Common Network Security Threats and Counter Measures

A. Mahmoud Haidar[1], **B. Nizar Al-Holou, Ph.D.**[2]
[1]Dialexa LLC, Dallas, Texas, U.S.A
[2]Electrical and Computer Engineering, University of Detroit Mercy, Detroit, Michigan, U.S.A

Abstract - *Despite the growing level of interest in the field of Network Security, there is still little knowledge about the actual issues involved in securing networks. The real dangers of cyber crime are of serious consequence. Individuals with sufficient technical knowledge of Information Technology (IT), networks, and networking devices can steal sensitive information and may exploit vulnerable network systems. In this paper we illustrate some of those threats and learn how to protect our network from attacks and exploits. For this purpose we also propose a set of developed theoretical and practical laboratory sessions that can serve as a complement to academic/professional introductory network security classes.*

Keywords: Network, Security, Labs, Software

1 Introduction

A computer crime is rarely detected by a victim, which makes it very hard to precisely determine the rate of its occurrence. The General Accounting Office reported approximately 160,000 successful attacks out of 250,000 attempts annually [1]. Moreover, the Defense Information Systems Agency found that 65% of attempted attacks were successful [2].

The cost of computer fraud and abuse in the US is over $3 billion each year, which is significant considering that over 90% of computer fraud cases are not reported. A study was conducted by BYTE magazine showed that 53% of its readers have experienced data losses that cost $14,000 on average. A survey of over 600 governmental agencies and companies in the United States and Canada revealed that around 63% reported that their computers were infected by at least one virus. In fact, there exist over 2,500 various viruses are spreading around global at any given instance of time. Moreover, the style and behavior of those viruses are evolving rapidly. That's why hackers with extensive experience are more able to automate the exploitation of networks and systems than any other day.

With the increasing security threats, protecting our networks and systems from unauthorized access and exploits has become an urgent necessity. Hence, arming Information Technology students and professionals with knowledge and experience necessary to identify and resolve these threats becomes imperative. For this reason, we propose in this paper an educational framework that employs freeware off the shelf tools to learn network security. In section 2, the methodologies to penetrate and exploit a network are presented, while the labs developed to learn those methods and how to address them are discussed in section 3.

2 Network Penetration and Exploitation Methodologies

Attacking a network is generally a two steps procedure. The first step is to penetrate the network and find a weakness you can take advantage of to gain access to the network. The second step would be attacking and exploiting computers in that network. For each step, there is a common set of tools and methodologies.

2.1 Network Penetration

2.1.1 Scanners

Scanners can be classified into different categories based on the software application they run, which are designed to either probe server-side or client-side. Examples include TCP/IP/UDP port Scanner, Shared Scanner on a network, and NetBIOS Scanner. These scanners are capable to scan standalone workstation, a group of computers that are connected to network, domains or sub-domains, providing detailed information regarding the scanned area such as open ports, active services, shared resources and the active directory information [5].

IP scanner is used mainly to identify if an IP address is active or not, the IP scanner should be able to scan range of IP addresses, C or B range or even sub range to give fast report for the active IP(s). The IP scanner is the first step in gathering the information a target network or system.

A port scanner, on the other hand, scans the host's Ports. It looks for open service ports on the target. Each port is associated with a service that may be exploitable or contain vulnerabilities. Port scanners can be used to scan specific ports or they can be used to scan every port on each host and is used as a next step after knowing if the system alive or not using the IP scanner.

2.1.2 Sniffers

Sniffers are programs that passively monitor and capture traffic. Almost any laptop or PC can be turned into a sniffer by installing sniffer software, much of which is freely available on the Internet. The system running the sniffer should have a network interface card that can be used in promiscuous mode. Promiscuous mode enables the sniffer to view but not respond to network traffic, thereby making the sniffer essentially invisible to the network.

Sniffers are very useful tools during penetration testing and network troubleshooting; we commonly use them to capture user names and passwords from FTP and telnet sessions. In addition, sniffers can be used to capture any network traffic that is not encrypted, such as e-mail, HTTP, and other clear text services. Sniffers are generally able to intercept network traffic only on their local network segment. For instance, if a sniffer is located on a shared network that uses hubs, it can view all traffic on the entire network. If a sniffer is located on a switched network (one that uses switches versus hubs), the sniffer can see only broadcast traffic and traffic directed to it. To sniff a switched network, the sniffer would have to be located on a switch port that mirrored the traffic to other ports. Emerging sniffers, such as Dsniff by Dug Song, can sniff switched networks. The thought that switched networks are safe from sniffers are no longer true. Hence, encrypting sensitive information is always recommended to eliminate the affect of sniffing. Data encryption will be discussed in more detail in later sections of this paper.

2.1.3 Denial of Service (DoS)

Denial-of-Service (DoS) attacks are well known attacks and have been the bane of the security professionals. The objective of a DoS attack is to exhaust the resources (memory, buffer space, or CPU) to make its service unavailable to legitimate users.

Denial-of-Service (DoS) attacks can be accomplished by two methods. The first and most commonly used method is to flood the target to exhaust its resources. The second method is to create multiple attacks to confuse and crash the target.

Denial-of-Service (DoS) attacks are considered one of the most marketed hacker attacks since its tools have been the destruction of many powerful security structures. The main objective of this kind of attacks is to prevent access to a service or a resource located on the server, and makes it unavailable by the end-user. This is often performed by using flooding techniques against the target server in order to exhaust its specific resources (CPU, memory or buffer) depending on the main service that it provides. Also several attacks aim to stop these services by sending confusing packets to the target, resulting in system crashes while processing these packets due to some bug in the target.

Avoiding penetration is accomplished using Intrusion Detection Systems (IDS) and Firewalls. The main function of IDS is similar to that of burglar alarms. While the firewall keeps its organization safe from any external spiteful attacks through the Internet [3], the IDS detects any illegal attempt to break in the firewall security or any plans to access the trusted side, and once one of the previous actions occurred, an alert is sent to the system administrator warning him/her of a breach security existence [4].

2.2 Network Attacks and Exploits

2.2.1 Viruses and Worms

Day after day, the reliance on computer applications and programs increases. But what most of us don't know, that programs by themselves often expose a security threat. The work done by a program is hidden from the user. We only know what input we gave to the program and the output displayed by the program. Hence, most of the programs are treated as a black box. A malicious code could be hidden in that black box causing an intentional harm to the computer or the network. The most common form of such codes are Viruses and Worms. The origin of the word "virus" is Latin; which means a poison. In biology, it is defined as an infectious agent that is unable to grow or reproduce outside a host cell. A computer virus, on the other hand, is a set of code instructions encapsulated within an executable file, made to cause damage on the host machine, in such a way it is executed when the host executable is executed. By June 2005, there had been 103,000 different computer viruses created. Viruses are distinguished either by their function or category/type. The function is the harm that the virus creates. Whereas the category/type defines the characteristics of the virus. The following defines some of the important computer viruses' categories [5]:

Polymorphic Viruses- Are viruses that can change themselves after every infection to avoid identification by virus scanners. They are considered by many the most dangerous type of viruses [6].

Stealth Viruses- Are viruses that hides the harm they have caused. This is mainly done by taking control over the system module responsible of reporting or detecting the harm caused by the virus. When the stealth virus takes over this module, it will report the correct information before infection and hides the damage done.

Fast and Slow Infectors- A slow infector virus is the traditional virus which infects the programs when they are created. As for the fast infector virus, it infects the programs when they run in the memory. The main reason behind creating a fast infecting virus, is to infect the anti virus when it runs in the memory so it would infect all the files being scanned.

Sparse Infectors- Is the virus that uses a certain technique to decrease the probability of its detection. For example, it might only infect files after being executed 12 times, infect files with a certain size range, etc..

Armored Viruses- This virus is made in a way that it is very hard for anti-virus engineers to reverse engineer it. Usually the reverse engineering of a virus is done by disassembling its code. For this purpose, virus writers write thousands and thousands of unnecessary assembly code to make the virus code look like a maze and confusing.

Virus Droppers- It is a regular program that doesn't cause any harm to your computer other than dropping or installing a virus.

Regardless of the virus category or type, every virus generally has four main parts as shown in *Figure 1*.

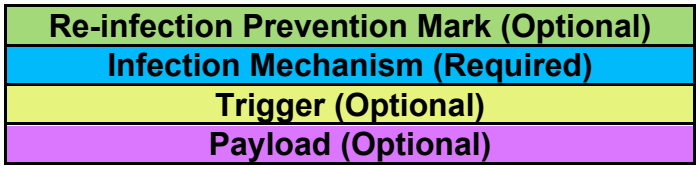

Figure 1: General Virus Stack

Re-infection Prevention Mark- It is a mark that the virus leaves on the infected file or system so it won't infect again. This part of the virus is optional.

Infection Mechanism- It is the method used by a virus to spread or infect other files on the computer. This part is required for any virus.

Trigger- Is an optional property. It defines the condition required to execute the virus's payload.

Payload- Is the damage that the virus causes to the infected computer besides spreading into other files.

A computer worm on the other hand, is a malicious program that copies itself over a network without the user consent, intervention, or knowledge [7]. It is very similar to computer viruses in design. However, it doesn't need a host file to replicate itself over the network. It takes advantage of the holes or weak spots in your system to travel across the network. It could send itself through your email, local network, or internet. When it is copied to another computer, it will copy itself to all the computers in that network and so forth. So, the worm can spread among computers exponentially. One the dangers is if the worm creates many copies of itself on the same computer, then each copy will send a copy of itself through the network. This will exploit the network and affect the bandwidth considerably.

A common method to avoid viruses and worms is to use an anti virus program. It is a software used to search for, identify, and recover a computer from hosted viruses. An anti virus software uses three main approaches to detect a virus:

Virus Dictionary- When a virus is first reported by a user to an anti virus software company, an anti-virus researcher will examine the virus, reverse engineer it, and create an anti virus to recover from it. The virus identified will be added to a virus dictionary, which contains records of all the viruses previously identified. Each record will contain the virus identifier, function, type, and actions need to be taken to recover from it. The dictionary can be later used by the anti virus software to detect viruses and recover from them. This is done by going through all the files in the computer and comparing each file by the records in the dictionary. If it matches a record, then the action defined will be executed.

Programs' Behavior Monitoring- Using this approach, the anti virus software will monitor the programs being executed in the computer at real time. If it detects a suspicious behavior, an alert will be popped to the user. A suspicious behavior could be a program trying to write data to another executable. This approach is used to detect newly created viruses.

Program Execution Emulation- An anti virus software may emulate a program execution to check if it is a virus or infected file. This is done by executing the beginning of the program to check if it will try to infect other files or execute a damaging payload.

2.2.2 Password Cracking

A password is a piece of information needed to access a private resource such as: emails, programs, bank accounts, computers, routers, house security systems, etc. Because of their importance and purpose, passwords should not be guessed, recovered, or bypassed by those who are not allowed to access the password-protected resource.

Since it must be known to the program or application it is protecting, a password is usually stored in a database or file to be later used for comparison with the password provided by the user to grant/revoke access. Securing this file is imperative. Moreover, precautionary measures should be taken to make it less probable for a person to crack passwords stored in that file, if the network or system was penetrated. One of the most commonly used approaches in this regard, is to prevent storing the passwords in clear text format. This can be done by applying an encryption function on the password. However, this encryption function must be a "one-way function"; which means if you have a password, you can get its encryption, but if you have the password encryption, you can't get the password. This type of functions is called a cryptographic hash function where its encrypted output is called a hashed password.

Most of the operating systems being used nowadays use hashed passwords. Linux-based systems, for example,

currently use MD5 hashed passwords. As for Windows systems, prior to Vista, uses LM hashed passwords [8]. Hashed passwords add more security to our system. However, attacks shown in the following list, can be used to recover a hashed password. The main idea behind those attacks is that they keep generating hashed passwords from a clear text password and they compare the result with the hased passwords in the password file. If, they match, then the clear text password is the password. You should note that unless you are trying to guess the password, the attacks presented will not work without knowing what algorithm was used to produce the hashed password.

Dictionary- This attack uses a dictionary file which contains most of the words that we use. Of course, the more words are there in the dictionary the higher the probability to crack a password. This attack is only effective if the password was a single or a combination of alphabetic words. Using this attack, you will have a fair chance to crack a password. A study that was conducted lately shows that 3.8% of the passwords are a single word passwords and 12% are a single word plus one numeric digit in which 66% of the time is "1" [9].

Brute Force- In this attack, the cracking program generates every possible combination of a password. Theoretically, this attack will always work and eventually crack any password. However, the larger the password the less practical the attack will be. For example, if we had a three digit alphanumeric password, we would need to generate 46,656 passwords (36 * 36 * 36) because in each digit we have 36 possibilities (26 letters and 10 numbers). Imagine now if the password used one of the other characters on the keyboard like ~!@#$%^&*()-_+=":;'?/.>,<{}|\[]`. That's another 32 possibilities for each digit, this will exponentially increase the number of needed passwords to generate. In real life, the average length of a password is between 8 -12. So, we need 9.77477912 × 1021 (6812) trials to be certain that we will crack a 12 character length password. This is true only if we know what is the length of the password. In case we don't know, we will need to generate all possible passwords with lengths between the minimum and maximum allowed password length. If we had a system that allows password lengths between 1 and 12, we will need 6812 + 6811 + 6810 + 689 ..... + 681 trials. It is obvious to see how unpractical it is to use the brute force attack when the password length is long or not known. A more practical approach is a hybrid attack between dictionary and brute force where we use a dictionary word and start generating a prefix and suffix. Of course, this will work only if the password had a dictionary password in part of it.

Pre-computation- It is similar to brute force attack except that it is done before attempting to crack a password. In this approach, we generate all possible passwords to create a hash lookup table containing the clear text passwords paired with their hash. The table will later be used for hash lookup. This will only take the searching time to crack the password. Although, it took the same time as brute force attack while generating the hash lookup table, the pre-computation attack is more effective if we want to crack many passwords on different systems.

2.2.3 Buffer Overflow and Shell Coding

Programming an application is not a straight forward task. Sometimes, an unintentional mistake, weak programming, or a bug in the program's code can be used as a back door to penetrate, take control over, and exploit the system. The most common example of such threat is Buffer Overflow.

Buffer overflow is the state where you write beyond the memory boundary specified for a buffer in your program. The overwritten memory could belong to a variable, return addresses, pointers, or other important data in the program [10]. Overwriting it, will lead to wrong results, errors, crashes, or exploits in the program. Buffer overflow is common on all platforms especially in applications written in C/C++ programming language. The reason behind that is C/C++ doesn't provide boundary check for allocated memory arrays. Take the following code as an example:

```
void BufferOverflowFunction(char *String)
{
//Buffer allocated of size 3 bytes
char BufferToOverfLow[3];
//Copies the contents of String buffer to another buffer until a
null character is reached.
strcpy(BufferToOverfLow, String);
}

int main() {
//Buffer allocated of size 34 bytes
char String[34];
//Add data to the buffer
String = "Network security class is awesome";
BufferOverflowFunction(String);
return 1;
}
```

As you can see, even though BufferOverflow buffer is only three bytes long, C/C++ will allow copying data of larger size. What is of concern, however, is what happens when you overflow the buffer? Before we dig deeper into this, we need to know the exact memory layout of a process or a program to predict the behavior of a program after a buffer overflow.

When a function is called, the CPU will store the data of that function (parameters, local variables, and address of where to go after the function is executed) in a Stack Frame (SF). The SF will also contain a Stack Frame Pointer (SFP) which contains the address of a fixed location within the stack so local variables can be located relative to that location.

In *Figure 2*, we show the stack frame for the above BufferOverflowFunction(char * String) function:

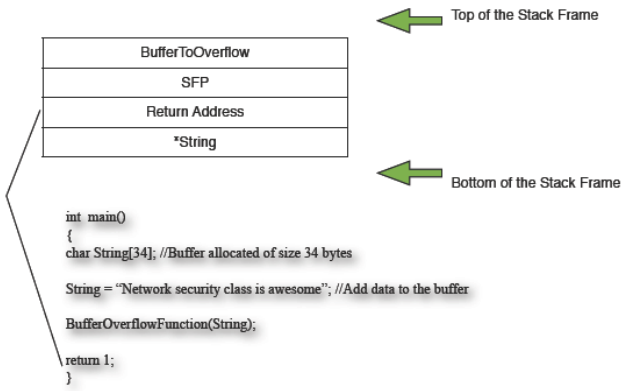

Figure 2: Stack Frame for the BufferOverflowFunction()

If you look at the above Stack Frame, overflowing BufferToOverFlow buffer will overwrite the SFP, Return Address, and *String. This will cause a segmentation fault error and will crash your program. Moreover, the return address can be overwritten in a way the program would jump to execute a malicious code after the execution of the program.

Here is where Shell Coding jumps in as a security threat since it can be used to exploit such mistakes to take over your system. A shell code is a piece of code that is injected into a vulnerable program to exploit your system [11]. Originally, it was called a shell code because the injected code was supposed to take over the operating system's shell command, which would give access to the kernel's functions. However, many of the current "shell codes" don't take over the shell command. Many attempts to change the naming to a more conceptually fitting name failed to succeeded. Shell codes are written in machine code.

To exploit a system using the Buffer Overflow vulnerability, all what the hacker needs to do is to put the shell code in memory and overwrite the return address in the SF to point to the address of the shell code and then the shell code will automatically be executed. That's why buffer overflow is considered to be one of the most dangerous threats being faced in computer and network security.

2.2.4 Attacks on Encrypted Information

Sniffing network data is not a hard task to do. Hence, sharing sensitive information of a high security application in plain text will give the system information to an eavesdropper on a silver plate. Hiding information by encryption is crucial for such applications in this case. Encrypting shared information will enhance the security of the system by making it less probable for attackers to recover shared information. This doesn't mean that the system is totally secure. Cryptography fails to claim that it is unbreakable [12]. In general, there are three methods of encryption:

Alphabetic substitution- is an encryption method which applies a two way one-to-one mapping on every character or ordered set of characters in the alphabet to a different character or set of characters in the same alphabet [13]. This method is hard to break using brute force since the mapping is a permutation of a different alphabetic sequence, which is very hard to guess. However, there is another way that will make breaking it a very easy task. A study was made on the average utilization relative frequency for every letter in the English alphabet. The results are shown in *Figure 3* [14]:

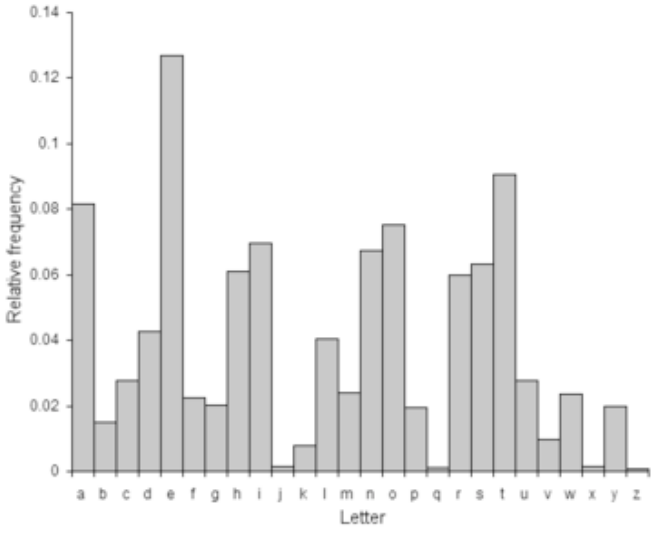

Figure 3: Letter Occurrence Frequency in English Alphabet

This information can be used to attack the Alphabetic Substitution Encryption method. The letter relative utilization frequency analysis can be applied in the same manner for the encrypted text and then use the result to guess the key. For example if we saw that the letter "t" is the most frequently used in the encrypted text then "t" is most probably "e" in the plain text. This can be applied on all other letters.

Symmetric Key Ciphers- is a branch of encryption algorithms that uses the same or trivially related key to encrypt and decrypt data [15]. A good example on Symmetric Key Ciphers would be the Data Encryption Standard (DES). DES is a widely used algorithm that was developed by IBM . The details of DES algorithm are out of the scope of this paper. However, we will introduce the concept to better understand Symmetric Key Ciphers. The key concept of the DES algorithm is that it divides the plain text into equally sized blocks and apply its encryption algorithm on every block [16]. It has two main block encryption mechanisms:

1- Electronic Code Book (ECB)- In ECB mode, DES will divide the plain text to blocks and encrypt each

block with the same key as shown in *Figure 4*. The main advantage of this algorithm mode is that it can be processed in parallel whereas the disadvantage is that identical blocks will produce the same encrypted block.

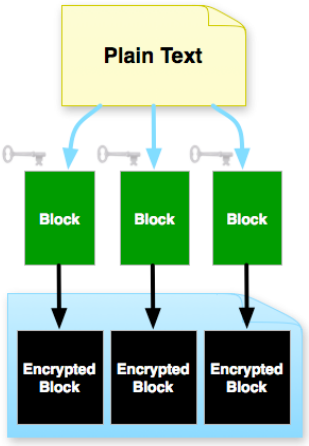

Figure 4: ECB Block Generation

2- Cipher Block Chaining (CBC): Similarly, CBC will divide the text into blocks. However, it XOR's each block with the encryption result of the previous block as shown in *Figure 5*. As for the first block, CBC uses an Initialization Vector (IV), which could be a pseudo randomly generated block or specified by the user. The main advantage of this algorithm mode is that looking at the encrypted text will not give any information to the attacker since the encrypted block depends on entire previous input. However, it has a disadvantage of that it is a sequential algorithm and can't be processed in parallel which makes it slower and less convenient for real time applications.

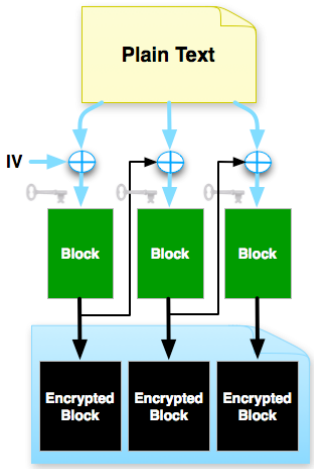

Figure 5: CBC Block Generation

After the encrypted text is transmitted, the receiving end will use the same key to decrypt the cipher. Language analysis can't be applied on this type of algorithms. However, the brute force attack is effective in breaking symmetric key Ciphers especially when the key size is small.

Asymmetric Key Ciphers- is another branch of encryption algorithms where the key used to encrypt data is different from the key that is later used to decrypt it [17]. The key used to encrypt data is called public key and it can be known to anyone whereas the key that is used to decrypt data is called private key and is only known to the host that is receiving the encrypted data. Asymmetric Key Ciphers are considered to be much more secure that Symmetric Key Ciphers and are the most commonly used nowadays.

3 Developed Laboratories

The principle of this study was initiated on the idea that says the fastest way to learn network security is by trying to break the network's security. This study was prepared to form the class note's core and lab session's experiments for introduction to network security class. The students for this class should have a background in Networking fundamentals and TCP/IP protocol. The students should also have basic C programming experience.

The laboratories were developed to accomplish several objectives. The first objective is to let students have the general background about hacking and network attacking methods. The second objective is to help students understand different methods on how to penetrate a network and detect its backdoors, open ports and any unsecured services or holes. The last objective is to teach the students how to secure the network from the discussed attacks by using a variety of defense measures.

Every lab session is ninety minutes long. At the beginning of each lab, students will be required to present chosen topics in network security that could form an introduction/support to the lab. Each lab will be directed toward a general network security problem. It will be composed of a theoretical background section and a set of exercises. The objective of the exercises is to allow students to learn the network security problem by practice. Each exercise will reveal a specific aspect of this problem. After every lab exercise, students are required to solve problems related to the given exercise and then write an explanation about this aspect of the problem and conclude how it could be resolved. In *Table 1*, we show the developed labs along with their objectives, and tools utilized in them.

Lab Name	Lab Objectives	Tools Used
Introduction to labs tools and using SMTP, POP3 and FTP	• Introduction to the Vmware environment. • Working with the Terminal Console through Windows and Unix • Sending anonymous emails using the Simple Mail Transfer Protocol (SMTP) and use its command lines. • Receiving emails using the Post Office Protocol v3 (POP3) and use its command lines. • Downloading, uploading and manipulating files using a File Transfer Protocol (FTP).	NS lookup (host) Sam Spade Pinger Traceroute Telnet Finger Whois/nicname Knowbot Netfind Archie Gopher
Penetration Testing	• Deal with some scanner tools on windows • Be familiar with scanner tools that require Unix. • Learn how to install .rpm and .tar files in Linux. • Be familiar with the command line in Linux/Unix. • Learn how to scan targets without being traced or detected. • How to block ports	Superscan Nmap
Sniffing	• Using Sniffers • Learn how to apply Monkey in the Middle Attack • Encryption as precaution	-Session Hijaking - Dsniff
Denial of Service (DoS)	• Understanding the Denial of Service principle. • Gain knowledge of Resource Exhaustion Attacks. • Knowledge in IP Fragmentation Attacks method. • Knowledge in the Distributed Denial-of-Service (DDoS) method.	Distributed DOS Syn Flood Win Nuke Smuf, snork

Viruses and Worms	• Learn computer viruses types and structure. • Program a simple virus. • Learn how does an anti virus work. • Program an anti virus for the previously created virus. • Differentiate between a computer virus and worm. • Learn how does a computer worm work using the famous "msblaster" worm example.	- Borland C compiler
Password Cracking	• Learn password cracking techniques. • Learn how to attack Windows LM hash passwords using dictionary attack. • Learn how to attack Windows LM hash passwords using brute force attack. • Learn how to attack Windows LM hash passwords using hybrid attack. • Learn how to choose a strong password.	- LCP password cracker.
Buffer Overflow and Shell Coding	• Buffer Overflow description and exploits. • Buffer Overflow programming tutorial (Practice). • Shell Coding in theory. • Shell Coding programming tutorial (Practice). • Take home exercise: Buffer Overflow exploit using Shell Coding	Borland C compiler or other IDA Pro
Cryptography	• Practice and learn Simple Encryption Using classic Caesar Algorithm • Practice and learn Alphabetic Substitution Encryption and Attack Using Language Analysis. • Practice and learn Data Encryption Standard "DES" and develop attacks on it. • Practice RSA Crypto systems. • Practice message signature generation.	CrypTool

Table 1: Developed Labs, Objectives and Tools

By the end of the class, the students are expected to have basic understanding of general issues in network security and the approaches that can be followed to resolve them.

4 References

[1] Rodger, Will. Cybercops Face Net Crime Wave. (1996, June 17). [Online]. Available at http://www.zdnet.com/intweek/print/960617/politics/doc1.html.

[2] Flick, Anthony R.. Crime and the Internet. (1997, October 3). [Online]. Available at http://www.rwc.uc.edu/bezemek/PaperW97/Flick.htm.

[3]. Guide to Firewalls and Network Security: Intrusion Detection and VPNs, Course Technology, by Greg Holden ISBN: 0-619-13039-3

[4]. Intrusion Detection Systems; Definition, Need and Challenges, Sans Institute 2001.

[5]. Computer Virus Tutorial, 2005, Computer Knowledge, http://www.cknow.com/VirusTutorial.htm

[6]. Péter Ször , Peter Ferrie, Hunting For Metamorphic, Symantec Security Response

[7]. Computer Worms Information: http://virusall.com/worms.shtml

[8]. How to prevent Windows from storing a LAN manager hash of your password in Active Directory and local SAM databases, http://support.microsoft.com/default.aspx?scid=KB;EN-US;q299656&

[9]. Net users picking safer passwords, http://news.zdnet.com/2100-1009_22-150640.html

[10]. Crispin Cowan, Perry Wagle, Calton Pu, Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade*

[11]. Yong-Joon Park, Gyungho Lee, Repairing Return Address Stack for Buffer Overflow Protection, CF'04 April 14-16, 2004, Ischia, Italy, Copyright 2004 ACM 1-58113-741-9/04/0004

[12]. Ross Anderson, Why Cryptosystems Fail, http://www.cl.cam.ac.uk/~rja14/wcf.html

[13]. Substitution Cipher: http://www.nationmaster.com/encyclopedia/Substitution-cipher

[14]. Cryptograms and English Language Letter Frequencies, http://www.cryptograms.org/letter-frequencies.php

[15]. Cook, D. L. and Keromytis, A. D. 2005. Conversion and Proxy Functions for Symmetric Key Ciphers. In Proceedings of the international Conference on information Technology: Coding and Computing (Itcc'05) - Volume I - Volume 01 (April 04 - 06, 2005). ITCC. IEEE Computer Society, Washington, DC, 662-667. DOI= http://dx.doi.org/10.1109/ITCC.2005.115

[16]. Walter Tuchman (1997). "A brief history of the data encryption standard". Internet besieged: countering cyberspace scofflaws: 275-280, ACM Press/Addison-Wesley Publishing Co. New York, NY, USA.

[17]. J. Katz; Y. Lindell (2007). Introduction to Modern Cryptography. CRC Press. ISBN 1-58488-551-3.