Detecting Cascade Vulnerability in Linear Time

J. D. Horton*
Faculty of Computer Science
University of New Brunswick
Fredericton, N. B., E3B 5A3
Canada
email: jdh@unb.ca

July 25, 2000

*supported by a research grant from NSERC

Abstract

The cascade vulnerability detection problem asks whether an opponent can use interconnections to pass data improperly across a network of individually accredited systems without having to defeat any single system that is rated high enuf to be judged safe for the particular data flow. In the most general setting, an algorithm is given of time complexity $O(CD^2U^2 + LDU)$, where C is the number of computers, D is the number of different classes of data, U is the number of different classes of user, and L is the number of links in the network. In the case with a linear data classification, the time complexity reduces to $O(CD^2 + LD)$, which improves on the previous best complexity of $O(C^3D)$. The case with parallel linear data classes is also solved in $O(CD^2 + LD)$ time.

Keywords: algorithm, network security, cascade vulnerability detection problem

1 The problem

Cascade vulnerability can occur when using the interconnected accredited system approach of the Trusted Network Interpretation, as defined in the Red Book [7], section C3.2, pp.249–259. Can a penetrator use the connections of a network to compromise information across a range of security levels that is greater than the accreditation range of the component systems one must defeat to do so?

For example, suppose that system A has both Top Secret and Secret data on it, and has some users who are cleared to only to read Secret but not Top Secret data. For this to be allowed, the system must have an accredited Trusted Computing Base (TCB) rating of B2[6]. Similarly, a second system B could have both Confidential data and Secret data on it, with some users who are only cleared to a Confidential level. System B needs to be accredited with a TCB rating of B1 for this to be allowed[6]. Suppose further that the two systems are connected in such a way that it is possible for Secret data on system A to be read by users on system B who have been cleared to the Secret level. Now if a penetrator could break both systems, it would be possible for Top Secret data on system A to be treated like Secret data, and to be sent to system B on the network. Then a user with only a Confidential clearance on system B would have access to the Top Secret data on system A if both systems have been pentrated. But a system with Top Secret data and Confidential users is supposed to have an accreditation rating of at least B3[6]. Breaking both a B1 and a B2 TSB accredited system should not be as difficult as breaking a single B3 accredited system.

Two problems arise regarding cascade vulnerability: detection and correction. The detection problem, deciding whether a network contains a cascade vulnerability, is the problem tackled in this paper. The correction problem, deciding the cheapest way in which to modify the network so as to remove all cascade vulnerabilities, is shown to be NP-hard in [4]. A simulated annealing approach to the correction problem is proposed in [3].

2 A generalization

The linear order of the classification of data (Confidential is less than Secret is less than Top Secret) is not the only way that data (and users) may be

classified. Typically the linear order (called hierarchical in the security literature) implies that a user cleared to a certain level also is automatically cleared for all lower levels. Sometimes the classes are also split into sets of parallel classes. For example, in a company a user may be cleared to read secret engineering data but not secret personnel data, or vice versa. For the much of this paper, no restrictions on the classification scheme is given. It is assumed only that the data to which each user is allowed to access is well defined. Thus the classification scheme forms a relation between the set of data, and the set of users. One simplifiction which can be done, is to consider two users to be equivalent if they are allowed access to exactly the same data. Similarly two pieces of data can be considered equivalent if they can be accessed by the same users. This gives use data classes and user classes defined by the equivalence relations.

A classification scheme now becomes a relation on the sets of these classes. In a linear (hierarchical) scheme with D data classes, the number of data classes is D, and so is the number of classes of user, since the classes of user perfectly mirror the data classes in this case. In a security classification scheme with K levels of data and P parallel streams, the number of data classes is KP, while the number of possible user classifications is K^P, as a user could be cleared to different levels in the different streams. For example, a sales manager might have top secret clearance on sales data, secret clearance on personnel data, confidential data on accounting information, but only unclassified clearance on engineering data. For the rest of this paper we assume that the number of data classes is D, while the number of user classes is U.

Operating systems can be given trusted computer base or TCB ratings, so that a user can have access to a computer on which is stored data to which the user is not cleared, along with data to which he is cleared, and requires access. In the usual totally ordered security scheme, the required TCB rating of a computer system is determined by the highest data classification on the machine, and the lowest clearance of a user[6]. In the more general setting considered in this paper, one can note that a rating is determined by a relation R on the set of data classes and the set of user classes. We define this relation R by defining (a,b) to be in R, if both data class a and user b are allowed to be on the same machine rated R.

When two computers are allowed to be connected together by some communication link, the situation becomes more complicated. We assume that

the link is only one way, and that only data in a given data class can be sent on the link. Any user who can read the given data class on the head of the link can obtain data of the given class on the link.

3 The algorithm

There have been several algorithms suggested which can detect whether a given computer network has a cascade vulnerability [2, 4, 5] but all make assumptions about the types of user and data classes, and all have time complexity which is cubic in the number of nodes in the network. Let C be the number of computers in the network, and let L be the number of communication links. We assume that the network is connected, so that C is at most $L+1$. In the next section an algorithm is given which has a time complexity linear in the size of the network, that is linear in $C+L$, assuming that D and U are constants.

The idea of the algorithm is to create a graph from the network in which paths correspond to the way that data can flow legally. Then for each data-user pair (d, u) modify the graph to correspond with all computers, whose TCB ratings are too low to have both data class d and user class u on them, being compromised. This is followed by searching the graph for a possible data flow from d to u. If such a path is found, then a cascade vulnerability exists.

The programmer has many choices in building the algorithm, due to symmetries in the problem definition. First we give one definition, and then we give some alternatives and some possible improvements in particular cases.

Constructing the graph For each machine c in the network, for each data class and user class on it, x, define a vertex of the graph, (c, x). Connect each data class vertex (c, d) to each user class vertex (c, u) if that user class u is allowed to read that data class d. In addition, for each data link in the graph from machine c to machine b at data class d, add an edge from (c, d) to (b, d). The graph now gives all the single step legal data flows in the whole computer network. The number of vertices is $O(C(D+U))$, the number of edges is $O(CDU + L)$.

The modification of the graph: For each pair (d, u), where d is a data class and u is a user class, the graph is modified for each computer c which has a TCB rating which does not include the pair (d, u) in its defining relation.

Edges are added in such way that paths are formed between all the vertices (c, e) where e is any data class, so that all users can read all data on computer c. In effect we are "breaking" all those computers which do not have a strong enuf accreditation rating to have both d and u on them. One way to do this, assuming that every user class is allowed to read some data class on the machine, is to connect all the data nodes (c, d) in a cycle, so at most d new edges are needed per computer. The total number of edges added is $O(CD)$, which does not affect the asymptotic upper bound on the number of edges in the graph. Adding these edges in effect simulates the "breaking" of the security system of computer c.

The search for a cascade vulnerability: The graph is now searched starting at the vertices (n, d) where n represents any machine with data class d on it. If a vertex (m, u) can be reached, where m is any machine, then a cascade vulnerability has been found. Since a graph search can be done in time linear in the size of the graph (number of edges plus the number of vertices), this step takes $O(CDU + L)$ time (see any text on graph algorithms such as [1, chapter 23].

The asymptotic complexity: The modification and search can be done for each pair (d, u), which is $O(DU)$ times. Then the overall complexity is $O(CD^2U^2 + LDU)$. This is linear in the number of computers plus the number of communication links, and hence is linear in the size of the computer network. However the number of classes occurs to the fourth power, which in the general case suggested above could be rather large.

4 Alternatives and improvements

One symmetry in the problem statement is that of interchanging the roles of data classes and user classes. The modification of the graph now consists of connecting nodes between pairs of user classes, instead of pairs of data classes. One point which must be checked is that every data class on a given machine also has a user class on it which can read it. Otherwise there could be data which is not readable by any user. In addition, any links between two different computers $c1$ and $c2$ at a level d now must be connected from a pair $(c1, u)$ to $(c2, d)$, where u is a user class on computer $c1$. Also the graph search algorithm could start from the user class and search for the data class, instead of the other way around. The resulting asymptotic complexity

is still the same in the worst case. It is also possible to treat each computer separately depending on whether the number of data classes or user classes is greater, but no asymptotic improvement is made.

A more important alternative, which can result in a faster algorithm, is to order the way in which the data-user pair searches are done. Suppose that the TCB ratings are linearly ordered by inclusion, which is a reasonable assumption. If all the (d, u) pairs are done one after the other for a given d, and the u's are ordered in such a way that the TCB ratings which are broken occur in order, then a factor of U can be removed from the upper bound time-complexity, as each edge needs to be searched only once over U different searches. Each of the U searches (other than the first one) starts from all the nodes found by the previous search. New edges are added to the list to be searched when they are created by the graph building procedure. This can always be done for all data classes d giving an upper bound of $O(CD^2U + LD)$. A similar speedup is possible in the symmetric situation by searching from a user class u and searching for the data classes in order consistent with the TCB ratings. The complexity becomes $O(CDU^2 + LU)$.

In the special case when the data classes are linearly ordered, then the algorithm can be improved further. Here the user classes can now be identified with the highest data class which the user can read, so $D = U$. Thus the data classes and the user classes can be identified with $1, 2, \ldots, D$. The graphs can now be constructed somewhat differently. Instead of needing nodes for both users and data, we can just have pairs (c, i) where c is a computer and i is a data class or user class on it. Let $h(c)$ be the class of the highest data on a computer c, and let $l(c)$ be the class of the lowest user on c. We only need to let i range from $l(c)$ to $h(c)$. The only edges that we need are (c, i) to $(c, i + 1)$ assuming both nodes meet the above requirement, that is, $l(c) <= i < h(c)$. Of course we still need the edges corresponding to the links between computers. The number of edges is now $O(CD + L)$.

We now must do a graph search for each pair (d, u) where $d > u$. When a pair (d, u) is searched, for each computer whose TCB rating which does not allow this pair, a single edge is added, from $(c, h(c))$ to $(c, l(c))$. Now all the nodes of c are connected in a simple circuit. Thus at most C new edges are added, leaving a graph with $O(CD + L)$ edges. Doing $\binom{n}{2}$ searches now requires $O(CD^3 + LD^2)$ time. If the TCB ratings are linearly ordered as well, as is the usual case, then again D searches can be done in the time of one

search and the time complexity is reduced to $O(CD^2 + LD)$. This compares favorably with the complexity of $O(C^3D)$ of the algorithm in [4], the fastest published algorithm, unless the number of computers is very small compared to the number of data classes. Below we call this the linear case.

5 The case of parallel data classes

Consider the case where there are P parallel data classes of size K, so that $D = KP$ and $U = K^P$. We assume that the TCB ratings of the computers are independent of the parallel classes, and are dependent only on the K levels of security. Thus if Top Secret engineering data is stored on one machine, and a user who is cleared only to confidential on the engineering parallel class is allowed access to that machine, then that machine must have a B3 TCB accreditation. This is a practical assumption, because it would be difficult to develop new accredition schemes for different parallel classes. We also assume that the TCB rating system is linear under set inclusion. The difficulty is that the number of user classes is too large for the direct application of the algorithm developed above, if the number of parallel classes is very large. However within each parallel class, the situation is exactly like that of the linear case considered above. Note that each user class is the union of at most P classes, one for each parallel data class. So instead of running the algorithm on all the data classes at once, one can run the algorithm on each parallel class independently. The number of user classes becomes exactly the number of data classes, thereby avoiding the exponential complexity.

One step in the algorithm for the linear case must be modified. When the computers are being "broken", it is assumed that all users on the machine can read some data on the machine. This assumption may not be true in this case, because a machine may not be used for some parallel class, or used only for a restricted set of data levels in some parallel class. In this case when the machine is "broken", all user classes must also be joined to the cycle of data classes. This does not affect the overall complexity of the algorithm.

The time complexity for each parallel class is $O(CK^2)$, so the overall complexity is now $O(CK^2P + LKP)$. Assuming that all parallel classes actually use all the data classifications, so that $D = KP$, this simplifies to $O(CKD + LD)$, which is also $O(CD^2 + LD)$.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, *Introduction to Algorithms*, MIT Press and McGraw-Hill, 1990.

[2] J. Fitch, L. Hoffman, The cascade problem: graph theory can help, *Proceedings of the 14th National Computer Security Conference*, 1991, 88–100.

[3] S. Gritzalis and D. Spinellis, The cascade vulnerability problem: the detection problem and a simulated annealing approach for its correction, *Microprocessors and Microsystems*, 21(1998), 621–627.

[4] J. D. Horton, R. Harland, E. Ashby, R. H. Cooper, W. F. Hyslop, B. G. Nickerson, W. M. Stewart, O. K. Ward, The cascade vulnerability problem, *J. Computer Security*, 2(1993), 279–290.

[5] J. K. Millen, Algorithm for the cascading problem, In Internet IEEE Cipher News Group, J.P. Anderson, Ed., June 25 IEEE Cipher Forum on DOCKMASTER.NCSC.MIL 1990.

[6] National Computer Security Center, Technical Rational Behind CSC-STD-003-85: Computer Security Requirements, Yellow Book CSC-STD-004-85, 1985.

[7] NCSC Trusted network interpretation of the trusted computer system evaluation criteria Red Book NCSC–TG–005, Library No. S228,526, Version 1 July 1987.