Interception-proof VoIP using dictionary-based encryption

Ahmad Hammoud, Daniel Bourget

Interception-proof VoIP using Dictionary-based Encryption

Ahmad Hammoud, Daniel Bourget

Abstract – **Due to its many advantages, VoIP is nowadays replacing the traditional analog communication. Although it is a promising technology, it suffers from a big disadvantage that makes wire tapping or sniffing much easier. The only counter-measure is encryption. Many companies provided plenty of products to minimize the chance for sniffers to receive readable packets. Unfortunately, those chances are minimized but not yet eliminated. Low enforcement agencies and the products' manufacturers are still capable of penetrating. In this article, we provide a way through which VoIP communication will be encrypted using a user-defined dictionary. The purpose of this article is to show to what extent such an idea can provide sufficient immunity.**

Keywords: Interception, Sniffing, Encryption, VoIP, Wire Tapping.

I. INTRODUCTION

VoIP is an excellent technology. Its main drawback is the fact that it travels on the LAN, WAN, WLAN, Internet, and other exposed networks that are accessible 24 hours a day. That's why encryption is a major concern [5]. All law enforcement agencies are highly interested in "key escrow" or "key-recovery" mandates [3]. CALEA (the Communications Assistance for Law Enforcement Act) is a United States wiretapping law. The sole purpose of CALEA is enhancing the ability of intelligence and law enforcement agencies to conduct electronic surveillance. CALEA does not just require telecommunication carriers to cooperate with it by allowing US government to wire tap all communication channels. It goes far beyond than this. It requires telecommunication carriers and manufacturers of telecommunication equipment to modify and design their facilities, services, and equipment so that CALEA has built-in surveillance capabilities. This would allow federal agencies to monitor all broadband internet, telephone, and VoIP traffic in real-time [1]. CALEA forced phone companies to modify hardware and software in their systems. As a result, the U.S. Congress had to fund such network upgrades. CALEA came into force on the first of January 1995. The pressure exerted by CALEA became stronger after September 11, 2000. This implies that commercial encryption solutions cannot be blindly considered safe and secure products. Thus, there is a crucial need for a solution that does not rely on commercial encryption boxes since manufacturers are required to provide the US government with the keys of their implanted back doors. This necessitates having a solution that provides maximum security. What we provide in this article is a solution through which a phone call is encrypted using a user-defined algorithm. We will create a user-defined dictionary. The caller will use this dictionary to encrypt his/her voice. Then, the encrypted voice will be transferred to the callee who will use his/her own copy of the same dictionary to decrypt the vocal message. Our approach does not rule out the need for VPN or encryption techniques and is not meant to replace any commercial product. All of those solutions can still be in use since many lines of defense are recommended [4]. Our only purpose is to try to find a solution that does not rely on commercial products.

There is a good reason why we focus on VoIP while the same approach can be applied to any other type of communication such as emails. The person's voice is a valid evidence. A country's ambassador might avoid having sensitive conversations on the phone because he/she is afraid of wire tapping. Sometimes, even individuals who are aware of the danger of wiretapping use the phone to convey sensitive information [2]. If an electronic message is caught by the media or an intelligence agency, it will not have the same effect as a vocal message because there is no way to prove that this email was sent by the person him/herself.

II. THE PROPOSED SOLUTION

Dictionary-based encryption is secure since it does not rely on mathematical methods that can be cracked. Our approach is very simple:

1. The caller will dial a number.
2. Asterisk will then start recording the voice of the caller.
3. When the caller is done, Asterisk will call an external program that will encrypt the recorded sound file.
4. The sound file is then sent to Asterisk on the other side where the callee is waiting to hear the caller's voice.
5. Asterisk on the callee's side will call a program to decrypt the message.
6. When the message is decrypted, Asterisk will play it to the callee.

Each of the two communicating parties has his/her own LAN that is protected behind a firewall. It would be better not to rely on commercial firewalls. Open source ones are a better choice since there is a good chance that commercial black boxes include CALEA-compliant interfaces. The presence of a firewall will protect the dictionary from being exposed to the outside world.

It is obvious that this approach sacrifices the real-time responsiveness that is required in normal VoIP scenarios. In this article, we study how efficient our

approach is. We will implement many enhancements and try to optimize the algorithm to see if the real time constraint can be achieved. We believe two or even three seconds of latency would be accepted since the gain is a interception-proof phone call.

III. Basic Scenario

Basically, the idea is to encrypt a digital media file using an algorithm different from those implemented in commercial products. In this article, we provide an algorithm to read chunks of the source media file, encrypt each byte of the chunks, and save them to another file. The encrypted file will be sent to the callee. When the file is received on the other side, bytes are decrypted and the media file is played. Using the CURL function, Asterisk will call an ASPX page which will in turn encrypt the content of the media file. The C# code that we used to retrieve the chunks is shown in "Fig. 1".

```
FileStream fsw = new FileStream(dest,
FileMode.OpenOrCreate, FileAccess.Write);
BinaryWriter bw = new BinaryWriter(fsw);

// The size of the "chunks"
int bufferLen = 128;

FileStream fsr = new FileStream(src,
FileMode.Open, FileAccess.Read);
BinaryReader br = new BinaryReader(fsr);

byte[] buffer = br.ReadBytes(bufferLen);

while (buffer.Length > 0)
{
  for (int i = 0; i < buffer.Length; i++)
    { buffer[i]++; }

  bw.Write(buffer);
  bw.Flush();
  buffer = br.ReadBytes(bufferLen);
}
// Close br, bw, fsw, fsr
```

Fig. 1 – Source Code of the ASPX Page that retrieve the chunks

The code shown in "Fig. 1" does the simplest form of encryption. It only adds a value of one to each byte in the array. Our goal is to use a user-defined dictionary for the encryption. Such a dictionary will be powered by some RDBMS package. Each time a byte needs to be replaced, the value is sent to the database that will return a corresponding value. Consulting the database thousands of times will no doubt have a huge impact on the performance.

We have implemented our approach and tested it using different hardware, operating systems, DBMS packages, and programming technologies. "Table 1" shows the different configurations used.

Table 1 – Servers Configurations

Hardware	Operating System	Web Technology	DBMS
Pentium IV 2.8 GHz 2 GB of RAM	Windows Server 2003	IIS 6 ASP	SQL Server 2000
Pentium IV 3.2 GHz 4 GB of RAM	Windows Server 2003	IIS 6 ASP.NET	SQL Server 2005
Pentium IV 3.2 GHz 3 GB of RAM	Linux Fedora 9	Apache PHP	MySQL
2 Quad-Core Xeon 2.5Ghz 8 GB of RAM	Linux Fedora 9	IIS on another separate server	Oracle Database 11g

The reason why we used all of those different configurations is the fact that the runtime of our first executions was not accepted at all. Using a dictionary to encrypt each byte slows down the whole process to an extent that it becomes unacceptable.

IV. Enhancements

In order to achieve our goal which is minimizing the latency to 2 or 3 seconds, we had to come up with some enhancements. All the improvements aimed only at achieving better runtime on both sides: caller (encryption) and callee (decryption).

Even with the powerful HW configuration shown in the last row of table 1, the runtime is still unacceptable. If the format of the recorded media file is gsm, an average phrase will need 30 to 40 KB. This means we have around 35,000 bytes that need to be encrypted. Fetching the database 35,000 times cannot be done in real time.

There is an obvious need to implement a set of enhancements. Those enhancements are listed in "Table 2" along with some details.

Table 2 – Enhancements

Enhancement	Details
Algorithm Fine Tuning	The main problem is that we need to access the database thousands of times. If there is a way to minimize this number, runtime will be better. For example, encrypting every other byte reduces the time to its half. It is so important to decide on the percent of

	the bytes that will be encrypted.
Parallel Execution	We implemented an array of servers, each of which will receive a part of the problem. Using .NET threading, we sent each chunk to a different server. That way, many servers will be working at the same time. We used 4 servers that helped us reduce the required runtime by 75%.
DB Design Improvement	Instead of storing the entire dictionary in the same table, we decided to partition the data. Since each byte is only a number between 0 and 255, we can create 256 tables each of which will contain a long list of numbers equivalent to the name of the table. For example, there will be a table called [17]. It might include 100,000 records. Every record is only one number. All of those numbers represent the encrypted 17. Whenever a byte is equal to 17, the first row in table 17 will be considered the encrypted byte. That way, we will not execute a statement that will retrieve data from a table containing millions of records. We will simply retrieve the first row. Numbers in all of the 256 tables should not overlap. Otherwise, we will not be able to get back the original value when it is the time to decrypt.
Code Optimization	We do not want to use the same record twice. If the value 17 is encrypted to 1234, there is no chance that the number 1234 will be used again. This dictates that we need to delete each number that is used in the encryption process. Such a deletion has a considerable impact and will slow down the encryption process. We have a suggestion to postpone this deletion.

The last enhancement presented in "Table 2" necessitates postponing the deletion until after the encrypted file is created. When the file is sent, deletion should start since it will not slow down the encryption process anymore. As explained in the 3rd enhancement presented in "Table 2", we will have 256 tables. When we use the first number in a table, we will not delete this record. Instead, we will keep a counter (say, C) that tells the number of used records. When the encryption is done, we will delete the first C records from the table. To do so, we created a multi-dimensional array to store the counters. Since we want to achieve parallel execution, the first dimension in the array is for the servers while the second is for the 256 tables. At any moment, each cell in the array will store the current position in the corresponding table. For example, the cell [2,256] will store the current position of the 256th table of the 2nd server. This means that when the encryption is done, we have to delete the first 2 records of that table.

When we applied all the above enhancements we got an acceptable latency. On average, it was less than 4 seconds; however, occasionally it went above 6. The reason is still unclear. There is a plenty of factors that might have caused this. The duration and the format of the sound file are among the most important factors.

V. LIMTATIONS

There are some limitations that affect the efficiency and responsiveness of our proposal. They are as follows:

- Latency – our approach does not provide real time VoIP phone calls. The caller will say something and wait for the callee's response. Then, the callee will respond and wait for the caller's reply. This is not a real time conversation but it is close to real time as the latency is less than 3 seconds. It is a huge price but the gain is getting interception-proof VoIP phone calls.
- Another limitation is the fact that the dictionary should exist on both sides. Such a dictionary should not be sent over the internet. Instead, it should be passed by hand.
- The core component of the security of our encryption algorithm is the dictionary. If this dictionary is unveiled, the whole process is jeopardized.
- The last limitation is the need for multiple powerful servers. Based on many tests we have made, each server can be replaced by many PCs. The algorithm shown in "Fig. 1" can, for example, split the file into 30 chunks and send each chunk to a different PC. Each PC will work on encrypting just one chunk. That way, average PCs can replace powerful servers.

VI. CONCLUSION

In this article, we presented the way through which we could achieve encrypted VoIP phone calls. The caller will talk while Asterisk will be recording his/her voice. Then, the recorded file will be encrypted based on a user-defined dictionary that has a corresponding value to each byte. The major problem was in runtime. However, we could introduce a set of enhancements that led to a better runtime. Those enhancements are encrypting a byte out of

each 3, using 4 parallel servers each encrypting a part of the file, creating a table for each byte value, and postponing the deletion.

VII. Future Work

Our lab tests were promising since we could achieve an acceptable latency, but we still have to get better results. Our simulated scenarios showed that our approach was valid; however, implementing such a solution in the real world is far more complicated than doing so in controlled environments. Our next step is to try to implement our approach in a call center to test how efficient, stable, and available it is. Another thing to investigate is loading the whole dictionary database in the RAM so that accessing it would be fast and efficient. This approach has a drawback which is consuming most of the RAM. The size of the database, available RAM, number of used servers, dictionary loading time, and other factors need to be taken into consideration.

REFERENCES

[1] Communications Assistance for Law Enforcement Act. (2009, November 9). In Wikipedia, The Free Encyclopedia. Retrieved from http://en.wikipedia.org/w/index.php?title=Communications_Assistance_for_Law_Enforcement_Act&oldid=324850309

[2] Diffie,Whitfield,& Landau, Susan. Privacy on the Line: The Politics of Wiretapping and Encryption. Retrieved from http://books.google.com/books?hl=en&lr=&id=nMY8yHaTQi4C&oi=fnd&pg=PR9&dq=encrypting+voip+is+more+important+than+other+communication&ots=DP_YEVDgch&sig=EY8psxSwT9nPKOa1RgiVTQQGo8M#v=onepage&q=&f=false

[3] Singleton, Solveig. (1998). ENCRYPTION POLICY FOR THE 21ST CENTURY: A Future without Government-Prescribed Key Recovery. Policy Analysis, 325. Retrieved from http://www.cato.org/pubs/pas/pa325.pdf

[4] Technolgy, Media and Telecomunications. (2006). Deloite: Protecting the Digital Assets. Retrieved from http://www.deloitte.com/assets/Dcom-Global/Local%20Assets/Document

[5] Tucker, Greg S. (2005). Voice Over Internet Protocol (VoIP) and Security. Retrieved from http://www.securitytechnet.com/resource/hot-topic/voip/1513.pdf

4