Almuatasim Alhabsi

CS427

HW3

**Covert Channel Ping**

**Concept: -**

The reader and the writer both require to have each other’s addresses and or only the sender/writer needs to know the address of the reader. The writer will have their vocabulary as a binary which is going to size of packet or length. The initial idea was to use time as a measurement are either odd or even which can be used for minutes to be safer and avoid detection and avoid accuracy mistakes. For even the binary would be 0 and odd would be 1 if the dictionary is not known for both reader and writer the reader can try both combination of reading and uses the one that builds a correct message. The writer would ping the readers machine based on the message they are trying to send if it is a 0 they will ping at time 17:02 or 17:02:20, if they are sending a 1 they will ping at time 17:03 or 17:02:30 based on which time measurement they are preferring. The implemented approach is to use the length or size of packet as the vocabulary even being

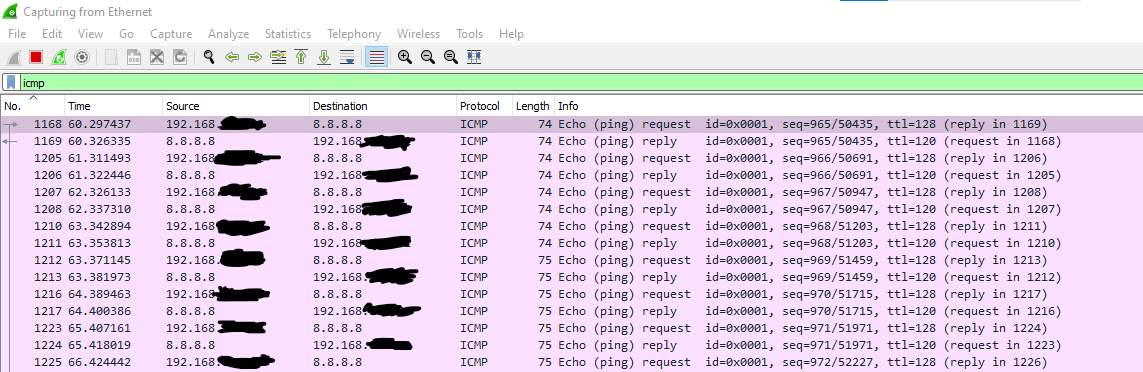
**Bandwidth: -**

The bandwidth can be sending 1 character every 2 minutes, 2 seconds, 2 ms, or 2 ns. It all depends on the accuracy of the time counter of the machines and the code written. The bandwidth of the current version is 4 pings per character which is an average of 10ms per binary value.

**Noise: -**

It is going to be a noisy covert channel since time is a resource everyone has but a log of pings might be the reason of detection. Although icmp protocol is not very common and the use of size of packet is not available information the implemented version is less noisy then the concept.

**Implementation: -**

For testing purposes, the destination machine or the reader is set to Google’s public DNS.

The attached image is an example of the code in this example only the requests are to be counted as binary values. The resulting values are 01 since 4 ping requests are length 74 which is even and translates to 0 and 4 ping requests are length 75 which is odd and translates to 1.

**Modifications and Detection: -**

Modifications that might help with avoiding detections is to setup multiple machines that will ping the same reader so it won’t look suspicious that all pings are coming from one machine although all pings are address to the same machine. Or a chain of machines will ping each other in a waterfall type which starts from one machine which is the writer but only sends 1 ping and a chain of machines to ping in similar fashion till the last write machine which only receives 1 binary from the ping which will have preset messages which will be triggered by either 1 for a message A or 0 for message B. in this approach detection is harder but the information shared is minimal or can only lead to a yes or no answer.

**Time log:**

* initial idea and beginning of research 3:35 pm April 29
* Documentation starts 3:40 pm April 29 ends 11:59pm April 30
* Implementation starts 1:30pm April 30 ends on 4pm April 30

**Resources:**

<https://stackoverflow.com/questions/2953462/pinging-servers-in-python>

<https://www.geeksforgeeks.org/python-convert-string-to-binary/>

<https://www.meridianoutpost.com/resources/articles/command-line/ping.php>