

Carleton University  
School of Computer Science  
Honours Project (Comp 4905)

Protecting VoIP Servers from DoS Attacks by Implementing The Cryptographic Puzzle  
Mechanism

by: Fadil Sutomo

Supervisor: Prof. Michel Barbeau

School of Computer Science

April 9<sup>th</sup>, 2008

## **Abstract**

The emergence of VoIP technologies brings people to think about moving away from the traditional PSTN network and switching to VoIP. Consequently, security and reliability issues of VoIP networks arise for those who use VoIP and since then VoIP security and reliability become interesting topics discussed by information security practitioners around the globe. This paper presents a mechanism in mitigating the denial-of-service attacks by using the cryptographic puzzle mechanism. In order to depict the real situation of VoIP networks, this paper will use VoIP technologies such as VoIP PBX, SIP protocols and IP phones to demonstrate the result of the cryptographic puzzle implementation. In the end of the paper, discrepancies and possible improvements for future works will also be discussed.

## **Acknowledgments**

I would like to thank Prof. Michel Barbeau for his support and valuable advice throughout the making of this project. I would also thank Felipe ('Umar) Silva who has given me some suggestions to improve the “English” of this paper. Finally, I would thank those people at *asterisk-dev* and *asterisk-security* mailing lists for their ideas and great efforts in improving Asterisk and helping Asterisk developers.

## **Table of Contents**

List of Figures and Tables.....	5
Chapter 1: Introduction.....	6
1.1 What is VoIP? .....	6
1.2 What makes a system secure?.....	7
Chapter 2: VoIP Security Issues.....	11
2.1 Where does it all come from?.....	11
2.2 VoIP defense to-date.....	12
Chapter 3: Session-Initiation Protocol (SIP).....	14
3.1 SIP overview.....	14
3.2 SIP architecture.....	15
3.3 SIP security issues.....	17
Chapter 4: Mitigating DoS Attacks with the cryptographic puzzle mechanism.....	18
4.1 What is a cryptographic puzzle?.....	18
4.2 Solving the puzzle.....	19
4.3 DoS attacks and how the puzzle handles them.....	21
Chapter 5: Puzzle Testing and Implementation.....	22
5.1 Environment and attack scenarios.....	22
5.2 Testing.....	23
Chapter 6: Conclusion.....	30
6.1 Discrepancies and future works.....	30
6.2 Conclusion.....	32
References.....	33
Appendix A: Adding fadil puzzle set workNumber <integer> into Asterisk Command Line Interface (CLI)	35
.....	35
Appendix B: Changes to asterisk.c.....	36

## List of Figures and Tables

Figure 1: The puzzle mechanism.....	17
Figure 2: Testing environment.....	24
Figure 3: Wireshark showing “500:Server Internal Error”.....	26
Figure 4: How our puzzle mechanism works.....	27
Figure 5: The ideal implementation.....	31
Table 1: Testing without the puzzle mechanism.....	25
Table 2: Testing with the puzzle mechanism.....	28

## **Chapter 1: Introduction**

### **1.1 What is VoIP?**

Voice over Internet Protocol (VoIP) is a rising technology that can revolutionize the way the world communicates. VoIP is also commonly known as *converged* networks since it 'converges' the voice and data networks into one network [Wallingford, 2005]. Advances in today's technologies make VoIP networks available to handle both voice and data transmission as well as multimedia applications. Using VoIP, people can talk to each other, chat, hold a teleconference, or even fax a document to a printer server with only one network.

There are many benefits that people can get from VoIP networks. First, converging data and voice under the same networks provides cost-saving solution for enterprises. Instead of focusing on several networks, enterprises can now focus on one and by running the data and voice packets in one stream, it makes the enterprise network more manageable. Although the implementation may be costly, enterprises can gain benefits from the lower operational cost in the long run [Wallingford, 2005].

The second benefit from VoIP network is that any mechanism that can be applied to IP networks is applicable to VoIP networks. In the legacy Public Switched Telephone Networks (PSTN), for instance, voice encryption is very expensive and restricted to certain government institutions. In VoIP networks, however, encrypting voice is as simple as encrypting the transmitted packet over IP. The general idea is that VoIP applications are easier to program.

There are more advantages than the two mentioned above. Using VoIP, people can call their families, relatives or business partners abroad at a very cheap price since roaming does not exist in the VoIP world. Unlike the mainstream telco providers, most VoIP providers give features such as voicemail, call-forwarding and 3-way call for free. With all these benefits, it is no wonder why people are moving to VoIP.

However, these benefits do not come for free. There are many challenges and concerns that need to be taken into account. How reliable is VoIP? How secure is it? Can someone eavesdrop on a conversation? Can someone steal the data that I send over the network? What is going to happen if the VoIP server crashes? Before answering these questions, we will first review what causes a system to be

considered as a secure and reliable system.

## **1.2 What makes a system secure?**

Prior to discussing VoIP security, it is a good idea to know what makes a system or an application secure. A system is secure when it can protect/deliver the data/services requested, stored or passed within it. The CIA triad (confidentiality, integrity and availability) are the three main widely accepted variables in information security. There is also another concept called AAA (access control, authentication and auditing)<sup>1</sup> which are the mechanisms that supports CIA [Dubrawsky, 2007]. If an application can ensure and implement these concepts appropriately, then it is categorized as secure application [Porter, 2006]. Let's go through these concepts briefly.

### **- Confidentiality**

Nowadays, there is a lot of sensitive data being sent over the Internet. Our names, dates-of-birth, addresses, and credit-card numbers are common types of data being sent over the Internet and considered as sensitive data. The challenge of protecting this data is increasing as more and more attackers try to steal it by exploiting through vulnerabilities that exist in applications or networks.

Security experts define confidentiality as the concept of protecting sensitive data from being viewable or modifiable by an unauthorized entity [Lehtinen, 2006]. Confidentiality can also be regarded as a concept that assures that any unauthorized third party should not know anything about what is stored or transmitted in the media [Adams, 2002]. If data can be sniffed, viewed, or modified by unauthorized parties, then the system in which the data is stored in or transmitted over does not have confidentiality and thus categorized as not secure.

---

<sup>1</sup> There are some people (such as those from Cisco) who refer AAA as authentication, auditing, and accounting. However, they still mean the same thing, just different terminology [Dubrawsky, 2007].

## **- Integrity**

Computer networks create a virtual connection between one person and another. Recall that the Internet is a computer network on a very large scale. Using the Internet, people no longer have to be physically present in order to talk to each other because of services like instant messaging or VoIP calling and they do not have to physically go to their banks to pay their bills because of online banking. However, the Internet is a public network. People have to be sure that the data that they are sending and receiving over the Internet will not be tampered by others. Imagine if we go to our bank website to pay \$25 for our monthly telephone bill, but in transit, there is another zero added behind the number 25. There will be many disorders and complaints as a consequence of such an error.

Integrity is the concept of keeping the data intact from any unauthorized entities while it is in transit or inside the storage [Lehtinen, 2006]. The presence of data-integrity in a system proves to be very important for a system to be deemed as secure. If there is no assurance that the data cannot be kept intact from unauthorized entity, then the system does not have integrity and thus categorized as not secure.

## **- Availability**

Availability is the concept of a system being usable when needed. In other words, the data or services requested by authorized clients have to be available. Sometimes, a service may not be available when users need it. For example, placing or receiving a call in VoIP is one service offered by the VoIP server to legitimate clients. When the VoIP server crashes, consequently the service is no longer available and placing or receiving a call becomes an impossible task. Another well-known example is when an application freezes. It then becomes unresponsive and no matter what the users do, they will not get the services or information requested.

We have briefly discussed what the CIA triad are and we now want to look at the mechanisms that

supports them which are AAA: access control, authentication and auditing. As we said earlier, AAA are the mechanisms used in protecting data as well as the equipment in which it is stored or over which it is transmitted [Dubrawsky, 2007]. The three aspects of AAA can be regarded as parts that work together in providing confidentiality, integrity and availability in a system. We will briefly discuss each aspect of AAA.

### **- Access Control and Authentication<sup>2</sup>**

As the name implies, we can think of access control as the process of controlling how access to some data in a system is governed while authentication is the process of ensuring that users are really who they claim to be. Access control governs which users should be able to access some data or services while authentication governs the mechanisms present before they are able to view or modify the data. For example, when a user wants to open an email account. He or she should first have access to the Internet. Second, he or she should have the right combination of username/password for that particular account.

### **- Auditing**

Auditing is the process of monitoring the interaction between the users and the system. Auditing may also include monitoring the activities of the system and the outside world. Auditing proves to be useful in ensuring that security policies are practiced appropriately by the users. It can be regarded as the mechanism of tracking who is doing what at what time. Therefore, it is useful for legal activities since it can collect the list of activities done by the users [Dubrawsky, 2007]. One well-known example of auditing is the log-file analysis. Log-file analysis is an activity where one analyzes the log files in a

---

2 There is a fine difference between access control and authentication. The problem comes from the notion that having access to a certain data does not mean having authorization to view or change the data. Simply put, access control is the key of your apartment and the authentication is the activity of putting the key into the lock and trying to open it [Dubrawsky, 2007]. Because of this difference, we will discuss access control and authentication together.

system. Tripwire ([www.tripwire.com](http://www.tripwire.com)) is an excellent auditing tool that uses changes in log files to report illegal activities. We should now have a general principal about what makes a system secure.

### **1.3 The goal of this paper**

This paper will be dealing with the availability issue in a VoIP system. This paper will also show that having a mechanism to keep a system available is critical. The goal of this paper is to demonstrate the effectiveness of the cryptographic puzzle (or puzzle) mechanism in protecting the availability aspect of a system. In particular, this paper will compare a VoIP server which is equipped with the puzzle mechanism and the one which is not when a denial-of-service attack is being launched against it. This paper will be using Asterisk ([www.asterisk.org](http://www.asterisk.org)) as the main VoIP server and Session-Initiation Protocol (SIP) as the main protocol.

## **Chapter 2: VoIP Security Issues**

### **2.1 Where does it all come from?**

VoIP security problems stem from the fact that VoIP networks are based on IP networks. Simply put, IP networks are intended for data, but not voice. VoIP networks are basically trying to treat voice as data which means that security practices for data networks should be implemented as well [Porter, 2006].

This is quite different than the traditional PSTN networks which are intended for voice and implement

different security practices.

Everything about VoIP sounds perfect until people realize that it runs over the TCP/IP network. This means that all vulnerabilities in IP networks are inherited to VoIP networks [Wallingford, 2005]. All attacks that are applicable to IP networks are applicable to VoIP networks as well. Consequently, eavesdropping, spamming, sniffing, or denial-of-service attacks are possible in VoIP networks but not in traditional PSTN networks. We will give some examples how VoIP networks can be compromised which show us that, without good security practices, confidentiality, integrity, and availability issues are big problems for VoIP.

The first issue that we will touch on is the confidentiality issue. Voice transmission in VoIP networks is not that secure since any unauthorized parties can eavesdrop or sniff the conversation relatively easily; while in PSTN, in order to sniff, permission from the government is required as well as special hardware to tap into the conversation. Tapping a conversation in VoIP networks is the same as sniffing regular IP packets in the network. It is known that there are many applications that can eavesdrop VoIP calls such as *voipong* (<http://www.enderunix.org/voipong/>). By running *voipong* when a VoIP call is established without encryption, then the “packetized” conversation can be sniffed. *voipong* also makes a .wmv file as a final result for anyone to hear.

Since it is easy to sniff a packet, by the same token, it is just as easy to inject a packet. For example, if Alice says, “Hello, how are you?” to Bob, then under VoIP networks, it will be 'packetized' before it is transmitted. En route, a third party can sniff the traffic, and inject (or delete) an extra packet. By the time Alice's message arrives, it may not be the same as when it was sent.

One of the largest issues of VoIP network is how to keep the system available to the users. Many attacks can be launched against a VoIP server which can crash it completely. One prominent attack is denial-of-service (DoS) attacks. Due to the nature of IP-based networks, VoIP networks are very vulnerable to this attack. In fact, DoS attacks are classified as one of the most dangerous attacks [<http://www.silicon.com/>]. By simply sending a worm that can generate network packets, a VoIP

network can be disrupted to the extent that voice quality becomes so low such that the communication is unintelligible. Even worse, this type of attack can eventually crash the server, causing the function of placing or receiving a call becomes unavailable.

## **2.2 VoIP Defense To-Date**

As with every new and immature technology, there are many gaps that VoIP has to fill. From the previous section, we know that VoIP can be attacked from all angles in the CIA triad. Attackers can eavesdrop a conversation easily. Denial-of-service attacks are still the worst nightmare of every VoIP system administrator [Mirkovic, 2004]. With VoIP technologies spreading rapidly, many security practitioners are conducting research to improve the stability and reliability of VoIP services. In this section we will discuss what has been done to VoIP in terms of its security.

### **- Network segmentation using Virtual LAN**

This method is applied to achieve logical separation between data and voice network so that a breach in one network does not affect the other. Using network segmentation, broadcast collision is also prevented from affecting VoIP channels. There are many papers that talk about this such as [Butcher, 2007] and [Kuhn, 2005].

### **- Encrypting Media Streams**

Encrypting the channel where the voice or media is transmitted over is critical in protecting the system from eavesdroppers. In fact, there are many new protocols developed to ensure that the data transmitted over is encrypted such as ZRTP (developed by PGP creator, Phil Zimmerman) or SRTP. [Clayton, 2007] and [Kopsidas, 2007] are two very interesting papers that discuss about securing media stream.

### **- Other works**

There are also other works that have been done in securing VoIP such as using VoIP firewall, using IPSec to authenticate signaling mechanism, authenticate client configuration, and so on. Please refer to [Butcher, 2007], [Porter, 2006], and [Thermos, 2007] for more details.

## **Chapter 3: Session Initiation Protocol (SIP)**

### **3.1 SIP Overview**

The nature of programs such as chat, voice messaging and videoconferencing creates the need for a protocol that is able to set up a session between users, maintain the connection and eventually tear it down. For these purposes, SIP comes as the *de facto* standard of VoIP signaling protocol [Porter, 2006]. The nature of SIP architectures makes it a good fit for setting up a chat, phone or even videoconference session for users to communicate with each other.

SIP is a light-weight protocol—it is text-based. SIP is also a request-response protocol. What makes SIP a *complete* multimedia protocol is its ability to *reuse* other protocols in order to facilitate the communication between users. Once a session has been established SIP can then decide whether the

session is audio or video, choose its appropriate codec, determine the address of the destination, handle the packets exchanged, and support certain functions such as using TLS for secure communication.

For example, in Asterisk, SIP uses the Real-time Transfer Protocol (RTP) to transfer packets to the endpoints. Thus, before any two users can talk to each other, they have to signal the server that they are ready to set up a communication session. This is done by SIP. Once the session is ready, then SIP uses RTP to transfer voice packets between the two users. For more details regarding SIP protocol such as its functionalities and network elements, please read RFC 3261 ([www.ietf.org/rfc/rfc3261.txt](http://www.ietf.org/rfc/rfc3261.txt)).

### **3.2 SIP Architecture**

SIP architecture can be divided into two main components: User Agents and SIP servers.

#### **– User Agents**

User agents are the end-points in the communications. They are basically the logical functions that initiate and/or respond activities in SIP transactions. User agents also have two components: client and server. The user agent client (UAC) is the one responsible for initiating SIP requests and accepting SIP responds. On the other hand, the user agent server (UAS) is responsible for accepting SIP requests and sending back SIP responds.

#### **– SIP Servers**

SIP servers are the computers in the middle of SIP communications. The servers are responsible for facilitating the establishment of the communications such as domain-name resolution. The servers can also determine the location of the endpoints in the network. For example, if a user moves to a different

computer, then another user will have problems in initiating a communication session since the IP is different. The servers solve this problem by providing the IP address to every client registered in the servers. With this mechanism, no matter what computer a user uses, as long as he or she is registered in the server, then the servers will provide the appropriate IP address to the registered username. Thus, setting up a communication session will be easy.

SIP servers can be thought of as being made of three components:

- proxy server : responsible for forwarding SIP request generated by a UAC to another entity whether it be a UAS or another proxy server.
- registrar server: responsible for handling registration of SIP clients
- redirect server: responsible for re-directing a UAC to find an alternative URI, which helps to push routing information for the server.

After briefly discussing SIP main components, it is now a good time to discuss the SIP messages. As we know, SIP is a request-response protocol. There several request messages that a SIP client can generate:

- REGISTER: a request generated by the UAC notifying that it wants to register itself. It is done when the client is on-line.
- INVITE: a request to invite another UAC to establish a SIP communication session.
- ACK: a signal informing that the user accepts an INVITE, and is ready to establish a session and to exchange message with the other client.
- OPTION: a request to the other client, inquiring what capabilities are available before establish a session.
- SUBSCRIBE and NOTIFY: a request to determine the presence of the other user agent. This is the message telling whether the user is offline, online, busy, and so on. SUBSCRIBE is the request, while NOTIFY is the reply.
- CANCEL: a request that will cancel a pending request.

- BYE: a request to terminate a SIP session.

As for the response messages, they indicate the processing of the requests, whether it was successful, failure, or in need of redirection. SIP has several response messages:

- 1XX: indicates that the request is being processed.
- 2XX: success; the request is accepted.
- 3XX: redirection
- 4XX: client error
- 5XX: server error
- 6XX: global failure

Due to this architecture, many people choose SIP as the preferred signaling protocol. Not only does SIP have a simple architecture, but it can collaborate with other protocols.

### **3.3 SIP security issues**

SIP is a lightweight protocol, but it is by no means easy to secure. SIP assumes that it can be used by any media with a no-trust relationship [Thermos, 2007]. There are also other features that make SIP hard to secure such as its ability to reuse other protocols and its peer-to-peer architecture. However, we will only discuss how to secure SIP signaling mechanisms so that it can minimize or event prevent DoS attacks, at least from the signaling point-of-view.

From the previous section, we know that SIP uses challenge/response mechanisms to authenticate the end-points. The end-points send INVITE message to the SIP proxy, and by default the proxy will reply with 407 Proxy Authorization Request message and a nonce to the end-points. By this time, the proxy will have a digest based on the nonce and password. The end-points will then calculate a digest using the nonce and password and send them back to the proxy. If the digest of the end-point matches the digest of the proxy, then the end-point is authenticated and INVITE will be processed. This

mechanism is vulnerable to DoS attacks. Any client can flood the server with thousands of INVITE (or any other messages) exhausting the server's resources. This paper will try to implement the cryptographic puzzle protocol to prevent DoS from this point-of-view. That is, instead of letting the attacker flood the server, the server will send a puzzle that the client has to solve which will mitigate the attack.

## **Chapter 4: Mitigating DoS Attacks with The Cryptographic Puzzle Mechanism**

### **4.1 What are cryptographic puzzles?<sup>3</sup>**

As we know from the previous chapter, we need a mechanism that can deter the attacker from flooding our server with requests. A cryptographic puzzle (or puzzle) mechanism is a protocol to handle DoS attacks. Basically, when a DoS condition is detected, the server will react by sending a cryptographic puzzle to the attacker who sends the request. From this time, any request that the attacker (or a valid client) sends to the server results in a puzzle being sent by the the server. Figure 1 below shows the puzzle mechanism.

---

<sup>3</sup> The following section and 4.3 are heavily based on the Internet draft: <http://tools.ietf.org/html/draft-jennings-sip-hashcash-06>

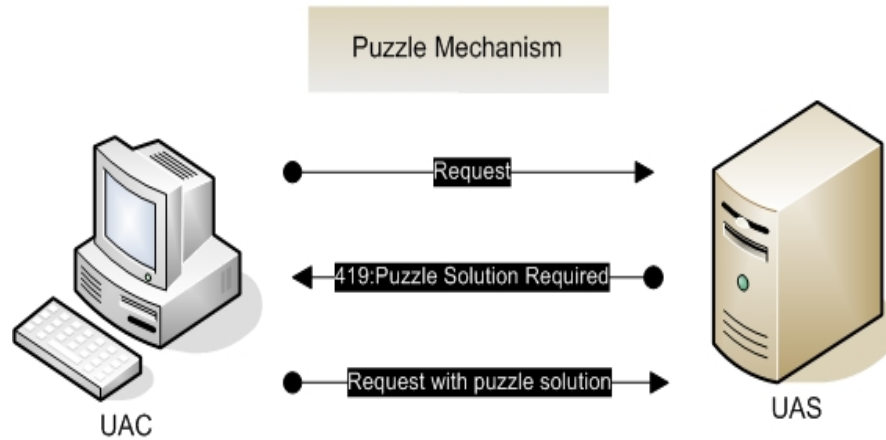


Figure 1: Cryptographic Puzzle Mechanism

A puzzle is a four tuple  $[W, X, Y, Z]$  where  $W$  is a *work* number,  $X$  is a *preimage* string,  $Y$  is an *image* string, and  $Z$  is a *value* number. A *preimage* is a string containing of random string and the current time-stamp<sup>4</sup>. An *image* is the SHA1 hash-value of the concatenation of the magic string 'z9hG4bK'<sup>5</sup> and *preimage*<sup>6</sup><sup>7</sup>. The *work* represents how much the client has to 'work' in order to solve the puzzle. For example, if the *image* is 10110101011011 and the work is 5, it simply means that the last five digits will be set to zero so that 10110101011011 will be 10110101000000. Then this will be the *preimage* that the server will send to the client/attacker while the original is kept. The *value* refers to how many bits match in the solution and is set to 160 since this is the maximum number of a SHA1 hash result.

So, a puzzle is a four tuple  $[W, X, Y, Z]$ <sup>8</sup>:  
*work*  $W$  = any integer from 1 - 160.

4 Some implementations also include URI of the request. [<http://tools.ietf.org/html/draft-jennings-sip-hashcash-06>]

5 It is called the 'magic cookie' in SIP protocol. Please refer to [www.z9hg4bk.org](http://www.z9hg4bk.org) for details.

6 The fact that the puzzle uses SHA-1 hash is important since SHA-1 is *generally* preimage resistant. The key thing to notice here is that by having preimage resistant hash function, it means that creating the puzzle is much easier than solving it which is very important property of the puzzle in handling DoS attacks. [Li, 2007]

7 The term "generally" in the previous footnote applies since there are reports that say collisions exist such as [wikipedia-crypto-hash-function] and [Schneier, 1996]. For more information about the properties of secure hash function in general or SHA-1 in particular, please refer to [Menezes, 1996].

8 Note that the puzzle is based on SHA1 hash function. If one would use other hash function, then  $Z$  and , consequently, the range of  $W$  might have to change.

*preimage X* = SHA1(random string | current time-stamp).  
*image Y* = SHA1("z9hG4bK" | X)  
*value Z* = 160.

## 4.2 Solving the puzzle

A puzzle is solved when the string solution matches the string *image*. Let's take the following puzzle as an example:

work value  $W = 10$ .

preimage string  $X = \text{SHA1}(\text{random string} \mid \text{current time-stamp} \mid \text{URI from INVITE})$   

$$= 01010110\ 00000101\ 01000110\ 01100010\ 00101100\ 01011011\ 01000110$$

$$00001101\ 00100110\ 01110101\ 00101100\ 00010011\ 01100011\ 01110110$$

$$00001000\ 01111100\ 00100000\ 01101110\ 01100010\ 01101000$$

$$= \text{"VgVGYixbRg0mdSwTY3YIfCBuYmg="} \quad (\text{Base64 Encoded})$$

image string  $Y = \text{SHA1}(\text{"z9hG4bK"} \mid X)$   

$$= \text{"NhhMQ2l7SE0VBmZFKksUC19ia04="} \quad (\text{Base64 Encoded})$$

value  $Z = 160$

The puzzle above is sent to the client with the 10 last bits of preimage string is set to 0:

$$01010110\ 00000101\ 01000110\ 01100010\ 00101100\ 01011011\ 01000110$$

$$00001101\ 00100110\ 01110101\ 00101100\ 00010011\ 01100011\ 01110110$$

$$00001000\ 01111100\ 00100000\ 01101110\ 01100000\ \underline{00000000},$$

which is equivalent to (Base64 Encoded) "VgVGYixbRg0mdSwTY3YIfCBuAAA=".

Thus the client will receive a puzzle:  $[W, X, Y, Z] = [10, \text{"VgVGYixbRg0mdSwTY3YIfCBuAAA="}, \text{"NhhMQ2l7SE0VBmZFKksUC19ia04="}, 160]$  and then check if  $\text{SHA1}(\text{"z9hG4bK"} \mid X) = Y$ . If they are not the same, then the client will decode  $X$  into its binary equivalent, add 1, encode it into  $X'$  and check again if  $\text{SHA1}(\text{"z9hG4bK"} \mid X') = Y$ .

To illustrate, the binary equivalent of the received  $X = \text{"VgVGYixbRg0mdSwTY3YIfCBuAAA="}$

“ 01010110 00000101 01000110 01100010 00101100 01011011 01000110

```
00001101 00100110 01110101 00101100 00010011 01100011 01110110
00001000 01111100 00100000 01101110 01100000 00000000 ”
```

When we are adding 1 to X, it will become:

```
“ 01010110 00000101 01000110 01100010 00101100 01011011 01000110
00001101 00100110 01110101 00101100 00010011 01100011 01110110
00001000 01111100 00100000 01101110 01100000 00000001 ”
```

But,  $\text{SHA1}(\text{"z9hG4bK"} \mid X) = Y$  is still false. Then we add 1 again, so that X:

```
“ 01010110 00000101 01000110 01100010 00101100 01011011 01000110
00001101 00100110 01110101 00101100 00010011 01100011 01110110
00001000 01111100 00100000 01101110 01100000 00000010 ”
```

We keep adding 1 until  $\text{SHA1}(\text{"z9hG4bK"} \mid X) = Y$  is true. Once the client finds the value of the X such that  $\text{SHA1}(\text{"z9hG4bK"} \mid X) = Y$ , then the client will send the puzzle with the solution [W, X, Y, Z] back to the server and the server will check whether the received X match the original preimage string.

### 4.3 DoS attacks and how the puzzle handles them

The Internet has become the very place where we do everything. Nowadays, it is very common to do most of our activities over the Internet such as reading news, buying books, exchanging stock markets, personal banking or even studying to get our bachelor degree. This has made the Internet becomes a very interesting environment for the attackers, especially to disrupt it or even make it unavailable to the people.

Denial-of-service or DoS is getting the attention in the Internet world as one of the most dangerous attacks [Cole, 2002]. Any IP-based networks are vulnerable to the DoS attack. As the name implies, the main goal of DoS attack is to disrupt or deny some services requested by users from a server. DoS attacks are usually achieved by overwhelming servers with so many requests or messages so that servers will slow down or eventually crash, causing it unavailable to give the service that legitimate users ask. This may disrupt services and cause them to not operate in the way they should.

Imagine if the server of a computer store is attacked with a DoS attack, causing it to go down for 1 hour. Within this hour, the store may lose many customers wanting to buy computer online. How much revenue will the store lose due to this attack? Another example would be a server of an online university attacked with a DoS attack and causes the server go down for 3 hours. How many students are prevented from their study activities? What if there is an exam on-line and suddenly the connection is lost?<sup>9</sup>

DoS attacks are deadly. Besides the fact that it is relatively easy to launch, its effect can range from a mild disruption to a system crash. We know that DoS attacks involve flooding some packets into a server, forcing the server to allocate some resources in order to process the packets. Thus, DoS attacks create a very high volume of traffic. It gets more complicated when the attacker will flood the server with different types of packets. Instead of flooding a 'virus' packet, the attacker can also send a 'worm'. These give us general ideas how to tackle DoS attacks. We can think that if we can put a “filter” so that we can profile all incoming packets, process the legit ones and dump the rest, the DoS attacks will not work. We can also think that if we can slowdown the rate of incoming packets so that the server does not have to exhaust its resources, then DoS attacks will not work as well. The puzzle uses the second approach.

The puzzle mechanism will slowdown the attacker because now the attacker has to divide its resources between sending the packets and solving the puzzle. Even though creating a puzzle requires some resources, the work that the attacker has to put in to solve the puzzle requires much more work. The ratio of resource usage of the server and of the attacker is large and thus will slowdown the speed of packet-flooding activity of the attacker [Li, 2007].

---

<sup>9</sup> In real life, the effects from these examples are usually achieved by launching DoS from many machines which is well-known as the distributed denial-of-service attacks (DDoS). This paper, however, focuses on preventing DoS attack launched from one machine, not the distributed one and because of this, the term DoS is kept instead of DDoS for simplicity

## Chapter 5: Puzzle Testing and Implementation

### 5.1: Environment and Attack Scenarios

In this chapter, we will simulate how puzzle implementation can mitigate DoS attacks. We will simulate how an attacker launches a DoS attack against a VoIP server. Here is the environment of our simulation:

Server	: Asterisk 1.4.18
Server IP address	: 192.168.0.100
Protocol	: SIP
Request type	: INVITE
Network analyzer	: Wireshark 0.99.7
DoS attack type	: Flooding Asterisk with INVITE messages
Attacking Program	: inviteflood <sup>10</sup>
Puzzle mechanism	: PuzzleServer.java, PuzzleMaker.java
Puzzle solver	: PuzzleClient.java <sup>11</sup>
Attacking Machine IP address	: 192.168.0.103
Processor	: AMD Turion 64 1.7 Ghz
Memory	: 512 MB
Client 1	: GXP 200 hardphone

---

<sup>10</sup> Available at [www.hackingvoip.com](http://www.hackingvoip.com)

<sup>11</sup> Due to the size of the programs such as PuzzleMaker, PuzzleServer, PuzzleClient, inviteflood, and all programs that are related, their source code will not be shown in this paper. Please ask the writer for the source-code at: fsutomo@gmail.com.

Client 1 IP address : 192.168.0.101  
client 2 : Linksys WIP 300 Voip Phone  
Client 2 IP address : 192.168.0.105  
Client 3 : Xlite  
Client 3 IP address : 192.168.0.103

The first scenario will begin with the attacker floods Asterisk with 500,000 INVITE<sup>12</sup> packets. The Asterisk, at first, will be without puzzle mechanism. We will analyze the attack as well as the effects after the attack. The second scenario will have puzzle mechanism implemented in Asterisk with the attacker only sends 100,000 INVITE packets. We will analyze how effective the puzzle mechanism is in mitigating DoS attacks. Figure 2 depicts our testing environment.

## 5.2: Testing

### - Scenario 1

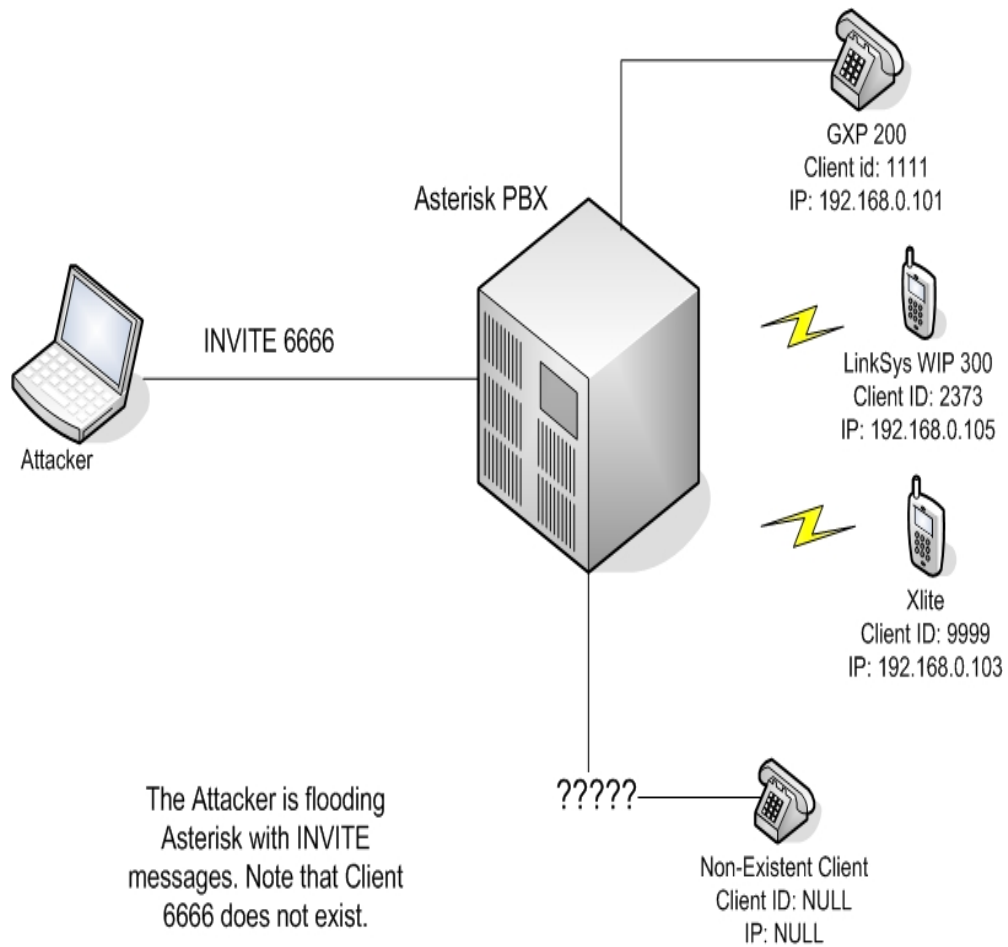
Execute the packet-flooding program, the *inviteflood*:

```
./inviteflood eth1 6666 192.168.0.100 192.168.0.100 500000
```

What *inviteflood* is doing is that it will send 500,000 INVITE messages through interface *eth1* for non-existent client 6666 to the server, whose IP address is 192.168.0.100.

---

<sup>12</sup> INVITE is one of the messages of Session-Initiation Protocol (SIP). For more information about INVITE and SIP, please refer to RFC 3261.



*Figure 2: Testing Environment*

We experimented the scenario 1 three times and the result is shown in Table 1.

No	# of packets	Sending time (s)	Interval (s)	# of call attempts	# call failed	Details
1	500,000	45	1-45	9	9	500:Server Internal Error
			45-50	3	2	500:Server Internal Error
			> 50	-	-	Back to normal
2	500,000	45	1-45	10	10	500:Server Internal Error
			45-50	2	2	500:Server Internal Error
			50-55	2	1	500:Server Internal Error
			> 55	-	-	Back to normal
3	500,000	46	1-46	11	11	500:Server Internal Error
			46-50	2	2	500:Server Internal Error
			50-55	2	2	500:Server Internal Error
			55-60	2	1	500:Server Internal Error
			> 60	-	-	Back to normal

*Table 1: Sending 500,000 INVITE packets to a non-existent client*

From Table 1, we know that while the packets were being sent (1-45s), it was (almost) impossible to place a call. In fact, approximately 13s after all packets were being sent, only several calls could go through. Figure 3 also shows Wireshark gave “500: Server Internal Error” showing that the server crashed due to the attack.

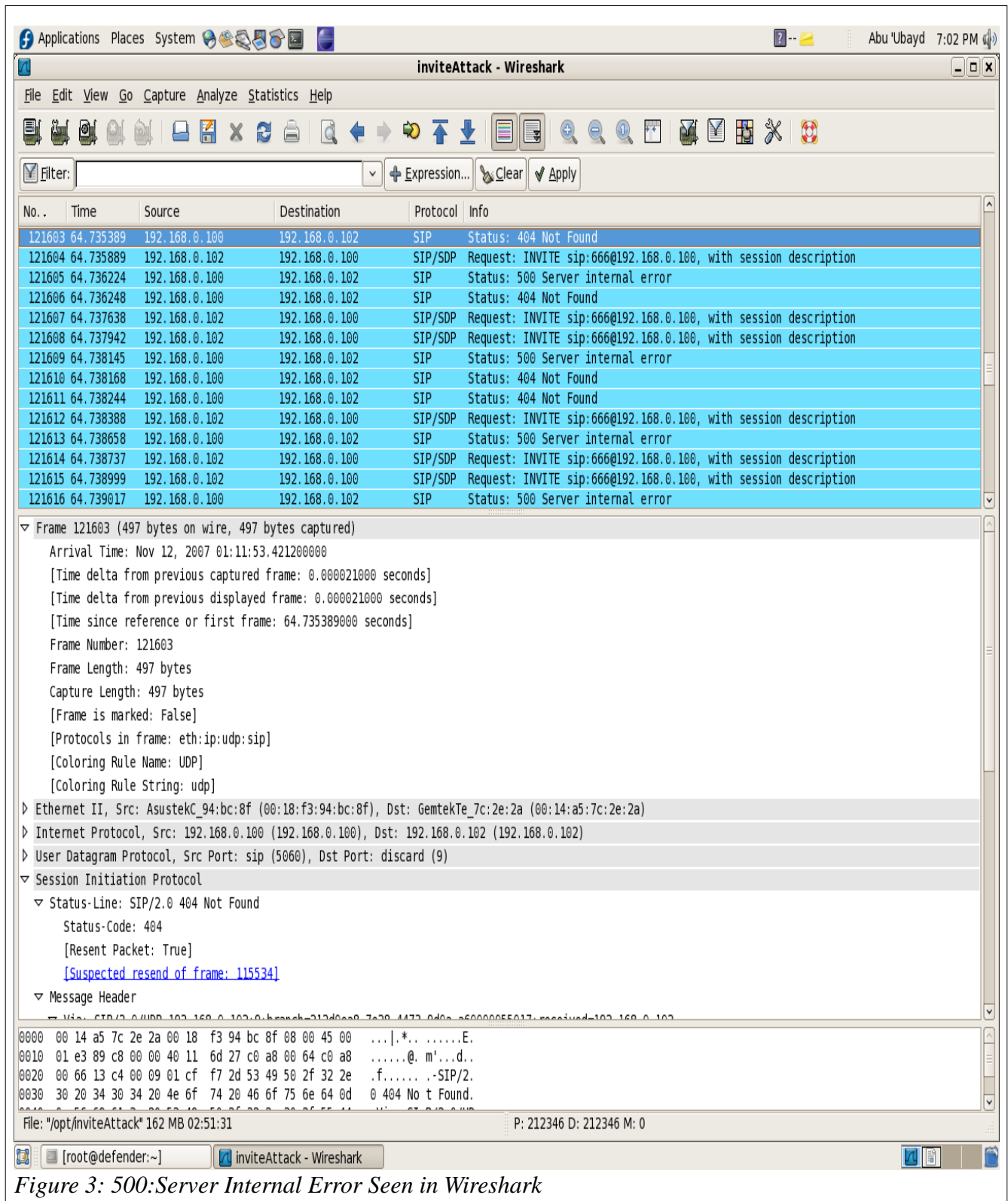


Figure 3: 500:Server Internal Error Seen in Wireshark

Before we test Scenario 2, we will explain first how the puzzle is implemented inside Asterisk. The puzzle mechanism will be called by Asterisk with the command:

```
fadil puzzle set workNumber <integer>13
```

With this command, Asterisk will activate PuzzleServer and PuzzleServer will ask PuzzleMaker to create a puzzle with *workNumber* = <integer>. Figure 4 shows us the architecture of the puzzle mechanism implementation in Asterisk to mitigate DoS attacks launched by the attacker.

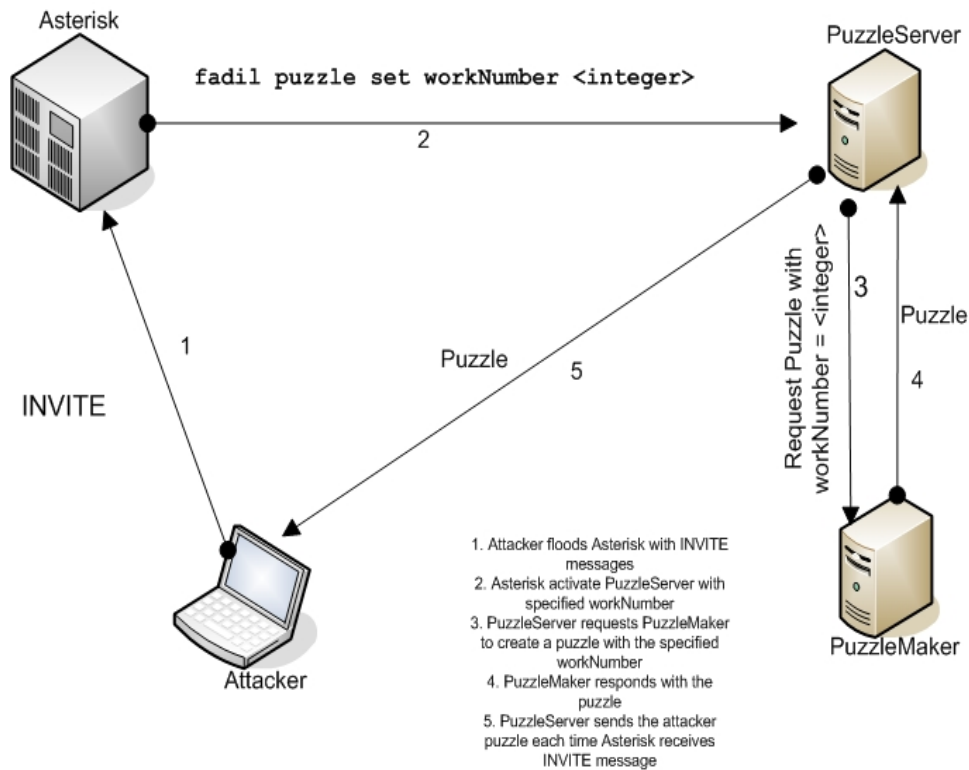


Figure 4: How our puzzle mechanism works

We will go now to Scenario 2.

## - Scenario 2

Activate the PuzzleServer with *workNumber* = 5.

---

<sup>13</sup> Please refer to the Appendix A for more details

Execute the packet-flooding program, the *inviteflood*:

```
./inviteflood eth1 6666 192.168.0.100 192.168.0.100 100000 true
```

What *inviteflood* is doing is that it sends 100,000 INVITE messages through interface *eth1* for non-existent client 6666 to the server, whose IP address is 192.168.0.100. But this time, the attacker's machine will have a *PuzzleClient* installed (notice the `true` at the end of the command). We will also analyze when the *workNumber* is changed to 7. The result is in Table 2.

No	# of packets	Puzzle size	Sending Time (s)	Interval (s)	# of call attempts	# of call failed	Details
1	100,000	5	128	1-124	42	0	Every call went through
				124-130	2	2	500:Server Internal Error
				130-135	2	2	500:Server Internal Error
				> 135	-	-	Back to normal
2	100,000	5	124	1-122	45	0	Every call went through
				122-125	2	2	500:Server Internal Error
				125-130	3	3	500:Server Internal Error
				> 130	-	-	Back to normal
3	100,000	7	300	1-300	102	0	Every call went through
				300-305	3	1	500:Server Internal Error
				> 305	-	-	Back to normal

*Table 2: Result with puzzle mechanism*

We can see from Table 2 that by using puzzle mechanism, DoS attacks are greatly mitigated. In fact, by increasing the *workNumber* from 5 to 7, the time needed for the attacker to flood Asterisk was more than doubled and from 105 call attempts only 1 call that did not go through. This table clearly shows us how effective the puzzle mechanism is in mitigating DoS attacks.

## **Chapter 6: Conclusion**

### **6.1 Discrepancies and future works**

Since our implementation is only for prototype and to show the effectiveness of the puzzle mechanism, there are many discrepancies that should be addressed in the future. In our implementation, Asterisk

will process all incoming INVITE messages , including those sent by attackers, without authenticating the puzzle solution. This will create a chance for attackers to have their message processed without solving the puzzle. Attackers can build bogus solutions and send them each time they receive a puzzle. In its ideal implementation, Asterisk should only process messages with a valid puzzle solution. To achieve this goal, our implementation should have a two-way interaction between Asterisk and PuzzleServer instead of a one-way interaction. With a two-way interaction, PuzzleServer can tell Asterisk if a sender has solved the puzzle from a legitimate client so that Asterisk can process the message.

Another deficiency of our mechanism is that PuzzleServer is vulnerable to DoS attacks. Attackers can switch their target from crashing Asterisk to crashing PuzzleServer (and subsequently Asterisk) by flooding bogus puzzle solutions. In our implementation, we are trying to mitigate DoS attacks by introducing something which is potential to DoS attacks. Waters *et al* suggests a very nice modification in his paper [Waters et al, 2004] where he discusses that the PuzzleServer should be “outsourced to a secure, DoS-resistant entity”. Waters also makes some suggestions to make the puzzle distribution is based on public-key infrastructure (PKI). Moreover, he also introduces the notion of time-based “channels” in which a legitimate client can only connect to the server through these channels. By using time-based channels, then attackers have restricted time to complete their attacks. For more detail explanation about this work, please refer to [Waters et al, 2004]. Figure 5 below shows us how the puzzle implementation should be.

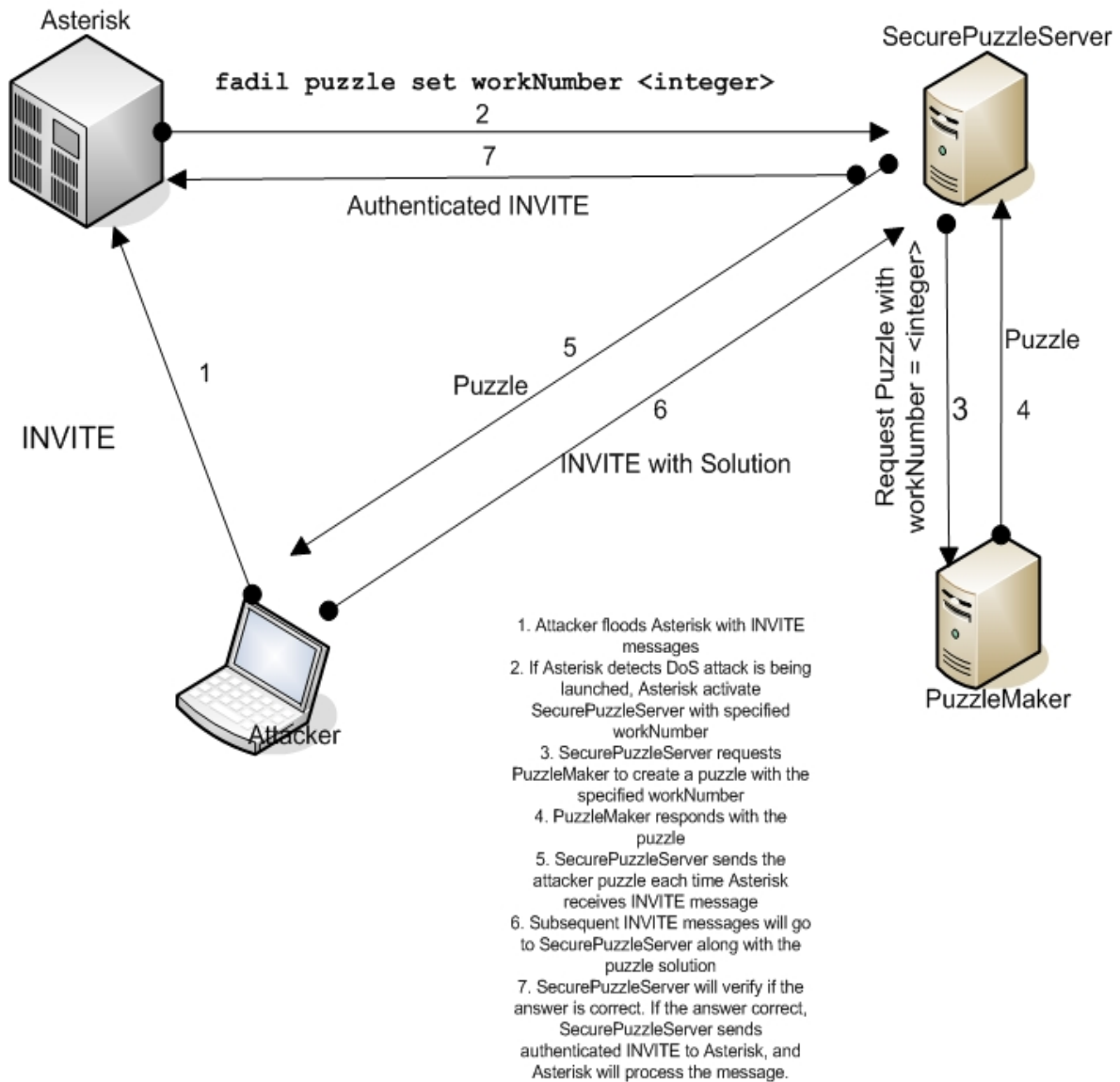


Figure 5: The ideal implementation

## 6.2 Conclusion

As VoIP technologies are widely spreading, it is expected that people and enterprises will switch to

VoIP. Consequently, the presence of security features for VoIP servers becomes critical as a lot of sensitive data are being transmitted over VoIP networks. This project has given a general picture of VoIP availability issue and a method to handle it which is the puzzle mechanism. To a certain extent, this project has also proven that the puzzle mechanism is an effective method in mitigating denial-of-service attacks. Although this project does not have the ideal implementation of the puzzle mechanism, but it is still useful as a prototype as well as a real-case scenario example in securing a VoIP server.

## **References**

Adams, C., and S. Lloyd. *Understanding PKI: Concepts, Standards and Deployment Considerations*. Addison Wesley, 2002.

Butcher, D., et al. "Security Challenges and Defense in VoIP infrastructure." *IEEE Transactions on*

*Systems, Man, and Cybernetics, Part C: Applications and Reviews*. Vol. 37, pp 1152-1162, 2007.

Clayton, B. "Securing media streams in an Asterisk-based environment and evaluating the resulting performance cost." Master's Thesis, Rhodes University, Grahamstown, South Africa, 2007.

Cole, E. *Hackers Beware: The Ultimate Guide to Network Security*. New Riders Publishing, 2002.

Davidson, J. *Voice Over IP Fundamentals*. Cisco Press, 2000.

Dean, D., "Using Client Puzzles to Protect TLS." *Proceedings of the 10th conference on USENIX Security Symposium*. Vol. 10, pp 1-1, 2001.

Dubrawsky, I., et al. *CompTIA Security+ Exam JK0-010: Study Guide and Practice Exam*. Syngress Publishing, 2007.

Endler, D., and M. Collier. *Hacking Exposed VoIP: Voice Over IP Security Secrets and Solutions*. McGraw-Hill Osborne, 2007.

Juels, A., and J. Brainard. "Client puzzles: A cryptographic countermeasure against connection depletion attacks." *Proceedings of NDSS '99 (Networks and Distributed Security Systems)*, pp 151-165, 1999.

Knudsen, J. *Java Cryptography*. O'Reilly, 1998.

Kopsidas, S., D. Zisiadis, and L. Tassioulas. "A Secure VoIP Conference System: Architecture Analysis and Design Issues." *Proceedings of the 3rd ACM workshop on QoS and security for wireless and mobile networks*, pp 180 – 183, 2007.

Kuhn, D. R., T. J. Walsh, and Steffen Fries. "Security Considerations for Voice over IP Systems Recommendations of the National Institute of Standards and Technology ." *NIST Special Publication*, SP 800-58, 2005.

Lehtinen, R., and G. T. Gangemi, Sr. *Computer Security Basics*. O'Reilly, 2006.

Li, Y., and J.H Chen. "A Denial-of-Service Resistant Authentication Solution for Security Protocols." [Workshop on Intelligent Information Technology Application](#), 2007.

Maynor, D., et al. *Emerging Threat Analysis: From Mischief to Malicious*. Syngress Publishing, 2006.

Menezes, A. J., Paul V. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

Mirkovic, J., S. Dietrich, D. Dittrich, P. Reiher. *Internet Denial of Service: Attack and Defense Mechanisms*. Prentice Hall, 2004.

Porter, T. *Practical VoIP Security*. Syngress Publishing, 2006.

Reilly, D. and M. Reilly. *Java Network Programming and Distributed Computing*. Addison-Wesley,

2002.

Schneier, B. *Applied Cryptography*. John-Wiley and Sons, 1996.

Sicker, D. C., and Tom Lookabaugh. "Voip Security: Not an Afterthought." *ACM Queue*, Vol. 2, Issue 6, pp 56 – 64, 2004.

Thermos, P., and A. Takanen. *Securing VoIP Networks: Threat, Vulnerabilities and Countermeasures*. Addison Wesley, Boston, MA, 2007.

Wallingford, T. *Switching to VoIP*. O'Reilly, 2005.

Waters, B., et al. "New Client Puzzle Outsourcing Techniques." *Proceedings of the 11th ACM conference on Computer and communications security*, pp 246 – 256, 2004.

<http://tools.ietf.org/html/draft-jennings-sip-hashcash-06>

[http://en.wikipedia.org/wiki/Cryptographic\\_hash\\_function](http://en.wikipedia.org/wiki/Cryptographic_hash_function)

[http://en.wikipedia.org/wiki/Information\\_security](http://en.wikipedia.org/wiki/Information_security)

[http://en.wikipedia.org/wiki/Access\\_control](http://en.wikipedia.org/wiki/Access_control)

<http://www.silicon.com/research/specialreports/voipsecurity/0,3800013656,39166479,00.htm>

RFC 3261

[www.java.happycodings.com/Core\\_Java/code3.html](http://www.java.happycodings.com/Core_Java/code3.html)

## **Appendix A: Adding fadil puzzle set workNumber <integer> into Asterisk**

### **Command Line Interface (CLI)**

Step 1: Define fadil puzzle set workNumber <integer> in Asterisk compiler flags

Details:

– go to /usr/src/asterisk-X.X.X/build\_tools/cflags.xml

- add the following:

```
<member name="FADIL_PUZZLE_SET_WORKNUMBER" displayname = "Set  
the worknumber of the puzzle">  
    <defaultenabled>yes</defaultenabled>  
</member>
```

Step 2: Modify `asterisk.c` (changes to `asterisk.c` will be give in Appendix B)

Step 3: Recompile Asterisk with `make`

Step 4: Reinstall Asterisk with `make install`

## **Appendix B: Changes to `asterisk.c`:**

- in `remoteconsolehandler` function:

```
static char fadil_puzzle_set_worknumber_help[] =
```

```
"Usage: fadil_puzzle_set_worknumber <integer-from-0-to-160>\n"
```

```
"Example: fadil_puzzle_set_worknumber 5\n";
```

- create a handler function for fadil puzzle set workNumber <integer>:

```
/* from fadil
```

```
 * This is the function if fadil_puzzle_set_worknumber is invoked  
from Asterisk CLI
```

```
 */
```

```
static int handle_fadil_puzzle_set_worknumber(int fd, int argc, char  
*argv[])
```

```
{
```

```
 //ast_cli(fd, "%d\n", argc);
```

```
 if(argc != 5)
```

```
     return RESULT_SHOWUSAGE;
```

```
     ast_cli(fd, "Puzzle Mechanism is on! Asterisk is now  
using  
puzzle with worknumber is set to %s \n",  
argv[4]);
```

```
 //pass the workNumber get from CLI into the shell.
```

```
 char command[] =  
"/home/fadil/Desktop/Comp4905/PuzzleProgram/executer-helper.sh ";
```

```
 strcat(command, argv[4]);
```

```
 system(command);
```

```
 return RESULT_SUCCESS;
```

```
}
```

- in ast\_cli\_entry cli\_asterisk[]:

```
 /*****
```

\*\*\*\*\*

\* from fadil

\* fadil\_puzzle\_set\_worknumber

\*/

{ { "fadil", "puzzle", "set", "worknumber", NULL },

handle\_fadil\_puzzle\_set\_worknumber, "Set the puzzle's  
worknumber",

fadil\_puzzle\_set\_worknumber\_help},