Packet Wars

By

Jaagrit Arora (11BCE033)

Parjanaya Vyas (11BCE066)

Darshil Patel (11BCE070)

Department of Computer Science and Engineering

Ahmedabad – 382481

November 2014

Packet Wars

Minor Project

Submitted in partial fulfilment of the requirements

For the course of

Seminar Project Lab

By

Jaagrit Arora (11BCE033)

Parjanaya Vyas (11BCE066)

Darshil Patel (11BCE070)

Guide

Prof. Zunnun Narmawala

Department of Computer Science and Engineering

Ahmedabad – 382481

November 2014

Certificate

This is to certify that the Minor Project entitled **Packet Wars** submitted by Jaagrit Arora (1BCE033), Parjanya Vyas (11BCE066) and Darshil Patel (11BCE070) towards the partial fulfilment of the requirements for the completion of the **Seminar Project Lab in Institute of Technology, Nirma University, Ahmedabad** is the record of work carried out by them under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this project, to the best of my knowledge, haven't been submitted to any other university or institution.

Zunnun Narmawala	Dr. Sanjay Garg
Assistant Professor,	HOD,
Institute of Technology,	Institute of Technology,
Nirma University,	Nirma University,
Ahmedabad	Ahmedabad

Acknowledgement

We would like to sincerely thank our guide, Prof. Zunnun Narmawala for his continuous guidance and encouragement throughout the course.

We would also like to thank Prof. Pooja Shah for her immense help towards our project and helping us to finalize the requirements of the project.

We are thankful to Dr. Sanjay Garg, Head of Department, Department Of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad and to Dr. Ketan Kotecha, Director, Institute of Technology for providing such a course in which we were able to get the insights to some booming topics in the field.

We are also thankful to our friends and classmates who have helped us in solving some of the language related problems when we were stuck.

-Jaagrit Arora, Parjanaya Vyas, Darshil Patel

11BCE033, 11BCE066, 11BCE070

Abstract

The project is a mix of three fields related to security: Network Security, Information Security and Log Analysis. The primary goal of this project is to make a platform on which various cyber security attacks can be performed in order to understand and prevent them. Various vulnerabilities are left into the pages and a manual for how to exploit them and use them to hack the site is to be prepared. After the description of particular attack, various techniques to prevent them are to be explained and implemented. SQL Injection, Session hijacking, cross site request forgery are considered here in this project. In addition to this, data forensics and recovery is also studied. The other goal of the project is to provide a platform for performing and understanding various encryption and decryption algorithms. Such algorithms have to be implemented either through PHP or JavaScript. The project also included analysing a sample application named “hacme casino” developed by “McAfee” using Ruby on Rails and other client side and server side scripting technologies. Hacme casino includes many types of well-known cyber security vulnerabilities like SQL Injection and session hijacking but also includes some unique techniques which shows us that cyber-attacks are not just using the well-known methods to exploit a vulnerability but is actually a creative art, which studies the whole website for any vulnerability available, whether known or unknown. Various attacks are tried to understand them properly on hacme casino.

CONTENTS

Certificate

Acknowledgement

Abstract

Table of Contents

Chapter 1

Introduction

In reality, Packet Wars an intense, real-time Cyber Warfare simulation in the form of a game. The game actually demands the players to break the security of their competitors and get into their system through the use of Network Security, Information Security and Log Analysis and also protect their systems from the others trying to break through. The project contains the three main parts.

The first part of the project deals with Network Security. Network Security deals with the protection of networks and their services from unauthorized modification, destruction, or disclosure. /*** BY PARJANYA***/

The second part of the project deals with Information Security through the use of cryptographic algorithms. Information Security is the way of protecting the information against unauthorized disclosure, transfer, modification, or destruction, whether accidental or intentional. The Cryptography section is further divided into the two main sections of encryption and decryption. A number of encryption and decryption algorithms have been implemented through JavaScript.

The third part of the project deals with the Log Analysis of LINUX operating system. A Perl Script has been generated to analyze the log file of LINUX. The script is generated such that the output of the script provides a set of user understandable statements corresponding to the log codes in the file.

Chapter 2

Cryptography

2.1 SQL Injection

As it might be pretty clear from the name above, SQL injection is a technique which injects code in to the SQL query, which is passed by the front end page to the back end page by the application. Though this technique is a very old candidate in the field of network security vulnerabilities, still it proves to be a perfect way to step in to the world of hacking, because it is 1) Impressive 2) Very easy to detect whether this vulnerability is present in the application or not 3) Pretty easy to understand how it works 4) Has a formal step by step procedure following which, it can be implemented. The 4^{th} point makes it most interesting because unlike any other vulnerability exploitation, this method can be used by just following a set of steps available from anywhere on the internet and you do not need to have much in depth knowledge of how stuff works, apart from the SQL syntax and semantics.

As the name suggests, this technique consists of two words. First is SQL, which is Structured Query Language, and is very well known in terms of its usage with databases. Second states “injection” which means to add your own code, somewhere between the codes that are already there in the application. Imagine that you can bypass the front end and directly query the back end database of the application. You can fetch, insert, manipulate or delete data from these database tables. But the question is how this miracle is possible? SQL Injection is after all a black-box testing technique. So user has a limited number of options where he/she can add self-generated codes and can pass it to the back end. First option is the address bar of the browser. But very few applications uses “get” requests to pass something as important as queries across the web pages so this option is not a good choice. Other option is the text boxes available in the web page. This is the perfect choice to implement SQL injection. Values inside the textboxes are passed directly to the back end server side codes and these codes do not usually have enough time to validate these

values. So they pass these values to the SQL queries and hence, if we manipulate the values inside a text box, we are manipulating the queries. Client side validation using Java script or similar things can be an obstacle to this technique but that too has solution. Simply turn off the Java script! That is perfectly possible in any well - known web browsers. But there are many solutions to prevent SQL injection and almost all current websites have implemented one of them to secure their sites.

So, how to detect whether SQL injection is possible or not? There is a hallmark string to do so. Simply enter a single quote (‘) into the textbox and submit the data. If the page directly gives an SQL error than, you are done! You have interfered with the SQL query using your string and hence you can inject your own SQL knowledge to play with the queries, detect number and name of tables and columns, extract data and if at all possible modify or delete data. Some classical examples of SQL injection strings are:

1) ' or 1=1--
2) " or 1=1--
3) or 1=1--
4) ' or 'a'='a
5) " or "a"="a
6) ') or ('a'='a
7) ") or ("a"="a

Using these strings, you can bypass authentication. Many other strings, according to the knowledge of SQL and the implementation of the web page on which the attack is performed, can be generated and used in various textbox fields. In this sample application, these strings can be tested.

The last thing that we want to know is that our web page is vulnerable to SQL injection. As shown earlier, it is an impressive, powerful and pretty easy to implement attack. So we must make sure that the web pages we make are not vulnerable to these attacks. As already stated, easiest way to do so is use client side validation to prevent user entering anything that is not supposed to be entered in the textbox but again, the problem with client side validation is, it is done on ‘client’ side and in our case, client is our attacker. So he can just turn it off and enter whatever he wants into the fields. Another thing that can be done to prevent SQL injection is to do the server side validation. It is a little costly solution to implement in terms of time because server has many important

things to do than validating the user inputs. But if we have enough memory and processing power on server side, this can be implemented pretty efficiently and easily. Many websites use this type of solution for preventing SQL injection attacks. The other slightly more complicated solution is to use built in SQL functions to execute the SQL queries. It requires in depth knowledge of the database used in server side and complicates the SQL queries but if done in the right way, than it can prevent this attack the most efficiently.

Concluding the section, here we have stated what is SQL injection, how is it dangerous, what effects it does, how is it implemented and how to prevent it. Cyber laws, across the nations, are very strict against SQL injection. In US, if it is detected and proved that you have entered a single quote (‘) into a text field where it was not required, than the developer can file a case on you, just for entering a single quote and charges up to six months of prison can be applied against you for trying to manipulate and hence illegally access the server database by using SQL injection. Though in India, rules are not that tight but you can be in trouble if you try this on any actual website. So better practice it in this application only and DO NOT TRY THIS AT HOME.

2.2 Cross site request forgery

This attack is actually “fooling” the webpage which we are trying to attack and hence, we are not actually fabricating things or injecting codes. All we have to do is observe how the webpage works very intensely and then try to exploit a weakness which in general case, we would definitely find if we are observant enough. Though this attack is also very primary in this area, it still actually persists and at the end of this section, an actual ‘real world’ example is shown where this attack was used by us to exploit a weakness in a very well-known website.

Here, two things are required apriori to the actual attack. A proxy server installed in your system and a data sniffer. It would be great if both of them are integrated in one single application and one such application is Paros proxy. These two things are needed because we have to analyse and infer the data and control flow of the whole application. From where to where and which messages are passed? What method (usually ‘get’ or ‘post’) is used? And most importantly, which parameters are passed? If the application is using ‘get’ requests to pass the parameters than it would be very easy to trace them and even the proxy is

not needed as just the address bar will tell everything we need to know. But in most, if not all, cases this is not true. Nearly everyone uses 'post' method which is more reliable, more secure and harder to trace. In that case, proxies come in handy. We have to set proxy in such a manner that every interaction between our browser and the server of the application is done via proxy server only and the data sniffer integrated into the proxy will trace all the messages and give us a report of how and which parameters are passed. Now after knowing this, all we need to do is generate different test cases and know what value of a certain parameter does what? And this is easier done than said. We can almost predict what is the influence of a value of certain a parameter on the application. E.g., commit = 'transfer_chips' & amount = '1000' obviously means that this value of the parameter 'commit' will cause the chips to be transferred and number of chips that are transferred would be 1000. Similarly, as we know these parameters are there, there must be some parameter in the same request message or in any nearby messages which states to and from whom the chips are to be transferred.

After the identification of different values of certain important parameters now we have enough information to fool the server that we are trying to attack. After observing the working of the web page we now have to find a weak spot inside this work flow of the application. E.g., in the application developed here, we can see that get requests are used to transfer the messages and hence each time a question is to be evaluated to check whether it is correct or not, the unique 'qid', 'aid' combination is sent to the server which determines whether this combination states a correct answer or a wrong one. Here, we have got one weak spot. Now what we can do is, answer one question correctly and store the combination of qid and aid which is known to us as a correct combination. Now, at the stage of answering a different question, instead of directly sending the generated URL, first change the combination of the generated URL's qid and aid to the one which we know is correct. Now send this modified URL and see what happens. YES! Our answer is treated as a correct answer and our marks are incremented. Keep doing this until the end of the quiz and you will get full marks by answering just the first question correctly.

Some of the phishing attacks can also be classified as cross site request forgery attacks. Phishing is actually sending some malicious URL to a user under the name of a trusted site. The user will click on this URL and get directed to the page which we want him to visit. In the case considered above for a chip transfer, once we identify all the necessary and sufficient parameters and their

values for transferring chips from one user to another on a casino website, say xyzcasino.com, we can use them to do a phishing attack. For example as stated above, we have identified that commit, amount, to_user and from_user are the parameters which we require to set in order to send chips from one account to another on this casino website. Now, if we this website uses 'get' method to pass the parameters, then we can generate a URL which looks like: www.xyzcasino.com?commit='transfer_chips'&amount='1000'&to_user='ronald'&from_user='rupert'. Now, all we need to do is send this URL to our friend named Rupert while he is logged into his account of the xyzcasino website. We can tell him that, this is a nice website that you should visit or any temptation so that he will click on the link. As soon as he clicks on the link, xyzcasino's server will get a message to transfer chips from Rupert's account to our, that is, Ronald's account and if Ronald is lucky enough that Rupert has enough chips in his account, he will get 1000 chips' worth gift from Rupert and Rupert will only get to know when he will check his account. So in this way, some of the phishing attacks are also performed after identifying parameters and hence by using the CSRF attack's results.

This attack is actually implemented by us on a website named www.harrypottertrivia.net in which, instead of 'get' they use 'post' requests to send the combination of qid and aid. This weakness is exploited and a self-generated html page, which always submits the correct combination to the server, is loaded. As a result the marks keep getting incremented.

As explained above, what this attack actually does is it finds any logical weaknesses in the webpage and then exploits them in our own advantage. So, to prevent this attack, our algorithm which is used to make the application should be tested for as many test cases as possible. In order to do so, it is always good to do as much black box testing as possible before actually making the application available to the global market and hence, attackers. Use of 'post' requests is always good to make the task more difficult for the attacker. But as we just saw, it is possible to do this attack even with 'post' requests, so always use cryptography while passing the parameters across the web pages. This is a pretty good solution and it will make it very hard for the attacker to interpret the meaning of various values of the parameters. The other uses of cryptography, various algorithms and their crypt analysis is presented in the latter section of this report.

2.3 Session Hijacking and exploiting improper session handling

Session hijacking, as the name suggests, is using a valid session (session key) in order to gain information or do some unauthorized tasks. Session keys are mainly stored in the cookies or are sent in initial messages to the client. So here too, proxies can be used to get a valid session key and use it in our own advantage. Initial POST messages usually contain a session key, which is used to validate further requests. Now, using proxy, along with the data sniffer as described previously, we can get the session key that is hidden in the initial messages. Now, whenever we forge our own requests we need to validate them on server side and using a proper session key is the only way for doing so. The session key that we got from the data sniffer is integrated into our forged message so that they are validated successfully on server side and our attacks become successful. Some of the cookie viewer and editor software like Karen's cookie viewer allow us to view and edit the cookies that websites save on our computer. Using them we can get access to the session keys stored in the cookies.

Session hijacking is an attack which is mainly dependent upon data sniffers and proxies. To prevent this attack, encryption is the most efficient technique. If session keys are encrypted using public-private key method, even if this encrypted session key is captured by the attacker, he/she cannot do anything with it because it requires a unique decryption key which is not the same as encryption key and hence capturing session key is of no use to the attacker. Another technique to prevent it is change the session key at every fixed time interval. So, even if initially attacker gets the key, s/he will be able to log in to the valid session for only a limited amount of time. After that, the session key will automatically change and hence attacker has to do everything again. If the timeout is sufficiently small then this technique proves to be very efficient. Another very efficient technique to prevent this attack is to make secondary checks and hence not relying entirely upon the session key. Web servers may check for the IP address of the user who has made previous request with the IP address of the current requesting user. This will not, however, prevent the attack when the attacker shares same IP address as user or is spoofing the IP address. Moreover, this technique will also be frustrating to the users whose IP addresses change dynamically after a certain amount of time.

Chapter 3

Cryptography

3.1 Introduction

Cryptography is a word which has its origin from Greek. Kryptos means hidden or secret and Graphein means writing making Cryptography to literally mean written in secret. In figurative sense Cryptography means the art of writing or solving codes. The formal definition of Cryptography is as follows:

Cryptography is a method of storing and transmitting data in a particular form so that only those for whom it is intended can read and process it.

Cryptography is usually misunderstood with two other terms related to it, Cryptanalysis and Cryptology. While cryptography means writing hidden messages; cryptanalysis means analyzing hidden messages and cryptology means reading the hidden messages.

3.2 Key Security Concepts

The major three security concepts for the data are: Confidentiality, Integrity and Availability. These three concepts are referred to as the CIA triad. The three concepts embody the fundamental security objectives for both data and for information.

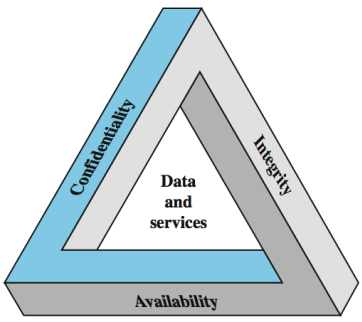

Figure 3.1: CIA Triads

The CIA concepts are explained below. Over and above the well-established CIA triads, some experts in the field also feel that two other commonly used concepts are used:

- Confidentiality: It deals with preserving authorized restrictions of information access including means for protecting personal privacy and proprietary information. A loss of confidentiality is the unauthorized disclosure of information.
- Integrity: It deals with guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity. A loss of integrity is the unauthorized modification or destruction of information.
- Availability: Availability means providing timely and reliable access to information. A loss of availability is the disruption of access to information.
- Authenticity: It is the property of being genuine and being able to be verified and trusted.
- Accountability: It is the property of a system that ensures that the actions of a system entity may be traced uniquely to that entity, which can be held responsible for its actions.

3.3 Symmetric Cipher Model

A symmetric cipher model has the below mentioned five elements:

- Plaintext: Plaintext is the data given as input to the algorithm used for encryption. It is the original data sent by the sender which is intended for a particular receiver or a group of receivers.
- Encryption Algorithm: This is the core of the encryption process. The input to the algorithm is the plaintext. It makes some substitutions to convert the plaintext into the text which is unreadable to everyone for whom it is not intended.
- Key: A key is also an input to the encryption algorithm. The plaintext and key together determine the output of the encryption algorithm. Depending on a different key, the output of the encryption algorithm differs.
- Ciphertext: It is the message obtained as an output from the encryption algorithm. Ciphertext is a scrambled stream of bits which is unreadable for most of the people and can be decoded only by the person intended.

- Decryption Algorithm: As an encryption algorithm takes plaintext as input and gives ciphertext as the output; in the same way, decryption algorithm takes ciphertext and key as the input and retrieves the plaintext at the receiver's end.

Figure 3.2 depicts the whole symmetric cipher model in a pictorial way:

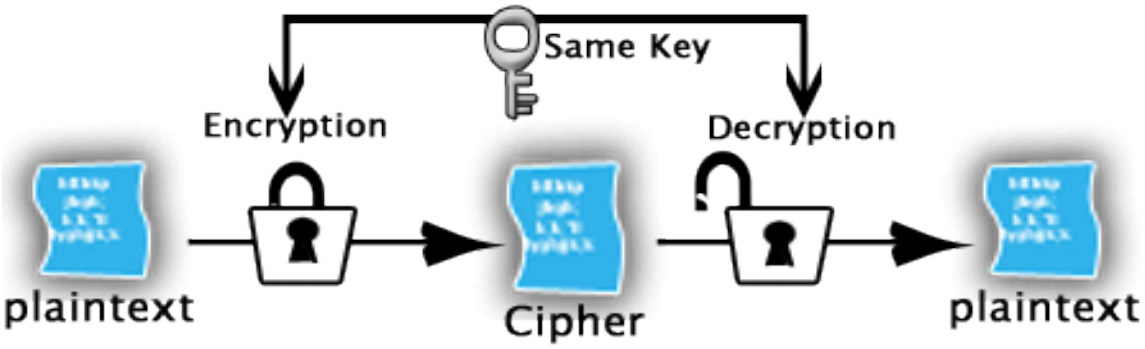

Figure 3.2: Symmetric Cipher Model

3.4 Caesar Cipher

3.4.1 Introduction

In the field of cryptography Caesar cipher is one of the simplest encryption technique. It is a substitution cipher wherein each letter is shifted by a fixed amount of number which is provided as key. The algorithm is named after Julius Caesar who was considered to be using this with a key of 3 for protecting his military correspondence. Being one of the simplest encryption technique to understand, it is also easy to implement. But as a serious drawback, it is also easy to break the ciphertext. Breaking the code is very easy if the attacker knows that some substitution cipher is used or in particular if Caesar Cipher is used. The major reason is that the value of key can take only 26 values. If the left and right shifts both are considered then at max 52 different values of key is possible. Thus an attacker can get the plaintext by trying out all the possible values of keys.

An easy way to implement Caesar cipher is initially writing the alphabets in one row by aligning the corresponding cipher character in the second row according to the given key. For example, if the key is 2, the following table will help to encode a particular plaintext to corresponding ciphertext.

Plain:	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher:	CDEFGHIJKLMNOPQRSTUVWXYZAB

3.4.2 Caesar Algorithm

The basic Caesar cipher algorithm uses the following two formulae for the purpose of encoding and decoding if M signifies the plaintext character, K signifies the value of key and C signifies the value of ciphertext:

Encryption: C = (M + K) % 26 and Decryption: M = (C - K) % 26

3.5 Vernam Cipher

Vernam Cipher was initially invented by Gilbert Sandford Vernam at AT&T Bell Labs during the world war I. Vernam cipher is a symmetrical stream cipher in which the plaintext is combined with the key to generate the ciphertext using the bitwise exclusive or function. Vernam Cipher too is not much secure. As it uses only the XOR function to create the ciphertext, the attacker can easily break the code if it is known that vernam cipher is applied for encryption. This is because of the unique property of XOR that the output obtained by XOR of a number with some fixed number, if XORed again with the same fixed number, returns the other number given as input. As in this case, the plaintext and key are XORed to get the ciphertext and thus an attacker can again XOR the ciphertext and the key to get the plaintext, provided that the key is known.

If the key length is same as the length of the plaintext then whole key is consumed to get the ciphertext. In case when the key length is longer than the plaintext, the remaining part of the key is discarded. Finally if the key length is less than the plaintext, the key is repeated, once it is exhausted. The Vernam Encryption and Decryption algorithm is implemented using the below given formulae. If M is the plaintext message, K is the key and C is the ciphertext then,

Encryption: $C = M \oplus K$ and Decryption: $M = C \oplus K$

3.6 Vigenère Cipher

Vigenère Cipher, proposed by Blaise de Vigenère, is essentially a form of polyalphabetic substitution. In Caesar cipher, each character is shifted along some fixed number of places, while Vigenère cipher consists of a series of Caesar ciphers with different shift value. Figure 3.3 shows the series of Caesar ciphers which are used to encrypt using Vigenère cipher.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figure 3.3: The series of Caesar ciphers used for Vigenère cipher

As in the Vernam cipher, in Vigenère cipher as well, the key is repeated if exhausted and discarded if long. The encryption is done using the figure given above. The row corresponding to the first character of the plaintext is considered and the column corresponding to the first character of key is considered. The character obtained by intersection of the column and the row will be the first character of the required ciphertext. Similarly for each character in plaintext, a character for ciphertext is obtained. As in the case of decryption, the row corresponding to the key is checked. Now a column is found such that the intersection of the row and the found column results in the ciphertext. Thus, the plaintext character is the character corresponding to the found column. If M is the message, K is the key and C is the ciphertext and i represent the number of character processed, then:

$$C_i = (M_i + K_i) \% 26 \quad \text{and} \quad M_i = (C_i - K_i) \% 26$$

3.7 Rail Fence Cipher

Rail Fence is a type of transposition cipher which follows the simple rule of mixing up the characters of the plaintext to form the ciphertext. In general, in rail fence algorithm, the value of key determines the number of imaginary rails to be made for the purpose of encryption. Each letter of the plaintext is kept in the subsequent rail and once a character is entered in the last rail, the next character is added to the first rail. The basic working goes on till the end of the plaintext. After all the rails are ready, the rails are read one at a time, to create the ciphertext. An improved version of rail fence algorithm is implemented in the project in which, the rail fence algorithm with two rails is worked upon for as many times as the value of the key. Thus providingan input of abcdef and key as 3, after the first iteration, the output will be acebdf which will again be fed into the algorithm providing the output as aedcbf. For the final iteration, this text is given as the input to get adbecf as the final output as the ciphertext.

3.8 RSA

Firstly described by Ron Rivest, Adi Shamir and Leonard Adleman, RSA is a public key cryptosystem used for secure data transmission. In this type of cryptosystems, the encryption and decryption key is different from one another and is kept secret. In RSA, the asymmetry of the two keys is due to the difficulty of factorizing a product of large prime numbers.

The algorithm is mainly divided in two parts: Key generation and Encryption or Decryption. In the key generation algorithm, initially two sufficiently large prime numbers are selected, say p and q. Now two more variables, n and t are found using the values of p and q where $n = p*q$ and $t = (p-1) *(q-1)$. Having the four basic elements p, q, n and t, the next step is to find the encryption and decryption keys e and d respectively. The encryption key e is selected such that e and t are co-prime numbers i.e. the GCD of e and t is 1. Also the value of e is bounded between 1 and t. The public key is a set {e, n} which is used for encryption. For the decryption purpose, the decryption key d is found such that $d = e^{-1} \% t$, in other words, $(d * e) \% t = 1$. Thus the private key is a set {d, n} which is used for decryption.

The Encryption algorithm will give the ciphertext C using the public key set {e, n} and message M with the formula:

$$C = M^e \% n$$

On the other hand, the decryption algorithm will give the plaintext M using the private key set {d, n} and the ciphertext C with the formula:

$$M = C^d \% n$$

Chapter 4

Log Analysis

syslog is a host-configurable, uniform system logging facility. The system uses a centralized system logging process that runs the program /etc/syslogd or /etc/syslog. Individual programs that need to have information logged send the information to syslog. The messages can then be logged to various files, devices, or computers, depending on the sender of the message and its severity.
Any program can generate a syslog log message. Each message consists of four parts:

- Program name
- Facility
- Priority
- Log message itself

For example, the message:
login: Root LOGIN REFUSED on ttya
is a log message generated by the login program. It means that somebody tried to log into an unsecure terminal as root. The messages's facility (authorization) and error level (critical error) are not shown.

syslog Facilities

Name	Facility
Kern	Kernel
User	Regular user processes
Mail	Mail system
Lpr	Line printer system
Auth	Authorization system, or programs that ask for user names and passwords (login, su, getty, ftpd, etc.)
Daemon	Other system daemons
News	News subsystem
Uucp	UUCP subsystem
local0... local7	Reserved for site-specific use
mark	A timestamp facility that sends out a message every 20 minutes

Messages are labeled with a facility code (one of: auth, authpriv, daemon, cron, ftp, lpr, kern, mail, news, syslog, user, uucp, local0 ... local7) indicating the type of software that generated the messages, and are assigned a severity (one of: Emergency, Alert, Critical, Error, Warning, Notice, Info, Debug).
The log analysis was performed on Unix systems.
Unix logs are managed by the syslogd utility. Since unix/linux uses a uniform byte format for all files, it is possible for a user written program to access these files and parse them to detect errors or any other anomalous activity.
Servers and desktops are both generally running linux(generally debian) as an operating system. Thus, scripts written for the desktop can also be applied to linux based server systems.

In order to find what events to look for, a system was run for a period of approximately 9-11 hours and events were repeatedly performed like usb plugins, shutdown, kernel events, normal programs and driver events etc.
Once found, the event logs were scanned for event signatures common to all events like usb plugins. After that, the events converted to regular expressions and written to a perl script as perl allows for regular expression parsing inbuilt.

TCP and wireshark practicals:

Wireshark is a packet capture and analyser for multiple platforms. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Wireshark allows the user to put network interface controllers that support promiscuous mode into that mode, in order to see all traffic visible on that interface, not just traffic addressed to one of the interface's configured addresses and broadcast/multicast traffic. However, when capturing with a packet analyzer in promiscuous mode on a port on a network switch, not all of the traffic travelling through the switch will necessarily be sent to the port on which the capture is being done, so capturing in promiscuous mode will not necessarily be sufficient to see all traffic on the network. Port mirroring or various network taps extend capture to any point on the network. Simple passive taps are extremely resistant to tampering.

Wireshark has a comman line interface called tshark that is capable of much of the commands used by wireshark. Again, packets were analysed by running a long session and performing some events over and over again at specific times to find signatures and defining traits of some common occurences.

The possible events that could be detected reliably are:

- Images requested with url
- Videos
- Websites requested
- TCP stream like file transfers
- Port scans
- telnet and ssh attempts
- file transfer attempts across the ftp protocol

The flags and other criteria used for filtering the raw packet data are as follows:

Along with the above, the script was capable of dumping raw TCP data into TCP streams from all servers. This allows users to analyse TCP streams for any illegal access. The script is incapable of analyzing the type of data but dumps raw data successfully into separate files.

Log analysis software for windows:

1. Logalyze: LOGalyze is an open source, centralized log management and network monitoring software. One can create their own rules for log analysis and the software does the parsing on its own. It can also generate reports and logs for events as required. It allows for the following functionalities:
    a. Collect
    b. Log/Store
    c. Parse/Analyze
    d. Report/Alert

2. Solar Winds LEM: Each log contains different types of logs like errors, warnings, information, success/failure audits. The very nature of windows logs, which are tantamount in number, needs strong analysis capabilities. LEM analyzes logs from a variety of network periphery security devices like firewalls, proxy servers, IDS, IPS, VPN. LEM has a library of built-in Active Responses which executes the automated responses to mitigate threats and responds instantly to operational issues. Some built-in responses include:
    a. Block an IP address
    b. Enable or disable domain user accounts, local user accounts, or Windows® machine accounts
    c. Kill processes by ID or name
d. Send incident alerts, emails, or popup messages

Log analysis tools for unix:

1. Syslog system utility: The syslog system utility is inbuilt into unix systems by default and newer versions generally contain GUI interfaces to the same utility. These GUI are generally powerful enough for linux users and advanced

users like sysadmins can use perl scripts like the one used developed for this project to look for anomalies. The logs are sperated by type into the following categories:

a. Kernel
b. Sys
c. User
d. Daemon
e. Auth

Based upon the level of the message and the log file it is in, actions can be instituted to ensure that an admin is notified of any anomalies.

Although third party software exists for unix based systems, it is merely a different GUI over the same syslog utility.