

search to traverse an exploit or condition for multiple times, the set union operation needs to keep duplicates. Hence, loops must be avoided by maintaining a predecessor list for each vertex as in standard breadth-first search (BFS) [2] (although the search discussed above is different from a BFS).

[0071] Definition 4 states the relational queries used to enumerate relevant exploits or to generate the logic formula in network hardening. The two queries simply traverse the attack graph given by Definition 2. The two relations in the definition keep duplicates in set union operations. Notice that the actual construction of the logic formula (adding the and or connectives) is external to the relational queries and may easily be incorporated.

[0072] Definition 4. Given relations hh(HH), hc(HC), cv(CV), vc(VC) and a nonempty relation $Q_7(HC)$, let $Q_6(EX)$ be an empty relation. Define

$$[0073] \quad Q_6 = \Pi_{Hs, Hd, v}((Q_7 - hc) \bowtie Q_{ec})$$

$$[0074] \quad Q_7 = \Pi_{H, C}(Q_6 \bowtie Q_{cc})$$

Example 10

[0075] FIG. 6 shows Table 4, an example of enumerating relevant exploits and network hardening. Specifically, Table 4 shows the iterations corresponding to the procedure in Example 8 and Example 9. Originally, $Q_7\{(1, y)\}$.

[0076] Reachability From Subsets of Initial Conditions and Incremental Updates of Attack Graphs: Many analyses ask a similar question, that is whether the goal condition is still satisfiable if a given subset of initial conditions are disabled. The question may arise when trying to determine the potential effect of enforcing a security measure (so some initial conditions will be disabled), or when trying to decide whether the goal condition is reachable with only stealthy attacks [18]. The question may also be asked simply because the network configuration has changed and some initial conditions are no longer satisfied (on the other hand, new initial conditions can be easily handled with more iterations of the queries in Definition 2.) In each case, it may be possible to recompute the attack graph from scratch, with the given conditions removed from the relation hc. However, this may not be desirable, especially when the attack graph is much larger than the set of conditions to be disabled. Instead, one may incrementally update the attack graph by computing the effect of disabling the given conditions. The conjunctive nature of the required relation may be taken into account, but in a different way, as illustrated in Example 11.

Example 11

[0077] In FIG. 1, suppose the condition (2, x) is disabled. Then the exploits (1, 2, A) and (3, 2, A) may no longer be realized. Then the condition (2, y) becomes unsatisfiable, because the condition (2, y) may only be implied by the above two exploits. Finally, the exploit (2, 1, A) may no longer be realized. However, the condition (1, y) should still satisfiable, due to another exploit (3, 1, A).

[0078] Example 11 shows that such a negative analysis is quite different from the previous ones. The previous searches are unidirectional in the sense that the edges are only followed in one direction (either forwards or backwards). However, the above analysis follows edges in both directions. For example, after the forward search reaches the

condition (1, y) from the exploit (2, 1, A), it must go back to see whether other exploits also imply the condition (1, y) (in this case, the exploit (3, 1, A) does so). Definition 5 states the relational queries for this purpose. The first query simply derives unrealizable exploits from unsatisfied conditions. The next three queries use two set difference operations to derive the unsatisfied conditions while taking into account the conjunctive nature of the require relation. Finally, the results may be collected.

[0079] Definition 5. Given relations hh(HH), hc(HC), cv(CV), vc(VC) and a nonempty relation $Q_{11}(HC)$ as a subset of hc, let $Q_8(EX)$, $Q_9(EC)$, $Q_{10}(EC)$, Q_e , and Q_c be empty relations. Define

$$[0080] \quad Q_8 = \Pi_{Hs, Hd, v}(Q_{11} \bowtie Q_{cc})$$

$$[0081] \quad Q_9 = Q_8 \bowtie Q_{ec}$$

$$[0082] \quad Q_{10} = Q_{ec} \bowtie \Pi_{H, C}(Q_9) - Q_9$$

$$[0083] \quad Q_{11} = \Pi_{H, C}(Q_9) - \Pi_{H, C}(Q_{10})$$

$$[0084] \quad Q_e = Q_c \cup Q_8$$

$$[0085] \quad Q_c = Q_c \cup Q_{11}$$

Example 12

[0086] FIG. 7 shows Table 5, an example of incremental updates. Specifically, Table 5 shows the iterations corresponding to the procedure in Example 11. Originally, $Q_{11} = \{(2, x)\}$.

[0087] Empirical Results: As proof of concept, the analyses discussed in the previous section were implemented. The queries were written in PL/SQL and tested in Oracle 9i in its default settings on a Pentium IV 2 GHz PC with 512 MB RAM. Preliminary experiments tested the queries against the attack scenario originally studied in [18, 1] 3. The results of the analyses match those in the previous work, which justifies the correctness of the techniques. Next, the performance of the techniques were tested. There were two main objectives. First, determine whether the running time of the queries is practical for interactive analysis. For most decision support systems, the typical delay to a query that is considered as tolerable in interactive analyses is usually in a matter of seconds. Such a short delay is also critical to the analysis of attack graphs, especially when the analysis is used for real-time detection and prevention of intrusions.

[0088] Second, determine whether the techniques scale well in the size of attack graphs. Although the attack graph may be very large for a large network, an analysis and its result usually only involves a small part of the attack graph. The running time of an analysis thus depend on how efficiently an analysis searches the attack graph. Mature optimization techniques available in most databases may transparently improve the performance and make the analyses more scalable. To test the queries against large attack graphs in a manageable way, the number of vertices in the original attack graph were increased by randomly inserting new hosts with random connectivity and vulnerabilities. The same set of analyses was executed in the new network and the running time of each analysis measured. The main results are shown in FIG. 8. All the results have 95% confidence intervals within about 5% of the reported values.

[0089] The left-hand side shows the running time of generating the attack graph in the size of that attack graph.