PAPER -1-COVERT COMMUNICATION USING ARP

- At that time to convey a secret message, the Greek ruler used to shave the heads of their slaves, tattoo the secret message on their heads and wait for the hair to grow again on the their heads to conceal the message naturally.
- evident that covert communication is not a new concept and is being used since ages.
- A Network Protocol packet has two parts: The Network Protocol Header and its Payload.

- The Network Protocol Header contains all the control information about the packet whereas the payload contains the actual data being carried by the packet.

- Channel techniques can be classified into two categories which are Storage Network Covert Channels and Timing Network Covert channels [2].

- Storage Network covert channels are a category of covert channels that hide the secret data in the Header and/or Payload part of the Network Protocol Packet.

The contribution made by this work is that

- firstly, instead of hiding secret data directly in the Target Protocol Address field, the proposed technique uses random numbers and ascii code (corresponding to the character to be hidden) to encode and strengthen the imperceptibility of secret data on this channel.

- Further, the proposed technique uses different encoding values for the same occurrences of any character in the covert message which makes the frequency analysis based deductions of characters difficult.

- The proposed technique works on an assumption that the covert sender generates and sends ARP Broadcast Request messages only for covert communications.

Input: A Secret Message M, Seed S
Output: ARP Broadcast Request Packets

1. Input a Seed value S and initialize random number generator with S.
2. Input a Secret Message and store it as M.
3. **While** len(M) != 0:
    a. Fetch the first character from the secret message M and store it as M_ch.
    b. Get the ascii code for character M_ch and store it as Ch_ascii.
    c. Update M by removing the M_ch character fetched in step 3a.
    d. Generate a random number between 128 to 255 using the random number generator initialized in step 1 and store it as R.
    e. Set IP = R – Ch_ascii
    f. Set IP_Addr = 192.168.1.IP
    g. Create an ARP Request packet with the Target Protocol Address Field set to IP_Addr.
    h. Broadcast this ARP Request.
4. Exit.

Fig. 7. Sender Side Algoithm

Input: Local IP Address of Covert Sender, Seed S
Output: Secret message

1. Input the common seed value S.
2. Initialize Covert_msg string to NULL.
3. Initialize random number generator with seed S.
4. Listen for incoming ARP Broadcast Requests having Source Protocol Address field as the local IP address of Covert Sender. Stop listening only when no ARP Broadcast Request packet is received for a time interval of one minute.
    a. **If** an ARP Broadcast Request is received having the Source Protocol Address as that of covert sender,
        i. Fetch the last octet from the IP address received in the Target Protocol Address field of this ARP Broadcast Request packet and store it as IP_Recd.
        ii. Generate a random number between 128 to 255 with the random number generator initialized in step 2. Store this number as R.
        iii. Set temp = R – IP_Recd
        iv. Get the character corresponding to temp in ascii encoding and store it as ch.
        v. Append ch to covert_msg string.
        vi. Goto beginning of step 4.
    b. **Else** If no ARP broadcast Request is received from the covert sender for one minute, then stop listening for more ARP requests and goto step 5.
5. Read the covert_msg string as the complete covert message sent by the sender.
6. Exit.

Fig. 8. Receiver Side Algoithm

Results:

- The covert message sender (Host A) and the covert message receiver (Host B) were connected to a Local Area Network having other actively connected nodes.
- Both Host A and Host B were assigned unique local IP addresses in the LAN. The secret message inputted at the sender side was “This is a covert message.”.
- This message was successfully sent from the sender’s side using ARP Broadcast Request messages. The successful sending of the secret message can be seen in the console snapshot taken at the sender’s site as shown in figure 9.
- Further, figure 11 shows the successful sending of ARP Broadcast packets with Wireshark, running at the sender side

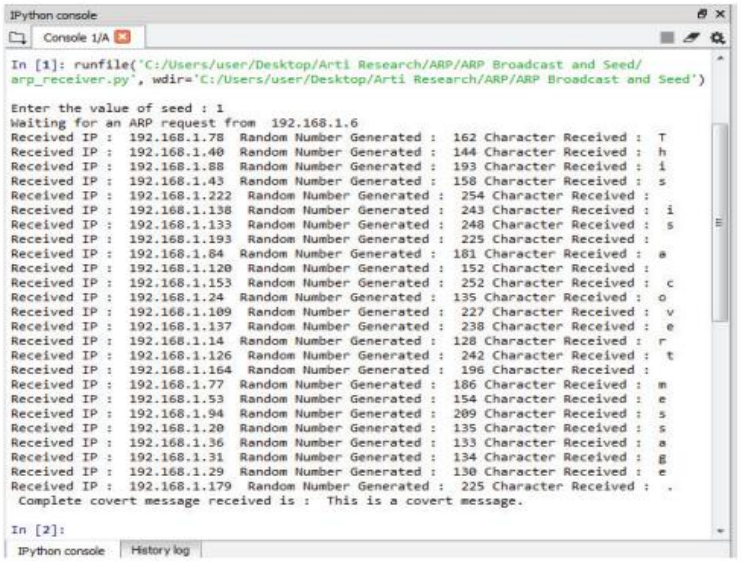

Fig. 10. Receiver’s side Console snapshot

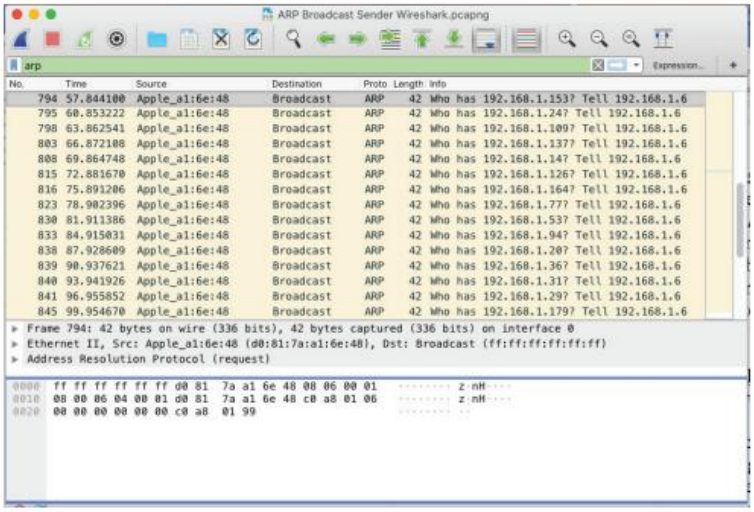

Fig. 11. Wireshark snapshot at Sender Side

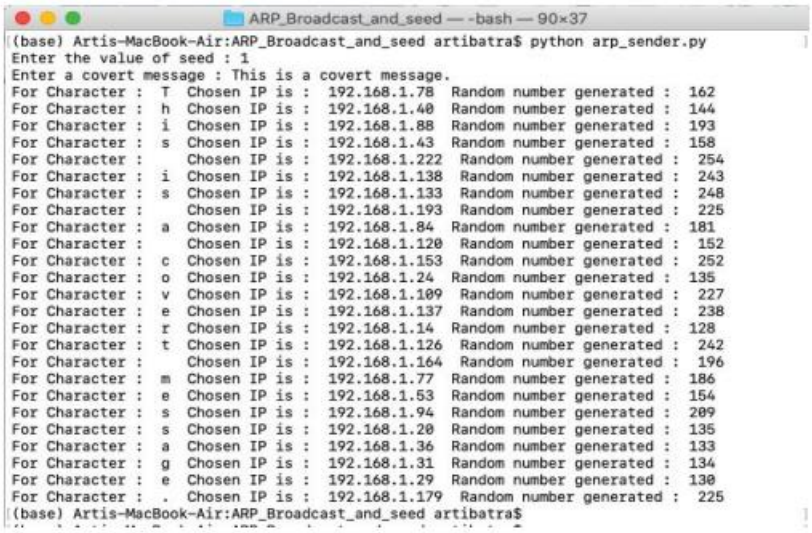

Fig. 9. : Sender's side Console snapshot

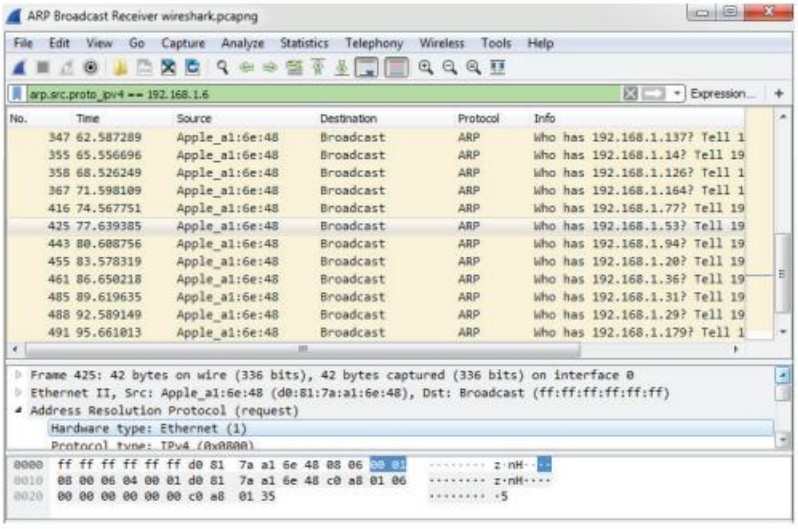

Fig. 12. Wireshark snapshot at Receiver Side

PAPER 2

Packet Length Covert Channel: A Detection Scheme

INTRO:

- A packet length-based covert channel is a network-based covert channel that exploits the network packets' lengths to convey secret information.

- In covert channels literature, three types of covert channel are defined. Storage, timing, and hybrid covert channel. In storage covert channels, a storage location is exploited as a channel to convey secret information such as a sender write a secret message in a shared location and the intended receiver picks that secret message by reading the shared location.

- In timing covert channels, a sender exploits a system rescore aspect to signal a secret message and then, the receiver observes and decodes the secret message. The third type of covert channels is a combination of storage and trimming covert channels.

- It is a hybrid covert channel which poses real challenge as it benefits from the advantages of storage and timing channels. Storage channels provide high bandwidth and timing channels are hard to detect.

Some important factors that paly essential role in developing covert channel techniques have been presented in [19]. These factors are summarized by the following points:

- The rapid development in computer network technologies, communication protocols, cloud computing, virtualization techniques, and data centers. This advanced development represents a rich area to develop different covert channel techniques.

- Switching techniques which give a covert channel the capability to switch its appearance from one filed to another in a given protocol or from one protocol to another. The switching techniques assist in developing a covert channel that hard to detect.
- Internal control protocol technique which provides reliable and trusted communication channel for a covert message. This technique uses a micro protocol to provide reliable communication and dynamic routing to a covert message.

RELATED WORK:

- The concept is known as network covert channel triangle (DSM, the rapid Development of network techniques, Switching techniques, and Micro protocol techniques). This concept is coined based on the three factors that stated above which have the most impact in create, develop, and secure covert communications.

- **In 2013, a packet length covert channel that doesn't require a shared key to be exchanged between the communication parties was developed [35]. This feature reflects the important of this covert channel technique compared with the previous packet length covert channels that are required a shared key\rules. This feature makes this this type of**

covert channel more secure and reduces its overheads. As per the authors claim, the existing detection techniques fail to detect their covert channel. Their covert channel sufficiently imitates normal traffic behavior. Based on our recent survey, to this end, there is no detection method is presented to detect such type of covert channel.

DETECTION SCHEME:

- . Each packet length is exploited to encode one bit of a secret message and that is based on one of two scenarios, either (i) the even number of a packet length encodes 1 and the odd number encodes 0, or (ii) the even number of a packet length encodes 0 and the odd number encodes 1.

- We notice that, mostly the covert messages are meaningful sentences and take the normal sentence format. In contrast, the messages that obtained from normal network packets do not form meaningful sentences as well they are not taking the normal sentence format and mostly they contain special symbols that are not usually used in forming sentences
- noticed that most time, a number of symbols are usually included at normal traffic but were not appeared at a covert traffic. This obtained feature has highly impact on enhancing our detection scheme capability. This finding is approved practically by comparing our classifier results before and after applying this feature.

Dataset:

- The dataset contains two types of network traffics, normal traffic and covert traffic.
- 100 records of covert and normal traffic are used to train and test the scheme classifier.
- 70% is used for training phase and 30% for testing phase. These training and testing phases are repeated 20 times using sampling random validation technique to reflect agreeable and reliable classification results.
- The normal traffic is captured using the Wireshark tool, which is a well-known capturing tool that is used to capture real network traffic
- . A part of this normal traffic is used to construct our covert traffic that is based on network packet length covet channel technique that described above. Python language and Scapy library are used for the purpose of modifying and padding network packet lengths according to the different secret message that are encoded to form our covert traffic.

Classifiers Used:

These classifies are Neural Network, Logistic Regression, Naïve Bayes, Support Vector Machine (SVM) and Random Forest

Results:

- It is clearly noticed that based on our conducted comparison results that is shown in Table 1, the neural network classifier outperforms the other classifier and Naïve Bayes classifier comes next and then Logistic Regression; while SVM and Random Forest are lagged behind.

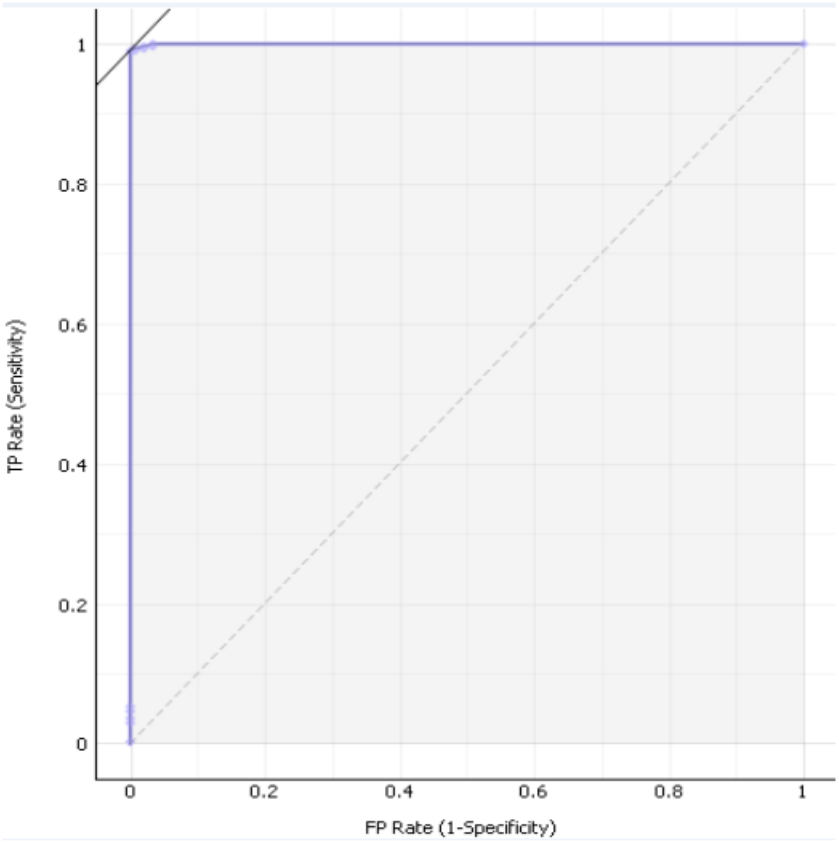

Figure 3. Neural Network ROC curve.

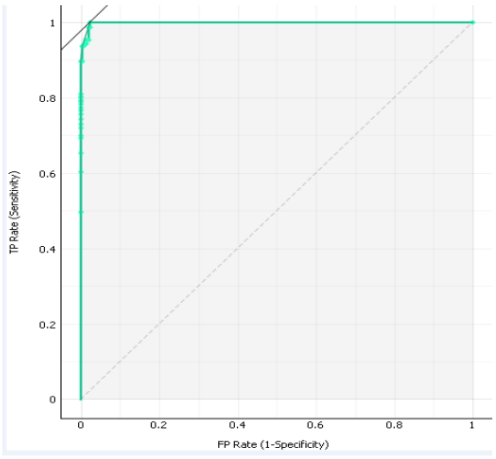

Figure 4. Naïve Bayes ROC curve.

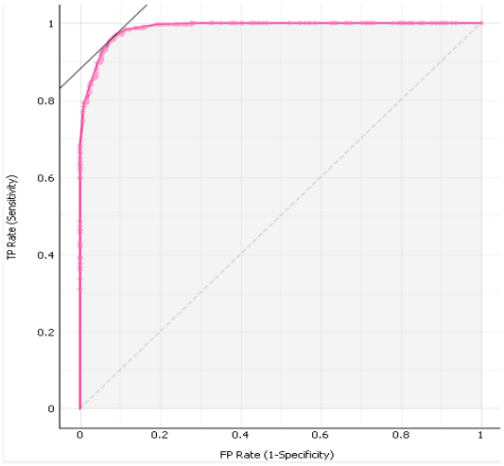

Figure 7. Random Forest ROC curve.

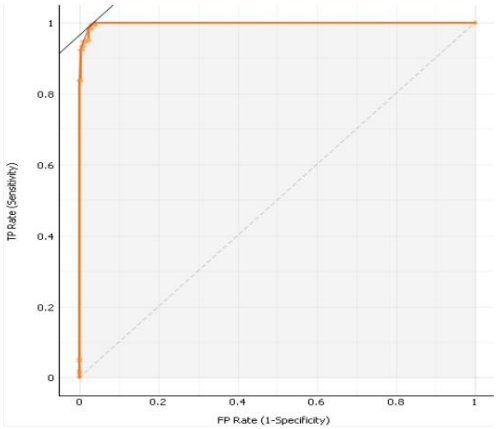

Figure 5. Logistic Regression ROC curve.

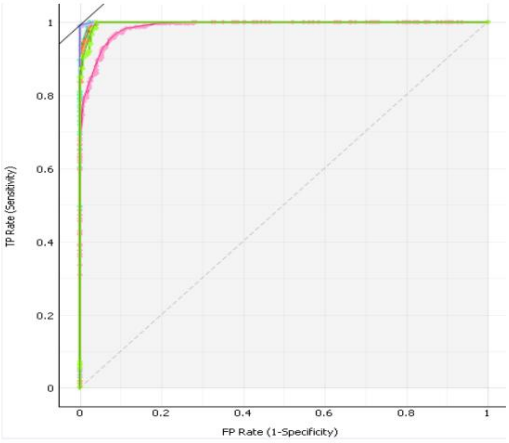

Figure 8. The ROC curves for all classifiers in one diagram.

PAPER-3

CODE AUGMENTATION FOR DETECTING COVERT CHANNELS :

Introduction:

Network covert channels can be created by directly hiding information in header fields, altering the structure of the packet, modulating the temporal evolution of the flow (e.g., throughput or jitter) or by exploiting complex interplays among multiple layers of the stack

- Among the others, methods targeting the Flow Label proven to be suitable for creating capacious and robust covert channels. Intended to help the network in routing operations, the Flow Label is seldom used in real-world deployments thus it can be manipulated for storing secret data [7], [8].

- Unfortunately, literature mainly focused on threats exploiting IPv4, hence leaving the IPv6 counterpart largely unexplored [6], [9]–[11]. Besides, the inspection process is often tightly coupled with the used information hiding method: this makes the detection poorly generalizable and could lead to non-negligible computational burdens (e.g., to check protocol fields via deep packet inspection [12])

- Specifically, the extended Berkeley Packet Filter (eBPF) appears to be suitable for gathering at run-time information on different protocol behaviors to enable the detection of covert communications [14], [15].

- Therefore, in this paper we present a method that uses the eBPF to monitor the usage of the IPv6 Flow Label in a computational-efficient manner.

- Such data can be combined with information provided by other monitoring or security tools deployed in the network (e.g., firewalls and IDS), which are insensitive to covert channels targeting IPv6 [8].

The contribution of this work is twofold.

- First, it proposes a lightweight method for spotting the presence of a covert communication hidden in the Flow Label field, also by taking into account measurements provided by other network security and monitoring tools.

ATTACK MODEL AND REFERENCE SCENARIO

- Concerning the reference scenario, we assume that a warden, i.e., a node able to inspect the traffic containing the covert channel, is present somewhere in the network. For the sake of simplicity, portraits a warden deployed at the end of the covert channel. Typically, the placement of the warden can be arbitrary, except when the attacker uses some form of reversibility for restoring datagrams to their original form [18]

Packet organisation and Inspection:

- **To this aim, we set a hook point in the tc queue management, which enables to run an eBPF program for each IPv6 packet to be processed.**
- **To provide scalability properties and prevent performance degradation in presence of large volumes of traffic, the 20-bit space of possible values is mapped into a bin-based data structure composed of B equally-capable bins.**
- **The mapping is based on the first log2 B bits of the Flow Label, which are used to index the array of bins.**
- **With S we denote the size in bits of each bin: in our case, S = 2 20/B bits. Such a mechanism demonstrated to be lightweight as the traffic processed by our framework experiences only delays of ∼10 ns per packet.**
- **The same utility computes the number N of non-empty bins, i.e., "dirty" bins that have been flagged by the inspection code. As it will be discussed in Section IV, N provides an estimate of the current number of flows. This is only an approximation, because different Flow Label values share the same bin, thus causing collisions.**

DETECTION METHODODLOGY:

- **Our detection methodology is based on the coarse grained estimation N of the number of flows, computed from the statistics gathered in the bin-based data structure. To avoid burdening the notation, we will drop any dependence on time t, except when doubts arise.**
- Since each IPv6 conversation is identified via a fixed, unique Flow Label value generated according to a uniform distribution [19], N can provide a coarse estimate of the number of flows.

- . As a consequence of this “grouping” scheme, the meaningful values of N are those available at the end of each window, i.e., just before values are reset. In general, other values could be discarded or stored to compute suitable models or datasets to be processed with machine-learning-based frameworks.

- The presence of an anomalous behavior in the usage of the Flow Label can be then detected by observing deviations of N from an expected value, for instance by feeding a model based on past observations, or by comparison with network analytics reports provided by a complementary tool

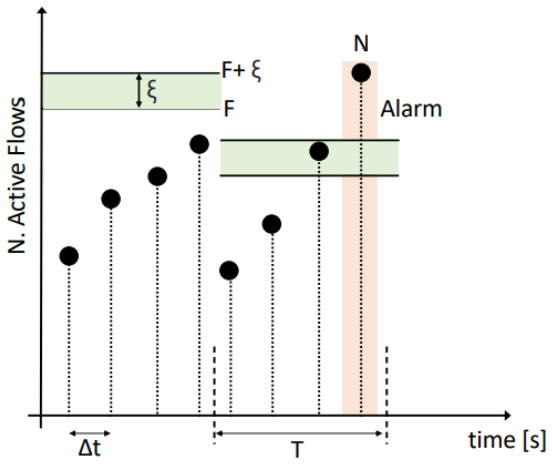

Fig. 3. Detection strategy based on the bin-based `Flow Label` partitioning.

- As shown, the basic idea is that if the number of flagged bins (e.g., those for which the corresponding Flow Label values have been observed in the considered window) is greater than the number of observed active IPv6 flows, a covert communication could be present and an alarm is raised.
- To adjust the responsiveness of the approach (e.g., to limit false positives), with ξ we denote a guard threshold that has to be exceeded

DISCUSSION AND LIMITS:

- As shown, code augmentation is suitable for developing simple filters able to inspect network traffic and detect covert channels.
- In general, adding a filter requires to write small portions of code and multiple filters can be deployed to concurrently gather data from different portions of the traffic, which is vital when the target of the covert communication is unknown.
- This paradigm can also help to “enrich” information provided by an IDS, firewalls and network monitoring tools insensitive to information-hiding-capable attacks or not ready to fully handle IPv6 security.

- Unfortunately, there is not a onefits-all solution and each filter-detection pair requires tuning. For instance, the Flow Label case requires to tune the bin-based structure for having a suitable matching between the observed traffic and the detection strategy.
- Therefore, the proposed mechanism should be properly engineered according to the scenario that has to be protected.
- As an example, the average traffic volume and number of hosts should be known to select the granularity of bins (i.e., B and S) and avoid under-/overestimations. Similar considerations are valid for the parameters α and ξ.