Using Vulnerability Trees for Decision Making in Threat Assessment

S Vidalis and A Jones

School of Computing Technical Report CS-03-2

Issued: June 2003

School of Computing, University of Glamorgan, Pontypridd, CF37 1DL, Wales, UK. www.comp.glam.ac.uk

Using Vulnerability Trees for Decision Making in Threat Assessment

Stilianos Vidalis and Andy Jones

Abstract

During the development of the TAME threat assessment methodology under framework 5 IST-2000–29601, the problem of analysing and examining vulnerabilities was identified. The methodology was developed for the assessment and analysis of threat and vulnerabilities within the context of a security risk management. TAME was developed with the needs of electronic payment systems in mind and it consists of four stages: a) Scope of Assessment, b) Scenario Construction & Modeling, c) Threat Agent & Vulnerability Analysis, and d) Stakeholder Evaluation. The requirement for this paper was identified in the "Vulnerability Complexity Analysis", which is part of the "Threat Agent & Vulnerability Analysis" stage of the methodology. Identifying vulnerabilities in isolation is not sufficient for analysing, assessing and securing a network. There is a need to identify vulnerability chains and to model their relationships.

Through use of the TAME methodology the user should be capable of producing a matrix identifying the most significant of the vulnerabilities for each asset involved with the system under analysis. From this there is a requirement to find out how easy (or difficult) it is for a vulnerability to be exploited by a threat agent. Does a threat agent need to exploit another vulnerability in order to achieve his goal or is it enough that one vulnerability is used? What are the different attack paths that the agent might follow in order to achieve his/her goals? How long will it take for an agent with a given set of capabilities to exploit an asset vulnerability? Will he be able to manifest a threat in that time window? How complex is for the different types of threat agents to exploit system vulnerabilities and how "worried" should the information security officers be? Vulnerability trees can provide the answers to the above questions by helping the users of the TAME methodology to identify key vulnerabilities that are common to more than one assets of the system and help them to counter them in a cost effective manner.

The phrase that can best describe the problem that is being addressed in this paper comes from Keeney and Raiffa: "In an uncertain world the responsible decision maker must balance judgments about uncertainties with his or her preferences for possible consequences or outcomes." Information security (IS) as a concept is chaotic. The four goals of IS: confidentiality, integrity, availability and non-repudiation, are in tension with each other in such a way that a shift in favor of one of these elements may make an attack on another of them more likely. As a result, a decision maker has to make assumptions and judge situations according to his/her experience. In the modern electronic world that we live in, there is a need for a more formal technique to help with the above process. In this paper we will present such a technique based on the Object Oriented principles, the utility theory, FTAs and other methodologies. The technique is part of a process that is aiming in minimising and controlling the threats against e-Business and e-Commerce.

Introduction

During the development of the TAME threat assessment methodology (Vidalis '01), the problem of analysing and examining vulnerabilities (Blyth '01) was identified. The concise oxford dictionary (Sykes '81), defines the term Vulnerability to mean: "is susceptible to damage". Vulnerability has been defined as follows:

- A point where a system is susceptible to attack (Kabay '96).
- A weakness in the security system that might be exploited to cause harm or loss (Pfleeger '97).
- Some weakness of a system that could allow security to be violated (A.J.C.Blyth '01).

However, for the purpose of a threat assessment we require a definition that is more general to information security and encompasses, information technology, communication systems, and business processes. Therefore we will define vulnerability as a measure of the exploitability of a weakness. According to (Pfleeger '97), (A.J.C.Blyth '01), (Summers '77), (Scambray '01), (M.Smith '93), (Forte '00), , there are six types of vulnerabilities that can exist in any system, and these are: Physical, Natural, Hardware/Software, Media, Communication, and Human. We need a process that will be able to analyze all of the above different types.

The problem examined in this paper was identified in the “Vulnerability Complexity Analysis” step, which is part of the “Threat Agent & Vulnerability Analysis” stage of the TAME methodology (Vidalis '01). For reference purposes the different stages of the TAME methodology are:

- *Scope of Assessment*: Business Analysis, Stakeholder Identification, System Boundaries Identification, and Threat Agent Identification & Selection
- *Scenario Construction & Modeling*: Scenario Generation, System Modeling, Asset Identification
- *Threat Agent & Vulnerability Analysis*: Threat Agent Preference Structuring, Threat Agent Capabilities, Vulnerability Type Identification & Selection, and Vulnerability Complexity Analysis
- *Evaluation*: Stakeholder Evaluation, Scenario Selection & Conflict Resolution, Threat Impact Analysis, Threat Statement Generation and Transfer

Through use of the TAME methodology the user should be capable of producing a matrix identifying the most significant of the vulnerabilities for each asset involved with the system under analysis. From this there is a requirement to find out how easy or difficult it would be for a vulnerability to be exploited by a threat agent (Stalling '00), (Carroll '96), (Ammann '02). Does a threat agent need to exploit another vulnerability in order to achieve his goal, or is it enough that one vulnerability is used? What are the different attack paths (Moore '01) that the agent might follow in order to achieve his/her goals? How long will it take for an agent with a given set of capabilities (Vidalis '01), (A.J.C.Blyth '01), (Barber '01), (Hoath '98), (Rees '96), to exploit a vulnerability, and will he be able to manifest a threat in that time window? How complex is for the different types of threat agents to exploit system vulnerabilities and how concerned should the information security officers be? Vulnerability trees can provide the answers to the above questions by helping the users of the TAME methodology to identify key vulnerabilities that are common to more than one assets of the system and help them to counter them in a cost effective manner (Summers '77).

State of the Art

There are quite a few tools that can be used for analyzing systems and identifying vulnerabilities. Some of the tools are: COPS (COPS '02), NESSUS , SystemScanner (SystemScanner '02), Retina, NetRecon, Whisker, and CyberCop. It is recognized in (Ammann '02) that just identifying individual vulnerabilities is not sufficient and adequate in today’s electronic era of cyber-crime (Bequai '01).

There are quite a few approaches when it comes to modeling vulnerabilities in order to perform some sort of analysis in a computing system. The safety critical systems field examines the hazard analysis process. Vulnerabilities can be perceived as being hazards for a computer system. The different techniques that analyse hazards include: checklists, fault tree analysis, event tree analysis, and cause-consequence analysis.

Checklists are static and cannot demonstrate the relationships between the vulnerabilities. Furthermore, they do not examine the how and the why two vulnerabilities are related to each other. Fault trees are just chronological orderings of events over time and are not adequate to visualize and model the different types of vulnerability relationships. Each level of the fault tree merely shows the same thing in more detail. Event tree analysis is a Boolean approach to examine vulnerabilities and failures. Most of the vulnerability types of a computing system though cannot be expressed with Boolean values. The technique work very well for hardware vulnerabilities, but according to (Nuemann '95) there are six other vulnerability types, that cannot be addressed effectively. Cause-Consequence Analysis (CCA) is a top-down or backward technique that can determine the causes of an event. It can model both time dependencies and casual relationships among the events. The negative side of CCAs is the size of the diagrams, their complexity and the fact that they cannot accept data from other diagrams.

Another technique is the use of history attack data for producing patterns and attack trees. This technique is trying to predict the path that the threat agent will follow by analyzing the exploits that might be used. Each path through an attack tree represents a unique attack on the enterprise. The problem with attack trees is that they cannot analyse big systems or large—size networks (Ammann '02) mainly due to their complexity. A different number of exploits might be used for attacking more than one vulnerabilities, and the same exploits can be used for attacking different vulnerabilities. Producing attack trees using exploits as nodes is not efficient for a system that changes constantly.

Vulnerability Trees Definition

Vulnerability trees are hierarchy trees (Keeney '93) constructed as a result of the relationship between one vulnerability and other vulnerabilities and/or steps that a threat agent (A.J.C.Blyth '01), (Carroll '96), (Rees '96), (Hinde '01), (Icove '95) has to carry out in order to reach the top of the tree. The top of the tree is known as the top vulnerability or the parent vulnerability and we will symbolise it with a capital 'V'. There are a large number of ways that such a top vulnerability can be exploited. Each of these ways will constitute a branch of the tree. The branches will be constructed by child vulnerabilities. Consequently the child vulnerabilities can be exploited by steps that the threat agent will have to perform in order to get to the parent. We will symbolise the child vulnerabilities with the lower case 'v' and the steps with the lower case 's'. Each vulnerability will have to be broken down in a similar way. Normally this will end up in more than one levels of decomposition. When the point is reached where the branches contain only steps, and no child vulnerabilities, then we know that we have reach the lowest level of decomposition. We will call that level the "step-only" level

For example let us consider the scenario of a main broker server of a micro-payment system (MPS) (Vidalis '01), (Manasse '95), (W3C '99), (O'Mahony '97). There is the physical vulnerability of a human threat agent to walk in the server room and steal the server. For doing that though, he would exploit a vulnerability related to the alarm system of the room, and with regard to the building, a vulnerability related to the security guards, and a vulnerability related to the high security doors which are installed in the server room. Hence, from what we have seen up to know, the tree should look like the one presented in figure 1.

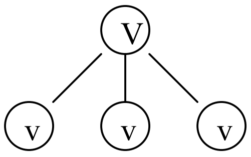


Figure 1 – Level 0 Tree example

For the graphical representation of the vulnerability trees we will use the notation presented in appendix A. The notation was inspired from the FTA notation taken from (Storey '96) and (Leveson '95). Taken under consideration the notation, the updated tree is presented in figure 2.

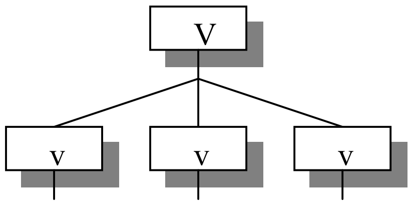


Figure 2 – Updated Level 0 Tree example

If we think of the vulnerability as being an object, like all objects, it has some attributes. According to Coad (Coad '91) attribute is any property, quality, or characteristic that can be ascribed to a person or thing.

Attributes describe values to be exclusively manipulated by the services of the object. Based on (Coad '91) the service is defined as an activity carried out to provide people with the use of something. The services of the vulnerability object fall outside the scope of this paper, hence they will not be analysed. The attributes of the vulnerability object that have been identified in this initial stage are the following:

- Vulnerability ID: Unique vulnerability identifier
- Name: Vulnerability name
- Type: Indicates if node is vulnerability or step.
- Category: There are six categories of vulnerabilities that can exist in any system, and these are: Physical, Natural, Hardware/Software, Media, Communication, and Human. Each type will have to be approached in a different way.
- CV (complexity value): The complexity value is defined as:

> *The* ***complexity value*** *'CV' of a vulnerability X is defined as the smaller number of vulnerabilities/steps that a threat agent has to exploit/utilize in order to achieve his objective.*

Let us consider the following example:

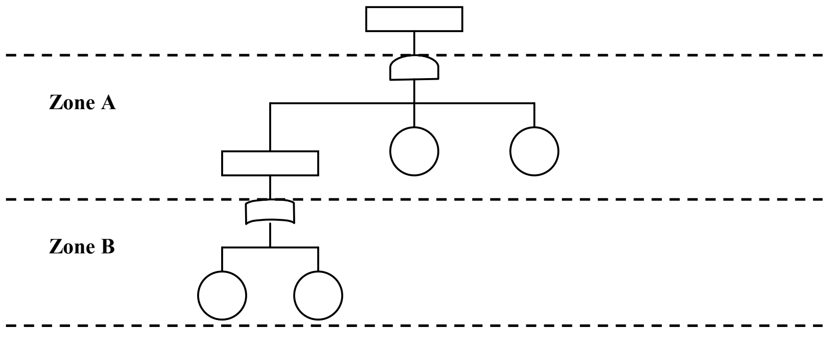


Figure 3 – FTA Example 1

Based on the notation presented in appendix A, the top vulnerability requires all of the vulnerabilities/steps in zone A and any one from zone B. Hence, according to the above definition, the complexity of that vulnerability is 3.

The above identifies the way with the least obstructions for an agent to achieve his goal. Each vulnerability has an educational complexity (EC) associated with it though; hence the first definition might not always be adequate. There is a need for showing the 'EC' of each step. The 'EC' is closely related to the educational level of the agent, hence different agents will follow different paths/techniques to achieve their goals. An additional feature is the Time to Exploit (TTE). The 'TTE', according to the asset will vary in importance. According to the agent capabilities, the 'TTE' attribute might drastically affect the agent's path in exploiting the goal vulnerability. The TTE of any vulnerability/step will be unique to each type of threat agent.

- EC (educational complexity): Everyone can open a door and step into the server room, if the door is unlocked, but not everyone can use ‘nmap’ to perform a port mapping. The educational complexity is defined as:

  *The **educational complexity** ‘EC’ of a vulnerability X is defined as the educational qualifications that a threat agent has to acquire in order to exploit that vulnerability.*

  Educational complexity is related to the educational level of the threat agent. Consequently, the educational level of the agent is related to the capabilities and the resources that they already possess or can acquire in the future. Hacking is a “hands-on” activity. Just by knowing the theory does not automatically put an agent in a certain category. That agent will have to exercise and apply the theory for acquiring the knowledge. Table 1 presents the different educational levels.

Educational Complexity	Qualifications	Alias
1	None - No computing education	
2	Primary education – Familiar with term “computer”	
3	Secondary education – Computer user	
4	GCSE – User, Basic knowledge of operating systems (OS) & Internet	
5	A level – User, Basic knowledge of OS & Internet, Basic programming skills	Script kiddy
6	University Level 1 – Power user, Knowledge of OS & Internet, Basic programming skills, Basic networking	Amateur
7	University Level 2 – Power user, Medium knowledge of OS, Internet, programming and networking, Familiar with Linux	Amateur
8	University Level 3 / BSc – Administrator, Advanced internet, programming, networking & OS, Basic scripting, Basic Linux	Amateur
9	MSc – Root, Expert internet, programming, networking, scripting, Medium Linux	Hacker
10	Post MSc – Root, Expert internet, programming, networking, scripting, Advanced Linux, active hacking	Expert Hacker
A	PhD – all the above, known entity to hacking community	Professional Hacker
B	Post PhD – all the above, criminal record	Computer Criminal
C		Supreme Being

Table 1 – Educational Complexity Table

The table is not meant to tie the educational complexity to the academia and to a certain academic qualification. Most hackers do not have academic qualifications on “hacking” or on computers in general. The qualifications are only there for reference purposes, and for understanding the level of expertise that can be expected for an agent that falls under a category.

- TTE (time to exploit): The TTE is defined as the time required for a threat agent type to exploit a certain vulnerability. Because the exploitation of a vulnerability has a number of steps, TTE is the sum of the time needed to perform each of the required steps. Because different threat agent types have different capabilities, they have different TTEs. The golden rule is that as long as our sensors are able to identify the attack inside the time window presented by the TTE, and still have enough time to deploy the necessary countermeasures, then the asset is secure.
- FP (family position): The level of the node: 0 indicates parent or top vulnerability and any other number indicates child and how “far” is the child form the parent.
- Head: Identifier of the head vulnerability (needed for producing automated tool in the future)
- Asset ID: Unique asset identifier, which link the asset with the vulnerability.
- Tree ID: Unique tree identifier, which link a tree with the vulnerability. There can be more than one instances of the same vulnerability as different assets may have the same vulnerabilities.
- Description: Additional details to describe the vulnerability

Let us consider the very simple example of walking into the unique Broker server of a micro-payment system (MPS). There is the physical vulnerability of a human threat agent walking in the server room and stealing it. In order to do that though, he must exploit a vulnerability related to the alarm system of the room, and of the building, a vulnerability related to the security guards, and a vulnerability related to the high security doors which are installed in the server room. The vulnerability tree of the above is presented in figure 4.

According to the definition, the complexity of the physical vulnerability of the server being stolen is 3. For the vulnerability to be exploited three events must take place: bypass the alarm system, bypass the secure doors, and bypass the security guards. For the alarm system to be exploited, one of the following must happen: power failure, software failure or hardware failure. This is one point because only one of the previous must happen in order to have a critical situation. For the bypassing of the secure doors one of the following must happen: find a structural vulnerability or find the security code. This is one more point. For getting the security code one of the following things must happen: crack the software that allocates the code, or get the code from an employee. Having in mind that a chain is as strong as the weakest link, we say that a threat agent hasn’t got to do anything from the last level of the diagram because he can succeed in his goal more easily and faster by going through the second level. After adding all the points together, we come to a complexity result of 3.

Just by having the vulnerability complexity is not adequate though. We need to calculate its educational complexity as well in order to identify critical paths for the different types of threat agents. The ‘EC’ of each vulnerability/step is displayed on the left and the TTE on the right. The time window in which a manifestation of such a threat would have an impact in the business is not applicable. If the unique Broker server of the MPS is stolen, then the business will be affected and the impact will be catastrophic[1]. The 'EC' of the top vulnerability is 3.

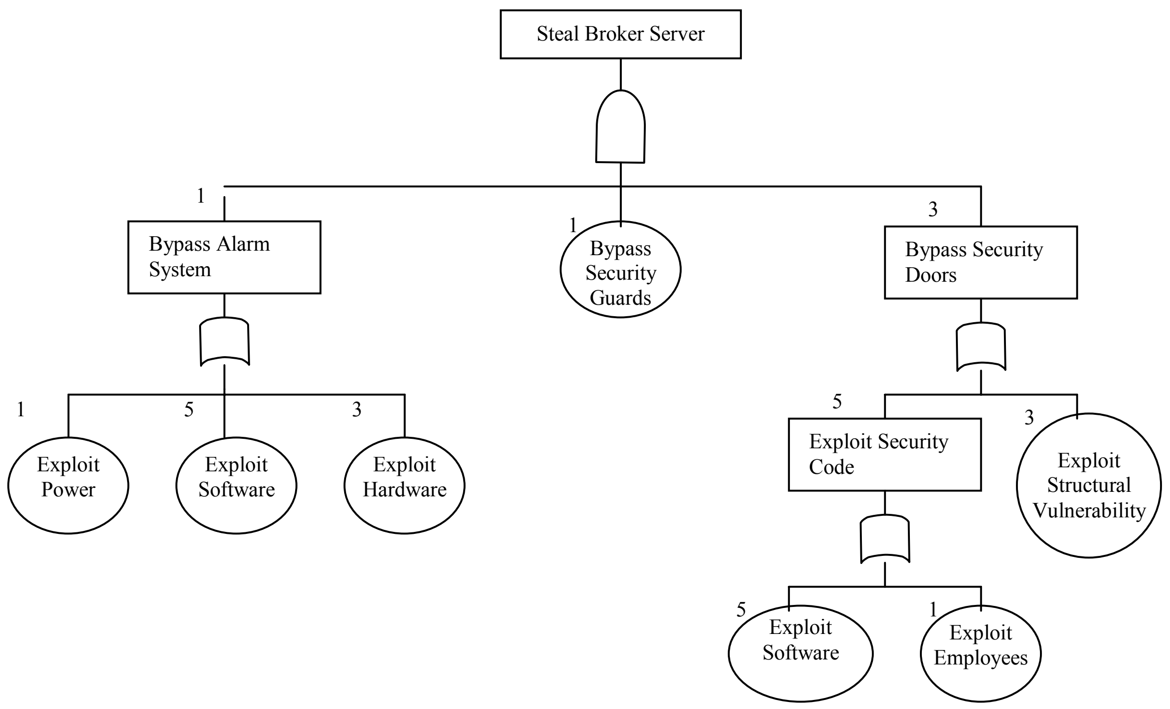


Figure 4 – Tree Example

Given a threat agent with a capability of 5 in the above example, he might not always decide to go for the "quickest way. The exploitation of a vulnerability is a combination of capabilities, motivations and opportunities. Depending on his motivations the threat agent might decide not to "reveal" himself as an agent of such a capability (educational level) and go for another more trivial way. Furthermore, if a threat agent, after completing the information gathering stage, realizes that certain countermeasures are deployed, he might decide to alter his course of action. Hence the presented opportunity will affect the course of a vulnerability exploitation.

Vulnerability Trees Construction

The purpose of the vulnerability trees is to help the information security officer to understand and analyse the different possible attack scenarios that a threat agent will follow in order to exploit the vulnerabilities of a system. As a result the officer will be able to identify the easiest and/or most possible attack scenario and deploy countermeasures for effectively defending his/her system asset while effectively allocating

[1] A classification of threat impacts taken from (Vidalis '01). Vidalis, S. (2001). TAMMPS: a threat assessment model for micro-payment systems. School of Computing. Pontypridd, University of Glmaorgan**:** 1-152 is the following:

- Minor: minor loss of a business asset, no change in business order
- Moderate: business disruption, moderate changes in way of conducting business
- Major: out of business unless countermeasures are deployed immediately
- Catastrophic: out of business from the moment that the threat was realized

defensive resources in a cost effective manner. That involves some decision analysis. According to Keeney (Keeney '93) a basic framework for decision analysis has at least five steps.

1. *Pre-Analysis*

From the modelling stage of the TAME methodology(Vidalis '01), and from the vulnerability identification step of that methodology, a vulnerability is identified. That vulnerability will be the top vulnerability V of our tree. All the assets related to that vulnerability are identified and selected from the asset table that is constructed during the asset identification step. These assets are associated with a number of vulnerabilities. All these vulnerabilities are identified and selected from the vulnerability table that was constructed during the vulnerability identification step. As it can be seen the maker has very little to do as all the identification has be done in the past as part of another step of the TAME methodology. Furthermore, automated tools can be developed for the selection of the necessary data from all the tables mentioned above.

2. *Structural Analysis*

In this step the trees are constructed by deciding on the branches and on the nodes. We are working from the top vulnerability downwards, until we reach the "step-only" level. It will be extremely unlikely to have only a level 0 vulnerability tree on its own. The level 0 tree will not have any steps as nodes. The number of levels depends on the decomposition that the maker is prepared to do and the level of detail he wants to achieve. The scope of the assessment might introduce restrictions on what types of vulnerabilities the maker should be concerned with, hence he might willingly decide not to analyse a child vulnerability. In the example presented in figure 4 the decision maker can further decompose the "Exploit Employees" node in a level 1 vulnerability tree. In that case it is expected that the attributes of the vulnerabilities will change, and that the level of detail will affect the complexity value. The maker then makes a distinction between nodes that are under his control (decision nodes), and those that aren't (chance nodes).

For the decision nodes it will be simple and straightforward to come up with a set of countermeasures that will have to be deployed. Because each vulnerability is associated with an asset, and each asset is associated with a stakeholder, assigning responsibilities to stakeholders will be a straightforward task.

For the identified chance nodes, SLAs should be signed. Vulnerability trees will help in defining word by word the SLAs as they will provide all parties involved in the agreement with all the potential outcomes of the disaster.

3. *Node Analysis*

The attributes of the vulnerabilities are being populated in this step. These attributes are the following: ID, name, type, category, complexity value, educational complexity, educational complexity, time to exploit, family position, head, asset ID, tree ID, and description. T.T.E., V.C. and E.C. are being examined and assigned values depending on the selected type of threat agent. Information about the threat agents can be found under the threat agent identification step of the TAME methodology.

4. Value Analysis

In this step the critical paths to the top vulnerability are being identified depending on the selected type of threat agent. The result of the critical path will give us the complexity value of the top vulnerability. The outcome will be passed to the impact analysis step of the TAME methodology for examination. Each top vulnerability will now have a complexity value and an educational value associated with it, hence it will be straightforward on selecting the threat agents that will have the capability of exploiting that vulnerability.

5. Optimisation Analysis

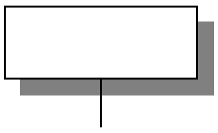
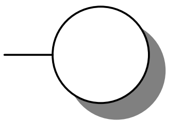
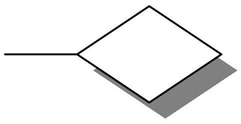
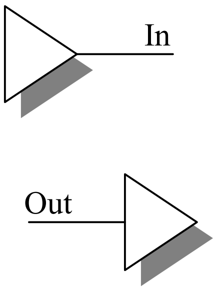
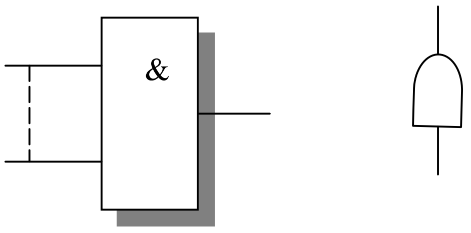
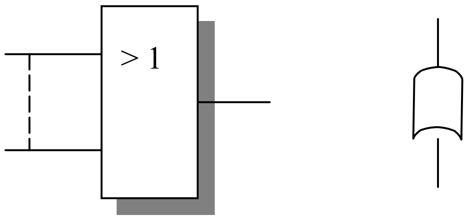
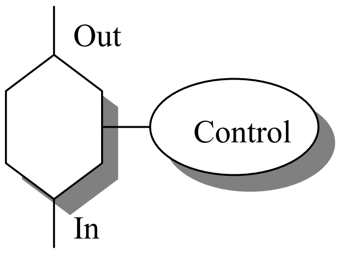
After the completion of the tree, vulnerabilities that occur more than once in a tree, or in different trees, are being identified and selected for minimising the countermeasure costs. The identification of duplicate vulnerabilities can be conducted out either by observation or by an automated tool that will examine all the vulnerability trees and decide on the vulnerabilities that will need to be considered. The more instances that a vulnerability has, the more cost efficient the countermeasures will be. Furthermore, the lower the position of a vulnerability in the family, the easiest and more efficient the countermeasure will be. We have to keep in mind that vulnerability trees have the result at the top and the initiating events at the bottom.

Conclusion

Vulnerability trees examine vulnerabilities and use their relationships in order to construct chains that will identify different scenarios of a threat agent exploiting a vulnerability. Common vulnerabilities in the chains can be identified and countered in order to secure the system in a cost effective manner. Furthermore, critical paths will differentiate the vulnerabilities that must be countered, from those that will have to be countered some time in the future.

Vulnerability trees presented the information security officers who wish to use the TAME methodology with a technique that integrates with the other steps of the methodology, taking input from more than one of its different steps. It can also be used for modeling the system under the “Scenario Construction & Modeling” stage, hence adding to the efficiency of the methodology. Having defined the vulnerability trees and a technique for constructing them, it is now possible to produce an automated tool that will be able to construct the trees and provide the user with feedback on common vulnerabilities, critical paths and assets that are greatly exposed to the threat agents. Such a tool will be capable of being integrated and used in conjunction with intrusion detection systems for better results. The Information Security Team of the University of Glamorgan is currently developing such a tool.

Appendix A – Vulnerability Trees Notation

Symbol	Meaning
	Event denoting vulnerability
	Basic event, denoting a step that a threat agent has to perform
	Event not fully traced to its source. It is taken as an input but its causes may be unknown
	The triangle symbol is used to link trees. The ‘in’ symbol indicates an input from another tree. The ‘out’ symbol appears in place of the ‘top event’ and indicates that this point forms the input to another tree.
	The output event occurs if ALL the input occur.
	The output event occurs if ANY of the inputs occur, either alone or in combination.
	The control condition determines whether the input event appears on the output.

Source: (Storey '96), (Leveson '95)

References

(Storey '96). Storey, N. (1996). Safety Critical Computer Systems, Addison Wesley.
(Leveson '95). Leveson, N. G. (1995). Safeware: system safety & computers, Addison Wesley.
(Vidalis '01). Vidalis, S. (2001). TAMMPS: a threat assessment model for micro-payment systems. School of Computing. Pontypridd, University of Glmaorgan**:** 1-152
(Blyth '01). Blyth, A. J. C. and L. Kovacich (2001). Information Assurance: Computer Communications & Networks, Springer-Verley.
(Sykes '81). Sykes, J. B. (1981). The Concise Oxford Dictionary, Clarendon Press.
(Kabay '96). Kabay, M. E. (1996). Enterprise Security: Protecting Information Assets, McGraw-Hill.
(Pfleeger '97). Pfleeger, C. P. (1997). Security in Computing, Prentice Hall Int.
(Summers '77). Summers, R. C. (1977). Secure Computing: threats & safeguards, McGraw-Hill`.
(Scambray '01). Scambray, J., S. McClure, et al. (2001). Hacking Exposed: network security secrets & solutions, Osborn/McGraw Hill.
(M.Smith '93). M.Smith (1993). Commonsence Computer Security: your practical guide to information protection, McGraw-Hill.
(Forte '00). Forte, D. (2000). "Information Security Assessment: Procedures & Methodology." Computer Fraud & Security **2000**(8): 9-12.
(Stalling '00). Stalling, W. (2000). Network Security Essentials, Prentice Hall.
(Carroll '96). Carroll, J. M. (1996). Computer Security, Butterworth-Heinemann.
(Ammann '02). Ammann, P., D. Wijesekera, et al. (2002). "Scalable, Graph-Based Network Vulnerability Analysis." Computer & Communication Security **18**(22): 217-224.
(Moore '01). Moore, A. P., R. J. Ellison, et al. (2001). Attack Modeling for Information Security and Survivability, Carnegie Mellon University**:** 1-21
(Barber '01). Barber, R. (2001). "Hacking Techniques." Computer Fraud & Security **2001**(3): 9-12.
(Hoath '98). Hoath, P. and T. Mulhall (1998). "Hacking: motivation & deterence, part 1." Computer Fraud & Security **1998**(4): 16-19.
(Rees '96). Rees, F. (1996). "New perspective on computer hackers." Computer Fraud & Security **1996**(6): 8.
(COPS '02). COPS (2002). Computer Oracle & Password System. **2002**.ftp.cert.org/pub/tools/cops
(SystemScanner '02). SystemScanner (2002). Internet Security System: System Scanner. **2002**.www.iss.net
(Bequai '01). Bequai, A. (2001). "Organised Crime Goes Cyber." Computers & Security **20**(6): 475-478.
(Nuemann '95). Nuemann, P. G. (1995). Computer Related Risks, Addison-Wesley.
(Keeney '93). Keeney, R. L. and H. Raiffa (1993). Decisions with Multiple Objectives, Press Syndicate of the University of Cambridge.
(Hinde '01). Hinde, S. (2001). "Cyberthreats: Perceptions, Reality and Protection." Computers & Security **20**(5): 364-371.
(Icove '95). Icove, D., K. Seger, et al. (1995). Computer Crime: a crimefighters handbook, O'Reilly & Associates.
(Manasse '95). Manasse, M. (1995). The Millicent Protocols for Electronic Commerce. 1st USENIX workshop on Electronic Commerce.
(W3C '99). W3C (1999). Common Markup for micropayment per-fee-links, MIT, INRIA, Keio**:** 1-33.www.w3.org/TR/1999/WD-Micropayment-Markup-19990825/
(O'Mahony '97). O'Mahony, D., M. Peirce, et al. (1997). Electronic Payment Systems, Artech House Inc.
(Coad '91). Coad, P. and E. Yourdon (1991). Object-Oriented Analysis, Prentice Hall Inc.