Developing an application to covertly transmit data within a VoIP session

Aidan Melvin

Mentored by Mike Bowen, Will Newton, and Emily Abid

Introduction

With the increasingly widespread use of VoIP (Voice over Internet Protocol) applications such as Zoom and Skype, maintaining the privacy of transmitted information continues to be of increasing importance. In areas with heavily monitored network traffic, the ability to covertly send and receive information is vital to those who wish to maintain secrecy.

Steganography is the art of hiding information in plain sight. A common example is image steganography, where the individual pixels of an image are modified imperceptibly to encode data. Steganalysis, on the other hand, is a class of techniques to detect steganography. Steganalysis typically relies on the use of statistical techniques to analyze properties of transmitted data to determine the probability that steganography is present. Note that, although they can be combined, steganography and cryptography are different and often confused. Steganography aims to transmit data in a hidden form. That is, if a third party intercepts a message with steganography applied to it, they could easily see the hidden information only if they know that it exists. By contrast, cryptography scrambles data in such a way that if the third party knew of the data, they would not be able to read it.

The goal of this project was to apply steganography to VoIP communications, taking advantage of the fact that steganography applied to VoIP frames without an active speaker detected does not yet have effective steganalytic countermeasures (Liu et al., 2018). This means that, until effective steganalysis of inactive VoIP frames becomes better established, the application created for this project could be used without much risk of detection.

Materials & Methods

To create the application, a few pre-built tools were assembled and then layered with a proxy server to create the final application.

Linphone, an open-source softphone for conducting VoIP calls, was downloaded on Android as an endpoint of the application. Linphone collected audio from microphones and packetized it to be sent to another device. Two proxy servers written in the Python programming language were used to connect Linphone endpoints. Finally, Linphone requires a registration process with a PBX (private branch exchange) to start making calls. To satisfy this requirement, a PBX called Asterisk was used. In Figure 1, which represents the data flow within

Materials & Methods (cont.)

the system, there are five packet types. Linphone sends "Register" packets as a request to register. "200 Ok" packets are a response from a device that accepts a request. For example, in the figure, "200 Ok" packets are sent from Asterisk to Linphone to confirm that registration was successful. "Invite" packets request to start a call. Finally, "ACK" packets are a means of confirming that a particular packet was received.

To use the application, the proxy was run in the command line on two computers. Once the program was running, Linphone was started on each Android endpoint, and registration occurred automatically. When the proxy started a call, a message (in the form of text) was injected into and extracted from RTP (Real-time Transport Protocol) audio packets by opposite proxies. To minimize degradation to audio quality, covert data was only inserted into audio payloads that carried information representing silence.

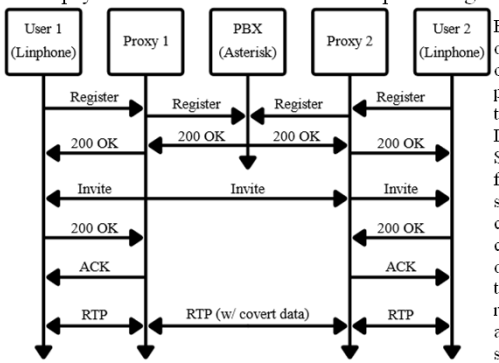

Figure 1: Flow diagram depicting the functionality of the application. All packets except for RTP are types of SIP (Session Initiation Protocol) packets. SIP packets are responsible for PBX registration, starting calls, terminating calls, and managing connectivity between devices. RTP packets, on the other hand, are solely responsible for carrying audio data once a call has started.

Results

```
Sending message:  "Message from first proxy"
Reconstructed message from other side: b'"Message from second proxy
"\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x
ff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\
xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff
\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff'
```

Figure 2: Output of first proxy, capped at 200 packets.

```
Sending message:  "Message from second proxy"
Reconstructed message from other side: b'"Message from first proxy"
\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xf
f\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\x
ff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\
xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff
\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff'
```

Figure 3: Output of second proxy, also capped at 200 packets.

The application was created and transferred hidden information while users stayed silent. Additionally, the application could extract the hidden

Results (cont.)

information and display it to users. As can be seen in Figures 2 and 3, each proxy collected the message sent from the other, as well as extra bytes of RTP audio packets, shown as "\xff", that were not needed to encode the message.

Conclusion

The goal of this project was to develop an application that used a VoIP audio stream to covertly transmit data. This goal was accomplished. However, much more development of this project is possible. A graphical user interface and functionality outside of local area networks (making the application work when users are calling each other from outside of the same Wi-Fi connection) could be added.

Future research could center around the creation of different covert data embedding schemes and minimizing data loss in a real-world environment. In this project, a connectionless protocol was used to transmit RTP audio packets. A connectionless protocol does not guarantee correct packet delivery, and although packet loss was rarely seen in the testing environment, due to strong network connection, it could easily occur if the application were to be used in an area with unreliable network connection. Packet loss would not just cause decreased audio quality but also result in the loss of pieces of covert data that users desired to transmit to each other. Additionally, since modification of VoIP audio payloads is subject to discovery by steganalysis (Lubacz et al., 2008), a more sophisticated audio manipulation scheme that exploits the encoding process could help maintain secrecy. Finally, the application relied on periods of silence in which to transmit covert data, which could be regarded as a weakness, since users must stay silent to send a secret message.

References

Liu, J., Tian, H., Chang, C.-C., Wang, T., Chen, Y., & Cai, Y. (2018). Steganalysis of Inactive Voice-Over-IP Frames Based on Poker Test. *Symmetry*, *10*(336), 1-12. doi:10.3390/sym10080336

Lubacz, J., Mazurczyk, W., & Szczypiorski, K. (2008, December 1). *Hiding Data in VoIP*. Research Gate. https://www.researchgate.net/publication/235059356_Hiding_Data_in_VoIP