Chapter 4

Our Proposed Approaches

This chapter focuses on the approaches one by one implemented in our research. The general notations used in our chapter are given below in Table 4.1:

4.1 Single Attack Detection

Work in terms of single attack detection with the 4 attacks that are probing, DoS, vampire and U2R in MANET environment is discussed here. Two tables are created by the detection engine, one that possesses a normal profile classified as TCP, UDP, AODV permissible formats and the abnormal table consisting of abnormality in characteristics of the various types of attacks. Characteristics generated by abnormal table are matched with the abnormality of attacks that helps us to identify the type of attack. In the simulation engine NS2 packet headers of TCP,UDP are fixed by the engine and packet if has a missing header is classified as a hampered packet. Our proposed algorithm is discussed in subsection 4.1.1.

4.1.1 Proposed Algorithm

Our algorithm is defined in three sections viz. input, procedure and output shown in algorithm 1.

Data is testes considering fifty nodes within a range of 800 × 800 m. I can be defined as a group of addresses in transitional nodes in the path of sender and receiver. This is contrasted with R (receiver node’s address) till get a similarity. As a match is detected, rpkt, that is a routing packet, is received by I and moved ahead till it arrives at the final node. As network related broadcasting is capable with both Transport, Communication and Application layers, TCP/UDP and AODV protocols are used. Attack identification is accomplished by behavioral analysis; as data goes through simulation engine, if data==b_a(n) (normal table) implies there are no traces of abnormal data is assumed that no attack has taken place. If data == b_a(ab) (abnormal table) it means attack of the following types has occurred:

1. In case of data getting captured it is identified as probe attack.

2. In the case of transmission of junk messages that do not match TCP,UDP protocol measures attack is DoS.
Algorithm 1 Single Attack Detection Algorithm

1: **Input Factors:**
2: M: Mobile node, S: Source node, R: Receiver node, \( \Psi \): Radio range = 550m, \( b_h(n,ab) \): behavior table containing normal and abnormal behaviors, I: Set of intermediate nodes, A: Attack types (probing, vampire, DoS, U2R), \( j_m \): Junk message, e: Energy of nodes, \( R_p \): AODV routing protocol, rpkt: Routing Packet
3: **Output Responses:** Throughput, Normal Routing Load (NRL), End-to-End Delay, Accuracy, Confusion Matrix i.e. True positives\( (t_p) \), True negatives\( (t_n) \), False positives\( (f_p) \), False negatives\( (f_n) \).
4: **Procedure:**
5: S ← broadcast (AODV, S, R)
6: if I \( \neq R \) and I in \( \Psi \) then
7: I ← receive rpkt
8: I ← forward rpkt to next hop
9: else if I == R then
10: R ← receives rpkt
11: Select shortest path
12: R ← create reverse route for ack
13: S ← receives ack
14: Send data (S, R, data)
15: else
16: Node out of range
17: Node unreachable
18: end if
19: **Attack detection module:**
20: Data passes into detection engine, Compare data and \( b_h(n,ab) \).
21: if data == \( b_h(n) \) then
22: data is normal TCP, UDP or AODV
23: else if data == \( b_h(ab) \) then
24: Data shows abnormal activity: A
25: if \( b_h(ab) \) == data capturing then
26: A ← probe
27: else if \( b_h(ab) \) == \( j_m \) and data \( \neq \) TCP, UDP, AODV then
28: A ← DoS
29: else if \( b_h(ab) \) == \( (e=0) \) and path disabled then
30: A ← vampire
31: else if \( b_h(ab) \) == ip modified or I node uses root ip then
32: A ← U2R
33: end if
34: end if
35: end if
3. In case of energy consumption if abnormal, path disability issues - the attack is vampire.

4. In case of IP tampering and modification, the attack type is U2R.

Figure 4.1 depicts flowchart for our proposed algorithm. Counterattack measures are started, once the attacks are detected. Counterattack measures are furnished by successful identification of the behavior of our studied attacks. This is accomplished by removing suspicious nodes and enabling secure data transfer among the dispatcher and receiver.

Table 4.2 depicts behavioral table, where S: send, R: Receives, F: Forward, D: Drop, N: Normal, H: High. In the case of no attack scenario, entire protocol values (TCP,UDP,AODV) are fulfilled and generate a positive acknowledgment at the receiving end and the queue usage is normal. In the attack scenarios as per behavioral analysis, packets tend to drop, no acknowledgment is generated and queue usage is more which show traces of faulty nodes.

### 4.2 Single Attack Prevention

This presents a prevention scheme for MANETs using a neighbor trust-based method which will help to identify our studied attacks. Our proposed security scheme verifies the neighboring node in case of faulty activity, that is if node varies its defined behavior. The objective of our scheme is routing, that is created to store a table of route information among nodes.

Our proposed scheme is depicted in Figures 4.2, 4.3, 4.4. Addresses of all intermediary nodes from the sending node and receiving node are recorded. They are matched with receiver’s addresses. Data is passed from source to destination nodes with the help of AODV protocol. Preventer (watcher) node is specially designed capable of capturing the defined path, by tracking their movement, blocking the nodes which generate faulty activities in a given range of network. Our designed preventer node operates in two stages: suspicion and blocking[45]. The criteria of watcher node(s) is trust value must be 1 that is computed at each instant of time and those node(s) that can be trusted are designated as watcher node(s). A node is classified as a suspicious node if it is detected as abnormal and its functionality does not match with the normal behavior that is tracked by preventer node. the preventer node confirms its maliciousness and blocks it in the next step. If the attacker is identified as a U2R one, its conditions depict that the intermediate nodes(s) update their self identification (ID) as receiver ID. Data drops by LOOP condition (node forwards itself to next node thus there is no change is incoming and outgoing node) in this attack and in this the preventer node identifies all this and blocks the node(s). In the case of vampire attack, symptoms are identified firstly of more energy consumption in the network by node(s) than its threshold energy. When extra threads are run that vary from those in data forwarding, higher energy is consumed than all nodes cumulatively, classify it as a vampire attack. In the case of probe
attacker, data capturing is identified by the preventer node in which routing module behaves correctly but drops data later in the network. Hence it is classified and blocked as an attacker node. In a DoS attack, preventer node checks if junk packets are overloaded in the system and deprive legitimate requests to be fulfilled. Also headers of Transmission Control Protocol (TCP)/User Datagram Protocol (UDP) are matched with all packets passed and computes data rate. For higher data detection than normal rate, preventer marks it faulty and blocks it.

Our algorithm is defined in three sections viz. input, procedure and output shown in algorithm 2,3.

**Algorithm 2 Single Attack Prevention Algorithm**

1: **Input Factors:** M: set of nodes, S: sender node ∈ M, R: receiver node ∈ M, W: watcher node ∈ M, I: intermediate node ∈ M, Iₙ: set of intermediate nodes from sender to receiver, E: energy of node, C: consume energy, eₜₚ: energy threshold, eₜₚ: (Sum of C energy of path)/2, AODV: routing protocol, Rr: 550 radio range

2: **Output Responses:** Send, receives, PDF, NRL, Average e-e delay (ms), Throughput

3: **Procedure:** Broadcast (AODV, S, R)

4: if I in Rr and I receives AODV and forward != true then

5: W ← watch neighbor activity

6: if I update the id as R then

7: I as suspicious node

8: end if

9: W execute (prevention module)

10: elseif forward == true and R found

11: R reply to S;

12: Send (data, S, R)

13: end if

14: Send (data, S, R)

15: if path established then

16: S ← generate TCP or UDP;

17: Send data by connected path;

18: Calculate C of path

19: if C > eₜₚ then

20: Identify C energy of each node whose participate in path

21: if C of Iₙ >= sum of C Iₙ₋₁ node whose e path then

22: Iₙ as suspicious node;

23: W execute (prevention module)

24: end if

25: end if

26: else

27: Route search

28: end if
Algorithm 3 Single Attack Prevention Algorithm: Prevention module

1: Prevention (time, I, behavior)
2: W watch the activity of I node whose in R
3: if I update the id as R id then then
4: I receive data and not forward to next hop or receiver;
5: I not congested and outgoing link is idle;
6: Analysis receive data whose drop by I
7: if data drop by LOOP then then
8: I set as U2R attack;
9: Call (block the node, record time, broadcast)
10: end if
11: if C of In >= sum of Ci node whose e path then then
12: W check the In activity
13: if In execute extra threads whose not for communication then then
14: I set as Vampire attack;
15: Call (block the node, record time, broadcast)
16: end if
17: end if
18: if I participate in route and true route info gives then then
19: S established route by I node to R;
20: Send (data, S, R)
21: if data == drop/capture || forward!= true and reason == null then] then
22: I set as probe attack;
23: Call (block the node, record time, broadcast)
24: end if
25: end if
26: if I generate data! = TCP/UDP and packet type != network packet type then
27: I is suspicious;
28: W (store I data, data rate, number of receivers, time)
29: if data == unknown and data rate > normal data rate and receiver in time n + 1
> receiver in time n then
30: I set as DoS attack;
31: Call (block the node, record time, broadcast)
32: end if
33: end if
34: end if
4.3 Multi Attack Detection and Prevention

This section focuses on a multi-attack scenario in which our studied attacks occur in the network simultaneously. Our proposed scheme named Distributed trust based prevention executes on multiple neighbor nodes, who trace the activity and act upon it in terms of classification as malicious and non-malicious. It is divided in 3 stages viz. (1.) route discovery state, (2.) steady state and (3.) execution state. In the first stage route discovery state, preventer nodes track the nodes in the neighborhood and assign them a trust value which helps to compute shortest path from source to destination. Steady state then analyzes the behavior of path as per active nodes. This state communicates non-faulty data and computes trust value for nodes. Trust value generated judges the trust factor of node(s) to be blocked or re-route their path. Finally, execution state computes the final packet delivery time in the network and works till the end of simulation. Figure 4.5 depicts the block diagram and data flow in our scheme.

The multi-attack implementation is also publicly available (see https://github.com/gurveenv/multiattackprevention).

4.3.1 Proposed Methodology

In order to counter malicious attacks (like DoS, probe, vampire, U2R) in a multi-attack environment (a combination of one or more attacks simultaneously) propose a trust based scheme. This is done through watcher nodes [46], which determine trust of other nodes, and a three-step mechanism, i.e., route discovery state, steady state, and execution state.

The preventer nodes contain the normal and abnormal behavior table for identification of node behavior [37]. The normal table stores the information of all possible combinations of protocol header formats permissible by the simulation engine. The abnormal table contains possible attack symptoms of the known attacks. Probing scans the device, DoS spreads junk messages, vampire consumes the device’s energy and U2R modifies the logical address and inserts a new unmatched header into the abnormal table. Data packets that match the headers(TCP/UDP/AODV) as defined by the simulation engine are classified as normal, whereas the data that does not match is treated as abnormal [37]. If any one of the 3 states (i.e., route discovery, steady state and execution state) detects malicious attacks, then multiple watcher nodes block the untrustworthy nodes.

4.3.1.1 Proposed Algorithm

In this subsection, the distributed trust-based multi-attack prevention algorithm is discussed. The complexity of the proposed algorithm is $O(n^2)$ as it is executed in multiple loops which makes it accurate in precision.

1. Route Discovery State: The process begins with the route discovery state in which
preventer nodes watch the neighboring nodes and assign them a trust value of 1 that assists in searching the shortest path from source to receiver for communication. The trust value depends on packet delivery ratio (PDR) of each node and gets stored in the behavior table. The energy of nodes ($E_n$) is compared with the threshold ($E_{th}$) to ensure there is no energy depletion due to attacks. $E_{th}$ is set to 10 joules as below this, nodes get regenerated and enter the dead condition. The route discovery state also gives information about the number of hops between the communicator and its capabilities such as processing power, received signal strength, node energy, and queue capacity. Figure 4.6 shows the route discovery state and algorithm 4 provides the algorithmic description.

2. Steady State: After the route discovery state, the steady state is activated by the preventer (watcher) nodes. The steady state in its first 5 seconds analyzes active nodes or path behaviors, transmit actual data and calculates the node trust value based on the number of data forwarded, route table details, energy utilization, processing power and memory utilization. If it is found that any parameter in the active path is abnormal, then all the watcher nodes send message to each other and calculate the average trust value as well as node behavior. If the behavior is abnormal and average trust value is less than 0.6 (as data cannot be regenerated by the receiver node if value is less than 0.6), then the node is blocked and local route repair method is called for re-establishment of a path. Otherwise, the established path is treated as a trusted path and data is sent for future communication. Figure 4.7 depicts steady-state and algorithm 4 provides the description.

3. Execution State: The last phase is the execution state that is run till the end of simulation or while the trust value is greater than 0.6. In this state, the trust value of active nodes is calculated at packet delivery time. $E_n$ is compared with $E_{th}$ to ensure there is no energy depletion due to attacks. If it is found that trust value is less than 0.6 in consecutive sequences, then the preventer nodes watch the behavioral activity of individual node and re-establish the path for communication. Figure 4.8 shows the execution state and algorithm 5 provides the description.

Our implementation is publicly available (see https://github.com/gurveenv/multiattackprevention_ises).

4.4 Multi Attack Detection and Prevention using Forensics and Neural Network

Network forensics refers to the domain in which apply forensic science to find out the cause of network crime. Its aim is to trace any faulty and illegitimate activity in the logs of traffic,
Algorithm 4 Multi Attack Detection and Prevention: Route discovery state and steady state

Initialization:
M: mobile nodes, S: source node ∈ M, R: receiver node ∈ M
I: mediator nodes ∈ M, Si: source Identification number
Ri: receiver identification number
Ii: intermediate identification number
r: AODV packet, bh: normal, abnormal,
ti: initial trust 1, th: trust value threshold (0.6), t: trust value
En: energy of node, Eth: energy threshold (10 joules)
P: preventer nodes, abnormal (DoS, probe, vampire, U2R)

Output: t, E, Path
Route discovery(S, R, Path)
S generate route packet (r, sequence no, Si, Ri)
S broadcast route packet to search R node
if I in range and Ii ≠ Ri and E ≥ Eth then
Assign t to I node
Forward r to next hop
else if Ii == Ri and E ≥ Eth and path ≥ 1 then
Select shortest path
Send acknowledgement to S
Call Steady state();
end if
Steady state(Ii, Pi)
Examine nodes 0 to 5 seconds
Activate Pi
while active path found do
All Pi watch activity of neighbor I node
Pi calculate trust value
t = ti + (forward/receive)
if t < th then
Send report to bh module
Compare header with bh(normal, abnormal)
if header is abnormal then
Calculate average t over all Pi
t = t1 + t2 + ... + tn/number of Pi nodes
if t < th then
Send block message to I node
Call local route repair module
Re-establish path
else
I is normal behavior
Call execution state()
end if
I is normal behavior
Call execution state()
end if
Update trust t
I is normal behavior
Call execution state()
end if
end while
Algorithm 5 Multi Attack Detection and Prevention: Execution state

Initialization:

- $E_n$: energy of node
- $E_{n,old}$: energy in previous time interval
- $E_{n,new}$: updated energy in current time
- $E_{th}$: energy threshold (10 joules)
- $t_j$: trust value in current time
- $t_{j,old}$: trust value in previous time interval

Output: Data sent, Data received, Energy consumed, Accuracy, False positive rate, NRL.

Execution state ($S_{id}$, $R_{id}$, $I_{id}$, $P_i$)

Examine nodes for trusted route.

$P_i$ active mode

Calculate $t_j$, $E_n$ in every packet base

if $E_n \geq E_{th}$ and $t_j \geq t_{th}$ then

Send data by selected path

$P_i$ watch I node

$E_{n,new} = E_{n,old} - E_n$ per packet

$t_j = t_{j,old} \pm$ (forward/receive)

if $P_i$ finds $t_j < t_{th}$ then

Steady state ();

else if $P_i$ finds $t_j > t_{th}$ and $E_{n,new} < E_{th}$ then

Local route repair ();

else

Trust route found from $S_{id}$ to $R_{id}$

end if

end if

Execute re-route for alternate path

end if
Our Proposed Approaches

completely detect their details, hence to identify damage done [47]. Forensic ways mostly are useful in collecting information of any system after going through an illegitimate attack (probe, vampire, U2R or DoS), or both, among other consequences. This can be helpful in order to gain legal evidence, but the loopholes can be found in the network by this and can be fixed.

In our proposed method two stages of forensic methods are applied, in the first level examine the network data through a lightweight forensic method, such as partial symptoms engine and filter all the unwanted data from network as depicted in Figure 4.11 through a partial forensics engine (PFE) as shown in Figure 4.9. The limitation of PFE is that it does not perform a detailed analysis of all parameters which is why the need of forensics deep learning engine (FDLE) arose as shown in Figure 4.10. After PFE, the unwanted data goes to FDLE and is critically analyzed by forensic system. PFE only analyzes the node’s behavior (N) as depicted in 4.4 wherein standard AODV new trace format is considered and parameters are taken from there. It is checked if the node has entered LOOP condition or buffer is not full etc. (IFQ). However, in FDLE as shown in Table 4.3, on getting a discrepancy in partial symptoms (by variable \( u_n \)) analyze each layer of the network in terms of MAC (Ma etc), Packet information (Pf etc), IP level information (It etc.) hence gives a detailed analysis of where there are symptoms of an attack. Forensic based attack detection is a critical aspect to inspect the network log files, as the header format of network protocols differ from each other (vary from protocol to protocol). But the forensic approach improves the accuracy of attack detection as per Table 4.3 and 4.4.

1. In the first row of table, while the attack type is U2R then abnormality exists in packet header in this case event (d) means data drops, network trace level is route error, and reason of drop is looping in the node. Packet header also contains the source ID that is similar to destination ID and number of forwarding of similar packet multiple times.

2. In the second row, the attack type DoS, the event type receives (r) or drop (d), network trace level queue is full, packet type is infected and MAC information contains all zeros such as duration, source and destination address as well as ethernet type.

3. In the third row, at the time of probe attack event type send (s) exists within the packet but there is no reason of drop, packet type is probe and the size of packet is variable. Similarly all DoS MAC information is all zeros such as duration, source and destination address as well as ethernet type.

4. In the last row, vampire attack exists in network while network resources are consumed due to unnecessary reasons so that packet header contain such information event is receives (r), drop (d) and send (s), network trace level is route error and reason of drop is LOOP. Packet header also contains the source ID is similar to destination ID and number of forwarding of similar packet in multiple times.
However, in Table 4.4 analyzes only symptoms superficially by studying some parameters such as network trace levels, node IDs etc in PFE.

In the proposed approach, attacks are detected by PFE and prevented using FDLE (neural network training based system). In the proposed intrusion prevention method apply the perceptron training method of neural networks to detect probe, vampire, U2R and DoS attack as depicted in Figure 4.12. Here sample data is taken as TCP, UDP, AODV, MAC etc. and train the network for normal behavior and after training fetch the actual data and detect the intrusive process by trained network. In this preceptron neural network take the single layer feed forward network and also bias (b) inbuilt. The known attacks are divided into four sections based on their behavior and protect the MANET from these attacks. In the prevention scheme consider all four attacks that exist in the network and spread unwanted messages in the network but they are prevented by neural network based approach. In this approach, all the network traffic passes to the neural network system and the trained network in its very first five seconds analyzes all incoming network traffic and compares it with neural network system. While the unwanted data is found by the system during communication then block the attacker nodes by message passing method (this is a method in which critical analyzer sends acknowledgment about a node for its maliciousness so it is blocked in future for data communication) and prevents the MANET. In the prevention time forensic approach associated with the neural network system for improving the accuracy of prevention in short period of time. Forensic system generates the attack log file that helps to identify the known attack and easily filter the unwanted data from network traffic and get the attacker node information in efficient manner. This log file acts as an input to PFE which checks superficially the behavior of attacks. PFE is unable to identify packets/nodes which are faulty hence transfers these malicious detection’s to FDLE to perform a complete analysis on the confirmation of attacks at all layers of the network.

### 4.4.1 Proposed Algorithm for Forensic based approach

This describes the forensic based attack detection and neural network based prevention scheme, Algorithm 6,7 depicts detection and prevention of attacks in MANET.

In this algorithm forensic system detect the multi attacks in MANET scenario, where consider four types of attack such as U2R, DoS, probe and vampire attack involved in network traffic. First network traffic is passed to the lightweight detection engine named as PFE and filter the suspicious data. Those suspicious data are sent to critical analyzer in FDLE wherein it checks every field value of suspicious data for identifying the attack behavior and its effect on analysis after that synchronizes attack log for future attack detection.

In the proposed algorithm 1, data is tested using 50,100 and 150 nodes in a range of 1000 \(\times\) 1000 m.
**Algorithm 6** Forensic based Multi-Attack Detection

1: **Input:**
2: PFE: Partial forensics engine, FDLE: Forensics deep learning engine, $A_l$: log file,
3: **Output:** Forensic Accuracy, percentage of attack, percentage of partial and complete symptoms.
4: **Procedure:**
5: M generate the network traffic ($N_t$)
6: if $N_t$ pass to PFE then
7: if $N_t$(event, $N_w$, $P_t$) $== A_l$(event, $N_w$, $P_t$) then
8: Pass to FDLE, $N_t$ as $p_r$,
9: $N_t$ data as $u_n$
10: Pass $u_n$ to $c_l$
11: $c_l$ compare each field of $u_n$ by $A_l$
12: if $A_l$ $==$ (Atk-type) then
13: $c_l$ confirm $u_n$ as attack packet
14: else
15: $A_l$ $\leftarrow$ $A_l$ + $u_n$
16: Updated $A_l$ synchronize for future attack detection
17: **end if**
18: else
19: $N_t$ is normal
20: Data send to next hop or receiver
21: **end if**
22: **end if**
4.4.2 FDLE based Prevention Algorithm

In this algorithm the ANN based multi-attack prevention system is designed, which filters the abnormal behavior and identifies the attacker node(s). Algorithm is divided into two phases such as training and testing, during the training time sample network traffic is passed to training system and completes the training (such as abnormal/normal profile separation, attack node criteria decision etc.). In testing case network traffic is passed to FDLE and identifies the attacker node and blocks the attacker node from the network. The training phase starts when network traffic takes as input where the initial output is set as 0 and desired output as normal or abnormal profile. While the desired output is not equal to the actual output and error is greater than threshold value and also training time has not ended then the network is trained using defined function and calculates the error. While error is higher then update the weight value till the network is not trained or output response equals to the desired output. At the testing time network traffic generates by mobile node. If network traffic receives data by the FDLE then analyze data and identify symptoms. While symptoms are abnormal then classify the attack class of symptoms and retrieve the node id of attacker. Finally broadcast the attacker id for blocking the particular node from the network.

4.4.3 Flow Chart of Algorithm

Forensic methods use log files and detect attacks through symptoms match based methodology. In Figure 4.13 we separate the normal and abnormal data in an accurate manner.

Figure 4.14 depicts training of a network for separation the normal and abnormal data in accurate way, for the network training sample input data pass to artificial neural network and classified the attacker node with normal node.

Figure 4.15 tests the network traffic through FDLE, where mobile node generate the network traffic and pass to the artificial neural network. ANN compares the received data from trained data set and identifies the abnormality as well as attacker nodes. That attacker information is spread over the network, so that in future the entire normal node do not communicate with attacker node.
Algorithm 7 Neural Network for Multi-Attack Prevention

1: **Input:**
   2: $N_n$: neural network, $t_h$: threshold: 0.01, $e$: error, $d$: desired o/p (1), $x_i$: input (count value of $S, r, f, d$), $b$: bias (if needed offset), $w_i$: weight (0 to 1), $y$: output as per value of $x$, $m$: mobile node, $N_i$: network traffic, $N_p$: normal profile, $A_b$: abnormal profile, $A_{t-id}$: Attacker node, Atk-type: U2R, DoS, probe, vampire, Simulation Time: $\mu$, Training time: $\alpha$, Testing time: $\Psi = (\mu - \alpha)$
3: **Output:** PDR, NRL, Delay and Energy utilization.
4: **Procedure:**
5: **Training Phase:**
6: Start $\alpha = 0$
7: $x_i = N_i$ (binary format)
8: $y = 0$
9: $d = (N_p, A_b)$
10: if $d \neq y$ and $e \geq t_h$ and $\alpha \neq end$ then
11: $y = \sum_{i=1}^{n} x_i \cdot w_i + b$
12: $e = (d - y)$
13: $w_i$: $\Delta w_i$ (update weight)
14: $\alpha = \alpha + 1$
15: end if
16: $d \leftarrow$ (Normal/ Abnormal log file create)
17: **Testing Phase:**
18: Start $\Psi = 0$
19: $N_i$ generate by m
20: if $N_i$ receives by $N_n$ then
21: Analyze $N_i$
22: if symptoms of $N_i = A_b$ then
23: $N_i$ is Atk-type
24: Retrieve $A_{t-id}$
25: Block $A_{t-id}$
26: Broadcast $A_{t-id}$ for elimination from network
27: else
28: $N_i$ is $N_p$ behaviour
29: end if
30: end if
### Table 4.1: General Notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Set of Mobile nodes</td>
</tr>
<tr>
<td>S</td>
<td>Source node</td>
</tr>
<tr>
<td>R</td>
<td>Receiver node</td>
</tr>
<tr>
<td>$\psi, R_r$</td>
<td>Radio range = 550 m</td>
</tr>
<tr>
<td>$b_{(n,ab)}$</td>
<td>behavior table containing normal and abnormal behaviors</td>
</tr>
<tr>
<td>I</td>
<td>Set of intermediate nodes</td>
</tr>
<tr>
<td>I_n</td>
<td>set of intermediate nodes from sender to receiver</td>
</tr>
<tr>
<td>A,Atk-type</td>
<td>Attack types (probing, vampire, DoS, U2R)</td>
</tr>
<tr>
<td>j_m</td>
<td>Junk message</td>
</tr>
<tr>
<td>e,E</td>
<td>Energy of nodes</td>
</tr>
<tr>
<td>R_p</td>
<td>AODV routing protocol</td>
</tr>
<tr>
<td>rpkt</td>
<td>Routing Packet</td>
</tr>
<tr>
<td>W</td>
<td>Watcher node $\in$ M</td>
</tr>
<tr>
<td>C</td>
<td>Energy consumed</td>
</tr>
<tr>
<td>e_{th}</td>
<td>Energy threshold</td>
</tr>
<tr>
<td>AODV</td>
<td>Routing protocol</td>
</tr>
<tr>
<td>S_{id}</td>
<td>Source Identification number</td>
</tr>
<tr>
<td>R_{id}</td>
<td>Receiver identification number</td>
</tr>
<tr>
<td>I_{id}</td>
<td>Intermediate identification number</td>
</tr>
<tr>
<td>r_p and f</td>
<td>AODV packet and Forward packet</td>
</tr>
<tr>
<td>t_i</td>
<td>Initial trust 1</td>
</tr>
<tr>
<td>t_{th}</td>
<td>Trust value threshold (0.6)</td>
</tr>
<tr>
<td>t_j</td>
<td>Trust value in current time</td>
</tr>
<tr>
<td>E_{th}</td>
<td>Energy threshold (10 joules)</td>
</tr>
<tr>
<td>P_i</td>
<td>Preventer nodes</td>
</tr>
<tr>
<td>E_{old}</td>
<td>Energy in previous time interval</td>
</tr>
<tr>
<td>E_{new}</td>
<td>Updated energy in current time</td>
</tr>
<tr>
<td>t_{old}</td>
<td>Trust value in previous time interval</td>
</tr>
<tr>
<td>FS</td>
<td>Forensic system</td>
</tr>
<tr>
<td>A_{l}</td>
<td>Log file</td>
</tr>
<tr>
<td>u_n</td>
<td>Suspicious data</td>
</tr>
<tr>
<td>p_r</td>
<td>Partial symptoms</td>
</tr>
<tr>
<td>c_{l}</td>
<td>Critical analyzer</td>
</tr>
<tr>
<td>N_{t}</td>
<td>Network traffic</td>
</tr>
<tr>
<td>N_{n}</td>
<td>Reason of drop</td>
</tr>
<tr>
<td>P_{t}</td>
<td>Packet type</td>
</tr>
<tr>
<td>N_{p}</td>
<td>Neural network</td>
</tr>
<tr>
<td>d</td>
<td>Desired output (1)</td>
</tr>
<tr>
<td>x_i</td>
<td>Input(count value of S,R,f,d)</td>
</tr>
<tr>
<td>b</td>
<td>bias (if needed)</td>
</tr>
<tr>
<td>w_{i}</td>
<td>weight (0 to 1)</td>
</tr>
<tr>
<td>y</td>
<td>output (as computed per formula)</td>
</tr>
<tr>
<td>N_{p}</td>
<td>Normal profile</td>
</tr>
<tr>
<td>A_{b}</td>
<td>Abnormal profile</td>
</tr>
<tr>
<td>A_{t-id}</td>
<td>Attacker node</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Simulation Time</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Training time</td>
</tr>
<tr>
<td>event</td>
<td>node (S,R,d,f)</td>
</tr>
</tbody>
</table>
Yes
Start counterattack measures
No intrusion has occurred

Start
Is data normal?

No intrusion has occurred

Attack is probe.

Attack is DoS.

Start counterattack measures

bh(ab) == data capturing.

bh(ab) == data !=TPC, UDP.

bh(ab) == (e=0) and path disabled.

bh(ab) == ip modified/
I node uses root ip.

Attack is U2R.

Attack is vampire.

bh(ab) == Jm and data !=TPC, UDP.

Figure 4.1: Flowchart for Single Attack Detection
### Table 4.2: Behavioral Table

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Normal</th>
<th>Abnormal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Vampire</td>
</tr>
<tr>
<td>Packet Type</td>
<td>TCP, UDP, AODV</td>
<td>Energy = 0</td>
</tr>
<tr>
<td>Event Type</td>
<td>S, R, f</td>
<td>d</td>
</tr>
<tr>
<td>Acknowledgement</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Queue Utilization</td>
<td>N</td>
<td>H</td>
</tr>
</tbody>
</table>
Figure 4.2: Flowchart for Single Attack Prevention: Step 1
Figure 4.3: Flowchart for Single Attack Prevention: Step 2

- S established route by I node to R
  - Send (data, S, R)
  - Is there a data drop or data capture and node not forwarded?
    - Yes: I set as Probe attack Call (block the node, record time, broadcast)
    - No: Stop
  - No: Does I not match TCP/UDP headers and packet is not network packet type?
    - Yes: I under suspicious W (store I data, data rate, number of receivers, time)
    - No: Is data unknown and data rate > normal data rate and receiving time > standard time pattern?
      - Yes: I set as DoS attack Call (block the node, record time, broadcast)
Figure 4.4: Flowchart for Single Attack Prevention: Step 3
Chapter 4
Our Proposed Approaches

Figure 4.5: Data flow in Multi-attack Prevention

- Assign initial trust
- Find energy of node
- Select path

Step 1

Route discovery

Send report

Real time trust and energy measure
Find untrusted node or attack
Execute local route repair

Step 2

Steady state

Send report

Calculate trust & energy
Identify untrusted node or attacker node
Select trusted path

Step 3

Execution state

Send data

Mobile Node

Execute

Watch

Contain

Normal profile

Prevention module

Abnormal profile

46
Chapter 4

Our Proposed Approaches

- **M**: Mobile node
- **S**: Source node
- **R**: Receiver node
- **I**: Mediator node
- **S_id**: Source identification
- **R_id**: Receiver identification
- **I_id**: Intermediate identification

**Route Discovery State**

- **S** generate route packet(r, sequence no, S_id, R_id)
- **S** broadcast route packet to search **R** node

**Visit all nodes in range**

- **Is I in range and I_id \neq R_id and E_n \geq E_{th}?**
  - **No**
  - **Yes**
    - Assign t_i to I node. Forward r to next hop
    - End route discovery state(S)
    - Call steady state()

- **Is I_id == R_id and E_n \geq E_{th} and path \geq 1?**
  - **No**
  - **Yes**
    - Select shortest path. Send acknowledgement to S

**Figure 4.6: Multi Attack Scenario : Route Discovery State**
Chapter 4

Our Proposed Approaches

Steady state \( (I_{id}, P_i) \)
Examine nodes 0 to 5 seconds
All \( P_i \) active mode

Active path found?

Path inactive or nodes out of range

Yes

All \( P_i \) watch activity of neighbor
I node. Calculate trust value
\( t_j = t_i + \) (forward/receive)

Is \( t_j < t_{th} \) ?

I has normal behavior
Call execution state()

Yes

Send report to \( b_h \) module
Compare header with \( b_h \).

Is header abnormal?

Send block message to I node.
Call local route repair module.
Re-establish path

No

Calculate average \( t_j \) over all \( P_i \).

No

Update trust \( t_j \).
I has normal behavior.
Call execution state()

Figure 4.7: Multi Attack Scenario: Steady state
Chapter 4  
Our Proposed Approaches

Execution state 
$(S_{id}, R_{id}, I_{id}, P_i)$
Examine nodes for trusted route $P_i$ active mode
Calculate $t_j, E_n$ in every packet base

Is $E_n \geq E_{th}$ and $t_j \geq t_{th}$ ?

Send data by selected path $P_i$ watch I node
$E_{new} = E_{old} - E_n$ per packet
$t_j = t_{jold} \pm$ (forward/receive)

Has $P_i$ found $t_j < t_{th}$ ?

Steady State() 

Has $P_i$ found $t_j > t_{th}$ and $E_{new} < E_{th}$ ?

Local route repair () 
End execution state $(S_{id}, R_{id}, I_{id}, P_i)$

Figure 4.8: Multi Attack Scenario : Execution state
Figure 4.9: Working of Partial Forensics Engine (PFE)
Perform depth analysis of each layer in network

Obtain partial symptoms by PFE

Deploy Mobile Nodes

Perform real time monitoring of nodes using Perceptron model

Retrieve complete abnormality of nodes

Apply complete abnormality table

Block node(s)

Reliable & secure route found

Figure 4.10: Working of Forensics Deep Learning Engine (FDLE)
Figure 4.11: Working of Forensics and Neural Network
Table 4.3: Attack Log file to analyze network traffic using FDLE

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Attack Type</th>
<th>Event</th>
<th>Complete Symptoms</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User to Root</td>
<td>d</td>
<td>-NI = RTR (Network trace Level)</td>
</tr>
<tr>
<td>2</td>
<td>DoS</td>
<td>r, d</td>
<td>-Nw IFQ (20 byte increase the pkt size)</td>
</tr>
<tr>
<td>3</td>
<td>Probe</td>
<td>s</td>
<td>-Nw —</td>
</tr>
<tr>
<td>4</td>
<td>Vampire</td>
<td>r, d, s</td>
<td>-NI = RTR -Nw = LOOP, IFQ</td>
</tr>
</tbody>
</table>
Table 4.4: Attack Log file to analyze network traffic using PFE

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Attack Type</th>
<th>Event</th>
<th>Partial Symptoms</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>U2R</td>
<td>d</td>
<td>-Nl = RTR  (Network trace Level)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-Nw = LOOP   (Drop Reason )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DoS</td>
<td>r, d</td>
<td>-Nw IFQ (20 byte increase the pkt size)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-It Infected -Il 404 -Nl RTR</td>
</tr>
<tr>
<td>3</td>
<td>Probe</td>
<td>s</td>
<td>-Nw —</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-It probe -Il varying -Nl AGT</td>
</tr>
<tr>
<td>4</td>
<td>Vampire</td>
<td>r, d, s</td>
<td>-Nl = RTR -Nw = LOOP, IFQ</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-It Infected, Probe -Il 404, varying -Nl RTR, AGT</td>
</tr>
</tbody>
</table>
Figure 4.12: Single Layer Perceptron Training Network

\[ y = f(x) = \sum_{i=1}^{n} x_i \cdot w_i + b \]

- \( x_i \): count values of S, R, f, d depending on symptoms count
- w: weighted sum 0 or 1
- y: output
- b: bias (offset)
Chapter 4  Our Proposed Approaches

PFE: Partial forensics engine,
FDLE: Forensics deep learning engine
A: log file,
Atk-type: U2R, DoS, vampire, probe,
u: suspicious data,
p: partial symptoms,
c: critical analyzer,
M: mobile node,
N: network traffic,
N_d: reason of drop,
P: packet type

M generate the network traffic (N_t)

If N_t pass to PFE

Yes

if

N_t(event, N_d, P_t)  

==  

A_l(event, N_d, P_t)

No

Pass to FDLE N_t as p, N_t data as u, Pass u to c, c_l compare each field of u by A_l

Yes

c_l confirm u as attack packet

if A_l == (Atk-type)

Yes

N_t is normal
Data send to next hop or receiver

No

A_l ← A_l + u
Updated A_l synchronize for future attack detection

Stop

Figure 4.13: Forensic Based Attack Detection
\( \alpha = 0 \)

\( x_i = N_t \) (binary format)
\( y = 0 \)
\( d = (N_p, A_b) \)

If \( d \neq y \) and \( e \geq t_h \) and \( \alpha \neq \)

\( e = (d - y) \)
\( \text{wi: } \delta w_i \)
(Update weight)
\( \alpha = \alpha + 1 \)

Figure 4.14: Flow Chart of ANN Based Training
Chapter 4  

Our Proposed Approaches

Start
\( \psi = 0 \)

\( N_t \) generate by \( m \)

if \( N_t \) receives by \( N_n \)

Yes

Analyze \( N_t \) Identify symptoms

if symptoms of \( N_t \) == \( A_b \) then

Yes

\( N_t \) is Atk-type
Retrieve \( A_{t-id} \)
Block \( A_{t-id} \)
Broadcast \( A_{t-id} \) for elimination from network

No

\( N_t \) is \( N_p \) behaviour

Stop

No

Figure 4.15: Flow Chart of ANN Based Testing

58