Certificate

It is hereby certify that work which is being presented in this thesis **"Network Vulnerability Detection Using Ant Colony Optimization "** , in partial fulfillment of the requirement for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala is an authentic record of research work carried out under the supervision of **Dr. Maninder Singh** and refers other researcher's work which are duly listed in the reference section. The matter presented in thesis has not been submitted for the award of other degree of this or any other university.

(Yogesh Kumar)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr.Maninder Singh)

Associate Professor

Computer Science and Engineering Department

Thapar Unversity

Patiala

Countersigned by

(Dr. Rajesh Bhatia)

Head

Computer Science & Engg. Dept

Thapar University,

Patiala

(Dr.R.K.Sharma)

Dean(Academic Affairs)

Thapar University,

Patiala

Acknowledgement

I would like to express sincerest thanks to my thesis supervisor Dr.Maninder Singh, Associate Professor, Computer Science and Engineering Department for his inspiration, guidance, stimulating suggestions, immense help and support throughout the period of this research work. He has provided me with all the necessary resources including motivation and research environment without which it would not have been possible to complete this work. It was a great opportunity for me to do this work under his supervision.

I would like to thank Dr.Rajesh Bhatia (Assistant Professor and Head), Computer Science and Engineering Department for his moral support and the research he had facilitated for this work.

I would also like to thank all my teachers for their stimulating discussions and invaluable support I received during this period of research. I am thankful to the authors whose work I have consulted and quoted in this work.

Finally, I wish to thank my dearest family and friends especially Vandana, tikka, Chinki, Shirin, Anoop, Rashmi for all their immense love, enthusiastic encouragement and support throughout my life without which it would not have been possible to complete this work. Last but not the least I would like to thank God who has always been with me in my good and bad times.

Yogesh Kumar

(800832027)

Network Vulnerability Detection Using Ant Colony Optimization

Thesis submitted in the partial fulfillment of the requirements for the award of degree of

Master of Engineering

In

Computer Science Enggineering

by

Yogesh Kumar

Roll No 800832027

under the supervision of

Dr. Maninder Singh

(Associate Professor)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

Patiala-147004

July-2010

Certificate

It is hereby certify that work which is being presented in this thesis **"Network Vulnerability Detection Using Ant Colony Optimization "** , in partial fulfillment of the requirement for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala is an authentic record of research work carried out under the supervision of **Dr. Maninder Singh** and refers other researcher's work which are duly listed in the reference section. The matter presented in thesis has not been submitted for the award of other degree of this or any other university.

(Yogesh Kumar)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr.Maninder Singh)
Associate Professor
Computer Science and Engineering Department
Thapar Unversity
Patiala

Countersigned by

(Dr. Rajesh Bhatia)
Head
Computer Science & Engg. Dept
Thapar University,
Patiala

(Dr.R.K.Sharma)
Dean(Academic Affairs)
Thapar University,
Patiala

Acknowledgement

I would like to express sincerest thanks to my thesis supervisor Dr.Maninder Singh, Associate Professor, Computer Science and Engineering Department for his inspiration, guidance, stimulating suggestions, immense help and support throughout the period of this research work. He has provided me with all the necessary resources including motivation and research environment without which it would not have been possible to complete this work. It was a great opportunity for me to do this work under his supervision.

I would like to thank Dr.Rajesh Bhatia (Assistant Professor and Head), Computer Science and Engineering Department for his moral support and the research he had facilitated for this work.

I would also like to thank all my teachers for their stimulating discussions and invaluable support I received during this period of research. I am thankful to the authors whose work I have consulted and quoted in this work.

Finally, I wish to thank my dearest family and friends especially Vandana, tikka, Chinki, Shirin, Anoop, Rashmi for all their immense love, enthusiastic encouragement and support throughout my life without which it would not have been possible to complete this work. Last but not the least I would like to thank God who has always been with me in my good and bad times.

Yogesh Kumar

(800832027)

Abstract

Security of the information in the computer networks has been one of the most important Research Area. To preserves the secure condition it is essential to be aware of the behavior of the incoming data. Is it a normal or abnormal data? It is a too vulnerable and complicated Question. Owing to the fact that intrusive data are in several and similar forms, distinguishing them from the normal ones is so astounding. Network Security is becoming an important issue for all the organizations, and with the increase in knowledge of hackers and intruders they have made many successful attempts to bring down high-profile company networks and web services.

Ant-colony optimization algorithm is an evolutionary learning algorithm which could be applied to solve the complex problems. ACO algorithm fundamental idea has been inspired by the behavior of the real ants. Ants deposit pheromone as a trace to direct the other ones in finding foods. They choose their path according to the congestion of the pheromone. The above behavior of the real ants has inspired an algorithm which a set of artificial ants, as a group of simple agents, cooperate with each other to solve a problem by exchanging information via pheromone deposited on the edges of the graph.

One of the most surprising behavioral patterns exhibited by ants is the ability of certain ant species to find what computer scientists call shortest paths. Biologists have shown experimentally that this is possible by exploiting communication based only on pheromones. ACO algorithms are the most successful and widely recognized algorithmic techniques based on ant behaviors.

We in this thesis attempt to augment Nessus Script using Java Plugin to include ACO behavior in order to detect common Vulnerabilities with ease.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to Network Security

A Computer Network is often refers to as a network, is used to share resources, applications and information through devices connected to the network. Computer network is a collection of autonomous computer interconnected by a single technology. The computers are said to be interconnected if they are able to exchange information. The connection need not be a copper wire, fiber optics, microwaves, infrared and communication satellites can be used. Networks come in many sizes, shapes and forms. The man issue here is resource sharing and the goal is to make all programs, equipment and especially data available to anyone on the network without regard to physical location of the resources and the user. There are two types of computer network configuration, peer-to-peer networks and client/server networks. Peer-to-peer are commonly implemented where less then ten computers are involved and where strict security is not necessary. Client/server networks are more suitable for larger networks.

Network security is becoming more and more important as people spend more

and more time connected to Internet. Compromising network security is often much easier than compromising physical or local security, and is much more common. Security has become more and more of an issue in recent years. Network Security is often viewed as the need to protect one or more aspect of network's operation and permitted (Access, behavior, performance, privacy and confidentiality included). It is impossible to talk about computer security without network. They two are interconnected with each other. Network Security can be views as local or global point of view depending upon the network design. Managing Security means understanding risks and deciding how to overcome if any security is violated. Network security is a level of guarantee that all the machines in a network are working optimally and the users of these machines only possess the rights that were granted to them. Network security is the most vital component in information security because it is responsible for securing all information passed through networked computers[1].

Network security is the process of preventing and detecting unauthorized access to your network. Prevention measures help you to stop unauthorized users (also known as "intruders") from accessing any part of your Network. Detection helps you to determine whether or not someone attempted to break into your system, if they were successful, and what they may have done[2]. The closest we can get to an absolutely secure machine is One unplugged from the network, power supply, locked in a safe, and thrown at the bottom of the ocean."

Network security refers to all hardware and software functions, characteristics, features, operational procedures, accountability measures, access controls, and administrative and management policy required to provide an acceptable level of protection for hardware, software, and information in a network. Network security, in order for it to be successful in preventing information loss, must follow three fundamental concepts[3].

First, a secure network must have integrity such that all of the information stored therein is always correct and protected against fortuitous data corruption as well as willful alterations. Next, to secure a network there must be confidentiality, or the ability to share information on the network with only those people for whom the viewing is intended. Finally, network security requires availability of information to its necessary recipients at the predetermined times without exception. The three principles that network security must adhere to evolved from years of practice and experimentation that make up network history.

1.1.1 Need of Network Security

There are great numbers of threat to a network's security; there are fortunately many preventative techniques to properly secure a network against those threats. There is some of the fact about the network security.

- Evolution of technology focused on ease of use.
- Increasing complexity of computer infrastructure administration and management.
- Decreasing Skill level needed for exploits.
- Direct impact of security breach on corporate asset base and goodwill.
- Increased networks environment and network based applications.

A major security objective is measuring the costs and benefits of security. If the cost is to be measure for securing an entity, whether it is data on networks, data on computers, or other assets of an organization, something has to be known about risk assessment. Generally, the assets of an organization have multiple risks associated with them, such as:

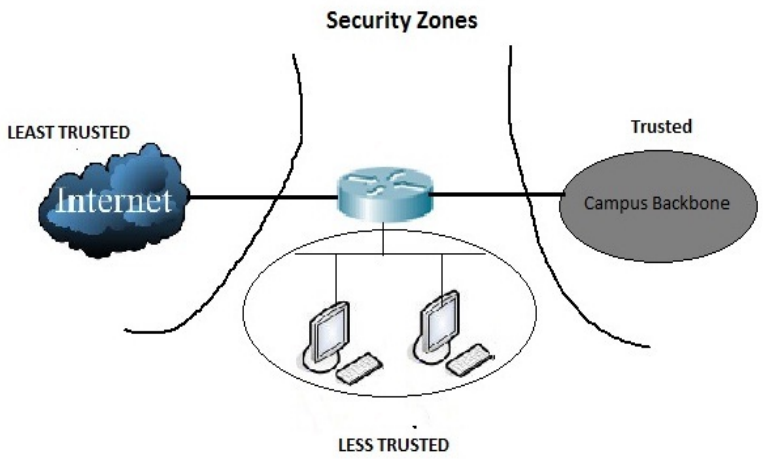

Figure 1.1 Security Zones [4]

- Equipment failure
- Theft
- Misuse
- Viruses
- Bugs

Security of a Network has three main objectives. When performing security tasks, security professionals try to protect their environments as effectively as possible. These actions can also be described as protecting confidentiality, integrity, and availability (CIA), or maintaining CIA. CIA stands for:-

- Confidentiality Ensure that no data is disclosed intentionally or unintentionally.
- Integrity Make sure that no data is modified by unauthorized personnel, that no unauthorized changes are made by authorized personnel, and that the data remains consistent, both internally and externally.

- Availability Provide reliable and timely access to data and resources[4].

1.1.2 Misconceptions about the Security

- "The goal of network security is to secure the network" (or "the computers"). Securing the network is easy, but it's not your goal. Your real goal ? and a more difficult job ? is securing the business. Security policies describe what you must secure, and the way you secure them, to support your business or mission. Firewalls, intrusion detection systems.

- "Security policies must be long and complex." In fact, just the opposite is true. We believe the well-known security axiom, "Complexity and security are inversely proportional." Complex systems are usually less secure than simple systems. Complex policies are usually ignored; simple policies might live.

- "Security policies have to be nearly perfect, or 100% complete." No. Good enough security now is better than perfect security never."

- "Security policies only have to be written once." Until there are no more bad guys in the world and everyone agrees to mind his or her own business, the process of managing a security policy never ends.

1.2 Most common types of network Attacks

- A stack overflow attack on the BIND program, used by many Unix and Linux hosts for DNS, giving immediate account access[6].

- Vulnerable CGI programs on Web servers, often supplied by the vendor as sample programs and not removed. CGI program flaws are the common means of taking over and defacing Web servers[6].

- A stack overflow attack on the remote procedure call (RPC) mechanism, used by many Unix and Linux hosts to support local networking, and which allows intruders immediate account access (this was used by most of the distributed denial of service attacks launched during 1999 and early 2000)[6].

- A bug in Microsoft's Internet Information Server (IIS) Web server software, which allowed immediate access to an administrator account on the server[6].

- A bug in sendmail, the most common mail program on Unix and Linux computers. Many bugs have been found in sendmail over the years, going back to The very first advisory issued by CERT in 1988. One of the recent flaws can be used to instruct the victim machine to mail its password file to the attacker, who can then try to crack it[6].

- A stack overflow attack on Sun's Solaris operating system, which allows intruders immediate root access[6].

- Guesses of usernames and passwords, especially where the root or administrator password is weak, or where a system is shipped with default passwords that people don't bother to change[6].

- The IMAP and POP protocols, which allow remote access to email but are often misconfigured to allow intruder access[6].

- Weak authentication in the SNMP protocol, used by network administrators to manage all types of network-connected devices. SNMP uses a default password of "public" (which a few "clever" vendors have changed to "private")[6].

1.2.1 Various Threats to Network Security

There are number threats to the Security of the network and to hack system there exists a methodological process which is discussed below.

- **Reconnaissance:** This is the first step in an attempt to gain access to the network. This is the first step used to scan the whole network and finding out the details about the target machine. There are so many tools are available for scanning a network- ranging from Ping to some of the automated tool available like SAM SPADE. These tools send packets to known host and come up with the information about that host from reply. A number of scan type can be distinguished.
  - Active Scanning
  - Passive Scanning

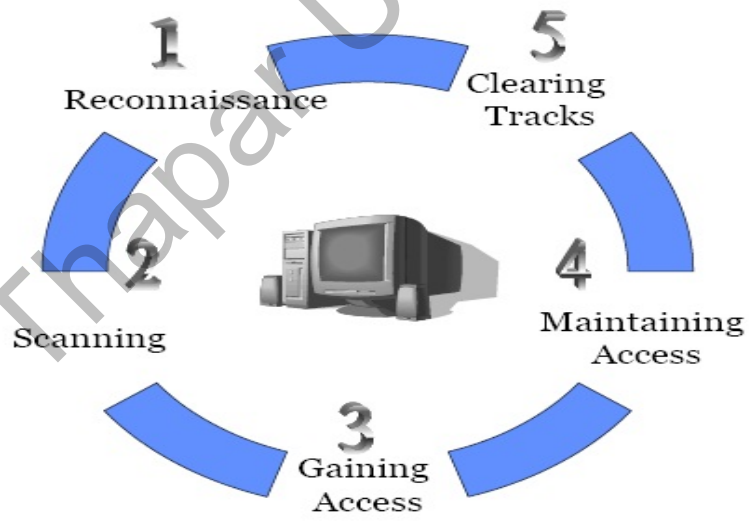

Figure 1.2 Hacking: A methodological process.

- **Scanning:** Scanning refers to the pre attack phase when the hacker scans the network with specific Information gathered during reconnaissance phase.

Hackers have to get a single point of entry to launch an attack and could be point of exploit when vulnerability of the system is detected. Scanning can include use of dialers, port scanners, network mapping, sweeping, vulnerability scanners etc.

- **Gaining Access:** Gaining access refers to the rue attack phase. The hacker exploits the system. The exploits can occur over a LAN, locally, Internet, offline, as a deception or theft. Examples include stack-based buffer overflows, denial of service, session hijacking, password filtering etc. Influencing factors include architecture and configuration of target system, skill level of perpetrator and initial level of access obtained.

- **Maintaining Access:** Maintaining access refers to the phase when the hacker tries to retain his ownership of the system. The hacker has exploited vulnerability and can tamper and compromise the system. Sometimes, hackers harden the system from other hackers as well by securing their access with backdoors, rootkits, Trojans and Trojan horse backdoors. Hackers can upload, download or manipulate data/ applications/ configurations on the owned system.

- **Covering Tracks:** Covering tracks refers to the activities undertaken by the hacker to extend his misuse of the system without being detected. Reasons include need for prolonged stay, continued use of resources, removing evidence of hacking, avoiding legal action etc. Examples include Steganography, tunneling, altering log files etc. Hackers can remain undetected for long periods or use this phase to start a fresh reconnaissance to a related target system.

1.2.2 More Threats to Network Security

- **Eavesdropping :** In general, the majority of network communications occurs in a unsecured or clear text format, which allows an attacker who has gained access to data paths in your network to listen in or interpret the traffic. When an attacker is eavesdropping on your communications, it is referred to as sniffing or snooping. The ability of an eavesdropper to monitor the network is generally the biggest security problem that administrators face in an enterprise[5].

- **Ping Scan :** This the simplest form of scan in which an attacker sends an ICMP echo request packet to a machine. In ping Echo Request is send to the end user and Echo Reply is noted down.

- **Data Modification :** After an attacker has read your data, the next logical step is to alter it. An attacker can modify the data in the packer without the knowledge of the sender or recover. Even if you don't require confidentiality for all communications, you don't want any of your messages to be modified in transit[5].

- **IP Spoofing :** Most networks and operating systems use the IP address of a computer to identify a valid entity. In certain cases, it is possible for an IP address to be falsely assumed- identity spoofing. An attacker might also use special programs to construct IP packet that appear to originate from valid address inside the corporate intranet. After gaining access to the network with a valid IP address, the attacker can modify, reroute, or delete your data[5].

- **Password-Based Attacks :** A common denominator of most operating system and network security plans is password-based access control. This means

your access rights to a computer and network resources are determined by who you are, that is, your user name and your password. Older applications don't always protect identity information as it is passed through the network for validation. This might allow an eavesdropper to gain access to the network by posing as a valid user[5].

- **Denail of Service attack :** Unlike a Password-Based attack, the denial-of-service attack prevents normal use of your computer or network by valid users[5]. After gaining access to your network, the attacker can do any of the following:
  - Randomize the attention of your internal information system staff so that they don't see the intrusion immediately, which allows the attacker to make more attacks during the diversion.
  - Send invalid data to applications or network services, which causes abnormal termination or behavior of the applications or services.
  - Block traffic, which results in a loss of access to network resources by authorized users.
- **Man-In-The-Middle Attack :** As the name indicates, a man-in-the-middle attack occurs when someone between you and the person whom your are communicating is actively monitoring, capturing, and controlling your communication transparently. For example, the attacker can re-route a data exchange. When computers are communicating at low levels of the network layer, the computers might not be able to determine with whom they are exchanging data. Man-in-the-middle attacks are like someone assuming your identity in order to read your message. The person on the other end might believe it is you because the attacker might be actively replying as you to keep the exchange

going and gain more information[5].

- **Compromised-Key Attack :** A key is a secret code or number necessary to interpret secured information. Although obtaining a key is difficult and resource-intensive process for an attacker, it is possible. After an attacker obtains a key, that key is referred to as a compromised key. An attacker uses the compromised key to gain access to a secured communication without the sender or receiver being aware of the attack. With the compromised key, the attacker can decrypt or modify data, and try to use the compromised key to compute additional keys, which might allows the attacker access to other secured communications[5].

- **Sniffer Attack :** A sniffer is an application or device that can read, and capture network data exchanges and read network packets. If the packets are not encrypted, a sniffer provides a full view of data inside the packet. Even encapsulated packets can be broken open and read unless they are encrypted and the attacker does not have access to the key.

  Using a sniffer, an attacker can do the following:

  - Analyze your network and gain information to eventually cause your network to crash or to be become corrupted.
  - Read your communications[5].

- **Application-Layer Attack :** An application-layer attack targets application servers by deliberately causing a fault in a server's operating system or applications. This results in the attacker gaining the ability to bypass normal access controls. The attacker takes advantage of this situation, gaining your application, system, or network, and can do any of the following:[5]

- Read, add, delete or modify your data or operating system.
- Introduce a virus program that uses your computers and software application to copy viruses throughout your network.
- Abnormally terminate your data applications or operating systems.
- Disable other security controls to enable future attacks.

1.3 Security Wheel

For the protection of any lose of information, security should be there to protect information from being lose. A company must implement a security policy. It is important to establish a good balance between the level of security and the ability of users to get to the information they need. The most secure PC is the one that is not connected to a network, but the problem with this approach is that nobody can access the data. A policy needs to be implemented, monitored, tested, and improved all the time[6]. Following figure shows that network security is a continuous process

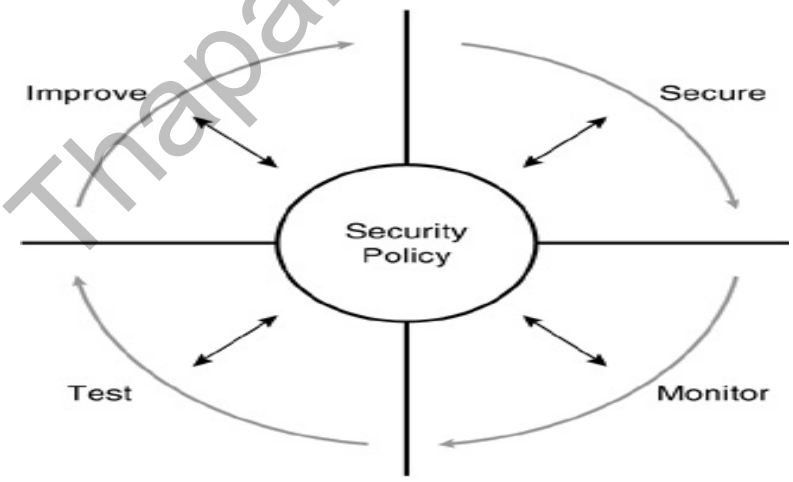

Figure 1.3 Security Policy[6]

built around a security policy. Securing your network is like a never-ending story.

Security improvements are always necessary. Hackers continually find new ways to attack your network. In the Secure phase shown in the figure, the person or department responsible for an organization's security implements security solutions to stop or prevent unauthorized access and to protect information by using the following methods:

- **Authentication**: This method is the recognition and the mapping to the policy of each individual user's identity, location, and the exact time logged on to the system. Authentication also encompasses the authorization of network services granted to users and what functions they are authorized to perform on the network.

- **Encryption**: Encryption is a method for ensuring the confidentiality, integrity, and authenticity of data communications across a network. There are several encryption methods available, and some of them, such as DES, 3DES, and AES.

- **Firewalls**: A firewall is a set of related services, located at a network gateway, that protects the resources of a private network from users from other networks. Firewalls can also be standalone devices or can be configured on most routers.

- **Vulnerability patching**: This method entails the identification and patching of possible security holes that could compromise a network and the information available on that network.

A security policy can be as simple as an acceptable use policy for the network resources, or it can be several hundred pages in length and detail every element of connectivity and associated policies. "A security policy is a formal statement of rules by which people who are given access to an organization's technology and

information assets must abide". A security policy is actually the center of the security wheel[6].

1.3.1 Importance of a Security Policy

Security policies provide many benefits and are worth the time and effort needed to develop them. Security policies are important to organizations for a number of reasons, including the following:[6]

- Create a baseline of your current security posture.
- Set the framework for security implementation.
- Define allowed and disallowed behavior.
- Help determine necessary tools and procedures.
- Communicate consensus and define roles.
- Define how to handle security incidents.

1.4 Network Vulnerability

Computers, despite being such a high technology devices, are extremely vulnerable. In fact it may be easier to steal national secrets from military computers than to steal "laddos" from a "mithai" shop. A Feature or Bug in a system or program that enable an attacker or hacker to exploits the system and to bypass security measures. An aspect of a system or network that leaves it open to an attacker. The Vulnerability is the weakness in the system that allows an attacker to reduce the system's Assurance. The Term "VULNER = NOT SECURE". Vulnerabilities are the tricks-of-the-trade for hackers, giving an intruder the ability to heighten one's

access by exploiting a flawed of logic inside the code of a computer. Vulnerabilities are usually quite mysterious and hard to prove they even exist. When one thinks of vulnerabilities, one considers a weakness in a security design, some flaw that can be exploited to defeat the defense. Computer vulnerability is a flaw in the security of the computer system. The security is the support structure that prevents unauthorized access to the computer.

When vulnerability is exploited, the person using the vulnerability will gain some additional influence over the computer system that may allow a compromise of the system's integrity. Computers have a range of different defenses, ranging from password to file permissions. Computer "virtual" existence is a completely unique concept that does not relate well to physical security. There are basic types of vulnerabilities, which are relative to two factors, what is the specific target of the vulnerability in terms of the computer or person, and the other is how quickly the vulnerability works[7].

Logic Error is a short cut directly to a security altering effect, usually considered a basic bug. This type of problem occurs due to a special circumstance that allows heightened access. This is the type of vulnerability usually thought of first[7].

Table 1.1 Vulnerability Affects

	Affects Person	**Affects Computer**
Instantaneous	Social Engineering	Logic Error
Requires a Duration of time	Policy Oversight	Logic Error Weakness

Weakness is a security measure that was put into place, but has as flow in its design that could lead to a security breach. They usually involve security that may or may not be distinctly solid, but is possible for people to bypass. The term "Security through Obscurity" fits in this arena, being that a system is secure because nobody can see or understand the hidden elements. All elements fits under this category

as it is possible to eventually break the encryption, regardless of how well it is constructed. The idea is not that security is not present, it is the fact that security is present with a method of defeating it also being present[7].

Social Engineering is a nebulous area of attacking associated with a directed attack against policy of the company. Policy is being used in a high level sense, because it could be an internal worker committing sabotage, a telephone scam directed at a nave employee, or digging for information that was thrown away in dumpsters[7].

Policy oversight is a flaw in the planning to avoid a situation, which would be such conditions as not producing adequate software backups, having proper contact numbers, having working protection equipment and so forth. The most common policy oversight seems to be not having support of the company's management to legally pursue computer criminals, which renders all the existing countermeasures established to protect the company useless[7].

1.4.1 Vulnerability Attributes

All four types of security problems ultimately have the same basic attributes, so any taxonomy of the problems for policy issues will have the same basic model for computer vulnerabilities. Vulnerabilities have five basic attributes, which are Fault, Severity, Authentication, Tactic, and Consequence. Examining these attributes can provide a complete understanding of the vulnerability[7].

Fault describes how the vulnerability came to be, as in what type of mistake was made to create the problem. The mistakes that occur which cause vulnerabilities are referred to as its fault[7].

Severity describes the degree of the compromise, such as if they gained administrator access or access to files a regular user normally would not see. All

vulnerabilities yield an outcome, therefore to judge the extent of the access level gained from a vulnerability, severity is used. There are six levels of Severity that can be used to define vulnerability[7]:

- Administrator Access
- Read Restricted Files
- Regular User Access
- Spoofing
- Non-Detestability
- Denial of Service

Authentication describes if the intruder must have successfully registered with the host proof of identity before exploiting the vulnerability. A basic Boolean yes-or-no value, authentication is a condition asking if the intruder must register identity with the host first. If the intruder must "log-in", they must have already bypassed a level of security to reach that point[7].

Tactic describes the issue of who is exploiting whom, in terms of location. If a user must have an account on the computer already, that is one situation. If the user can come from a location other than the keyboard, that is another. The way that a vulnerability is exploited is very critical, so tactic describes who can exploit whom and where. A local user will have access to far more resources than an intruder without access and so internal access is desirable and may gain access from a server function[7].

Consequence describes the outcome. Consequence is the mechanics behind access promotion and demonstrates how a small amount of access can lead to far

greater compromises. Unlike severity, which states the outcome of a single vulnerability, consequence builds a road map for almost any level of access to promote itself to fully interactive administrator rights. One can think of this aspect as the function component of the vulnerability[7].

1.5 Ant Colony Optimization

Ant Colony Optimization (ACO) is a general searching technique for the solution of difficult problems which is inspired by the pheromone trail behavior of real ant. It's an adaptive nature inspired algorithm, concretely implemented, and is applied to routing protocols in wired and wireless networks. Ant colony optimization (ACO) is a meta- heuristic algorithm inspired by the behavior of real ants. Ant colony optimization was first proposed by Dr. Marco Dorigo and his colleagues. Ant colony optimization algorithms are part of swarm intelligence. Ant can find the food and bring it back to the nest and they converge to the shortest path. Wherever they go, they let pheromones behind them, and mark the area as explored and communicating to the other ants that the way is known.

ACO is Inspired by the foraging behavior of the real ants. The first algorithm was aiming to search for an optimal path in a graph; based on the behavior of ants seeking a path between their colony and a source of food. Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets marched over faster, and thus the pheromone density remains high as it is laid on the path as fast as it can evaporate. Pheromone evaporation has also the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the

paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained. Thus, when one ant finds a good (i.e., short) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads all the ants following a single path. The idea of the ant colony algorithm is to mimic this behavior with "simulated ants" walking around the graph representing the problem to solve.

Each individual ant is relatively small and simple. On their own they are capable of running around quite well, and when they find food they are able to take it back home. Different ant species use quite a few sensory techniques to find their way home. Some use the sun, some remember how much they have turned and how far they have moved. Whatever most ants can find "home", as long as it is not too far away. Finding home is quite clever, but the really clever bit about ants is that they can cooperate. Obviously for cooperation you need some sort of communication. Well ants often use a fairly indirect method of communication, they lay pheromones. Pheromones are just chemicals that ants can "smell" using their antenae. A high concentration of pheromones can attract ants similar to their behaviour we try to exhibit their behaviour in Network.

This thesis work attempts to apply ACO Algorithm to find out Vulnerabilities in the Network. Chapters to follow will discuss it in detail.

Chapter 2

Literature Survey

Vulnerability has been discussed in brief in the previous chapter, however if we go through the literature, we came to know about the researchers who have previously worked upon this topic. How to overcome from vulnerabilities, what are the major types of vulnerability, detection of the vulnerability. ACO has been in the spot light for almost two decades, many of the work have been done in this field by different researchers. Ant Colony Optimization as the name suggest is the problem for finding out the optimal result. Started in the early 90s, a researcher Marco Dorigo, during his Ph.D thesis, came to know about the behavior of the real Ants, how they came to know about the food from the nest, and how they communicate with each other to tell other where the food is. Marco with his colleague name Di Caro and Gambarella have worked upon Ant Colony Optimization.

Vulnerability in the system means having weakness in system. These weaknesses are greatly exploits by the hacker to gain access into your system. Any vulnerable system is open to the hacker they can do anything to your system. They can steal any type of information from your computer. Main cause of presence of any type of vulnerabilities in the system is due to lack of programming. And it is due to some

flaws in the Software. When hackers came to know about this weaknesses about your system they can easily hook on to your system and can exploits them up to any extent. Some methods need to adopted to overcome from these weaknesses. Main cause of Vulnerabilities is as follows:

- **Complexity** : large, complex systems increase the probability of flaws and unintended access points.
- **Familiarity** : Using common, well-known code, software, operating system, and hardware increases the probability an attacker has or can find the knowledge and tools to exploits the flaw.
- **Connectivity** : More physical connections, privileges, ports, protocols, and services and time each of those are accessible increase vulnerability.
- **Password management flaws**: The computer user uses weak passwords that could be discovered by brute force. The computer user stores the password on the computer where a program can access it. Users re-use passwords between many programs and websites.
- **operating system design flaws**: The operating system designer chooses to enforce sub optimal policies on user/program/ management. For example operating systems with policies such as default permit grant every program and every user full access to the entire computer. This operating system flaw allows viruses and mal ware to execute commands on behalf of the administrator.
- **Fundamental operating system design flaws**: The operating system designer chooses to enforce sub optimal policies on user/program/ management. For example operating systems with policies such as default permit grant every program and every user full access to the entire computer. This operating

system flaw allows viruses and mal ware to execute commands on behalf of the administrator.

- **Internet Website Browsing** : Some internet websites may contain harmful Spyware or Adware that can be installing automatically on the computer systems. After visiting those websites, the computer systems become infected and personal information will be collected and passed on to third party individuals.
- **Software Bugs** : The programmer leaves an exploitable bug in a software program. The software bug may allow an attacker to misuse an application.
- **Unchecked user input** : The program assumes that all user input is safe. Programs that don't check user input can allow unintended direct execution of commands or SQL statements.

Type of vulnerabilities

- **Software Vulnerabilities**
- **Network Vulnerabilities**

Software Vulnerabilities Once we get to know about the identity, goals and motivations of attackers, we need to understand the various ways for software exploits to be delivered as an attack and some of the issues that surround those delivery mechanisms.

- **Buffer Overflow** : A buffer overrun is when a program allocates a block of memory of a certain length and then tries to stuff too much data into the buffer, with extra overflowing and overwriting possibly critical information crucial to the normal execution of the program. Consider the following source code: When the source is compiled and turned into a program and the program

is run, it will assign a block of memory 32 bytes long to hold the name string.

```
#include < stdio.h >
int main()
{
char name[31];
printf("Please type your name: ");
gets(name);
printf("hello,%s", name);
return 0;
}
```

Protection against Buffer Overflows

- Buffer overflow vulnerabilities are inherent in code due to poor or no error checking
- General ways of protecting against buffer overflows:
- Close the port of service
- Filter specific traffic at the firewall
- Test key application
- Run software at the least privilege required

Network Fuzzing Fuzzing involves sending random data to a network port or application in an attempt to find buffer overruns and denial-of-service vulnerabilities from incorrect parsing of the data. Fuzzing is useful in binary protocols and any other protocols that lack documentation. It is also useful in forcing exception handling code within an application to be invoked that could not be easily reached otherwise. One way of doing this is to use the random device in Linux to generate some random data, and Netcat to send it.

The command:

$ cat /dev/random — nc 192.168.1.1 80

Many security vulnerabilities are the result of programming errors that are easily detectable by inspecting the source code. Programmers might use unsafe versions of library functions, such as strcpy or strcat, and security objects like encryption keys might be freed in memory without overwriting their contents. Source scanners attempt to identify these potential vulnerabilities by searching a source file for the signature of the programming error.

- **Trojan** A Trojan in software security means a seemingly attractive or innocuous program that hides malicious software inside. Trojans aren't typically capable of spreading themselves, but instead they require a separate method of distribution, and that consists of the file containing the Trojan being transmitted to potential victims using methods like e-mail, instant messaging, IRC, ICQ etc. when the potential victim opens the file, the Trojan is installed. Trojans can also be staged on download sites and disguised as utility programs, games, etc. and the victim is tricked into downloading them because they look like a useful program the victim might want to use[14].

- **Trojan Horse Virus** This is a hybrid between a Trojan and a virus. Most Trojan horse viruses infect like a Trojan in that they need to be run or executed by the victim, and then the virus behavior takes over and the Trojan horse virus automatically spreads itself to other systems. So, it spreads like a biological virus. Sometimes it sends itself to your address book etc[14].

- **Virus** A computer virus is a program, typically malicious, that reproduces by adding itself to other programs, including those belonging to the operating system. It cant run independently but needs a Host program to be run in order to activate it. The

source of the name is a reference to biological viruses that are not considered alive in the usual sense and cant reproduce independently, rather invades host cells and corrupts them into producing more viruses. Following are the type of viruses:[14]

  - Boot Sector Viruses
  - Master Boot Record Viruses
  - File Infector Viruses
  - Macro Viruses
  - Multi-Partite Virus

- **Worm** A worm is a program that can copy a fully functional version of itself to other machines across a network without intervention. It doesn't usually require another program in order to run, but worms can, and do, sometimes hide behind other programs. The source of the name is a derivative of "tapeworm" which is parasitic organism that lives inside a host and saps the host's resources in order to maintain itself[14].

Countermeasures against Software Vulnerabilities

There are several counter measures that may help ensure that unauthorized and possibly hostile virus or tojan software does not run on your systems. These countermeasures also limit the scope of vulnerability. Countermeasures include[8]:

- Run virus scan software on every organizational computer and update the virus scan database at least twice per week. Perform a full scan at least once per week.
- Keep software security patches updated- Get on computer security advisory mailing lists and update applicable software. With some systems such as

windows systems you can set up a server to automatically update systems on your network.

- Only allow approved software to be run on your computer systems so hostile torjan programs are not run. This may involve locking your users down so they cont install software on their computer systems.

- Limit services on all servers and workstations to the minimum required. Be sure the network administrator is aware of all operating services especially on all servers.

- Run vulnerability scanners both inside and outside your network to find computers with vulnerabilities so you will know which ones need patched. The cost of this should be weighed against the security need

Network Vulnerability

Network vulnerabilities are present in every system. Network technology advances so rapidly that it can be very difficult to overcome vulnerabilities altogether. Networks are vulnerable to slowdowns due to both internal and external factors. Internally, network can be affected by overextension and bottlenecks, external threats, DOS/DDOS attacks, and network data interception. Following are the type of vulnerabilities an administrator should take care of this:[11]

- Internal network vulnerabilities result overextension of bandwidth (user needs exceeding total resources) and bottlenecks(user needs exceeding resources in specific network sectors). These problems can be addressed by network management systems and utilities such as traceroute, which allow administrators to pinpoint the location of network slowdowns. Traffic can then be rerouted within the network architecture to increase speed and functionality.

- DOS and DDOS are external attacks as the result of one attack or a number of coordinated attacks, respectively. Designed to slow down or disable networks altogether, these attacks are among the most serious network performance in order to catch these threats as soon as possible. Many monitoring systems are configured to send alarms or alerts to administrators when such attacks occur, allowing for network access by intruders to be quickly terminated.

Data interception is another of the most common network vulnerabilities, for both LANs and WLANs. Hackers within range of a WLAN workstation can infiltrate a secure session, and monitor or change the network data for the purpose of accessing sensitive information or altering the operation of the network. User authentication systems are used to keep such interception from occurring. Firewalls can keep unauthorized users from accessing the network in the first place, while base station discovery scans allow for the rooting out of intruders on a given network.

How Network Vulnerability exploits

- Detecting Live system on target network.
- Discovering service running/listening on target systems.
- Understanding port scanning techniques.
- Identifying TCP and UDP services running on target network
- Discovering the Operating system.
- Understanding active and passive fingerprinting.
- Automated discovery tools

Most Vulnerable Ports

There are many of the ports remain open when a connection is established to internet. Even when the connection is not in use some of the ports remain open to the

attacker to exploit the system. These services on such port help the hacker/attacker to exploit the system, hacker came to know about the running services and open port during the scanning phase, that is done by the help of WAR-DIALERS.
A war dialer is a tool used to scan a large pool of telephone numbers to detect vulnerable modems to provide access to the system. Following are the list of most vulnerable ports[13]:

- 139 (SMB over NetBIOS)
- 80 (HTTP)
- 25 (SMTP)
- 23 (Telnet)
- 20 21 (FTP)

Port 139 vulnerabilities

NetBIOS applications employ NetBIOS mechanisms to locate resources, establish connections, send and receive data with an application peer, and terminate connections. Windows machines use NetBIOS services for authentication and service sharing. The Windows implementation of NetBIOS uses ports 135, 137, 138 and 139. A NetBIOS null session allows a machine, or a user of that machine, to gather information about another machine's users, shares, and services. Windows processes, for example, use null sessions to browse for network services and relay the findings to the system's user. In these service 'browsing' sessions, a username and password is not used. Null sessions can be considered a security risk because of the amount of information they can provide about a victim[13].

Port 80 vulnerabilities

Port 80 is used by the HTTP protocol, mainly in conjunction with web access.

Many different web server software packages are in use on the Internet, but in an environment where a very large percentage of the systems use the Windows operating system, it can be assumed that Internet Information Services (IIS) is the most widely used web server. Buffer overflow vulnerability within the Windows Indexing Service used by IIS allows remote attackers to run code with system privileges. A denial of service vulnerability (exists) that could enable an attacker to temporarily disrupt service on an IIS 5.0 web server[13].

Port 25 vulnerabilities

Port 25 is most widely used by the Simple Mail Transport Protocol (SMTP) for sending electronic mail. SMTP servers open and listen for incoming connections on port 25. Another SMTP server, or a personal eMail client, will connect to the server on its port 25 to transfer some eMail into it for subsequent forwarding toward its destination. The writable service vulnerability discovered allows the attacker to write arbitrary files to the SMTP server system by forging the receipt field with an absolute file path and file name. The SMTP server does not validate the information in this fieldreceipts[13].

Network Vulnerability Analysis

Vulnerability analysis, also known as vulnerability assessment, is a process that defines, identifies, and classifies the security holes (vulnerabilities) in a computer, network, or communications infrastructure. In addition, vulnerability analysis can forecast the effectiveness of proposed countermeasures and evaluate their actual effectiveness after they are put into use. Vulnerability analysis consists of several steps[12]:

- Defining and classifying network or system resources.
- Assigning relative levels of importance to the resources.

- Identifying potential threats to each resource.
- Developing a strategy to deal with the most serious potential problems first.
- Defining and implementing ways to minimize the consequences if an attack occurs.

If security holes are found as a result of vulnerability analysis, a vulnerability disclosure may be required. The person or organization that discovers the vulnerability, or a responsible industry body such as the Computer Emergency Readiness Team (CERT), may make the disclosure. If the vulnerability is not classified as a high level threat, the vendor may be given a certain amount of time to fix the problem before the vulnerability is disclosed publicly. The third stage of vulnerability analysis (identifying potential threats) is sometimes performed by a white hat using ethical hacking techniques. Using this method to assess vulnerabilities, security experts deliberately probe a network or system to discover its weaknesses. This process provides guidelines for the development of countermeasures to prevent a genuine attack.

The basic aim of vulnerability analysis is to identify scenarios that would lead to severe consequences for the society and the business community and can be considered to have some likelihood of being realized in the future. The basic goal of vulnerability analysis can be disaggregated and refined in many sub goals. An important part of the analysis is to identify critical points or regions in the road network where some kind of incident would lead to particularly severe consequences, and where the potential for such incidents in the future can't be considered negligible. Network vulnerability analysis gives the road authorities the ability to counteract discovered vulnerabilities before they are realized.

Network Auditing Tools

Security auditing tools are utilities that detect vulnerabilities in systems that could be exploited to compromise a system. The types of vulnerabilities detected vary with each system. LC4 is a tool created for the sole purpose of testing password strength. Other tools like the IIS Lockdown tool test a service for configuration errors and make reparations where necessary. Tools such as Nessus and the Microsoft Baseline Security analyzer check for exploits against a database of well known attacks. The port scanning tool Nmap aids in detecting unauthorized services. Using a combination of tools is the typical way to maximize the number of detected vulnerabilities in a networked environment. Nessus uses a user-configurable database of tests to drive vulnerability analyses. Comparison of NMAP and NESSUS[13].

NMAP

NMAP is a network auditing tool that scans network hosts for open ports. Port scans can determine if a host is offering errant services or failing to offer required services. Examples of errant service are an http daemon on a host not listed as a web server and a backdoor opened by a Trojan horse. Nmap can also determine the operating system running on a scanned host, and scan firewalls to determine the parts a firewall effectively filters. NMAP can scan network host using one of six methods: TCP connect() scan, TCP SYN scans, stealth FIN scans, XMAS tree scans, NULL scans, UDP scans, and ping scans.

The TCP connect() scan, the simplest port scanning method, attempts to create a TCP connection between the scanning client and the host. The auditor determines the TCP ports that Nmap will scan. Nmap allows any user to execute a TCP connect() scan, regardless of privilege. The TCP connect() scan is highly traceable by intrusion detection systems, and an unlikely choice of attackers [Fyodor].

The TCP SYN scan mimics the TCP connect() scan but does not complete the TCP connection. Nmap sends a SYN (synchronize) packet to the host and awaits a reply.

A RST (reset) packet from the host indicates there are no services available on the scanned TCP port. An ACK (acknowledgement) packet indicates a service is available on the scanned TCP port. When the host receives an ACK packet, Nmap sends a RST packet to destroy the pending connection. Nmap limits the use of TCP SYN scans to root users, since standard TCP stacks do not track reset connections. FIN scans, Xmas tree scans, and null scans use irregular TCP packets to discover open ports. The TCP header contains a 6-bit control block with six flags: URG (urgent data), ACK (positive acknowledgement), PSH (push data to receiver), RST (reset connection), SYN (synchronize), and FIN (finish transfer) . The Stealth FIN scan sends a FIN packet to the host and awaits a reply. If the connection does not exist (CLOSED) then a reset is sent in response to any incoming segment except another reset. In particular, SYNs addressed to a non-existent connection are rejected by this means1[13].

Ports that fail to reply with an RST packet are assumed to be open.

Xmas Tree scans and Null scans are variations of the Stealth FIN scan. The Xmas Tree scan sets the FIN, URG and PSH flags. The Null scan turns off all flags n the 6-bit control block. Stealth FIN, Xmas Tree and Null scans do not work against some operating systems because of TCP protocol modifications. In particular, the Microsoft Windows TCP protocol will not send an RST packet in response to these scans .

Nmap detects open UDP ports by sending a zero-byte UDP packet to every host UDP port. If the host replies the port is unreachable, the scanned port is closed. No response from the host indicates the UDP port is open. Using a ping scan, Nmap can determine the existence of systems at specified IP addresses. Nmap ping scans use standard ICMP echo packets and TCP ACK packets sent to port 80. This feature of Nmap is most useful when sweeping a network segment to determine the number

of hosts on a network segment. The TCP ACK packet discovers hosts behind a firewall that filters standard ICMP echo packets. TCP ACK packets will only be sent when a root user initiates a ping scan[13].

Nessus

Nessus is a UNIX-based security scanner that checks for well-known vulnerabilities on various platforms. The specific vulnerabilities that Nessus checks are determined as part of an initial configuration step that involves choosing the plug-ins of the associated vulnerability that the user wishes to discover. Tests for vulnerabilities can be created by anyone using the NASL scripting language. The tool's authors determine which plug-ins are distributed with the scanner for use by the Nessus community. Active support from the open source community has ensured a constantly increasing supply of new, downloadable, vulnerability tests[13].

Nessus emulates the first well-known security scanner, the Security Administrator Tool for Analyzing Networks (SATAN) . SATAN, like Nessus, attempts to break into a host to determine the level of host security. Unlike Nessus, SATAN cannot detect recently discovered vulnerabilities, and was therefore rejected for this study. The Nessus program uses two different executables to scan hosts and collect data. The 'server,' called the Nessus daemon (nessusd), is responsible for 'attacking' each host. The Nessus daemon must be executed on a UNIX-based machine. The 'client,' called nessus, collects the data from the attack. Currently, the client portion of the Nessus program is available for UNIX and Window.

The server portion of Nessus supports various options for multiple situations. Nessus offers multi-user support and the ability to grant different rights to each user. Multi-user support allows security administrators to grant rights to scan selected subsets of a network. Nessus can also use multithreading to conduct concurrent audits. The number of hosts that can effectively be audited simultaneously is de-

Figure 2.1 Nessus Scan [15]

termined by available network bandwidth and the processing speed of the Nessus daemon computer. The Nessus client collects and arranges data received from the vulnerability assessment. Nessus offers the following twenty-three rule sets for use in a security audit[13].

- **Trojan backdoors**: Trojan backdoor servers are often distributed through e-mail as attachments. The unsuspecting recipient executes the infected program expecting to view something interesting or amusing. Instead, the attachment installs the Trojan backdoor program onto the system. Scanning well-known Trojan ports detects the presence of a Trojan backdoor[13].

- **CGI script vulnerabilities**: A CGI script is a server-side executable. CGI scripts with programming errors are vulnerable to security flaws. Searching the UNIX

  cgi-bin directory detects possible flawed CGI scripts[13].

- **Cisco network device vulnerabilities**: Nessus, in most cases, uses SNMP to check the version of the system IOS and compare it to version numbers of the Cisco IOS with known vulnerabilities. Use of these plug-ins requires

knowledge of a Cisco device's read-only SNMP community string. This group of plug-ins was added to Nessus too late for inclusion in this study[13].

- **Default UNIX accounts**: This rule set tests certain well-known UNIX accounts for having a default password, or no password at all. This group of plug-ins was added to Nessus too late for inclusion in this study[13].

- **Denial of Service (DoS) attack**: This rule set checks for DoS susceptibility in applications, servers, switches and routers. A denial of service (DoS) attack is launched by issuing commands that crash the target by means of 'confusion' or buffer overflow. Nessus attempts to crash a host using a DoS attack to detect software and hardware vulnerabilities[13].

- **Finger daemon vulnerabilities**: The UNIX 'finger' command shows user information. Depending on the daemon's configuration, the user being 'fingered' can belong to a local or remote network. Finger daemons provide the names of active accounts, which is a security threat by itself. Some finger daemons have security flaws that offer private information about users, or allow attackers to gain system control[13].

- **Firewall configuration error**: This rule set attempts to bypass the firewall and alerts administrators of subsequent firewall configuration errors. Malicious parties can 'hop' a poorly configured firewall to access network resources[13].

- **FTP vulnerabilities**: FTP (File Transfer Protocol) servers distribute files to users with proper credentials. Some FTP servers exhibit flaws that allow attackers to retrieve arbitrary information or gain control of the system. Detection of FTP server vulnerabilities involves attempting well-known exploits[13].

- **UNIX shell exploits**: A UNIX shell accepts user commands for execution.

Shell access is normally reserved for authorized system users. Gaining shell access allows a user to execute hazardous commands, change a host's configuration and access possibly sensitive data. Not all possible commands and applications may be executed with just shell access, as root privileges may be necessary[13].

- **Remote UNIX root exploits**: This rule set tests whether outside parties can gain root access to a system, and with it, the ability to execute any command or application, or examine any data file on the exploited host[13].

- **General**: This rule set tests for programs and daemons that provide their name and version number. Attackers can use this information to determine if a host might exhibit well-known vulnerabilities. Nessus attempts to gather operating system and version information from all queried hosts[13].

- **Miscellaneous**: This rule set checks for blank or default passwords within daemons and hardware devices. These plug-ins can determine if an initial password was not changed, or left blank after the daemon or hardware device installation. When a password is blank or unchanged, an attacker can gain administrative access to the daemon or hardware device. The attacker can also change the password to block authorized users. Nessus will test certain hardware devices and daemons for default or blank passwords and alert the administrator if needed[13].

- **NIS (Network Information Service) vulnerabilities**: The NIS server gathers information about network services. Systems query the NIS to locate appropriate network services. NIS servers should not be accessible from outside of the network. An outsider can obtain a network layout by querying an easily

accessible NIS server. Network layout information will help an outsider launch an attack with a higher probability of success[13].

- **Peer-to-peer file sharing**: Some well-known peer-to-peer file sharing programs like Kazaa, Morpheus and Gnutella, if configured incorrectly, allow anyone to access a computer's file system.

- **Sensitive file retrieval vulnerabilities**: Password file retrieval allows an attacker to decipher username/password combinations for use in a consequent break-in. Nessus also uses this rule set's plug-ins to download any file with a known path name and file name[13].

- **Remote Procedure Call (RPC) vulnerabilities**: Although most remote procedure calls do not pose an immediate security threat, unused remote procedure calls should be disabled in case a vulnerability is discovered in the future[13].

- **SMTP server vulnerabilities**: An SMTP (Simple Mail Transport Protocol) server delivers e-mails to their destination. Some tests in this rule set attempt to crash the SMTP server. Other tests determine if outsiders can use the SMTP server to send or relay e-mail. Nessus detects and alerts administrators of these security holes and other undesirable SMTP server settings[13].

- **Simple Network Management Protocol (SNMP) information disclosure vulnerabilities**: The Simple Network Management Protocol allows administrators to gather user, service and executing process information. Nessus issues SNMP queries to obtain private system or network information[13].

- **Useless daemons and services**: Some useless daemons and services can be exploited to gain user and network information. Other useless daemons and

services can be exploited to commandeer network bandwidth[13].

- **Windows vulnerabilities, including absent hotfix**: As security holes are detected in Windows operating systems and applications, Microsoft offers patches, or hotfixes, to alleviate the security threat. Good network security practices involve actively patching vulnerable software. Nessus can detect vulnerabilities that can be patched by applying a hotfix issued by Microsoft[13].

- **Windows Access Control List (ACL) enumeration**: Determining a membership of a Windows system ACL can provide a starting point for launching attacks against privileged account, such as accounts belonging to Administrators or Domain Administrators[13].

2.1 Ant Colony Optimization

Ant Colony Optimization is an approach inspired by ants. When ants walk to and from food source, they deposited a chemical on the background which is called Pheromone. By this approach the effectively transport the food to nest. When ants walks and they want that other ant should know about this path and then deposits pheromone to that path. Pheromone is a type of tag to that path which has followed. This pheromone value decries with the time due to evaporation. By this evaporation and does not get confused between old and new path. This pheromone is also use for the optimization. If there is a number of paths to the same source of food. Then these path get optimized by the pheromone deposition. To understand the basic approach of ants, we take an example of two bridges When ant starts to walk and reach to a junction they have to choose a way. They probabilistically choose any one from that way, but this probability depends on the pheromone value of that path which indicates how much that path has been used. If the pheromone

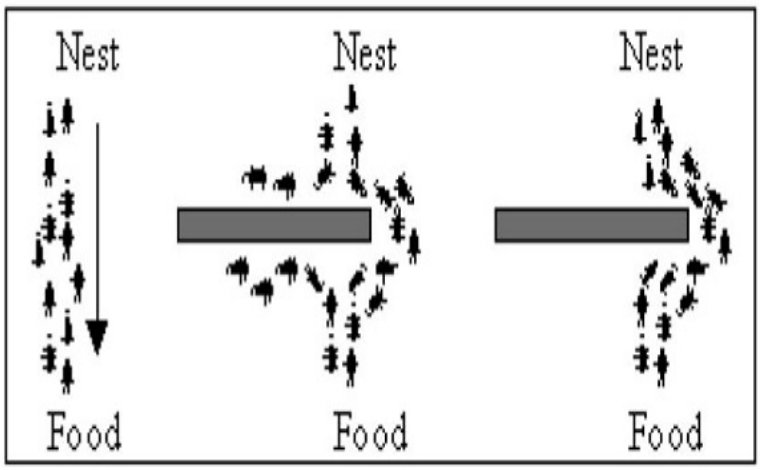

Figure 2.2 How ants find food from the nest.

value is at high any path then probability to choose that path is high. The basic idea is that an ant which at shorter path. take the less time to reach other side. So that pheromone value at that bridge updates frequently as compare to longer bridge. Higher value of pheromone attracts more number of ants. This processor leads to optimize path of ants.

In ant societies and more in general, in insect societies the activities of the individuals, as well as of the society as a whole, are not regulated by any explicit from of centralized control. Among the different works inspired by the ant colonies, the ant colony optimization meta heuristic (ACO) is probably the most successful and popular one. The ACO meta heuristic is a multi-agent framework for combinatorial optimization, whose main components are, a set of ant-like agents, the use of memory and of stochastic decisions, and strategies of collective and distributed learning. Each ant can smell a special signal called Pheromone. Ants utilize pheromone to communicate with each other. Figure 2.1 shows the behavior of real ants. In figure 2.1(a), the ants find food and move to direction of the food. They leave pheromone along the path to which they move. As shown in Figures 2.1(b) and (c), a barrier blocks stop the ant's direct path to the food, so the ants by pass the barrier by tak-

ing either the upper or lower path around the barrier. The ants moving along the upper path get back quicker than those moving along the lower path because that the upper path is shorter than the lower one. Therefore, the amount of pheromone, the following ants know which path is the shortest one, as shown in the Figure 2.1(d)[22]. Figure 3.1 (a) the ant find the food and move to food. (b) A barrier

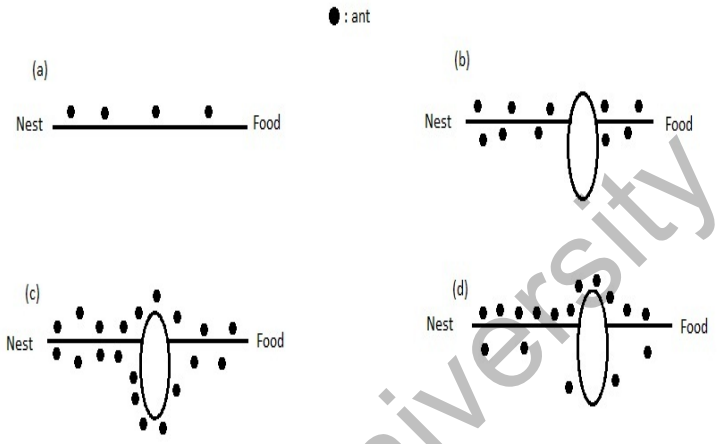

Figure 2.3 Ant Communication[22].

prevents the ants from moving. (c) the ants try to move around all paths and leave pheromone. (d) the following ants know which path is shorter by sensing the amount of pheromone on the paths.

know which path is shorter by sensing the amount of pheromone on the paths.

Step 1. All artificial ants construct random paths.

Step 2. By analyzing all the paths constructed in Step 1, the amount of pheromone that should be added on the paths can be determined.

Step 3. Update the amount of pheromone on all the paths.

Step 4. If the best solution in a iteration is not changed after some iterations. Then the best solution is the result. Otherwise iteration is repeated (go to step 1)[22].

Ant Colony optimization (ACO) algorithms are heuristic optimization methods that simulate the behavior of real ants. ACO are probabilistic search approaches which

are founded on the evolutionary process. The algorithm can be applied to solve combinatorial problems such as TSP. In 1992, Marco Dorigo first presented the algorithm in his Ph.D thesis. Dorigo, Di Caro and Gambarella have defined the Ant Colony Optimization meta heuristic in 1999.

They observed that real ants were able to select the shortest path between their nest and food resource, in the existence of alternate paths between the two. The search is made possible by an indirect communication known as stigmergy among the ants [13]. While traveling their way, ants deposit a chemical substance, called pheromones, on the ground. When they arrive at a decision point, they make a probabilistic choice, based on the intensity of pheromone they smell. This behavior has an autocatalytic effect because of the fact that choosing a path will increase the probability that the corresponding path will be chosen again by future ants. When they return back, the probability of choosing the same path is higher (due to the increase of pheromone). New pheromone will be released on the chosen path, which makes it more attractive for future ants. In short, all ants will select the shortest path. More the ants follow the path more the trail become attractive.

2.1.1 How Ant Communicate

Language is essential to effective government among social creatures. Without means of communication of some sort, it would be impossible for societies to hold together and to act together in those communal movements which are alike the evidence and the end of social organizations. The language of insects may be regarded as mimetic, when emotions are expressed by gestures or acts; pteratic, when by wing vibrations; spiracular, when made known by sounds issuing from the breathing tubes or spiracles; stridulatory, when conveyed by the friction of one organ against another;

and antennal, when the antennae, or "feelers," are the media of communication. In the real world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails[9].

If other ants find such a path, they are likely not to keep travelling at random, but to instead follow the trail, returning and reinforcing it if they eventually find food. Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets marched over faster, and thus the pheromone density remains high as it is laid on the path as fast as it can evaporate. Pheromone evaporation has also the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained. Thus, when one ant finds a good (i.e., short) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads all the ants following a single path. The idea of the ant colony algorithm is to mimic this behavior with "simulated ants" walking around the graph representing the problem to solve[10].

The original idea comes from observing the exploitation of food resources among ants, in which ants' individually limited cognitive abilities have collectively been able to find the shortest path between a food source and the nest. Ants use the environment as a medium of communication. They exchange information indirectly by depositing pheromones, all detailing the status of their "work". The information exchanged has a local scope, only an ant located where the pheromones were left has a notion of them[10].

Introduction to ACO Algorithms

ACO algorithm belongs to the class of Meta heuristics and therefore follows the latter goal. The central component of an ACO algorithm is parameterized probabilistic model, which is called pheromone model. The pheromone model consists of vector of model parameters T called pheromone trail parameters. The pheromone trail parameters $Ti\varepsilon T$, which are usually associated to computers solutions, have value ti called pheromone values. The pheromone model is used to probabilistically generate solutions to the problem under consideration by assembling them from a finite set of solution components. At runtime, ACO algorithms update the pheromone values using previously generated solutions. The update aims to concentrate the search in regions of the search space containing high quality solutions. In particular, the reinforcement of solution components depending on the solution quality is an important ingredient of ACO algorithm. It implicitly assumes that good solutions consist of good solution components. To learn which components contribute to good solutions can help assembling them into better solutions. ACO was applied to may CO problems. In general, the ACO approach attempts to solve an optimization problem by repeating the following steps:

- Candidate solutions are constructed using a pheromone model, that is, a parameterized probability distribution over the solution space.
- The candidate solutions are used to modify the pheromone values in a way that is deemed to bias future sampling toward high quality solutions.

ACO algorithms are stochastic search procedures. Their central component is the pheromone model, which is used to probabilistically sample search space. The pheromone model can be derived from a model of tackled CO problem [10].

Different types of ACO Algorithms

Several special cases of the ACO Meta heuristic have been proposed in the literature. Here we briefly overview, in the historical order in which they were introduced, the three most successful ones: ant system (Dorigo 1992, Dorigo et al. 1991, 1996), ant colony system (ACS) (Dorigo Gambardella 1997), and MAX-MIN ant system (MMAS) (Sttzle Hoos 2000).

- Elitist Ant System :The global best solution deposits pheromone on every iteration along with all the other ants.
- Max-Min Ant System (MMAS)
  - Added Maximum and Minimum pheromone amounts $[\tau max, \tau min]$.
  - Only global best or iteration best tour deposited pheromone.
  - All edges are initialized to τmax and reinitialized to τmax when nearing stagnation.
- pseudo-random rule. it has presented above.
- Rank-Based Ant System (ASrank) . All solutions are ranked according to their fitness. The amount of pheromone deposited is then weighted for each solution, such that the solutions with better fitness deposit more pheromone than the solutions with worse fitness.[10]

Ant System

Ant System is the first ACO algorithm proposed in 1991 by Dorigo. It encouraging initial results on TSP but inferior to state-of-the-art. Main role of ANT SYSTEM is"Proof Of Concept" and stimulation of further research on algorithmic.

Algorithm of ANT System[10]

Initilize Pheromones

While(termination condition not met) do

for i=1 to n-1 do

for k=1 to m do

ApplyProblisticActionChoiceRule(Mk,t,n)

End

End

GlobalPheromoneTrailUpdate

End

2.1.2 Ant Colony System

It was the first major improvement over the ANT System.

- Decision Rule- Pseudo random proportional rule.
- Local Pheromone Update.
- Best only offline Pheromone Update.

With probability Q0 of an ant k located at the city i chooses successor city j with minimal $\tau_{i,j}(t).[\eta_{i,j}]^{\beta}$(Exploitation)
with probability 1- Q0 an ant chooses k the successor city j according to action choice rule used in Ant System.
Global pheromone update rule
Only the best so far solution $\tau_(gb)$ deposits the pheromone after each iteration. $\tau_{i,j}(t)= (1-P).\tau_(i,j)(t-1) + P.\tau_(i_j)gb(t-1)$ Pheromone update only affects edges contained in $tau_(gb)$)[10].

2.2 Max-Min System

Extension of Ant System with stronger exploitation of best solutions and additional mechanism to avoid search stagnation. Only the iteration-best or best-so-far ant deposit pheromone[10].

$\tau_(i,j)(t+1) = (1-P).\tau_(i,j)(t) + \tau_(i,j)^{B}est$ Additional limits on the feasible pheromone trails

- For all $\tau_(i,j)(t)$ we have τmin ¡= $tau_(i,j)(t) <= \tau max$
- counteracts stagnation of search through aggressive pheromone update
- Heuristics for determining τmin and τmax
- Pheromone values are initialized to τmax stronger exploration at the start of the algorithm.

Pheromone trail re-initialization to increase exploration

Pheromone Update

Different ACO variants mainly differ in the update of the pheromone values they apply. In the following, we outline a general pheromone update rule in order to provide the basic idea. This pheromone update rule consists of two parts. First, a pheromone evaporation, which uniformly decreases all the pheromone values, is performed. From a practical point of view, pheromone evaporation is needed to avoid a too rapid convergence of the algorithm toward a sub-optimal region. It implements a useful form of forgetting, favoring the exploration of new areas in the search space.

Deterministic backward Ants and Pheromonoe Update

When forward ant reaches to destination then destination takes all the information from it and kills it. After that create backward ant and this ant follows the same

path followed by forward ant. The main job of the backward ant is to deposit pheromone value at links of the path for goodness of that link for that path. A large number of iteration of forward and backward ants give optimize path which have maximum value of pheromone. For the iteration calculation there is an experiment on double bridge. In this experiment two strategy uses for update of pheromone. One strategy is that simply update pheromone by same unit of value update at all time($\delta\tau(k = constant)$). But in second strategy, update pheromone value depends on length of the path($\delta\tau(k = 1/Lk)$.

Successful ACO Variants[24]

Table 2.1 Various ACO Algorithm variants

Problem name	Authors	Algorithm name	Year
Travelling Salesman	Dorigo, Maniezzo, Colomi	AS	1991
	Gamberdella , Dorigo	ANT-Q	1995
	Dorigo, Gamberdella	ACS, ACS 3 opt	1996
	Stutzle, Hoos	MMAS	1997
	Bullnheimer, Hartl Strauss	Asrank	1997
	Cordon, et al.	BWAS	2000
Quardratic Assignment	Maniezzo, Colomi Dorigo	AS-QAP	1994
	Gamberdella, Taillard, Dorigo	HAS-QAP	1997
	Stutzle, Hoos	MMAS-QAP	1998
	Maniezzo	ANTS-QAP	1999
	Maniezzo, Colomi	AS-QAP	1994
Scheduling Problems	Colomi, Dorigo, Maniezzo	AS-JSP	1997
	Stutzle	AS-SMTTP	1999
	Barker et al	ACS-SMTTP	1999
	Den Besten, Stutzle, Dorigo	ACS-SMTWTP	2000
	Merkle, Middenderf, Schmeck	ACO-RCPS	1997
Vehicle Routing	Bullnheimer, Hartl, Strauss	AS-VRP	1999
	Gamberdella, Taillard, Agazzi	HAS-VRP	1999
Connection-Oriented network routing	Schoonderwood et al.	ABC	1996
	White, Pagurek, Oppacher	ASGA	1998
	Di Caro, Dorigo	AntNet-FS	1998
	Bonabeau et al.	ABC-smart ant	1998
Connection-Less network routing	Di Caro, Dorigo	AntNet, AntNet-FA	1997
	Subramanian, Druschel, Che	Regualar Ants	1997
	Heusse et al.	CAF	1998
	Van der put, Rethkrantz	ABC-backward	1998
Graph Coloring	Gamberdella, Dorigo	HAS-SOP	1997
Frequency Assignment	Maniezzo, Carbonaro	ANTS-FAP	1998

Chapter 3

Problem Statement

100% testing is not possible for any software product, due to competitor pressure or time to market; companies launch the software earlier than they should have done so. This problem is core at network security paradigm. The bugs thus induced become pertinent and create weaknesses in the software. This vulnerable software is holly grail for hacking community. In order to detect these Network Vulnerabilities before they become threat, a robust framework needs to be designed and developed. This thesis is an effort to accomplish this task by using Ant Colony Optimization method of Network Vulnerability detection.

Objectives

- To study and explore various network vulnerabilities and their respective exploits.
- Design and Develop Network Vulnerability detection framework using ACO.
- Demonstrate the use of framework using isolated university network.

Chapter 4

Implementation and Results

Operating System Fingerprinting

Network scanning, and particularly remote OS/application detection, is generally the first step in mapping out a network; whether for penetration testing or simply maintaining a network device inventory. Remote active operating system fingerprinting is the process of determining the identity of a remote host's operating system. This is done by actively sending packets to the remote host and analyzing the responses. Tools like Nmap and Xprobe2 take the responses and form a fingerprint that can be queried against a signature database of known operating systems. Learning which operating system is running on a remote host can be very valuable for both pentesters and black-hats. It's valuable because when vulnerabilities are found they are normally dependent on the OS version. Determining the OS on the remote host was done by a technique known as "banner grabbing". Banner grabbing consists of either looking at the banner displayed when trying to connect to a service like ftp or by downloading a binary file like /bin/ls to determine what architecture it was built for.

OS Fingerprinting Through NMAP

Nmap is a network exploration tool and security scanner. It is designed to allow users to scan networks to determine which hosts are up and what services they offer. Nmap supports a number of scanning techniques that use the following protocols: TCP, ICMP, UDP, and IP. Nmap also includes features like remote OS detection, parallel scanning, port filtering detection, timing options, and flexible target and port specification.

Before Nmap runs it's OS detection method it runs a port scan against the target machine. It performs a port scan so it can find some open and closed ports on the target machine. Nmap works best when it finds at least one open TCP port, one closed TCP port, and one closed UDP port. Nmap works by conducting a set of tests against the target machine to try to determine what OS it is running. nmap is an active scanner and arguably one of the most popular network scanners in use today. nmap was introduced to the Internet community by its original author, Fyodor, in 1997. The primary purpose of nmap is to rapidly scan large segments of a network for devices that are active, with the ability to report which ports are open on those devices. "nmap OS fingerprinting works by sending up to 15 TCP, UDP, andICMP and probes to known open and closed ports of the target machine."

The following output of nmap show the OS fingerprinting on the remote computer with ip 192.168.1.2. First it tells us about the DNS server. Then it tells us about the ports which are open and which ports are closed. In the above output port no. 991 is closed. And port number 135,139 445,49152,49153,49154,49155,49156,49175 are open. And port no. 139 is open which is the most vulnerable port on the host

```
File  Edit  View  Terminal  Help
root@vandana-laptop:~# nmap -O 192.168.1.2

Starting Nmap 5.00 ( http://nmap.org ) at 2010-07-09 16:26 IST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
 Try using --system-dns or specify valid servers with --dns-servers
Interesting ports on 192.168.1.2:
Not shown: 991 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49156/tcp open  unknown
49175/tcp open  unknown
MAC Address: 00:16:36:30:63:0F (Quanta Computer)
Device type: general purpose
Running: Microsoft Windows Vista|2008|7
OS details: Microsoft Windows Vista SP0 or SP1, Server 2008, or Windows 7 Ultima
te (build 7000)
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at http://nmap.org/s
ubmit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.90 seconds
root@vandana-laptop:~#
```

Figure 4.1 Banner Grabbing Process

machine. It is using the netbios service. After the port scanning it tells us about the MAC address of the machine, type of device, and then it is displaying which Operating System is running on this remote machine. After this it tells us about the Network distance from source machine to destination machine, which is 1 hop away. This was the OS fingerprinting thorough NMAP(Network Mapper).

Advantages of NMAP

The Nmap tool offers many advantages over other remote active OS fingerprinting tools. Here are some reasons users might want to utilize this tool for its OS detection method:

- "Half-open" scanning2 is supported. The TCP SYN scan is one of these scan types. The advantage to this scan type is that it doesn't complete the TCP three-way handshake which means that it will often not be logged by basic intrusion detection systems.

- The tests that Nmap performs include multiple flags and protocols so that it can still make an OS guess even if filtering devices block some of the tests. Some tests might not work because filtering devices might block protocols like

ICMP from entering their network.

- There is a large OS signature database. The latest Nmap (3.27) recognizes 867 fingerprints while the latest Xprobe2 (0.1) recognizes 53.
- There is a lot of support for network devices like routers, firewalls, switches, and printers.
- Strict signature matching is utilized by default, while fuzzy matching can be enabled by specifying the undocumented –fuzzy option.
- It has built-in learning functions.

Defences

There are some defensive measures to protect against the OS detection method used by Nmap. Among the defenses are:

- In a normal network environment systems should sit behind some type of firewall4. Machines that are viewable from the Internet should only keep the needed ports open While the rest of the ports should be filtered by the firewall. This is a good defense since Nmap works best when it finds at least one open TCP port, one closed TCP port, and one Closed UDP port. If all the needed ports aren't found then Nmap's accuracy drops off.
- Users can change characteristics of the machines TCP/IP stack. This can also be accomplished via kernel patches.
- Network intrusion detection systems (IDS) can be used, if configured correctly, to detect the OS detection method used by Nmap because of the utilization of malformed packets.

Common Vulnerability Exposure

An information security "vulnerability" is a mistake in software that can be directly used by a hacker to gain access to a system or network. CVE considers a mistake a vulnerability if it allows an attacker to use it to violate a reasonable security policy for that system (this excludes excluding entirely "open" security policies in which all users are trusted, or where there is no consideration of risk to the system). For CVE, a vulnerability is a state in a computing system (or set of systems) that either:

- Allows an attacker to execute commands as another user
- Allows an attacker to access data that is contrary to the specified access restrictions for that data
- Allows an attacker to pose as another entity
- Allows an attacker to conduct a denial of service

Examples of vulnerabilities include:

- Phf (remote command execution as user "nobody")
- world-writeable password file (modification of system-critical data)
- Default password (remote command execution or other access)
- Denial of service problems that allow an attacker to cause a Blue Screen of Death
- smurf (denial of service by flooding a network)

Exposure

An exposure describes a state in a computing system that is not a vulnerability, but either

- Allows an attacker to conduct information gathering activities.
- Allows an attacker to hide activities.
- Includes a capability that behaves as expected, but can be easily compromised
- Is a primary point of entry that an attacker may attempt to use to gain access to the systm or data.
- Is considered a problem according to some reasonable security policy.

Examples of exposures

- Running services such as finger(useful for information gathering, though it works as advertised)
- Inappropriate setting for windows NT auditing policies.
- Running services that are common attack points
- Use of applications or services that can be successfully attacked by brute force methods.

Common Vulnerabilities and Exposures (CVE) is a dictionary of standard terms related to security threats. These threats fall into two categories, known as vulnerabilities and exposures. A Vulnerability is a fact about a computer, server or network that presents a definite, identifiable security risk in a certain context. An exposure is a security-related situation, event or fact that may be considered vulnerability by some people but not by others. CVE was developed and is maintained by the MITRE Corporation to facilitate the sharing of data among diverse security interests. It can simplify the process of searching for information in security-related databases and on the Internet. The dictionary is the product of collaboration among

experts and representatives from security-related organizations worldwide. Items in CVE are given names according to the year of their formal inclusion and the order in which they were added to the list in that year. According to the MITRE Corporation, the content of CVE should not depend on the perspective of the individual user. Any CVE entry that can be considered a vulnerability from all perspectives is known as a universal vulnerability. All other entries are categorized as exposures. An unpatched, previously exploited security loophole in an OS would constitute a universal vulnerability according to the CVE standard.

Banner Grabbing with NetCat

Netcat is known as the Swiss-knife of the hacker or the security-conscious internet user. However, alot of people don't know some of the most precious uses of this great tool. It can be used for banner grabbing, it can be used for port scanning as well as the better known uses of this tool : Port listening, and backdoor properties.

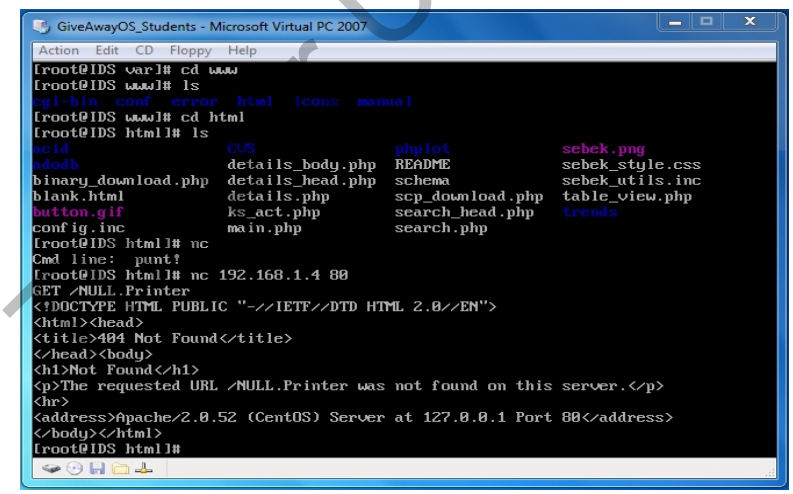

Figure 4.2 Banner Grabbing with NetCat

NetCat shows that the machine is running CentOS and Apache webserver with version 2.0.52 at port no 80. And this was done on the 127.0.0.1 ip which is loopback address.

Banner Grabbing with TELNET

Banner grabbing can also be done with the TENLET. It will tell you about the web server which is running on the remote machine and what the version of that web server is and it was found that there was no printer attached to the remote machine. Following is the syntax for implementing TELNET banner grabbing.

C:/192.168.1.2 80

Then type "get /" it will grab the banner of the remote machine.

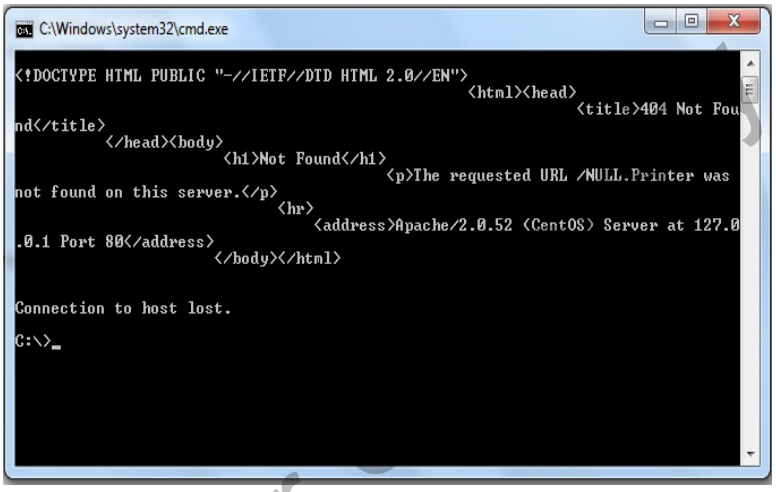

Figure 4.3 Banner Grabbing with TELNET

Fuzzing

Fuzzing is the art of automatic bug finding. This is done by providing an application with semi- valid input. The input should in most cases be good enough so applications will assume it's valid input, but at the same time be broken enough so that parsing done on this input will fail. Such failing can lead to unexpected results such as crashes, information leaks, delays, etc. It can be seen as part of quality assurance, although only with negative test cases. Fuzzing is mostly used to uncover security bugs, however, it can often also be used to spot bugs that aren't security critical but which can non-the-less improve robustness. When building a fuzzing tool there are 2 common approaches. The first one is to randomly send some kind

of data in an endless loop. this random fuzzing has the potential to uncover a lot of bugs but often misses quite a few because an application parsing the data might consider it to be invalid before it reached a faulty piece of code. In most cases this can be worked around by implementing adleast some intelligence into these kind of fuzzing tools. The second one is where it has been carefully studied what will likely cause problems and iterate over all possible combinations thereof, it comes close to fault injection. This kind of fuzzing is usually finite. One problem here is that it is almost always impossible to generate all possible combinations that will trigger a bug and often some bugs get missed. In reality random fuzzing usually finds the first couple of bugs faster. The second type of fuzzing is often more complete[21].

Nessus Nessus was created to be a free, powerful, remote security scanner. It is one of the top-rated security software products, and is endorsed by professional information security organizations such as the SANS Institute. The "Nessus" security scanner is a software which will audit remotely a given network and determine whether someone (or something - like a worm) may break into it, or misuse it in some way.

- Nessus can Perform over 900 security checks.
- Accept new plug-ins and patches to expand to new checks and security threats.
- List security concerns as well as recommend courses of action to correct them.

Nessus was created as an open source project typically for use on Unix, Linux, BSD, and other similar systems. Nessus is a client/server application - a client can connect to a remote server and run a scan remotely.

Operation Mode of Nessus

Nessus can operate in two modes, a banner check mode, in which only signatures are used to determine whether a host application is vulnerable and destructive mode,

in which an exploit itself is used to test for vulnerability. The advantage to banner check mode is that it does not compromise a system to determine vulnerability. However, banner checks are subject to both false positives (system is determined to be exploitable when it is not) and false negatives (system is determined to be not exploitable when it is). In fact, some applications and network appliances even manipulate banners to disguise potentially vulnerable services. Destructive mode ensures an accurate test result, but with the side effect of accomplishing whatever malevolent actions are contained in the exploit. Figure 4.4 shows the Architecture of Nessus. Nessus is used primarily with Unix-based systems. For Windows, Microsoft provides its own free comprehensive security scanner, the Baseline Security Analyzer. Microsoft is working to incorporate security scans for all of its major applications into this single tool

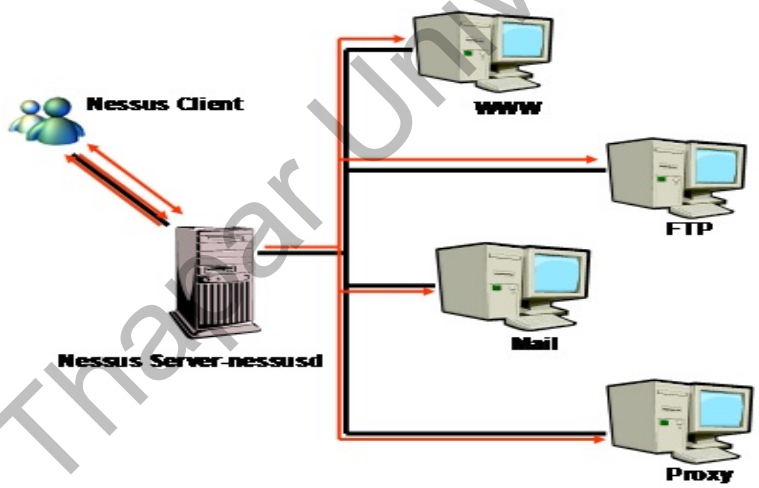

Figure 4.4 Architecture of Nessus

NESSUS script for Vulnerability Scanning

A penetration-test runs actual exploits on the identified machine and clarifies wheather is safe from a hacker attack. If a pen-test fails then it is certain that any internal or external entity can exploit your computer resources. These exploits

grow by leaps and bounds day-to-day and thus the number of pen tests also increases accordingly. Someone has to keep writing these newer pen tests. The single largest used penetration-testing tool in the world today is Nessus, which is available for the operating systems Linux/Unix and Windows. Nessus has over 8000 pen tests, with fresh one being written every day. All the tests are coded not in c or perl but with NASL- A scripting language solely used for writing these test. The software comes free for the Linux operating system whereas for windows, there is a cost attaced to it.

Script for Banner Grabbing in NASL

Following is script for Grabbing Banner with NASL:

include("$http\_func.inc$");

$sock = http\_open\_socket(80)$;

$req = string("GET/HTTP/1.0", "Accept : */* /r/n", "/r/n")$;

$send(socket : sock, data : req)$;

$r = recv(socket : sock, length : 4096)$;

$display(r, "/n")$;

$http\_close\_socket(sock)$;

HTTP/1.1 200 OK

Date: Sat, 10 Jul 2010 07:12:22 GMT

Server: Apache/1.3.17(Win32)

Content-Location: index.html.en

Vary: negotiate, accept-language,accept-charset

TCN: choice

Last-Modified: Fri, Jan 2010 08:12:44 GMT

ETag:"0-54a-3a67f64c;41e6035c"

Accept-Ranges: bytes

Content-Length: 1354

Connection: close

Content-Type: text/html

Content-Language: en

Expires: Mon, 04 Sep 2010 04:56:14 GMT

$<!DOCTYPE\ HTML\ PUBLIC"\ -\ //W3C//DTD\ HTML\ >$

3.2 $Final//EN" >$

$< HTML >$

$< HEAD >$

$< TITLE > Test Page for Apache Installation < /TITLE >$

$< /HEAD >$

Webserver Fingerprinting with NASL

include("http_func.inc");

sock=open_sock_tcp(80);

req=string("GET / HTTP/1.0 ° ","Accept: */* ° "," ° ");

send(socket:sock,data:req);

r=revc(socket:sock, length:4096);

if("Server: Apache"><r)

display("Apache Server running on host");

else if("Server: Microsoft-IIS"><r)

display("IIS Server running on host");

http_close_socket(sock);

Java based Ant Colony Optimization for Network Vulnerability Detec-

tion

WHILE termination conditions not met DO

PerformActivities

ACO_NVD()

PheromoneUpdate()

ScheduledActions()

END PerformActivities

ENDWHILE

ACO_NVD():

This method builds a solution to the problem by detecting vulnerability moving from node to node and constructing graph G. Ants move by applying a stochastic local decision policy that makes use of the pheromone values (NVD score: Candidate or Non Candidate) on running apps. While moving, the ant keeps in memory the partial solution it has built in terms of the vulnerability score it was walking on the construction graph.

PheromoneUpdate():

When adding a component to the current partial solution, an ant can update the values of the pheromone trails that were used for this construction step. This kind of pheromone update is called online step-by-step pheromone update. Once an ant has built a solution, it can retrace the same path backward and update the pheromone trails of the used apps according to the quality of the solution it has built. This is called online delayed pheromone update. Another important concept in Ant Colony Optimization is pheromone evaporation. Pheromone evaporation is the process by means of which the pheromone trail intensity on the apps decreases over time. From

a practical point of view, pheromone evaporation is needed to avoid a too rapid convergence of the algorithm toward a sub-optimal region. It implements a useful form of forgetting, favoring the exploration of new areas in the search space. Each attack scenario is depicted by an attack path which is essentially a series of exploits with a severity score that presents a comparative desirability of a particular network service. In an attack graph with a large number of attack paths, it may not be feasible for the administrator to plug all the vulnerabilities. Given an attack graph and the severity scores of the exploits, an optimal attack path is detected using customized ACO algorithms.

ScheduledActions():

Scheduled actions can be used to implement centralized actions which cannot be performed by single ants. Examples are the use of a local search of a vulnerability applied to the solutions built by the ants, or the collection of global information that can be used to decide whether it is useful or not to deposit additional pheromone to bias the search process from an non-local perspective.

As a practical example, the scheduler can observe the vulnerability found by each ant in the colony and choose to deposit extra pheromone on the components used by the ant that built the best solution. Pheromone updates performed by this method are called offline pheromone updates.

Java Nessus pulgin code is appended at Appendix 1

Following nessus script is fabricated to create new packets and send over the network using

send_packet() function. $Ip = forge\_ip\_packet(ip\_hl : 5, ip\_v : 4, ip\_tos : 0, ip\_len : 20, ip\_id : 12, ip\_off : 0, ip\_ttl : 255, ip\_p : 2, ip\_src : 172.31.9.15)$;

ACO_NVD()

Display(this_host(),"");

Send_packet(ip,pcap_active: FALSE);

172.31.9.91

The ip packet can be created using the function forge_ip_packet. This function takes up a large number of parameters. The first four bits is the version of ip used, 4 and set this value as the ip_c parameter. The next four bits are the length of the ip header and in this case as nothing is added to ip, it is 5. The length of the ip header can vary minimum 20 to maximum 60 as four bits hold a number from 0 to 15. The parameter name is ip_hl. Then we have the type of service which signifies the importance of packets to the routers. Unfortunately most routers ignore this field called ip_tos. Then there are two bytes that give the total length of Most of the time this field is ignored. Packet has an id of 12.

The maximum length of an ip packet is 65536 but Ethernet networks can't carry a packet larger that 1500 bytes. Thus if a packet is 3000 bytes large it has to be send as two packets or fragments. The field ip_off talks about the fragmentation. 0 refers to no fragments. Then is a field ip_ttl or time to live that decides on the number of routers our packet can cross before it is to be killed or dropped. We choose the maximum value of 255 but even 20 is too large.packet. The field ip_len is set to 20 as our packet is only an ip packet. The ip_id field holds the id of the packet.

The job of ip is simply to deliver a packet to the destination. IP never travels alone, there is some other protocol like TCP or ICMP or UDP following. The field ip_p is the protocol following ip for which we have given some non existent protocol

2. Then we have 2 bytes of the ip address where forging ip 172.31.9.15 is given, even though our ip address is 172.31.9.91. The field ip_src takes the ip address in a dotted decimal notation. The last four bytes are destination ip address. The field ip _ttl is an optional field. Which means that it can forge all the bytes of an ip packet but need to keep the -t and the ip_ttl fields the same.

The function this_host return our ip address. Thereafter, call to a function send_packet is processed with the ip packet to send across a packet. The experimentation was done on an isolated virtual network of 40 heterogenous nodes, created by "Honeyd". Network were seprated with the help of software router "Zebra".

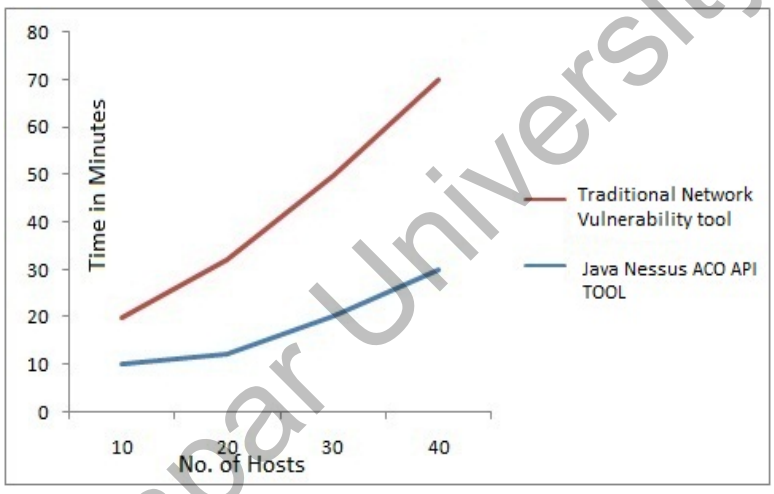

Figure 4.5 Comparision of Nessus ACO API with other

Graph showing the comparison of Java Nessus ACO API with the network vulnerability tool. It takes much less time in comparison with other algorithm. thus, validating the research work.

Chapter 5

Conclusion and Future Scope

Network Vulnerabilities are most critical for any network. This thesis work investigated Network Vulnerability detection process and its current status. Use of Banner grabbing, OS fingerprinting and Vulnerability detection has been successfully demonstrated. Nessus_JAVA and ACO integration has been successfully implemented and demonstrated, on isolated university Network. The results obtain show Network Vulnerability detection using traditional methods is comparatively slow against the result obtained by ACO implementation. **Future Work** Ant Colony Optimization is a vast area of research. In future, work can be extended to include real network, augmentation based on Intrusion detection can also be applied.

References

[1] Introduction to Network Security, Dr. Rahul Banerjee, BITS-Pilani, India www.discovery.bits-pilani.ac.in/rahul/CompNet/index.htm

[2] http://www.cert.org/tech_tips/home_networks.html

[3] A Brief History of Network Security and the Need for Adherence to the Software Process Model, by Paul Innella, www.tdisecurity.com/resources/assets/NetSec.pdf

[4] Network Security fundamentals, By Gert De Laet, Gert Schauwers, Cisco press.

[5] http://technet.microsoft.com/en-us/library/cc959354.aspx

[6]] Network Attack and Defence, By Roger Needham and Butler Lamson. www.cl.cam.ac.uk/ rja14/Papers/SE-18.pdf

[8] Efficient countermeasures for software vulnerabilities due to memory management errors,Prof. Dr. ir. W. JOOSEN, Prof. Dr. ir. F. PIESSENS. [7] Computer Vulnerabilities, Written by Eric Knight, C.I.S.S.P. Original Publication: March 6, 2000. www.ussrback.com/docs/papers/general/compvuln$_d raft.pdf$

[9] http://www.antcolonies.net/howantscommunicate.html

[10] http://en.wikipedia.org/wiki/Ant_colony_optimization

[11] http://www.javvin.com/etraffic/network-vulnerabilities.html

[12] http://searchmidmarketsecurity.techtarget.com/sDefinition
/0,,sid198_gci1176511,00.html

[13] A Vulnerability Assessment of the East Tennessee State University Administrative Computer Network, Dr. Phillip E. Pfeiffer, IV, chair Dr. Gene Bailey Dr. Qing Yuan.

[14] http://www.infosectoday.com/Articles/Exploiting
_Software_Vulnerabilities.htm

[15] Vulnerabilities threats and exploits, sabyasachi chakrabarty, anil sagar www.cert-in.org.in/knowledgebase/presentation/threatsandexploits.pdf

[16] Using Ant Colony Optimization to Modeling the Network Vulnerability Detection and Restoration System, Xie Hui 1,Wu Min 2,Zhang Zhi-ming.

[17] M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," Artificial life, Vol. 5, No.3, pp. 137-172, 1999.

[18] M.Dorigo and L. M. Gamardella, "Ant colonies for the traveling salesman problem," BioSystems, No. 43, pp. 73-81, 1997

[19] Ant colony optimization theory: A survey, Marco Dorigo, Christian Blum www.portal.acm.org/citation.cfm?id=1125369

[20] Ant Colony Optimization and its Application to Adaptive Routing in Telecom-

munication Networks, Gianni Di Caro
www.idsia.ch/ gianni/Papers/thesis-abstract.pdf

[21] Fuzzing,Breaking software in an automated fashion, Ilja van Sprundel, December 8, 2005 www.events.ccc.de/congress/2005/fahrplan/.../582-paper$_f$*uzzing.pdf*

[22] Marco Dorigo IRIDIA, Universit Libre de Bruxelles, Avenue Franklin Roosevelt 50, CP 194/6, 1050 Bruxelles, Belgium.

[23] An Ant Colony Optimization Algorithm for Network Vulnerability Analysis,M. Abadi and S. Jalili,Iranian Journal of Electrical Electronic Engineering, Vol. 2, Nos. 3 4, July 2006

[24] Ant Colony Optimization, Ahmad Elshamli, Daniel Asmar, Fadi Elmasri. www.scribd.com/doc/22599034/Ant-Colony

Publications

"Network Vulnerability detection using Ant Colony Optimization", Yogesh Kumar, Maninder Singh, "International Journal of Network Security", Communicated June, 2010.

Appdendix

```
package com.nessus.clientlib;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.net.SocketTimeoutException;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;
public class NessusClientAPI
private Map¡String, String¿ mServerPrefs = null;
private NessusSocket mCtrlSocket = null;
private String mHostToScan = null;
private Set¡NessusProgress¿ mNotifiers = new HashSet¡NessusProgress¿();
private NessusPluginContainer mPlugins = null;
private NessusClientPrefs mClientPrefs = null;
private String mServerName = null;
private int mPort = -1;
private String mUserName = null;
```

```
private String mPassword = null;
private int mEncType = -1;
private int mAuthType = -1;
private static final int timeoutMs = 2000;
private static final String NTP_DELIMITER = " < | > ";
/ * *
* Sendlinetoserver
* @paramdata
* @throwsNessusException
* / private void sendLn(String data) throws IOException
send(data+"");

/**
* Send string to server
* @param data
* @throws NessusException
*/
private void send(String data) throws IOException
send(data.toCharArray(), data.length());

/**
* Send bytes to server
* @param data
* @param size
* @throws NessusException
*/
private void send(char data[], int size) throws IOException
try
mCtrlSocket.send(data, size);
```

```
catch (IOException e)
disconnect();
throw e;
/**
* Receive string from server
* @param maxSize
* @return
* @throws NessusException
*/
private String read(int maxSize) throws IOException
char[] buffer = new char[maxSize];
try
int size = mCtrlSocket.recv(buffer, maxSize);
if (size == 0)
return "";
return new String(buffer, 0, size);
catch (IOException e)
disconnect();
throw e;
/**
* Receivealinefromserver
* @return
* @throwsNessusException
*/
private String readLine() throws NessusException
try
return mCtrlSocket.recvLine();
catch (IOException e)
throw new NessusException(e.getMessage());
```

```
/**
* ConnecttotheNessusserver
* @paramserverName
* @paramport
* @paramuserName
* @parampassword
* @paramencType
* @paramauthType
* @throwsNessusException
*/
public boolean connect(String serverName, int port,
String userName, String password,
int encType, int authType)
throws NessusException
return connect(serverName, port, userName, password, encType, authType, false);

public boolean connect(String serverName, int port, String userName, String password, int encType, int authType,
boolean verifyOnly)
throws NessusException
mServerName = serverName;
mPort = port;
mUserName = userName;
mPassword = password;
mEncType = encType;
mAuthType = authType;

try  mCtrlSocket = new NessusSocket(mEncType, mAuthType);
if (Boolean.getBoolean("nessus.client.debug"))
```

mCtrlSocket.setDebugPrints(true);

mCtrlSocket.connect(mServerName, mPort, timeoutMs);

catch (UnknownHostException e)

throw new NessusException(e.getMessage());

catch (SocketTimeoutException e)

throw new NessusException(e.getMessage());

catch (IOException e)

throw new NessusException(e.getMessage());

String response = new String();

try

send(" $< NTP/1.2 >$ ");

$if(!(response = readLine()).equals(" < NTP/1.2 >"))$

$thrownewNessusException("Serverdoesn'tsupportNTP/1.2protocol.Connectionterminated.");$

// Send username and password response = read(256);

if $(response == null||!response.equals("User :"))$

throw new NessusException("Unexpected response while authenticating. Connection terminated.");

sendLn(mUserName);

response = read(256);

if $(response == null||!response.equals("Password :"))$

throw new NessusException("Unexpected response while authenticating. Connection terminated.");

if $(authType == NessusSocket.AUTH_CERT)$

sendLn("*****");

else

sendLn(mPassword);

if $(!(response = readLine()).equals("SERVER < | > PLUGIN_LIST < | >"))$

throw new NessusException("Invalid username or password supplied. Connection

terminated.”);
if ($verifyOnly$)
disconnect();
return true;

return readPlugins();
catch (IOException e)
disconnect();
throw new NessusException(e);
catch (NessusException e)
disconnect();
throw e;
public void disconnect()
if ((mCtrlSocket == null) —— (mCtrlSocket.isConnected() == false))
return;
try
mCtrlSocket.disconnect();
catch (IOException e)
// Do nothing;
/ * *
* $RegisteraProgressNotifiertothenotificationsmechanism$
* @$paramprogress$
* /
public void registerProgressNotifier(NessusProgress progress)
if (isNotifierRegistered(progress))
return;
mNotifiers.add(progress);
/ * *
* $UnregisteraProgressNotifierfromthenotificationsmechanism$

```
* @paramprogress
* /
public void unregisterProgressNotifier(NessusProgress progress)
if (isNotifierRegistered(progress))
mNotifiers.remove(progress);
/**
* Checksifagivenprogressnotifierisregistered
* @paramprogress
* @return
* /
private boolean isNotifierRegistered(NessusProgress progress)
return mNotifiers.contains(progress);
/**
* Readthepluginsfromtheservermessage
* @throwsNessusException
* /
private boolean readPlugins() throws NessusException
int id;
String name, type, copyright, description, summary, family;
mPlugins = new NessusPluginContainer();
// System.out.println("NessusClientAPI :: getPlugins :: Getting plugins");
int count = 0;
while (true)
String received;
if((received = readLine()).equals("< | > SERVER"))
break;
String parts[]=parseNtpMsg(received);
// This is a workaround for corrupt plugins that are split into several lines
while (parts.length < 7)
```

```
received += readLine();
parts = parseNtpMsg(received);

id=Integer.parseInt(parts[0]);
name=parts[1];
type=parts[2];
copyright=parts[3];
description=parts[4].replaceAll(";", "");
summary=parts[5];
family=parts[6];

    NessusPlugin temp=new NessusPlugin(id,
name,
type,
copyright,
description,
summary,
family,
true); // The user can set the plugin to not active later
mPlugins.addPlugin(temp);
count++;
for (NessusProgress progress : mNotifiers)
if (progress.pluginLoaded(count) == false)
return false;
// System.out.println("NessusClientAPI :: getPlugins :: " + new Inte-
ger(count).toString() + " Plugins loaded");
readServerPrefs();
return true;
/**
```

```
*
* @return
* /
public NessusPluginContainer getPlugins()
return mPlugins;
/ * *
* ParsesaNTPmessagefromtheserver
* @parammsg
* @return
* /
private String[] parseNtpMsg(String msg)
int pos=-5, num=0;

    while ((pos=msg.indexOf("< | >",pos+5)) != -1)
num++;
String[] res = new String[num+1];
res[0] = msg.substring(0,(pos=msg.indexOf("< | > ")));
res[num] = msg.substring(msg.lastIndexOf(" < | >")+5);

    for (int i=1; i¡num; i++)
res[i] = msg.substring(pos+5, (pos=msg.indexOf("< | >", pos+5)));
return res;
/ * *
* CreatesanewNTPmessage
* @paramparts
* @return
* /
private String makeNtpMsg(String[] parts)
return makeNtpMsg(parts, false);
```

/ * *

∗ $CreatesanewNTPmessage$

∗ @$paramparts$

∗ @$paramappendDelimiter$ ∗ @$return$

∗ /

private String makeNtpMsg(String[] parts, boolean appendDelimiter)

StringBuffer msg = new StringBuffer();

for (int i=0; i¡(parts.length-1); i++)

msg.append(parts[i]).append(" ").append(NTP$_D ELIMITER$).$append$("");

$msg.append(parts[parts.length - 1])$;

$if(appendDelimiter)$

$msg.append("").append(NTP_DELIMITER)$;

$returnmsg.toString()$;

/**

* Reads preferances from the server message

* @throws NessusException

*/

$privatevoidreadServerPrefs()throwsNessusExceptionmServerPrefs = newHashMap < String, S$

¡—¿"))

$thrownewNessusException("getServerPrefs : Unexpectedresponsefromserver :$

" + $received$);

$while(!(received = readLine()).equals(" < | > SERVER"))intpos; if(received.indexOf("[")! = -1received.indexOf(" : ")! = -1)//mustbeaplugi$

5));

$readServerRules()$;

/**

* Start a new attack on a given list of hosts

* @param host

* @throws NessusException

*/

$returnpi$;