

the two conditions in Q_c . The edges in Q_{cc} and Q_{ec} interconnect the conditions and exploits.

[0052] Typical Analyses of Attack Graphs in Relational Queries: Typical analyses of attack graphs and how to rewrite those analyses as relational queries based on our model will now be disclosed. In the following discussion, the queries are against the relations (or views) given by Definition 2.

[0053] Vulnerability-Centric Alert Correlation and Prediction: The alert correlation method maps a currently received intrusion alert to the corresponding exploit. Then, it reasons about previous exploits (alerts) that prepare for the current one and possible exploits in the future [20]. The key difference between this analysis and the one used to generate the attack graph is that the conjunctive nature of the required relation should be ignored here. The relationship between alerts is usually regarded as casual instead of logical [10, 3]. Such a conservative approach is more appropriate in this context because alerts may have been missed by intrusion detection systems.

Example 6

[0054] In FIG. 1, suppose the current alert maps to the exploit (2, 1, A). The backward search will first reach conditions (1, x) and (2, y) and then follows (2, y) to (3, 2, A) and (1, 2, A) to find a previous correlated alert if there is any, or to make a hypothesis for a missing alert, otherwise. The search continues from (1, 2, A) to (1, y) and (2, x), then from (1, y) to (3, 1, A) (the branch to (2, 1, A) is a loop and hence ignored) and consequently to (1, x) and (3, y). The search stops when it reaches only initial conditions or if a loop is encountered.

[0055] Definition 3 states the relational queries corresponding to the backward search in Example 6. The forward search may be realized in a similar way and hence is omitted. First, the relation Q_3 includes the conditions reachable from the current exploits while ignoring the conjunctive relationship between those conditions. Second, subtracting from Q_3 the initial conditions in hc and the previously visited conditions in Q_5 (to avoid loops) yields the reachable conditions and consequently the exploits in Q_4 . The above two steps should be repeated until no more conditions are left (that is, all the conditions are in hc or in Q_5). The exploits encountered in this process may be collected in Q_A as the final result. Loops should be avoided in this process because the set union operation does not keep duplicates and the relation Q_5 ensures each condition to be visited at most once.

[0056] Definition 3. Given hh(HH), hc(HC), cv(CV), vc(VC), and (h_s, h_d, V) , let $Q_3(HC)$, Q_5 , and Q_A be empty relations and $Q_4(EX)=\{(h_s, h_d, V)\}$. Define

$$\begin{aligned} \text{[0057]} \quad Q_3 &= \Pi_{hd,c} \\ & (Q_4 \bowtie \sigma_{F=D}(cv)) \cup \Pi_{hs,c} (Q_4 \bowtie \sigma_{F=S}(cv)) \end{aligned}$$

$$\begin{aligned} \text{[0058]} \quad Q_4 &= \Pi_{Hs, Hd, V} (\sigma_{Hd} \quad d=H \\ & \quad \sigma_{VC} \quad C=vc.c((hh \times (Q_3 - hc - Q_5) \times vc)) \end{aligned}$$

$$\text{[0059]} \quad Q_5 = Q_5 \cup Q_3$$

$$\text{[0060]} \quad Q_A = Q_A \cup Q_4$$

Example 7

[0061] Table 3, shown in FIG. 5, illustrates an example of analyzing attack graphs for alert correlation and prediction.

Specifically, Table 3 shows the three iterations corresponding to the backward search in Example 6. The first iteration starts from the given exploit (2, 1, A) and reaches two exploits (1, 2, A) and (3, 2, A) through the condition (2, y). The second iteration reaches (3, 1, A) and (2, 1, A) through (1, y). The exploit (2, 1, A) leads to two previously visited conditions (that is, a loop) and the other exploit (3, 1, A) reaches only initial conditions. Consequently, no new exploit appears in Q_4 in this iteration and the search terminates.

[0062] Enumerating Relevant Attacks and Network Hardening: Enumerating the relevant exploits (those that appear in at least one sequence of attacks leading to the goal conditions [1]) and finding a network hardening solution (given goal conditions represented as a logic formula of initial conditions [12]) share a similar backward search in the attack graph, as illustrated in Example 8 and Example 9, respectively.

Example 8

[0063] As illustrated in FIG. 1A, one starts from a given goal condition (1, y) and searches backwards in the attack graph. First, the two exploits (3, 1, A) and (2, 1, A) are reached. The former branch ends at initial conditions, and the latter leads to one initial condition (1, x) and an intermediate condition (2, y). The condition (2, y) then leads to (3, 2, A) and (1, 2, A). The former ends at initial conditions, and the latter leads to a loop back to (1, y). The relevant exploits with respect to the goal condition (1, y) are thus (2, 1, A), (3, 1, A), and (3, 2, A) (the exploit (1, 2, A) is not relevant because it can never be realized before satisfying the goal (1, y) itself).

Example 9

[0064] With a similar search, one can transform the goal condition (1, y) into a logic formula of initial conditions as follows (by regarding the exploits and conditions as Boolean variables). In the fourth line, the value FALSE replaces the second appearance of the goal condition (1, y), because it is a predecessor of (1, 2, A), indicating a loop. The final result says that if any of the two conditions (1, x) and (3, y) is disabled, then the goal can no longer be satisfied.

$$\text{[0065]} \quad (1, y) = (3, 1, A) \vee (2, 1, A)$$

$$\text{[0066]} \quad = (1, x) \wedge (3, y) \vee (1, x) \wedge (2, y)$$

$$\text{[0067]} \quad = (1, X) \wedge (3, y) \vee (1, x) \wedge ((3, 2, A) \vee (1, 2, A))$$

$$\text{[0068]} \quad = (1, x) \wedge (3, y) \vee (1, x) \wedge ((3, y) \wedge (2, x) \vee (2, x) \wedge \text{FALSE})$$

$$\text{[0069]} \quad = (1, x) \wedge (3, y)$$

[0070] The key differences between the above backward search and that used for correlating alerts are as follows. First, the conjunctive nature of the require relation should be considered. In Example 8, the exploit (1, 2, A) is not relevant, because one of its required conditions (1, y) is not satisfiable, even though the other required condition (that is, (2, x)) is already satisfied. Second, duplicate appearances of exploits and conditions should be kept. This is required for obtaining sequences of relevant exploits leading to the goal, as well as for generating the logic formula in network hardening. In the former case, different sequences may share common exploits or conditions, whereas the logic formula in the second case clearly contains duplicates. In order for the