ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 3, March 2015

Client Hacked Server: Application Scenario for Malicious Web Control

Ajit Prakash Alijad[1], Pritam Vishnu Aher[2], Sujata Ashok Chechare[3], Vishal Eknath Ghume[4]

BE Scholar, Department of Computer Engineering, S.R.E.S.'s College of Engineering, Kopargaon, India[1,2,34]

Abstract: Client-Server model is the backbone of today's internet communication. In which normal user can not have control over particular website or server. By using the same processing model one can have unauthorised access to particular server. In this paper, we discussed about application scenario of hacking for simple website or server consist of unauthorized way to access the server database. This application emerge to autonomously take direct access of simple website or server and retrieve all essential information maintain by administrator. In this system, ip address of server given as input to retrieve user-id and password of server. This leads to breaking administrative security of server and acquires the control of server database. Where as virus helps to escape from server security by crashing the whole server

Keywords: Hacking, Vulnerabilities, Dummy request, Virus, Server monitoring

I. INTRODUCTION

Hacking is an illegal and unauthorised activity in today's world wide web communication. But this unauthorised way can be used to improve security scenario of the global communication network i.e. internet.

Different websites over the internet might having the vulnerabilities in their security structure or some of the websites used to carrying illegal work or contain some confidential data.

In such a cases, this system plays an important role as it will bring benefits for illegal data detection, checking vulnerabilities present in websites, checking malicious functions etc.

In client-server model communication, number of client machines are ultimately connected to main server.

Client made request to the server and in response to client's request server process it and return result back to client. This approach can be useful to retrieve username and password of the server.

This user-id and password helps to take control of administrative panel of server machine which is considered as an one of the security mechanism of the server.

Once this administrative control taken successfully then it is easy to change remaining security mechanisms and access server as user wants.

In this paper, we discussed about some technological work towards achieving the way to access server in un-authorized manner ultimately serving to cyber applications.

II. SYSTEM USE CASES

This section provides more detail on some of the use cases, covering the most important user requirements in order to clarify user scenario on this system.

All of these application s, have some common working requirements. The security of the information and trustworthy operation of these application s is of paramount importance, as is the ability to manage over the air.

A. Use Case 1: Cyber Forensics

Cyber forensics with working environment to detect or to keep track of all activities on the internet. Which requires to have direct access to particular web-site.

B. Use Case 2: Cyber Investigation

In order to find illegal work on internet cyber investigation department requires direct access to server database in unauthorised manner.

III. SYSTEM OVERVIEW

Fig. 1 delineats the fundamental architecture of proposed system, consisting of server security levels (proxy server, firewall), username-password program, server monitoring, virus etc.

Processing in this system is typically done by high speed wired/wireless network mechanism connects client system, servers for services and applications.

In this paper, we focus on wired as well as wireless connectivity, which allows flexibility to application.

The proposed architecture consist of username and password retrieval, secure storage of data, main software program, virus execution as main execution modules for this application.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 3, March 2015

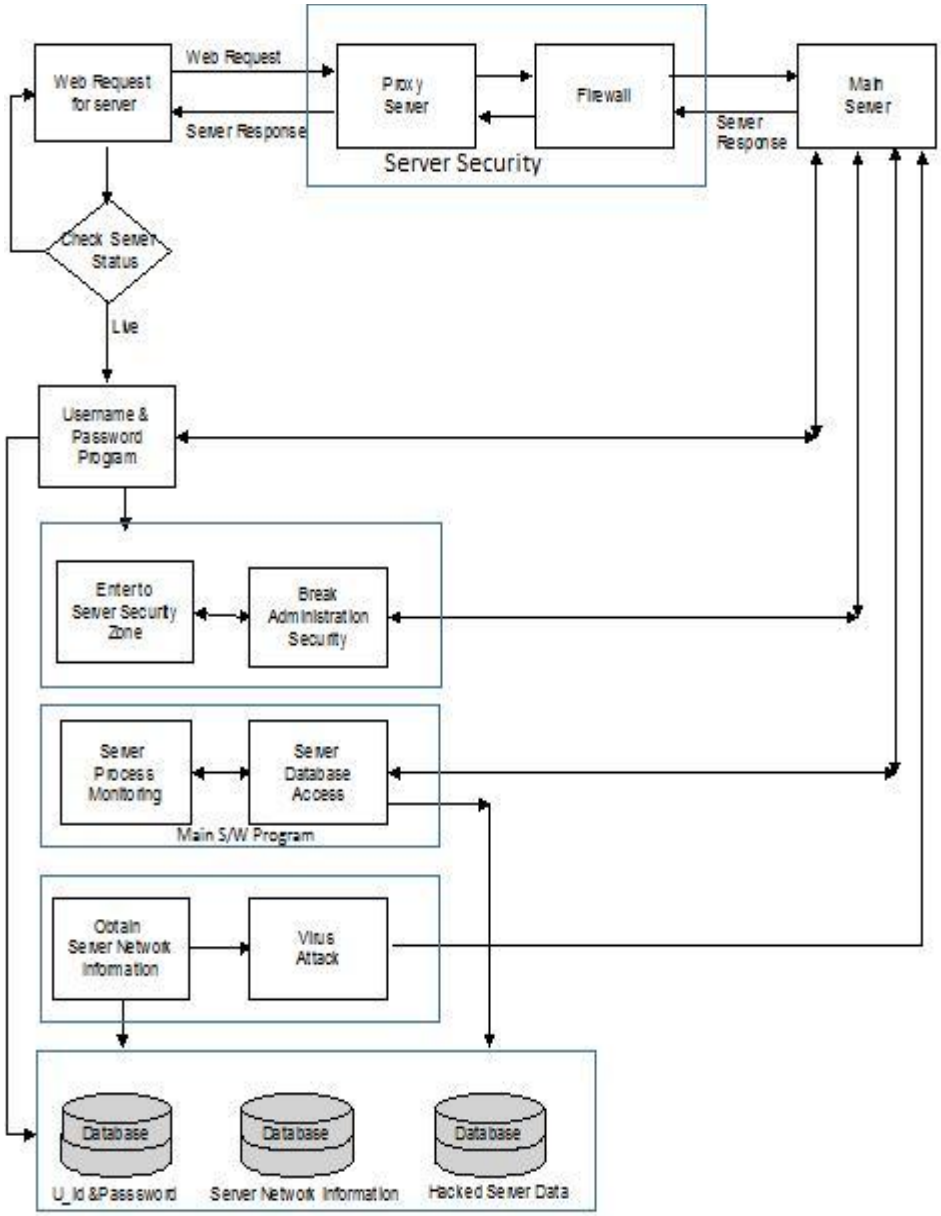

Fig.1. System Architecture

A. Username and Password Retrieval

This module provides username and password of main processing server. Initially dummy request made to the client to check whether server is live or not. If server response is live then the system executes program which retrieves certain information. Once the server information retrieved then it will stored on client machine.

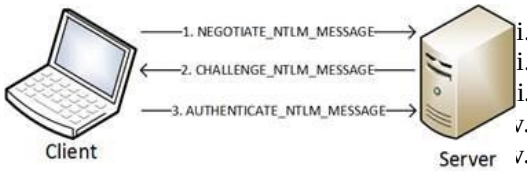

Fig.2. Initial Process Phase

This ultimately involves different hashing algorithm as well as some of the decryption techniques.

1) LM-Hash Algorithm: Server administrator use a strong password to file, system, drive etc. It may be 8 - character or 16-character in length. It is difficult for attackers to crack that file and hack the same. This algorithm mainly focus on combination of string, numeric, character etc.

Algorithmic Steps For LM-Hash-

Following are the execution steps for given algorithm.

i. Define methods for error reporting.
i. Define program execution path.
i. Configure HTTP request authentication.
v. Set user to authorised user.
v. Use combination parts of all possible ones.
vi. Assign specific patterns of ones.
vii. Match possible combination of number and character.
viii. Save pattern match.
ix. Execute command path.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 3, March 2015

B. Secure Storage Of Hacked Data

Once the data has been hacked from server then it is necessary to store that data in secured way. This module involves automatic storage of server data on client machine.

C. Unauthorised Access To Server Database

Databases are most valuable data rest corporate data, customer data, financial data etc. Databases have many entry points i.e. web applications, internal networks, partners network etc. If the OSs and networks are properly secured databases still could be misconfigured, have weak password, vulnerable to known/unknown vulnerabilities. If the passwords are blank or not strong they can be easily bruteforced[7]. Databases can be hacked from the internet. Firewalls are completely bypassed. Basic NTLM authentication schema –

Client → connects → Server
Client ← sends challenge ← Server
Client → sends response → Server
Client ← authenticates ← Server

Application objects are usually stored in an object-oriented database. Sometimes, however, the customer dictates a database platform; in that case the database is usually relational. As also noted in [10], all object databases have significally different programming interfaces. Even if a standard exists [11], this standard is somewhat incomplete, and every implementor fails to support some parts of it. This means in practice that an application written for a specific object database is not portable to other databases. The situation is even worse if a relational database is used, potentially requiring a major rewrite of large parts of the application.

This module used to take control of server by using username and password retrieved by initial phase. It mainly involves breaking of administrative security which is in the form of secure authentication. Once the control of the server taken successfully then client can easily access server data as well as able to manipulate server data with the help of this module.

1) *Web Server Hacking*: A Early web hacking frequently meant exploiting vulnerabilities in web server software and associated software packages, not the application logic itself. In this paper, we discuss about vulnerabilities associated with popular web server platform software such as Microsoft IIS/ASP/ASP.NET, LAMP(Linux/Apache/MySQL/PHP), BEA Web-Logic, IBM Web-Sphere, J2EE, and so on. These types of vulnerabilities are typically widely publicized and are easy to detect and attack. Some of the most devastating Internet worms have historically exploited these kinds of vulnerabilities (for example, two of the most recognizable Internet worms in history, Code Red and Nimda, both exploited vulnerabilities in Microsoft's IIS web server software). Although such vulnerabilities provided great "Low Hanging Fruit" for hackers of all skill levels to pluck for many years.

D. Monitoring Server Processes

It mainly views all actions takes place on the server as server process analysis.

1) Session Hi-Jacking:

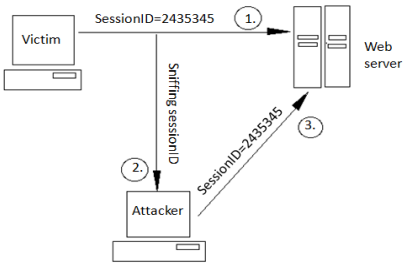

Fig.3. Process Of Session Hi-Jacking

It can be done at two levels *Network Level* and *Application Level*. Network layer hijacking involves TCP and UDP sessions, whereas Application level session hijack occurs with HTTP sessions. Successful attack on network level sessions will provide the attacker some critical information which will than be used to attack application level sessions, so most of the time they occur together depending on the system that is attacked. Network level attacks are most attractive to an attacker because they do not have to be customized on web application basis; they simply attack the data flow of the protocol, which is common for all web applications.

TCP hijacks are meant to intercept the already established TCP sessions between any two communicating parties and than pretending to be one of them, finally redirecting the TCP traffic to it by injecting spoofed IP packets so that your commands are processed on behalf of the authenticated host of the session[12]. It desynchronizes the session between the actual communicating parties and by intruding itself in between. As authentication is only required at the time of establishing connection , an already established connection can be easily stolen without going through any sort of authentication or security measures concerned. TCP session hijacks can be implemented in two different ways: Middle Man Attack (suggested by Lam, LeBlanc, and Smith) and the Blind attack.

Since UDP does not use packet sequencing and synchronizing; it is easier than TCP to hijack UDP session. The hijacker has simply to forge a server reply to a client UDP request before the server can respond. If sniffing is used than it will be easier to control the traffic generating from the side of the server and thus restricting server's reply to the client in the first place.

E. Virus Execution

A computer virus is a malware program that, when executed, replicates by inserting copies of itself (possibly modified) into other computer programs, data files, or the boot sector of the hard drive; when this replication succeeds, the affected areas are then said to be "infected".

DOI 10.17148/IJARCCE.2015.4304

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 3, March 2015

When server security tries to block the client then virus programme gets into action which performs SQL-Injection attack and hold up system for two-hours as well as make server system to be crash if needed.

1) SQL Injection Attack:

SQL injection attacks pose a serious security threat to Web applications: they allow attackers to obtain unrestricted access to the databases underlying the applications and to the potentially sensitive information these databases contain. SQL injection refers to a class of code-injection attacks in which data provided by the user is included in an SQL query in such a way that part of the user's input is treated as SQL code. By lever aging these vulnerabilities, an attacker can submit SQL commands directly to the database[13]. These attacks are a serious threat to any Web application that receives input from users and incorporates it into SQL queries to an underlying database. Most Web applications used on the Internet or within enterprise systems work this way and could therefore be vulnerable to SQL injection.

1.1) Mechanism for Injection: Malicious SQL statements can be introduced into a vulnerable application using many different input mechanisms.

***Injection through user input*:** In this case, attackers inject SQL commands by providing suitably crafted user input. A Web application can read user input in several ways based on the environment in which the application is deployed. In most SQLIAs that target Web applications, user input typically comes from form submissions that are sent to the Web application via HTTP GET or POST requests. Web applications are generally able to access the user input contained in these requests as they would access any other variable in the environment.

***Second-order injection*:** In second-order injections, attackers seed malicious inputs into a system or database to indirectly trigger an SQLIA when that input is used at a later time. The objective of this kind of attack differs significantly from a regular (i.e., firstorder) injection attack. Second-order injections are not trying to cause the attack to occur when the malicious input initially reaches the database. Instead, attackers rely on knowledge of where the input will be subsequently used and craft their attack so that it occurs during that usage.

In the example, a user registers on a website using a seeded user name, such as "admin' -- ". The application properly escapes the single quote in the input before storing it in the database, preventing its potentially malicious effect. At this point, the user modifies his or her password, an operation that typically involves checking that the user knows the current password and changing the password if the check is
successful. To do this, the Web application might construct an SQL command as follows:
queryString="UPDATE users SET password='" + newPassword + "' WHERE userName='" + userName + "' AND password='" + oldPassword + "'"
newPassword and oldPassword are the new and old passwords, respectively, and userName is the name of the user currently logged-in (i.e., ''admin'--''). Therefore, the query string that is sent to the database is (assume that newPassword and oldPas-sword are "newpwd" and"oldpwd"):
UPDATE users SET password='newpwd'
WHERE userName= 'admin'--' AND password='oldpwd'

Second-order injections can be especially difficult to detect and prevent because the point of injection is different from the point where the attack actually manifests itself. A developer may properly escape, type-check, and filter input that comes from the user and assume it is safe. Later on, when that data is used in a different context, or to build a different type of query, the previously sanitized input may result in an injection attack.

1.2) *Example Of SQL Injection Attack:* Following example illustrates application that contains SQL Injection vulnerabilities.

```
1. String login, password, pin, query
2. login = getParameter("login");
3. password = getParameter("pass");
3. pin = getParameter("pin");
4. Connection conn.createConnection("MyDataBase");
5. query = "SELECT accounts FROM users WHERE login='" +
6. login + "' AND pass='" + password +
7. "' AND pin=" + pin;
8. ResultSet result = conn.executeQuery(query);
9. if (result!=NULL)
10. displayAccounts(result);
11. else
12. displayAuthFailed();
```

The above code implements the login functionality for an application. It is based on similar implementations of login functionality that we have found in existing Web-based applications.

The code in the example uses the input parameters login, pass, and pin to dynamically build an SQL query and submit it to a database.

For example, if a user submits login, password, and pin as "doe," "secret," and "123," the application dynamically builds and submits the query:
SELECT accounts FROM users WHERE login='doe' AND pass='secret' AND pin=123. If the login, password, and pin match the corresponding entry in the database, doe's account information is returned and then displayed by function displayAccounts(). If there is no match in the database, function displayAuthFailed() displays an appropriate error message.

F. Experimental Results

After implementing different hacking as well as non-hacking techniques, this system hacks simple web sites

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 3, March 2015

with normal security credentials. It provides access to server database and allow attacker to perform database operations from client machine.

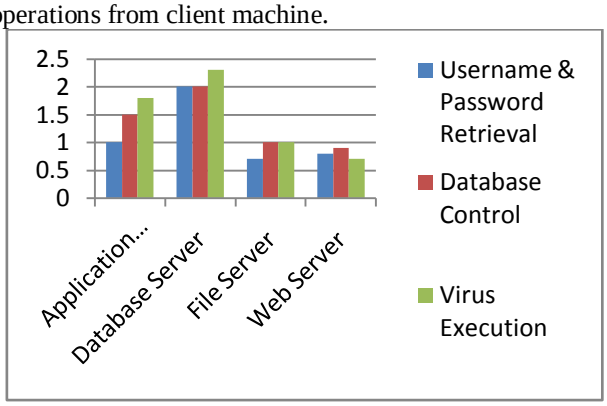

Fig.4. Experimental graph for the system

. Above Figure shows the experimental result of this application upon different servers and provides satisfactory results as required.

IV. CONCLUSIONS

In this paper, we have presented a to view to hack the server which include some hacking as well as non-hacking methods. These algorithms and methods provides efficient way to hack server database. By breaking the network security allow to introduce new and better security framework. The term "Hacking" not only consider for it's illegal activities but also it should be use for strengthen our global network.

ACKNOWLEDGMENT

We gratefully acknowledge H.O.D of computer engineering department of our college for their kind support for this project. We also thank our project guide and co-guide for highlighting our path and their gracious guidance. In last we like to thank all the friends who had given some valuable contribution for this system.

REFERENCES

[1] NeisYangsou, XiaoouTang "*Classi_cation of SQL injection attacks counter measures*" International conference
[2] Chan Liu "*Hacking databases for owning your data*" International conference
[3] G. Antoniol and Y. Gueheneuc, "*Malicious threats vulnerability in instant messaging"* volume 1.10
[4] A.Alexandrov ,P.Kmiec, and K. Schauser Consh "*A confined execution environment for internet computations*"
[5] Karen Scarfone, Miles Tracy "*Guide to general server security*" National Institute Of Sciense and echnology.In Proc.IEEE Intl.Conf. on SQL,2011.
[6] Andrew Colin and Dr.Dobbs, *"Oracle Application Server 10g Security"* April 2009.
[7] A. Marcus, "*Hacking expose*" volume 4.3.In Proc. Intl. Conf. on pages487-499, 2009.
[8] A. Rohatgi, A. H. Lhadj and J. Rilling, "*Storing User Passwords Securely: hashing*" volume 2.1.Journal, June 1996.
[9] Amichai Shulman, "*Top Ten Database Security Threats*", In Proc Intl. Work-shop, pages 180-187, 1998.
[10] J. Wikman, " *Database Independence"*. Nokia Research Center,February1996.
[11] E. G. G. Cattell (ed.) *ODMG-93: "The Object Database Standard"*. Morgan Kaufman, 1994.
[12] Shray Kapoor, *"Session Hijacking exploiting TCP, UDP and HTTP Sessions"*.
[13] William G.J. Halfond, Jeremy Viegas, and Alessandro Orso, "*A Classification of SQL Injection Attacks and Countermeasure"*

DOI 10.17148/IJARCCE.2015.4304