi: RUID=1000 e EUID=0

Dopo l'abbassamento dei privilegi: RUID=1000 e EUID=1000

Dopo il ripristino dei privilegi: RUID=1000 e EUID=1000

drop_rest_unix.c

```
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {

        int uid, euid;
        int priv_uid;

        uid = getuid();
        euid = geteuid();
        priv_uid = euid; /* nell'ipotesi che il binario
               sia SETUID root */
        printf("Prima dell'abbassamento privilegi:
               RUID del processo = %d\n", uid);
        printf("Prima dell'abbassamento privilegi:
               EUID del processo = %d\n", euid);
```

Si imposti il creatore a root:

sudo chown root drop_rest_unix

Si imposti il bit SETUID:

sudo chmod u+s drop_rest_unix

Si esegua drop_rest_unix una seconda volta:

./drop_rest_unix

Viene stampato:

Prima dell'abbassamento dei privilegi: RUID=1000 e EUID=0
Dopo l'abbassamento dei privilegi: RUID=1000 e EUID=1000
Non sono riuscito a ripristinare i privilegi

```
        if (setuid(uid) == -1) {
                printf("Non sono riuscito ad abbassare i
                                privilegi.\n");
                exit(1);
         }

         uid = getuid();
         euid = geteuid();
         printf("Dopo l'abbassamento privilegi:
                        RUID del processo = %d\n", uid);
         printf("Dopo l'abbassamento privilegi:
                        EUID del processo = %d\n", euid);
        if (setuid(priv_uid) == -1) {
               printf("Non sono riuscito a ripristinare
                        i privilegi.\n");
               exit(1);
        }

        uid = getuid();
        euid = geteuid();
        printf("Dopo il ripristino privilegi:
                        RUID del processo = %d\n", uid);
        printf("Dopo il ripristino privilegi:
                        EUID del processo = %d\n", euid);
        exit(0);

}
```

Viene introdotta una ***nuova chiamata di sistema*** per cambiare il valore dell'EUID di un processo:

- *seteuid(uid)* imposta l'EUID del processo al valore uid in input (*abbassamento dei privilegi temporaneo*)

Per effettuare un abbassamento dei privilegi basta eseguire il comando *seteuid(getuid());*
Supponiamo che un processo parta SETUID root, abbassi i suoi privilegi con *seteuid(getuid()*, invochi *seteuid(0)*.
Allora *seteuid(0)* termina correttamente, perché root è l'utente effettivo, dopodiché imposta l'EUID al valore 0 (root).
Abbiamo quindi due differenti modalità di abbassamento dei privilegi:

- ***Permanente***: *setuid(getuid());*
- ***Temporaneo***: *seteuid(getuid());*

Nell'ipotesi che un processo parta SETUID (tipicamente root), abbassi i suoi privilegi con *seteuid(id)*, è possibile ***ripristinare i privilegi*** con la chiamata *seteuid(privileged_id);* dove *privileged_id* è lo user ID dell'utente privilegiato di partenza (tipicamente root).

Nei ***sistemi BSD*** viene introdotta una chiamata di sistema per impostare contemporaneamente RUID e EUID: *setreuid(uid,euid)*. Per lasciare inalterato uno dei due valori, basta fornire in input -1.

- L'*abbassamento permanente* dei privilegi si ottiene impostando entrambi i parametri ad un valore non privilegiato *setreuid(uid,uid);*
  In seguito, i parametri RUID e EUID non potranno che assumere il valore uid, e il ripristino a root è impossibile.
- L'*abbassamento temporaneo* dei privilegi si ottiene con la chiamata *setreuid(geteuid(),getuid());*
  NOTA: RUID ed EUID sono scambiati. Dopo lo scambio RUID è da *utente privilegiato* e EUID è da *utente non privilegiato*.
- Il *ripristino dei privilegi* si ottiene ancora con la chiamata *setreuid(geteuid(),getuid());*
  Dopo lo scambio RUID è da *utente non privilegiato* e EUID è da *utente privilegiato*.

Il programma *drop_rest_bsd.c* effettua l'abbassamento e il ripristino dei privilegi.

Si compili *drop_rest_bsd.c*:

gcc –o drop_rest_bsd drop_rest_bsd.c

Si esegua *drop_rest_bsd* una prima volta come *utente normale*:

./drop_rest_bsd

Viene stampato:

Prima di setreuid(euid,uid): RUID = 1000
Prima di setreuid(euid,uid): EUID = 1000
Dopo setreuid(euid,uid): RUID = 1000
Dopo setreuid(euid,uid): EUID = 1000
Dopo setreuid(euid,uid): RUID = 1000
Dopo setreuid(euid,uid): EUID = 1000

Si imposti il creatore a root:

sudo chown root drop_rest_bsd

Si imposti il bit SETUID:

sudo chmod u+s drop_rest_bsd

Si esegua *drop_rest_bsd* una seconda volta come *root*:

./drop_rest_bsd

Viene stampato:

Prima di setreuid(euid,uid): RUID = 1000
Prima di setreuid(euid,uid): EUID = 0
Dopo setreuid(euid,uid): RUID = 0
Dopo setreuid(euid,uid): EUID = 1000
Dopo setreuid(euid,uid): RUID = 1000
Dopo setreuid(euid,uid): EUID = 0

drop_rest_bsd.c

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {

 uid_t uid = getuid();
 uid_t euid = geteuid();

 printf("Prima di setreuid(euid, uid):
              RUID del processo = %d\n", uid);
 printf("Prima di setreuid(euid, uid):
              EUID del processo = %d\n", euid);
 if (setreuid(euid, uid) == -1) {
       printf("Non sono riuscito a scambiare
                     UID <-> EUID.\n");
       exit(1);
 uid = getuid();
 euid = geteuid();
 printf("Dopo setreuid(euid, uid):
              RUID del processo = %d\n", uid);
 printf("Dopo setreuid(euid, uid):
              EUID del processo = %d\n", euid);

 if (setreuid(euid, uid) == -1) {
       printf("Non sono riuscito
              a scambiare UID <-> EUID.\n");
       exit(1);
 }
 uid = getuid();
 euid = geteuid();
 printf("Dopo setreuid(euid, uid):
              RUID del processo = %d\n", uid);
 printf("Dopo setreuid(euid, uid):
              EUID del processo = %d\n", euid);
}
```

Nel processo che esegue *drop_rest_bsd.c* subito dopo l'abbassamento dei privilegi si ha che *RUID=0 (root)* e *EUID≠0 (utente normale)*.
In tal caso ci si chiede quali privilegi ha un processo con RUID=0.

In particolare, si consideri il programma *d_r_open_bsd.c* che:

1. Prova ad aprire il file */etc/shadow* subito dopo l'abbassamento dei privilegi;
2. Il file */etc/shadow* è di proprietà di root ed ha i seguenti permessi: *rw-------*;
3. È quindi leggibile e scrivibile solo da root (a differenza del file */etc/passwd* che ha permessi *rw-r--r--* ed è leggibile da tutti);
4. Il file */etc/shadow* contiene le password cifrate degli utenti, corrispondenti alle x in ciascuna riga di */etc/passwd*)

Dopo l'abbassamento dei privilegi (anche con RUID=0) *non è possibile leggere /etc/shadow*.

Si compili *d_r_open_bsd.c*:

gcc –o d_r_open_bsd d_r_open_bsd.c

Si esegua *d_r_open_bsd* una prima volta da *utente normale*:

./d_r_open_bsd

Viene stampato:

Prima di setreuid(euid,uid: RUID = 1000
Prima di setreuid(euid,uid): EUID = 1000
Dopo setreuid(euid,uid): RUID = 1000
Dopo setreuid(euid,uid): EUID = 1000
Non sono riuscito ad aprire /etc/shadow
Dopo setreuid(euid,uid): RUID = 1000
Dopo setreuid(euid,uid): EUID = 1000

Si imposti il creatore a root:

sudo chown root d_r_open_bsd

Si imposti il bit SETUID:

sudo chmod u+s d_r_open_bsd

Si esegua *d_r_open_bsd* una seconda volta:

./d_r_open_bsd

Viene stampato:

Prima di setreuid(euid,uid: RUID = 1000
Prima di setreuid(euid,uid): EUID = 0
Dopo setreuid(euid,uid): RUID = 0
Dopo setreuid(euid,uid): EUID = 1000
Non sono riuscito ad aprire /etc/shadow
Dopo setreuid(euid,uid): RUID = 1000
Dopo setreuid(euid,uid): EUID = 0

d_r_open_bsd.c

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {

 uid_t uid = getuid();
 uid_t euid = geteuid();

 printf("Prima di setreuid(euid, uid):
              RUID del processo = %d\n", uid);
 printf("Prima di setreuid(euid, uid):
              EUID del processo = %d\n", euid);
 if (setreuid(euid, uid) == -1) {
       printf("Non sono riuscito a scambiare
                    UID <-> EUID.\n");
       exit(1);
 }
 uid = getuid();
 euid = geteuid();
 printf("Dopo setreuid(euid, uid):
              RUID del processo = %d\n", uid);
 printf("Dopo setreuid(euid, uid):
              EUID del processo = %d\n", euid);

 if (open("/etc/shadow", O_RDONLY) == -1)
       printf("Non sono riuscito ad aprire /etc/shadow.\n");
 else
       printf("Sono riuscito ad aprire /etc/shadow.\n");

 if (setreuid(euid, uid) == -1) {
       printf("Non sono riuscito a scambiare UID <-> EUID.\n");
       exit(1);
       }
 uid = getuid();
 euid = geteuid();
 printf("Dopo setreuid(euid, uid):
              RUID del processo = %d\n", uid);
 printf("Dopo setreuid(euid, uid):
              EUID del processo = %d\n", euid);

}
```

Un limite dei sistemi BSD è che non gestiscono una ***terza coppia di credenziali***:

- *Saved User ID* (***SUID***);
- *Saved Group ID* (***SGID***).

Quando un processo parte, tali credenziali contengono una copia di quelle effettive: SUID=EUID, SGID=EGID.

Tale limite viene superato nei ***sistemi POSIX*** (***Portable Operating System Interface***) è una famiglia di standard specificati dalla IEEE Computer Society per mantenere la compatibilità tra diversi SO. Nei sistemi POSIX la chiamata di sistema introdotta da BSD, ovvero *setreuid(uid,uid)* è stata deprecata e viene sostituita da *setuid(uid)*. Tale chiamata imposta non solo l'effettivo ma anche il salvato, ovvero il SUID, ma POSIX non specifica meccanismi per consentire la lettura del SUID, quindi, queste credenziali in più sono quasi fantasma, perché non viene specificato come leggerle e non si capisce il loro utilizzo. I dettagli di tale gestione sono demandati allo specifico SO. Anche nei sistemi POSIX abbiamo *due differenti modalità* di abbassamento dei privilegi:

- ***Permanente***: *setuid(getuid());*
- ***Temporaneo***: *seteuid(getuid());*

I ***SO GNU/Linux*** implementano la ***LSB*** (***Linux Standards Base***). Si tratta di una evoluzione della SUS (Single UNIX Specification), che è l'evoluzione attuale dello standard POSIX. LSB sostituisce alcune funzionalità errate o insicure di POSIX/SUS con meccanismi propri.

- La chiamata di sistema *getresuid(uid,euid,suid)* consente di recuperare tutti gli user ID del processo invocante.
- La chiamata di sistema *setresuid(uid,euid,suid)* consente di impostare tutti gli user ID del processo invocante.

L'***abbassamento permanente*** dei privilegi si ottiene impostando tutti i parametri ad un valore non privilegiato *setresuid(uid,uid,uid);*

In seguito, i parametri RUID, EUID e SUID non potranno che assumere il valore *uid*. Il ripristino a *root* è impossibile.

Si compili *drop_priv_gnulinux.c*:

gcc –o drop_priv_gnulinux drop_priv_gnulinux.c

Si esegua il programma una prima volta:

./drop_priv_gnulinux

Viene stampato:

Prima dell'abbassamento: RUID = 1000
Prima dell'abbassamento: EUID = 1000
Prima dell'abbassamento: SUID = 1000
Dopo l'abbassamento: RUID = 1000
Dopo l'abbassamento: EUID = 1000
Dopo l'abbassamento: SUID = 1000
Non sono riuscito a ripristinare i privilegi

Si imposti il creatore a root:

drop_priv_gnulinux.c

```
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {

 uid_t uid, euid, suid;

 if (getresuid(&uid, &euid, &suid) == -1) {
       printf("Non sono riuscito ad
              ottenere uid, euid, ruid.\n");
       exit(EXIT_FAILURE);
 }
```

sudo chown root drop_priv_gnulinux

Si imposti il bit SETUID:

sudo chmod u+s drop_priv_gnulinux

Si esegua drop_priv_gnulinux una seconda volta:

./drop_priv_gnulinux

Viene stampato:

Prima dell'abbassamento: RUID = 1000
Prima dell'abbassamento: EUID = 0
Prima dell'abbassamento: SUID = 0
Dopo l'abbassamento: RUID = 1000
Dopo l'abbassamento: EUID = 1000
Dopo l'abbassamento: SUID = 1000
Non sono riuscito a ripristinare i privilegi

L'***abbassamento temporaneo*** dei privilegi si ottiene impostando EUID ad un valore non privilegiato *setresuid(-1,geteuid(),-1);*
In questo modo si preserva lo user ID salvato e si può effettuare il ripristino in seguito.
NOTA: *setresuid(-1,uid,-1)=seteuid(uid)*.

Il ***ripristino temporaneo dei privilegi*** si ottiene impostando EUID ad un valore privilegiato *setresuid(-1,privileged_ID,-1);* dove *privileged_id* è lo user ID dell'utente privilegiato ottenuto in partenza (tipicamente root).
Il programma *drop_rest_gnulinux.c* implementa la strategia vista.

Si compili *drop_rest_gnulinux.c*:

gcc –o drop_rest_gnulinux drop_rest_gnulinux.c

Si esegua il programma una prima volta:

./drop_rest_gnulinux

Viene stampato:

Prima dell'abbassamento: RUID = 1000
Prima dell'abbassamento: EUID = 1000
Prima dell'abbassamento: SUID = 1000
Dopo l'abbassamento: RUID = 1000
Dopo l'abbassamento: EUID = 1000
Dopo l'abbassamento: SUID = 1000
Dopo il ripristino: RUID = 1000
Dopo il ripristino: EUID = 1000
Dopo il ripristino: SUID = 1000

Si imposti il creatore a root:

sudo chown root drop_rest_gnulinux

Si imposti il bit SETUID:

sudo chmod u+s drop_rest_gnulinux

Si esegua *drop_rest_gnulinux* una seconda volta

./drop_rest_gnulinux

Viene stampato:

Prima dell'abbassamento: RUID = 1000
Prima dell'abbassamento: EUID = 0
Prima dell'abbassamento: SUID = 0
Dopo l'abbassamento: RUID = 1000
Dopo l'abbassamento: EUID = 1000
Dopo l'abbassamento: SUID = 0
Dopo il ripristino: RUID = 1000
Dopo il ripristino: EUID = 0
Dopo il ripristino: SUID = 0

```
 printf("Prima dell'abbassamento privilegi:
                RUID del processo = %d\n", uid);
  printf("Prima dell'abbassamento privilegi:
                EUID del processo = %d\n", euid);
  printf("Prima dell'abbassamento privilegi:
                SUID del processo = %d\n", suid);
  if (setresuid(uid, uid, uid) == -1) {
        printf("Non sono riuscito ad
                  abbassare i privilegi.\n");
        exit(EXIT_FAILURE);
  }
 if (getresuid(&uid, &euid, &suid) == -1) {
        printf("Non sono riuscito ad ottenere
                uid, euid, ruid.\n");
        exit(EXIT_FAILURE);
 }
 printf("Dopo l'abbassamento privilegi:
                RUID del processo = %d\n", uid);
 printf("Dopo l'abbassamento privilegi:
                EUID del processo = %d\n", euid);
 printf("Dopo l'abbassamento privilegi:
                SUID del processo = %d\n", suid);
 if (setresuid(-1, 0, -1) == -1) {
        printf("Non sono riuscito a
                ripristinare i privilegi.\n");
        exit(1);
 }
if (getresuid(&uid, &euid, &suid) == -1) {
        printf("Non sono riuscito ad
                ottenere uid, euid, ruid.\n");
        exit(EXIT_FAILURE);
 }
 printf("Dopo il ripristino privilegi:
                RUID del processo = %d\n", uid);
 printf("Dopo il ripristino privilegi:
                EUID del processo = %d\n", euid);
 printf("Dopo il ripristino privilegi:
                SUID del processo = %d\n", suid);
}
```

drop_rest_gnulinux.c

```
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {

    uid_t uid, euid, suid;
    uid_t priv_uid;

 if (getresuid(&uid, &euid, &suid) == -1) {
       printf("Non sono riuscito ad
              ottenere uid, euid, ruid.\n");
       exit(EXIT_FAILURE);
 }
 priv_uid = euid;
 printf("Prima dell'abbassamento privilegi:
              RUID del processo = %d\n", uid);
 printf("Prima dell'abbassamento privilegi:
              EUID del processo = %d\n", euid);
 printf("Prima dell'abbassamento privilegi:
              SUID del processo = %d\n", suid);
 if (setresuid(-1, uid, -1) == -1) {
       printf("Non sono riuscito ad
              abbassare i privilegi.\n");
       exit(EXIT_FAILURE);
 }
 if (getresuid(&uid, &euid, &suid) == -1) {
       printf("Non sono riuscito ad
              ottenere uid, euid, ruid.\n");
       exit(EXIT_FAILURE);
 }
 printf("Dopo l'abbassamento privilegi:
              RUID del processo = %d\n", uid);
 printf("Dopo l'abbassamento privilegi:
              EUID del processo = %d\n", euid);
 printf("Dopo l'abbassamento privilegi:
              SUID del processo = %d\n", suid);
 if (setresuid(-1, priv_uid, -1) == -1) {
       printf("Non sono riuscito a
              ripristinare i privilegi.\n");
       exit(1);
 }
 if (getresuid(&uid, &euid, &suid) == -1) {
       printf("Non sono riuscito ad
              ottenere uid, euid, ruid.\n");
       exit(EXIT_FAILURE);
 }

 printf("Dopo il ripristino privilegi:
              RUID del processo = %d\n", uid);
 printf("Dopo il ripristino privilegi:
              EUID del processo = %d\n", euid);
 printf("Dopo il ripristino privilegi:
              SUID del processo = %d\n", suid);
}
```

5. ALBERO DI ATTACCO

Le ***tipologie di vulnerabilità*** rispondono, di norma, alle seguenti domande:

- Sotto quali ipotesi si verificano?
- Quali conseguenze hanno?
- Come si possono mitigare?

Il ***modo non corretto di agire*** prevede di provare comandi a casaccio o copiare soluzioni messe a punto da altri, senza avere idea di cosa si stia facendo o utilizzare strumenti automatici di attacco, senza avere idea del loro funzionamento.

Il ***modo corretto di agire*** invece prevede la piena consapevolezza delle proprie azioni, conoscendo tutti i dettagli dell'ambiente che si sta studiando, identificando tutti i modi possibili (plausibili ed improbabili) di condurre un attacco, provare l'attacco sui sistemi su cui si ha il permesso di operare, capire nel dettaglio modalità e conseguenze dell'attacco e capire come mitigare l'attacco.

L'***albero di attacco*** è uno strumento utile per la conduzione ragionata di attività di attacco, rappresenta una vista gerarchica dei possibili attacchi ad un sistema, dove:

- Ogni nodo dell'albero è un'azione;
- Il nodo radice è l'azione finale dell'attacco;
- Ciascun nodo foglia è una azione iniziale dell'attacco;
- Ciascun nodo intermedio rappresenta un'azione preliminare per poter svolgere l'azione rappresentata dal nodo padre.

Esempio:

Supponiamo di voler aprire una cassaforte:

- Il ***nodo radice*** dell'albero rappresenta proprio l'obiettivo dell'attacco;
- La cassaforte può essere aperta se almeno una delle azioni rappresentate nei ***nodi foglia*** ha successo;
- Le ***azioni intermedie*** hanno bisogno, a loro volta, del successo di almeno un'altra azione preliminare. Alcune azioni necessitano l'esecuzione di più azioni preliminari, si modellano con un AND e un arco;
- Un ***possibile attacco*** è un OR di percorsi (incrocianti su un nodo AND) da nodi foglia al nodo radice.

Una volta definito, l'albero di attacco può essere arricchito con opportune ***etichette*** sui nodi, come fattibilità dell'azione (Possibile, Impossibile), costo dell'azione o probabilità di successo. Aggregando le etichette nel percorso da una foglia alla radice, è possibile ***stimare l'attacco***.

La macchina virtuale ***Nebula*** contiene esercizi di sicurezza, organizzati come sfide (***challenge***), ciascun esercizio corrisponde a un livello, per un totale di 20 livelli (Level00, Level01, …, Level19). Ciascun livello ***dichiara un obiettivo non banale*** che l'utente deve cercare di ottenere con ogni mezzo possibile. I livelli dovrebbero essere eseguiti in sequenza.

Gli account a disposizione sono di due tipi:

- ***Giocatori***:
  Un utente che intende partecipare alla sfida (simulando il ruolo dell'***attaccante***) si autentica con le credenziali seguenti.
  Username: levelN (N=00, 01, …, 19) e Password: levelN (N=00, 01, …, 19);
- ***Vittime***:
  Gli account flag00, …, flag19 simulano una vittima e contengono vulnerabilità di vario tipo.

C'è anche un account che simula un ***amministratore di sistema***: Username: nebula! e Ø Password: nebula!

L'elevazione dei privilegi a root può essere effettuata manualmente tramite il comando *sudo*.