AVP Project draft - 0.1

Barbon - De Zotti - Moscardo

June 24, 2014

1 Aim of our project

Vogliamo verificare se delle variabili dell'applicazione contengono dati sensibili, a seguito dell'uso di determinati metodi che in esse hanno salvato i loro risultati.
Nota: utilizzeremo l'abbreviazione "metodi sensibili" per metodi che generano dati contenenti informazioni sensibili e "dati sensibili" per indicare i dati generati da metodi sensibili. Utilizzeremo il verbo "esportare" per indicare tutti i possibili metodi di salvataggio/esportazione di dati sensibili.
La domanda principale diventa quindi: in un'applicazione che genera dati sensibili, sono questi dati esportati?

2 Solution Proposal

Utilizziamo Data Flow Analysis. Ipotizziamo di nominare quest'analisi "Safeness Analysis". Quest'analisi determina le variabili che esportano dati sensibili ad ogni entry ed exit point di un nodo. Consideriamo l'insieme di valori formato da (x, x[...], n), dove x[...] è la catena formata da variabili usate nella definizione di x. La variabile x e la sua catena sono esportate e potenzialmente sensibili se:

- esiste un percorso p da n a m
- la variabile x è esportata e potenzialmente sensibile in n
- la variabile x non è mai ridefinita in p

Consideriamo quindi le operazioni che generano e killano l'informazione:

gen[B] = {*var x is exported*
OR
var x is used by an exported variable y (*so x is added in the chain of y*)}

$kill$[B] = {*var x is not created by a sensible method*
OR
chain of var x contains only variables that are not sensible }

Definiamo ora:

- direzione dell'analisi: backward
- confluence operator: unione, $\text{out}[B] = \cup \text{in}[S]$, *over the successor S of B*
- inizializzazione: insieme vuoto $\emptyset$

Possiamo formalizzare le equazioni della Safeness Analysis:

$$SA_{exit}(p) = \begin{cases} \emptyset \qquad if\ p\ is\ a\ final\ point \\ \\ \cup \{SA_{entry}(p) \mid q\ follows\ p\ in\ the\ CFG\} \end{cases}$$

$$SA_{entry}(p) = \text{gen}_{\text{SA}}(p) \cup (SA_{exit}(p) \setminus \text{kill}_{\text{SA}}(p))$$

Se alla fine dell'analisi otteniamo l'insieme vuoto, il programma sarà SAFE. Altrimenti il programma sarà UNSAFE e avremo ottenuto l'insieme di tutte le variabili sensibili esportate.

3 Examples

3.1 Example 1

```
x = mysensibledata();      ...
...
y = x + 1;                 No kill
...
export(y);                 gen : y
```

Partiamo dalla fine. Gen sull'ultima istruzione ci genera y. Sulla penultima dobbiamo killare y? No. Facciamo gen di x sulla catena di y. Infatti, anche se non viene generato da un metodo sensibile, non sappiamo ancora se x sia sensibile o meno. Nel caso lo fosse, considereremmo anche y sensibile. Quindi per il momento non abbiamo kill. Continuando a risalire nel codice, troviamo il punto dove viene definita la variabile x. Si tratta di un metodo sensibile, quindi non facciamo kill. Inoltre, ora sappiamo che anche y è sensibile. Questo esempio risulta quindi UNSAFE.

3.2 Example 2

```
x = 5;
...
y = x + 1;
...
export(y);
```

In questo caso abbiamo un comportamento simile al precedente, ma la situazione cambia nel momento in cui scopriamo che x non proviene da una funzione sensibile. A questo punto, verranno killate sia x che y. Alla fine avremo l'insieme vuoto, ciò significa che quest'esempio è SAFE.

4 Conclusioni

Risulta quindi necessario dover utilizzare una catena/array di riferimenti, per sapere tutta la lista di variabili che ipotizzate sensibili che sono collegate. Ciò ci permetterebbe inoltre di gestire tutte le problematiche relative alle variabili passate per riferimento. Questa bozza prende in considerazione esempi in pseudocodice 'imperativo'. Tutto questo lavoro, se confermato, dovrà quindi essere rivisto per poter essere adattato al paradigma ad oggetti del linguaggio android (considerando quindi le relative problematiche).