DEFENDING SERVERS AGAINST NAPTHA ATTACK BY USING AN EARLY CLIENT AUTHENTICATION METHOD

CHENG HAN PIN

UNIVERSITI SAINS MALAYSIA

2008

DEFENDING SERVERS AGAINST NAPTHA ATTACK BY USING AN EARLY CLIENT AUTHENTICATION METHOD

by

CHENG HAN PIN

Thesis submitted in fulfilment of the requirements for the degree of Master of Science

JUNE 2008

I dedicate this thesis to my beloved parents, brothers, sister, and Yuan Jin.
For their continuous loves, supports and encouragements.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my deepest gratitude to my supervisor, Associate Professor Dr. Sureswaran Ramadass, for his advices, guidance, helps, encouragements and motivations throughout the progress in completing my research. I would not have chosen this fascinating area of research without his inspiration. I still remember the moment we met and he suggested to me: “Why not Denial of Service attacks”.

This research was supported by University Postgraduate Research Scholarship Scheme (PGD) under the auspices of Ministry of Science, Technology and Innovation (MOSTI), Malaysia. I am grateful to this scheme; the interviewers: Associate Professor Dr. Bahari Belaton and Associate Professor Dr. Azman Samsudin; Also, to Dr. Shahida Binti Sulaiman, Miss Tee Yuan Jin, Miss Cheng Lee Siang, and Associate Professor Dr. Rahmat Budiarto who have helped me in completing the agreement.

Lastly, to all my friends and lecturers, you have enlightened my life, thank you very much.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACK	Acknowledgement in Three-way Handshake
ARP	Address Resolution Protocol
CA	Certificate Authority
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CUSUM	Cumulative Sum
DoS	Denial of Service
DDoS	Distributed Denial of Service
ECAM	Early Client Authentication Method
FTP	File Transfer Protocol
HTTP	Hyper Text Transfer Protocol
HIP	Host Identification Protocol
ICMP	Internet Control Message Protocol
IKEv2	Internet Key Exchange version 2
IPSec	Internet Protocol Security
pTCP	Puzzle TCP
SA	Security Associations
SMTP	Simple Mail Transfer Protocol
SRP	Secure Remote Password
SSH	Secure Shell
SYN	Synchronize in Three-way Handshake
TCB	TCP Control Block
TCP	Transmission Control Protocol
URL	Universal Resource Locator

Melindungi Pelayan daripada Serangan Naptha dengan Menggunakan Kaedah Pengesahan Awal Pelanggan

ABSTRAK

Serangan Naptha bertujuan mengganggu layanan TCP yang ditawarkan oleh sesuatu pelayan dengan menjanakan banyak sambungan palsu terhadap pelayan tersebut. Pelayan sasaran termasuklah pelayan capaian selamat, pelayan mel, pelayan laman web dan pelayan fail. Pelayan-pelayan ini biasanya membuka satu proses atau satu talian untuk setiap permintaan yang ditubuhkan, sama ada permintaan itu sah atau tidak. Sebaik sahaja had maksimum proses atau had maksimum talian tercapai, permintaan baru akan ditutup dan digugurkan. Dengan itu, serangan Naptha juga merupakan sejenis serangan penolakan layanan. Dalam kajian ini, kami mengutarakan kaedah pengesahan awal pelanggan dalam usaha untuk melindungi pelayan daripada serangan Naptha. Kaedah ini menampal jurang di antara penubuhan sambungan dan pengesahan pelanggan, di mana ia wujud dalam pelaksanaan TCP masa kini. Kelebihan kaedah pengesahan yang dicadangkan ialah mudah, cekap dan ia tidak memperkenalkan transaksi tambahan di antara pelanggan dan pelayan. Keputusan daripada mesin ujian kami menunjukkan kaedah pengesahan awal pelanggan adalah cekap dalam menguruskan serangan Naptha sewaktu melayani pelanggannya.

DEFENDING SERVERS AGAINST NAPTHA ATTACK BY USING AN EARLY CLIENT AUTHENTICATION METHOD

ABSTRACT

Naptha attack aims to disrupt TCP service a server provides by generating large amount of forged connections to the server. The targeted server includes secure shell server, mail server, web server and file server. These servers typically create one process or one thread for each established incoming request regardless of whether the client is legitimate or not. Once the maximum process limit or thread limit is reached, new request will be closed and dropped. Hence, Naptha attack is also a Denial of Service attack. In this research, we propose Early Client Authentication Method (ECAM) in defending server that required client login against Naptha DoS attack. This method patches the gap between connection establishment and client validation which appears in current TCP implementation. The advantages of the proposed authentication method are simple, efficient and it does not introduce additional transaction between client and server. Results show that the proposed ECAM is capable in handling the attack on our test machine while continue serving its client.

CHAPTER ONE
INTRODUCTION

1.0 Introduction

There are two known Transmission Control Protocol (TCP) stack attacks, namely TCP SYN attack and Naptha attack. Both attacks prevent legitimate users from using TCP services on a server and therefore, are categorized under Denial of Service (DoS) attacks. Before in depth explanation on Naptha attack is carried out, we first describe the threat of DoS attacks.

1.1 Denial of Service and Distributed Denial of Service Attacks

Denial of Service (DoS) attacks are exploits which could bring services to be temporary unavailable or permanently disabled until recovery actions are taken. Unlike intrusion attack, DoS attacks aim to disrupt services a server provides. These attacks may last for a few minutes to several hours. During this period, the targeted victim is either not responding to any request or experiencing significant drop in performance.

DoS attacks had successfully brought down several well-known web servers in the early of 21st century. Yahoo, CNN, Amazon and e-Bay were among the victims of the attacks (David, 2000). According to Symantec Internet Security Threat Report in 2005: In the first half of 2005, the number of DoS attacks has grew by more than 680% from 119 attacks per day to 927 attacks per day on average (Symantec, 2005); for the next six months, Symantec detected an average of 1,402 DoS attacks per day which is an increase of 51% from the first half of 2005 (Symantec, 2006).

Ever since the first wide spread DoS attack was discovered in 1988 (Mell and Wack, 2000), new exploit emerges in exponential order. Numerous researches had

been done in recent years, and counter measurements were proposed. Some of the techniques effectively mitigate these attacks but none could stop it completely.

DoS attacks could be classified into two major categories: vulnerability attacks, and flooding attacks. In the first type of attacks, attacker exploits system's security hole to force the targeted victim to reboot, crash, shutdown or freeze. Ping of Death, software exploits and teardrop are examples of vulnerability attacks. Normally, a simple system update is sufficient to patch the security hole.

For flooding attacks, attacker consumes all resources on a targeted victim and thus prevents it to continue serving its clients. Resources here could be bandwidth, CPU, memory, process table, buffer or anything that is limited and consumable. Under these attacks, even the finest system could be a victim. In flooding attacks, attacker sends large amount of traffics to the victim with the goal to deplete victim's resources. The victim is unable to differentiate between the attack traffics and normal traffics in this situation, all traffics look alike. The victim could not simply filter out particular traffics because this might drop legitimate users too. This is the main reason why DoS attacks still remain as a threat until today. The situation becomes worse when a few exploits are bundled and wide spread. Anyone could download and launch a large attack without having to understand how it actually operates.

Flooding attacks often involve zombies because one attacker alone is not enough to deplete victim's resources. All the zombies together form a network which is listening to the attacker and they will start flooding the targeted victim upon receiving the correct command. A DoS attack which originates from more than one source is known as Distributed Denial of Service (DDoS) attack. The scale of DDoS attacks depend on the number of zombies that participate in the exploits. With enough zombies, any server could be a victim. Due to its effectiveness, DDoS attacks are

known as major threats in network security communities. More information could be found by referring to Mirkovic et al. (2005) and Handley & Rescorla (2006).

1.2 Transmission Control Protocol (TCP) and Naptha Attack

Transmission Control Protocol (TCP) is a connection oriented protocol, which designed to provide reliable connection over the Internet (Postel, 1981). This protocol uses sequence number to detect lost or duplicated packets and to determine the next expected packet. Retransmission of packet is done automatically when a connection timed out to ensure the user at the other end could receive all information he sent. Most of the services we use everyday rely on TCP, such as web browsing using Hyper Text Transfer Protocol (HTTP), file exchange using File Transfer Protocol (FTP) and email service using Simple Mail Transfer Protocol (SMTP).

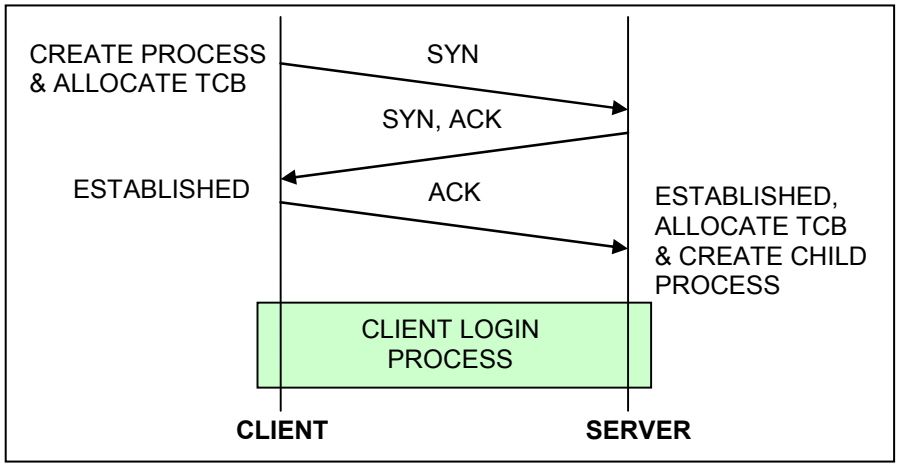

Figure 1.1: Three-way Handshake, Resources Allocation, and Client Login Process

A TCP communication begins with three-way handshake. First, the client (indicator) sends a connection request to server (responder) using a SYN (synchronization) packet. Then the server responses packet with SYN and ACK (acknowledgement) flags on. After the server receives an acknowledgement from the client, connection is established. A new child process will be created at the server to handle client requests and information such as the incoming IP address, outgoing IP

address, incoming port number, outgoing port number and sequence number will be kept in TCP Control Block (TCB). A graphical representation of three-way handshake, resources allocation and client login procedures is shown in Figure 1.1. In current login server communication, client login process starts after the connection establishment. Since the client stays anonymously while resources were allocated, it can be easily exploited and resources can be consumed.

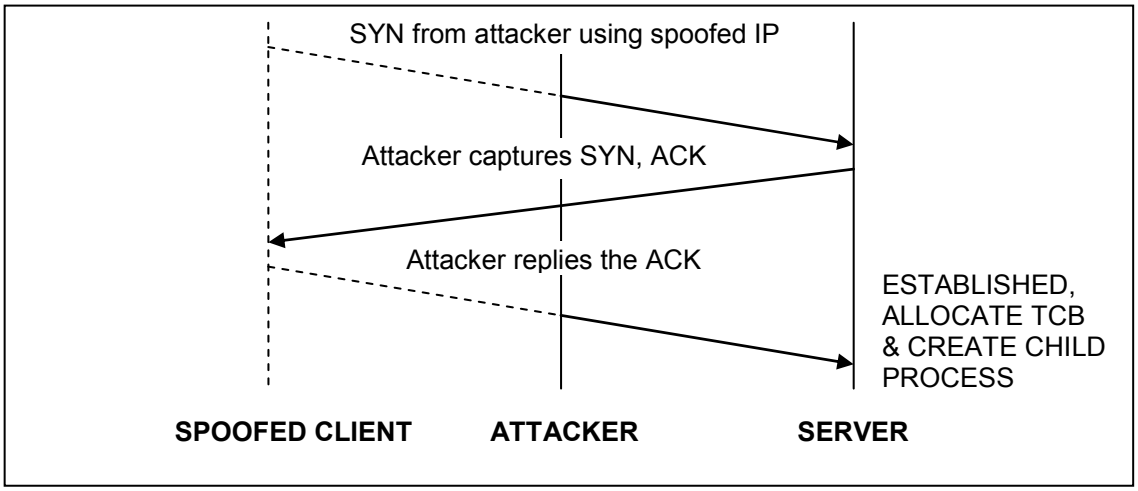

Figure 1.2: Naptha Attack

In Naptha attack, attacker tries to fill up the maximum processes limit by generating large army of spurious connections to the targeted victim and leave them alive until the TCP session time out. Once the limit is reached, server will drop any new incoming connection request. The similarity between Naptha attack and the well-known TCP SYN attack is the disruption of TCP services through flooding. The difference is that they target different resource. TCP SYN attack tries to exhaust the application's backlog while Naptha attack tries to deplete the maximum processes limit of a system. In TCP SYN attack, attacker merely send large amount of crafted SYN packet to victim, however, in Naptha attack, the procedures involved are slightly more complex because attacker must complete all bogus connections he made as illustrated in Figure 1.2. The figure shows that the attacker sits between forged client and server during Naptha attack. The spoofed host does not actually exist, but the server who cannot validate

this will still establish the connection and allocate resources to the peer. Server that creates new thread for each incoming connection denies new incoming request as well when the maximum thread limit is reached.

1.3 Research Problem

Although Naptha attacks were discovered and documented since 2000 (CERT/CC, 2000), the threat still exist. There is no solution to prevent or stop these exploits thoroughly. Naptha attack can effectively disable a server for a period of time depends on the attacker. Since all attack connections to the server are completely identical to normal legitimate connections, it helps to cover the attacker's track and makes prevention harder to achieve.

During Naptha attack, it fills up the victim's processes limit with crafted hosts. Recent operating system tries to mitigate this problem by increasing the total number of supported concurrent processes to several thousands and by closing idle connections quickly. While this tactic seems to work fine, the threat remains. Even though the server could continue to serve during the attack in this situation, CPU power and memory usage are wasted to process bogus data and this directly decreases server performance.

Internet services nowadays are vulnerable to Naptha attack despite of whether the services required user login or not. We proposed Early Client Authentication Method (ECAM) to defense servers against Naptha attack. Login information is appended to the three-way handshake to allow server to identify the client at connection establishment stage. Our primary goal is to prevent resources allocation to any forged connections generated by Naptha attack.

1.4 Research Objectives

The objectives of this research are:

1. To examine the behavior and impact of Naptha attack.
2. To propose and develop algorithm for ECAM.
3. To implement and evaluate ECAM.

The first objective of this research is to examine the behavior of Naptha attack. This goal focuses on how the attack exploits the TCP stack implementation, the target of Naptha attack and its damages to a system.

The next objective is to propose and develop ECAM algorithm. The goal aims to prevent resources consumption to unidentified user by authenticate all incoming requests at the early stage. Any bogus connections which failed the authentication process will not be entertained, and thus fails the attack.

The third objective is to implement and evaluate ECAM. We used OpenBSD, an open source operating system, in our research. The default system and ECAM system will be tested against Naptha attack. It is expected Internet services that required user login will become immune to the attack on an ECAM enabled system.

1.5 Significant of the Research

This research aims to protect TCP services that required user login against Naptha DoS attack. The newly proposed ECAM will verify the client before the system proceeds to serve him. Bogus incoming requests will eventually be dropped and consequently, all active processes and threads are maintained only for clients who had login successfully. This method ensures Naptha attack that consumes maximum process limit using forged TCP connection will never be succeeded.

1.6 Thesis Overview

This thesis is organized into six chapters. Chapter 1 presents the background of this thesis. It starts by defining Denial of Service attacks and Distributed Denial of Service attacks. Then, we describe the current TCP stack problem and Naptha attack. The research problem and research objectives are explained next. We discuss the significant of our research in section 1.5.

Chapter 2 is a literature review of related protection methods against Naptha attack. These methods could generally be divided into two categories, which are the filtering mechanism and the prevention mechanism. We concentrate in the prevention mechanism because our method falls into this category. We also discuss some other immediate defensive methods.

Chapter 3 details our proposed ECAM to protect servers against Naptha attack. First, we describe the design of ECAM and the advantages of it compared to the current FTP authentication method. The security considerations in ECAM and their counter measurements are discussed next in the chapter. The methodology to carry out this research is also presented.

Chapter 4 details the ECAM implementation at kernel level and application level of both ECAM client and ECAM server. Chapter 5 presents the results and discussions of ECAM system and current default system under Naptha attack. Chapter 6 is the final chapter of this thesis. It gives a conclusion of ECAM and it also outlines the possible future works to enhance the proposed ECAM.

CHAPTER TWO
LITERATURE REVIEW

2.0 Introduction

We described the threat of Naptha attacks in the previous chapter. In this chapter, we discuss the related defence techniques from the perspective of TCP stack implementation. Researches and counter measurements in this area generally could be divided into two categories. The first category concentrates on detection and filtering mechanisms. During network monitoring, any detected attack attempt will activate the filtering action. The second category concentrates on prevention mechanisms. During the development process, the Internet protocol is reviewed and enhanced in order to prevent future exploit on the specific protocol. Figure 2.1 below is a graphical representation of common defence techniques in TCP stack attacks. Our research falls into the second category.

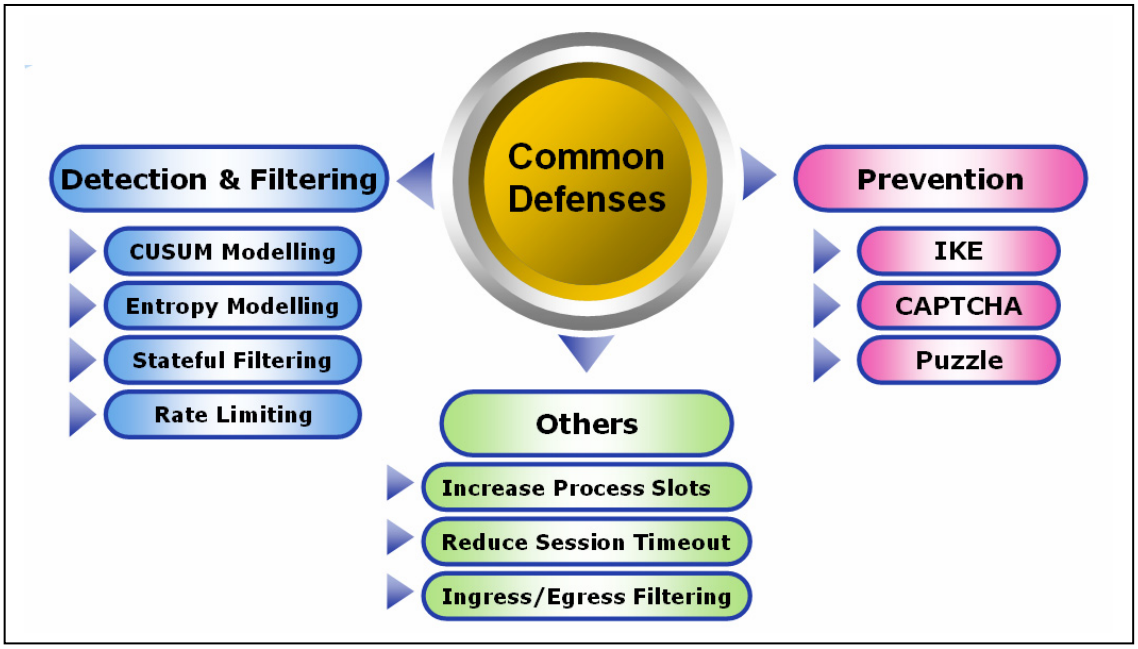

Figure 2.1: Common Defence Techniques in TCP Stack Attacks

2.1 TCP Stack Attacks Detection and Filtering Mechanisms

TCP stack attacks detection techniques often involve statistical modelling of incoming and/or outgoing traffics. The traffic collected will be compared to previous data in real time and any abnormal behaviour will trigger the alarm. This mechanism is suitable to detect large scale of TCP stack attacks in which the attacker tries to flood and congest the victim. Two of the most widely used detection methods are entropy and Cumulative Sum techniques (CUSUM) (Feinstein et al., 2003). Entropy allows the server to measure the randomness of the traffic while the CUSUM can detect any abrupt change to the traffic pattern. For example, these methods could be used to detect sudden increment of SYN packet, ICMP (Internet Control Message Protocol) packet and ARP (Address Resolution Protocol) packet. Usually, the detection focuses on one or more fields in the IP header as targets of monitoring, such as the port number, IP address and TCP flags.

DoS attacks detection is not complete without proper reactive action. The reactive mechanisms employed depend heavily on the detection techniques and the location of the action took place. For example, a stateful filtering mechanism is suitable to be implemented at the host base because it required deep inspection during detection and precise instructions to filter the traffic. Even a tiny mistake will drop legitimate users. Another common technique used by the upstream router is rate-limiting mechanism which is low resources intended. This approach drops traffic exceeded certain threshold.

Naptha attack detection was carried out by Information System Technology group (IST) of MIT (Massachusetts Institute of Technology) Lincoln laboratory in 1998 and 1999, categorized under DoS – process table attack. However, the purpose of the evaluation on process table was not to protect the server against it but to check against the buggy code that might create many connections in rapid fire sequence (Haines et

al., 2001). The following quote is picked from the technical report by Haines et al. The report stated that the only way to detect process table attack is by monitoring the total number of connections reaches the server at a time.

> *"Because this attack consists of abuse of a perfectly legal action, an intrusion detection system trying to detect a Process Table attack will need to use somewhat subjective criteria for identifying the attack. The only clue that such an attack is occurring is an unusually large number of connections active on a particular port. Unfortunately an 'unusual' number of connections is different for every host … …" (Haines et al., 2001).*

Normally, the challenges faced by this intrusion detection are false positive and false negative. Also, the total number of connections handled by a server is different from one domain to another, as well as from time to time. Network administrator normally set the total number of allowed connections to several percents higher than usual incoming requests before the server triggers the alarm. False positive is a condition where the detector sounds the alarm but an attack does not take place; this happens when there is sudden increase in legitimate users accessing services a server provide, called flash crowd. On the other hand, false negative is a condition where the detector could not detect a present attack; this occurs if the rate of attack is lower than the detection threshold. In order to bypass this detection, the attacker could run a series of training to fool the server to increase the threshold limit so that the actual attack stays undetected as long as the attacker hold the attack rate well below the limit.

2.2 TCP Stack Attacks Prevention Mechanisms

In TCP stack attacks prevention, developers modified the TCP implementation to make the attacks completely unsuccessful or less destructive. Although prevention is effective, it might require kernel recompilation. The algorithm developed is often attack specific and cannot be used to defend against other type of DoS attack. However, this method does mitigate some DoS attacks that are hard or impossible to be differentiated

or filtered from daily traffics. Furthermore, well defined prevention technique does not produce false positive or false negative.

Since Naptha attack takes place before any authentication process starts, attacker normally leave little or no track to be traced. Recent researches in this area concentrate on lower layer protection to mitigate similar DoS attacks before a connection is established. Some examples are: Internet Key Exchange (IKEv2) used by Internet Protocol Security (IPSec) and computational intensive puzzle used by Host Identification Protocol (HIP).

We describe Internet Key Exchange (IKEv2) in section 2.2.1 and computational intensive puzzle in section 2.2.2. Both techniques are designed to be implemented at network layer and transport layer respectively to protect system against DoS attacks. CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) at application layer is explained in 2.2.3, while section 2.2.4 briefly discusses other defensive methods.

2.2.1 Internet Key Exchange (IKEv2)

Internet Key Exchange (IKEv2) is a protocol used to negotiate, set up and maintain security associations (SAs) (Kaufman, 2005). A SA contains shared security information needed by IPSec to provide confidentiality, data origin authentication, anti-replay, connectionless integrity, and limited traffic flow confidentiality (Kent & Seo, 2005). Figure 2.2 shows the initial exchange of IKEv2 and the notations are listed in Table 2.1. In the first two transactions, the initiator and the responder agree on a set of cryptographic suite, and the following messages flow between them will be encrypted and integrity protected. Integrity protection assured that unauthorized alteration of data in the middle of transaction will be detected.

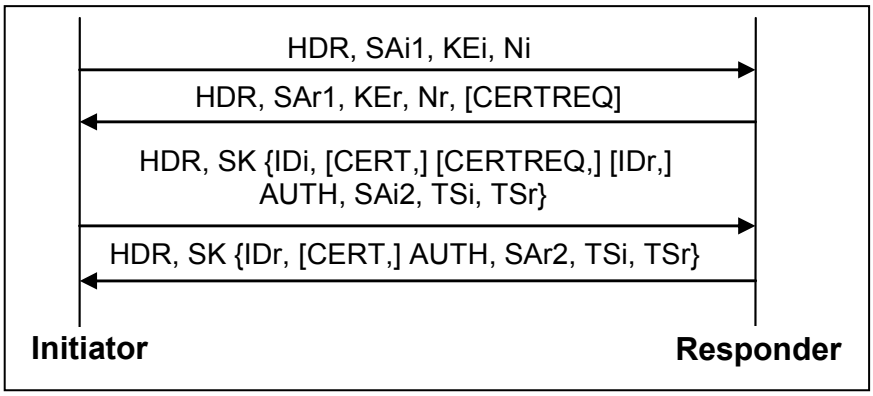

Figure 2.2: Initial Exchange of IKEv2

Table 2.1: Notation in IKEv2

Notation	Description
AUTH	Authentication
CERT	Certificate
CERTREQ	Certificate Request
HDR	IKE Header
ID	Identification
KE	Key Exchange
Ni, Nr	Nonce
SA	Security Association
SK{ … }	Payload in brackets are encrypted and integrity protected
TSi, TSr	Traffic Selector

In the third and forth transactions, identification payload and optional certificate payload are used together with authentication field to assert peer's identity. This is very important because when we browse the Internet, there is no guarantee that the URL (Universal Resource Locator) we typed will always point to the location we want. Sometime it gets redirected to somewhere else. Normally, an attacker will try to redirect the user to a location that mimic the original web site (called a phishing site), so that when the user keyed in the username, password and credit card number, the attacker could steal them. The certificate is used to verify the genuineness of identification payload and it relies on the third trusted person, called Certificate Authority (CA) to provide the verification.

IKEv2 together with IPSec provide solutions to many security problems that haunted the network administrator for many years. However, a network administrator should not rely on IPSec completely, because there is no guarantee that all hosts inside the network will never be compromised. A host could easily be inflected through external source such as thumb drive, or laptop which plugged into various network. An attacker could also get through when the user browses some malicious web site, opened email attachment or received file from peer during chatting. The inflected host that we mentioned will be able to attack any machine through an IPSec link because IPSec blocks traffic based on rules in the configuration and it does not know how to block an inflected host dynamically.

Current deployment of IKEv2 requires the privileged root user to manually set up the configurations such as IP address, encryption algorithms, and security policy at both initiator and responder in order to communicate securely. There would be no problem if IKEv2 were to be established between two known gateways permanently, such as between two campuses or between two offices remotely. In a client-server communication however, if a user wishes to login into an email server but the server is not included as an allowed IPSec peer; then, there will be no protection on that connection. Password, credit card number and all other information transmitted will be sent in plaintext and be revealed to the attacker in this case.

2.2.2 Computational Intensive Puzzle

Computational intensive puzzle is a challenge-response test aims to be executed automatically between the server and client. The solution to the puzzle acts as a proof of work or proof of existence challenge. There are some different proposals on the implementation of computation intensive puzzle as suggested by Host Identification Protocol (Moskowitz et at. 2007), Client Puzzles (Juels & Brainard, 1999) and Puzzle TCP (McNevin et al., 2004). Generally, the server sends challenge on

second handshake and the client reply in third handshake, as illustrated in Figure 2.3. Initially, the client has to send a service request to the server, then the server reply with a challenge or puzzle back to the client. Basically, the host machine must do some computation, such as add, subtract, multiply, divide and modulus, in order to solve the puzzle and send the result back to the server. The client will be granted permission to access the service if the result is correct or be rejected if the result is wrong. As explained in chapter 1, DoS attacks aims to deplete the TCB so that it drops further service request. The purpose of this puzzle challenge is to delay the automate attack generated by an attacker because it must solve the puzzle in order to have an entry in TCB.

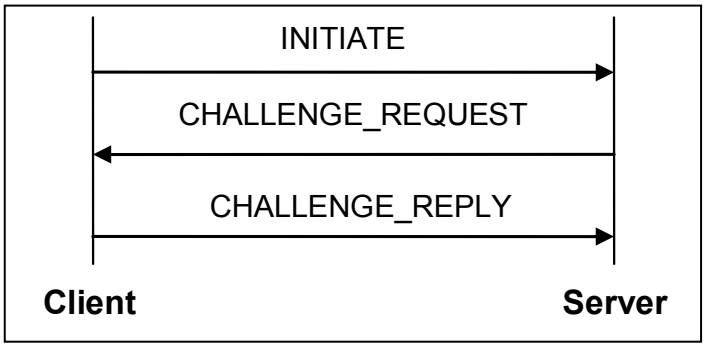

Figure 2.3: Exchange of Computational Intensive Puzzle

The limitations faced by this method are the determination of which puzzle to be used due to the processing power of the personal computer nowadays vary in wide range. Some machines can solve a puzzle rapidly whereas some are slower. The system might drop legitimate client who uses slower machine, either because the TCB is filled up by the other fast machines or the session timed out before the client could solve the challenge.

2.2.3 Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA)

CAPTCHA is an application layer challenge-response test used to prevent automated attack. This test consists of distorted alphabetical letters which is considered hard to be recognized by machines. The user must enter the correct phrase as displayed in order to proceed. Some examples are shown in Figure 2.4. This protection technique has been widely implemented in public services such as during creation of new email account at Yahoo, Gmail and Hotmail.

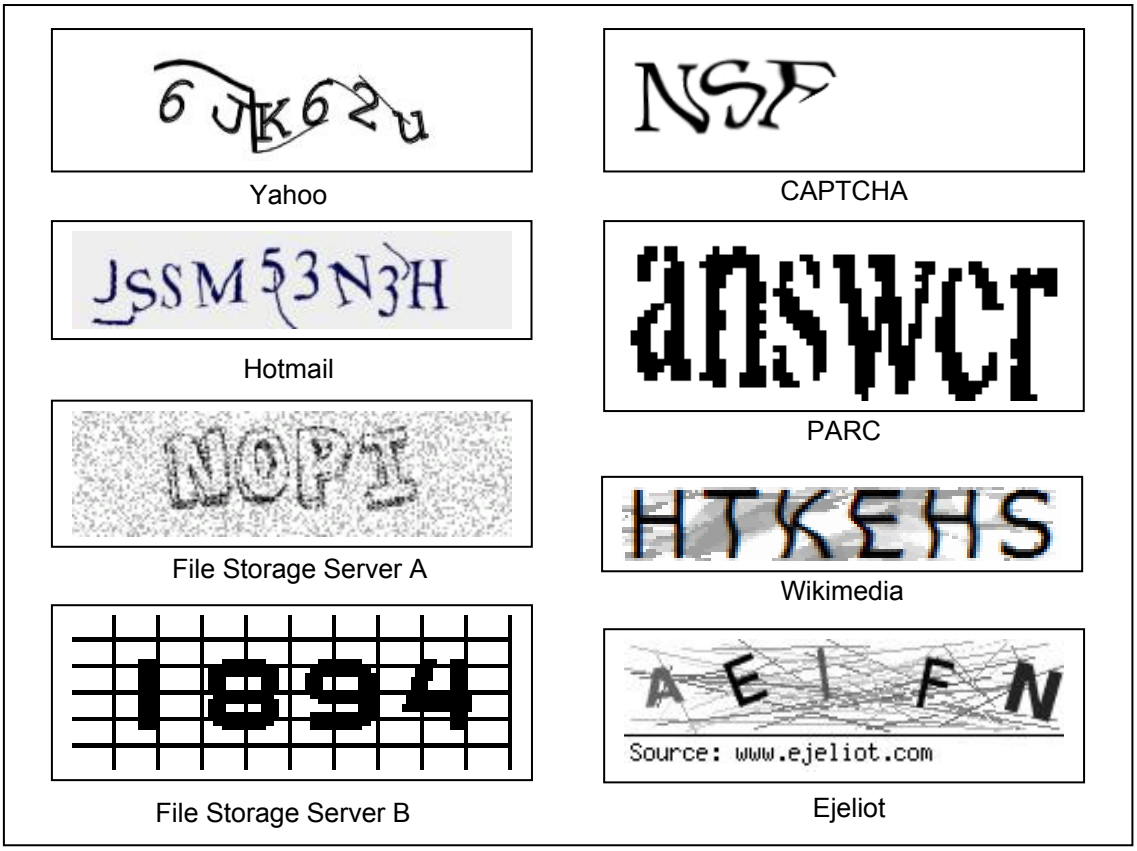

Figure 2.4: Example of CAPTCHA Used on Web Server

CAPTCHA successfully introduces some degree of difficulties to the attacker. The effectiveness of CAPTCHA in differentiate crafted connection from daily traffic depends on the algorithm employed. There are many CAPTHA algorithms already broke by the attackers and there will be more.

2.3 Other Defensive Methods

Other defensive methods include raising the total number of allowed connections to increase the survivability of server and shorten the connections timeout so that the idle or not responding connections could be terminated faster. Besides these, ingress filtering makes sure the incoming packets are actually from the networks that they claimed to be from and egress filtering prevents unauthorized traffic from leaving the internal network (Ferguson & Senie, 2000). Nevertheless, the first two techniques raise other problems, such as longer time to locate the TCP Control Block when the connection table is large and the server might drop legitimate clients if a connection timed out too fast. Also, the global deployment of ingress and egress filtering is neither guaranteed nor likely while the attacker can use compromised zombies to bypass the filtering (Eddy, 2007; Mirkovic et al., 2004; Faber et al., 1999).

2.4 Summary

We have reviewed some related defensive methods in this chapter, which are, IKEv2 with IPSec, computation intensive puzzle and CAPTCHA. The summary is listed in Table 2.2. Detection and filtering mechanisms limit the attack traffic; IKE and IPSec provide a set of security suite including authentication to identify the remote peer; Computational intensive puzzle and CAPTCHA protect server against automated attack.

However, the threat of Naptha attack still exists due to the limitations of the defence techniques or Naptha attack is outside the coverage of a protection schemes. Although a system administrator could minimize the impact of Naptha attack through proper configurations and complicated network surveillance system, we proposed a simple and transaction saving method to guard servers against Naptha attack in the next chapter.

Table 2.2: Summary of Related Defensive Methods

Method	Strength	Limitation
Detection and Filtering	• Detect almost instantly any statistical change to incoming and outgoing traffic. • Normally the action taken is rate limiting.	• Since the attack traffic could not be isolated, Naptha attack still can get through the firewall. • False positive and false negative
IKE and IPSec	• Provide confidentiality, data origin authentication, anti-replay, and connectionless integrity.	• Security features depend on system configuration which controlled by system administrator, not end user. • It does not identify and block attack traffic from getting through IPSec.
Computational Intensive Puzzle	• The client must solve a puzzle in order to get access. • This slow down the attacker from filling up the resource rapidly.	• The system might drop legitimate client who uses slow machine.
CAPTCHA	• Generate test that is hard to be solved by machines and hence it prevent automated attack.	• Many algorithms used to generate the test already broke by attackers.
Increase Process Slots	• Increase the survivability of server against Naptha attack.	• Performance drops when the TCB grows.
Reduce Session Timeout	• Clear idle connection faster.	• Legitimate client get disconnected from server frequently.
Ingress/Egress Filtering	• Make sure the incoming traffic from router is from where it claims to be from. • Prevent unauthorized traffic from leaving the network.	• Global deployment is neither guaranteed nor likely. • Attack from zombies using valid IP address is not filtered.

CHAPTER THREE
THEORETICAL FRAMEWORK

3.0 Introduction

We have already seen that the root cause of successful Naptha attack is resources allocation to all incoming requests. Those requests were not verified nor identified first before the server allocates TCB to them. The challenge to Naptha attack protection is to differentiate legitimate traffics and attack traffics from the daily packets flow. In this chapter, we take FTP as an example to illustrate the problem in current TCP stack implementation, then, we propose Early Client Authentication Method (ECAM) to overcome the problem by identifying each and every request during connection establishment and to isolate the Naptha attack traffics. Advantages and security considerations regarding the ECAM are also discussed in this chapter.

3.1 File Transfer Protocol (FTP)

The main objective of FTP is to enable file sharing among hosts across the Internet (Postel & Reynolds, 1985). We concentrate our discussion on the initial exchange between FTP client and FTP server, the weaknesses in FTP and how the attacker launches Naptha attack on the server. Figure 3.1 illustrates the transactions between FTP client and FTP server in details. The first three transactions is the three-way handshake we described in section 1.2. It followed by the FTP user login process. When the client's machine received server ready message, it prompts user for login name. Next, the server asks the client for password of the login name. Finally, the server sends welcome message and login successfully message to the client if both the username and password are correct. Notice that the server acknowledges all messages sent from client, so does the client, who acknowledges all messages sent from the server. Acknowledgment could be made cumulatively, which acknowledges

two or more messages from the peer, as did by the client at the end of the login process.

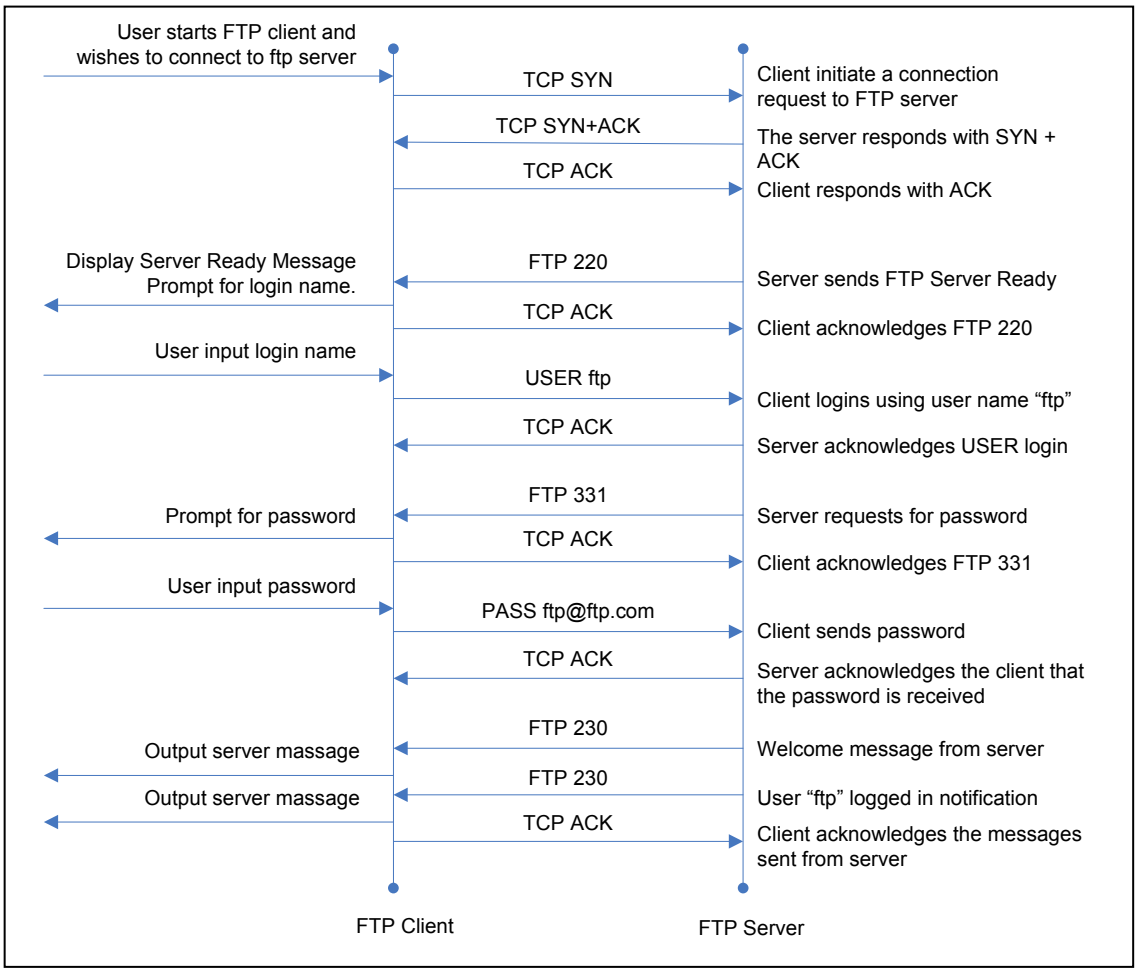

Figure 3.1: Three-way Handshake and FTP Login Process

When the FTP server starts, it creates a process which always waiting to accept new client connection. For each accepted connection, the server spawns a new child process which will handle all server-client operation, such as client login process, file transfer and client session termination. The purpose of child process spawning is to support multiple concurrent client requests at a time and it is created once the three-way handshake is completed. Figure 3.2 illustrates the state of FTP server accepts new connection and spawns new child process when a connection is established. The problems are there is a limit to the maximum concurrent processes that could co-exist

on a machine and that the server did not identify the peer first before spawning new process. This means, one attack connection request also consumes one process slot. The server could not simply terminate a connection within a short time frame because it must wait for reply from the client during client identification process. The server rejects new connection request when the process table is full. An attacker could fill up the table easily by flooding the server using only connection requests, and this is what happening during Naptha attack. There is not much things that a network administrator could do because of the similarity of attack traffic to the normal traffic.

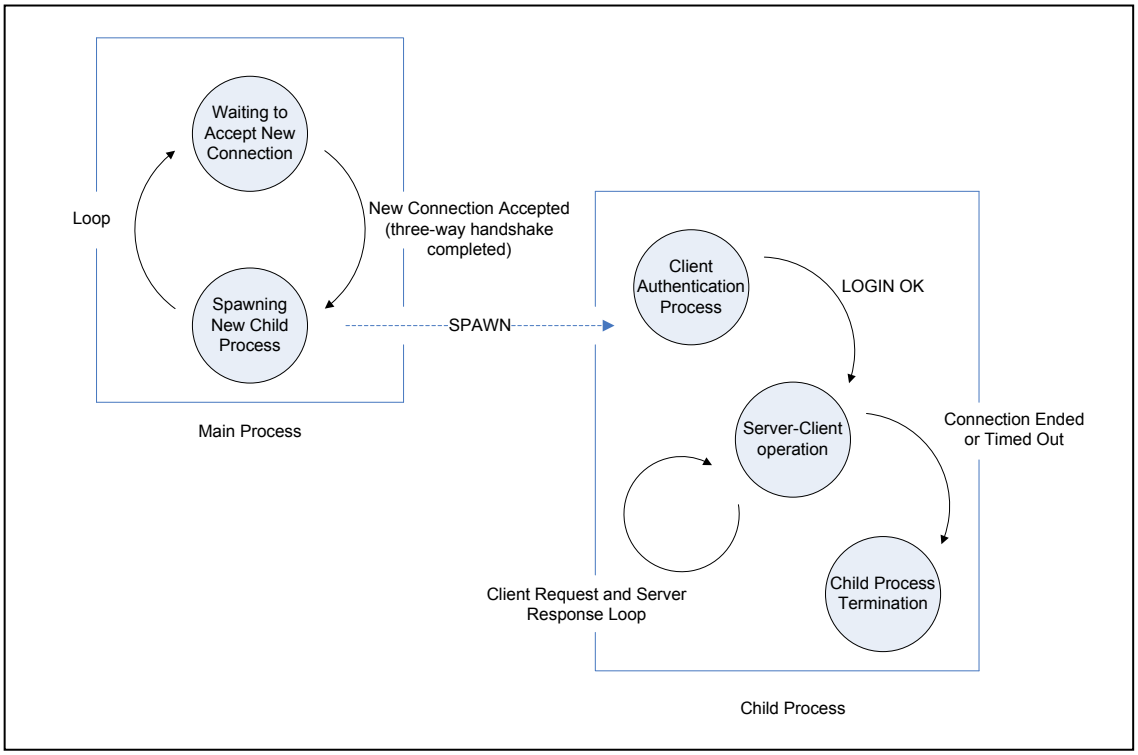

Figure 3.2: State of Current Server when A New Connection is Established

3.2 Early Client Authentication Method (ECAM)

In order to distinguish between the attack connections and legitimate connections, there is a need to verify the genuineness of all incoming requests. We propose Early Client Authentication Method to include login information in the third handshake of connection establishment. This action is completely legal and it does not

violate Transmission Control Protocol stated in RFC 793 (Postel, 1981). Figure 3.3 illustrates the idea of ECAM and Figure 3.4 shows the advantages of ECAM.

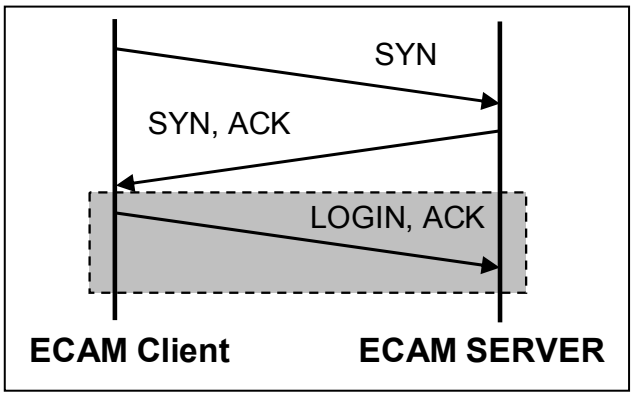

Figure 3.3: Early Client Authentication Method

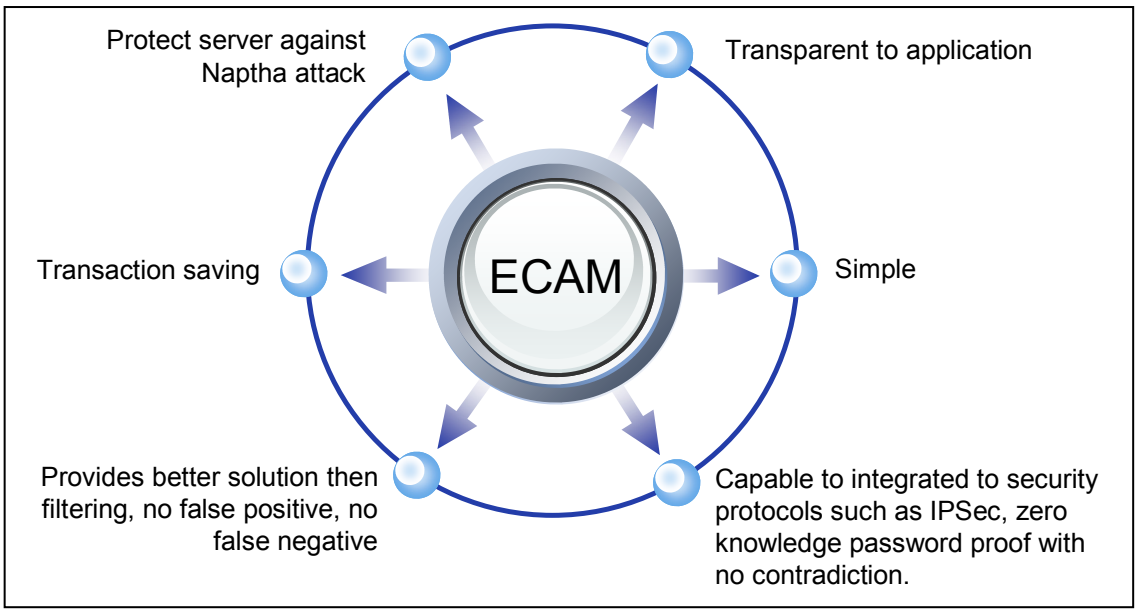

Figure 3.4: Advantages of ECAM

There are many advantages of ECAM. First is the protection against Naptha attack, which is our main purpose. Naptha attack launches a DoS attack by depleting the free process slot using only simple connection requests. For that large amount of connection requests, the server could not simply terminate them. As shown in Figure 3.1, the server must wait for human responses twice in order to verify the peer, once for the login name and another for the password. The waiting time lasted for at least a

few seconds each, which creates a ‘golden period’ for the attacker to launch a flooding attack. During this period, the server could not drop that connection. With ECAM, each and every request must be appended with login information. Since there is no waiting time during the login period, as shown in Figure 3.5, and no unauthorized access to the process table, we can assure that the Naptha attack will never be succeed.

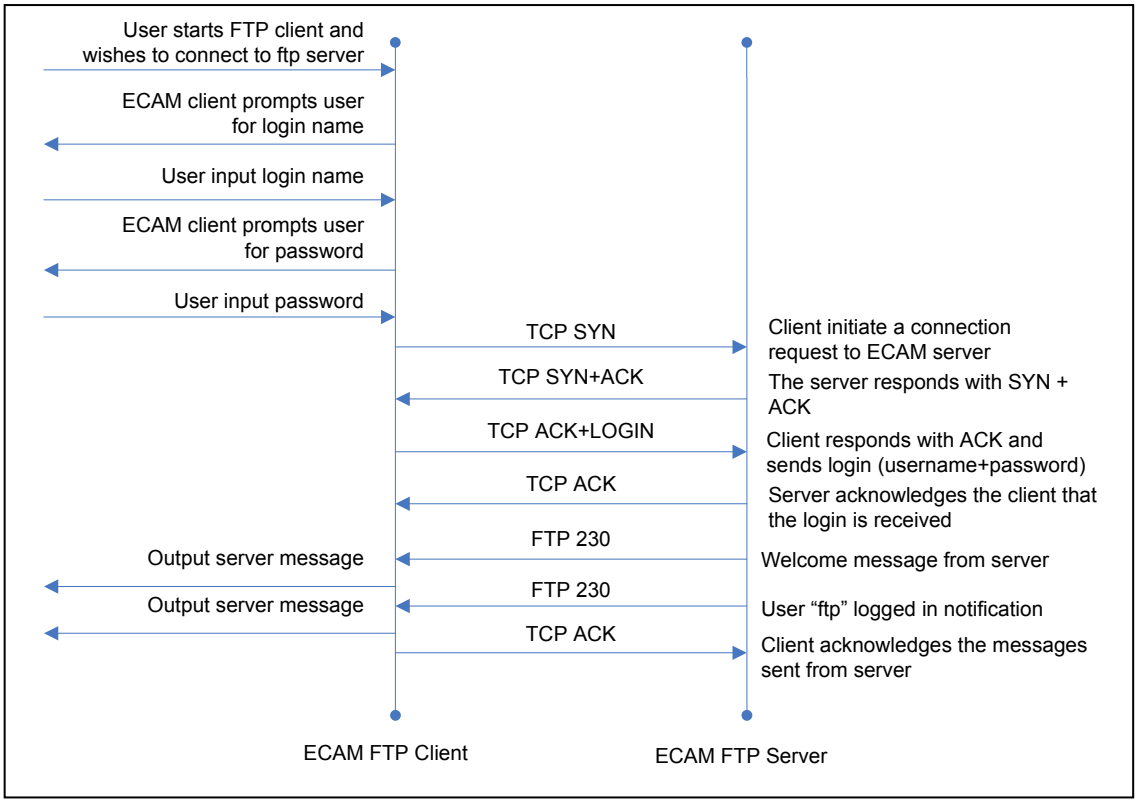

Figure 3.5: Three-way Handshake and FTP Login Process using ECAM

Figure 3.5 shows the three-way handshake and FTP login process using ECAM. By comparing Figure 3.5 to Figure 3.1, we see that current FTP uses 14 transactions to login the system while FTP with ECAM uses only 7 transactions, which is half of the original FTP. Obviously, ECAM consumes much less bandwidth. If both the FTP and FTP-ECAM were given a limited bandwidth, FTP-ECAM system could certainly serve more clients. We gain better performance in FTP-ECAM.

As mentioned in Chapter 2, statistical modelling and filtering mechanisms are not able to isolate or stop Naptha attack. Moreover, those mechanisms might produce false positive and false negative results. Also, the threshold setting to trigger the filtering action is different from site to site and it is possible to be tricked by an attacker. ECAM on the other hand, does not need extra computational power to model the incoming and outgoing traffic. There is no threshold setting too, ECAM just stays in the system, identify the requests and isolates the attack.

ECAM is implemented at the kernel of an operating system. Any application that uses TCP/IP, such as email server and email client; file storage server and file storage client; web server and we client, could take advantage of it. Network programmer does not have to write their own function for each program in order to prevent Naptha attack. He/She just needs to enable the ECAM communication at the TCP socket and the operating system will handle the attack silently.

By inspection, ECAM is simple and efficient. It uses less computation and transactions than normal TCP/IP communication but yet, ECAM is capable to defeat the Naptha attack. It does not employ long mathematic to judge if an attack has took place. It simply identifies the client at the beginning of a connection. Our intention is not to complicate the problem, but to solve the attack effectively.

ECAM could be used with any other security protocol which tries to provide secure communication across the Internet without contradiction. Currently, the Secure Remote Password (SRP), a zero knowledge password proof security protocol, negotiates a security session with the peer after the three-way handshake, which is vulnerable to Naptha attack. The vulnerability is not due to the protocol itself, but where it takes place. The negotiation could actually be carried out earlier by integrated with ECAM to protect the server against the attack. There is no contradiction because the

area of research is different. SRP is a protocol to allow the server to identify the client without having the client to actually send out the password while ECAM concentrates on Naptha DoS attack.

Figure 3.6 illustrates the state of ECAM server when a new connection is established. Previously, we mentioned that the current server could not terminate an unauthorized connection because the client might in the process of login the server. In ECAM however, the state is eliminated. The username and password is collected by the ECAM client in the early stage and the server could proceed with client authentication process immediately without client input. In this way, we can assure that the process table will never be filled by unauthorized connection.

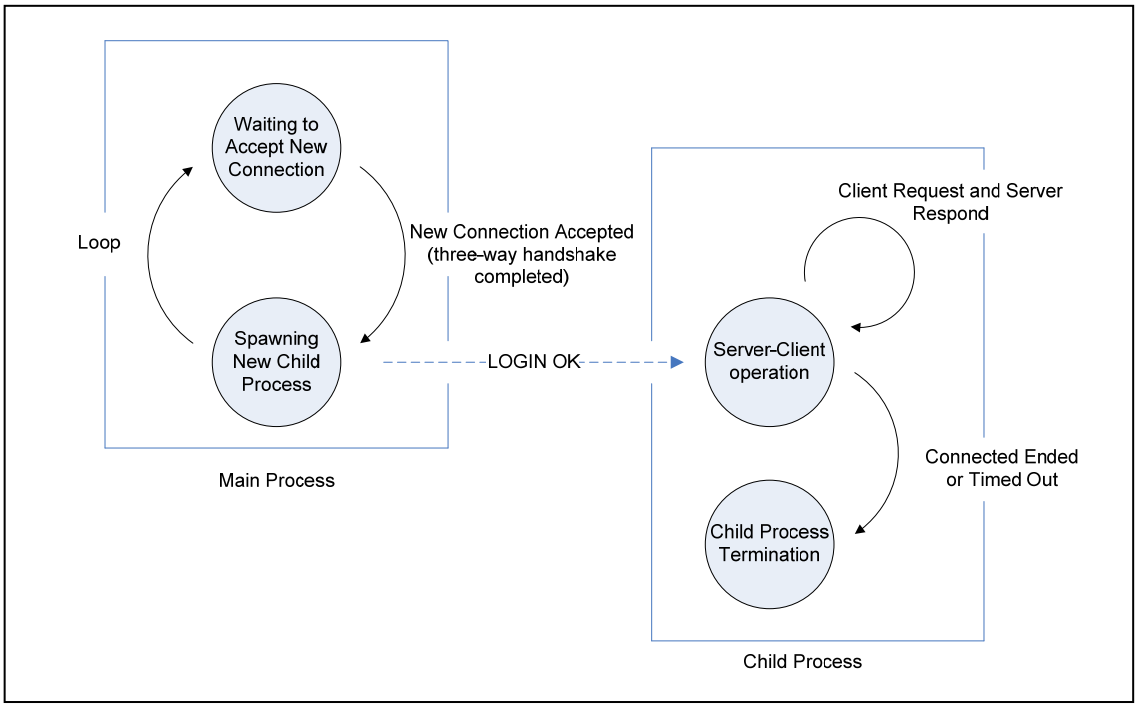

Figure 3.6: State of ECAM Server when A New Connection is Established

3.3 Security Considerations

The login information sent from client to server is in plaintext form. This means, if an attacker could grab the third packet in three-way handshake, he stole the access