SECURITY ISSUES OF THE DYNAMIC PROXY FRAMEWORK

Choong Khong Neng[1*], Borhanuddin Mohd Ali[1], Veeraraghavan Prakash[1], Elok Robert Tee[2] and Yee Yoke Chek[3]

Received: Nov 26, 2002; Accepted: Dec 12, 2003

Abstract

The security issues of a proxy-based service creation and content delivery framework called the Chek Proxy Framework (CPF) was dissussed. CPF is an application level approach that provokes the use of client machine to host at runtime, an intermediate object called the Dynamic Application Proxy Server (DAPS) to offload both the central server and the network backbone. However, introducing DAPS on the client machine to provide intermediary active services breaks the end-to-end nature of the communication, and thus triggers additional security breaches. These include the threat of executing malicious DAPS, the lost of data integrity and illegal redistribution of contents. A security framework to tackle these security breaches was presented, with an objective to derive a secure CPF hand-shaking setup protocol, based on analysis conducted on a video broadcasting system.

Keywords: Chek Proxy Framework; CPF, application level active network, proxy security, agent security, fingerprint, watermarking, layered coding

Introduction

Proxy has been a popular solution for extending the Internet architecture to cope with problems, such as the scarcity of server and network resources, traffic congestion, network heterogeneity, and diverse application needs (Fahmi *et al.,* 2001; Ghost *et al.,* 2001). It has been used initially to masquerade client connections and cache contents, to recent practice on compressing, transcoding, and distilling contents, given the capability of hosting dynamic application objects (Ghost *et al.,* 2001; Smith *et al.,* 1999; Yatin, 2000). However, the merits of deploying proxy do not come without deficiencies. By breaking the end-to-end nature of the communication, proxy is regarded much more difficult to guarantee end-to-end security, and even infeasible in certain cases, such as to distill contents while assuring confidentiality (Portmann and Seneviratne, 2000).

Deriving security measures for proxy should not be started without a clean segregation

[1*] *Department of Computer and Communication, Faculty of Engineering, University Putra Malaysia, 43400 UPM Serdang, Malaysia. E-mail:choongkn@hotmail.com or choongkn@yahoo.com*

[2] *Department of Communication Technology and Networking, Faculty of Computer Science University Putra Malaysia, 43400 UPM Serdang, Malaysia*

[3] *Motorola Multimedia Sdn BhdGlobal Software Group, 16th Floor Menara Luxor, 6B Jalan Persiaran Tropicana, 47410 Petaling Jaya, Selangor, Malaysia*

[*] *Corresponding author*

of proxy. In proxies could be generally classified into either the network/transport enhancement proxies, or content adaptation proxies. (Portmann and Seneviratne, 2000). In this study, however, we classify proxies based on their configurations and operations, resulting both the static and dynamic proxy categories, before differentiating them further based to their functionalities as in (Portmann and Seneviratne, 2000). Proxy in static category refers to the conventional cache proxy that is usually manual-configured, and installed at fixed and dedicated host. Dynamic proxy denotes to host that works cooperatively with the server or other proxies to perform tasks. Further, it has the ability to host mobile codes and thus achieves better scalability and flexibility than the static arrangement. Details on classifying proxy can be found in (Choong *et al.*, 2002). With such classification, we believe that the security issues of proxy could be studied better in an incremental approach, i.e. from the initial categorisation into the specific OSI layer that the proxy works at, and to the type of functionality the proxy offers.

In relevant to the requirements of CPF, attention is paid on addressing the security issues of dynamic proxy at the application layer. Unlike working at the network layer, studying at the application layer renders the security challenges more tractable, as distinctive application semantics can be tailored to derive specific security policies more efficiently and effectively (Portmann and Seneviratne, 2000; Brown, 2001). Given that deploying network layer proxy services, such as Active Network (AN) is unrealistic (Ghost *et al.*, 2001), we believe that deriving security solutions at the network layer seems less exploitable and appropriate, and hence we focus on the application layer solutions.

The flexibility of allowing object/code from various sources to run on the routers (as in AN), or on the end-hosts (such as ALAN (Ghost *et al.*, 2001), ScatterCast (Yatin, 2000) and CPF) introduces numerous security concerns that have been addressed in the context of safe languages (Hicks *et al.*, 1998) and restricted execution environment (as in mobile agent technology) (Ghost *et al.*, 2001). Less effort is made on studying the security breaches and possible impacts in an end-to-end perspective, given the presence of intelligent intermediary objects such as proxies and active routers, which are capable of modifying any packet in transit. In this paper, we present a framework that deals with the common and unconventional threats in a dynamic proxy environment, of a system called CPF.

This paper is organised as follows. Section 2 provides the backgrounds and literatures related to the security issues of proxy. Section 3 gives a brief introduction of CPF and its associated threat model. The framework that addresses the security matters of CPF is described in Section 4. Section 5 summarises the paper, followed by our ongoing works.

Related Works

Security Issues of Proxy

This section discusses various proxy security features under both the static and dynamic proxy category. (Table 1)

Table 1. Checklist of the security measures in both the static and dynamic proxy.

	Authenticity	Confidential assurance	Aninymity guarantees	Secure OS	Restricted EE	Safe language	Contents originaliry	Delivery tracing
Static	✓	✓	✓	✓	✗	✗	✗	✗
Dynamic	✓	✓	✓	✓	✓	✓	✓	✓

The first four columns are the basic security measures readily incorporated into the most conventional static proxy systems. The "ticks" at the first four columns are allocated given that the proxy is non-malicious (since it was statically allocated). The subsequent four columns are additional requirements imposed on the dynamic proxy systems, where both the proxy object (which is mobile) and the proxy host (machine) are regarded non-trustable.

Authentication is commonly conducted with the use of cryptography, to guarantee that every connection to any external objects (such as the central server) is validated. Confidentiality is preserved by means of encryption on the data streams. Anonymity is an inherent feature available within the proxy system, wherein packets going through a proxy (out-bound) are disguised from their original senders. Masking the network addresses of the original senders in such a way thwarts (to certain degree) any imposter from analysing the traffic. Secure OS deals with memory management to prevent one application from eavesdropping on the data of another application. These four measures serve as the basic requirements to craft further security framework as follows.

Execution Environment (EE) is one of the main characteristics that were absence in the conventional static proxy systems, but it is commonly found in the today's dynamic proxy system, as the basic requirement to gain scalability and flexibility. EE defines a restrictive environment, normally by means of secure module loader and access control policy, to enable namespace protection and to limit resource accessibility. However, the fact that the execution environment must be trusted, removes the option of placing proxies at arbitrary locations in the network (particularly for the network-level proxy systems), and limits the flexibility and possibly the effectiveness of the proxies (Portmann and Seneviratne, 2000).

Safe language (Hicks *et al.*, 1998) is another approach to safeguard both the proxy object and proxy host from unforeseen attacks, by constraining their programmability, i.e. limiting their functionality.

The major challenge to dynamic proxy solution is the breach of contents originality, given that contents maybe manipulated in transit, by either the trusted proxies (for filtering and transcoding purposes), or the intermediate imposters (on modifying and removing the contents). Illegal tampering renders the contents non-trustable, and thus impeding the large-scale deployment of dynamic proxy. The common approach to ensure data integrity is with the use of cryptography, to encrypt each packet with a timestamp imposed on each packet that is to be transmitted from the central server to the clients.

Incorporating such feature into the real-time multimedia data transmission has been studied in (Gennaro and Rohatgi, 1997; Golle and Modadug; 2001; Baugher *et al., 2001*) Technique of converting encrypted contents with one key, to encrypted contents with another key, without access to the data in its plaintext has also been studied (Blaze *et al.*, 1998). However, the limited specific functionality of such approach propells further alternatives to be proposed. These alternatives include co-locating the proxies with either the server or the clients, to perform distillation before encryption is applied (Portmann and Seneviratne, 2000); believing that the intermediate proxies will distill, re-encrypt and forward any packet based on the principle of trust management (Josang, 1999); and adopting the layered coding schemes (Musmann, 1995; Fankhauser *et al.*, 1998).

If cryptography is less adequate, it is suggested that contents could be traced by means of hidden copyright marks and other secret information embedded in the contents (Bender *et al.*, 1996; Wolfgang and Delp, 1997). Such marks were designed to guard against tampering, illegal copying and distribution of contents.

Standard Layer of Security Protocol

The section analyses the three most common types of security protocols in the context of proxy security. These protocols include the Internet Protocol Security (IPSec), Transport Layer Security (TLS) and the application-level security. A detailed and unified model of the communications security, based on the standard seven OSI layers plus an additional human-computer layer in the context of AN services are well documented in (Brown, 2001).

IPSec (Thayer *et al.*, 1998) introduces two mechanisms to protect data packet. The first mechanism uses the Authentication Header (AH) to assure both authenticity and integrity. The second mechanism employs the Encapsulating Security Payload (ESP) to guarantee confidentiality. The main advantage of IPSec is its transparency to applications. However, neither AH nor ESP are compatible with the use of proxy. This is simply because IPSec operates underneath the layers where most common proxies work at, in the network protocol stack. Installing security features at the IPSec prohibits the proxies from interpreting any packet in transit, and thus renders intermediate services (such as distillation and transcoding) infeasible.

TLS (Dierks and Allen, 1999), the standardised successor to Secure Socket Layer (SSL), share similar goals to the IPSec. Operating at the transport-level allows the TLS to serve as an implicit security protocol to deliver secure connections in a typical client and server connection. It supports proxy at both the transport and application layers.

Working at the highest layer in the protocol stack, the application security layer gains the merits of serving specific application requirements over the IPSec and TLS (Oppliger, 1997), and stays compatible with all types of proxies. It is easier to be deployed but much harder to write, as detailed analysis and testing are compulsory to guarantee its security strength.

Table 2 summaries the compatibility of these security protocols to both the application and transport proxies, with "ticks" denoting compatible.

Table 2. Compatibility of security protocols.

	Application proxy	**Transport proxy**
Application Layer	✓	✓
Transport Layer		✓
Network Layer		

From Table 2, it is shown that security protocols that operate at layers that are lower than the proxy, renders the protocol useless, i.e. incompatible with the proxy. Much greater restriction will be posed on the type of security features, as the proxy goes higher in the protocol stack. In summary, security measures should be enforced at the same or higher layer of where the proxy is in service. In the case of CPF, we derive solutions on the application layer, to gain the advantage of being customisable to specific application needs, while leveraging on existing protocols at the lower layers.

CPF: Overview and the Threat Model

CPF is an application level approach that deals with the growing Internet, by deploying intermediate object called the DAPS into the network. DAPSs are placed dynamically at runtime on nominated voluntary client hosts by the central server, to execute tasks on behalf of the central server. DAPS scales the central server and mitigate the bandwidth consumption by reducing the number of direct client-to-server connections over the WAN. The overview of the CPF distributed model is depicted in Figure 1. It shows the client machine (running the EE software called CPFnode) hosting the DAPS as the local server for all local subnetwork clients, including the DAPS-hosting client itself. The human object represents the agent called ObjectBasket (OB) that is responsible for delivering and setting up both the client application (e.g. Video Viewer) and/or the DAPS (e.g. Video Proxy).

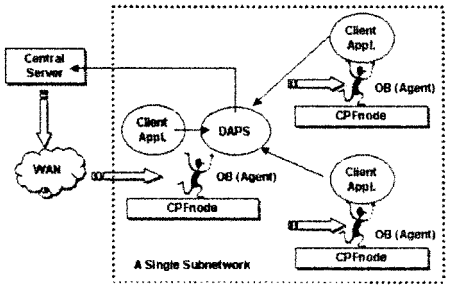

Figure 1. The distributed model of CPF.

The uniqueness of CPF lies on the use of client machines to host DAPS services. This is done by appointing selected clients (as proxy) that have already downloaded, or are downloading contents, to turn around and serve the contents to other local clients at runtime, thereby relieving the server and network from redundant loads while accelerating the speed of delivery. Contents can be of both static and dynamic nature such as web pages, video streams etc. Tapping into the large availability of client resources in a secure manner offers several advantages, including the reduction of server loads (with distributed DAPS), masking of DAPS failures (given large number of client machines) and possibly exploiting the use of local IP multicasting.

Despite the potential benefits of CPF for being an economical approach to conserve both the server and network resources, to attain better performance (Choong *et al.,* 2002), large-scale deployment of CPF would not be possible in the absence of detailed analysis on its security flaws.

Figure 2 depicts the possible security breaches (with numbers denoting the setup sequence) of CPF at the component interaction level, in setting up both the proxy (which is on the first client) and the second client in a typical video broadcasting session. In brief, the setup starts as follows. The first client connects to the central server and is assigned as the local proxy. The agent (OB) is responsible for setting up both the DAPS and client application on the first client host. Subsequently, the second client participates in the same session. Given the presence of a local DAPS, the second client is redirected to the local DAPS for downloading a copy of the application object. Then, the client application object is activated. Finally, the server starts sending contents to the client via the intermediate proxy. Detailed of the CPF hand-shaking procedure can be found in (Choong *et al.,* 2002).

As a rule of thumb, every host in the Internet should be regarded non-trusted (line 1), until the proof of identity or certificate of the hosts are given and agreed. This implies that the

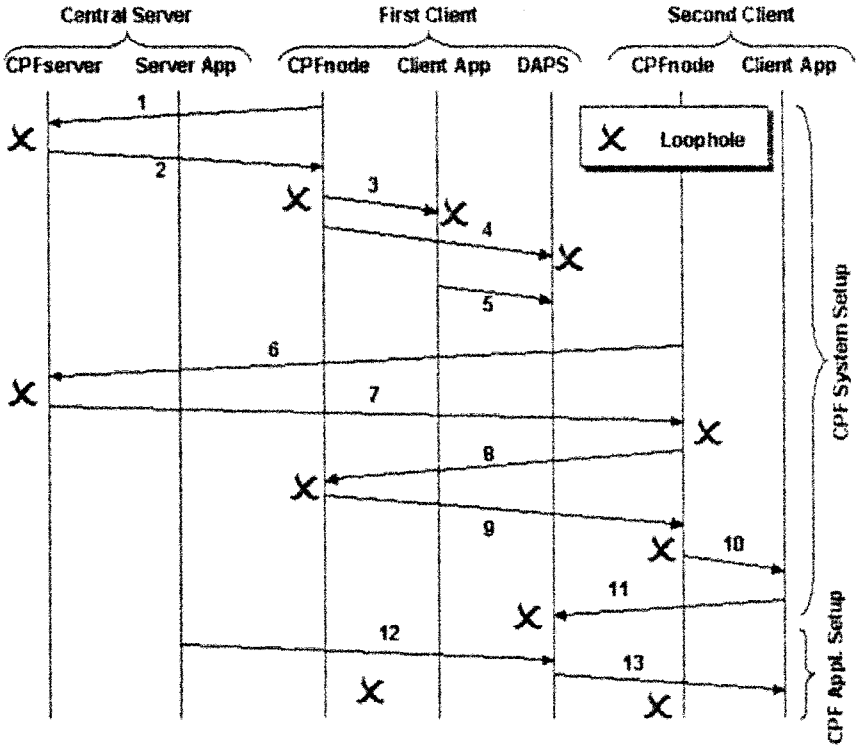

Figure 2. Security loopholes in the hand-shaking procedure of CPF.

clients must be sure of whom they are connecting to.

Upon receiving any object (i.e. agent or proxy), proper validations and restrictions must be imposed (line 2). Activating the object without proper validations could jeopardise the hosting machine, resulting illegal resource utilisation that brings about the Denial-of-Service (DoS) attacks and possibly damaging the host. Line 3-4 shows how the OB could allocate and execute malicious codes that endanger the voluntary client host. In general, the same sets of security hassles (Line 1-4) will be encountered by every CPF client, i.e. during their initial setup procedures.

Given that an OB (on the first client host) has been made available, subsequent clients must challenge again the threat of possibly connecting to a non-trustable OB (Line 8). Line 9 again reflects the risk of receiving and then executing non-malicious codes, similar to Line 2. Once the client application is activated, an application-level network connection shall be established to the respected DAPS, which again could be malicious, e.g. distributing fake contents (Line 11).

Connections denoted by Line 12-13 (application-level connections) are exposed to the threats of information leakage and contents tampering. Information leakage occurs whenever a transmission is not encrypted, which subject to snooping on data stream by any party who can gain access to the network. Illegal access to the information further allows several unethical activities to be taken place: re-distribute the contents to other users without the consent from the sender (violating copyright issues); modify the contents before passing it down to the right receiver (contents tampering).

In general, attacks on CPF can be summarised as follows:

- Non-trusted source (either the central server or the local OB)
- Execution of malicious proxies (illegal resource utilisation)
- Information leakage (data snooping and redistributions)
- Contents tampering (lost of data integrity)

Ways to tackle these attacks have been well studied (Portmann and Seneviratne, 2000; Brown, 2001; Gennaro and Rohatgi, 1997; Golle and Modadugu, 2001; Bender *et al.*, 1996; Wolfgang and Delp, 1997), however proposal of addressing them within a single framework, particularly to deal with the CPF context with dynamic proxy executing at runtime on client host seems unavailable. In the next section, we propose a CPF-related security framework, and ask and answer the following questions:
How to ensure codes executed on the client machine is non-malicious:

- How to confine the accessibility of any mobile code?
- How to avoid packets in transit from being sniffed by unsubscribe receivers or imposters?
- How to enable clients to identify fake contents distributed by malicious proxy?
- How to prevent both legitimate and illegitimate subscribers (i.e. either the proxy or client) from redistributing of the contents to unsubscribe clients?
- Given that a data stream is encrypted, how intermediate proxy services (e.g. distillation) could be introduced without the risk of generating fake contents?

Security Framework of CPF

Delivery of Non-malicious Objects

Problem of malicious objects could be solved by adopting the public-key cryptography to validate the OB, DAPS and client application, which will be delivered and executed dynamically on the CPFnode. The central server signs each of these objects with its private key. The CPFnode then uses the public key of the central server to validate the identity of the OB, subsequently unpacking the bundled DAPS and client application if the OB is originated from a trusted server.

The use of OB is regarded efficient because the CPFnode needs only to verify the digital signature of the OB to infer the validity of the application objects it carries. Based on

the principle of trusts, this approach shortens the application setup duration as the signed OB is only verified once regardless of the number of application objects it holds.

To further ensure that the client application is connecting to the valid DAPS instead of other intermediate forged objects, both DAPS and client application can mutual authenticate each other's identity by adopting a scheme based on the public-key cryptography. This scheme works as follows. Initially, both the DAPS and client applications are hard-coded with an identical hidden message (could be numbers or texts) at compile time. Also, they mutual-exclusively take either of the public or private keys (hereinafter the system key) that were generated at the central server. Upon activation, both the DAPS and client application would request each other to send an encrypted version of the hard-coded message with their own system keys. To validate each other, both of them would decrypt the received message and match with their own hard-coded message for similarity. Only a correct match would cause the operation to proceed with the subsequent CPF setup activities. Imposters would not be able to tamper the hidden message and realise the system key of either the DAPS or client application because they were bundled within the OB, which was digitally signed by the trusted server.

The use of system key seems reasonable at this stage. However, if the DAPS again needs to authenticate with the central server using the same approach or in a scenario where the client application has to authenticate and establish a 2-tier connection directly to the central server (in the absence of DAPS), the feasibility of such approach is suspected. The design of an authentication scheme based on the public-key cryptography that works in both 2-tier and 3-tier arrangements seem to be more complex. Figure 3 shows three different scenarios (labeled a, b and c) of authenticating objects from various tiers in both the 2-tier and 3-tier arrangement.

Figure 3 (a) depicts the concept described earlier in authenticating both the DAPS and client application, wherein both adopt different key obtained from a common asymmetrical key pair. Assigning the server with the opposite key to that of DAPS seems logical. However, if the client were to link directly to the server without any intermediate DAPS service, such scheme reveals a flaw. The identical *Key1* of both the client and server does not work cooperatively. An alternative (b) is to use three keys, one for encryption *(Key3)* and the rest for decryption *(Key1, Key2)*, based on the technique discussed in (Canetti, 1997). However, the complexity of such technique renders it too costly for implementation in CPF in terms of algorithm complexity and performance. A much economical and yet feasible solution (c) would be the use of session key, where every party uses the same key *(Key1)* generated dynamically during initial connection establishment. The details of creating and delivering the session key are provided in the next few subsection.

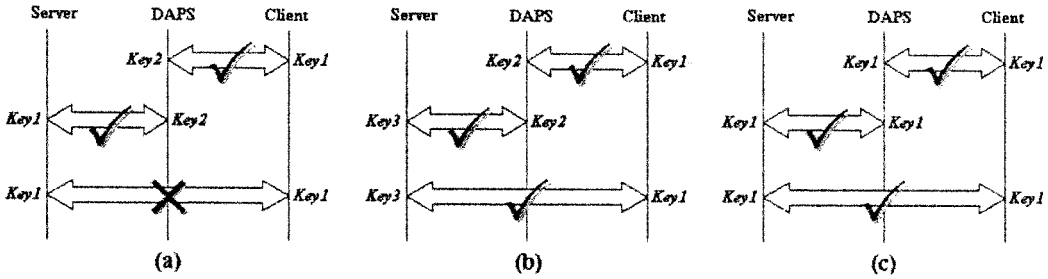

Figure 3. Authentication schemes: (a) Adopting the asymmetrical key pair; (b) Using three separate keys; and (c) Operating with a common session key.

Uncontrolled Resource Utilisation on the Client Machine

As in Java applet, applications launched with CPF by default run in a restricted environment where they have limited access to resources. To increase their accessibility, each application must define its own access privileges via a policy file. This policy file, which is also known as the Access Control List (ACL), consists of a list of access control entries. These entries identify the types of accesses that the principal (DAPS and client object) may or may not make on a particular resource. These accesses are described in terms of permissions. Permission specifies a permitted access to a particular resource, for example read or write access to certain file.

To ease the adaptation of access level into applications, a role-based access control (with defined security domain) has been used. Here, each object is categorised by role, i.e. according to its functionality or behavior, as shown in Figure 4 (different applications are given different sets of access permission). For example, a network-monitoring agent should be given the permission to collect the system information, gather network statistical values and any on-going sessions on the system to a certain extent. A network management agent has higher authority than the monitoring agent whereby it is able to query, interact with, or terminate any object on the machine. Together, both network monitoring and management agent form an object management system, denoted by Object B in Figure 4. A normal standalone application should not be given any network accessibility permission, as it should operate without any external interactions, such as a text editor. These sets of rights are known as the permission (3rd column in Figure 4). Both the DAPS and client application must carry their ACLs to inscribe the resource accessibility level based on the role-based formats.

To guard against DoS attacks, the CPFnode should also monitor the usage of resources such as the CPU and memory utilisation. This is to prevent any program from monopolising the entire computing resources.

Illegal Access to Contents

To guard against illegal access and illegitimate redistribution of contents, cryptography has been again adopted. Incorporating cryptography on real-time media is less feasible given the significant amount of computing resources and lengthy time spent on decryption. However, by adopting the cryptography with shorter key lengths that is based on the symmetric key algorithm such as the Data Encryption Standard (DES), illegal tapping on delivery stream is render more difficult while not sacrificing the quality of data transmissions.

Table 3 (extracted from (Coulouris *et al.*, 2001)) shows the time spent on performing cryptography based on different key size of several public- and secret-key cryptographic algorithms. Where available, the readings in the last two columns are based on algorithm written using different languages, i.e. C and assembly language, respectively. The reported figures could be taken as rough estimates of the performance of the algorithms on a 333 MHz Pentium II processor.

Given the CPU speed of today's computer systems, it is believed that software-based cryptography would serve a viable alternative to hardware-based solutions (Brown, 2001). Researchers have studied the feasibility of software-based cryptography in authenticating streamed data. In (Gennaro and Rohatgi, 1997), a one-time signature was used together with a

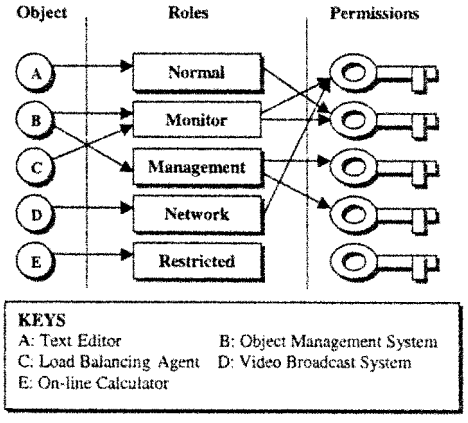

Figure 4. Role-based access control scheme.

Table 3. Performance of encryption for different key size.

	Key size (bits)	Extrapolated speed (kbytes/s)	PRB optimized (kbytes/s)
TEA	128	700	-
DES	56	350	7,746
Triple-DES	112	120	2,842
IDEA	128	700	4,469
RSA	512	7	-
RSA	2,048	1	-

single digital signature to authenticate a sequence of packets. This scheme however, is not resistant to packet loss. A better scheme that is both efficient in terms of authentication delay, and being tolerable to packet lost, has been subsequently proposed in (Golle and Modadugu, 2001). Unfortunately, such scheme does not cope with the problems of illegal access and redistribution of contents.

The central of the CPF security framework is the use of secret-key cryptography (session key) to safeguard CPF from various possible security threats. Before the session key approach can be applied, attention must be paid on delivering the key securely (without the knowledge of any eavesdropper) between the sender (central server) and the receiver (either the DAPS or client application). Again, the public-key cryptography comes in handy for mutually authenticating each other for agreeing on a session key in a standard manner as shown in Figure 5.

The client starts by encrypting his/her identity (e.g. subscriber ID) and a random number, RC, using the server public key, ES. Upon receiving this message, the server replies with a message (encrypted with the client public key, EC) containing the client RC, his own random number, RS, and a proposed session key, KSS. The client decrypts the message using his own private key and finds that the RC is intact, fresh and not a replay, since it has just been requested. The client agrees on the proposed KSS

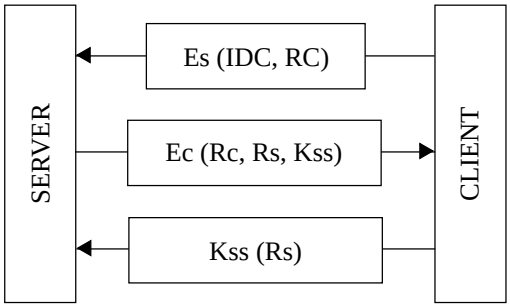

Figure 5. Hand-shaking procedure for agreeing on a session key.

by sending back an encrypted message. Looking at the RS that it generated, the server is convinced that the client is the receiver of message 2. Hence, both authenticated each other to establish a secure connection.

By agreeing on the session key, the client can decrypt the encrypted contents delivered by the central server. The plain content is then processed accordingly. If caching is permitted (given that the client is also the proxy), the contents should again be encrypted with the system key before saving on the local file system. Doing this prevents users from identifying and hence copying the cached contents for subsequent off-line broadcasting. However, assuming some imposters (who could be a valid subscriber using the CPF system) are able to sniff and capture all the plain packets during the decryption process, there is no way to prevent

illegal broadcasting from taking place. In the worst case, the imposter may alter the original contents and subsequently fooling the receivers with fake contents. These critical hurdles put the intellectual property of the content owner at risks. The next section describes a mechanism to deal with such impediments.

Lost of Data Integrity and Illegal Redistribution of Contents

The technique to mitigate the likelihood of both lost of data integrity and redistribution of contents is actually quite simple. It is a combination of logging down the subscriber's information (such as the subscriber ID, IP address and port number together with a timestamp), and attaching the subscriber's fingerprint (constructed according to the subscriber's information) on every packet transmitted from the central server. With the fingerprint affixed on each packet, it is believed that no imposter (even being a valid subscriber) is willing to take the risk of disclosing his/her identity for illegal content forwarding. Tampering the packet is also impossible as the fingerprint could be accidentally removed or overwritten, resulting in the packet to become unrecognisable by valid subscribers.

To avoid imposter from identifying and hence overwriting or removing the fingerprint from the packet, the fingerprint must be randomly placed in the packet without the knowledge of imposters. To trade off better performance over security, the packet could be partially encrypted, i.e. only randomly selected range of bits in the packet are encrypted. To agree on a common position for sliding in fingerprint and performing encryption, two or more random numbers (denoting a series of both the starting position and range of bits) can be encrypted with the session key and sent to the receiver during the earlier hand-shaking session. To further avoid imposter from identifying the position of where the encryption starts and ends using the brute-force approach, different encrypted random number(s) could be used and piggybacked on each data packet. With such approach, we believe that tampering the contents without knowing the exact starting position and range of encryption is made difficult and infeasible, but at the cost of burdening the proxy, i.e. to decrypt, distill, encrypt and forward the contents to the rest of the clients. Handling these additional workloads either demands proxy host with higher processing capability to be available, or renders the number of sessions or clients that could be supported/served to reduce.

As a result, we adopted the concept of layered coding (Musmann, 1995; Fankhauser *et al.,* 1998) as the solution to maintain the serving capability of the DAPS without imposing higher requirements on the DAPS hosting machine. The main idea of layered coding is to divide a data stream into several substreams, called layers. It has a cumulative property and thus allowing receivers who receive more packets to achieve better quality. Such property renders the issues of receivers' heterogeneity to be tackled at ease. Moreover, it also lightens the proxy workloads from content decryption, distillation, encryption and forwarding, to only forwarding, i.e. without the need of performing distillation. The distillation process has been camouflaged into the layers organisation and forwarding process at the DAPS. The DAPS just need to forward the number of layers according to the capability of the clients. Such approach allows the fingerprint scheme we discussed above to be supported seamlessly and transparently.

Fingerprint attachment is just one of many solutions to address the issues of data integrity. Others include the deniable encryption (Canetti, 1997), which adopts one encryption key, and two or more decryption keys (owned by individual receivers) on the same piece of message to ensure data integrity. However, such scheme is computation intensive and thus it is not suitable for transmission of dynamic contents. We have also given a thought on the idea of gauging the normal duration of relaying packets by the valid proxy against a malicious proxy, and attempted to derive an acceptable range of ratios as intervals of non-intercepted delivery. However, the heterogeneity and fluctuating workloads of the proxy machine, coupled with the unstable backbone throughput and buffering time intervals, render this approach infeasible. The

proposed fingerprint and partial encryption approach is regarded much more viable than the others.

The Secure Hand-shaking Setup Protocol

Figure 6 gives a pictorial description of how the security policies previously discussed are integrated and used as a single framework on fortifying the CPF hand-shaking protocol described earlier. The figure exemplifies on how the first client is authenticated, and this applies to all subsequent clients (excluding the setting up of DAPS).

The first three interactions in Figure 6 are for negotiating on a common session key, similar to Figure 5. The subsequent interaction is to deliver the OB to the CPFnode. This is followed by the verification on the identity of OB as described earlier. Positive validation result allows the CPFnode to proceed further, i.e. to activate both the DAPS and client application in sequence, with each fed with the session key, KSS. The authentication between the DAPS and client application is then started using the session key. Upon passing the authentication, the DAPS is then required to send an encrypted message containing the hidden message, particulars such as the subscriber ID, its IP address, port number, the current time, and a random number, RD, to the server using the session key. The server then logs down these information for accounting and backtracking purposes. Then it starts decrypting and matching the hidden message against its own message. A correct match further allows the fingerprint of the receiver (in this case is the DAPS) to be generated according to the given information.

An encrypted message that contains the same RD specified earlier by the DAPS is replied to the DAPS (without the fingerprint attached). The server shall then wait for a connection from the client application, which should be initiated by the DAPS. The client application must then undertake the same procedure as carried out by the DAPS to authenticate the server and itself. Here, the server should reply the client with an encrypted message that encloses also the earlier registered DAPS particulars such as its functioning URL. Discovering a known random number, RC in the encrypted message permits the client application to connect to the DAPS at

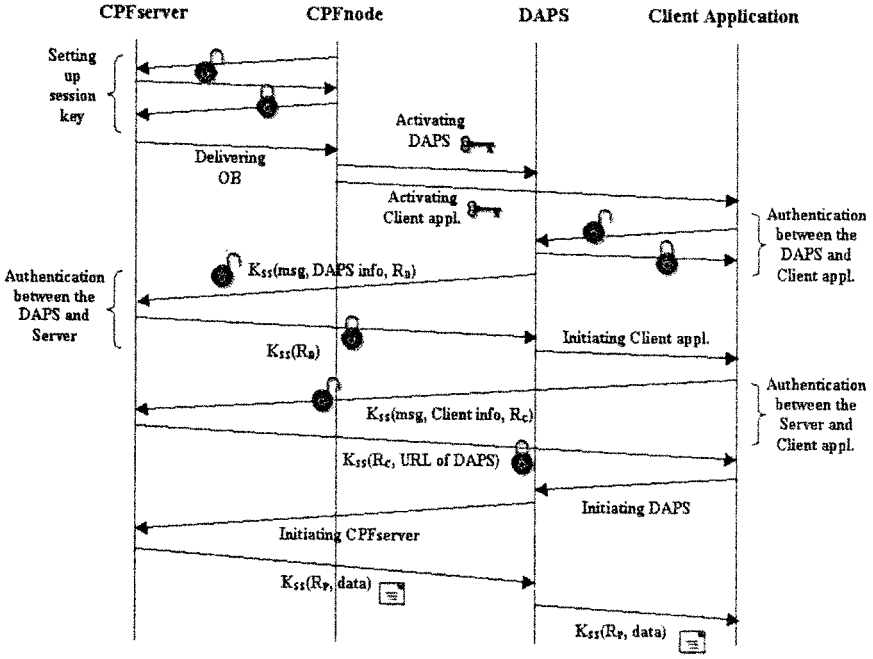

Figure 6. The secure CPF hand-shaking protocol.

the given (trusted) URL. The DAPS shall then start receiving data packets that have been fingerprinted, partially encrypted and time-stamped by the server. To identify the fingerprint and to decrypt the packets, both the DAPS and client application have to know the random position, RP, suggested by the server.

Capturing all packets without the knowledge of the RP leaves the imposter with no chance to replay (with the presence of timestamp), tamper (that results disrupted contents), or resell (given the fingerprint of the DAPS is secretly embedded) the contents. Guessing the RP is rendered difficult as the RP can be changed across subsequent transmission depending on the security level configured as the central server.

Conclusion

In this paper, we have discussed the security issues of proxies, particularly the dynamic proxy in the context of CPF. Given that the CPF promotes the execution of dynamic proxy on the client host at runtime, our main contribution in this paper lies in tackling some common and unconventional threats in a dynamic proxy environment, within a single security framework. We address the security issues of non-trusted source, execution of malicious proxies, information leakage, contents tampering and illegal redistribution of contents. We propose the use of fingerprint together with the layered coding scheme to deliver contents from the server to the clients via intermediate DAPS, without breaking the end-to-end communication integrity. Towards the end, we present a secure CPF hand-shaking protocol, and describe how secure CPF-enabled connections could be established in sequence.

We perceive that the layered coding scheme limits the clients with high capability from receiving data at higher quality, in cases where the DAPS is hosted on machine with lower capability. This provokes the studies of electing DAPS-hosting machine to ensure DAPS is always hosted on the best available client host. Other future works include implementing the security framework to verify its effectiveness, performance and cost.

References

Baugher, M., Blom, R., Carrara, E., McGrew, D., Naslund, M., Norrman, K. and Oran, D. (2001). The secure real time transport protocol. IETF draft.

Bender, W., Gruhl, D., Morimoto, N. and Lu, A. (1996). Techniques for data hiding. IBM Systems Journal. 35(3 & 4):313-336.

Blaze, M., Bleumer, G. and Strauss, M. (1998). Protocol divertibility and atomic proxy cryptography. Proc. Eurocrypt, and Lecture Notes in Computer Science 1403. p. 127-144.

Brown, I. (2001). End-to-end security in active networks, PhD. thesis. Computer Science. University of London.

Canetti, R. (1997). Deniable encryption. Proc. of Advances in Cryptology (CRYPTO), and Lecture Notes in Computer Science 1294. Springer-Verlag. p. 90-104.

Choong, K.N., Mohd, A, B., Prakash, V., Tee, E.R., Yee, Y.C. (2002). The framework of a dynamic proxy system. Suranaree Journal of Science and Technology. 10(1): 7-18.

Coulouris, G., Dollimore, J. and Kindberg, T. (2001). Distributed systems: Concepts and design. 3rd Edition. Addison-Wesley.

Dierks, T. and Allen, C. (1999). The TLS Protocol Version 1.0. RFC 2246.

Fahmi, H., Latif, M., Sedigh-Ali, S., Ghafoor, A., Liu, P. and Hsu, L. (2001). Proxy servers for scalable interactive video support. IEEE Computer. 34(9):54-60.

Fankhauser, G., Dasen, M., Weiler, N., Plattner, B. and Stiller, B. (1998). The WaveVideoSystem and network architecture: Design and implementation. TIK technical report No. 44. ETH, Zurich.

Gennaro, R. and Rohatgi, P. (1997). How to sign digital streams. Proc. of Advances in Cryptology (CRYPTO), and Lecture Notes in Computer Science 1294. Springer-Verlag. p. 180-197.

Ghost, A., Fry, M. and Maclarty, G. (2001). An infrastructure for application level active networking. Elsevier Computer

Publication. Networks. 36(1):5-20.

Golle, P. and Modadugu, N. (2001). Authenticating streamed data in the presence of random packet loss, Extended Abstract. Available from: http:// www.isoc.org/isoc/ conferences/ ndss/01/ 2001/papers/golle.pdf

Hicks, M.W., Kakkar, P., Moore, J.T., Gunter, C.A. and Nettles, S. (1998). PLAN: A packet language for active networks. Proc. of the 3rd ACM SIGPLAN International Conference on Functional Programming (ICFP). Baltimore, Maryland. USA. p. 86-93.

Josang, A. (1999). Trust-based decision making for electronic transactions. Proc. of the 4th Nordic Workshop on Secure IT Systems (NORDSEC). Stockholm, Sweden.

Musmann, H.G. (1995). A layered coding scheme for very low bit rate video coding, Signal Processing: Image Communication. 7:267-278.

Oppliger, R. (1997). Internet security: Firewalls and beyond. Communications of the ACM. 40(5):92-102.

Portmann, M. and Seneviratne, A. (2000). The problem of end-to-end security for proxy-based systems. Proc. of Protocols for Multimedia Systems (PROMS), October 22-25. Available at: http://mobqos. ee.unsw.edu.au/mobqos/Papers/ID15_ proms2000.pdf. Access date: Jan 2002.

Smith, J.M., Calvert, K., Murphy, S., Orman, H.K. and Peterson, L.L. (1999). Activating- networks: A progress report. IEEE5 Computer. 32(4):32-41.

Thayer, R., Doraswamy, N. and Glenn, R. (1998). IP Security Document Roadmap. RFC 2411.

Wolfgang, R.B. and Delp, E.J. (1997). A watermarking technique for digital imagery: Further studies. Proc. of the IEEE International Conference on Imaging, Systems and Technology. Las Vegas, NV, USA. p. 279-287.

Yatin, D.C. (2000). Scattercast: An architecture for Internet broadcast distribution as an infrastructure service. Ph.D. thesis, Computer Science Division. University of California. Berkeley.