

(51) International Patent Classification:
G06F 21/00 (2006.01) *G06F 17/10* (2006.01)(21) International Application Number:
PCT/AU2009/001348(22) International Filing Date:
13 October 2009 (13.10.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
2008905300 13 October 2008 (13.10.2008) AU(71) Applicant (for all designated States except US): SWIN-
BURNE UNIVERSITY OF TECHNOLOGY
[AU/AU]; John Street, Hawthorn, Victoria 3122 (AU).

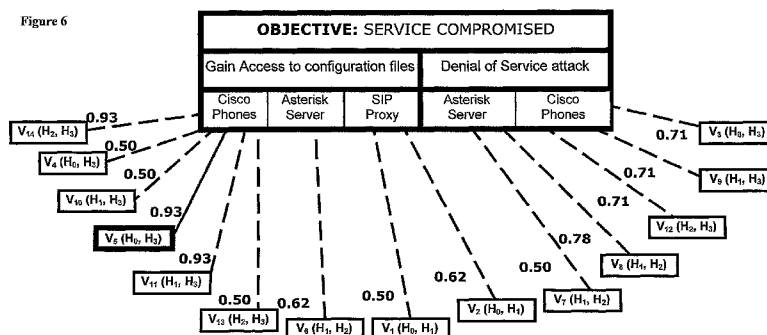
(72) Inventors; and

(75) Inventors/Applicants (for US only): VU, Hai, Le
[AU/AU]; 43 Lowan Avenue, Templestowe Lower, Vic-
toria 3107 (AU). KHAW, Kenneth, Ken-Min [MY/AU];
501/25 Wills Street, Melbourne, Victoria 3000 (AU).
KUO, Fei-Ching [AU/AU]; 6/33 Zetland Road, Mont
Albert, Victoria 3127 (AU). CHEN, Tsong, Yueh
[AU/AU]; 3 Stroma Avenue, Balwyn North, Victoria
3104 (AU).(74) Agents: DAVIES COLLISON CAVE et al.; 1 Nichol-
son Street, Melbourne, Victoria 3000 (AU).(81) Designated States (unless otherwise indicated, for every
kind of national protection available): AE, AG, AL, AM,
AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ,
CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO,
DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT,
HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP,
KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD,
ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI,
NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD,
SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT,
TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.(84) Designated States (unless otherwise indicated, for every
kind of regional protection available): ARIPO (BW, GH,
GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM,
ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ,
TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE,
ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV,
MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM,
TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW,
ML, MR, NE, SN, TD, TG).

Published:

— with international search report (Art. 21(3))

(54) Title: PROCESS AND SYSTEM FOR ASSESSING NETWORK VULNERABILITY



(57) Abstract: A process for assessing network vulnerability, including: accessing vulnerability data representing components of a communications network vulnerable to security attacks, measures of vulnerabilities of the components, and partial paths interconnecting corresponding pairs of the components; and iteratively selecting at least one terminated path interconnecting an attack goal component of the network and an attack originating component of the network, selecting one or more unterminated paths, each unterminated path interconnecting the attack goal component of the network with a corresponding one of the components other than the attack originating component, and extending only the selected one or more unterminated paths by one or more corresponding partial paths towards the attack originating component; wherein the selection of terminated and unterminated paths is based on the measures of vulnerabilities of the components of the terminated and unterminated paths.

- 1 -

PROCESS AND SYSTEM FOR ASSESSING NETWORK VULNERABILITY

TECHNICAL FIELD

The present invention relates to a process and system for assessing network vulnerability.

BACKGROUND

The protection of communications networks against malicious intrusions is crucial to the overall security of the networks, but becomes increasingly difficult with increasing network size. Attackers can combine multiple vulnerabilities (or exploits) on network hosts to achieve their goals, such as gaining access to or disrupting services offered by the network. In order to assess the vulnerability of a network to such attacks, graphical representations (referred to as attack trees or attack graphs) representing possible sequences of vulnerabilities used to reach an attack goal are constructed manually. Once an attack tree or graph has been constructed, system administrators can then improve the security of their network by patching vulnerabilities and configuration errors identified by the attack tree, in particular those which pose the greatest risk to the network.

However, the manual construction of attack graphs is error-prone and is impractical for large networks. In order to address these difficulties, recent studies have suggested the use of model checking, such as NuSMV (as described at [http://nusmv.IRST.ITC.IT](http://nusmvIRST.ITC.IT)) to automate the process of constructing attack graphs, and to ensure that they are exhaustive and succinct, as described in R.W. Ritchey and P. Ammann, "Using model checking to analyse network vulnerabilities," in *Proc. of the IEEE Symposium on Security and Privacy*, May 2001, pp. 156—165, and in O. Sheyner, J. Haines, S. Jha, R. Lippman, and J. Wing, "Automated Generation and Analysis of Attack Graphs," in *Proc. of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 2002, pp. 273-284 (hereinafter referred to as "Sheyner"), and also in T. Zhang, M. Hu, D. Li and L. Sun, "An Effective Method to Generate Attack Graph," in *Proc. of the 4th International Conference on Machine Learning and Cybernetics*, August 2005, pp. 18-21. Although such automation makes it easier to analyze and evaluate the overall vulnerabilities of a network and its security,

- 2 -

scalability remains the main concern for graph-based network vulnerability analysis because the size of the attack graph quickly becomes too large to be practical as the size of the network increases. For example, as reported by Sheyner, a network with 5 hosts and 8 vulnerabilities would result in an attack graph of 5,948 nodes and 68,364 edges in a state space of 229 bits.

There have been two main approaches in the literature attempting to address the scalability problem. In P. Ammann, D. Wijesekera and S. Kaushik, "Scalable, Graph-Based Network Vulnerability Analysis," in *Proc. of the 9th ACM Conference on Computer and Communications Security*, 2002, pp. 217—224, (hereinafter "Ammann") the authors introduce a *monotonicity* assumption that dramatically reduces the size of the attack graph without losing the information required to facilitate the network vulnerability analysis. Specifically, the monotonicity assumption assumes that the preconditions (as well as postconditions) of a vulnerability are conjoined, and that an attacker will never have to backtrack, *i.e.*, relinquish a state that the attacker has successfully exploited before.

A different approach is proposed in K. Ingols, R. Lippmann and K. Piwowarski, "Practical Attack Graph Generation for Network Defence," in *Proc. of the 22nd Annual Computer Security Applications Conf.*, Dec. 2006, pp. 121-130 (hereinafter "Ingols"), where the reachability matrix that represents the connectivity between an arbitrary node pair in a network is collapsed into smaller sub-matrices and therefore reduces the computing cost. This is done by grouping hosts that are treated identically by filtering devices and therefore have the same reachability within the subnet and across different subnets of the network. Ingols reports results for a real operational network consisting of 252 hosts, 3,777 ports, and 8,585 vulnerabilities. The resulting attack graph contains 8,901 nodes and 23,315 edges and can be further simplified by grouping nodes together for better visualization. However, although the Ingols method does reduce the size of the attack graph, the graph nevertheless remains undesirably large, and in particular is still too large and complicated for a human to interpret and analyze.

- 3 -

It is desired to provide a process and system for assessing network vulnerability that alleviate one or more difficulties of the prior art, or that at least provide a useful alternative.

SUMMARY

In accordance with the present invention, there is provided a process for assessing network vulnerability, including:

accessing vulnerability data representing components of a communications network vulnerable to security attacks, measures of vulnerabilities of said components, and partial paths interconnecting corresponding pairs of said components; and

iteratively selecting at least one terminated path interconnecting an attack goal component of said network and an attack originating component of said network, selecting one or more unterminated paths, each unterminated path interconnecting the attack goal component of said network with a corresponding one of said components other than the attack originating component, and extending only the selected one or more unterminated paths by one or more corresponding partial paths towards the attack originating component;

wherein the selection of terminated and unterminated paths is based on the measures of vulnerabilities of the components of the terminated and unterminated paths.

The present invention also provides a process for assessing network vulnerability, including:

accessing vulnerability data representing components of a communications network vulnerable to security attacks, measures of vulnerabilities of said components, and partial paths interconnecting corresponding pairs of said components; and

processing said vulnerability data to generate from said partial paths at least one terminated network path interconnecting an attack goal component of said network and an attack originating component of said network, said processing including:

- 4 -

selecting one or more of the partial paths that interconnect the attacker's goal component with one or more corresponding others of said components to provide respective unterminated paths; and

iteratively selecting one or more of the unterminated paths and extending only the selected unterminated paths by one or more corresponding ones of the partial paths towards the attack originating component to provide at least one terminated path that interconnects the attack goal component of said network and the attack originating component of said network;

wherein the one or more selected unterminated paths are selected from the unterminated paths based on the measures of vulnerabilities of said components.

The present invention also provides a process for assessing network vulnerability, including:

accessing vulnerability data representing components of a communications network vulnerable to security attacks, measures of vulnerabilities of said components, and corresponding partial paths interconnecting corresponding pairs of said components; and

processing the vulnerability data to generate one or more terminated network paths between an attack goal component of said network and an attack originating component of said network, wherein said generating includes:

(i) selecting one or more of said partial paths that are connected to said attack goal component to provide one or more unterminated paths from said attack goal component to one or more others of said components;

(ii) selecting one or more of the unterminated paths from the attack goal component based on the measures of vulnerabilities of the components of the unterminated paths;

(iii) extending only the one or more selected unterminated paths by corresponding ones of said partial paths to provide one or more corresponding extended paths; and

(iv) if one or more of the extended paths remains unterminated, then determining whether to repeat steps (ii) to (iv) using the one or more unterminated extended paths

- 5 -

as the unterminated paths, said determining being based on the measures of vulnerabilities of the extended paths.

The present invention also provides a network vulnerability assessment system, including a data processing component configured to process vulnerability data representing components of a communications network vulnerable to security attacks, measures of vulnerabilities of said components, and corresponding partial paths interconnecting corresponding pairs of said components; and to iteratively select at least one terminated path interconnecting an attack goal component of said network and an attack originating component of said network, select at least one unterminated path interconnecting the attack goal component of said network with a corresponding one of said components other than the attack originating component, and extend only the selected at least one unterminated path by one or more corresponding partial paths towards the attack originating component;

wherein the data processing component is configured to select the terminated and unterminated paths based on the measures of vulnerabilities of the components of the terminated and unterminated paths.

By effectively performing a directed search that selects (in a stepwise manner) only those one or more partial paths that are most vulnerable to attack, embodiments of the present invention include scalable processes for identifying the most vulnerable paths from an attacking node to a goal node. It is therefore practical to use the processes to determine and evaluate critical paths in large networks, where prior art methods, which generate complete attack graphs or trees, would be prohibitively large and complex.

In yet another aspect, the present invention also provides a process for assessing network vulnerability, including:

accessing vulnerability data representing scores for respective vulnerabilities of components of a communications network;

- 6 -

processing said vulnerability data to determine one or more sequences of consecutive vulnerabilities (paths) and their respective vulnerability scores that are most critical for the said network; and

starting the search of critical paths with vulnerabilities that directly achieve the attack objective and progressing toward a termination point where the attack originates.

The process thus avoids any need to construct attack graphs or trees.

Advantageously, the process may include extending the set of paths by searching paths based on relative ordering of vulnerabilities.

This reduces the sensitivity of the resulting assessments to the actual metric (*e.g.*, CVSS) used to evaluate the vulnerabilities.

In contrast to the prior art, the processes for assessing network vulnerability described herein are scalable. In some embodiments, this is achieved by determining only the most critical path or paths (*i.e.*, sequences of consecutive exploits), and without constructing large and complex representations such as attack graphs or trees. The search for critical paths starts with vulnerabilities that directly achieve the attack objective and progress toward the termination node, typically being the attacker's host computer or at least a compromised computer under the attacker's control. This progression involves determining the terminated path or paths with the highest overall vulnerability metric(s), and step-wise extending one or more other unterminated paths having a higher overall respective vulnerability metrics.

In terms of attack trees, the processes described herein allow less vulnerable paths to be pruned as soon as possible and thus eliminated from further consideration, thus reducing the computational load and making the processes highly scalable compared to prior art methods.

The vulnerability of paths constructed by extending multiple vulnerable (sub-)paths can be assessed from the vulnerabilities of the individual components of the path.

However, these vulnerabilities do not need to be assigned absolute values, as the critical paths can be determined based only on the relative vulnerabilities of the components.

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments of the present invention are hereinafter described, by way of example only, with reference to the accompanying drawings, wherein:

Figure 1 is a schematic diagram of an attack tree representing relationships between vulnerabilities of components of a communications network that can be exploited to achieve an attacker's goal;

Figure 2 is a flow diagram of a process for assessing network vulnerability in accordance with some embodiments of the present invention;

Figure 3 is a flow diagram of a critical path identification process of the process for assessing network vulnerability, in accordance with a first embodiment of the present invention;

Figure 4 is a flow diagram of a critical path identification process of the process for assessing network vulnerability, in accordance with a second embodiment of the present invention;

Figure 5 is a schematic diagram of a VoIP network used to illustrate the use of the process for assessing network vulnerability;

Figure 6 is a schematic diagram illustrating results generated by the process for assessing network vulnerability when applied to the VoIP network of Figure 5;

Figure 7 is also a schematic diagram illustrating results generated by the process for assessing network vulnerability when applied to the VoIP network of Figure 5, but following an upgrade of the Cisco VoIP phone software of the network;

Figures 8 and 9 are identical to Figures 6 and 7, but showing results generated by an alternative embodiment of the process for assessing network vulnerability; and

Figure 10 is a schematic diagram of a computer system in which the present invention can be embodied.

DETAILED DESCRIPTION

A process for assessing network vulnerability, as shown in Figure 2, alleviates or overcomes the scalability problems that plague prior art network vulnerability analyses. In particular, the processes described herein do not require the construction of an attack tree or graph which, while extremely useful for visualising and understanding the vulnerability of small communications networks, quickly becomes untenable (or at least undesirably large and/or complex) as the size of the network increases.

Notwithstanding this beneficial characteristic, an attack graph is used in the following description solely to facilitate the description and understanding of the process for assessing network vulnerability. However, it should be understood that the process does not in practice involve the construction of attack graphs or trees.

Figure 1 is an attack tree such as might be generated by an automated generation technique such as the one proposed in Sheyner. An attack tree is a graph in which each possible combination of vulnerabilities (or exploit chain) ends in a leaf state that satisfies the attacker's goal. An attack graph, on the other hand, is a collection of several attack trees in which some or all common states are merged. When the objective (or goal) of the attacker is specific, then the attack graph and attack tree both represent the same sequences of exploits the attacker can take to achieve its goal.

In the example of Figure 1, an attacker can utilize a number of vulnerability sequences to reach the goal G . Specifically, there are five vulnerabilities denoted by V_i , $i = 0, \dots, 4$, and the attacker can choose any of the following vulnerability sequences to reach the goal G : $\{V_1 \rightarrow G; V_0 \rightarrow V_1 \rightarrow G; V_2 \rightarrow V_4 \rightarrow G; V_3 \rightarrow V_4 \rightarrow G\}$. For reasons that will become apparent below, each node from which an attack originates is referred to as a termination node and is represented in Figure 1 as a bold circle. Any other node (e.g., V_4 in the example of Figure 1) is referred to as an intermediate node. Each sequence of vulnerabilities that an attacker can exploit consecutively to reach the attack goal is

referred to as a path. Paths can be classified as being critical or non-critical, as described below.

To analyze the attack tree shown in Figure 1, a vulnerability metric is first defined for each of the five vulnerabilities on the tree. Ideally, the vulnerability metric should represent the intrinsic qualities of a vulnerability, but at the same time reflect any characteristics of a vulnerability that change over time or are unique to any user's environment. The intrinsic qualities of a vulnerability are those that are constant with time and across user environments. These may include the availability, complexity and whether or not additional conditions are required to exploit the vulnerability. Each of these qualities should also be considered in the context of the impact of the vulnerability on services (if exploited) in terms of confidentiality, integrity, and availability. Furthermore, as the threat posed by a vulnerability may change over time, the vulnerability metric should also reflect factors such as the availability of exploit code or the remediation status of the vulnerability. Finally, a specific environment or service being offered can influence the potential damage that the vulnerability could cause, and the user should be allowed to alter the metric value correspondingly. The choice of vulnerability metric may depend on the specific network and application, such as the example described below where the level of security of a VoIP network is assessed.

For a vulnerability metric that satisfies the above principles, two operations are defined to capture the dependency among vulnerabilities within an exploit chain, and to reflect the relationship between different exploit chains within an attack tree. These operations are defined based on the domain of vulnerability metric denoted by V .

The first operation, denoted by $\cup: (V \times V) \rightarrow V$, gives a vulnerability metric that represents the overall vulnerability value of an exploit chain consisting of several vulnerabilities. For example, given the metrics $\{a, b, c, d, e\} \in V$ associated with

- 10 -

vulnerabilities V_i , $i = 0, \dots, 4$ as shown in Figure 1, the metric of the $V_3 \rightarrow V_4 \rightarrow G$ exploit chain is $d \cup e \in V$. We define the $\cup$ operation such that $d \cup e < d, e$ which indicates that the overall vulnerability measure of any exploit chain is less than the vulnerability measure of any individual exploit on that chain. This is because, within an exploit chain, one vulnerability has to take advantage of the successful penetration of another vulnerability preceding it in the chain, and only the last exploit achieves the attacker's goal.

The second operation, denoted by $\cap: (V \times V) \rightarrow V$, gives a vulnerability metric that represents the overall vulnerability value of two or more vulnerabilities (or exploit chains) that induce the same postcondition after the vulnerabilities have been successfully exploited. For example, as shown in Figure 1, both V_2 and V_3 vulnerabilities reach the same node (V_4) on the attack tree (*i.e.*, induce the same postcondition), and thus their overall vulnerability value (metric) is determined as $c \cap d \in V$. We define the $\cap$ operation such that $c \cap d \geq c, d$ reflecting the fact that an attacker can choose either of the exploits to achieve the same objective (postcondition), and therefore the overall metric is greater than or equal to vulnerability value of individual exploits.

The following properties are associated with these two operations :

- (i) $a \cup b < a, b$ and $a \cap b \geq a, b$;
- (ii) $\cup$ is a commutative and associative operation, *i.e.*, $a \cup b = b \cup a$ and $a \cup (b \cup c) = (a \cup b) \cup c$ (however, although the commutative property holds in theory, reversing the order of this operation might not be valid in practice because vulnerabilities usually follow a strict order in an exploit chain); and
- (iii) $\cap$ is a commutative and associative operation, *i.e.*, $a \cap b = b \cap a$ and $a \cap (b \cap c) = (a \cap b) \cap c$.

In order to identify the most critical path or chain of exploits that poses the greatest risk to the system, in the described embodiment, the $\cap$ operation is defined as a maximum operation, *i.e.*, $a \cap b = \max\{a, b\} \in \mathbf{V}$. However, it should be understood that other functions can be used in alternative embodiments, as described further below. A *minimum cut-set* is then defined for an arbitrary attack tree (or attack graph) as consisting of a set of exploits such that the attacker's goal will not be achieved if any of the vulnerabilities is removed from the set. Each minimum cut-set is a chain of exploits, referred to herein as a *critical* path, within the attack tree that an attacker can follow to achieve its goal. In contrast, a *non-critical* path is a path that can still be used by an attacker to achieve its objective, even if some of the vulnerabilities are removed from that path. Generally, an attack tree (or graph) consists of both critical and non-critical paths; for example, an automated model checker can generate every possible combination of exploits that satisfies the attacker's goal. However, the process described herein determines the most critical path whose overall vulnerability metric represents the vulnerability measure of the network represented by the attack tree.

The following lemma states an important result using the operations described above.

Lemma 1: For any attack tree consisting of nodes (network states) and directed edges (vulnerability exploits), applying the $\cup$ and $\cap$ operations over all possible goal-induced exploit chains of the attack tree provides the overall vulnerability metric of the most critical path within that tree.

Lemma 1 implies that, regardless of how many critical and non-critical paths exist in an attack graph, the resulting metric obtained via $\cup$ and $\cap$ operations over all possible paths is the overall vulnerability metric of the most critical path only. The fundamental principles of Lemma 1 can be illustrated using the example shown in Figure 1. In particular, there are three critical paths (minimum cut-sets): $\{V_1 \rightarrow G; V_2 \rightarrow V_4 \rightarrow G; V_3 \rightarrow V_4 \rightarrow G\}$, and one non-critical path: $V_0 \rightarrow V_1 \rightarrow G$ in Figure 1. Both V_1 and $V_0 \rightarrow V_1$ reach the attacker's goal,

- 12 -

and their overall metric is calculated as $(a \cup b) \cap b = \max\{(a \cup b), b\} = b$, which is a metric of the vulnerability V_1 . Thus the resulting metric will only depend on V_1 and not on the metric of $V_0 \rightarrow V_1$ attack chain. This is because exploiting V_1 alone satisfies the attacker's objective, regardless of whether V_0 is exploited or not, and from the vulnerability analysis point of view, V_1 is a critical path (or a minimum cut-set) that leads to system compromise. Continuing over the remaining exploit chains $V_2 \rightarrow V_4 \rightarrow G$ and $V_3 \rightarrow V_4 \rightarrow G$ gives the metric of the most critical path of the example attack tree.

Based on the above, a process 200 for assessing network vulnerability, as shown in Figure 2, begins by determining network design information, including network topology and connectivity information, as described below, at step 202. This information is typically provided by network administrators, but can also (or alternatively) be provided (or supplemented) by network scanning tools such as the network mapper nmap, as described at <http://nmap.org/>, and Wireshark, as described at <http://www.wireshark.org>.

At step 204, network vulnerability information is determined. In part, such information can be provided by network administrators by identifying existing security measures in the network such as policies, procedures and rules placed in firewalls, routers/switches and other network devices. In practice, some of this information might not be available, and the process can then rely on information discovered by software tools. For example, in addition to network design information, nmap and Wireshark also gather more detailed information on individual network components, including open ports, operating system version identifiers, and running services. More particularly, network vulnerability scanning tools such as Nessus, as described at <http://www.nessus.org/nessus/>, and Retina, as described at <http://www.eeye.com/html/Products/Retina/index.html>, can be used to identify specific vulnerabilities of the network, and gather any related and available security data. These tools, however, do not specify which network component(s) each vulnerability is associated with.

- 13 -

In the prior art, such vulnerabilities are used to generate an attack graph representing all possible sequences of exploits, an approach that quickly becomes impractical for large networks. In contrast, the process described herein performs a directed search for the most critical path and its overall vulnerability metric based on the design and vulnerability information determined at steps 202 and 204. Because the search is directed and does not need to consider all possible combinations of vulnerabilities, it is substantially more scalable than prior art methods.

In order to describe the process in detail, the following notations are used. Let $\mathbb{H} = \{H_0, H_1, H_2, \dots\}$ be a set of network components, where H_i , $i \geq 0$ are basic components of the targeted network such as computers, routers and telephone devices, etc, with the possible exception of H_0 , which is a host used by the attacker to launch an attack, and usually resides outside the targeted network but is connected to it via a wide-area communications network such as the Internet). Let $\mathbb{R}$ be a reachability matrix representing the connectivity between network components, as determined from the available design information of step 202.

At step 206, a set $\mathbb{V} = \{V_1, V_2, \dots\}$ of vulnerabilities associated with the targeted network is determined from the available design and vulnerability information determined at steps 202 and 204. Specifically, each vulnerability $V_i \in \mathbb{V}$ is a tuple (*source*, *target*, α , C_1 , C_2) where *source* is a unique identifier of the host/device inside or outside the targeted network where the attack originates from, *target* is a unique identifier of the component in the targeted network where the corresponding vulnerability exists, and $\alpha \in \mathbb{V}$ is a vulnerability metric for that vulnerability, as described below. C_1 represents a precondition that must be satisfied for the vulnerability to be exploited, and C_2 represents a postcondition that is satisfied after the specific vulnerability has been exploited. It will be apparent that the *source* and *target* components of each tuple effectively defines a partial path interconnecting these two components within the network.

- 14 -

A chain of exploits consisting of two vulnerabilities V_i and V_j , $i \neq j$ is represented by $V_{i,j} = (source(i), target(j), \alpha(i) \cup \alpha(j), C_1(i), C_2(j))$, where the exploit chain starts from the network component $source(i)$ having the vulnerability V_i and finishes at the network component $target(j)$, with the corresponding pre- and post-conditions $C_1(i)$, $C_2(j)$ and vulnerability metric $\alpha(i) \cup \alpha(j)$. The length of this exploit chain is represented by $|V_{i,j}| = 2$, which indicates that the attacker has to exploit two vulnerabilities V_i and V_j sequentially to achieve its goal. If the exploit chain $V_{i,j}$ requires a third vulnerability V_k to achieve the attack goal, the full exploit chain is given by $V_{i,k} = (source(i), target(k), \alpha(i) \cup \alpha(j) \cup \alpha(k), C_1(i), C_2(k))$. It will be apparent that longer exploit chains can be represented correspondingly.

Having generated the tuples, a critical path identification process 300, as shown in Figure 3, is then executed to determine the most critical path(s) and their respective vulnerability metric(s) at step 300. This process is based on the vulnerability tuples described above, the attacker's goal (objective) G in the targeted network, and an input value $d > 0$ representing the maximum length of any exploit chain that will be considered by the critical path identification process 300.

The critical path identification process 300 assumes a worst case scenario where the attacker knows about all of the identified vulnerabilities and will exploit them whenever possible in the targeted network to achieve its goal. The process also assumes monotonicity, meaning that the attacker will not backtrack during an attack (meaning that any given vulnerability will not be exploited twice in any attack).

The critical path identification process 300 searches for critical paths, beginning with vulnerabilities that directly achieve the attack objective, and progressing towards the termination node, being the node from which the attack originates (*i.e.*, the attacker's node or a node under the attacker's control). A path that reaches the termination node is referred to as a terminated path. Conversely, paths constructed from one or more partial paths

- 15 -

beginning with the attacker's goal component but that do not extend to the attacker's node are referred to as unterminated paths.

With those assumptions, the critical path identification process 300 of Figure 3 can be represented by the following pseudo-code.

- 16 -

Require: $G, \mathbb{H}, \mathbb{R}, \mathbb{V}, d > 0$

```

1:  $\mathbb{O} = \{\}, \mathbb{T} = \{\}; \nu = 0$  // Initialization
2: Calculate a vulnerability metric  $\nu$  of the most critical path
3: for  $V_i \in \mathbb{V}$  do
4:   if  $C_2(i) = G$  then
5:     for  $H_j \in \mathbb{H}$  do
6:       if  $H_j = source(i)$  then
7:         Add  $V_i$  to  $\mathbb{O}$ 
8:       end if
9:     end for
10:   end if
11: end for
12: while  $\mathbb{O} \neq \emptyset$  do
13:   for  $V_i \in \mathbb{O}$  do
14:     if  $source(i) = H_0$  then
15:       Add  $V_i$  to  $\mathbb{T}$ 
16:     end if
17:   end for
18:   if  $\mathbb{T} \neq \emptyset$  then
19:     Find  $V_i$  with  $\max \alpha$  in  $\mathbb{T} \rightarrow k : \alpha(k)$  is maximum
20:     Remove all  $V_{i \neq k}$  having  $\alpha(i) \leq \alpha(k)$  from  $\mathbb{O}$ 
21:     Set  $\mathbb{T} = \{\}$ 
22:   end if
23:   if  $\exists k \ \& \ \forall V_{i \neq k} \in \mathbb{O} : \alpha(k) \geq \alpha(i)$  or  $|V_k| > d$  then
24:     Set  $\nu = \alpha(k) \rightarrow$  FINISH
25:   else
26:     for  $V_i \in \mathbb{O}$  ( $i \neq k$  if  $\exists k$ ) do
27:       Find  $V_j \in \mathbb{V} : (target(j) = source(i)) \ \& \ (C_1(i) = C_2(j))$ 
28:       If  $\exists V_j$  then find all  $source(j) \in \mathbb{H}$  satisfying  $\mathbb{R}$ 
29:       Set  $V_i = V_{i,j}$ 
30:     end for
31:   end if
32: end while

```

- 17 -

As shown in Figure 3 (and with reference to the pseudo-code listing above), the critical path identification process 300 begins at step 302 (corresponding to lines 3 to 11 of the pseudo-code listing) by defining a new set of vulnerabilities $\mathbb{O}$ as those vulnerabilities in the set of identified vulnerabilities $\mathbb{V}$ whose post-condition is the attacker's goal G .

At step 304 (code lines 13 to 17), a new set of vulnerabilities/paths $\mathbb{T}$ is defined as those vulnerabilities/paths (if any) from the set $\mathbb{O}$ that can be directly exploited from the attacker's host H_0 ; *i.e.*, $\mathbb{T}$ is the set of terminated paths.

At step 306 (lines 19 and 20), the one or more most vulnerable paths k in set $\mathbb{T}$ are selected (*i.e.*, the one or more vulnerabilities/paths V_k in set $\mathbb{T}$ whose vulnerability metrics are equal to the maximum value $\alpha(k)$ in $\mathbb{T}$, and at step 308, any other paths (*i.e.*, with $V_{i \neq k}$) in set $\mathbb{O}$ having a lower or equal vulnerability metric than $\alpha(k)$ are then deleted from set $\mathbb{O}$. Thus the set $\mathbb{O}$ then consists only of (i) the set of terminated paths (also in set $\mathbb{T}$), and (ii) unterminated paths whose vulnerability metric values α are greater than the maximum vulnerability metric of the terminated paths in the set $\mathbb{T}$. The reason for this is that the most vulnerable paths in set $\mathbb{T}$ are candidates for the most vulnerable terminal paths because they are more direct than any other terminal path involving the other members of $\mathbb{O}$ (since they will necessarily require at least one additional node with a non-zero vulnerability metric).

- 18 -

Then, if at step 310 it is determined that there are no unterminated paths remaining in the set $\mathbb{O}$, or (step 312) if the length of the path in the set $\mathbb{T}$ is equal to or exceeds the user-selectable maximum path length set for the process 300, then (code listing lines 23 and 24) the process 300 ends, with the maximum value $\alpha(k)$ being returned as the metric for the critical path.

Otherwise, if there is at least one unterminated path remaining in the set $\mathbb{O}$ that is more vulnerable than the terminated paths in set $\mathbb{T}$, then (step 314), every unterminated path V_i in the set $\mathbb{O}$ (line 23) is extended with any another vulnerabilities V_j (line 27) from the set $\mathbb{V} = \{V_1, V_2, \dots\}$ whose postcondition and destination match the precondition and the source of its preceding vulnerability on that path (line 27), respectively. Note that if there are several sources where the above V_j vulnerability could be carried out, then a path is extended and updated for each individual source (line 28), provided that the source is reachable from the attacker's node H_0 , as determined from the reachability matrix $\mathbb{R}$. The selected vulnerability/path V_i is then replaced with the joint vulnerability $V_{i,j}$, as described above (line 29 of the pseudo-code listing). The process then loops back to select the next vulnerability/path in $\mathbb{O}$. When all such vulnerabilities have been processed, the critical path identification process 300 ends, thus also completing the process 200 for assessing network vulnerability of Figure 2.

As indicated above, the vulnerability metric α is a value that provides a measure of the vulnerability or risk of the corresponding vulnerability being exploited, or equivalently the risk of the corresponding network component being compromised. In the described embodiment, the vulnerability metric is based on the common vulnerability scoring system (CVSS), as described at <http://www.first.org/cvss/>, which provides numeric vulnerability scores between 0.0 and 10.0, and that are conveniently provided as part of the output generated by the Nessus vulnerability scanning package described above. However, it

will be apparent to those skilled in the art that any of a wide variety of other measures could be alternatively used, as described further below.

In the described embodiment, the CVSS score of a given vulnerability is scaled by 0.1 to provide a value between 0.00 and 1.00, and the resulting scaled value is used as the vulnerability metric α in the process 200 for assessing network vulnerability. This vulnerability metric α in this embodiment is interpreted as a measure of the likelihood (*i.e.*, probability) of that particular vulnerability being exploited, and thus the $\cup$ operation can be conveniently defined as a multiplication operation of probabilities. Consequently, the vulnerability metric of an exploit chain is defined as the product of the vulnerability metrics of each individual vulnerability in that chain. However, it should be understood that other operations on vulnerability metrics can be used in other embodiments, in some cases based also on the form of the vulnerability metrics, as described further below.

EXAMPLE I

To illustrate the use of the process 200 for assessing network vulnerability, it was applied to identify and evaluate critical vulnerability paths in a real VoIP network. As shown in Figure 4, the VoIP network consists of four Cisco Unified 7960 IP Phones (collectively identified as H_3) with firmware version 7.9.2; four VoIP Softphones installed on three desktop computers and a laptop (collectively identified as H_3); a SIP proxy server H_1 , and a SIP registrar server H_2 for handling calls and session initiation protocol (SIP) signalling for VoIP. The SIP Proxy server H_1 runs an open source daemon, and the SIP registrar server H_2 runs the Asterisk (version 1.2.1) open source application, as described at <http://www.asterisk.org>. The attacker's computer H_0 uses RedHat's Fedora Core 7 OS Linux-based operating system, as described at <http://fedoraproject.org>, and has various VoIP hacking tools (e.g. tcpdump, netstat, nmap) installed. In order to provide additional vulnerabilities for the process 200 to identify, superseded versions of several software packages were deliberately installed on the network components.

- 20 -

As described above, the first step 202 of the process 200 is to provide design information about the VoIP network before carrying out the actual vulnerability analysis 300. The design information, namely the network topology and relations (connectivity) between network components, are shown in Figure 4.

At step 204, the Nessus program was then used to identify vulnerabilities associated with all of the network components of the VoIP network. Simplified outputs obtained from Nessus for selected vulnerabilities are summarized in Table I below, together with corresponding vulnerabilities in the tuple format described above, namely $V(source, target, \alpha, C_1, C_2)$, with the parameters in each tuple (including the vulnerability metric α) determined from the Nessus output and the available design information.

TABLE I
SAMPLES FROM SCANNING OUTPUTS OF NESSUS.

Network Components (target)	Nessus Vulnerability Output	Vulnerability
SIP Proxy (H_1)	CVE Reference: CVE-2005-0449 Description: Allows attacker to bypass netfilter/iptables or initiate a denial of service attack. CVSS Base Score: 5.0 Pre-Condition: Linux, Kernel 2.6.x Post-Condition: Denial of Service, bypassing firewall	V (source, H_1 , 0.50, Linux-kernel-2.6.1, bypassing_firewall)
	CVE Reference: CVE-2008-1483 Description: Attackers are able to hijack a SSH Session CVSS Base Score: 6.2 Pre-Condition: OpenSSH-4.3p2 and earlier Post-Condition: Allows remote connection, Gain access to system.	V (source, H_1 , 0.62, OpenSSH-4.3p2, hijack_session)
Asterisk Server (H_2)	CVE Reference: CVE-2008-1483 Description: Attackers are able to hijack a SSH Session CVSS Base Score: 6.2 Pre-Condition: OpenSSH-4.3p2 and earlier Post-Condition: Allows remote connection, Gain access to system.	V (source, H_2 , 0.62, OpenSSH-4.3p2, hijack_session)
	CVE Reference: CVE-2008-0095 Description: Allows remote attackers to cause Denial of Service to an Asterisk via a BYE message with an Also header CVSS Base Score: 5.0 Pre-Condition: Asterisk, Open Source, 1.4.0 and previous Post-Condition: Denial of Service to Asterisk/VoIP Services	V (source, H_2 , 0.50, Asterisk-1.2.1, DoS)
	CVE Reference: CVE-2007-1306 Description: Allows remote attackers to cause Denial of Service to an Asterisk using a SIP packet without a SIP-version header. CVSS Base Score: 7.8 Pre-Condition: Asterisk, Open Source, 1.4.16 and previous Post-Condition: Denial of Service to Asterisk/VoIP Services	V (source, H_2 , 0.78, Asterisk-1.2.1, DoS)
Cisco IP Phones (H_3)	CVE Reference: CVE-2007-4459 Description: Allows remote attackers to cause Denial of Service to Cisco IP Phones using a SIP packet with 10 invalid SIP INVITE and OPTIONS. CVSS Base Score: 7.1 Pre-Condition: Cisco, VoIP Phone, CP-7960, 8.70 and previous Post-Condition: Denial of Service to Cisco IP Phones/VoIP Services	V (source, H_3 , 0.71, Cisco_IP-7.90, DoS)
	CVE Reference: CVE-2005-2181 Description: Allows remote attackers to bypass verifications with a spoof NOTIFY message. CVSS Base Score: 5.0 Pre-Condition: Cisco, VoIP Phone, CP-7960, 8.70 and previous Post-Condition: Spoof Cisco IP Phones/Gain access to VoIP Services	V (source, H_3 , 0.50, Cisco_IP-7.90, spoof_ciscophone)
	CVE Reference: CVE-2008-0531 Description: Allows remote attackers to cause buffer overflow to Cisco IP Phone and allow arbitrary code to be executed remotely via a crafted message CVSS Base Score: 9.3 Pre-Condition: Cisco, VoIP Phone, CP-7960, 8.70 and previous Post-Condition: Gain access to Cisco IP Phones/VoIP Services	V (source, H_3 , 0.93, Cisco_IP-7.90, buffer_overflow)

TABLE II
LIST OF POSSIBLE VULNERABILITIES IN THE VOIP TESTBED AND THE CORRESPONDING RELATIONS.

Network Components (source)	Relations	Possible Vulnerabilities
Attacker's PC (H_0)	$H_0 \rightarrow H_1$	V_1 (H_0, H_1 , 0.50, Linux-kernel-2.6.1, bypassing_firewall) V_2 (H_0, H_1 , 0.62, OpenSSH-4.3p2, hijack_session)
	$H_0 \rightarrow H_3$	V_3 (H_0, H_3 , 0.71, Cisco_IP-7.90, DoS) V_4 (H_0, H_3 , 0.50, Cisco_IP-7.90, spoof_ciscophone) V_5 (H_0, H_3 , 0.93, Cisco_IP-7.90, buffer_overflow)
SIP Proxy (H_1)	$H_1 \rightarrow H_2$	V_6 (H_1, H_2 , 0.62, OpenSSH-4.3p2, hijack_session) V_7 (H_1, H_2 , 0.50, Asterisk-1.2.1, DoS) V_8 (H_1, H_2 , 0.78, Asterisk-1.2.1, DoS)
	$H_1 \rightarrow H_3$	V_9 (H_1, H_3 , 0.71, Cisco_IP-7.90, DoS) V_{10} (H_1, H_3 , 0.50, Cisco_IP-7.90, spoof_ciscophone) V_{11} (H_1, H_3 , 0.93, Cisco_IP-7.90, buffer_overflow)
Asterisk Server (H_2)	$H_2 \rightarrow H_3$	V_{12} (H_2, H_3 , 0.71, Cisco_IP-7.90, DoS) V_{13} (H_2, H_3 , 0.50, Cisco_IP-7.90, spoof_ciscophones) V_{14} (H_2, H_3 , 0.93, Cisco_IP-7.90, buffer_overflow)

TABLE III
RELATIONS BETWEEN NETWORK COMPONENTS AND THEIR
VULNERABILITIES.

Relations	H_0	H_1	H_2	H_3
H_0	-	V_1, V_2	-	V_3, V_4, V_5
H_1	V_1, V_2	-	V_6, V_7, V_8	V_9, V_{10}, V_{11}
H_2	-	V_6, V_7, V_8	-	V_{12}, V_{13}, V_{14}
H_3	V_3, V_4, V_5	V_9, V_{10}, V_{11}	V_{12}, V_{13}, V_{14}	-

Combining both design and discovery information gives the list of possible vulnerabilities in the VoIP network and the corresponding relations between network components based on network topology, as shown in Tables II and III, respectively. With the inputs shown in Tables II and III, the critical path identification process 300 was then used to identify the most critical path and to evaluate the overall vulnerability metric of that path, assuming that the attacker's objective is to compromise the voice service in the network.

The results of applying the critical path identification process 300 of Figure 3 are shown in Figure 5. After just one iteration, and without building up a full-scale attack tree, the process 300 has identified the most critical path as consisting of only one vulnerability V_5 (see Table II), with the overall vulnerability metric being 0.93. For a given objective (in this case compromising the voice service), the process 300 indicates that, from a network vulnerability analysis point of view, an attack that exploits a buffer overflow vulnerability in any one of the Cisco IP phones H_3 is the most critical one. The vulnerability metric value of 0.93 indicates that this VoIP network is extremely vulnerable to attack.

To demonstrate further the capabilities of the process 300, the firmware of the Cisco IP phone was upgraded as a result of the above analysis, and the process 300 then re-applied to provide the results shown in Figure 6. After the first iteration, the most critical path is associated with vulnerability V_8 , with $\alpha = 0.78$. However, because V_8 is a later vulnerability on a chain of exploits, the process 300 executed a second iteration to produce the full exploit chain of $V_1 \rightarrow V_8$. As a result, this vulnerability chain is represented as $V_{1,8}$ with an overall vulnerability metric of 0.39, being the product of the respective metrics of

the V_1 and V_8 vulnerabilities. After the second iteration, V_2 becomes the most critical path. Therefore the vulnerability of the VoIP network is now given by the vulnerability metric of V_2 , which is 0.62. This is a substantial improvement compared to the 0.98 vulnerability metric for the previous case, reflecting the effectiveness of the action taken (*i.e.*, upgrading the firmware of Cisco IP phones) to improve network security. The scalability of the process is achieved by examining only the most critical path at any stage.

Embodiments Based on Ranking of Vulnerabilities

The processes described above evaluate critical paths based on the actual numeric values of vulnerability metrics, which depend on the particular methodology or database (in this case CVSS) used to determine these metrics. As a result, the outcome of this process is sensitive to the vulnerability metric used. In an alternative embodiment, this sensitivity is reduced and the robustness of the resulting assessment is also improved by using a relative ordering of the vulnerabilities instead of the actual numeric values. In this alternative embodiment, the vulnerabilities in the set $\mathbb{V} = \{V_1, V_2, \dots\}$ are sorted in decreasing value of the vulnerability metric α . Consequently, the index i of a vulnerability metric α_i represents the rank of the V_i exploit relative to other vulnerabilities in $\mathbb{V}$, so that vulnerabilities with lower index are individually more critical to network security. Although the vulnerability metric values α_i can be based on CVSS values as in the previous embodiment described above, in this alternative embodiment the CVSS scores are used (at least initially) only to determine the ranking of vulnerabilities.

The definition of a non-critical path is extended to a more general definition as follows. Consider two paths (chains of exploits): $X = \{V_1^{(x)}, V_2^{(x)}, \dots, V_m^{(x)}\}$ and $Y = \{V_1^{(y)}, V_2^{(y)}, \dots, V_n^{(y)}\}$ that both achieve the attacker's goal, where $V_i^{(x)}, V_j^{(y)} \in \mathbb{V}$; $1 \leq i \leq m, 1 \leq j \leq n$. Path Y is said to be non-critical if the following condition is satisfied

- 24 -

$$\prod_{i=1}^m \alpha_i^{(x)} = \alpha_1^{(x)} \times \alpha_2^{(x)} \times \dots \times \alpha_m^{(x)} \geq \quad (1)$$

$$\prod_{j=1}^n \alpha_j^{(y)} = \alpha_1^{(y)} \times \alpha_2^{(y)} \times \dots \times \alpha_n^{(y)},$$

where $\alpha_i^{(x)}, \alpha_j^{(y)}$ are the corresponding vulnerability metrics of exploits associated with each path. Observe that any chain of exploits that is not a minimum cut-set will be non-critical, because there exists a path (minimum cut-set) after removing some vulnerabilities from the chain that satisfies (1). But at the same time, some minimum cut-sets may also be non-critical according to the above definition. The following corollary summarizes the relationship between a minimum cut-set and a critical path.

Corollary 1: In any attack tree a critical path (chain of exploits) is also a minimum cut-set in that tree, but the reverse is not true.

Note that the condition (1) is also satisfied when there is a direct relative ordering between every pair of vulnerabilities in X and Y paths. This special case (which is referred to as an *ordered pair-wise* comparison) is explained in detail below. Let the vulnerability metrics belonging to the X and Y paths be sorted in descending order, and the vulnerability index being changed such that the following holds $\alpha_i^{(x)} \geq \alpha_{i+1}^{(x)}$ and $\alpha_j^{(y)} \geq \alpha_{j+1}^{(y)}$, $1 \leq \forall i < m$, $1 \leq \forall j < n$. Denote the number of vulnerabilities on X and Y by $|X|$ and $|Y|$, respectively. Thus $|X| = m$ and $|Y| = n$ and, without loss of generality, let $n \geq m$. The following corollary sheds some light on a special case where there are m ordered pair-wise vulnerabilities between the X and Y paths.

Corollary 2: For two exploit chains (or paths) $X = \{V_1^{(x)}, V_2^{(x)}, \dots, V_m^{(x)}\}$ and $Y = \{V_1^{(y)}, V_2^{(y)}, \dots, V_n^{(y)}\}$, $n \geq m$, with vulnerability metrics in descending order, path Y is said to be non-critical if $\exists Y' \subset Y$ where $Y' = \{V_1^{(y')}, V_2^{(y')}, \dots, V_m^{(y')}\}$ and $\alpha_j^{(y')} \geq \alpha_{j+1}^{(y')}$, $1 \leq \forall j < m$, such that $\alpha_i^{(y')} \leq \alpha_i^{(x)}$, $1 \leq \forall i \leq m$.

- 25 -

Note that the relative ordering $\alpha_i^{(y')} \leq \alpha_i^{(x)}$, $1 \leq i \leq m$, in Corollary 2 is maintained pairwise between vulnerabilities in X and $Y' \subset Y$, respectively. A set of these ordered pairwise vulnerabilities is referred to herein as a *pairwise set*, and this pairwise set is denoted by $\langle X, Y' \rangle$. The next lemma states a condition for a more general case when the Y path is said to be non-critical.

Lemma 2: Consider two exploit chains (or paths) $X = \{V_1^{(x)}, V_2^{(x)}, \dots, V_m^{(x)}\}$ and $Y = \{V_1^{(y)}, V_2^{(y)}, \dots, V_n^{(y)}\}$, $n \geq m$, with vulnerability metrics in descending order. Path Y is said to be non-critical if $\exists X' \subset X$ and $Y' \subset Y$ such that $\langle X', Y' \rangle$ is a pairwise set of k , $0 \leq k \leq m$, ordered pair-wise vulnerabilities between the X and Y paths and

$$(\min_{V_i \in \tilde{X}} \alpha_i^{(x)})^{|X|-k} \geq (\max_{V_j \in \tilde{Y}} \alpha_j^{(y)})^{|Y|-k},$$

where $\tilde{X}$ consists of all vulnerabilities of X not in X' , and $\tilde{Y}$ consists of all vulnerabilities of Y not in Y' .

Proof: Denoting the overall vulnerability metric of the exploit chain Y by B , we have

$$B = \prod_{i=1}^n \alpha_i^{(y)} = \prod_{i=1}^k \alpha_i^{(y')} \times \prod_{V_j \in \tilde{Y}} \alpha_j^{(y)}.$$

Similarly, the overall vulnerability metric A of the exploit chain X can be expressed as

$$A = \prod_{i=1}^m \alpha_i^{(x)} = \prod_{i=1}^k \alpha_i^{(x')} \times \prod_{V_j \in \tilde{X}} \alpha_j^{(x)}.$$

Since $\langle X', Y' \rangle$ is a pairwise set, the following relative ordering holds $\alpha_i^{(y')} \leq \alpha_i^{(x')}$, $1 \leq i \leq k$, and therefore

Putting this together, we have

- 26 -

$$\begin{aligned}
\prod_{i=1}^k \alpha_i^{(y')} &\leq \prod_{i=1}^k \alpha_i^{(x')}. \text{ Putting together, we have} \\
B &\leq \prod_{i=1}^k \alpha_i^{(y')} \times \left(\max_{V_j \in \bar{Y}} \alpha_j^{(y)} \right)^{|Y|-k} \leq \prod_{i=1}^k \alpha_i^{(x')} \times \left(\max_{V_j \in \bar{Y}} \alpha_j^{(y)} \right)^{|Y|-k} \\
&\leq \prod_{i=1}^k \alpha_i^{(x')} \times \left(\min_{V_i \in \bar{X}} \alpha_i^{(x)} \right)^{|X|-k} \leq A,
\end{aligned}$$

which proves the lemma.

Note that the condition in Lemma 2 is tighter than the condition given in (1) which is therefore less general, however; the comparison is only conducted between metrics of two individual vulnerabilities, rather than between the products of metrics from all vulnerabilities as in (1). As a result, Lemma 2 is less sensitive and requires fewer assigned values of vulnerability metrics in order to eliminate non-critical paths compared to (1). Note that if there exist $k = m$ ordered pair-wise vulnerabilities between X and Y , then Lemma 2 reduces to Corollary 2.

Knowing the relative ordering between different vulnerabilities, the objective is to find a set of all the critical paths and to eliminate paths that are non-critical without the need to build a complete attack graph and exploit every possible exploit chain in that graph.

Lemma 3: Given the relative ordering among vulnerabilities, for any attack tree consisting of nodes (network states) and directed edges (vulnerability exploits), applying the $\cap = \max$ and $\cup = \times$ operations over all possible goal-induced exploit chains of the attack tree will provide a set of critical paths of that tree.

Proof: The first part of the proof is the same as that of Lemma 1, i.e., applying the $\cap = \max$ and $\cup = \times$ operations on any goal-induced exploit chain X of the attack tree will provide the overall vulnerability metric of a minimum cut set of X . Knowing the relative ordering of vulnerability metrics, the overall vulnerability metric of some of the minimum cut sets might be definitively less than that of others, and therefore are not critical

- 27 -

according to the definition in (1). Applying the $\cap = \max$ operation between minimum cut sets will eliminate those that are non-critical, but retain those where the comparison between the overall vulnerability metrics is not conclusive. Consequently, the final result will be a set of critical paths (rather than one) of the attack tree.

Let $\mathbb{P} = \{P_1, P_2, \dots\}$ be a set of critical paths obtained after applying the $\cap = \max$ and $\cup = \times$ operations over all possible goal-induced exploit chains as stated in Lemma 2. It is assumed that the attacker is sophisticated and will not backtrack during attack, i.e., monotonicity, as described by Ammann, and D.M. Kienzle and W.A. Wulf, "A Practical Approach to Security Assessment", in Proc. of the 1997 workshop on New security Paradigms, pp. 5-16, 1998.

Denote the rank of a vulnerability with the smallest metric value in a critical path P_i by r_i , and let $|P_i|$ be the number of vulnerabilities on the path P_i . The following theorem gives the upper bound on the length of any critical path in $\mathbb{P}$.

Theorem 1: For any critical path $P_i \in \mathbb{P}$ the following inequality holds (assuming monotonicity)

$$|P_i| < \min_{\forall P_j \neq i \in \mathbb{P}} (r_j + |P_j|).$$

Proof: Suppose that $\exists P_i$ and P_j critical paths for which the above inequality does not hold; i.e.,

$$|P_i| \geq (r_j + |P_j|).$$

Observe that there are $|P_i|$ vulnerabilities in P_i that are distinct from each other due to the monotonicity assumption. Consequently, there are at least $n = |P_i| - r_j \geq |P_j|$ vulnerabilities in P_i that have smaller vulnerability metrics than the smallest metric value vulnerability in P_j . In this case, however, the P_i path is non-critical according to the Corollary 2, which contradicts our assumption that both P_i and P_j are critical paths. And since the above proof is true for any $P_j \neq i \in \mathbb{P}$, the number of vulnerabilities on the path P_i must be less than

$$\min_{\forall P_j \neq i \in \mathbb{P}} (r_j + |P_j|), \text{ and the theorem is proved.}$$

Based on the above, a process for assessing network vulnerability begins in the same manner as the previous embodiment by determining network design information, including network topology and connectivity information, at step 202. At step 204, network vulnerability information is determined.

In the prior art, such vulnerabilities are used to generate an attack graph representing all possible sequences of exploits, an approach that quickly becomes impractical for large networks. In contrast, the processes described herein perform a directed search for the most critical path based on design and discovery information available from steps 202 and 204. Because the search is directed and does not need to consider all possible combinations of vulnerabilities, it is substantially more scalable than prior art methods.

At step 206, a set $\mathbb{V} = \{V_1, V_2, \dots\}$ of vulnerabilities associated with the targeted network is determined from the available design and vulnerability information determined at steps 202 and 204.

A chain of exploits consisting of two vulnerabilities V_i and V_j , $i \neq j$ is represented by $P = (V_i, V_j) = (V_i(\text{source}), V_j(\text{target}), V_i(\alpha) \cup V_j(\alpha), V_i(C_1), V_j(C_2))$, where the exploit chain starts from the network component $V_i(\text{source})$ having the vulnerability V_i and finishes at the network component $V_j(\text{target})$ with the corresponding pre- and post-conditions $V_i(C_1)$, $V_j(C_2)$ and vulnerability metric $V_i(\alpha) \cup V_j(\alpha)$. If this exploit chain P requires a third vulnerability V_k to achieve the attack goal, then the full exploit chain is given by $(P, V_k) = (V_i(\text{source}), V_k(\text{target}), V_i(\alpha) \cup V_j(\alpha) \cup V_k(\alpha), V_i(C_1), V_k(C_2))$.

Having generated the tuples, a critical path identification process 400, as shown in Figure 4, is then executed to determine all critical path(s) based on the relative order of their respective overall vulnerability metric(s). The search of critical paths starts with any vulnerabilities that directly achieve the attack objective, and then progresses toward the termination node.

- 29 -

As shown in Figure 4 (and with reference to the pseudo-code listing below), the critical path identification process 400 begins at step 402 (corresponding to lines 3 to 12 of the pseudo-code listing) by defining new sets of vulnerabilities $\mathcal{O}$ and $\mathcal{P}$ as those vulnerabilities in the set of identified vulnerabilities $\mathcal{V}$ whose post-condition is the attacker's goal G , with terminated and unterminated paths being placed into sets $\mathcal{P}$ and $\mathcal{O}$, respectively. At step 404 (code lines 14 to 17 and 19 to 20), non-critical paths are removed from $\mathcal{P}$ and $\mathcal{O}$ based on Corollary 2 and Theorem 1, respectively.

At step 406 (code lines 23 to 25), for each of the unterminated paths in $\mathcal{O}$, a new set of vulnerabilities $\mathcal{T}$ is defined as those vulnerabilities from the set $\mathcal{V} = \{V_1, V_2, \dots\}$ determined in step 206 that are possible candidates for the extension of that path (*i.e.*, vulnerabilities defining (partial) paths that adjoin the unterminated paths in $\mathcal{O}$ and that are also reachable from the attacker's host H_0 , as determined from the reachability matrix $\mathcal{R}$).

At step 408, (code lines 26 to 34), for each of the unterminated paths in $\mathcal{O}$, select the vulnerability with the highest vulnerability metric in its corresponding set $\mathcal{T}$. If the selected vulnerability originates from the termination point, at step 410, extend each path in $\mathcal{O}$ with that vulnerability and move the resulting terminated path(s) to the set $\mathcal{P}$. Otherwise, if the selected vulnerability does not originate from the termination point, then each unterminated path in $\mathcal{O}$ is extended into two individual paths at step 412: one extended by the selected vulnerability with the highest vulnerability metric, and another with a vulnerability (if it exists) having a termination point and highest the vulnerability metric of any such vulnerabilities in $\mathcal{T}$. Any resulting terminated paths are moved to set $\mathcal{P}$.

The process then loops back to step 404, and when all paths in $\mathcal{O}$ have been processed, the critical path identification process 400 ends, thus also completing the alternative embodiment of the process 200 for assessing network vulnerability of Figures 2 and 4. The resulting critical paths are in set $\mathcal{P}$ at step 414.

- 30 -

Require: $G, \mathbb{H}, \mathbb{R}, \mathbb{V}$ and the relative ordering of $\{\alpha_i\}$

```

1:  $\mathbb{O} = \{\}, \mathbb{P} = \{\}; \nu = 0$  // Initialization
2: Find all critical paths  $\mathbb{P}$ 
3: for  $V_i \in \mathbb{V}$  do
4:   if  $V_i(C_2) = G$  then
5:     for  $H_j \in \mathbb{H}$  do
6:       if  $H_j = V_i(\text{source})$  then
7:         Set  $P_i = V_i$ 
8:         If  $H_j = H_0$  then add  $P_i$  to  $\mathbb{P}$ ; otherwise add it to  $\mathbb{O}$ 
9:       end if
10:    end for
11:  end if
12: end for
13: while  $\mathbb{O} \neq \emptyset$  do
14:  if  $\mathbb{P} \neq \emptyset$  then
15:    If  $\exists P_i \in \mathbb{P}$  is non-critical (refer to Corollary 2) then remove  $P_i$  from  $\mathbb{P}$ 
16:    Set  $\nu = \min_{\forall P_i \in \mathbb{P}} (r_i + |P_i|)$ 
17:  end if
18:  for  $P_i \in \mathbb{O}$  do
19:    if  $\nu > 0$  &  $|P_i| \geq \nu - 1$  then
20:      Remove  $P_i$  from  $\mathbb{O}$  (refer to Theorem 1)
21:    else
22:      Set  $\mathbb{T} = \{\}, P = P_i$ 
23:      for  $V_j \in \mathbb{V}$  do
24:        If  $(V_j(\text{target}) = P_i(\text{source}))$  &  $(P_i(C_1) = V_j(C_2))$ 
          and  $V_j(\text{source}) \in \mathbb{H}$  satisfies  $\mathbb{R}$  then add  $V_j$  to  $\mathbb{T}$ 
25:      end for
26:      Find  $V_k \in \mathbb{T}$  with maximum  $V_k(\alpha)$ 
27:      Set  $P_i = (V_k, P)$ 
28:      if  $V_k(\text{source}) = H_0$  then
29:        Move  $P_i$  to  $\mathbb{P}$ 
30:      else
31:        Find  $V_{h \neq k} \in \mathbb{T}$  with maximum  $V_h(\alpha)$  and  $V_h(\text{source}) = H_0$ 
32:        If  $\exists V_h$  then add  $P_h = (V_h, P)$  to  $\mathbb{P}$ 
33:      end if
34:    end if
35:  end for
36: end while

```

EXAMPLE II

Applying the alternative process 400 to the VoIP network of Figure 4, the vulnerabilities described above in the context of Example I are the same, but are now reordered according to their metric values α_i , as shown in Table IV below.

To simplify the presentation, Table IV only shows seven selected vulnerabilities $\{V_1, V_2, \dots, V_7\}$ detected on the three network devices $\{H_1, H_2, H_3\}$ of the VoIP network, where H_1 is the SIP Proxy, H_2 is the Asterisk server, and H_3 represents any of the Cisco IP Phones. Based on the individual CVSS scores, the relative ordering of the seven vulnerability metrics is then $\alpha_1 \geq \alpha_2 \geq \dots \geq \alpha_7$, where, for example, the buffer overflow attack on the Cisco IP Phone with the CVSS score of 9.3 will have the highest vulnerability metric α_1 followed by the denial of service (DoS) attack with $\alpha_2 < \alpha_1$ and the CVSS score of 7.8 on the Asterisk server as shown in Table IV. This relative ordering is used to find all possible critical exploit chains using the process described above, and no specific values of $\alpha_i \in (0, 1]$, $1 \leq i \leq 7$ are required in this process.

Furthermore, based on the design information (*e.g.*, the network topology, firewall rules and policies etc.), the *source* parameter can be determined for each of the vulnerabilities. The full list of vulnerabilities, their parameters and the corresponding relations between network components (*i.e.*, the reachability matrix $\mathbb{R}$) are shown in Tables III and IV, respectively.

Given that the attacker's objective is to compromise the voice service in this network, the process is applied to find a set of critical paths and to evaluate the overall vulnerability of the network. The results of the process are shown in Figure 7. After just one iteration, and without building up a full-scale attack tree or exploiting every possible chain of exploits, the process determines a set of critical paths consisting of only one path with a vulnerability $V_1 = (H_0, H_3, \alpha_1, \text{Cisco\_IP-7.90, buffer\_overflow})$ and an overall vulnerability metric of α_1 . For a given objective (*i.e.*, compromising the voice service), this result

indicates that, from the network vulnerability analysis point of view, the attack that exploits buffer overflow vulnerability in the Cisco IP Phone is the critical one. The vulnerability of this network can then be sufficiently quantified by the value of the vulnerability metric α_1 alone.

As with the previous Example I, the firmware of the Cisco IP Phone was then upgraded and the process re-applied while keeping the same objective, providing the results shown in Figure 8. After the first iteration, there are only two paths that are critical for this network and require further investigation. One of them is an exploit chain that consists of only one vulnerability $V_4 = (H_0, H_1, \alpha_4, \text{OpenSSH-4.3p2, hijack\_session})$ and has an overall vulnerability metric of α_4 . The other path has more than one vulnerability on its chain, i.e., it has not terminated after the first iteration of the process. As shown in Figure 8, the last vulnerability on this chain is $V_2 = (H_1, H_2, \alpha_2, \text{Asterisk-1.2.1, DoS})$ and the process is continued to obtain the full exploit chain of $V_7 \rightarrow V_2$, where $V_7 = (H_0, H_1, \alpha_7, \text{Linux-kernel-2.6.1, bypassing\_firewall})$. This chain has an overall vulnerability metric of $\alpha_7 \times \alpha_2$, and according to Lemma 2 is a non-critical path. This is because there exists one ordered pairwise vulnerability between V_4 and $V_7 \rightarrow V_2$ exploit chains, and that is $\alpha_4 \geq \alpha_7$. In other words, the process stops after two iterations, and the set of critical paths now again consists of only one critical exploit chain V_4 . In this scenario, i.e., the VoIP network after upgrading the firmware of all Cisco IP Phones, the vulnerability of the network can be sufficiently quantified by α_4 alone. Given the same objective (i.e., compromising voice service), this result indicates that, from the network vulnerability analysis point of view, the attack that attempts to hijack a SSH session on the SIP Proxy is the critical one. Observe in Table IV that the CVSS score of the vulnerability V_1 is 9.3, while that of V_4 is 6.2, indicating a large improvement in network security of the VoIP network after the firmware upgrade. Thus the overall security of a network can be significantly improved by first addressing those critical vulnerabilities obtained from the analysis provided by the process.

TABLE IV

Samples from scanning outputs of Nessus.

Network Components (<i>target</i>)	Nessus Vulnerability Output	Vulnerability
SIP Proxy (H_1)	CVE Reference: CVE-2005-0449 Description: Allows attacker to bypass netfilter/iptables or initiate a denial of service attack. CVSS Base Score: 5.0 Pre-Condition: Linux, Kernel 2.6.x Post-Condition: Denial of Service, bypassing firewall	V_7 (<i>source</i> , H_1 , α_7 , Linux-kernel-2.6.1, bypassing_firewall)
	CVE Reference: CVE-2008-1483 Description: Attackers are able to hijack a SSH Session CVSS Base Score: 6.2 Pre-Condition: OpenSSH-4.3p2 and earlier Post-Condition: Allows remote connection, Gain access to system.	V_4 (<i>source</i> , H_1 , α_4 , OpenSSH-4.3p2, hijack_session)
Asterisk Server (H_2)	CVE Reference: CVE-2008-1483 Description: Attackers are able to hijack a SSH Session CVSS Base Score: 6.2 Pre-Condition: OpenSSH-4.3p2 and earlier Post-Condition: Allows remote connection, Gain access to system.	V_4 (<i>source</i> , H_2 , α_4 , OpenSSH-4.3p2, hijack_session)
	CVE Reference: CVE-2008-0095 Description: Allows remote attackers to cause Denial of Service to an Asterisk via a BYE message with an Also header CVSS Base Score: 5.0 Pre-Condition: Asterisk, Open Source, 1.4.0 and previous Post-Condition: Denial of Service to Asterisk//VoIP Services	V_6 (<i>source</i> , H_2 , α_5 , Asterisk-1.2.1, DoS BYE msg)
	CVE Reference: CVE-2007-1306 Description: Allows remote attackers to cause Denial of Service to an Asterisk using a SIP packet without a SIP-version header. CVSS Base Score: 7.8 Pre-Condition: Asterisk, Open Source, 1.4.16 and previous Post-Condition: Denial of Service to Asterisk//VoIP Services	V_2 (<i>source</i> , H_2 , α_2 , Asterisk-1.2.1, DoS)
Cisco IP Phones (H_3)	CVE Reference: CVE-2007-4459 Description: Allows remote attackers to cause Denial of Service to Cisco IP Phones using a SIP packet with 10 invalid SIP INVITE and OPTIONS. CVSS Base Score: 7.1 Pre-Condition: Cisco, VoIP Phone, CP-7960, 8.70 and previous Post-Condition: Denial of Service to Cisco IP Phones/VoIP Services	V_3 (<i>source</i> , H_3 , α_3 , Cisco_IP-7.90, DoS)
	CVE Reference: CVE-2005-2181 Description: Allows remote attackers to bypass verifications with a spoof NOTIFY message. CVSS Base Score: 5.0 Pre-Condition: Cisco, VoIP Phone, CP-7960, 8.70 and previous Post-Condition: Spoof Cisco IP Phones/Gain access to VoIP Services	V_6 (<i>source</i> , H_3 , α_6 , Cisco_IP-7.90, spoof_ciscophone)
	CVE Reference: CVE-2008-0531 Description: Allows remote attackers to cause buffer overflow to Cisco IP Phone and allow arbitrary code to be executed remotely via a crafted message CVSS Base Score: 9.3 Pre-Condition: Cisco, VoIP Phone, CP-7960, 8.70 and previous Post-Condition: Gain access to Cisco IP Phones/VoIP Services	V_1 (<i>source</i> , H_3 , α_1 , Cisco_IP-7.90, buffer_overflow)

Although various embodiments of the present invention have been described above in terms of values representing risks or probabilities of respective vulnerabilities being exploited, it will be apparent to those skilled in the art that other measures of vulnerabilities could be additionally or alternatively used. For example, other useful

metrics or measures of a vulnerability include measures of the damage that can be caused by exploiting the vulnerability, the cost to an attacker of exploiting the vulnerability, and the cost to a network operator to mitigate or remove the vulnerability.

Moreover, although the vulnerability metrics have been described above as scalar variables, multi-dimensional metrics can be alternatively used with corresponding operations defined on those variables. For example, the vulnerability metric of a vulnerability can be a multi-dimensional or vector variable incorporating two or more respective measures associated with the vulnerability, such as, for example, the damage that can be caused by exploiting the vulnerability, the cost to the attacker of carrying out the exploit, and the likelihood of the attacker successfully carrying out that exploit.

Similarly, although the $\cap$ and $\cup$ operations in the described embodiments are maximum and product operations on the corresponding vulnerability measures, other operations can be alternatively used, and it will be apparent that this will depend in part on the particular measures used. For example, even where scalar vulnerability measures are used, the $\cap$ operation can be defined as a linear combination of metric values that is greater than any of the individual input metric values. Where other measures are used, other operations can be used depending on the desired outcome.

For example, the embodiments described above have been directed to determining the most vulnerable path in the network; *i.e.*, the path that is most likely to be exploited to reach an attacker's goal. However, where other measures such as the cost to mitigate or remove a vulnerability are included, it will be apparent to those skilled in the art that the processes described above can be modified to identify vulnerable components or paths in the network that meet other criteria as desired. For example, the process can be modified to identify those vulnerabilities that can be mitigated or removed to provide the maximum improvement to the security of the network at the lowest implementation cost to the network operator. The ability of the present invention to be applied to flexibly identify vulnerable paths based on a wide range of criteria whilst discarding or pruning other paths

- 35 -

provides powerful scalable methods of network vulnerability assessments that can be applied to large communications networks.

As will be apparent to those skilled in the art, the present invention can be embodied in a wide variety of computing systems and devices, including standard computer systems such as an Intel IA-32 or IA-64 based computer system 1000, and the processes described herein can be implemented as programming instructions of one or more software modules 1002 that operate on user-supplied input data 1004 (representing the network design information, including network topology and connectivity information), both being stored on computer-readable non-volatile and tangible (*e.g.*, hard disk, solid-state drive, flash memory, and/or optical disk) storage 1006 associated with the computer system 1000. However, it will be apparent to those skilled in the art that in other embodiments, at least one or more portions, and possibly the entirety, of the processes can alternatively be implemented in the form of one or more dedicated hardware components, such as application-specific integrated circuits (ASICs) and/or field-programmable gate arrays (FPGAs), for example.

Being a standard computer system 1000, the system 1000 includes standard computer components, as shown generically in Figure 10, including random access memory (RAM) 1008, at least one processor 1010, and external interfaces 1012, 1014, 1016, all interconnected by at least one bus 1018. The external interfaces include universal serial bus (USB) interfaces 1012, at least one of which is connected to a keyboard and pointing device such as a mouse 1020, a network interface connector (NIC) 1014 which may connect the computer system 1000 to a communications network 1024, and a display adapter 1016, which is connected to a display device 1022 such as an LCD panel display. The computer system 1000 also includes an operating system 1016 such as Microsoft Windows.

Many modifications to the described embodiments will be apparent to those skilled in the art without departing from the scope of the present invention.

- 36 -

The reference in this specification to any prior publication (or information derived from it), or to any matter which is known, is not, and should not be taken, as an acknowledgment or admission or any form of suggestion that the prior publication (or information derived from it) or known matter forms part of the common general knowledge in the field or fields of endeavour to which this specification relates.

CLAIMS:

1. A process for assessing network vulnerability, including:

accessing vulnerability data representing components of a communications network vulnerable to security attacks, measures of vulnerabilities of said components, and partial paths interconnecting corresponding pairs of said components; and

iteratively selecting at least one terminated path interconnecting an attack goal component of said network and an attack originating component of said network, selecting one or more unterminated paths, each unterminated path interconnecting the attack goal component of said network with a corresponding one of said components other than the attack originating component, and extending only the selected one or more unterminated paths by one or more corresponding partial paths towards the attack originating component;

wherein the selection of terminated and unterminated paths is based on the measures of vulnerabilities of the components of the terminated and unterminated paths.

2. The process of claim 1, including, prior to said iteratively selecting and extending, extending a plurality of unterminated paths by corresponding partial paths until at least one of the resulting extended paths is a terminated path interconnecting the attack goal component of said network and the attack originating component of said network.
3. The process of claim 1 or 2, wherein said iteratively selecting and extending includes selecting from one or more terminated paths only the at least one terminated path having the greatest vulnerability, and only the at least one unterminated path whose vulnerability is greater than the vulnerability of the selected at least one most vulnerable terminated path is selected for extending.

- 38 -

4. A process for assessing network vulnerability, including:

accessing vulnerability data representing components of a communications network vulnerable to security attacks, measures of vulnerabilities of said components, and partial paths interconnecting corresponding pairs of said components; and

processing said vulnerability data to generate from said partial paths at least one terminated network path interconnecting an attack goal component of said network and an attack originating component of said network, said processing including:

selecting one or more of the partial paths that interconnect the attacker's goal component with one or more corresponding others of said components to provide respective unterminated paths; and

iteratively selecting one or more of the unterminated paths and extending only the selected unterminated paths by one or more corresponding ones of the partial paths towards the attack originating component to provide at least one terminated path that interconnects the attack goal component of said network and the attack originating component of said network;

wherein the one or more selected unterminated paths are selected from the unterminated paths based on the measures of vulnerabilities of said components.

5. The process of claim 4, wherein the one or more selected unterminated paths are selected from the unterminated paths based on the measures of vulnerabilities of the components of the unterminated paths and the measures of vulnerabilities of components of a corresponding terminated path.

6. The process of any one of claims 1 to 5, wherein the iteratively selecting and extending includes generating a plurality of terminated paths that interconnect the attack goal component of said network and the attack originating component of said network, selecting at least one of the plurality of terminated paths based on the measures of vulnerabilities of the components of the plurality of terminated paths,

- 39 -

and the one or more selected unterminated paths are selected from the unterminated paths based on the measures of vulnerabilities of the components of the unterminated paths and the measures of vulnerabilities of components of the at least one selected terminated path.

7. A process for assessing network vulnerability, including:

accessing vulnerability data representing components of a communications network vulnerable to security attacks, measures of vulnerabilities of said components, and corresponding partial paths interconnecting corresponding pairs of said components; and

processing the vulnerability data to generate one or more terminated network paths between an attack goal component of said network and an attack originating component of said network, wherein said generating includes:

(i) selecting one or more of said partial paths that are connected to said attack goal component to provide one or more unterminated paths from said attack goal component to one or more others of said components;

(ii) selecting one or more of the unterminated paths from the attack goal component based on the measures of vulnerabilities of the components of the unterminated paths;

(iii) extending only the one or more selected unterminated paths by corresponding ones of said partial paths to provide one or more corresponding extended paths; and

(iv) if one or more of the extended paths remains unterminated, then determining whether to repeat steps (ii) to (iv) using the one or more unterminated extended paths as the unterminated paths, said determining being based on the measures of vulnerabilities of the extended paths.

8. The process of claim 7, wherein at step (iii) if a selected unterminated path can be extended by more than one of said partial paths, then selecting at least one of the

- 40 -

partial paths based on the measures of vulnerabilities of the partial paths and whether the resulting extended path would be a terminated path, and the selected unterminated path is extended only by the selected at least one partial path.

9. The process of claim 8, wherein at step (iv) if at least one of the extended paths is a terminated path, then the determining is also based on one or more measures of vulnerabilities of the at least one extended terminated path.
10. The process of any one of claims 1 to 9, wherein said measures of vulnerability include measures representing probabilities of respective vulnerabilities being exploited.
11. The process of any one of claims 1 to 10, wherein said measures of vulnerability include measures representing costs to an attacker of exploiting respective vulnerabilities.
12. The process of any one of claims 1 to 11, wherein said measures of vulnerability include measures representing costs associated with removing respective vulnerabilities.
13. The process of any one of claims 1 to 12, wherein the selection of paths is also based on numbers of vulnerable components in corresponding paths.
14. The process of one of claims 1 to 13, wherein only the one or more of the unterminated paths whose corresponding vulnerabilities are more likely to be exploited than at least one corresponding terminated path are selected for extending.

- 41 -

15. The process of claim 13, wherein only the one or more of the unterminated paths whose corresponding vulnerabilities are more likely to be exploited than the at least one terminated path and that include a smaller number of vulnerable components than at least one corresponding terminated path are selected for extending.
16. The process of any one of claims 1 to 15, wherein the iteratively performed sequence of selecting and extending is repeated until no unterminated paths remain, or until no unterminated paths are selected based on the measures of vulnerabilities, or until the number of components of the at least one selected terminated path exceeds a predetermined value.
17. The process of any one of claims 1 to 16, wherein said measures of vulnerabilities include numeric scores, and a measure of vulnerability of an extended path generated from a plurality of corresponding partial paths is determined from the numeric scores of the plurality of corresponding partial paths to provide a corresponding score for the extended path.
18. The process of any one of claims 1 to 17, wherein said scores include common vulnerability scoring system (CVSS) scores.
19. The process of any one of claims 1 to 18, wherein said measures of vulnerabilities include relative measures provided only as a relative ordering of said vulnerabilities rather than as absolute values.
20. The process of claim 19, wherein said relative measures are determined from numeric scores.

- 42 -

21. The process of any one of claims 1 to 20, wherein the vulnerability data for each vulnerability of a component represents the component, a corresponding partial path to a corresponding other of said components, at least one measure of the vulnerability, a corresponding pre-condition to be met in order for the vulnerability to be exploited, and a corresponding post-condition resulting from the vulnerability being exploited.
22. The process of any one of claims 1 to 21, wherein the unselected unterminated paths at each iteration are discarded.
23. The process of any one of claims 1 to 22, wherein the one or more terminated network paths between the attack goal component and the attack originating component are the most likely paths to be exploited.
24. The process of any one of claims 1 to 23, wherein process identifies one or more of said vulnerabilities that can be mitigated or removed to provide the maximum improvement in security of said network.
25. The process of any one of claims 1 to 24, wherein process identifies one or more of said vulnerabilities that can be mitigated or removed to provide the maximum improvement in security of said network at the lowest cost to an operator of said network.
26. A system for assessing network vulnerability and having components configured to execute the process of any one of claims 1 to 25.

- 43 -

27. A computer-readable storage medium having stored thereon one or more software modules with programming instructions configured for execution of the process of any one of claims 1 to 25.

28. A computer program product configured for execution of the process of any one of claims 1 to 25.

29. A network vulnerability assessment system, including a data processing component configured to process vulnerability data representing components of a communications network vulnerable to security attacks, measures of vulnerabilities of said components, and corresponding partial paths interconnecting corresponding pairs of said components; and to iteratively select at least one terminated path interconnecting an attack goal component of said network and an attack originating component of said network, select at least one unterminated path interconnecting the attack goal component of said network with a corresponding one of said components other than the attack originating component, and extend only the selected at least one unterminated path by one or more corresponding partial paths towards the attack originating component;

wherein the data processing component is configured to select the terminated and unterminated paths based on the measures of vulnerabilities of the components of the terminated and unterminated paths.

1/10

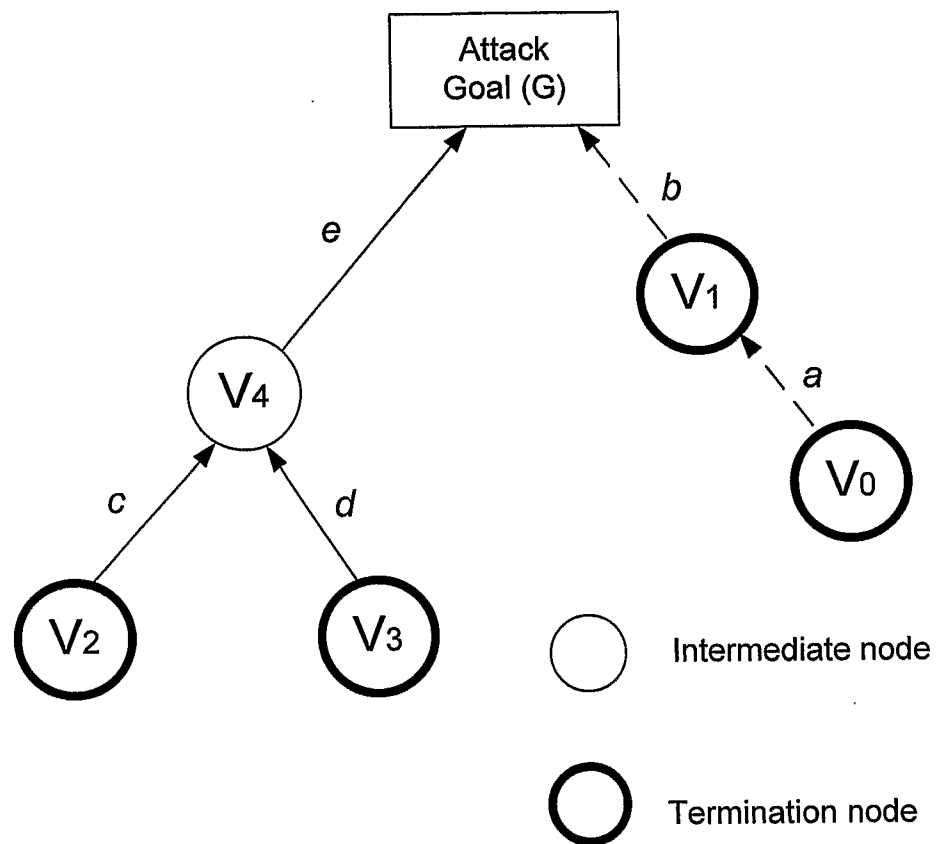
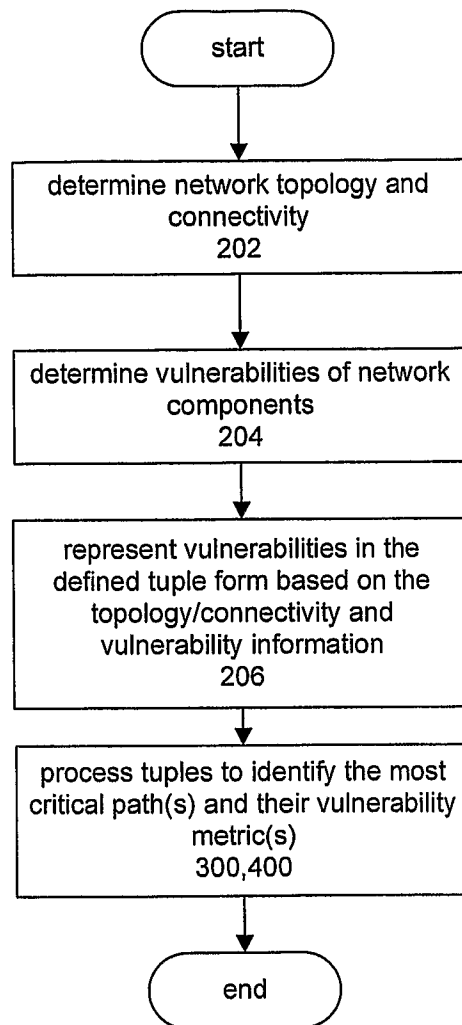


Figure 1

2/10

**Figure 2**

3/10

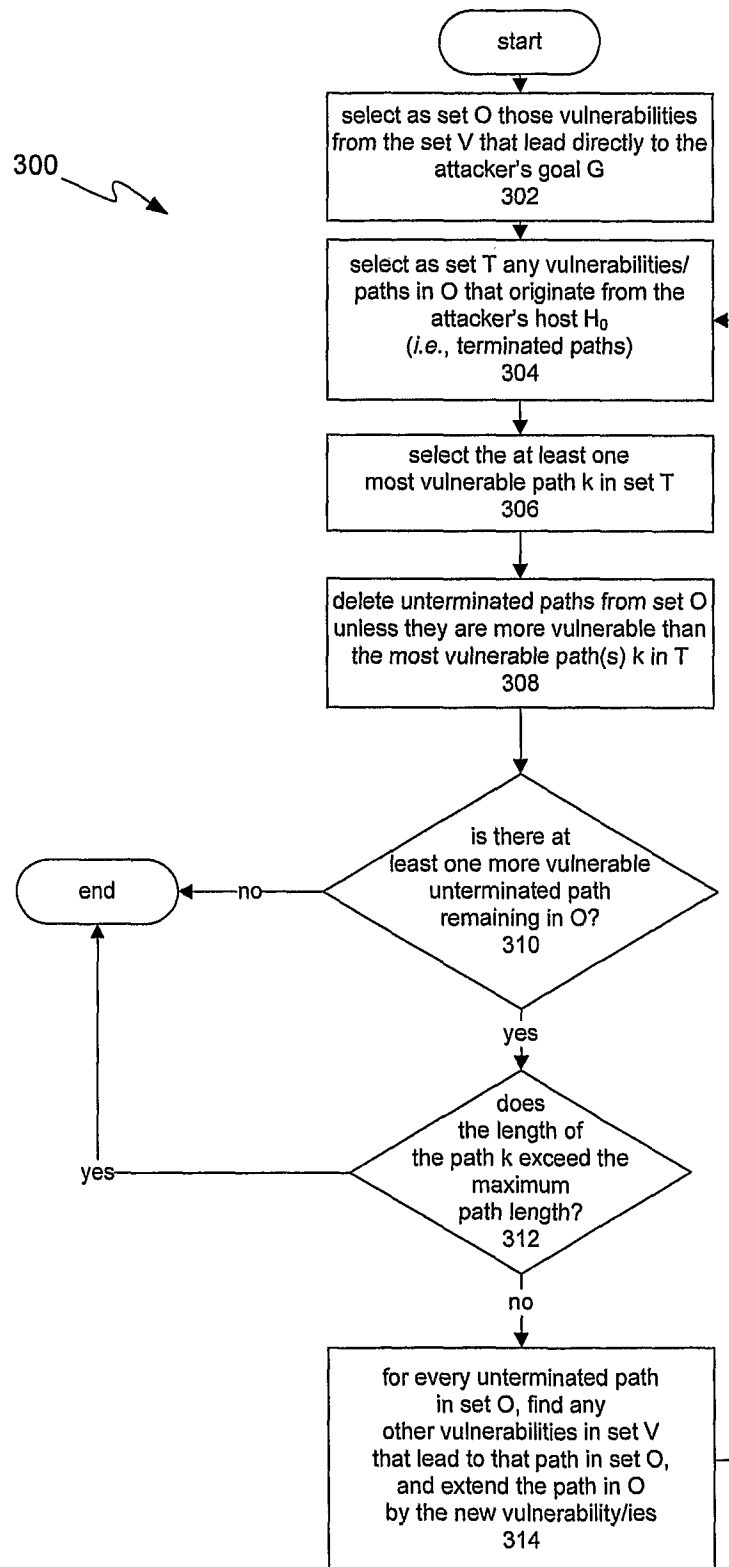


Figure 3

4/10

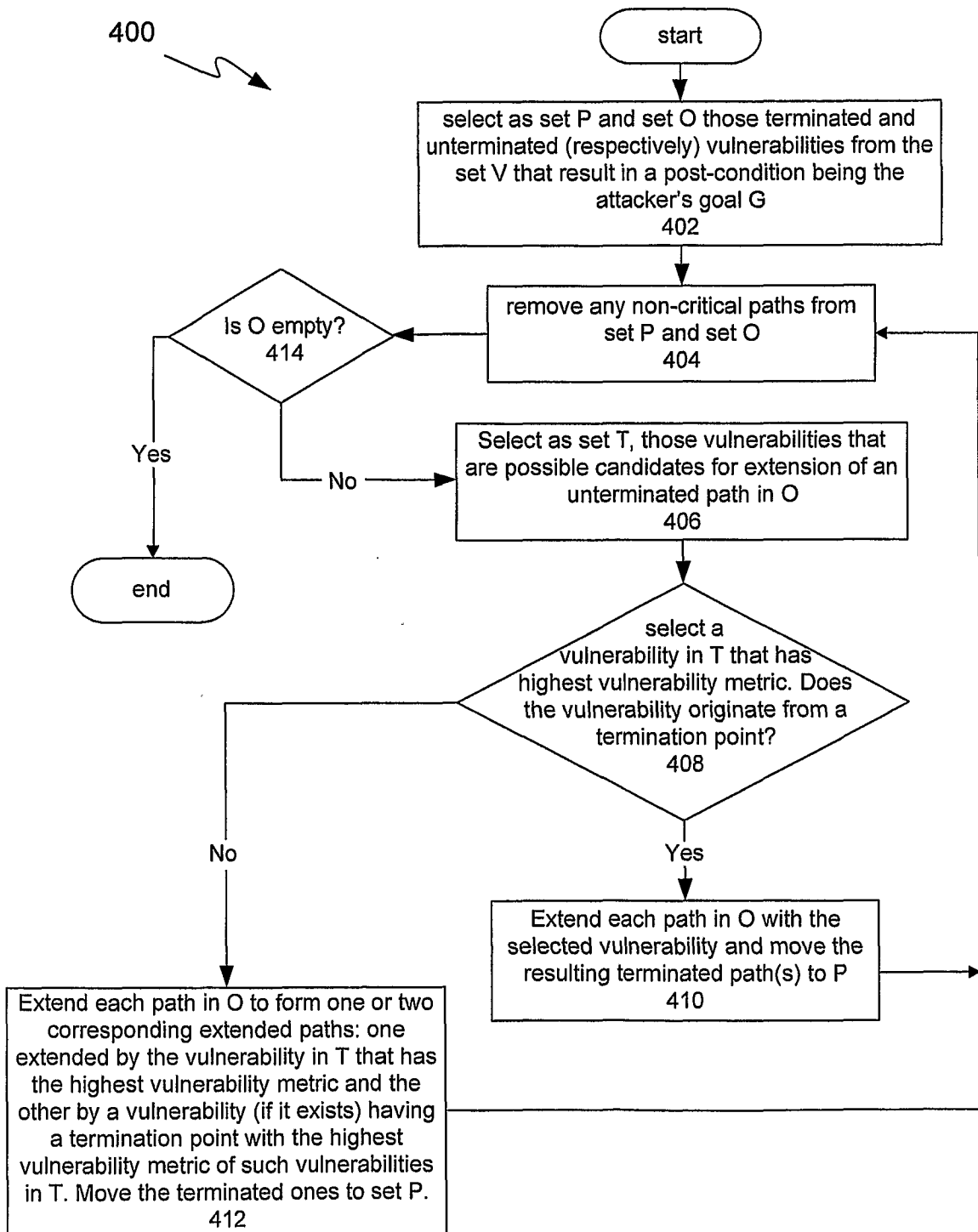


Figure 4

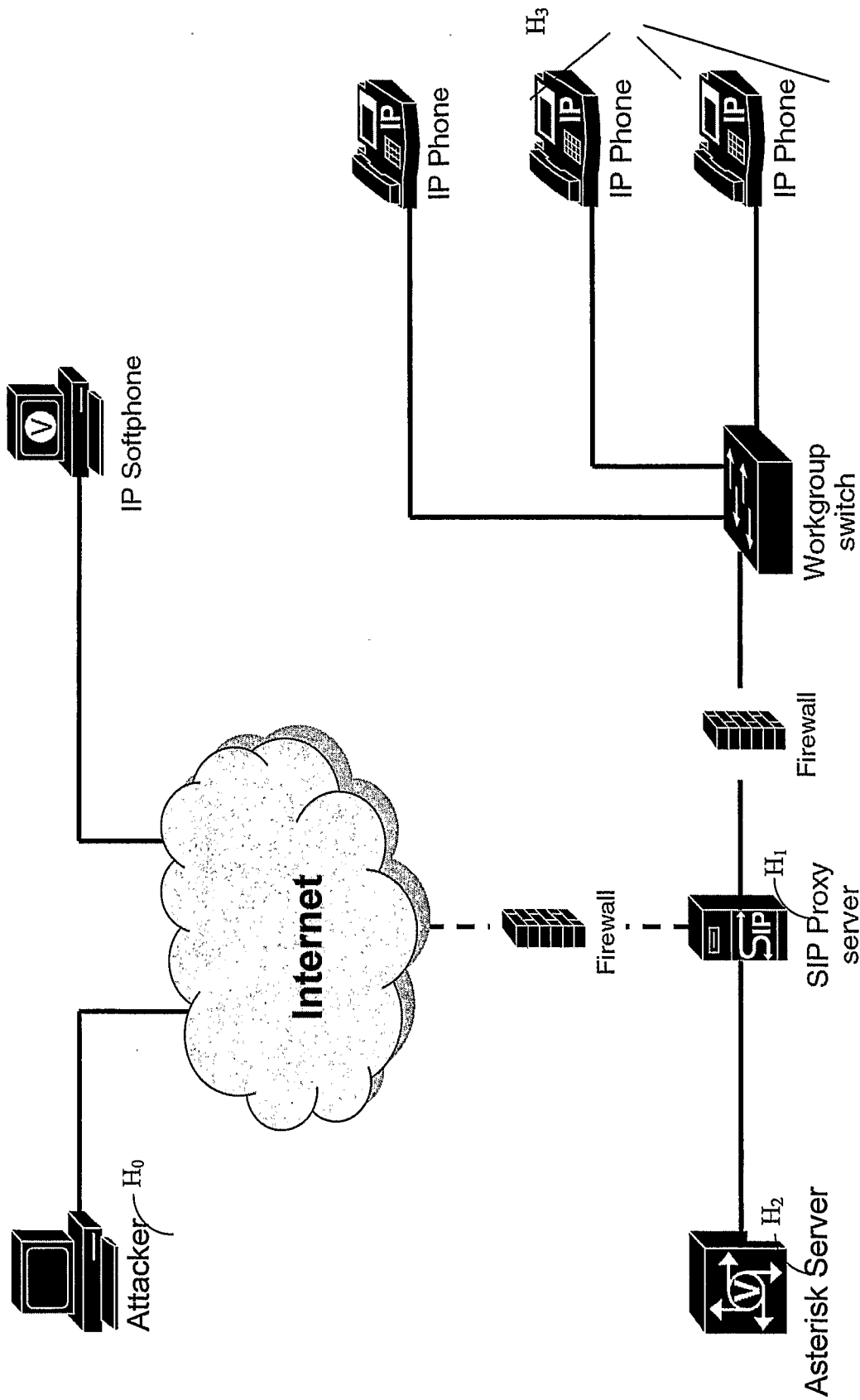


Figure 5

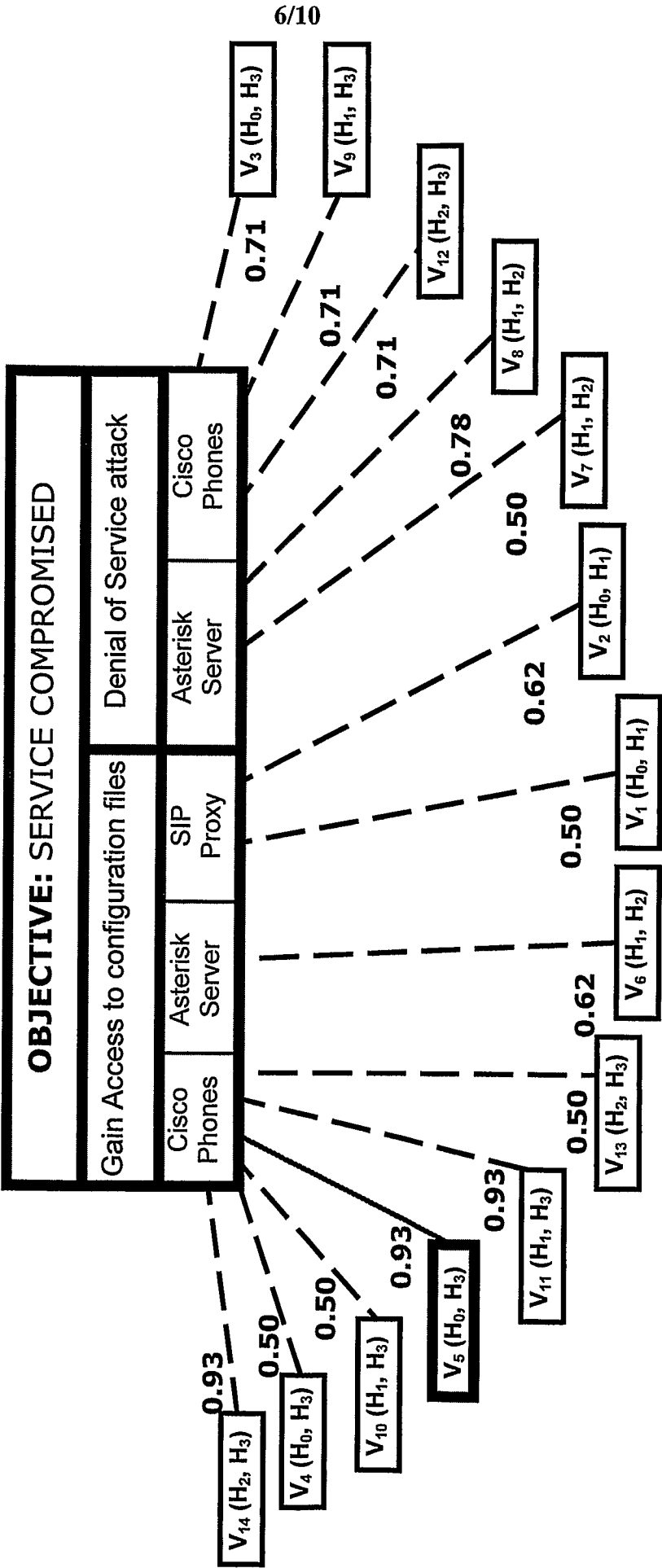


Figure 6

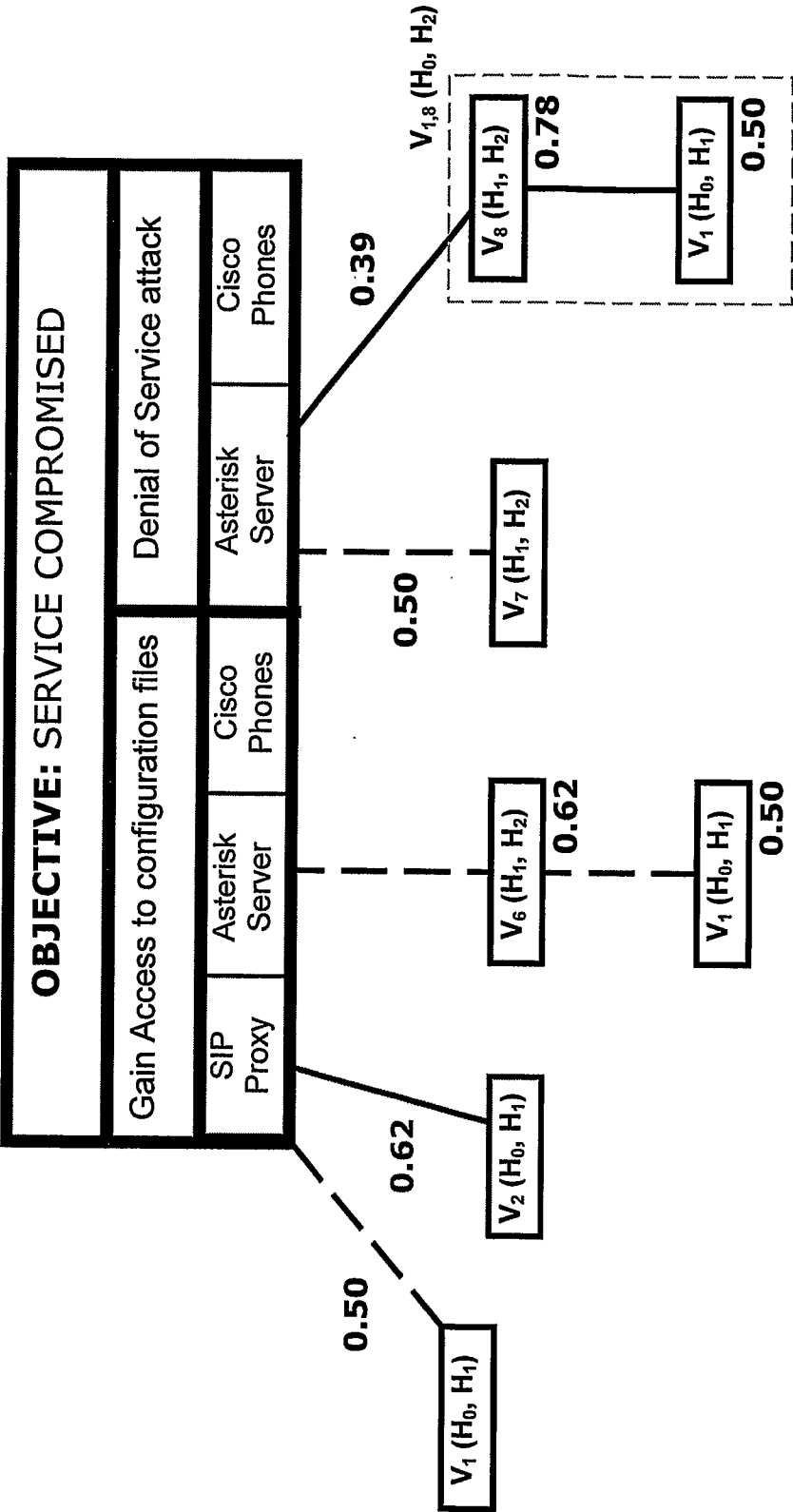


Figure 7

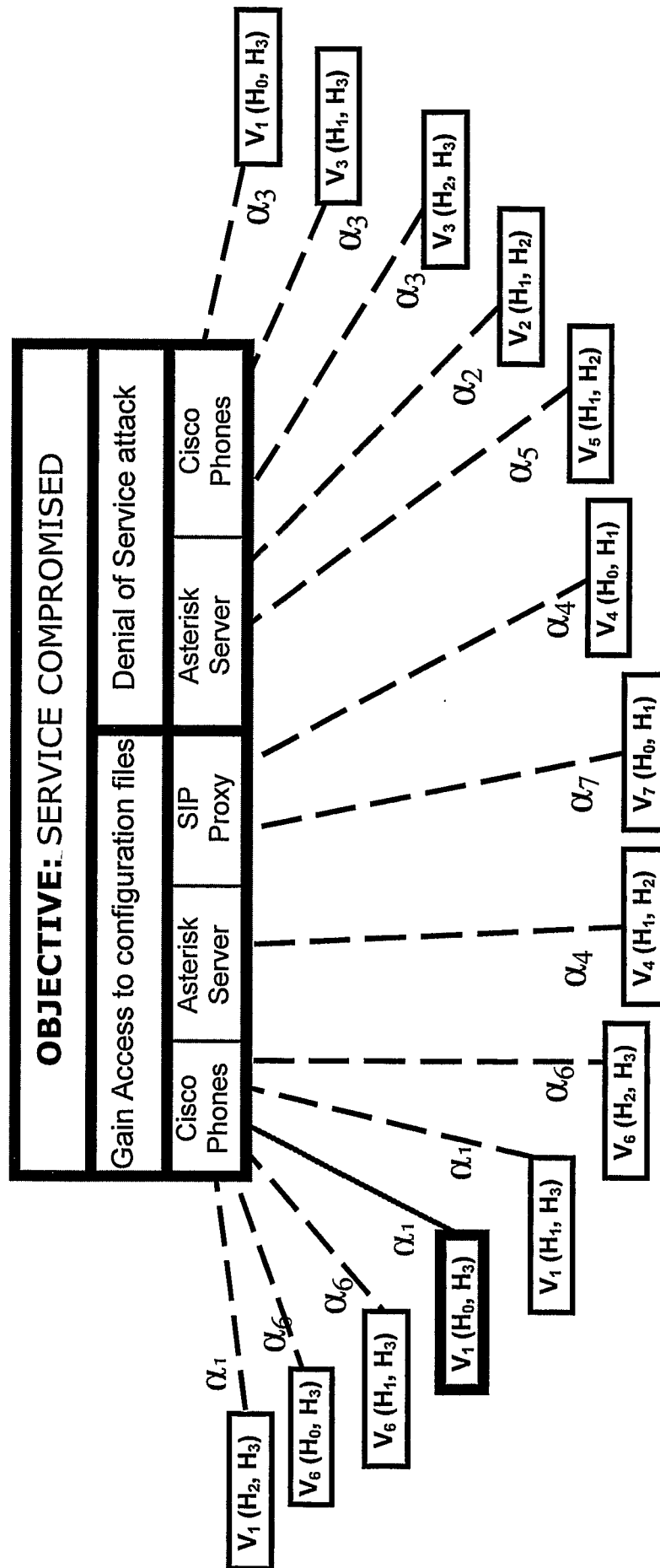


Figure 8

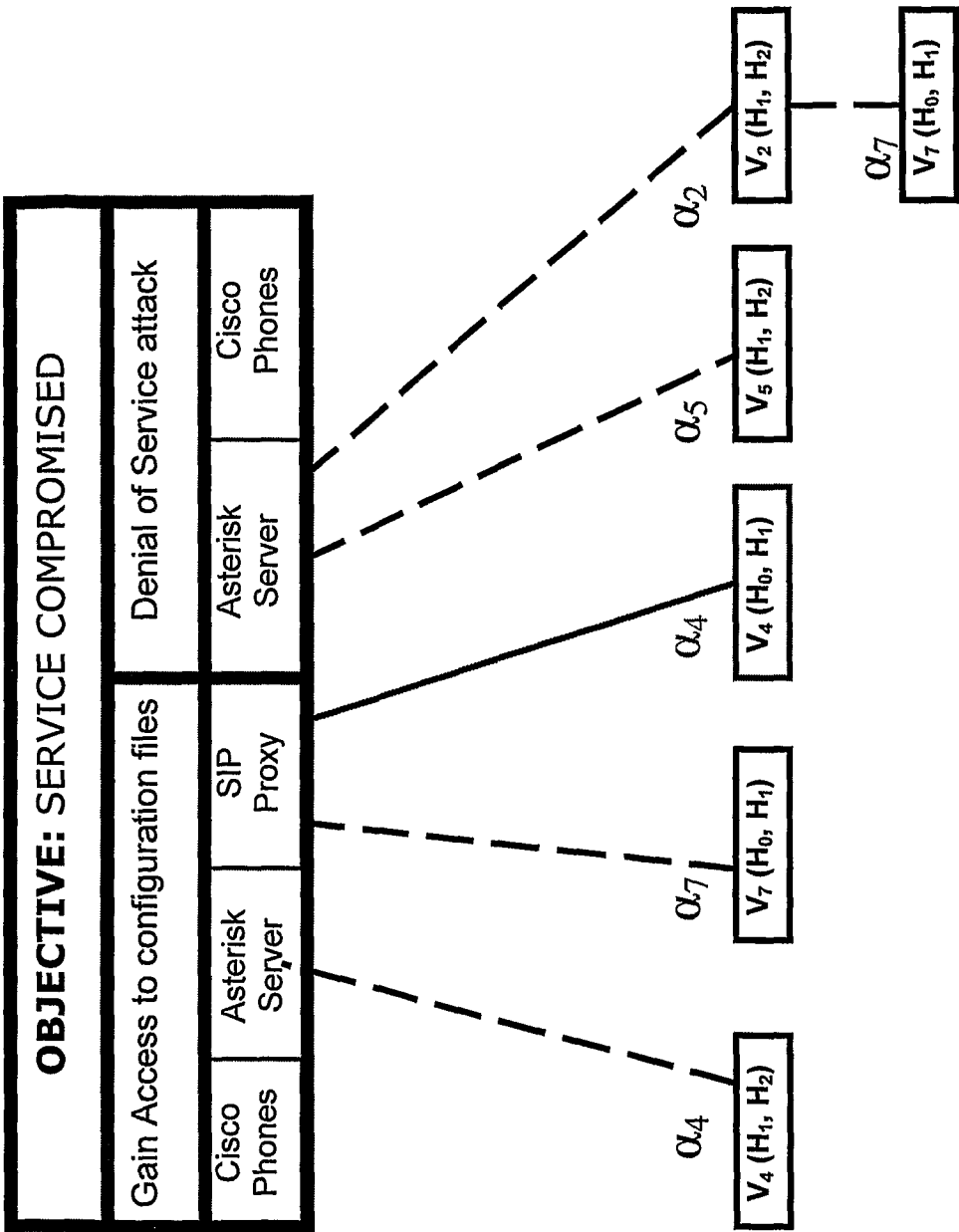


Figure 9

10/10

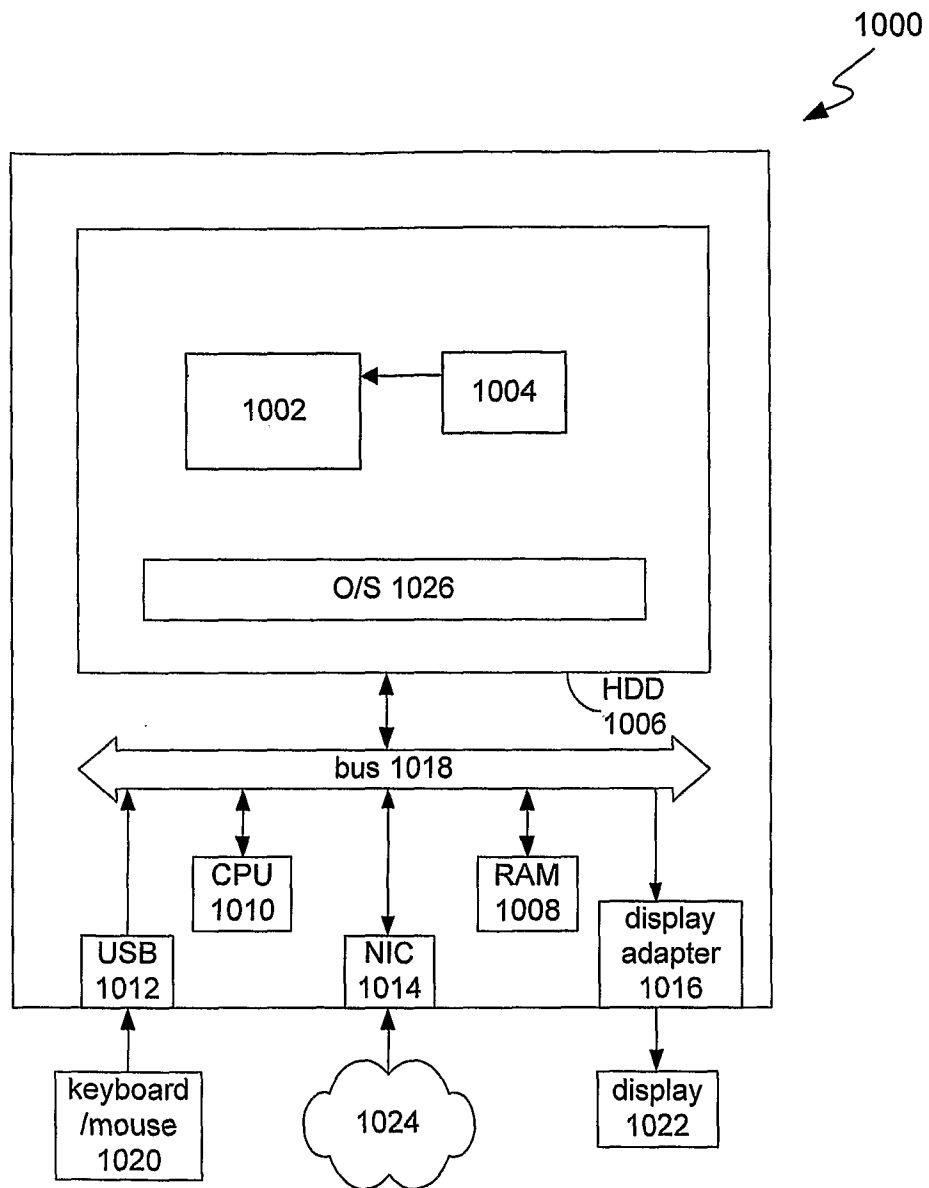


Figure 10

INTERNATIONAL SEARCH REPORT

International application No.

PCT/AU2009/001348

A. CLASSIFICATION OF SUBJECT MATTER

Int. Cl.

G06F 21/00 (2006.01)

G06F 17/10 (2006.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
 EPODOC, WPI with IPC marks and keywords including network, vulnerability, threat, exploit, risk exploit chain, critical, partial, incomplete, (un)terminated, graph, tree, path, etc

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	D1: US 2005/0138413 A1 (LIPPMANN et al.) 23 June 2005 - Abstract, figures 11, 17A, 17B, 18, 19, 25, 29A, paragraphs [0008]-[0010], [0016]-[0017], [0037], [0061], [0064]-[0065], [0098], [0116], [0123]-[0125], [0155], [0211] Note: D1 can be combined with either of documents D2 or D3 for claims 1-29.	1-29
Y	D2: US 2006/0085858 A1 (NOEL et al.) 20 April 2006 - Abstract, figure 2, paragraphs [0006], [0019]-[0020], [0041]	1-29

☒ Further documents are listed in the continuation of Box C☒ See patent family annex

* Special categories of cited documents:	
"A" document defining the general state of the art which is not considered to be of particular relevance	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"E" earlier application or patent but published on or after the international filing date	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O" document referring to an oral disclosure, use, exhibition or other means	"&" document member of the same patent family
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search
18 January 2010

Date of mailing of the international search report

20 JANUARY 2010

Name and mailing address of the ISA/AU

AUSTRALIAN PATENT OFFICE
 PO BOX 200, WODEN ACT 2606, AUSTRALIA
 E-mail address: pct@ipaustalia.gov.au
 Facsimile No. +61 2 6283 7999

Authorized officer

MLADEN MITIC

AUSTRALIAN PATENT OFFICE
 (ISO 9001 Quality Certified Service)
 Telephone No : +61 2 6283 3193

INTERNATIONAL SEARCH REPORT

International application No.

PCT/AU2009/001348

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	D3: Muthuprasanna et al. "Distributed divide-and-conquer techniques for effective DDoS attack defenses," in Proc. IEEE Intl. Conference on Distributed Computing Systems (ICDCS), pp. 93-102, Beijing, China, June 2008. [retrieved on 15 January 2010]. Retrieved from the Internet: <URL:http://vulcan.ee.iastate.edu/~gmani/personal/papers/confs/icdcs08.pdf> - Section 3, p95, left hand side column, paragraph 1	1-29
A	D4: US 7013395 B1 (SWILER et al.) 14 March 2006 - Whole document	1-29
A	D5: Savage et al. "Network support for IP traceback", Applications, Technologies, Architectures, and Protocols for Computer Communication. Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. Stockholm, Sweden, 2000. Pages: 295 – 306. [retrieved on 15 January 2010]. Retrieved from the Internet: <URL:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.73.8645&rep=rep1&type=pdf> - Whole document	1-29

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/AU2009/001348

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report		Patent Family Member			
US	2005138413	US	7194769	WO	2005060525
US	2006085858	US	7555778		
US	7013395	NONE			
Due to data integration issues this family listing may not include 10 digit Australian applications filed since May 2001.					
END OF ANNEX					