Scalable Attack Graph for Risk Assessment

Jehyun Lee[*], Heejo Lee[†], Hoh Peter In[‡]

Division of Computer and Communication Engineering
Korea University, Seoul, Korea
{arondit[*], heejo[†], hoh_in[‡]}@korea.ac.kr

***Abstract*— The growth in the size of networks and the number of vulnerabilities is increasingly challenging to manage network security. Especially, difficult to manage are multi-step attacks which are attacks using one or more vulnerabilities as stepping stones. Attack graphs are widely used for analyzing multi-step attacks. However, since these graphs had large sizes, it was too expensive to work with. In this paper, we propose a mechanism to manage attack graphs using a divide and conquer approach. To enhance efficiency of risk analyzer working with attack graphs, we converted a large graph to multiple sub-graphs named risk units and provide the light-weighted graphs to the analyzers. As a result, when *k* order of time complexity algorithms work with an attack graph with *n* vertices, a division having *c* of overhead vertices reduces the workloads from n^k to $r(n+c)^k$. And the coefficient *r* becomes smaller geometrically from 2^{-k} depended on their division rounds. By this workload reduction, risk assessment processes which work with large size attack graphs become more scalable and resource practical.**

I. Introduction

For the past several decades, the task of protect information assets has become increasingly complex and substantially harder. The difficulties have been caused by the growth of objects to concern, nodes of networks and their vulnerabilities. As the result of protection problems, according to a CSI&FBI survey report in 2007 [1], there have been double the average annual loss in 2007 in contrast with 2006. In order to reduce the burst of damage occurring from network attacks, practitioners are adopting risk management techniques to provide network security.

In the report, almost one-fifth of respondents who experienced security incidents said they had suffered a "targeted attack" aimed at an organization. Particularly, a kind of targeted attacks called to multi-step attacks, using one or more vulnerabilities as stepping stones, are focused by security risk managers for the following reasons. At first, they must face more and more new vulnerabilities not just the large numbers of vulnerabilities that they have experience in the past. The increase numbers of new vulnerabilities have been accelerated. Statistically, in June 2008, nearly five hundred new vulnerabilities were published on CAS (National Cyber Alert System) [2]. The other reason is that there are, in risk management, much more factors what must be considered not only the security problems. They are organizational factors such as economic and political things. Against an attack scenario, the number of concerned factors increases when there are more related assets on its attack route. In these situations, security managers must decide precedence of works for resistance as fast as possible, even considering more than the final target of an attack.

Attack graphs are a well-formed method for network hardening and security analysis. Past researches [3], [4], [5], [6] using an attack graph were focused on faster graphs generation and better readability. Here, we focus on the latest usage of attack graphs, the risk management. While they are used into the risk management process in order to get various pieces of risk information and decision supports, there are various criteria, and algorithms that work with the attack graphs.

The most important factor about attack graphs is their scalability. The causes of scalability problems with these graphs are the computational complexity of the generation and the large size of the generated graph. Both of them have been caused by the increase of the network size and numbers of vulnerabilities. There were several studies achieving reasonable generation time [7]. But the large size of a graph is a quite different problem. The generated graphs were not necessarily small. Thus, researchers have suggested various approaches to manage the complexity of these graphs. However, most of them were not for risk assessment since they were focused on reducing visual complexity for user readability [8], [9]. They contributed to enhance the readability and interactivity of an attack graph. But manual graph analysis using visual representation is a different usage of attack graph with computational analysis using a graph as a data structure. Some of the other approaches [10], [11] were dependent on specific analysis algorithms. Their approaches needed consideration of the algorithms for special objects what they defined on attack graphs. It means that they may not have enough flexibility on standard risk assessment processes which have component based structures. In other words, attack graphs can have fast generation but slow analysis. To keep pace with complex and expensive risk analysis algorithms, we need to enhance scalability by managing their working data, the attack graphs.

In this paper, we propose a mechanism to make attack graphs more scalable to be used, by taking a divide and conquer approach. For the consistency of the analysis result and the flexibility of an attack graph, we have established two principles what will be explained in Section 3. As most divide and conquer algorithms do, this method shows more contribution on a more complex algorithm running on a larger graph.

II. Related work

Using attack graphs and managing risk have the same goal, enhancing network security. Researches have concentrated on using attack graphs more for technical problems than risk management. Also, the risk management standard has had economic and political problems that have received more attention than security problems.

There were lots of attack graph researches, and many of them were improved by the studies of Noel, and Jajodia at George Manson University [5], and Ou at Kansas State University [7]. In their approaches, the attack graph had various information sources and visualization ability. However, it was not in the scope of their studies to be concerned about cooperation with a risk management process especially that had a circular flow.

Noel's suggestion about complexity of attack graph [8] archived enough feasibility by aggregating common properties of vertices, but it was not suitable for automated risk assessment. Because it managed graph complexity by hiding detailed information hierarchically. We surely agree that it was helpful in enhancing user readability, but it did not contribute to reducing computing workload of analysis algorithms which require most detailed internal data of each vertex. Homer et al. [9] suggested a more fundamental way to reduce the complexity of an attack graph. They introduced several rules to remove the redundant part of attack graphs. The rules may be coordinated by the security policy of a user group. We found out that the approach was useful in removing the ring topologies from attack graphs.

To evaluate relative risk caused by inter hosts and inter sub-networks as well as the risk of each of the hosts, attack graphs are a most suitable method. However, in the risk assessment process standards, there are so many non-technical properties under the heading of 'Organizational Gap' [12]. Since that, the assessment methods are burdened with a more heavy workload than past risk analysis studies had to deal with. If attack graphs had enough scalability on practical environments, they were useful tools for attack prediction and efficient response with considering organizational properties, but most past studies were not concerned with these usages. Kotenko and Stepashkin [11] focused on a similar problem and suggested a risk assessment architecture. They did not dealt with the detail inside of each component. Also, they did not analyze concrete benefit. Their method about complexity reduction indicated that they took a graph split and aggregation method by using a dynamic programming approach. Their methodology may contribute a few additional benefits if a risk management process already utilized data reusability not limited to attack graphs. However, we think it is a good case for the assessment process so that our approach can be adopted.

III. Proposed architecture

To keep flexibility with risk analysis and quantification algorithms, we established two principles onto enhancing scalability.

- *Component independent* : "It should be archived without modifying the algorithms which use an attack graph as an input data"

- *Lossless reduction* : "It should be archived without lossy aggregation or removal of graph elements, except in explicit cases on policy"

These principles are based on the recommendation of standard risk management process [13], [12]. Security risk assessment has various points of view on a case by case basis. From the necessities of organizations, there can be a number of criteria and proper algorithms as components. For this reason, it is too hard to modify or develop every candidate algorithms for better performance. There are only a few heuristic algorithms optimized for attack graphs. For the second principle, for example, if we aggregate security conditions and relative exploits including internal data as one vertex, algorithms to find the most critical or efficient patch candidate do not work correctly.

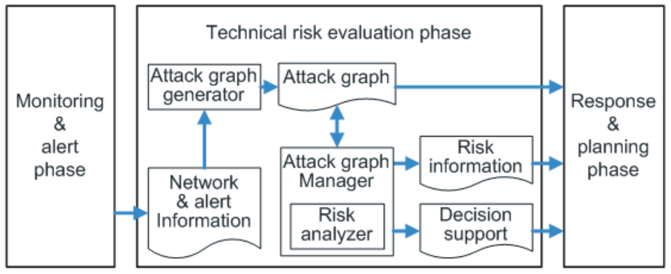

Fig. 1. The conceptual architecture of the risk assessment process using an attack graph.

Our mechanisms are fundamentally based on divide and conquer approach. In our architecture described in Fig. 1, all work is performed before and after analyzing an attack graph. Attack graph manager provides divided attack graphs to the risk analyzer. After analysis, the manager check the consistency of overlapped area between related sub attack graphs. The overlapped area is generated on division time by cloning shared vertices and edges to divide a connected graph to two sub graphs. Set of these components can be considered as a phase of a risk assessment process. The phase which uses attack graph get the environmental and political information from the previous phases and provides analysis result to the next phase.

A. The structure of the attack graphs

Attack graphs what we used in this paper have three types of vertices:square, circle, and diamond vertices. Square vertices are shown c_1, c_2, ... c_n at Fig.2. They mean represent the security conditions in a network. Most of them are caused by vulnerabilities of software installed on hosts, but some are caused by the network configuration. Circle vertices e_1, e_2, ... e_n are exploits which cause new conditions using conditions. At last, diamond vertices are penetrated conditions of a machine or a target resource. In this case, directed edges

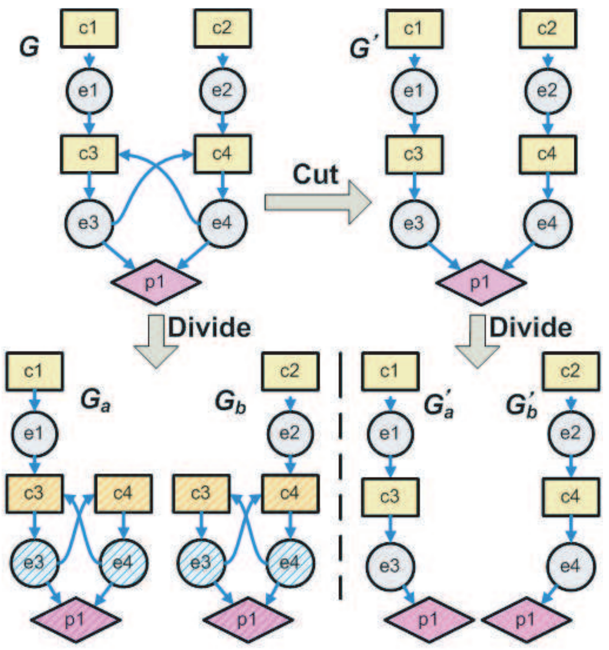

Fig. 2. An example of cut and divide operation.

```
Assume. for every initial-conditions pI, and for index of new sub-graph nSet
PROC FindSet ( Condition pI, Set nSet )
   pCur = pI
   IF (pCurMark == nSet ) THEN GOTO ENDPROC ENDIF
   CALL Mark( pCur, nSet )   // Mark its set number at the vertex
   CALL Push( pCur, nSet )   // Insert the vertex to the sub-graph vertex set
   LOOP (nIdx = 0 ; nIdx < pCurnNumberOfOut ; nIdx++)
      pCur = pCurpOut( nIdx )
      SWITCH (pCurType)
      CASE _Condition:
         CALL FindSet( pCur, nSet ) ENDCASE
      CASE _Exploit:
         IF (pCurnNumberOfIn > 1) THEN
         LOOP (nIdx2 = 0; nIdx2 < pCurnNumberOfIn ; nIdx2++)
            pCur = pCurpIn( nIdx2 )
            IF ( pCurMark ≠ nSet ) THEN
            CALL FindSet(pCur, nSet) ENDIF
         ENDLOOP
         CALL FindSet(pCur, nSet) ENDIF ENDCASE
      CASE _Penetrate:
         IF ( Rule( StopAtPenetrate )∨Rule( Any rules ))
         THEN
            CALL Mark(pCur, nSet )
            CALL Push(pCur, nSet )
            GOTO ENDPROC
         ELSE CALL FindSet(pCur, nSet ) ENDIF ENDCASE
      ENDSWITCH
   ENDLOOP
ENDPROC FindSet
```

Fig. 3. C like pseudo code for division procedure, *'FindSet'*

started from the origin conditions point on an exploit vertex that is using the conditions. Also, directed edges starting from an exploit vertex, points to other conditions that become active by the exploit. In other words, a security condition is caused by performing exploits that points to the condition vertex, and an exploit needs the conditions that point to the exploit vertex. However, there is a difference between exploit pointing edges and condition pointing edges. The all edges pointing to an exploit vertex mean that the origin vertices are necessary conditions of the pointed exploit; however, the edges pointing to a condition vertex mean that the origin vertices are sufficient conditions of the pointed condition.

B. Division process - Cut and divide

Cut and divide operations make a connected attack graph into two isolated sub graphs. By repeating this operation multiple times, an attack graph becomes divided into multiple sub graphs.

1) Cut operation: The cut operation removes a group of edges and vertices that satisfy the cutting rules. Cutting rules can be determined to remove attack paths that have extremely rare feasibility or risk factor from the graph. Homer's rules [9] are good example of cutting rule for making an attack graph more obvious. Fig.2 shows the reduction of ring structure by cutting two edges from the original graph G. As a result, the cut graph G' has more independency between the vertices.

2) Divide operation: The divide operation makes a connected attack graph to two sub-graphs. Conceptually, because it is for finding a set of related vertices on a depth first attack path, if there are three dividable sub-graphs, the graph needs to perform the divide operation twice. As a result of the above cut operation, G' has less size of intersection vertex set between the attack paths starting from c_1, and c_2 than the original graph G. Since the vertices of the intersection set becomes to overlapped vertices of the divided sub-graphs, less intersection size means less division overhead. In Fig.2, the sub-graph G'_a and G'_b have smaller overlapped area and the entire graph than sub-graph G_a and G_b. The graph G is converted to two sub-graphs having 14 vertices, but the cut graph G' is converted to two sub-graphs having 10 vertices.

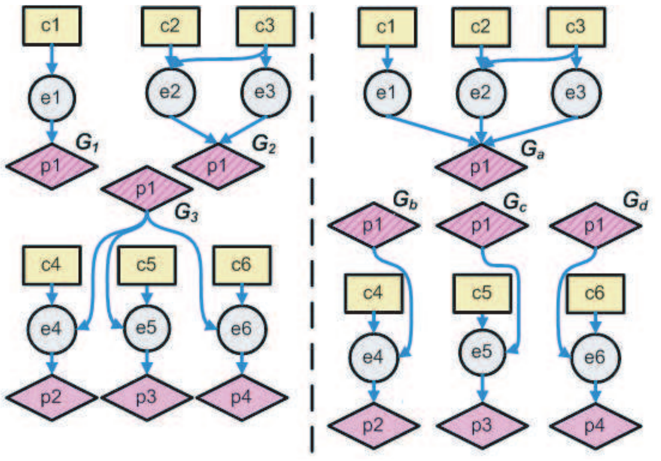

Fig. 4. An example of top-down(left) and bottom-up(right) directed division.

The divide operation has two kinds of dividing forms that depend on the assessment direction of the risk. The assessment direction is determined from the kind of risk that a user wants to obtain from the attack graph. When a user wants to assess criticality, countermeasure, consequence, or the maximum effect area of a security condition, in other words, an user want to know the worth of a patch, or a security product. The assessment, then, is performed with a top-down direction. On the contrary, the bottom-up direction may be used to obtain the security risk that a specific machine has. Fig.4 shows two different results of each of the dividing forms. In the easy sense, the top-down sub-graphs G_1, G_2, and G_3 are started from one or set of conditions having an indispensable relation and end with a set of penetrate (goal) conditions. Opposite to this case, the bottom-up sub attack graphs G_a, G_b, G_c and G_d are started from a set of initial conditions, and end to a penetrate condition. Fig.3 is the pseudo code for the division procedure. The procedure finds a set of vertices from the original attack graph to make a sub-graph. We named these divided sub attack graphs to risk units. A risk unit is a working unit for risk assessment and quantifying. This procedure finds out connected nodes and inserts them into the node set of the start node. To find related condition nodes which share exploit nodes, the procedure is recursively called and find branches of the condition or exploit nodes. It searches the graph with depth first policy and stops until there is a kind of penetrate condition nodes which satisfy pre-defined stop rules or any other grouping rules. To avoid duplicated search in loop connections we took the bit marking method in this algorithm, but it can be implemented by any other method.

The one of most significant problem of dividing an attack graph is the ring structure in the graph. In the ring structured graph, our algorithm outputs similar or almost identical sub-graph. However, it just means an overlapped area that cannot be divided anymore. Therefore, if ring structured vertices take reasonable portion of the sub graphs, under the overhead practicality limit, there still are workload benefits from the divide operation. We analyze the practical range of the overhead in the evaluation section of this paper.

C. *The update and merge process*

When there is a new security condition or a newly discovered exploit, an attack graph must be updated. If the updated data has an effect on the relationship among sub attack graphs, the sub attack graphs are restored by merge operation. This special case is a breakout of when a new exploit needs plural conditions that are in different sub graphs, or when a new condition causes an exploit which makes a new path between two conditions in different sub graphs. On the other hand, if an update causes a vertex removal in a sub graph, the sub-graph is going to be able to take a chance of dividing if it has sufficient conditions. Fig.5 is an example of the case of a merge operation caused by a new security condition.

D. *Consistency process*

After any changes in an attack graph, it must be checked for consistency among the related sub-graphs, especially, if the plural sub attack graphs have a set of overlapped vertices. However, fortunately, it does not mean that the risk

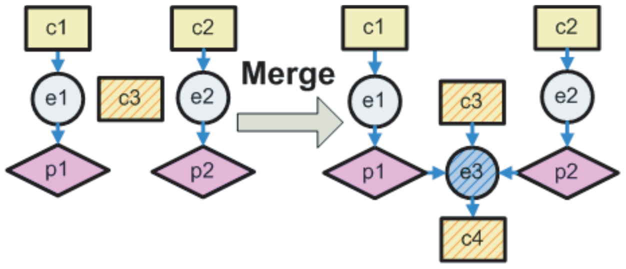

Fig. 5. An example of merge operation.

of vertices in the set should be re-assessed in every case. If there were no topological changes on the overlapped vertices, keeping consistency is just update of numerical values which is already quantified on the sub-graphs.

In some cases, there can be a dependency problem by change of topological position of a vertex. Risk and worth of a vertex that several sub-graphs share might be depended on the analysis result of a graph sharing the vertex. In this case, an analysis process should wait for the results of related graphs. However, for independency of inter risk units, the overlapped area of sub graphs can be expanded over a vertex. By this expansion, a condition which is distinctly depended on other conditions can shift a problem position to another vertex. It uses that there are two kinds of vertices. Some vertices have absolute worth and risk, and the other vertices have relative worth and risk. These vertices which have absolute properties do not cause the dependency problem wherever they are.

IV. EVALUATION

In this paper, we addressed the workload problem of risk assessment on attack graphs. Reduction of entire workload on risk assessment means less cost and faster decision. In other words, users can get benefits on the trade-off between versatile, accurate, but expensive analysis algorithms and non-optimal, heuristic but low cost algorithms.

In the suggested mechanism, workload reduction is accomplished by the external divide-and-conquer process. The process provides more light-weighted attack graphs to an analysis component. In this section, we show the quantified benefit of our suggestion. We also present the ranges of allowed overhead, and they are high enough to practical degree.

A. *Analysis*

Our architecture achieves more benefits with respect to time consumption with more balanced and divided graphs. In perfectly balanced dividing and a no overhead ideal situation, the architecture shows geometrically increasing reduction benefits, 2^{-k} times that of the original workload, where the vertex size of an attack graph is *n* and the target algorithm has n^k of time complexity. With the overlapped vertices appearing at the dividing operation as the overhead, the benefits are decreased depending on the portion of the overlapped area, but there

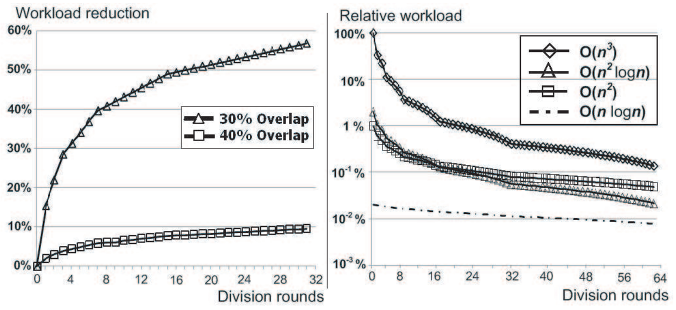

Fig. 6. Workload reduction and relative workloads comparison by graph division.

TABLE I
COMPARISON BETWEEN DIVIDED AND ORIGINAL CASES

	n^3	$n^2log(n)$	n^2	$nlog(n)$
Original	1954.31 sec	78.12 sec	19.53 sec	0.78 sec
Divided	78.12 sec	4.87 sec	2.90 sec	0.68 sec

are still increasing benefits by more division in Fig. 6 (left).

As shown at Fig.6 (right), since this approach involves managing the complexity of input data proportionally, it shows a better result at the target algorithms which have a higher time complexity. The allowed limitation of overhead, the size of the overlapped area, is determined by the time complexity of the target algorithm. Assuming that the size of each of the risk units are nearly the same, by many divisions, the divided graphs having a graph size less than $2^{(1-1/k)}$ times that of the previous attack graph has benefits for processing workload reduction. In other words, on division time, divide operations that have expected overlapped area larger than $1 - 2^{(1-1/k)}$ times that of the target graph are not allowed. According to our analysis, n^3, $n^2log(n)$, n^2, and $nlog(n)$ complexity algorithms have toughly 50%, 45%, 40%, and 10% of overhead limitations in each of the cases.

B. Experiment

To evaluate how the reduced workloads contribute to actual computing time, we performed the architecture using dummy algorithms. It was performed on Intel Pentium4 3.0Ghz CPU, 1GB RAM, and Windows XP service-pack 3 OS environment. We generated randomly, but considering vertex types, a connected graph with 100 vertices, and performed algorithms which have matrix calculations, file I/O, and string matching with internal data assigned to vertices. In divided cases, we performed top-down direction division algorithm for 32 rounds with a 30% overhead limitation. The computation times include time overhead for division computations near 50ms for 32 running rounds. Table I shows the comparison results of time consumptions with different complexity algorithms.

V. CONCLUSION

We proposed a simple but clear approach to enhance scalability of risk assessment on an attack graph considering the practical limitations of the risk management process. Our external divide and conquer approach does not require adaptation of analysis methods and is not on exclusive relation with other approaches for attack graph complexity management. By cloning a set of vertices which are necessary to multiple sub-graphs, it becomes to be available to divide a connected graph without loss of adjacency and information. The approach directly contributes to reducing the workload of analyzers in a risk assessment process by providing a light-weighted input object, named a risk unit. It is a fully logical aggregation and automatically generated by some simple sub-graph separation rules. We have shown how to divide a connected attack graph by cloning the overlapped vertices among risk units. We have evaluate expected benefits by workload reduction in various cases and range of practicality by analyzing the size limitation of overlapped area. The results of analysis and experiments show that this method contribute to reducing heavy workload of the complex analysis algorithms with very practical overhead tolerance.

ACKNOWLEDGMENT

This work was supported in part by the ITRC program of the Korea Ministry of Knowledge Economy, and the Basic Research Program of the Korea Science & Engineering Foundation.

REFERENCES

[1] “CSI&FBI Report 2007,” http://www.gocsi.com/.

[2] “National Cyber Alert System,” http://www.us-cert.gov/cas/.

[3] K. Ingols, R. Lippmann, and K. Piwowarski, “Practical attack graph generation for network defense,” *Proc. of the 22nd Annual Conf. on Computer Security Applications*, pp. 121–130, Dec. 2006.

[4] R. Lippmann and K. W. Ingols, “An annotated review of past papaers on atack graphs.” MIT Lincoln Laboratory, Tech. Rep., March 2005.

[5] S. Noel, S. Jajodia, B. O’Berry, and M. Jacobs, “Efficient minimum-cost network hardening via exploit dependency graphs,” in *Proc. of the 19th Annual Conf. on Computer Security Applications*. IEEE Computer Society, 2003, p. 86.

[6] L. Wang, A. Singhal, and S. Jajodia, “Toward measuring network security using attack graphs,” in *Proc. of ACM workshop on Quality of Protection*. ACM, 2007, pp. 49–54.

[7] X. Ou, W. F. Boyer, and M. A. McQueen, “A scalable approach to attack graph generation,” in *Proc. of the 13th ACM Conf. on Computer and communications security*. ACM, 2006, pp. 336–345.

[8] S. Noel and S. Jajodia, “Managing attack graph complexity through visual hierarchical aggregation,” in *Proc. of the 2004 ACM workshop on Visualization and data mining for computer security*. ACM, 2004, pp. 109–118.

[9] J. Homer, A. Varikuti, X. Ou, and M. A. McQueen, “Improving attack graph visualization through data reduction and attack grouping,” in *Proc. of the 5th Int’l Workshop on Visualization for Cyber Security*, Sep. 2008.

[10] R. Dantu, K. Loper, and P. Kolan, “Risk management using behavior based attack graphs,” in *Proc. of the Int’l Conf. on Information Technology: Coding and Computing*, vol. 2. IEEE Computer Society, 2004, p. 445.

[11] I. Kotenko and M. Stepashkin, “Attack graph based evaluation of network security,” in *Proc. of the 10th IFIP Int’l Conf. on Communications and Multimedia Security*, ser. LNCS, vol. 4237. Springer, 2006, pp. 216–227.

[12] CMU SEI, “OCTAVE,” http://www.cert.org/octave/.

[13] G. Stoneburner, A. Goguen, and A. Feringa, “Risk management guide for information technology systems,” *NIST Special Publication*, no. SP 800-30, July 2002.