TEST - LANGUAGE BASED TECHNOLOGY FOR SECURITY

Appello: 11 Giugno 2021

...

1. (SecurityAutomata)

   Consider a cloud that provides services for editing documents, storing them on a remote repository, and sharing them with other users. The front-end of the editor is run by a suitable App. The user can tag any of her documents as private. To avoid direct information leakage, we assume a security policy stating that private files cannot be sent to the remote repository in plain text, yet they can be sent encrypted.
   Define the security automaton describing the security policy described above.

   start_state-->s_0 -->(Encryption)-->s_1-->(Transfer)-->s_2-->final_state
   s_0-->(Not Encryption)-->s_0
   s_1-->(Not TRansfer)-->s_1

2. (Memory Corruption)

   Control Flow Integrity (CFI) is a defense mechanism that restricts the freedom of an attacker to corrupt memory. Briefly discuss the main features of CFI.

La CFI permette di osservare il comportamento del programma, quindi abbiamo un modello tramite cui osserviamo il comportamento del programma in modo tale da avere un' astrazione di quello che deve fare ovvero per fare in modo esegua le azioni volute dal programmatore. Nel caso in cui il programma non rispetti le intenzioni che il programmatore aveva in mente, vuol dire che il programma è compromesso e quindi bisogna bloccare l'esecuzione. L'idea è che il flusso di controllo del programma viene astratto tramite una struttura che si può costruire a partire dalla rappresentazione intermedia dei programmi, quindi dal codice sorgente, che si chiama CONTROL FLOW GRAPH. Esso tiene conto esattamente del flusso di esecuzione. Ci sono degli strumenti che permettono di ricostruire il control flow graph del programma anche facendo un'analisi dei binari, quindi non è soltanto un qualcosa che ha a che vedere con il codice sorgente, ma sono strumenti di analisi statica. In questo modo, quindi, si ha a disposizine un astrazione del comportamento che da un'idea del profilo dell'esecuzione del programma, perché ovviamente è un' astrazione dell'esecuzione, fa solo vedere bene come il flusso dell'esecuzione fluisce all'interno della struttura del programma. A questo punto, avendo la possibilità di definire delle politiche esplicite di sicurezza, si uniscono le politiche esplicite di sicurezza col flusso di esecuzione del programma e in questo modo si ha un meccanismo che permette di fare quello che si chiama la detection del programma in base ai comportamenti. Gli ingredienti della CFI sono: il comportamento aspettato definito dal CFG, un inline reference monitor che va a controllare se queste etichette uniche che vado ad inserire nel programma siano quelle aspettate e questo avviene per ogni chiamata e ritorno al programma e infine la randomizzazione che mi permette di avere protezione contro l' attaccante cos' che non possa fare delle operazioni di guess sull esecuzione andando analizzare il programma sulla sua macchina e poi aspettare di utilizzare le etichette che ha scoperto, i pattern unici che ha scoperto. L'algoritmo di enforcement, che mette in opera la CFI, prevede alcune ipotesi : il codice è immutabile, vuol dire che ci sono dei meccanismi di certificazione della struttura, dei binari che non permettono di far modificare il codice e inoltre l'attaccante non può modificare le etichette, quindi gli indirizzi targhe che sono univocamente determinate dal compilatore e fanno parte della trusted computed base. Tramite questo approccio è possibile evitare la coding injection di una una etichetta legale, perché ovviamente la trusted computer base del mio apparato che carica in memoria il mio programma, viene mandato in un area che viene chiamata eseguibile. L'altra area, quella dei dati su cui il programma può operare, è non eseguibile, quindi io sono certo che anche se ho individuato l'etichetta giusta, cioè voglio servire qualcosa che sta nella parte della memoria non eseguibile, a run time non potrà accadere. Lo stack potrebbe essere modificato, però non si possono modificare i valori dei registri perché i valori dei registri fanno parte della parte hardware della trusted computer base e quindi facendo parte della trusted computer base l'attaccante non li può modificare. Si potrebbe però cercare di manipolare il CFG. Si chiamano mimicry Attack. Ci permette di definire delle politiche e delle politiche di comportamento sul flusso di esecuzione, però non ci dice niente, ad esempio sui parametri. Si potrebbero avere degli argomenti sbagliati, avere degli attacchi che hanno dei dati passati come parametri, ad esempio a chiamate di sistema non corretti.

3. (Type Systems for Information Flow)

   Consider the following program fragment

   if x then {y:=1;} else {y:=2;} z := 3; halt;

   In which case the program above does not leak information?
   Outline the corresponding type derivation.

   Il contesto è definito come gamma(x), dobbiamo andare a verificare come proprietà di certificazione che il contesto in modo particolare le etichette di sicurezza associate alla x join il livello di sicurezza dell espressione che è esattamente 1 o 2, deve essere compatibile con livello di sicurezza della x. Allora se noi riusciamo a verificare che questa proprietà contestuale vale vuol dire che questo condizionale soddisfa la non interferenza. Se gamma(x) = L, gamma(y)=L, gamma(Z)=L per esempio non vi è alcun leak di informazione avendo tutte le variabili e di conseguenza il contesto un livello low, ma supponiamo che gamma(X) = H e gamma(y)=L in questo caso è possibile avere leakage di informazione in quanto il valore che assumerà y determinerà se x è true o false.

4. (Taint Analysis)
   Consider the program below written in a simplified language whose variables can only take string values.

   a= read ()
   q = null
   if(a) ! = null {q = “select”+ a;}{q = “ select * from *”;}
   makeQuery (q)

   Explain if a taint warning should be issued when the query statement is going to be executed.

Leggendo qualcosa dalla tastiera assumo che il dato sia tainted, pertanto a<=tainted, prendendo il ramo then avrò tainted <= q prendendo il ramo else avrò untainted <= q, ma qui abbiamo che il ramo then e il ramo else hanno due valori di taintness differenti in particolare nel caso in cui venisse preso il ramo else avremmo che q è untainted ma nel caso del ramo then si avrebbe q tainted che è l'unico valore che soddisfa il sistema pertanto si dovrebbe sollevare un eccezione.

5. Discuss benefits and drawbacks of applying static analysis techniques to handle security policies

Consente di avere un maggiore overhead a tempo statico evitando a run time di avere un interruzione del programma, ma con una sovrapprossimazione offerta dall'analisi statica a causa della sua natura predittiva è possibile avere un maggior controllo, a volte anche superiore rispetto al necessario, consentendo un flusso corretto per ogni possibile esecuzion a runtime.